Politecnico di Torino

Network Dynamics

Homework 3

Micol Rosini s302935

1 Influenza H1N1 2009 Pandemic in Sweden

This section will show the simulation of the pandemic of the H1N1-virus in Sweden 2009.

1.1 Epidemic on a known graph

The graph that will be used is a symmetric k-regular undirected graph with node set $\mathcal{V} = \{1, \ldots, n\}$, where every node is directly connected to the $k = 4$ nodes whose index is closest to their own modulo n. Fig. 1 shows an example with 8 nodes.

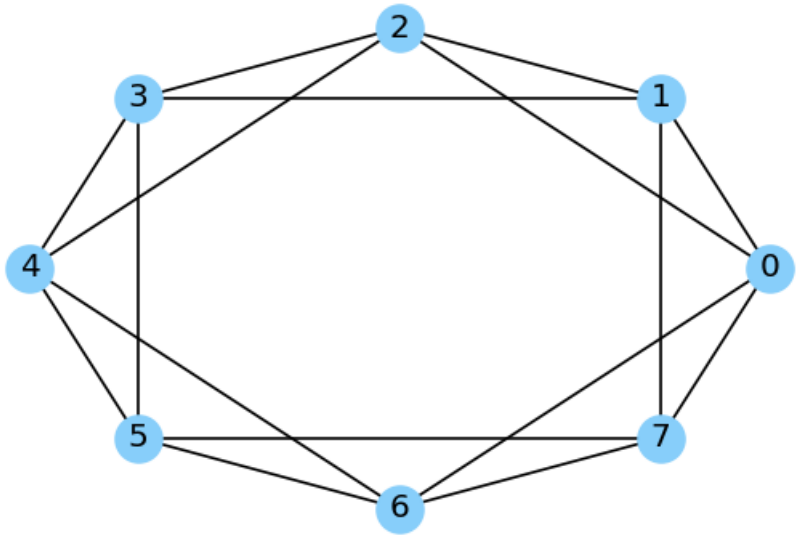

Figure 1: Symmetric 4-regular graph.

The disease propagation model used is a discrete-time simplified version of the SIR epidemic model. The probability that individual i does not get infected by any of the neighbors during a one-time step is $(1 - \beta)m$ where m is the number of infected neighbors of i, and the probability that an infected individual will recover during a one-time step is ρ.

The graph contains 500 nodes and $k = 4$. Let $\beta = 0.3$ and $\rho = 0.7$. With one week being one unit of time, the simulation lasts 15 weeks. The initial configuration has 10 infected nodes selected at random.

Fig. 2 represents the average on 100 repetitions of the simulation of the number of newly infected individuals each week, i.e, how many people become infected each week (on the average). In this case, the maximum number of new infected is reached in the *first week*, with *14* new infected people.

Instead, Fig. 3 compares the average total number of susceptible, infected, and recovered individuals each week for 100 repetitions of the simulation. The number of people infected by the epidemic is almost 25, and after week 8 people stopped to be infected: all the ones that were infected become recovered.

1.2 Preferential attachment model of random graph

This sub-section will show how to simulate a random graph according to the preferential attachment model to obtain a graph with an average degree close to k. The initial graph is complete with $k + 1$ nodes, and the probability that a new node becomes connected to an old one is proportionate to the degree of the last one. The following random graph reported in Fig. 4 has $\|\mathcal{V}\| = 1000$ nodes and $k = 8$.

1.3 Simulation of the epidemic without vaccination

The graph and the SIR epidemic model of the previous subsections will be used in order to simulate the epidemic without vaccination with a graph G of 500 nodes, and the average degree $k = 6$. With one week being one unit

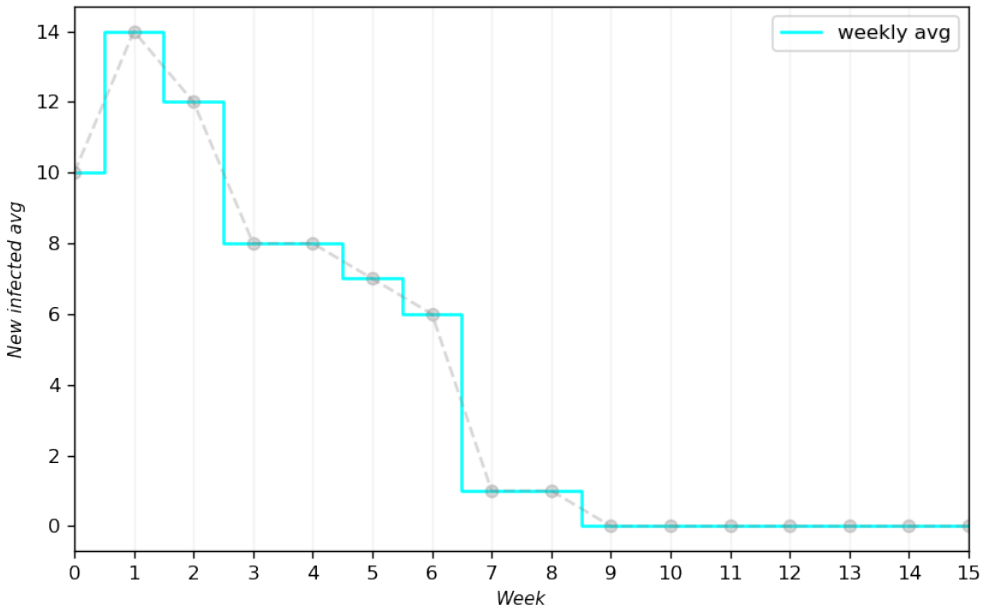

Figure 2: Weekly average new infected

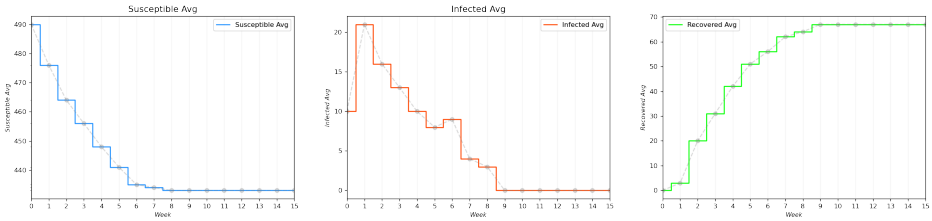

Figure 3: Weekly average of the total number of susceptible, infected, and recovered people.

of time, the simulation lasts 15 weeks. The initial configuration has 10 infected nodes selected at random.

Fig. 5 represents the average on 100 repetitions of the simulation of the number of newly infected individuals each week. In this simulation, we can see that the maximum number of new infected is reached in week 3 and it is 120.

Fig. 6 compares the average total number of susceptible, infected, and recovered individuals each week for 100 repetitions of the simulation.

The number of people infected by the epidemic is almost 150 and all the ones that were infected become **recovered** eventually.

