# Network Dynamics and Learning Homework 3

Giovanni Sciortino, Giuseppe Suriano

s302959@studenti.polito.it, s296605@studenti.polito.it

March 10, 2023

**The code of the homework can be found in the following link here**

## 1 Exercise 1: Influenza H1N1 2009 Pandemic in Sweden

### 1.1 Preliminary parts

In this first part of this exercise we will simulate a pandemic on a known graph and we will generate a random graph with preferential attachment.

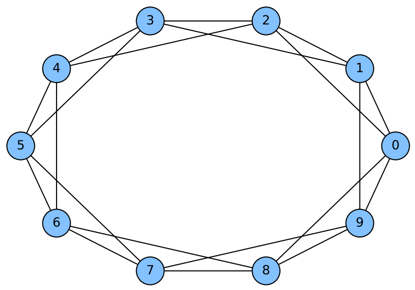

Figure 1: Symmetric k-regular graph with $n = 10$ nodes

### 1.1.1 Epidemic on a known graph

In this part we simulate an epidemic on a Symmetric k-regular graph of 500 nodes where every node is directly connected to the k = 4 nodes whose index is closest to their own modulo n. Fig (1) represents an example of a Symmetric k-regular graph.

To simulate the epidemic, we used a discrete-time simplified version of the SIR epidemic model. At any time $t$, the nodes are in state $X_i(t) \in \{\mathrm{S}, \mathrm{I}, \mathrm{R}\}$, where S is susceptible, I is infected and R is recovered. The probability of an infection being transmitted from an infected individual to a susceptible individual (connected by a link) in one time step is represented by $\beta \in [0, 1]$. If a susceptible individual $i$ has $m$ infected neighbors, then the probability that individual $i$ is not infected by any of their neighbors in one time step is $(1 - \beta)^m$.

Therefore, the probability that individual $i$ becomes infected by one of their neighbors in one time step is:

$$P\left(X_i(t+1) = I \mid X_i(t) = S, \sum_{j \in \mathcal{V}} W_{ij} \delta^I_{X_j(t)} = m\right) = 1 - (1 - \beta)^m \tag{1}$$

Additionally we define $\rho \in [0, 1]$ which represents the probability that an infected individual will recover in one time step, more formally:

$$P\left(X_i(t+1) = \mathrm{R} \mid X_i(t) = \mathrm{I}\right) = \rho \tag{2}$$

Note that in both equations (1) and (2) $\sum_{j \in \mathcal{V}} W_{ij} \delta^I_{X_j(t)}$ is the number of infected neighbors for node $i$.

Given these premises we simulated an epidemic, given $\beta = 0.3$ and $\rho = 0.7$, for 15 time unit and 10 infected nodes selected randomly for the initial configuration. So at each time unit we checked the state of each node of the graph and change the state of it according to the probability described above. More in particular, at each time-step:

- a susceptible individual can become infected or remain susceptible.
- an infected individual can become recovered or remain infected.
- a recovered individual remain recovered.

By simulating the pandemic for $N = 100$ times we achieved results showed in Fig. (2) and Fig. (3).

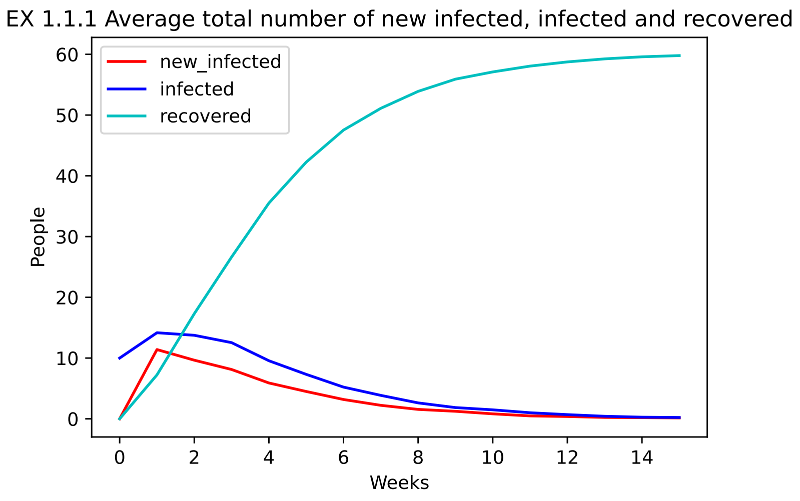

Figure 2: The average number of newly infected, total infected and recovered individuals each week

