Homework assignment 3

Jiahao Zhang

January 15, 2023

Abstract

This is a proposed solution for homework assignment 3 of the course Network Dynamics and Learning 2022/23. For the result, some numbers have been truncated for a better visualization quality

Exercise 1

Problem 1.1

- In this part you will simulate an epidemic on a symmetric k-regular undirected graph with node set $V = \{1, ..., n\}$ where every node is directly connected to the $k = 4$ nodes whose index is closest to their own modulo n. The graph you will simulate the epidemic on will however contain n = 500 nodes. The disease propagation model that you will use to simulate the epidemic is a discrete-time simplified version of the SIR epidemic model The following transition probabilities drive the epidemic:

$$P(X_i(t+1) = I | X_i(t) = S, \sum_{j \in (V)} W_{ij} \delta^I_{X_j(t)=m}) = 1 - (1-\beta)^m$$
$$P(X_i(t+1) = R | X_i(t) = I) = \rho$$

  where $\sum_{j \in (V)} W_{ij} \delta^I_{X_j(t)}$ is the number of infected neighbors for node i. You should simulate an epidemic on a symmetric k-regular graph $G = (V, E)$ with $|V| = 500$ nodes and $k = 4$. Let $\beta = 0.3$ and $\rho = 0.7$. Simulate the epidemic for 15 weeks. You can choose an initial configuration with 10 infected nodes selected at random from the node-set V, or make a different choice of initial configuration. Do this N = 100 times and plot the following:

  1 The average number of newly infected individuals each week. In other words, you should plot how many people become infected each week (on average).

  2 The average number of susceptible, infected, and recovered individuals each week. In other words, you should plot how many individuals in total are susceptible/ infected/ recovered each week (on average).

sol In order to create the symmetric k-regular graph with 500 nodes, we start creating its adjacent matrix W by filling the relative sub-diagonals, as shown in the figure

$$W = \begin{pmatrix} 0 & 1 & 1 & 0 & \cdots & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & \cdots & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \end{pmatrix}$$

where a node is connected only to the 4 closest indices node. An example is given for n = 10 in Fig 1

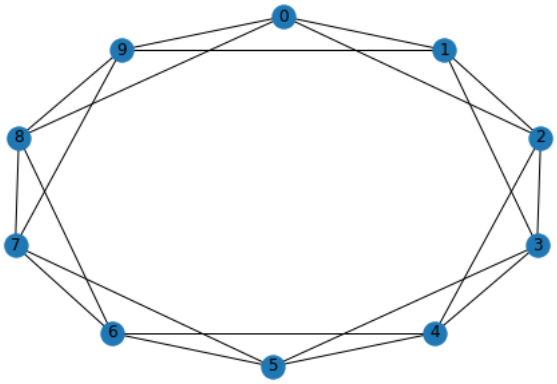

Figure 1: Example of a 4-regular graph with 10 nodes

After creating the 4-regular graph with 500 nodes, we can proceed with the simulation of the discrete SIR distributing at random the initial 10 infected nodes. The graphs 2 and 3 shows the SIR epidemics on the 4-regular graph

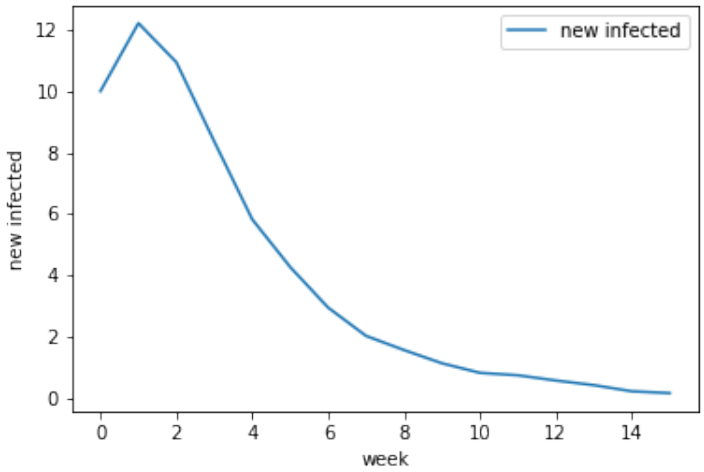

Figure 2: Number of newly infected nodes per each week

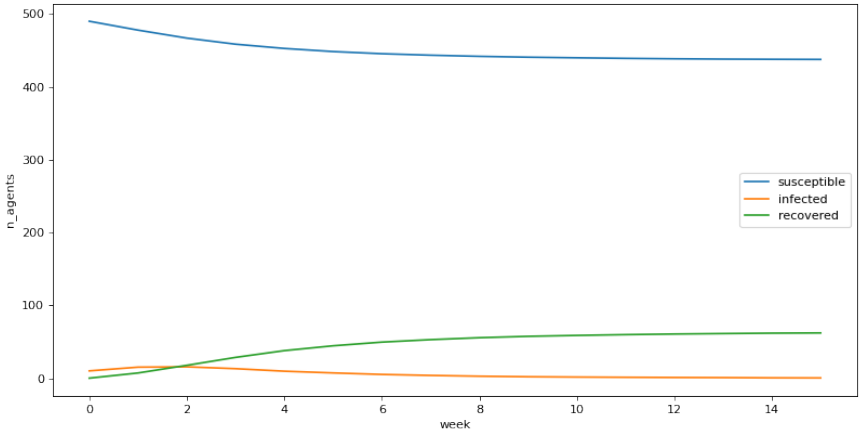

Figure 3: accumulated number of susceptible, infected, and recovered for each week

Problem 1.2

- The goal is to, by using the preferential attachment, generate a random graph of a large size (at least 900 nodes) with an average degree $k \in Z+$. Let the initial graph $G_1 = (V_1, E_1)$ be a complete graph with $|V1| = k_0 = k + 1$ nodes

sol Following the hint given by the exercise, we generate a random graph with the preferential attachment, where the average degree is equal to k. This method will be used in the next exercises for generating more random SIR epidemics.

