Homework 3 - Network Dynamics and Learning

Fabrizio Pisani s305391, Matteo Zulian s310384

January 20, 2024

The code and the report has been done in collaboration without a strict division of work.

Exercise 1

In this section we were asked to simulate a pandemic on a network, by means of a theoretical model known as SIR, with the goal of learning the network-structure characteristics and disease-dynamics parameters of the pandemic in Sweden 2009 and its vaccination campaign.

1.1 Preliminary Part

SIR model

The theoretical model SIR is applied to a network, where nodes represent people and links represent their interactions, namely possible source of contagion. This method consists in labeling each node with one of three state : S *susceptible*, I *infected* and R *recovered*.

When a node i is in state S, it can be infected by its neighbours with probability:

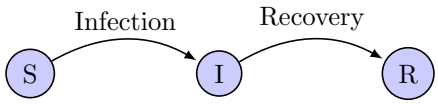
$$\mathbf{P}(X_i(t+1) = I | X_i(t) = S, m) = 1 - (1-\beta)^m \tag{1}$$

where β is the *interaction frequency parameter*; notice how $\mathbf{P}$ increases with the number of infected neighbours m.

On the other hand an infected node can heal and enter the recovery state with probability:

$$\mathbf{P}(X_i(t+1) = R | X_i(t) = I, m) = \rho. \tag{2}$$

In the following state diagram we show schematically the state evolution of any node in this kind of model. It can be also seen that R is a *trapping state*, which means that once a node becomes a Recovery node, it won't change state anymore. A *trapping configuration* is a configuration of states that if reached it's no longer possible to change and in this kind of simulation it is given by all the combinations of states of S and R. Which means that once there are no more I nodes it is not possible to have other infections.



SIR Epidemic on a known graph

The first warm-up problem consisted in simulating an epidemic on a symmetric k-regular undirected graph with 500 nodes and $k = 4$ for 15 weeks (steps). In this kind of graph, every node i is connected to the k nodes, whose index is closest to i.

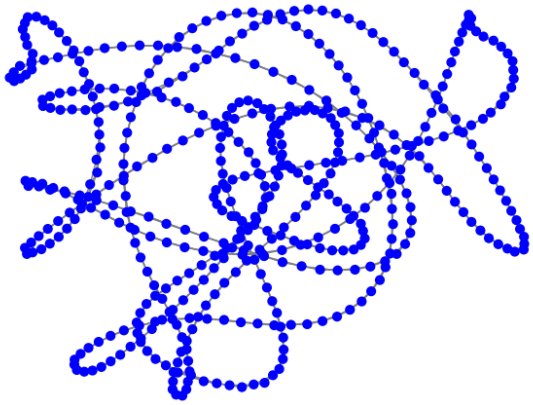


Figure 1: k-regular Graph with 500 nodes and k = 4

In this problem, we assumed $\beta = 0.3$ and $\rho = 0.7$ and we operated a simulation with 10 initial infected nodes.
We figured the following algorithm to simulate the SIR model:

Algorithm 1 SIR Epidemic on a known graph

Initialization Simulation Parameters

Set initial Infected nodes

for *every simulation* **do**

  for *every week* **do**

    for *every Susceptible node* **do**

      Compute the probability of infection $\boldsymbol{P}$

      Infect with probability $\boldsymbol{P}$

    for *every Infected node* **do**

      Recover with probability ρ

    Save week results

  Update simulation averages

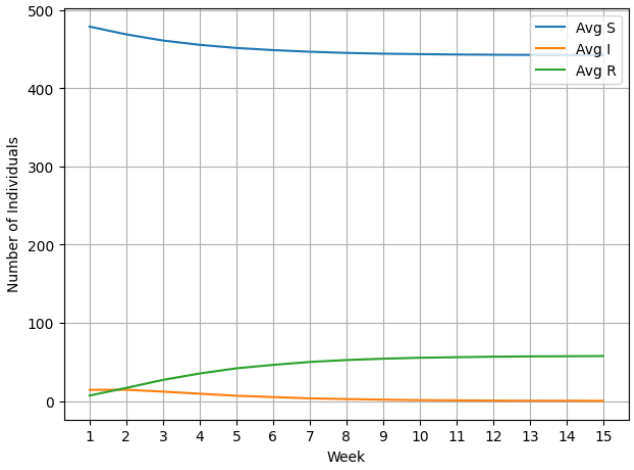


Figure 2: SIR on a k-regular Graph

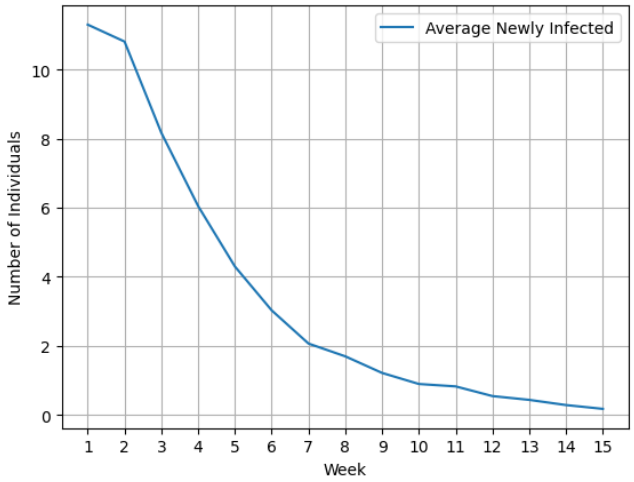


Figure 3: SIR on a k-regular Graph

Figure (4) and (3) show how nodes that are initialized randomly on the graph behave and extend the infection. However the epidemic slows down early and does not reach new nodes fast enough to infect most individuals in the population.