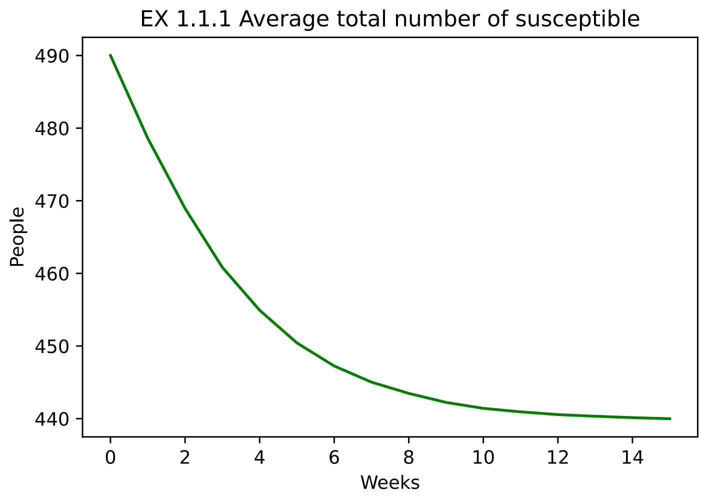

Figure 3: The average total number of susceptible individuals each week

### 1.1.2 Generate a random graph

In this part we had to generate a random graph according to the *preferential attachment model*. Starting from a complete graph with $k + 1$ nodes we add one node at each step connecting it to the graph according to some stochastic rules. In particular, at each step $t$ the degree of the new added node is:

$$w(t) = k/2 = c$$

After that, we should decide where to add the links proportionally to the current degree of each node:

$$P\left(W_{n_t i} = W_i n_t = 1\right) = \frac{w_i(t-1)}{\sum_{j \in \mathcal{V}_{t-1}} w_j(t-1)}$$

where $W(t)$ is the adjacency matrix for the next time-step t, $n_t$ is the new node at step $t$ and $w_i(t-1)$ is the degree of node i prior to adding the new node.

In order to generate a proper random graph such that the average degree will be $k$, we handled the case in which k is odd by simply alternating between adding $\lfloor k/2 \rfloor$ and $\lceil k/2 \rceil$ links when adding a new node to the graph.

## 1.2 Simulate a pandemic without vaccination

In this part of the homework we deployed a random graph generated with *preferential attachment* according to Section (1.1.2) and then simulate an epidemic on it. The disease propagation model is again the discrete-time version of the SIR epidemic model used in Section (1.1.1).

So we generated a random graph according to *preferential attachment* with 500 nodes and average degree $k = 6$. Then with the following parameters we simulated the epidemic for 100 times:

- $\beta = 0.3$
- $\rho = 0.7$
- 15 unit of times (weeks)
- 10 random infected nodes as initial configuration

The results in this section are showed in Fig. (4) and Fig. (5).

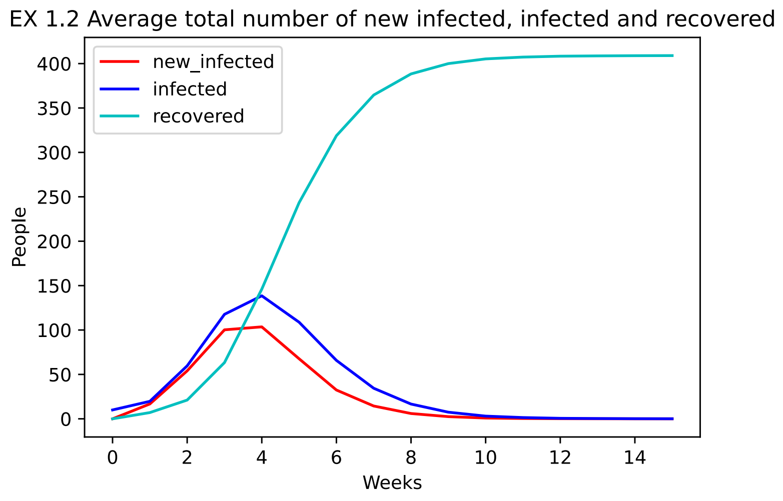

Figure 4: The average number of newly infected, total infected and recovered individuals each week

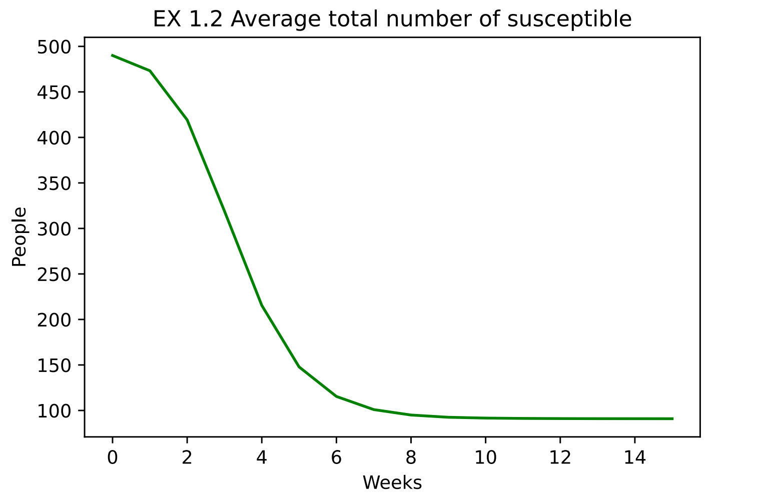

Figure 5: The average total number of susceptible individuals each week

## 1.3 Simulate a pandemic with vaccination

In order to slow down the epidemic, in this session we had to introduce a vaccination campaign. So basically we had to simulate a pandemic with vaccination according to the previously developed SIR discrete model in Section (1.1.1).