Problem 2

- Using the methods developed in Problem 1.2 generate a preferential attachment random graph $G = (V, E)$, with $|V| = 500$ nodes. The average degree should be k = 6. Let $\beta = 0.3$ and $\rho = 0.7$. Do this N = 100 times and plot the following:

  1 The average number of newly infected individuals each week. In other words, you should plot how many people become infected each week (on average).

  2 The average number of susceptible, infected, and recovered individuals each week. In other words, you should plot how many individuals in total are susceptible/ infected/ recovered each week (on average).

sol To create the random graph, we use the functionality developed in Problem 1.2. The resulting graphs are as follows :

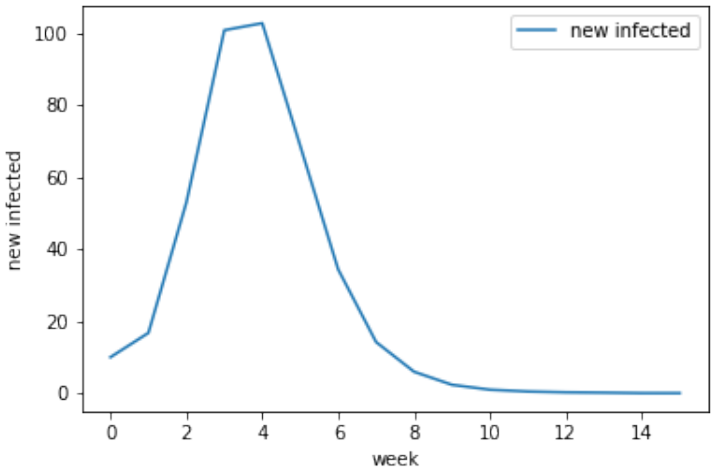

Figure 4: Number of newly infected nodes per each week

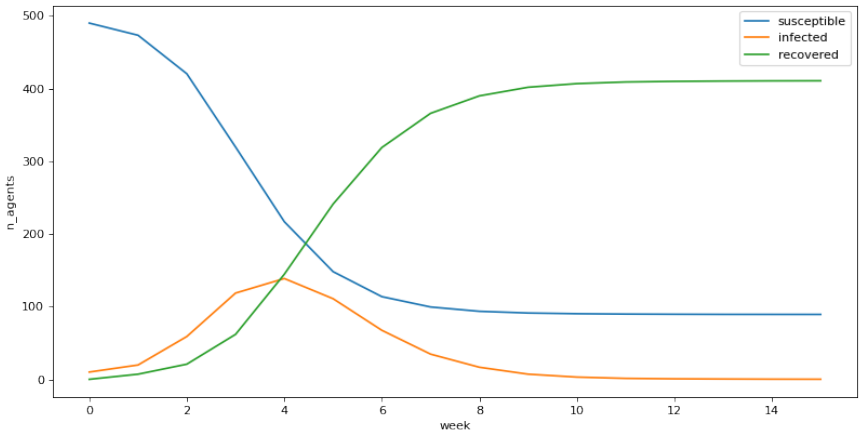

Figure 5: accumulated number of susceptible, infected, and recovered for each week

We notice that despite using the same amount of edges as done in Problem 1.1, we have a large increase in the number of newly infected agents in the first few days. This is due to different facts:

1 We have a larger average degree with respect to the 4-regular graph (in fact here we have on average 6 adjacent nodes for each node).

2 Despite having an average degree equal to 6, we have in general an imbalanced graph in terms of degree: in fact, some nodes may have degrees very low, meanwhile, some of them may have very large degrees because of the preferential attachment property. This leads to some particular nodes having huge infecting effects on their neighbors, explaining why we have a large increase of newly infected nodes.

Problem 3

- In this part you will essentially do the same thing as before, but you will also try to take some action to slow down the epidemic. This is normally done using vaccination. Therefore, during each week, some parts of the population will receive the vaccination. The vaccination is distributed in the weeks as follows

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

sol The resulting graphs after adding the vaccination are as follows (keeping all the same parameters of Problem 2)