Generation of a random graph

In this section we learned how to generate a random graph according to the *preferential attachment model*. The goal is to have a randomly generated graph with average degree close to k. The idea is to start with a complete graph of $k+1$ nodes, then add a node and connect it with $k/2$ links to the existing nodes. Repeat until the graph reaches the desired number of nodes. The probability to construct an edge between the new node n and a node i at step t is

$$\mathbf{P}(W_{n,i}(t) = W_{i,n}(t) = 1) = \frac{w_i(t-1)}{\sum_{j \in V_{t-1}} w_j(t-1)}. \tag{3}$$

In this algorithm we summarized the steps of the construction of a random graph using the *preferential attachment model*.

Algorithm 2 Creation of a random graph with n nodes and average degree k

create complete graph with k+1 nodes
for *every new node until it reaches* n *nodes* **do**
    add new node
    Compute probability $\boldsymbol{P}$ *of attaching new node to the existing nodes*
    choose k/2 nodes to attach to, based on $\boldsymbol{P}$
    add links to those nodes

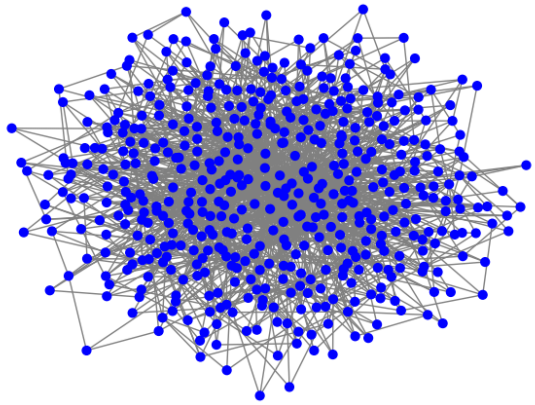


Figure 4: Random Graph with 500 nodes and k = 6

1.2 Pandemic without vaccination

The following problem consisted in simulating an epidemic based on the discrete-time SIR model using the preferential attachment model explained in previous section. We performed 100 simulations and plot the average total number of susceptible, infected and recovered individuals each week and the average number of newly infected individuals each week.

The parameters of the simulations k, β, ρ, number of nodes, number of weeks and number of initially infected individuals remained fixed for all 100 simulations.

In our simulations, we generated a graph with average degree $k = 6$ and 500 nodes. We set $\beta = 0.3$ and $\rho = 0.7$, and carry out each simulation for 15 weeks, with an initial configuration of 10 infected nodes.

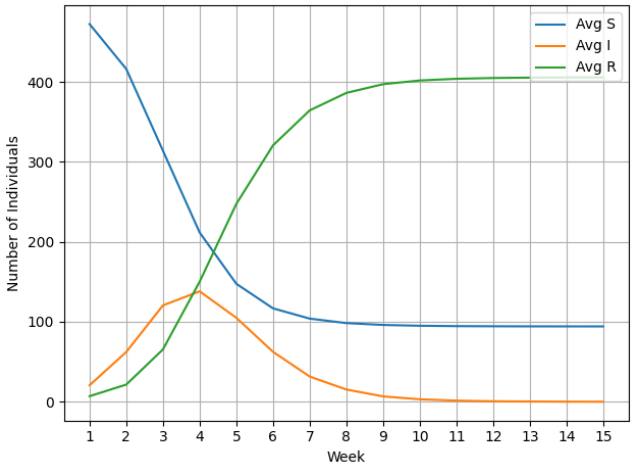


Figure 5: SIR on a random Graph

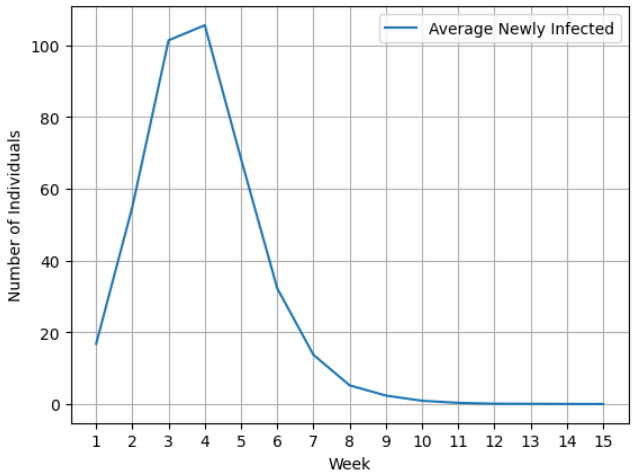


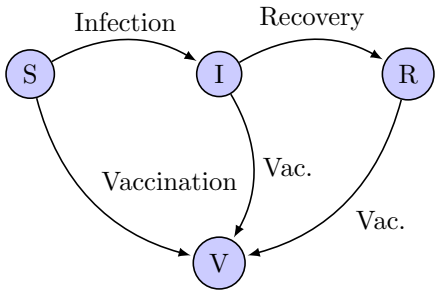
Figure 6: SIR on a random Graph

Since in this example we used $k = 6$ instead of $k = 4$, as in the previous example, we can see how the infection spread more rapidly due to the higher interconnections.