In order to distribute the vaccination over the population, the following vector that represents the total fraction of population that has received vaccination by each week is provided by the trace:

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

This array should be interpreted as: 55% of the population has received vaccination by week 7 , and 5% received vaccination during week 7 .

Given these premises we simulated an epidemic with vaccination on a random graph, generated according to Section (1.1.2) with 500 nodes and average degree $k = 6$, for 100 times with the following parameters:

- $\beta = 0.3$
- $\rho = 0.7$
- 15 unit of times (weeks)
- 10 random infected nodes as initial configuration

Starting from the considerations made in Section (1.1.1), at the beginning of each time-step the first action we perform is the vaccination since we presume that the vaccine take effect immediately. So in particular, we select the individuals to vaccinate uniformly at random from the population that has not yet received vaccination, even infected individuals can receive vaccine. Clearly, once vaccinated, the agent will not be able to become infected nor infect any other individuals and remain in the 'vaccinated' state until the end of the simulation.

The results in this section are showed in Fig. (6) and Fig. (7).

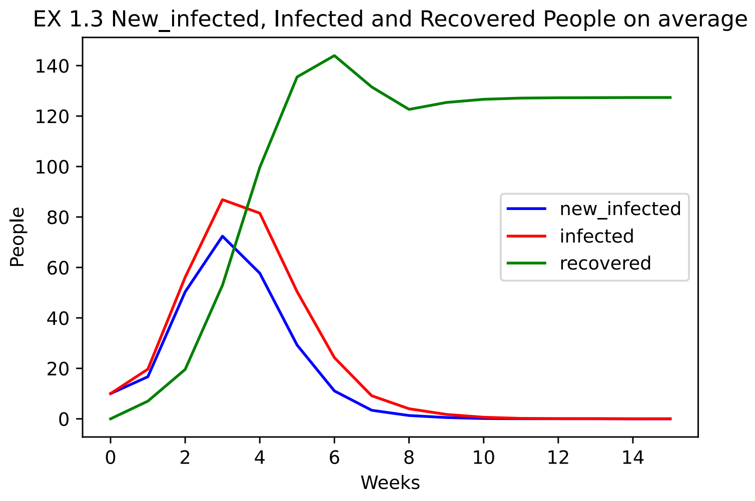

Figure 6: The average number of newly infected, total infected and recovered individuals each week

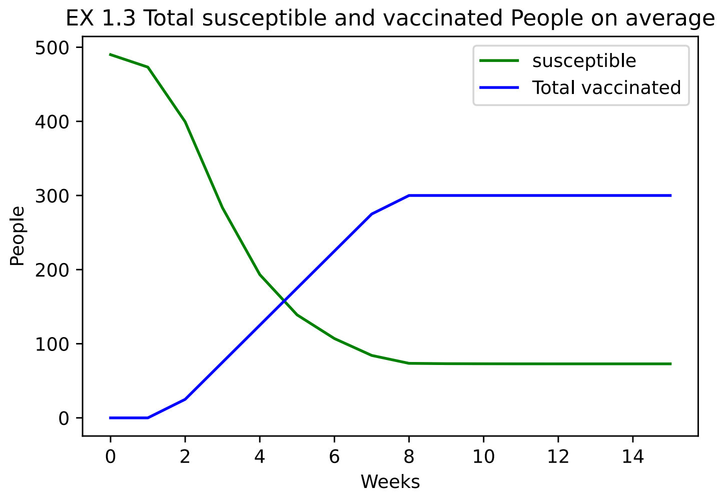

Figure 7: The average total number of susceptible and vaccinated individuals each week

## 1.4 The H1N1 pandemic in Sweden 2009

In this exercise, we were required to estimate the social structure of the Swedish population and the disease-spread parameters during the H1N1 pandemic of 2009. In order to not spend too much time running simulations, we will scale down the population of Sweden by a factor of $10^4$; in fact we start from a graph of 934 nodes.

Since we are dealing with a pandemic with vaccination, the vector $\text{Vacc}(t)$ representing the fraction of population that has received vaccination is provided.

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

This vector is perfectly suitable for the procedure implemented in Section (1.3), allowing us to simulate the epidemic properly.

Moreover, the following vector $I_0(t)$, that is the real evolution of newly infected people, is given to evaluate our prediction:

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

To predict the new infected people during the pandemic, we develop an optimization algorithm to minimize the root meas square error,

$$RMSE = \sqrt{\frac{1}{15}\sum_{i=1}^{15} \left(I(t) - I_o(t)\right)^2}$$

where $I(t)$ is the predicted vector.