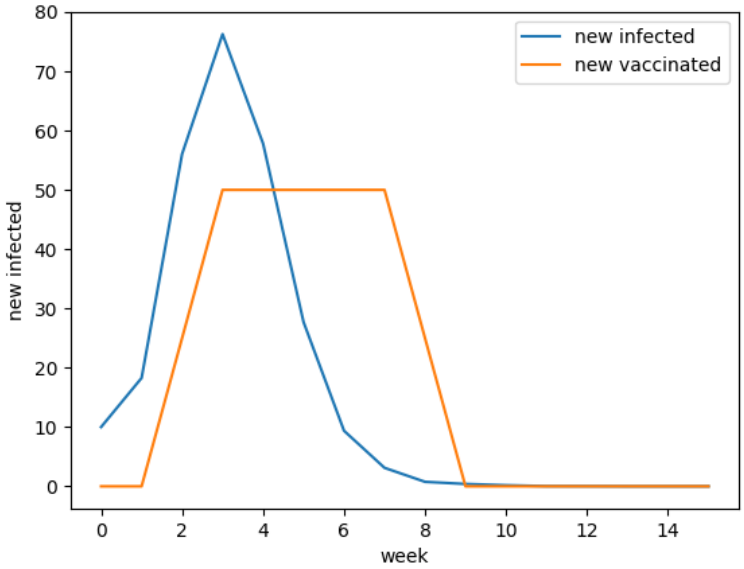

Figure 6: Number of newly infected nodes per each week

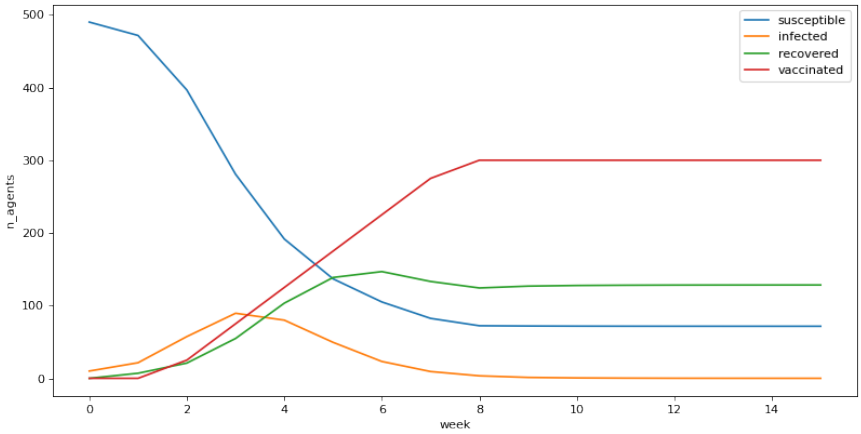

Figure 7: accumulated number of susceptible, infected, recovered, and vaccinated for each week

It is easy to see that the number of new infected each week decreased because of the effect of the vaccine, showing that in a general case, the vaccine has a very good effect on controlling the spread of an epidemic.

Problem 4

- In this part you will use all the previous parts in order to estimate the social structure of the Swedish population and the disease-spread parameters during the H1N1 pandemic. The vaccination is distributed in the weeks as follows

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

The number of nodes is $|V| = 934$ and the goal is to find the set of parameters (k, β, ρ) in order to get the newly infected node for each week similar to the real case :

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

sol Starting from the suggested parameters $(k = 10, \beta = 0.3, \rho = 0.6)$, setting an appropriate variation for each parameter, and reducing this variation by half when we reached each

multiple of 10 iterations because otherwise, the time consumed by the algorithm would be too long, we obtain some best-performing parameters. Since the gradient-based search is done by setting the number of simulations $n = 10$, after obtaining a set of possible best parameters, we run those parameters on a simulation with $n = 100$ to ensure accuracy and reproducibility. This is done because the are many random variables in our experiments, specifically:

- The randomly generated graph with average degree k
- The positioning of the 1st infected node
- The positioning of the vaccinated people

Among the gradient search, the best set of parameters found associated with their respective root mean square error (RMSE) are reported in the following graph:

n	k	β	ρ	RMSE10	RMSE100
1	10	0.2	0.4	5.72	8.88
2	9	0.2	0.45	**4.56**	6.35
3	7	0.25	0.45	5.69	**5.57**

Table 1: possible parameters set

Since the score may vary very highly according to the generated graph, we show some of the simulations with the best simulations both for n = 10 experiments and for n = 100 experiments.

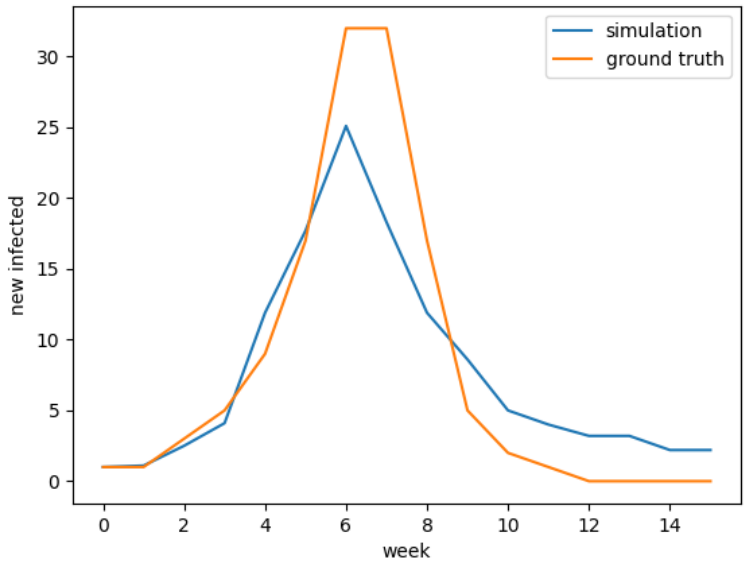

Figure 8: Number of newly infected nodes per each week for 10 simulations with parameters $(k = 9, \beta = 0.2, \rho = 0.45)$

The picture above shows the best-obtained simulation with only 10 simulations on the set of parameters $(k = 9, \beta = 0.2, \rho = 0.45)$. Observe that despite the newly infected behavior being very similar to the one we are looking for, the number of simulations is too low to consider those parameters as a good approximation of the model. If we set the number of simulations to 100 with the same set of parameters we will obtain the following graph