1.3 Pandemic with vaccination

In this section we introduced a vaccination plan for the population, therefore we added a new state to the possible states: V *vaccinated*.
Weekly vaccinations are carried out following the vaccination plan given by the

reference $Vacc(t)$, provided in the text of the exercise. Every week a designated percentage of the population, that hasn't received the vaccine yet, is vaccinated regardless of their previous state (S, I or R). Since we assume that the vaccination takes effect immediately once given, the nodes that receive vaccination during week i will not be able to become infected nor infect any other node in that same week, regardless of the state they were in immediately before the beginning of week i.
In the following state diagram we show schematically the state evolution of any node during a simulation, notice how the *trapping state* now is V. The *trapping configurations* now are all the combinations of S,R,V nodes state.



In each of the 100 simulations, we generated a graph with average degree k and 500 nodes and simulate for 15 weeks with parameters β and ρ and an initial configuration of 10 randomly chosen infected nodes. These settings are exactly the same as the previous simulation without vaccination, so results are more easily comparable. In addition, we specify the total fraction of population that has received vaccination by each week according to the following array.

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

In figure (7) we can see how the vaccination slowed the spreading of the epidemic.

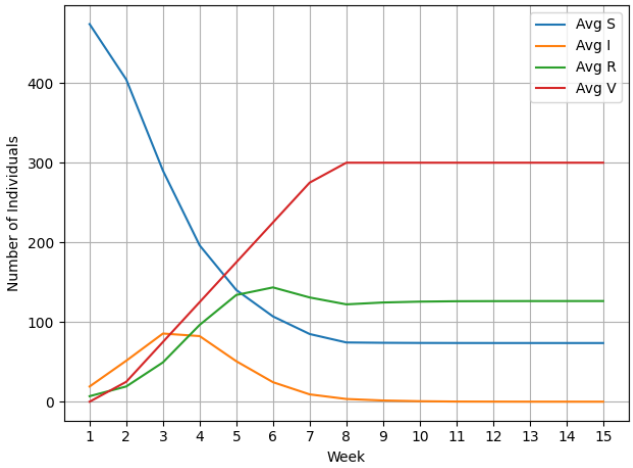


Figure 7: SIRV on a random Graph

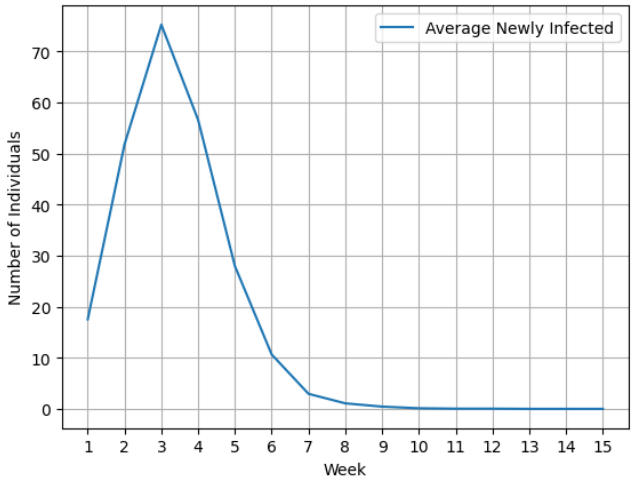


Figure 8: SIRV on a random Graph

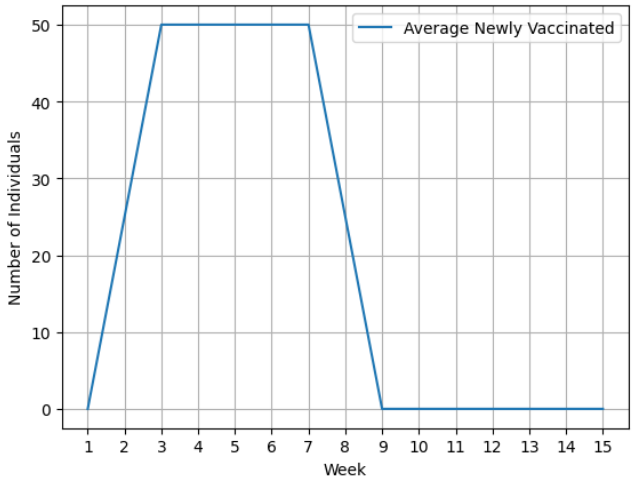
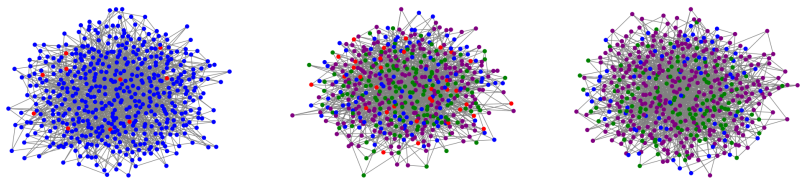


Figure 9: SIRV on a random Graph

Another view of the SIRV epidemic with susceptible, infected, recovered and vaccinated nodes.