We set the initial number of infected agents $I(0)$ equals to 1 . At each step we generate a parameters set with all the triplets obtained by the combination of the following three vectors:

$$\{k_o - \Delta k, k_0, k_0 + \Delta k\}, \{\beta_o - \Delta\beta, \beta_0, \beta_0 + \Delta\beta\}, \{\rho_o - \Delta\rho, \rho_0, \rho_o + \Delta\rho\}$$

For each triplet we generate a random graph with $preferential\ attachment$ and we simulate the pandemic 10 times. At the end we compute the average of the new infected agents and the RMSE. Once we have done with all configurations we pick the lowest RMSE and the parameters associated, from this triplet of parameters we generate the combination for the next iteration. If the new parameters found are the same as the previous iteration we stop the algorithm.

Starting from the following parameters:

- $k_0 = 10$, $\beta_0 = 0.3$ and $\rho_0 = 0.6$
- $\Delta k = 1$, $\Delta\beta = 0.1$ and $\Delta\rho = 0.1$

the best set of parameters and the lowest average $RMSE$ for that iteration simulated 10 times found are:

- $k_0 = 9$, $\beta_0 = 0.2$ and $\rho_0 = 0.6$.
- $E\left[RMSE_{\text{best}}\right] \approx 5.25$

The results computed in this section are showed in Fig.8 and Fig.9.

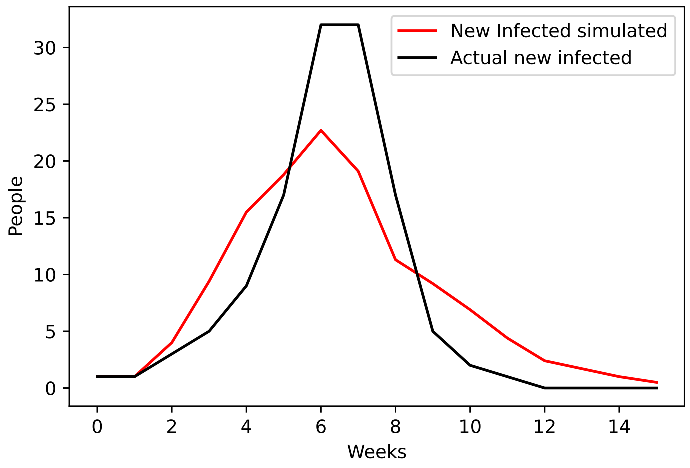

Figure 8: The average number of newly infected individuals each week according to the model (with our best parameters) compared to the true value of newly infected individuals each week.

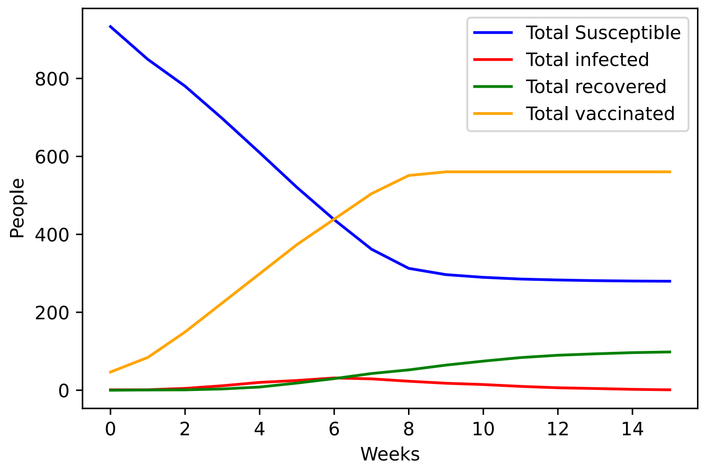

Figure 9: The total number of susceptible, infected, recovered and vaccinated individuals at each week according to the model.

## 1.5 Challenge (optional)

The last task required to find a better way of both creating the random graph and find the best parameters to model Sweden's H1N1 epidemics' diffusion described in Section (1.4). Different improvements both in graph creation and in best parameters finding have been tested and, in the following, they will be discussed one at a time.

### 1.5.1 Graph creation

Concerning random graph creation algorithms, other two algorithm have been tested: *Small world* and *configuration model*.

Empirical observations suggest that real world graphs have *small diameter* and *high clustering*. That's why Small World models for random graphs have been introduced. The main idea is to start with a simple graph where every node is connected to $k$ closest nodes modulo $n$. Then, add to the existing graph $l$ additional undirected links, where:

$$I \approx \text{Bin}\left(\frac{nk}{2}, p\right)$$

and the new links connect pairs of nodes chosen independently and uniformly at random. In this way it's possible to combine models for geographical proximity and some rare long distance.