1.4 Simulate a pandemic with vaccination

In this part, we can see how the epidemic changes if the vaccine is introduced. Therefore, during each week, some parts of the population will receive the vaccination. Once a person is vaccinated he cannot be infected. The percentage of the population that has received vaccination by each week is according to the following vector:

$$Vacc(t) = [0\%, 5\%, 15\%, 25\%, 35\%, 45\%, 55\%, 60\%, 60\%, 60\%, 60\%, 60\%, 60\%, 60\%, 60\%]$$

These individuals are selected uniformly at random from the population that has not yet received the vaccination, and we assume that regardless of the state of an individual prior to the vaccination, once vaccinated the individual will not be able to become infected nor infect any other individuals. The graph is generated using the methods of the previous section and all the parameters are equals to the Section 1.3. Fig. 7 shows the weekly average new infected.

Figure 4: Random graph with preferential attachment model.

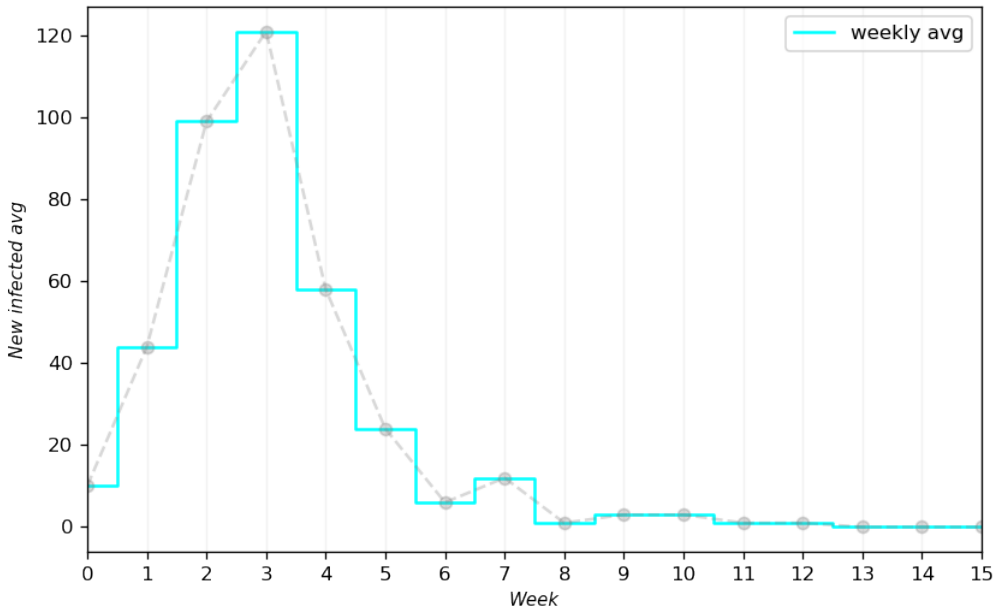

Figure 5: Weekly average new infected

In this simulation, the maximum number of new infected is reached in week 4 and it is around 60. The number is smaller compared to the result obtained in Section 1.3: the use of vaccines decreases it.

Instead, Fig. 8 shows the weekly average new vaccinated.

The comparison between the average of 100 simulations of the total number of susceptible, infected, recovered, and vaccinated individuals are shown in Fig. 9. The total number of infected is approximately half of the one in the simulation in Section 1.3.

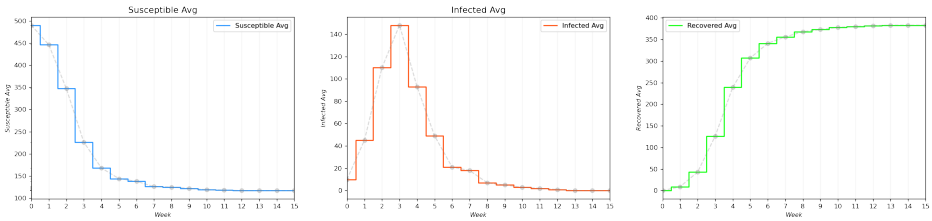

Figure 6: Weekly average of the total number of susceptible, infected, and recovered people.

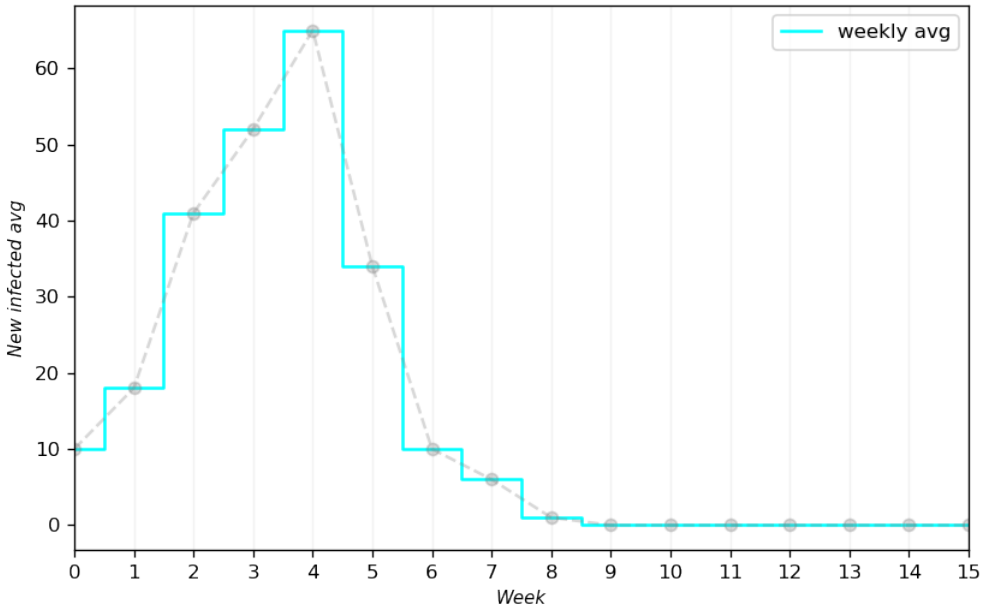

Figure 7: Weekly average new infected

1.5 The H1N1 pandemic in Sweden 2009

The task is to estimate the social structure of the Swedish population and the disease-spread parameters during the H1N1 pandemic. During the fall of 2009 about 1.5 million people out of a total population of 9 million were infected with H1N1, and about 60% of the population received the vaccination.
The percentage of the population that had received vaccination was:

$$\text{Vacc(t)} = [5\%, 9\%, 16\%, 24\%, 32\%, 40\%, 47\%, 54\%, 59\%, 60\%, 60\%, 60\%, 60\%, 60\%, 60\%, 60\%].$$

The population during the simulation will be smaller: $n = \|\mathcal{V}\| = 934$. For the scaled version, the number of newly infected individuals each week in the period between week 42, 2009, and week 5, 2010 was:

$$I_0(t) = [1, 1, 3, 5, 9, 17, 32, 32, 17, 5, 2, 1, 0, 0, 0, 0]$$

Using a gradient-based search over the parameter space of k, β, and ρ, a lot of experiments can be performed in order to find the set of parameters that best matches the real pandemic.
The general algorithm is the following: starting with an initial guess of the parameters, k_0, β_0, and ρ_0 along with some $\Delta k, \Delta\beta$, and$\Delta\rho$ for all the possible combination of the parameters in the parameter-space:

$$k \in \{k_0 - \Delta k, k_0, k_0 + \Delta k\}$$

$$\beta \in \{\beta_0 - \Delta\beta, \beta_0, \beta_0 + \Delta\beta\}$$

$$\rho \in \{\rho_0 - \Delta\rho_0, \rho_0, \rho_0 + \Delta\rho_0\}$$

replicate the same simulation of subsection 1.5. Do this N = 10 times, and compute the average number of newly infected individuals each week, $\mathbf{I(t)}$. Each time the root-mean-square error (RMSE) between the simulation and the real pandemic must be computed:

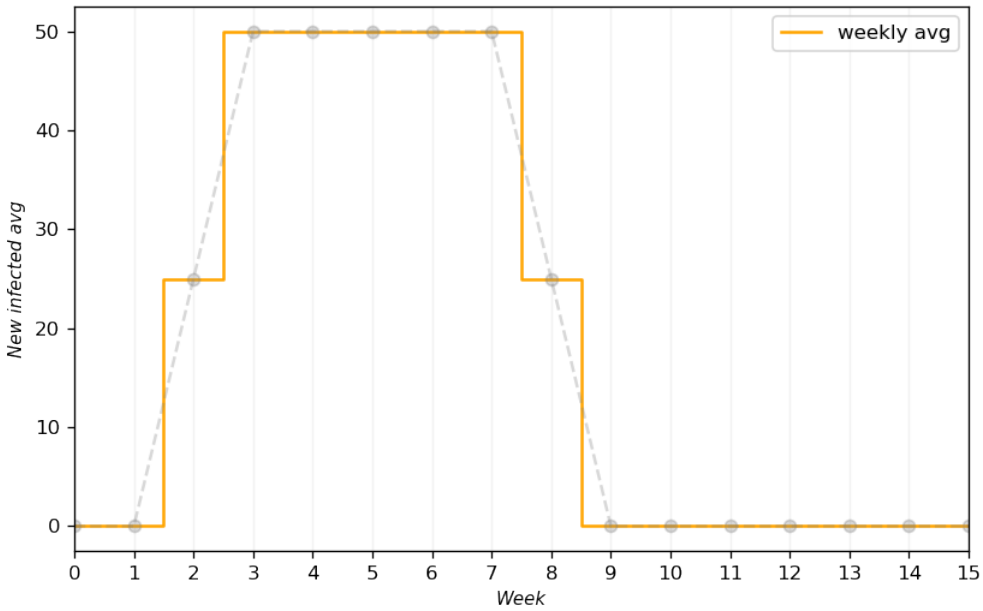

Figure 8: Weekly average new vaccinated

$$RMSE = \sqrt{\frac{1}{15}\sum_{t=1}^{15}(I(t) - I_0(t))^2}$$

where $I(t)$ is the average number of newly infected individuals each week in the simulation and $I_0(t)$ is the true value of newly infected individuals each week. k_0, β_0 and ρ_0 must be updated to the set of parameters yielding the lowest RMSE. If the result was the same set of parameters, the algorithm should stop. The can be infinite ways to update the parameters, in this work, three algorithms will be explored.

1.5.1 First algorithm

The first algorithm starts the simulation with the following values:

$$\begin{cases} k = 10 \\ \beta = 0.3 \\ \rho = 0.7 \end{cases}$$

and

$$\begin{cases} \Delta k = 1 \\ \Delta\beta = 0.1 \\ \Delta\rho = 0.1 \end{cases} .$$

During the simulation $\Delta k, \Delta\beta$ and $\Delta\rho$ are kept fixed and the only thing that changes is the centre of the interval : k_0, β_0, ρ_0. The result gives an **RMSE** $= 8.37$ with the following parameters $[k = 8, \beta = 0.2, \rho = 0.6]$.

1.5.2 Second algorithm

In the second algorithm, also the values of Δk, $\Delta\beta$ and $\Delta\rho$ change but only when the *RMSE* goes down a threshold (in the simulation it is equal to 4). This is because if the *RMSE* goes down to a good value,the right set of parameters is quite likely to be nearby: in fact, in the algorithm, $\Delta\beta$ and $\Delta\rho$ will be halved and k will become fixed. The result gives an **RMSE** $= 7.18$ with the following parameters $[k = 8, \beta = 0.1, \rho = 0.4]$.

1.5.3 Third algorithm

The last algorithm is done by changing both the centers of the intervals and the Δk, $\Delta\beta$ and $\Delta\rho$, but these last ones will change only for epoch that are odd. The set of parameter should be more various and the probability to

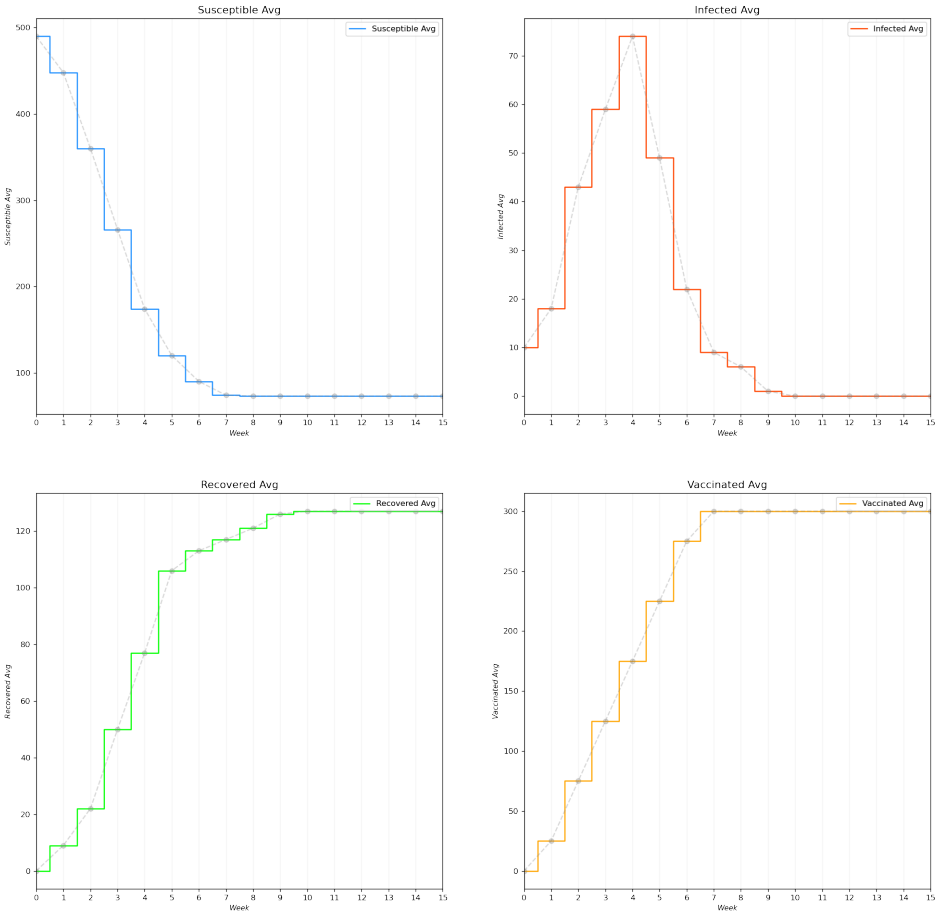

Figure 9: Weekly average of the total number of susceptible, infected, recovered, and vaccinated individuals.

find the best set of parameter should be higher. The result gives an **RMSE**= 6.4 with the following parameters $[k = 9, \beta = 0.1, \rho = 0.5]$.