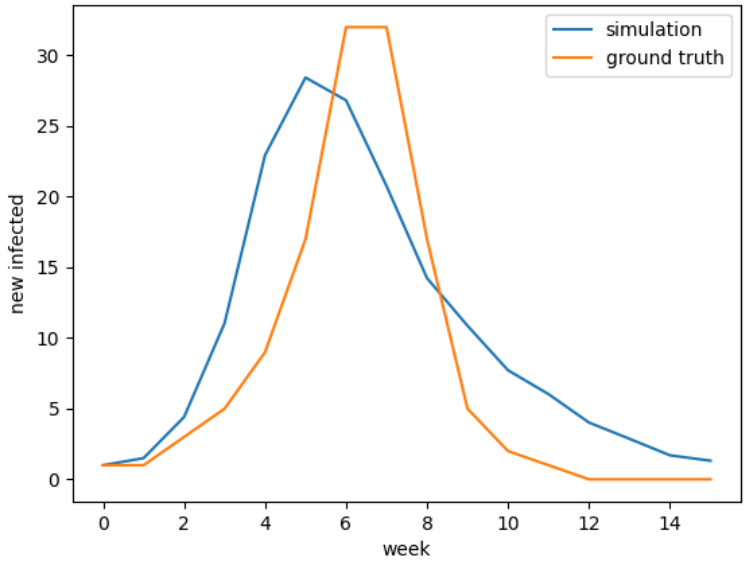

Figure 9: Number of newly infected nodes per each week for 100 simulations with parameters $(k = 9, \beta = 0.2, \rho = 0.45)$

which is still acceptable as performance but the infection increases too early from the expected behavior. Instead, for the set of parameters with larger k, despite not having that good performance on 10 simulations, we observe that for 100 simulations we have a better approximation. In particular, considering the set $(k = 7, \beta = 0.25, \rho = 0.45)$, we obtain the following graph with the stats related to the population

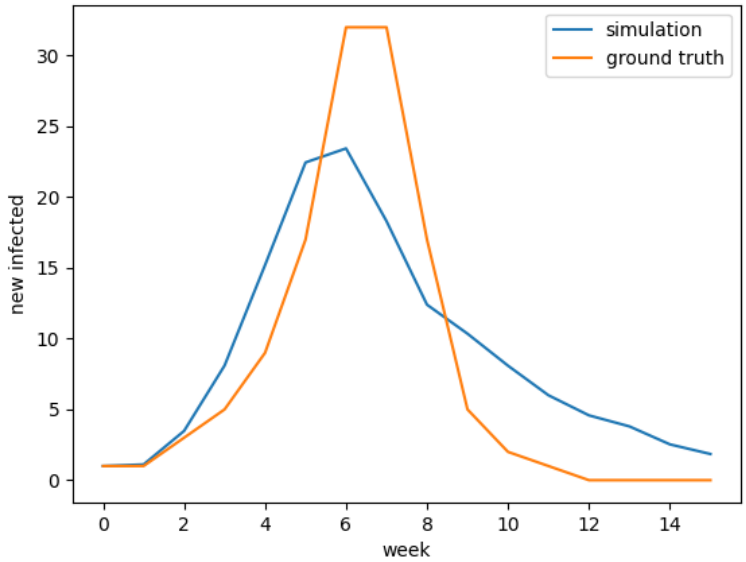

Figure 10: Number of newly infected nodes per each week for 100 simulations with parameters $(k = 7, \beta = 0.25, \rho = 0.45)$

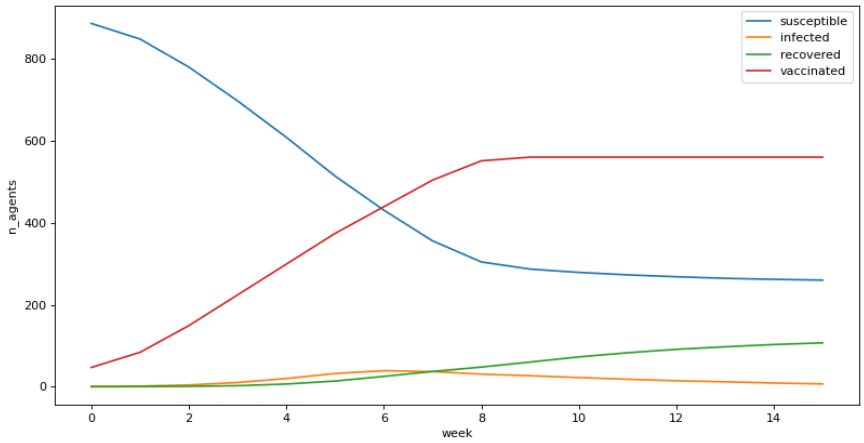

Figure 11: population stats each week for 100 simulations with parameters ($k = 7, \beta = 0.25, \rho = 0.45$)

The graph 10 better approximates the real case of H1N1. As reported in the table 1, this combination of parameters leads to the lowest RMSE, which is exactly the goal of this exercise.

Problem 5

- In this section, it is required to find a new graph or a new algorithm for fine-tuning and running the experiments in Problem 4.

  By keeping the same algorithm, a proposed approach would be to run the algorithm on a random graph created according to the **small world** approach. Recall that in this case, we cannot guarantee the average degree of the graph because we are considering as starting point a k-regular graph, and we add edges according to the probability p, which would be an additional parameter that we need to fine-tune.
  Since for the creation of the small world graph, we have a complexity of $O(n^2)$ where n is the number of nodes, we are likely to spend more time running the algorithm. We consider as starting configuration ($k = 10, \beta = 0.3, \rho = 0.6, p = 0.003$) with respective proportionate variance (recall that in this case $\delta k = 2$ because otherwise, we cannot obtain a k-regular graph with odd k.
  We also notice that in this case, the algorithm will take longer to complete a whole set of parameters, and this is because we have an additional parameter so we are likely to run $3^4 = 81 * 10$ simulations for each epoch. Due to time and computation resources, this algorithm will be run only once and the results are shown below with the relative graph.
  The best set of parameters found after 10 iterations of the algorithm is the following: ($k = 10, \beta = 0.15, \rho = 0.85, p = 0.005$) returning a $RMSE = 5.58$. Running the experiment with 100 simulations using the same parameter returns us a $RMSE = 5.74$ which is very similar to the one returned by the preferential attachment random graph. The relative graph related to newly infected per week and statistics related to susceptible/ infected/ recovered/ vaccinated per each week are the following:

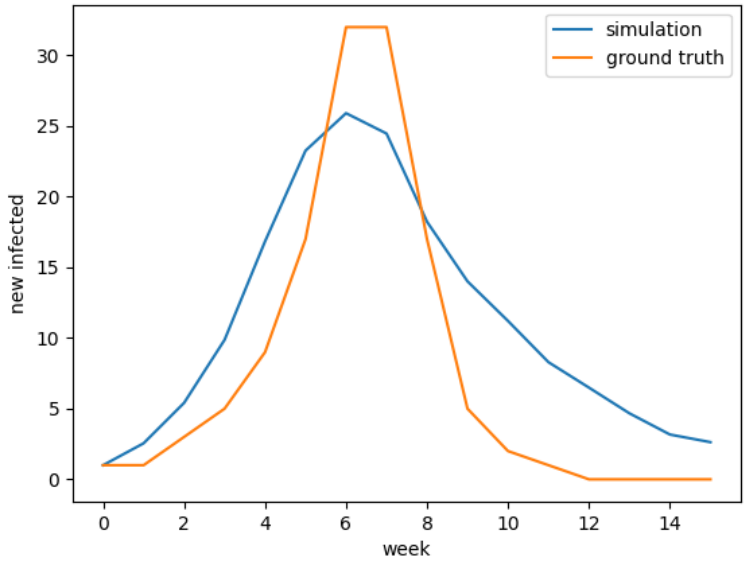

Figure 12: Number of newly infected nodes per each week for 100 simulations with parameters $(k = 10, \beta = 0.15, \rho = 0.85, p = 0.005)$

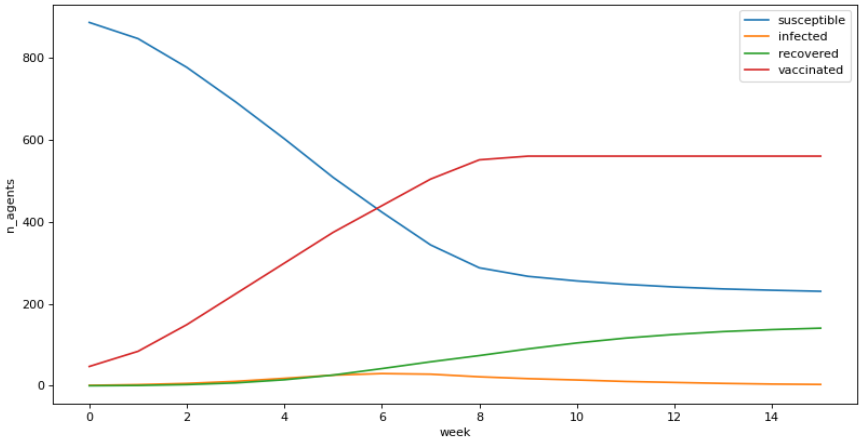

Figure 13: population stats each week for 100 simulations with parameters $(k = 10, \beta = 0.15, \rho = 0.85, p = 0.005)$

Notice how despite the approximate graph is very similar to the one we are looking for, the model and parameters are not exactly the ones we are looking for.

- For what concerns the algorithm for fine-tuning, a possible different approach would be to set a window for each set of parameters (for example, 5 experiments per window). Then, at the end of each window, we do the average among all the parameters and set those parameters as the starting point for the next window. To guarantee some stability (to avoid the case that the starting configuration is very bad), we could for example reduce by half the delta for every 2 windows. The strong point of this approach is that we take into consideration all the previous parameters, meanwhile, in the algorithm proposed in Problem 4, we only take into consideration the best parameters we have in the previous step. The code for this algorithm is not implemented due to time constraints, but it could be a possible approach for future work.

Notice that in general, RMSE is just a possible method to measure the distance between the ground truth and the simulation, and this may influence the grid-search algorithm in finding the

best parameters to simulate the epidemic.
Another interesting observation is the starting parameters. For our problems, we start from 2 different choices of starting parameters. But a more intuitive way would be to enlarge the search space in the first few iterations and then based on them, start the search in a more restrictive area to avoid being stuck in local minima. This approach is very similar to the dynamic learning rate concept present in machine learning when training the best-performing model.

Exercise 2

Problem 1

- In this part, we will study graph coloring as an application of distributed learning in potential games. We start by analyzing a line graph with 10 nodes. Denote the $i-th$ node state by $X_(t)$ and the set of possible states by $C = \{red, green\}$. At initialization, each node is red, i.e, $X_i(t) = red$ for all $i = 1, ..., 10$. Every discrete time instance t, one node $I(t)$, chosen uniformly at random, wakes up and updates its color according to the NBR rule, equivalently

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

where the cost can be computed as

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

For simplicity, we define $\eta(t)$ as

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

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

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

sol Starting from the problem, we conclude that the solution is obtained when the potential $U(t) = 0$.
The following graph shows the potential function over time for a single run of the game, setting as starting configuration the color red which will be encoded as zero.