For what concerns the configuration model random graph we constructed it by giving to the networkx library function *nx.configuration_model* a sequence of degree according to a poisson distribution with $\lambda = k$. Once the graph is created, we manually remove the self loop, since the function of the library create a graph with self-loops that for our simulation task is conceptually wrong.

### 1.5.2 Algorithm improvement

Starting from the algorithm described in Section (1.4), we changed the stopping condition. In fact, when the new parameters found are the same as the previous iteration now we divide by 2 the values of $\Delta\beta$ and $\Delta\rho$ while $\Delta k$ is set to 0 from which we generate the combination of parameters for the next iteration. The new stopping condition is that $\Delta\beta$ and $\Delta\rho$ get lower than a certain threshold, that we fixed to 0.01.

### 1.5.3 Test performed and comparison of results

We performed the simulations with *preferential attachment*, *small-world*(with $p$=0.4) and *configuration model* as random graph, each one combined with the algorithm described before in the subsection (1.5.2). The best result found with the respective RMSE found are displayed in the Table (1). The plot of the average new infected individuals per week simulated with the three configuration according to Table (1) configurations is showed in Figure (10).

| configuration | $\beta$ | $\rho$ | $k$ | $E[RMSE_{\text{best}}]$ |
|---|---|---|---|---|
| *Preferential attachment* | 0.400 | 0.400 | 6 | 3.47 |
| *Small world* | 0.325 | 0.575 | 10 | 4.06 |
| *Configuration model* | 0.650 | 0.850 | 5 | 4.42 |

Table 1: Comparison of parameters and result in terms of RMSE between all the three random graph discussed in this exercise. Notice that all the simulations are performed with the algorithm improved described in (1.5.2).

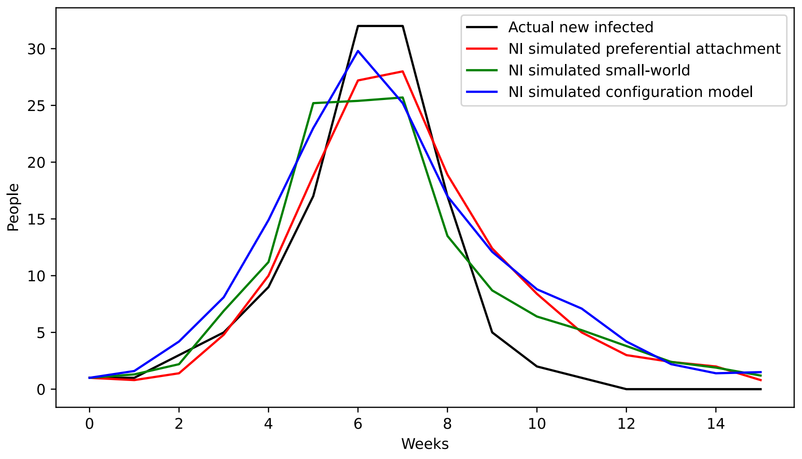

Figure 10: Average new infected individuals per week comparison between the simulation performed with *Preferential attachment*, *Small world* and *Configuration model*. Notice that all the simulations are performed with the algorithm improved described in (1.5.2).

## 2 Exercise 2: Coloring

In this part, we will study graph coloring as an application of distributed learning in potential games. The aim of graph coloring is to assign a color to each node in a given undirected graph, such that none of the neighbors of a node have the same color as that node.

### 2.1 Point a

In this section we study a line graph with 10 nodes. The possible state of the nodes are two: red and green. After initializing each of them to *red* state, every discrete time instant $t$, one node is chosen uniformly at random and updates its color according to the following probability distribution:

$$P\left(X_i(t+1)=a \mid X(t), I(t)=i\right)=\frac{e^{-\eta(t)\sum_j W_{ij}c(a,X_j(t))}}{\sum_{s\in\mathcal{C}} e^{-\eta(t)\sum_j W_{ij}c(s,X_j(t))}} \tag{3}$$

where $X_i(t)$ is the state of the node at the time $t$ and the cost is given by:

$$c\left(s, X_j(t)\right)=\begin{cases}1 & \text{if } X_j(t)=s\\ 0 & \text{otherwise.}\end{cases} \tag{4}$$

Moreover, in the expression (3), $\eta(t)$ is the inverse of the noise. As suggested in the Homework trace we used:

$$\eta(t)=\frac{t}{100}$$

Then to study how close to a solution the learning algorithm is, we consider the following potential function:

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

where $\mathcal{V}$ is the set of nodes. Clearly if the potential is zero, there are no conflicting nodes and a solution is found. Simulating the learning dynamics described above potential reached zero after 195 iterations as shown in Fig. (11).