1.5.4 Considerations about the algorithms

From the Fig. 10 which compares the true number of new infected and the weekly average of new infected of the simulation,it is possible to see that the third algorithm gives the best performance and the best rmse. The weekly average of the total number of susceptible, infected, recovered and vaccinated individuals of the third simulation is in the Fig. 11:

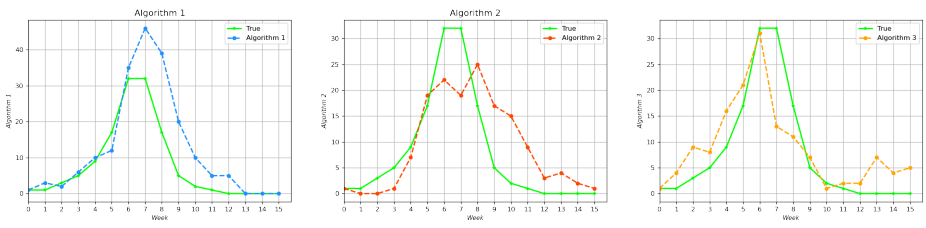

Figure 10: Comparison of the algorithms

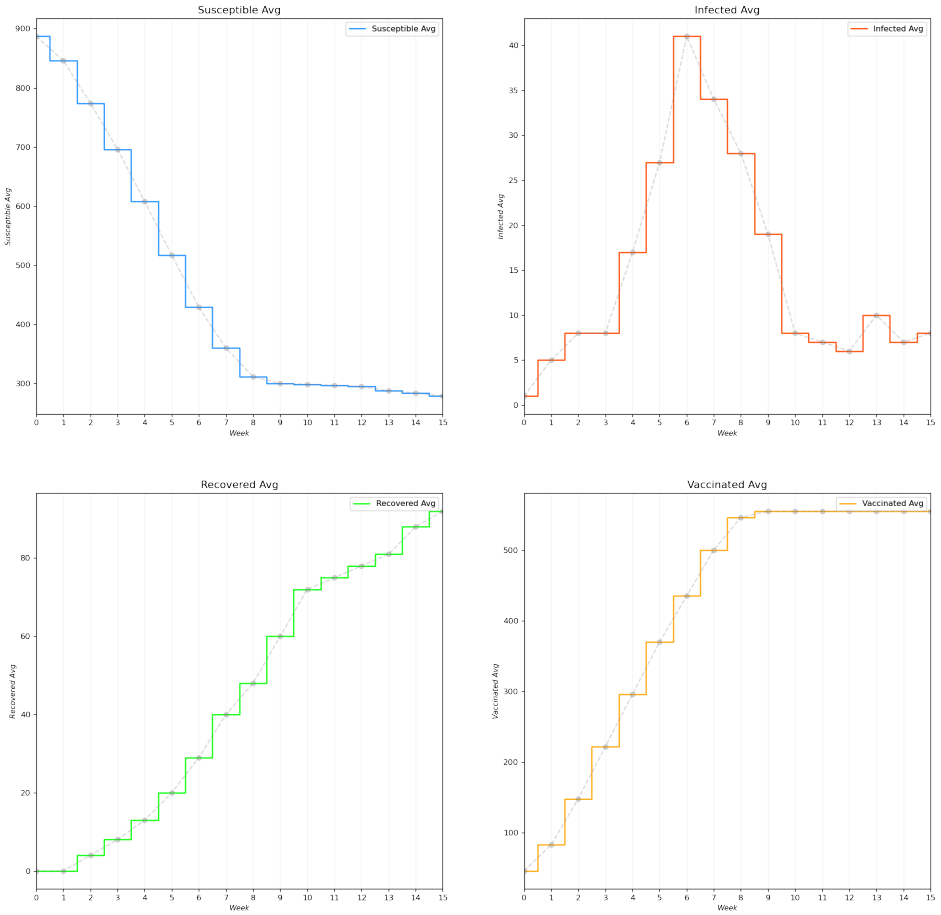

Figure 11: Weekly average of the total number of susceptible, infected,recovered, and vaccinated individuals

1.6 Different approaches

Preferential attachment model for random graphs is the best model for representing a citation network, whose nodes represent articles and links citations.
But, there are random graphs that can better represent a population during an epidemic. This is the case of the **small world graph**.
In a population, there is a huge number of connections between people, a friend of a friend could be related to many other people. In this sense, the graph that we need to represents in order to simulate a population must have really big connected components. It is quite impossible to find people that are not connected with anyone and so the graph shouldn't contain isolated nodes. Also, if two people are friends of a third person, it is quite likely that they are friends too. Keeping these considerations in mind, the small world graph is characterized by:

- small diameter (sub-linear in n): the maximum distance between any two nodes in G is small
- large clustering coefficient (i.e., large number of triangles in the graph): in terms of social networks, clustering coefficient in some sense describes how many friends of a given node are friends to each other.

In the 1960's, Stanley Milgram carried out an experiment that indicated that any two individuals in the United States were connected by a short sequence of acquaintances,i.e., the graph that better represents a population is connected with a small diameter.
The Fig. 12 represents a random graph with small world model that contains only $\|\mathcal{V}\| = 100$ nodes. Its average degree k is equals to 8

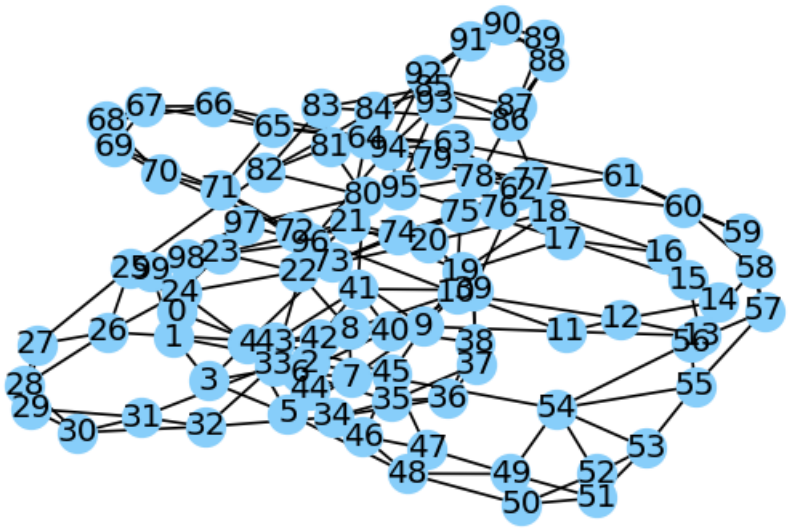

Figure 12: Random graph with small-world model and average degree = 8

The comparison between I_O and the new infected average of the simulation using the algorithm in subsection 1.5.1 and the random graph generated with a small world model is shown in Fig. 13. The best rmse obtained is 5.3 with the following values: $\{k = 12, \beta = 0.3, \rho = 0.9\}$