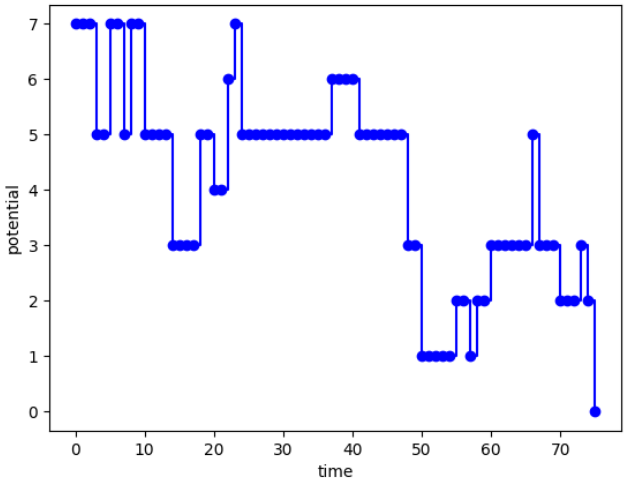

Figure 14: potential function over time

Notice that here we are simulating a noise best response (NBR) distributed learning algorithm. That's why we also make moves that increase the potential function. In the end,

after a reasonable amount of time interval, we have the convergence of the potential game, meaning that we have a feasible solution (marked as $U(t) = 0$). In fact, when we check the states of each node, we will find the final configuration as follows:

$$config = [0, 1, 0, 1, 0, 1, 0, 1, 0, 1]$$

Figure 15: coloring of the final configuration

which corresponds to one of the 2 possible solutions to our problem.

Problem 2

- Next, we use the coloring algorithm for the problem of assigning WiFi channels to routers. The set of possible states is

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

  and the cost function is

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

sol We first do a simple inspection of the given graph.

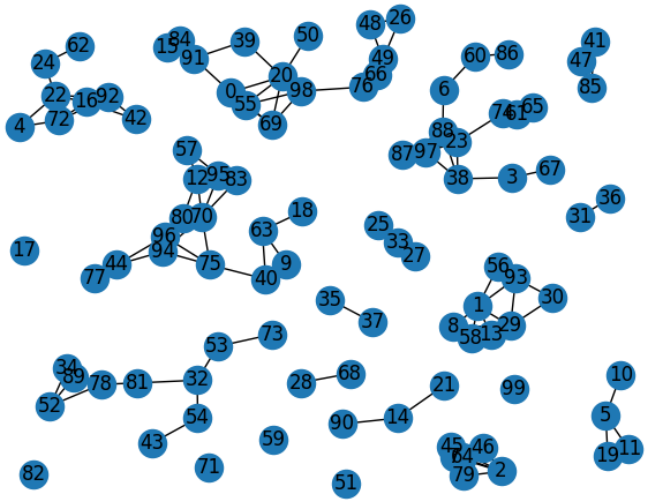

Figure 16: WiFi network

It is easy to notice that the given graph is not fully connected (For example, node 17 and node 82 are isolated nodes).

This composition may facilitate the convergence of the graph because the isolated nodes have zero contribution to the potential function (they have no adjacent nodes).

After this brief observation, let's start with the experiment with the various configurations where we suppose uniform color as starting configuration, and run the distributed algorithm.

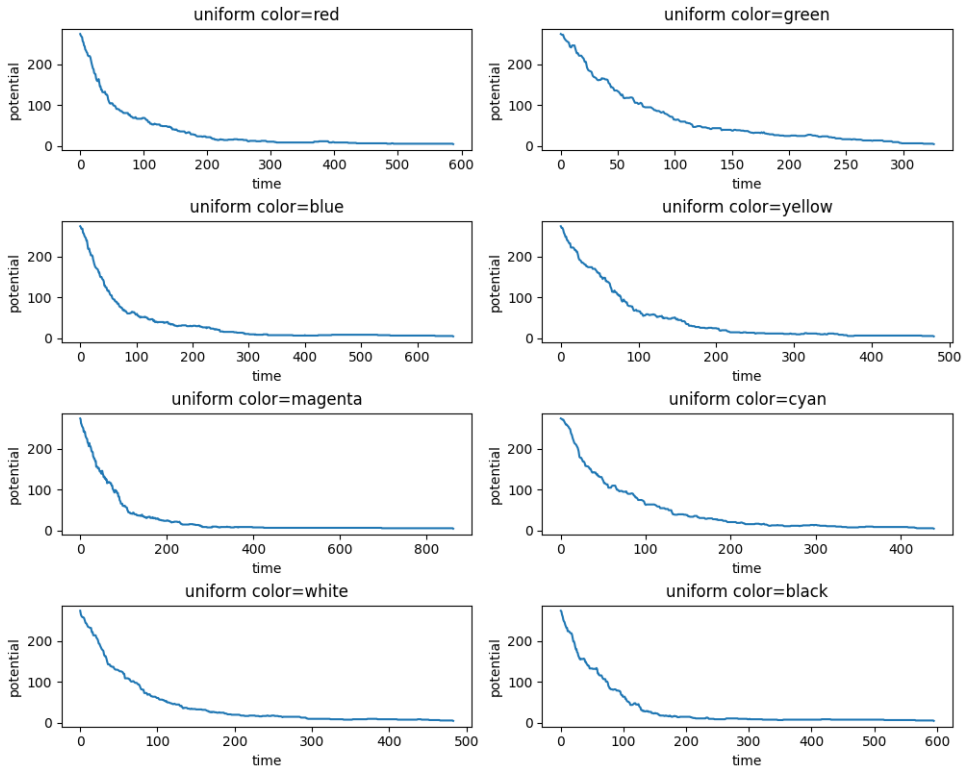

Figure 17: Potential function over different uniform starting color

Few observations on the obtained graphs:

- For each initial configuration, the starting potential is all the same (actually, it corresponds to the maximal potential value we can have). This should not be strange because the cost depends on the different colors of nodes we have, but it is a sign that the simulations are correct.
- We notice that actually, the algorithm uses a very long time to reach the imposed threshold $U = 5$. This behavior may depend on the parameter $\eta(t)$ which in this case may increase in time too fast (at a certain time we will have that for the machine the probabilities are null because t is too large so the nominator of the NBR rule results to be zero).
- the minimum threshold for which the algorithm doesn't converge is obtained by setting $U = 3$, which means that the potential function never reaches the value $U = 3$. We could check if this is also the case for the random starting configuration

Now let's analyze the case where we have a random starting configuration:

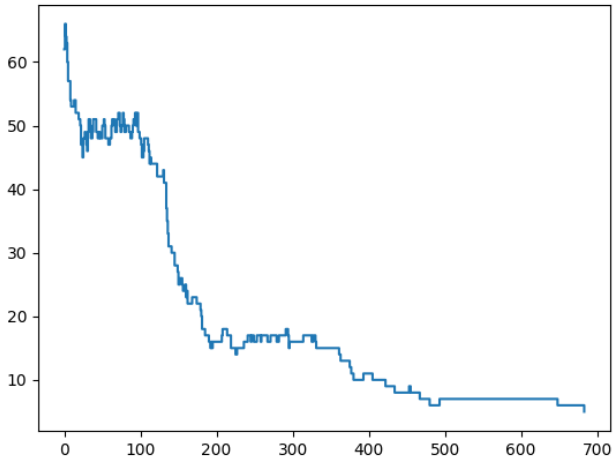

Figure 18: Potential function over random color starting configuration

In this case, it is intuitive to see that the starting potential value is lower because of the randomness of the colors for each graph. But despite that, we still have convergence after a long time, and setting as threshold the potential value $U = 5$, we obtain the result after a time interval comparable to those for the uniform color. We also notice that in this case, the minimum threshold not reachable is the same as for the uniform starting configuration, which is $U = 3$.
In the end, we show a possible configuration of colors, as requested by the problem

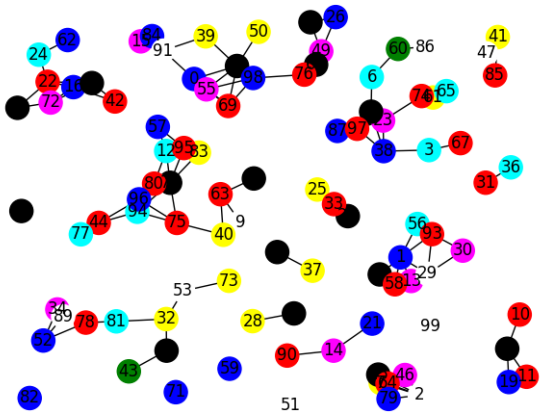

Figure 19: possible coloring starting from random configuration

Problem 3 (optional)

- In this section, we will try different values of the function $\eta(t)$, show their behavior, and give a small comment on the different chosen values. 3 different cases will be analyzed as follows :

  - starting from a constant value $\eta(t) = 2$
    In this case, since we do not have an increasing function, we are likely to have a very long convergence time over the same threshold we have set in the case of Problem 2.2. We should set the threshold for $U = 9$ to have a convergence time similar to those simulations done in Problem 2.2, both in the case of uniform starting configuration or random starting configuration. Here we report the graphs by setting the threshold $U = 9$