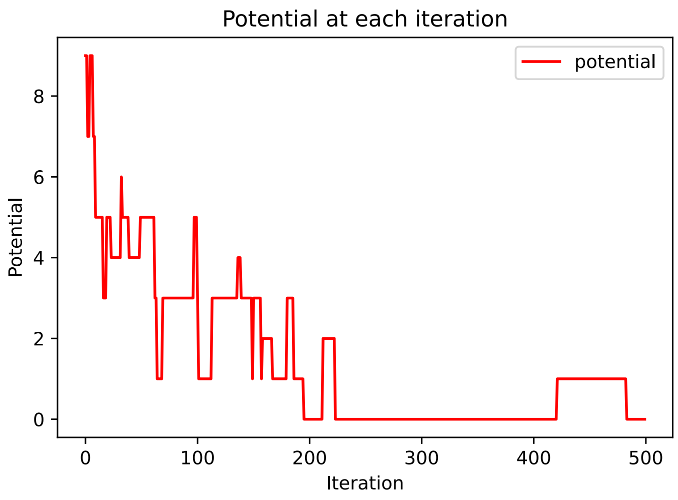

Figure 11: Plot of the potential function at each iteration Point a.

## 2.2 Point b

Using the adjacency matrix provided in the file 'wifi.mat' we create the graph on which we have to simulate the learning dynamics in this second part of this exercise, where each node represent a rooter. In this case starting from the premises in the point before (2.1), in this case the set of possible states is $\mathcal{C} = \{1 : \text{red}, 2 : \text{green}, 3 : \text{ blue}, 4 : \text{yellow} \ , 5 : \text{magenta}, 6 : \text{cyan} \ , 7 : \text{white}, 8 : \text{black} \ \}$,where colors represent frequency band, and the cost function is:

$$c\left(s, X_j(t)\right) = \begin{cases} 2 & \text{if } X_j(t) = s \\ 1 & \text{if } |X_j(t) - s| = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

So using $\eta(t) = \frac{t}{100}$ a near zero solution is found after 611 iteration as shown in Fig. (12). Moreover an illustration of the node coloring corresponding to the smallest potential is displayed in Fig. (13). The lowest potential reached is 4.0 since two isolated component of the graph can't reach the zero conflict state since they are composed of a K5 and K6 complete graph respectively.

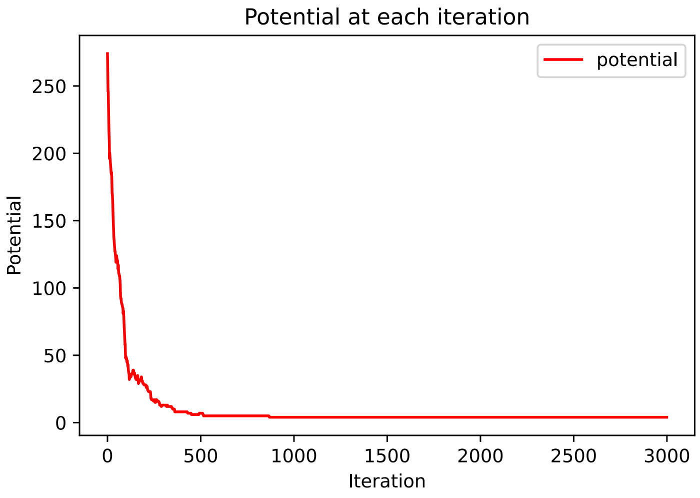

Figure 12: Plot of the potential function at each iteration Point b.

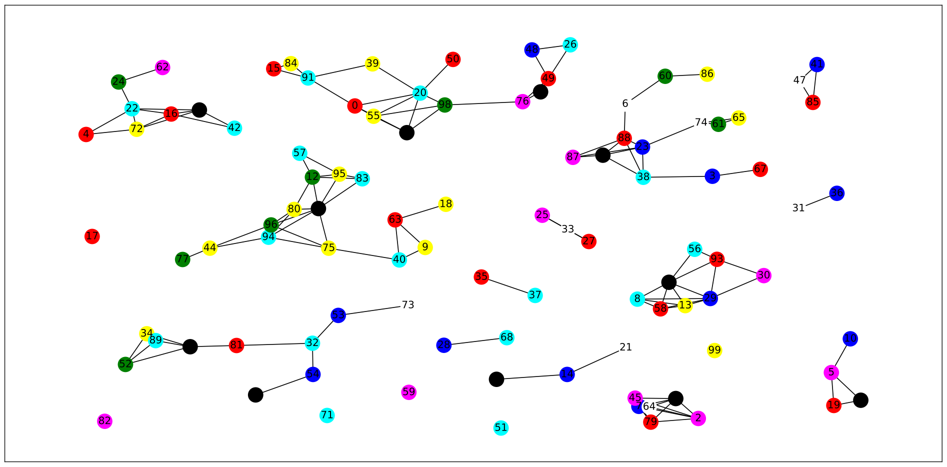

Figure 13: Illustration of the node coloring corresponding to the smallest potential reached in Point b.