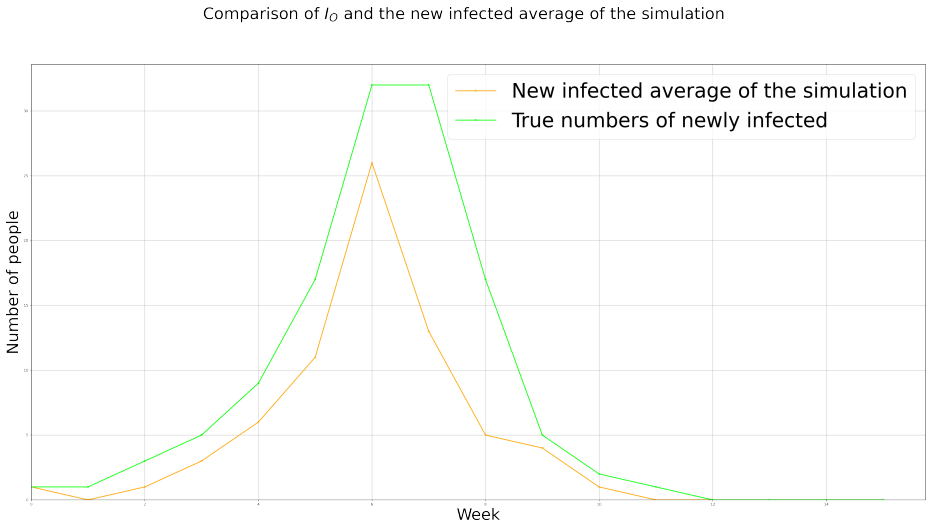

Figure 13: Comparison between the true number of infected individuals and the new infected average of the simulation

A logical other possible algorithm to find the best set of parameters is to generate randomly from the parameters-spaces, the three parameter, compare them to another set randomly chosen, and each time keeping saved the set of parameter which gives the best RMSE. This is done in the function `random_choice_sim`. The pros is that the algorithm is fast and easy, the cons are that you need to know or assume in advance what will be the parameter-space and the performance of the algorithm is random. The algorithm stops when there is no

improvement of the best rmse after 500 iteration. The parameters-spaces considered here is:

$$\begin{cases} k \in \{4, 14\} \\ \beta \in \{0, 1\} \\ \rho \in \{0, 1\} \end{cases}$$

The average degree k can be at least 4 because it is quite likely for a person to be connected with *at least* 4 people. The random graph used is the one with small world model. However, with this algorithm the best result is obtained: an **RMSE = 3.19**.The best set of parameters is: $\{k = 9, \beta = 0.2, \rho = 0.8\}$. The comparison between I_O and the new infected average of the simulation is reported in Fig. 14

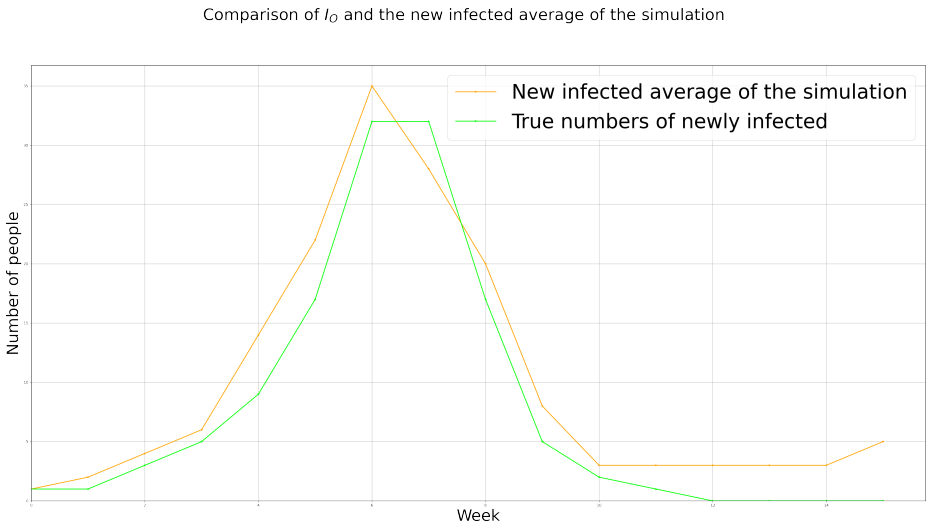

Figure 14: Comparison between the true number of infected individuals and the new infected average of the simulation

The total number of susceptible, infected, recovered and vaccinated individuals is in the plot.
In order to overcome to the cons of the previous algorithm, a more general algorithm that could be adapted for more simulation of an epidemic beside this one, is obtained using an easier version of simulated annealing technique: in the context of feature selection, simulated annealing finds the subset of features that gives the best predictive model performance amongst the many possible subset combinations. The steps of the algorithm are:

1. Given the initial centers of the interval k_0, β_0, ρ_0, consider the parameter space equal to:

$$k \in \{k_0 - \Delta k, k_0, k_0 + \Delta k\}$$

$$\beta \in \{\beta_0 - \Delta\beta, \beta_0, \beta_0 + \Delta\beta\}$$

$$\rho \in \{\rho_0 - \Delta\rho_0, \rho_0, \rho_0 + \Delta\rho_0\}$$

2. Generate randomly 27 different set of the three parameter from the previous parameter space and do it for each different $\Delta k, \Delta\beta, \Delta\rho$:

$$\begin{cases} \Delta k = [1, 1, 2, 3] \\ \Delta\beta = [0.025, 0.05, 0.1, 0.2] \\ \Delta\rho = [0.025, 0.05, 0.1, 0.2] \end{cases}$$

3. Compute the best (local) rmse between all these different parameter spaces, also, set the *global* best rmse equals to this best *local* rmse. Repeat the algorithm using as centres of the intervals the values that gave the best rmse.

4. If the values which gives the new local best rmse are the same of the global best rmse's values,stop the algorithm. If, values are different and the new local best rmse is better than the global one, change the center of the interval and repeat the previous steps. If it is not better, set a high given probability of accepting to change the centre of the interval with the values that gives a worst performance and explore different subsets

5. gradually reduce the probability of keeping poor-performing subsets so that high-performing subsets are increasingly likely to be kept. This phase is considered 'exploitation'.