(a) week 0 (b) week 5 (c) week 15

Figure 10: SIRV with 500 nodes, k = 6, 10 initial I

Notice how the vaccinated nodes and the spontaneous recovery block the infection, until there are no more I nodes, leading the network to a *trapping configuration* of S-R-V only.

1.4 The H1N1 pandemic in Sweden 2009

Our goal is to estimate parameters k , β and ρ for our model which best match the H1N1 pandemic in Sweden in 2009, which we simulate for 15 weeks. The fraction of population that had received vaccination during these weeks is reported in the following vector.

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

As a reference we were given a vector showing the number of newly infected in the 15-week period from week 42, 2009 to week 5, 2010:

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

$Vacc(0)$ and $I_0(0)$ must be interpreted respectively as percentage of nodes vaccinated and number of initial infected nodes, **before** the simulation even started. In other words in the beginning there is 5% of the population vaccinated while 1 node is infected.
The idea is to exploit a *gradient-descent* algorithm over the parameters k, β, ρ using as cost function the root-mean-square error (RMSE) between the simulation and the real pandemic:

$$RMSE = \sqrt{\frac{1}{n_W} \sum_{t=0}^{n_W} (I(t) - I_0(t))^2} \tag{4}$$

Where n_W is the number of weeks.

Basic Algorithm

One way to find the right parameters is to start from an arbitrarily set of initial values k_0, β_0, ρ_0 and then explore their neighbour space: $\{k - \Delta k, k, k - \Delta k\} \times \{\beta - \Delta\beta, \beta, \beta + \Delta\beta\} \times \{\rho - \Delta\rho, \rho, \rho + \Delta\rho\}$. Which gives $3 \times 3 \times 3 = 27$ possible combinations (26 if we don't count the non-explorative combination k, β, ρ). During the exploration phase the algorithm runs simulations and keeps track of the RMSE of each combination of parameters. Then it updates the 'central' point from which explore the neighbour. It iterates like that until there are no improvement in the RMSE. Here a little scheme of the algorithm:

Algorithm 3 H1N1 basic search

set $k^* = k_0, \beta^* = \beta_0, \rho^* = \rho_0$
for *every step* **do**
  for *every set* k, β, ρ *in* $\{k^*, k^* \pm \Delta k\} \times \{\beta^*, \beta^* \pm \Delta\beta\} \times \{\rho^*, \rho^* \pm \Delta\rho\}$ **do**
    Run 10 simulations with k, β, ρ
    Compute RMSE
  if *min RMSE is improved from previous step* **then**
    $k^*, \beta^*, \rho^* = k, \beta, \rho$ that gave RMSE min
  else
    end algorithm

In our algorithm we used as initial values: $k_0 = 10, \beta_0 = 0.3, \rho_0 = 0.6$ and as deltas : $\Delta k = 1, \Delta\beta = 0.1, \Delta\rho = 0.1$.
Due to the randomness of the graph and of the simulations we obtained different

optimal values for the parameters at each run.
Here a couple of results:

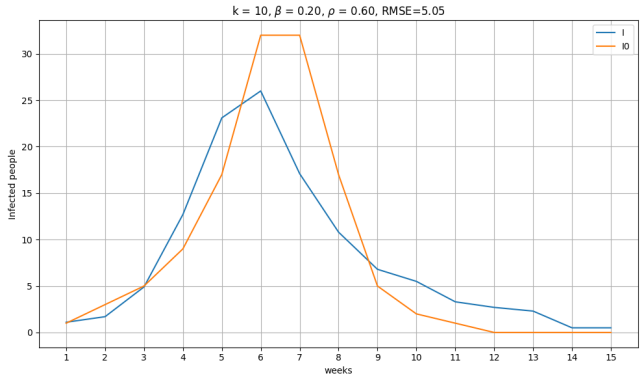


Figure 11: H1N1 run 1

k = 10, $\beta = 0.20$, $\rho = 0.60$, RMSE = 5.05
I = [1 1 4 12 23 26 17 10 6 5 3 2 2 0 0]
I0 = [1 3 5 9 17 32 32 17 5 2 1 0 0 0 0]