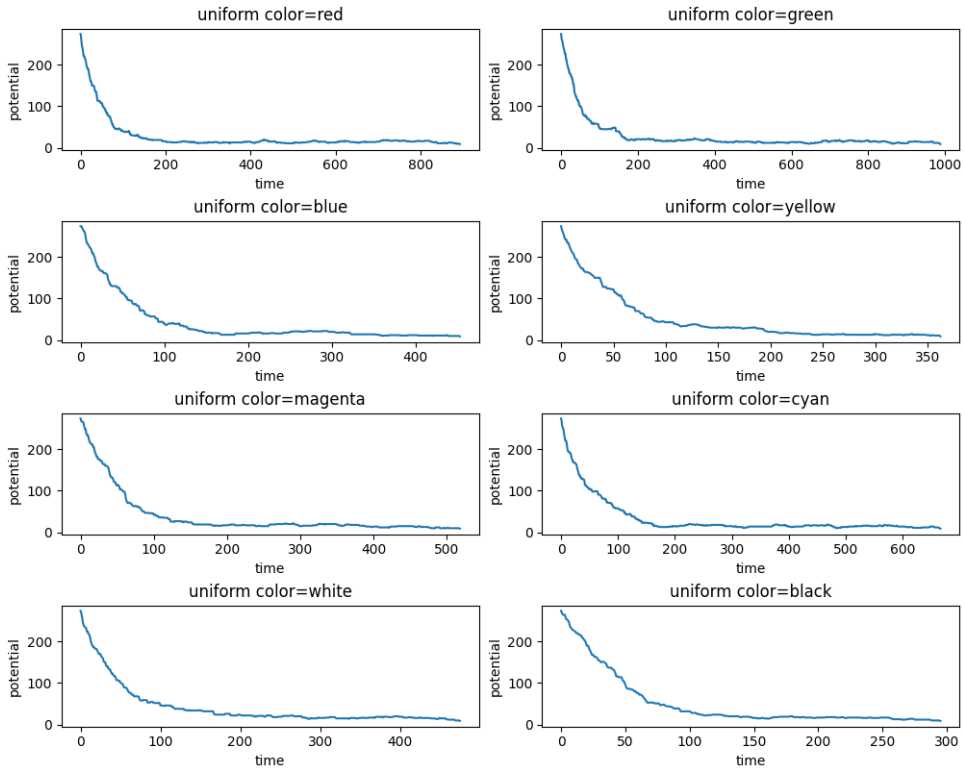

Figure 20: Potential function over different uniform starting color

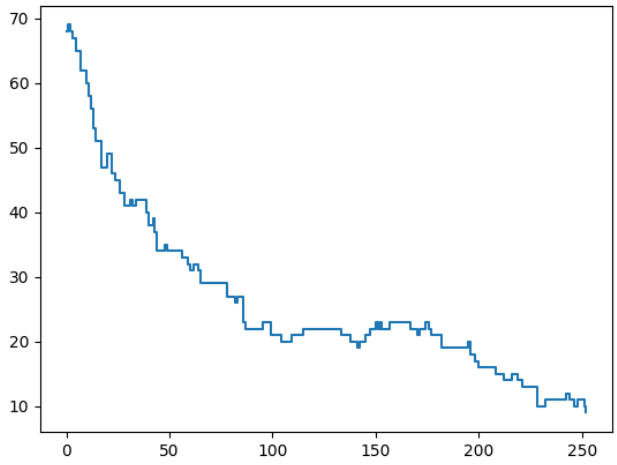

Figure 21: Potential function over random color starting configuration

- Starting from a quadratic function $\eta(t) = t^2/100$
  We notice that the simulation cannot go on for a very long time because since the function is quadratic increasing, when put in the negative exponential it has a faster effect on reducing the value to zero, leading us to have zero probability on the calculator. Thus this choice of $\eta(t)$ does not apply to the NBR algorithm.
- Starting from a logarithmic function $\eta(t) = \log(t/100)$
  We notice in this case that we have convergence over near zero potentials, but still we are not able to surpass the threshold $U = 3$. Setting as threshold $U = 5$ as done in Problem 2.2 returns us a larger convergence time concerning the function $\eta(t) = t/100$. The following graph shows the returned potential graph when setting potential threshold $U = 5$

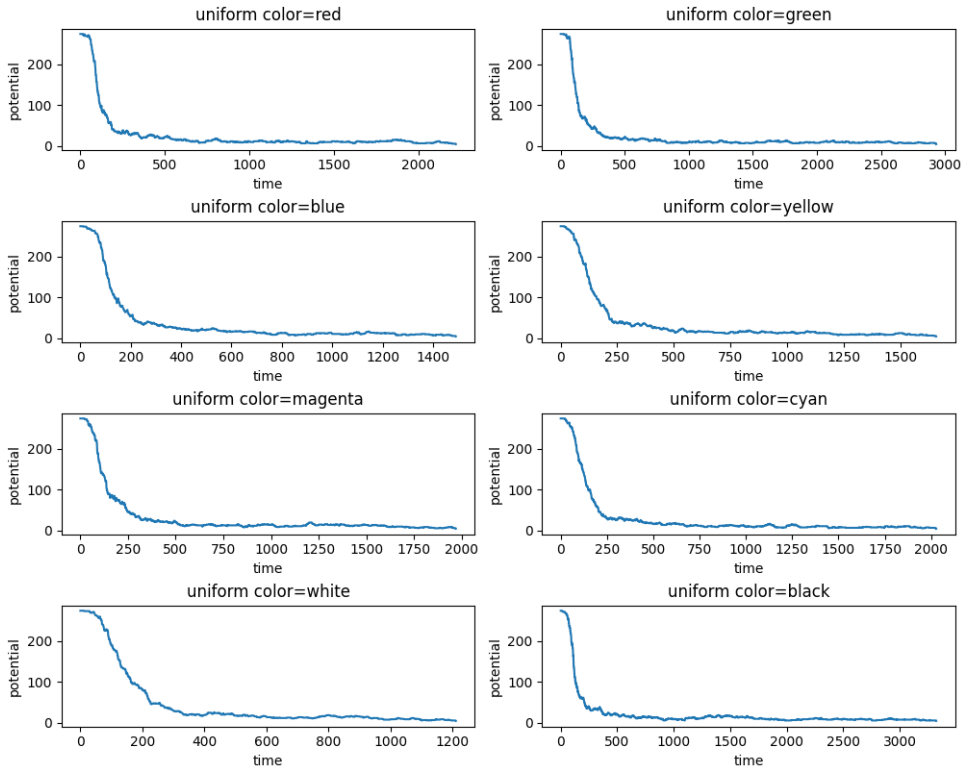

Figure 22: Potential function over different uniform starting color

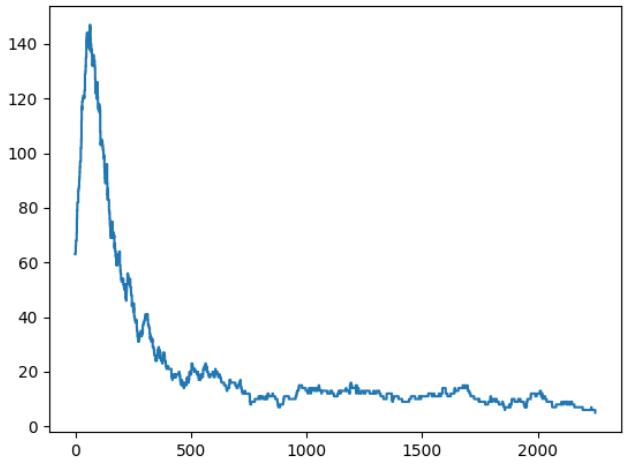

Figure 23: Potential function over random color starting configuration