The probability used is:

$$p = e^{-(\text{LOCAL BEST RMSE}-\text{GLOBAL BEST RMSE})/T}$$

where T is definde as:

$$T = T_0 \cdot \alpha^{\text{ITERATION}}$$

In the simulation $T_0 = 5$ and $\alpha = 0.95$
In this way the probability of accepting a worst performance is both connected to the difference between the two performances and the number of iterations done by the algorithm, since, as the number of iterations grows the algorithm will become more precise and it will become quite hard to improve the best rmse.
The algorithm gives a **best rmse = 5.27** with values $k = 12\beta = 0.24\rho = 0.83$. Even if the performance is worst than the one using `random_choice_sim`, the algorithm is more general, faster, and it is not necessary to know in advance the possible set of parameters. The comparison between the true quantity of new infected people for each week and the new infected average of the simulation is reported in Fig. 15
The total number of susceptible, infected, recovered and vaccinated individuals is in the Fig. 16

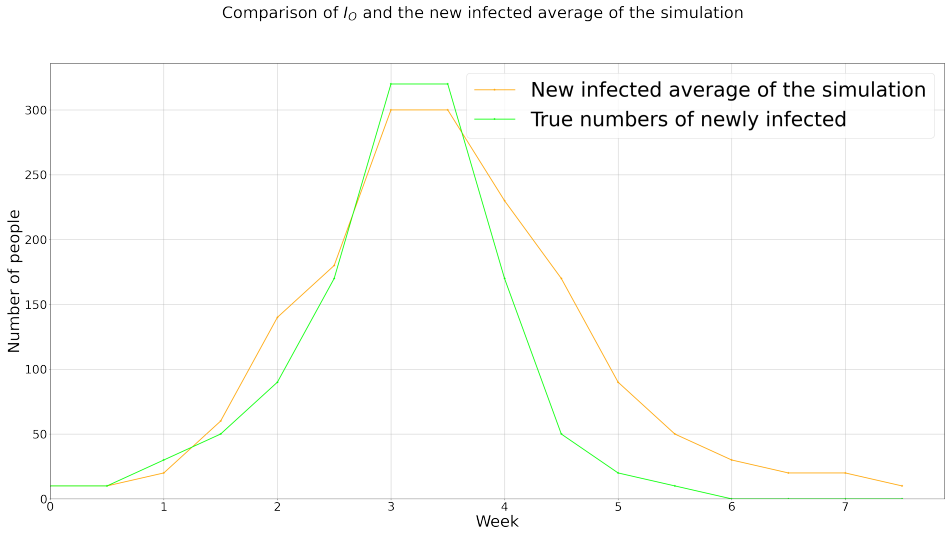

Figure 15: Comparison between the true number of infected individuals and the new infected average of the simulation

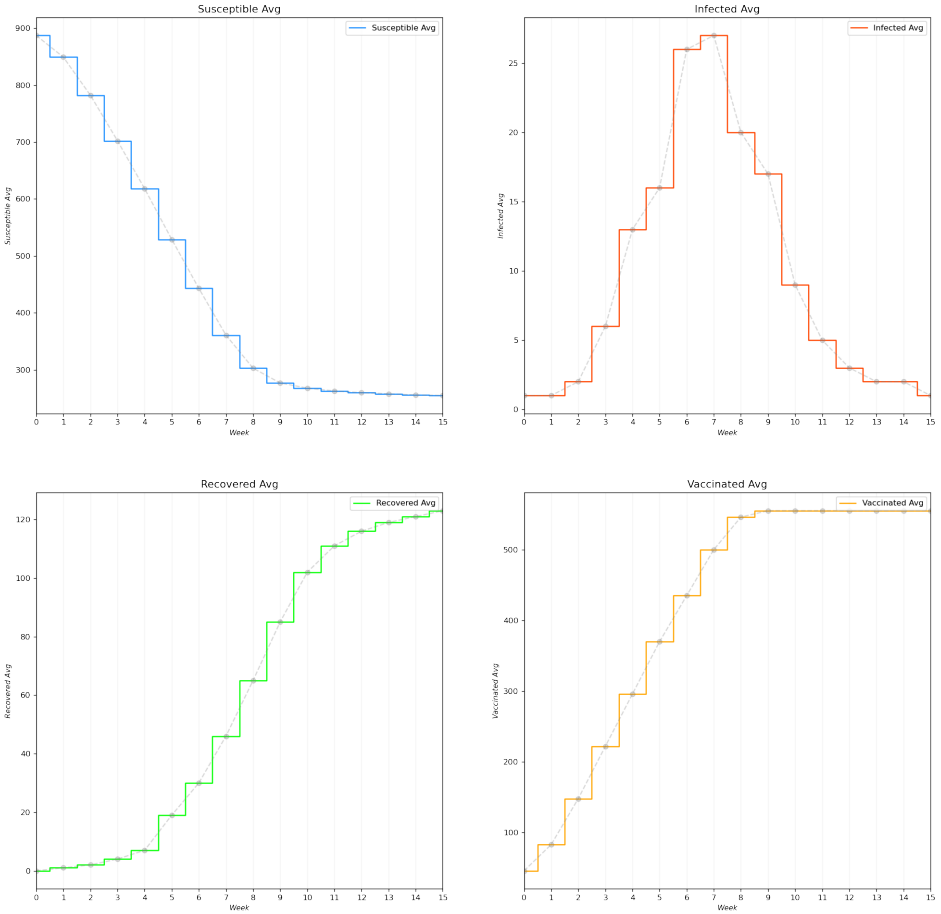

Figure 16: Weekly average of the total number of susceptible, infected,recovered, and vaccinated individuals with the new algorithm

2 Coloring

The graph coloring can be applied for distributed learning techniques in potential games.

2.1 Distributed learning for a line graph

The graph used here is a line graph with 10 nodes, represented in Fig. 17

Figure 17: Line graph of the exercise

The set of possible states is $C = \{red, green\}$. At initialization, each node is red. Every discrete time instance t, one node $I(t)$, chosen uniformly at random, wakes up and updates its color. The new color (resulting from a node's update), is chosen from a probability distribution given by:

$$P(X_i(t+1) = a | X(t), I(t) = i) = \frac{e^{-\eta(T)\sum_j W_{ij} c(a, X_j(t))}}{\sum_{s \in C} e^{-\eta(t)\sum_j W_{ij} c(s, X_j(t))}}$$

where $\eta(t) = \frac{t}{100}$ and the cost is given by:

$$c(s, X_j(t)) = \begin{cases} 1 & \text{if } X_j(t) = s \\ 0 & \text{otherwise} \end{cases}$$

The potential function is:

$$U(t) = \frac{1}{2} \sum_{i,j \in \mathcal{V}} c(X_i(t), X_j(t))$$

Using the following algorithm a Nash equilibrium is found, and the potential function is equal to U = 0. The Fig. 18 represents the learning dynamics that reaches a Nash equilibrium only in 7 time-step.

2.2 Assigning WiFi-channels to routers

Here, the same learning dynamic is used to assign WiFi-channels to routers: routers a link between two nodes means that the two routers are able to interfere with each other. The cost function is equal to:

$$c(s, X_j(t)) = \begin{cases} 2 & \text{if } X_j(t) = s \\ 1 & \text{if } |X_j(t) - s| = 1 \\ 0 & \text{otherwise} \end{cases}$$

and the set of possible colors

$$C = \{1 : red, 2 : green, 3 : blue, 4 : yellow, 5 : magenta, 6 : cyan, 7 : white, 8 : black\}$$