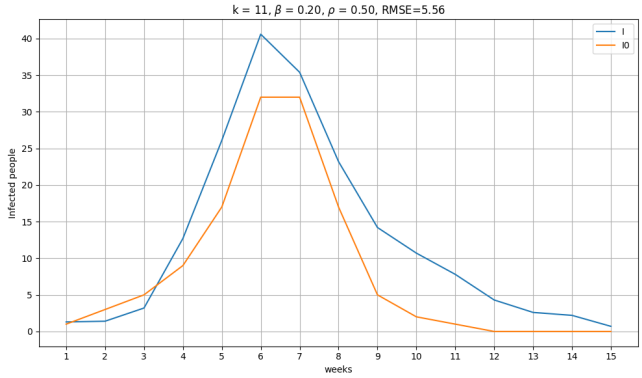


Figure 12: H1N1 run 2

k = 11, $\beta = 0.20$, $\rho = 0.50$, RMSE = 5.56
I = [1 1 3 12 26 40 35 23 14 10 7 4 2 2 0]
I0 = [1 3 5 9 17 32 32 17 5 2 1 0 0 0 0]

Improved Algorithm

We tried to improve the previous algorithm by changing the deltas during its progress. More precisely once the algorithm doesn't find an improved RMSE, instead of stopping, it decreases the deltas, looking for a more precisely solution.

In our run we used the same set of 3 deltas for β and ρ ($\Delta_1 = 0.1, \Delta_2 = 0.05, \Delta_3 = 0.01$), while we kept $\Delta k = 1$ for all the algorithm. This algorithm is more computational demanding than the basic one, but we gained more precised results:

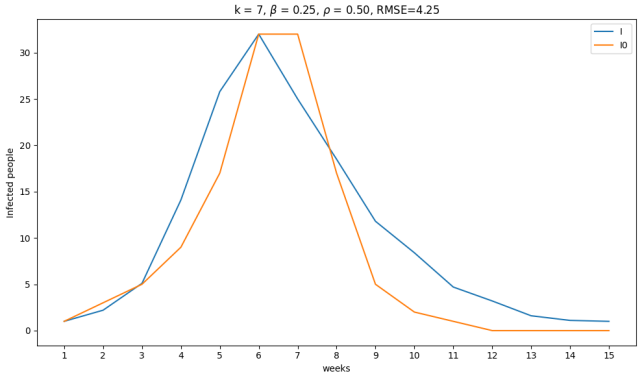


Figure 13: H1N1 run 1

k = 7, $\beta = 0.25$, $\rho = 0.50$, RMSE = 4.25
I = [1 2 5 14 25 32 24 18 11 8 4 3 1 1 1]
I0 = [1 3 5 9 17 32 32 17 5 2 1 0 0 0 0]

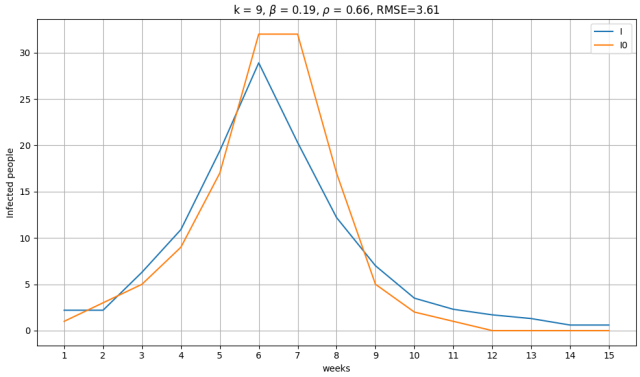


Figure 14: H1N1 run 2

k = 9, $\beta = 0.19$, $\rho = 0.66$, RMSE = 3.61
I = [2 2 6 10 19 28 20 12 7 3 2 1 1 0 0]
I0 = [1 3 5 9 17 32 32 17 5 2 1 0 0 0 0]

Exercise 2

In this section we were asked to study the problem of coloring, as an application of distributed learning in potential games. The goal was to find a *perfect coloring*: every node i was characterized by a color c and in the neighbor of i there was no node with the same c.

2.A Coloring on a Line Graph

In this exercise we studied a coloring problem over a line graph with 10 nodes, with two possible colors: red and green. The starting graph is represented in the figure 15, where every node is red. The goal was to reach a perfect coloring (an example in figure 16).

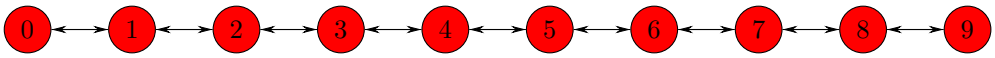


Figure 15: Initial line graph

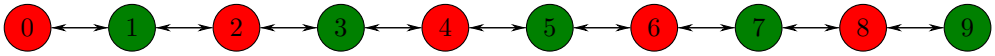


Figure 16: Example of a perfect coloring over a line graph

In order to reach our goal, we wrote and ran the algorithm 4. The node to update was chosen uniformally and the new color was selected from a probability distribution given by

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

where

- c was zero if the colors of the two considered nodes were different, and was one otherwise,
- $\eta = \frac{t}{100}$ (inverse of the noise).