## 2.3 Point c: optional

In this final part of this homework we have to evaluate what happens for different choice of $\eta(t)$. In order to give an interpretation to the expression (3) we can say that the probability with which a new action (change of color in our case) is adopted by a node is decreasing as a cost like in expression (4) or (5) increases; for $\eta \to 0$ the dependence on the cost vanishes and (3) converges to a uniform probability distribution on the action that a node can perform, in our case along the possible colors. In fact for example if we put $\eta$=0 or $\eta$=0.1, we can notice that in either way the system doesn't converge since the noise introduced is very high, basically during the entire simulation the nodes change state randomly. But clearly, we can see an improvement in the first iterations since the nodes are all initialized with the same color, in fact every color chosen correspond to a decrease of the potential. (See Figures 14, 15 and 16)

On the other hand, as $\eta \to +\infty$, 3 converges to a probability distribution that remains inversely proportional to the conflicts caused from a node in taking a certain action. Since the noise introduced become less relevant as the function grows, more $\eta$ is fast-growing and sooner the system converges to the minimum potential. (See Figures 17 and 18)

For the function like $\eta = t^2/100$ or $\eta = t^4$ there are approximation problems that comes in python, since over a certain value the computation of the the negative exponential is approximated to 0.0, making the trend of the simulation inconsistent with the theoretical trend, that should lead to the lowest potential in less iterations with respect to previously mentioned simulations (with slower growing functions). The plot of potential function in these cases are showed in Figure (19) and (20).

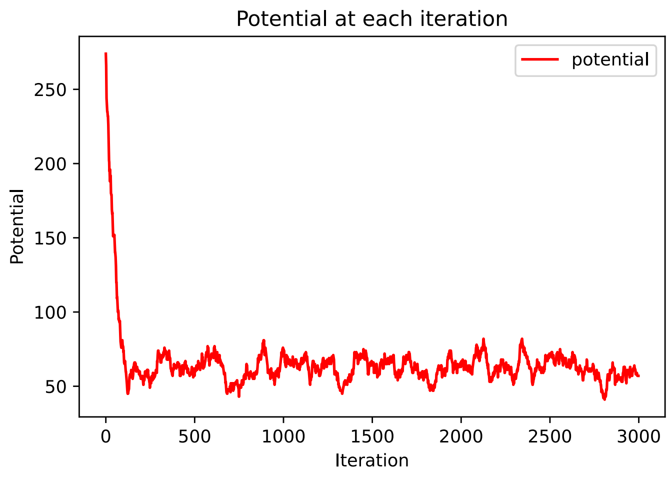

Figure 14: Plot of the potential function at each iteration with $\eta = 0$. The minimum potential reached is 41.0

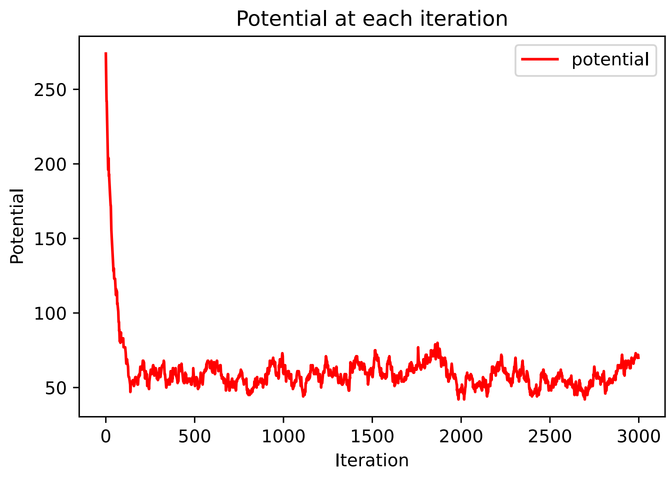

Figure 15: Plot of the potential function at each iteration with $\eta = 0.1$. The minimum potential reached is 42.0

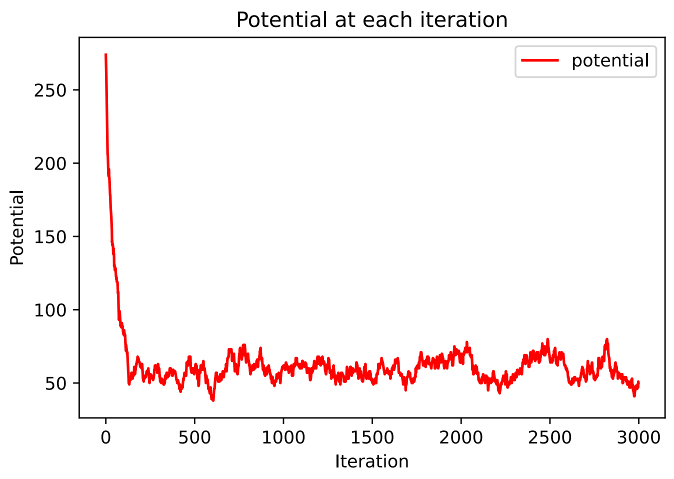

Figure 16: Plot of the potential function at each iteration with $\eta = \ln(t)/100$. The minimum potential reached is 38.0