The learning dynamics is the same of the previous subsection 2.1

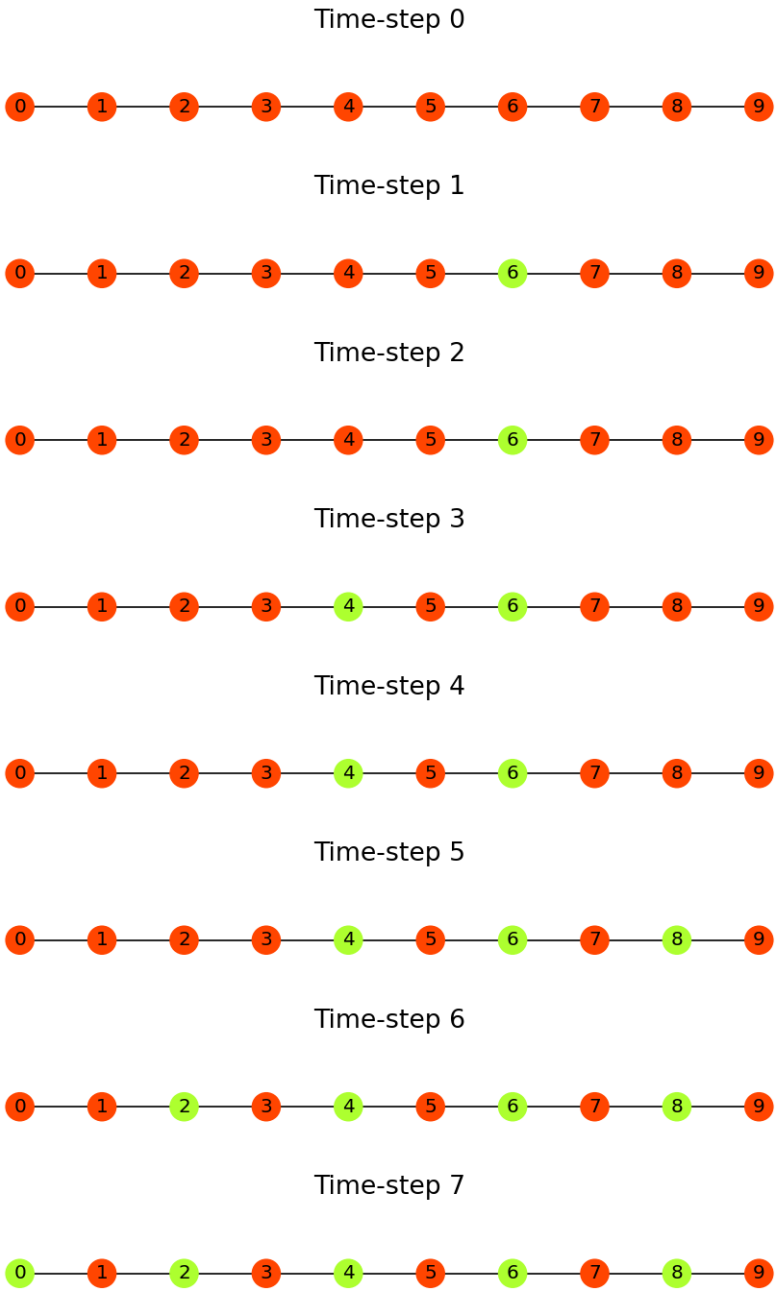

Figure 18: Line graph of the exercise

The Fig. 19 represents the network of routers: each node represents a router and links between two nodes means that the two routers are able to interfere with each other.
Here, after 1000 iterations, the best potential function found is: $U = 4$. The Fig. 20 plot the potential function

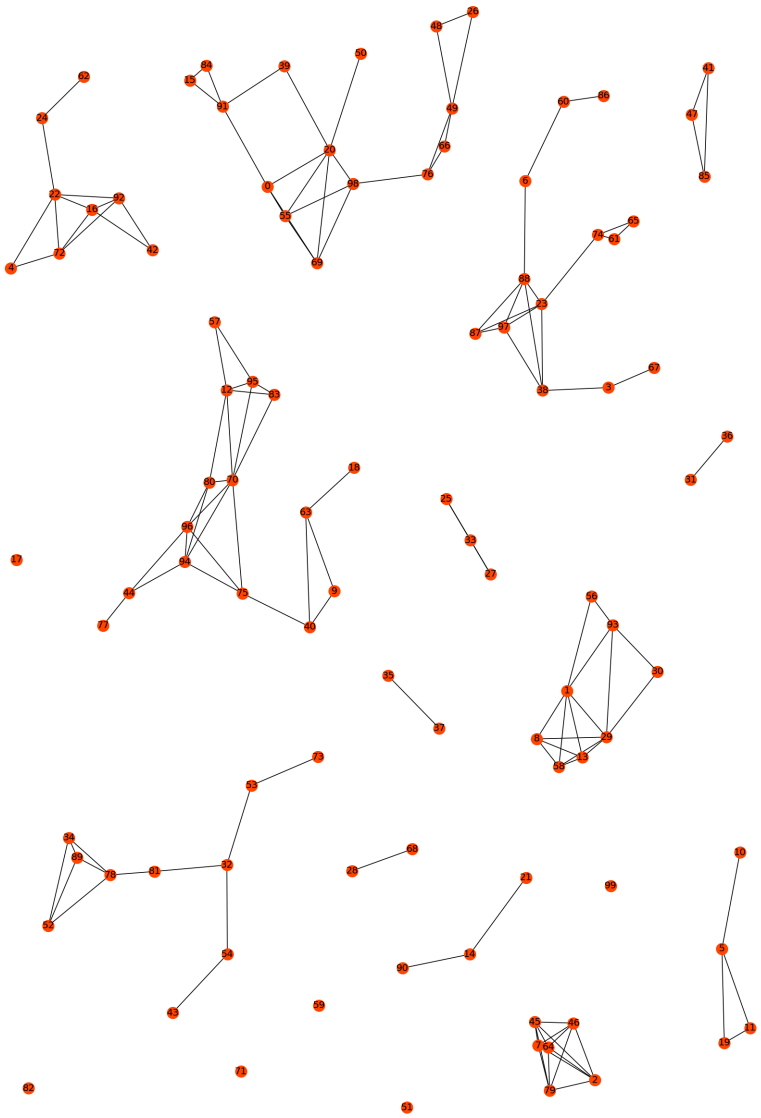

Figure 19: Network of 100 routers

U for different tested cases, usually after 500 iterations, it reaches its best value 4.
The Fig. 21 represents the coloring of the routers' graph which corresponds to the best potential function.

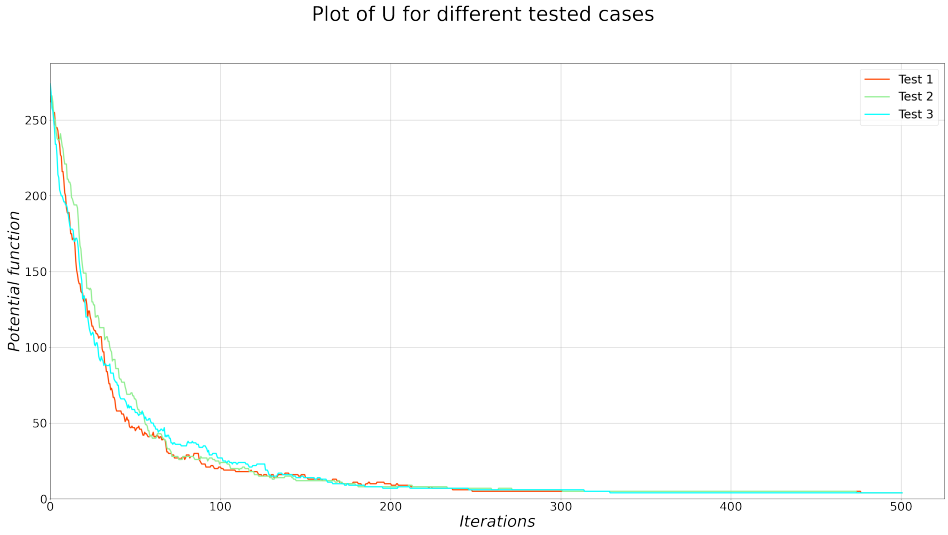

Figure 20: Plot of U for different tested cases

2.3 Exploration for different values of η

The probability

$$P(X_i(t+1) = a | X(t), I(t) = i) = \frac{e^{-\eta(T)\sum_j W_{ij} c(a, X_j(t))}}{\sum_{s \in C} e^{-\eta(t)\sum_j W_{ij} c(s, X_j(t))}}$$