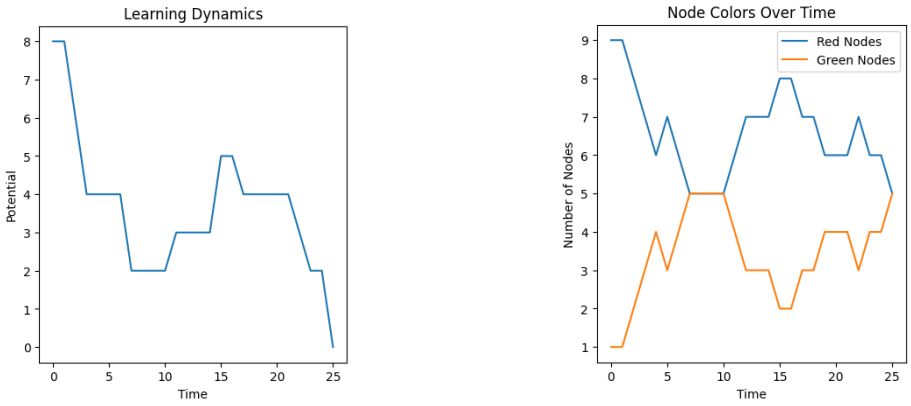
The stop condition of the algorithm depended on the potential: when

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

was zero, we found a perfect coloring.

We ran few times the algorithm and we reached a perfect coloring. This was possible also because we assumed that the noise decreased over time.

In the figure 17a there is the evolution over time of the potential in a specific simulation. In the figure 17b there is the evolution over time of the number of red nodes (in blue) and of green nodes (in orange): as expected, in the end there are five red nodes and five green nodes.



(a) Potential

(b) Nodes color

Figure 17: Plot showing the results of the simulation

Algorithm 4 Coloring problem

Initialization Simulation Parameters
for *every step* **do**
    Initialization of noise
    Random choice of node to update
    Compute the probability of choosing a color
    Random choice of updating color of the selected node
    Compute and store potential $\boldsymbol{U}$

    if $\boldsymbol{U} = 0$ **then**
        break

The maximum number of iteration was a crucial hyper-parameter: in fact, in the majority of the cases, the simulation reached the goal in less than 100

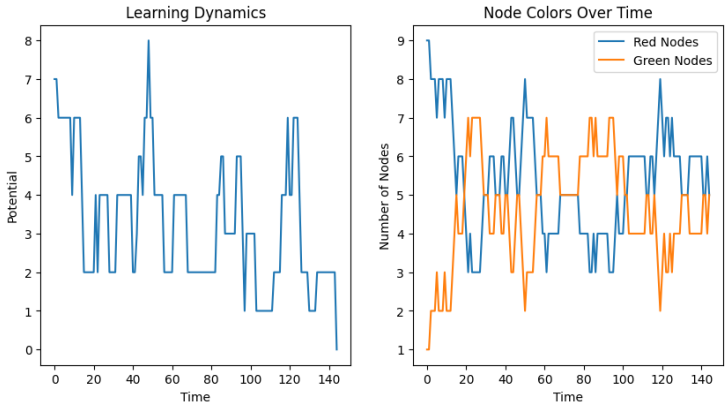


Figure 18: Scenario with more than 100 iteration

iteration. However, in same runs, the algorithm needed more iteration to reach the perfect coloring. This result was expected, because the selection of the node to update was random and no matter how close the perfect coloring was. We reported in the figure 18 this scenario.

2.B Coloring on an router-wifi assignment problem

In this section, we studied a generalized coloring problem. The analyzed scenario was a problem of assigning wifi-channels to router, and we modeled it as a coloring problem with 8 colors (red, green, blue, yellow, magenta, cyan, white and black) and 100 node (routers). The goal was to search a perfect coloring (i.e. find a coloring where *potential* $= 0$), or a near-zero potential. An other difference respect to the exercise $2.A$. was the cost function:

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

In this exercise, we used the same noise function of the previous exercise: $\eta(t) = \frac{t}{100}$ (the noise $\frac{1}{\eta}$ decreased at every time step). In figure 19 there is plot of noise over time.

In order to solve this problem, we adapted the algorithm 4 to the new scenario, by changing cost function, number of possible colors, number of nodes and matrix W.

We ran the algorithm with two different initial conditions:

1. at $t = 0$, every node has the some initial color, chosen randomly with uniform distribution;
2. at $t = 0$, every node has a random color.

Same Color at the beginning. In the first case, every node had the same color at the beginning. Also in this exercise, the number of iterations was crucial: with 150 iterations we found an high value of potential (around 25, figure 20a); with 600 iterations we reached a very low value of potential (around 3, figure 21a).

As expected, we did not reach the perfect coloring ($potential = 0$), but we found a near-zero potential solution after a sufficient number of iterations. This was possible also because we assumed that the noise decrease over time. We tested the algorithm also for 1500 iterations, but the potential did not improve, so we stopped.

We considered a solution with $potential = 3$ a good one. Furthermore, the node that had to update its color was chosen randomly. So, it was normal that the algorithm (after an initial transitory) chose a node that had no neighbours with the same color.