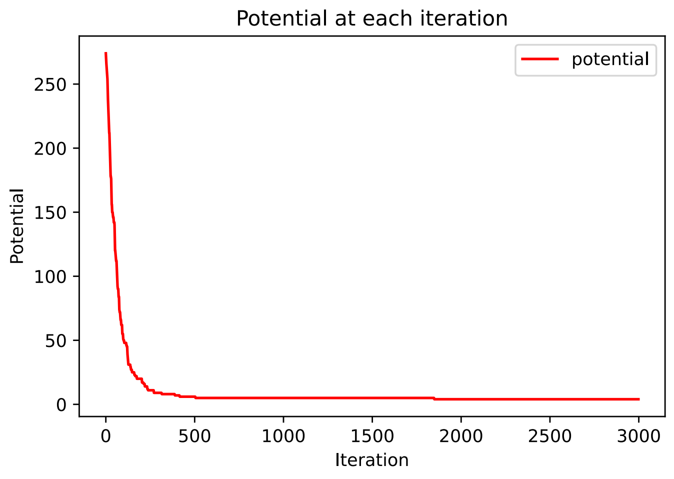

Figure 17: Plot of the potential function at each iteration with $\eta = 100$. The simulation converges.

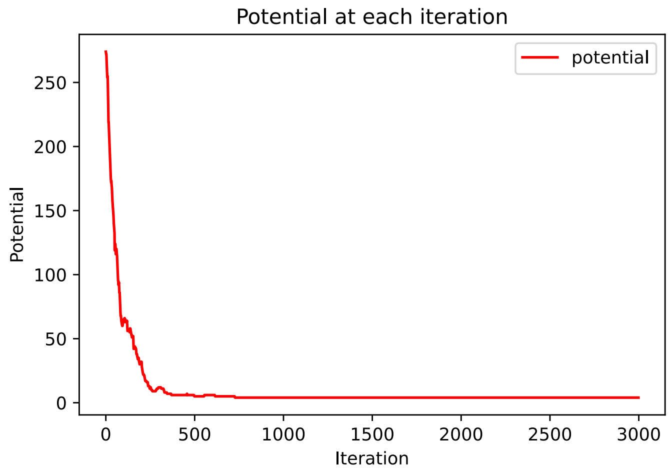

Figure 18: Plot of the potential function at each iteration with $\eta = t/100$. The simulation converges.

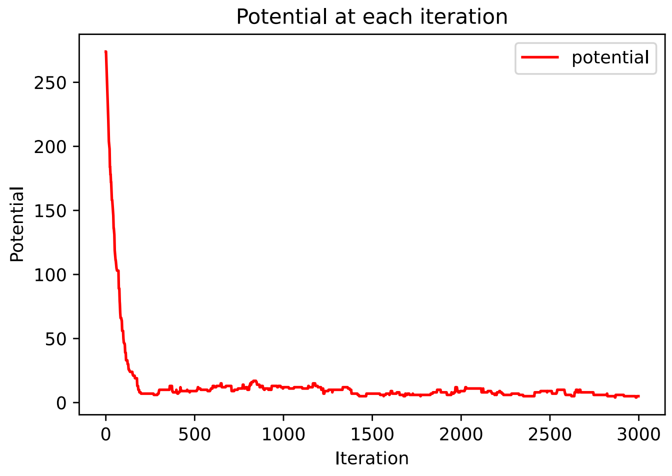

Figure 19: Plot of the potential function at each iteration with $\eta = t^2/100$.

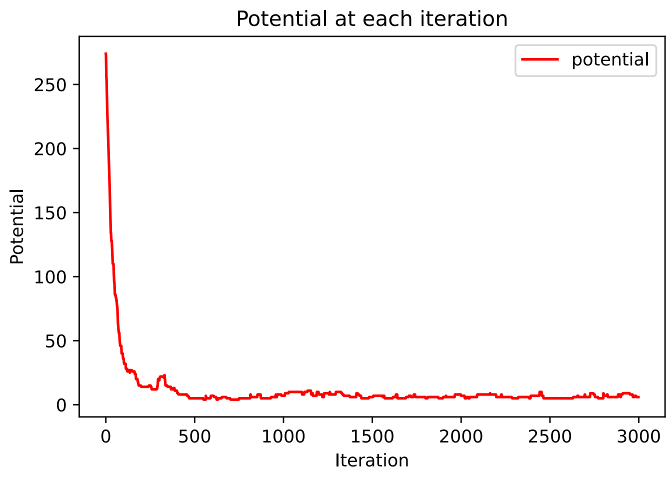

Figure 20: Plot of the potential function at each iteration with $\eta = t^4$.