is a clear example of **softmax function**: an activation function that scales numbers into probabilities

$$S(y)_i = \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}}$$

when y is a big negative value the probability obtained is a very small value close to 0, instead if y is a negative value really close to 0, the probability will be close to 1. Let

$$y = -\eta(T) \sum_j W_{ij} c(a, X_j(t))$$

when $\eta = t/100$ the probability that a node is assigned to a frequency band equal to or close to the one of its neighbors decreases with the iterations: infact, the absolute value of y will increase and since y is negative, the probability will approach 0.
Let's explore the other values of η:

- $\eta = 0.01$: the probability of assigning a frequency band equal to or close to that of its neighbors will be really high and close to 1, and it will be impossible to obtain a near-zero potential function, indeed after 100 iteration the potential function is $U = 67$.
- $\eta = 100$: y has a high negative value, and the probability will be really small. In this way, an optimal potential function is found: $U = 4$.
- $\eta = e^{t/1000}$: the absolute value of y will increase with iterations and the probability will decrease. However, the maximum value that η can assume in this way is 2.71 and it is still a little value, for this reason a good potential function but not too close too zero is found: $U = 7$.
- $\eta = \log(t+2)$: also in this case the absolute value of y will increase with iterations and the probability will decrease,the maximum value that η can assume in this way is almost 7. In fact, a near-zero potential function is found: $U = 4$
- $\eta = t^2$: in this way the $|y|$ value will increase really fast and the probability will decrease, the best potential function is found at the first 100 epoch, the fastest of all tests.

To summarize, values of η that increase with time, will generate a better potential function. The more η grows with time, the best performance is obtained. Values of η that are small will not allow to find a good potential function.

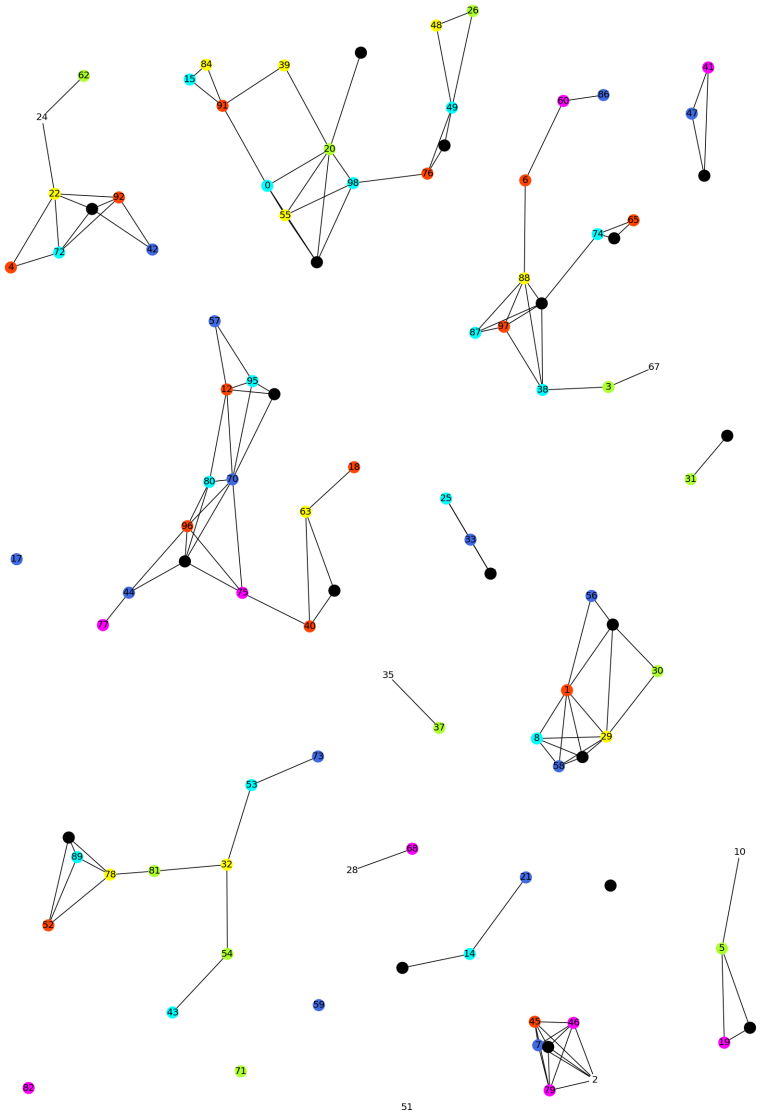

Figure 21: 'Simulation of coloring algorithm in the routersǵraph

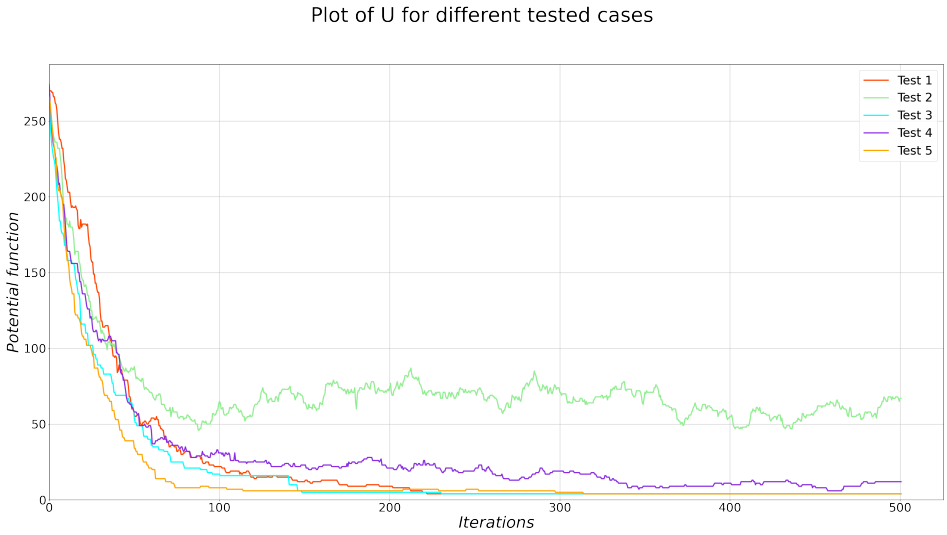

Figure 22: Comparison for different values of η