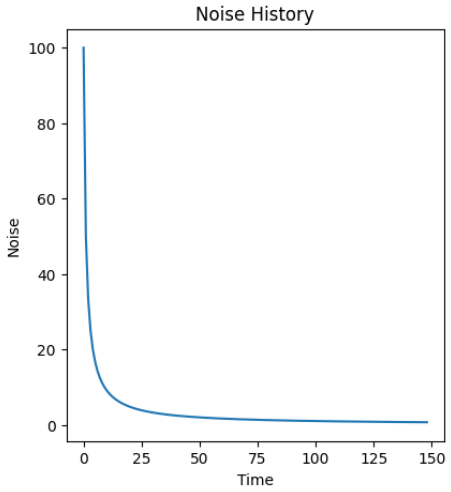
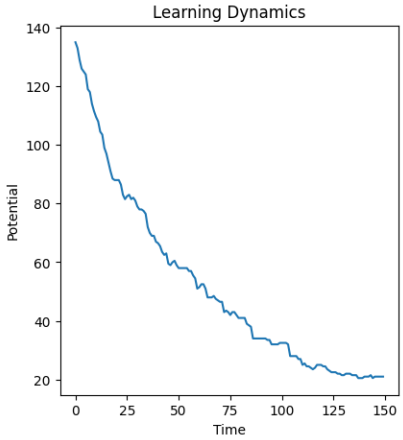
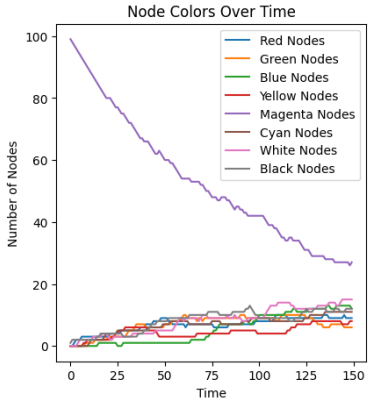


Figure 19: Noise over time ($\eta(t) = \frac{t}{100}$)

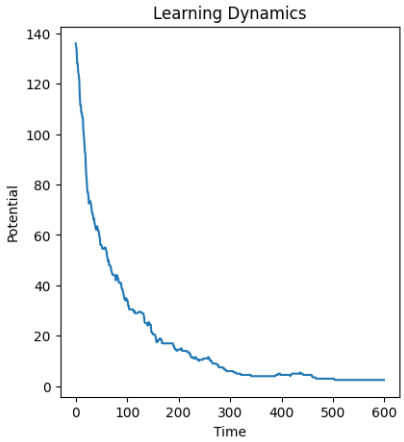


(a) Potential over time

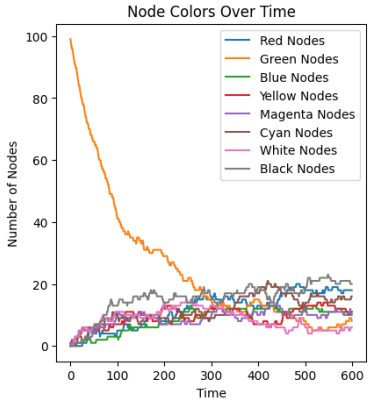


(b) Evolution over time of coloring

Figure 20: Plot showing the results of the simulation (150 iterations)



(a) Potential over time

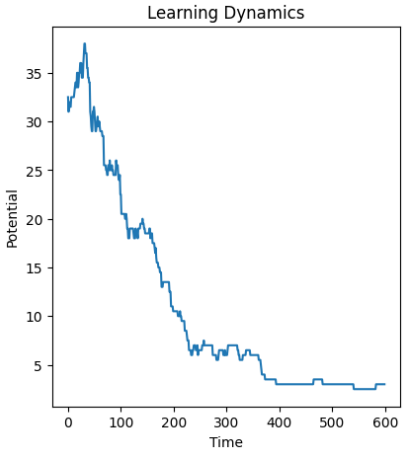


(b) Evolution over time of coloring

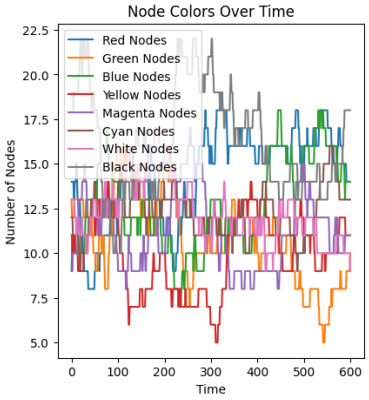
Figure 21: Plot showing the results of the simulation (600 iterations)

Random initial assignment color-node. In the second case, we modified the initial condition: there was a random assignment color-node. The results (represented in figure 22, after 600 iterations) are quite similar to the previous case. The only difference is in transitory.

Also in this scenario, increasing the number of iteration did not improve the results.



(a) Potential over time



(b) Evolution over time of coloring

Figure 22: Plot showing the results of the simulation with Random initial assignment color-node

2.C Coloring with different values of noise

In this section we ran the algorithm of the exercise 2.B with some different value of noise:

1. decreasing noise (with small and large value);
2. increasing noise (with different value than the exercise 2.B);
3. constant noise (with small and large value).

In table 1 we report the η functions used ($noise = \frac{1}{\eta}$. In table 2 we report our results in the various cases tested.

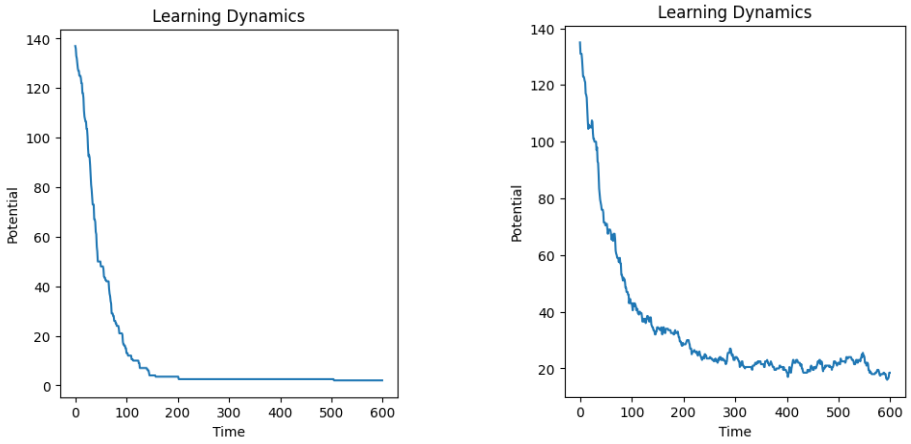
Table 1: Function of noise

-	SMALL	MEDIUM	LARGE
DECREASING	$\eta = \frac{t}{10}$	$\eta = \frac{t}{100}$	$\eta = \frac{t}{1000}$
INCREASING	$\eta = \frac{1000}{t}$	$\eta = \frac{100}{t}$	$\eta = \frac{1}{t}$
CONSTANT	$\eta = 100$	$\eta = 1$	$\eta = 0.01$

Table 2: Final potential in various cases of noise

-	SMALL	MEDIUM	LARGE
DECREASING	2	3	18
INCREASING	10	24	38
CONSTANT	3	12	33

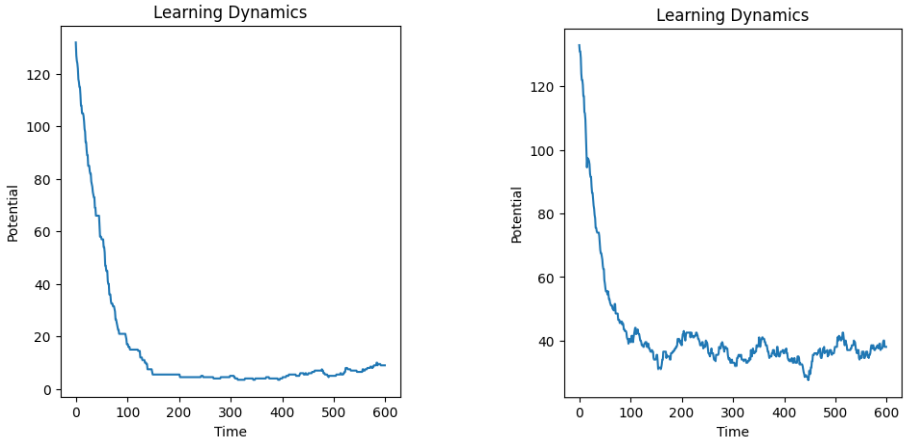
In figures 23, 24 and 25 there are potential values over time in various cases of noise. We noticed is that in the large noise cases (figures 23b, 24b and 25b), the potential varied inconsistently and abruptly.



(a) Small noise

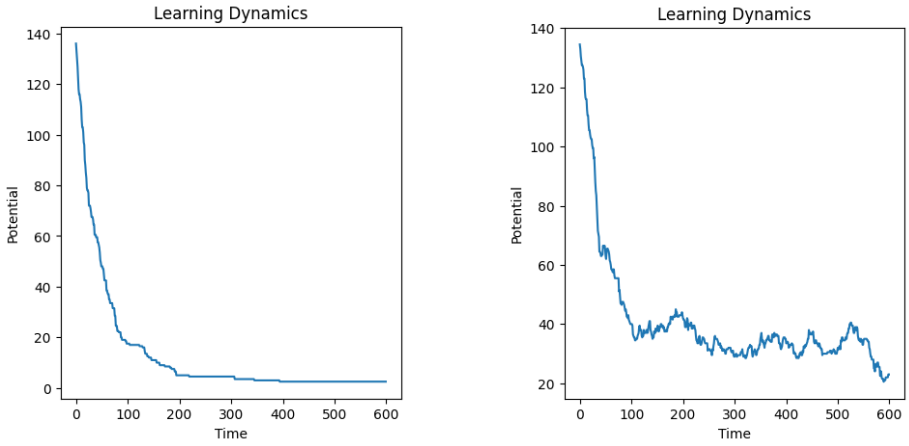
(b) Large noise

Figure 23: Potential over time in case of decreasing noise



(a) Small noise (b) Large noise

Figure 24: Potential over time in case of increasing noise



(a) Small noise (b) Large noise

Figure 25: Potential over time in case of constant noise