Network Dynamics and Learning
Homework III

Francesco Grandi
Student id: 306272
Politecnico di Torino
Turin, Italy
s306272@studenti.polito.it
Exchanged ideas with Alessandro Casella (student id:306081) and Elisa Feraud (student id: 295573)
In this report, some results can change at every run of the related code. For this reason, not all the results have been reported.

1. Influenza H1N1 2009 Pandemic in Sweden

The first exercise asks to simulate pandemic scenarios, taking as example the pandemic in Sweden 2009.
Specifically four different scenarios will be considered:

1) simulation of a pandemic on a known graph without vaccination.
2) simulation of the disease propagation on a random graph without vaccination.
3) simulation of the disease propagation on a random graph with vaccination.
4) estimation of the network-structure characteristics and disease-dynamics parameters for the pandemic in Sweden during the fall of 2009.

1.1. Preliminary parts

1.1.1. Epidemic on a known graph

In this part it is asked to simulate an epidemic on a symmetric k-regular undirected graph. Specifically, each vertex is connected to the k = 4 nodes whose index is closest to their own modulo n (where n is the cardinality of the node set V).
For this point, a graph of 500 nodes is considered.
The disease propagation model considered is a discrete-time simplified version of the SIR epidemic model. In this model, each node can assume a state X_i(t) at each time t. In particular, the possible values of a state X_i(t) are:

- S: the node is susceptible.
- I: the node is infected.
- R: the node is recovered.

In the code, these three distinct states are converted into a numerical format:

- S=0
- I=1
- R=2

Thus, two different probabilities are considered:

- $\beta \in [0, 1]$: it defines the probability that a susceptible node becomes infected during one time step (if it is connected by a link with an infected vertex). Furthermore, the probability that a node does not become infected (even if it is connected to an infected one) is (1 - β) during one time step.
- $\rho \in [0, 1]$: it is the probability for an infected node to recover during one time step.

1.1.1.1 Problem 1.1

Firstly, it is required to define a symmetric k-regular graph G = (V,E) and to simulate an epidemic on this known graph. The values of the parameters used in this point are reported in table 1.
Therefore, it is asked to plot the average of people that become infected each week and the average total number of susceptible, infected, and recovered nodes at each week.

TABLE 1: Problem 1.1

	Problem 1.1
β	0.3
ρ	0.7
number of nodes (n)	500
k	4
number of weeks	15
initial infected	10
time steps (N)	100

To build graph G, it is defined a method, get_known_graph(number_nodes,k). Given the number of nodes and the parameter k, an undirected cycle graph is defined. This type of graph (undirected cycle graph) is chosen in order to satisfy the requests of the exercise, which takes into account a symmetric k-regular graph G (as written before).
Then, two main methods are defined:

- epidemics_without_vacc(graph, beta, to, number_weeks, initial_infected): to simulate the epidemic.

Figure 1

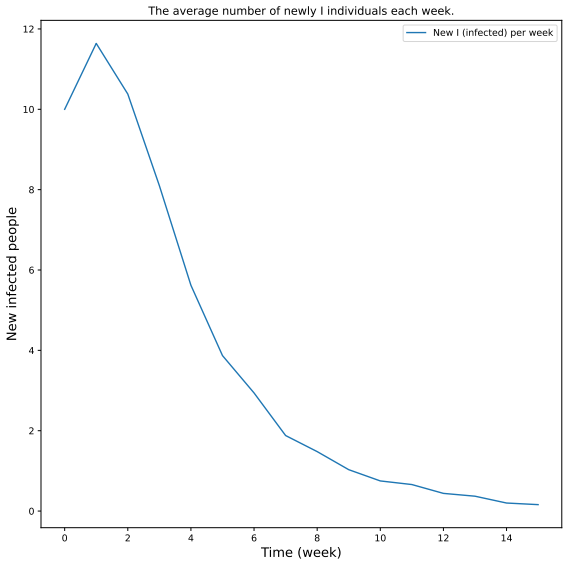
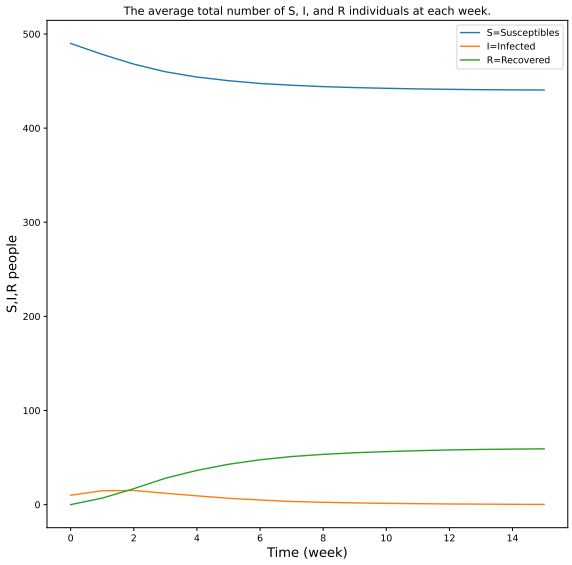


Figure 2



- getAvgsAndPlotKnownGraph(number_nodes, k, beta, ro, number_weeks, initial_infected, N): to compute the averages requested and to plot them.

The first function, given a graph and the parameters specified, computes a simulation of the epidemic. Firstly, it is defined the initialization of a matrix (SIR) in which are stored, for each week, the states (with the values indicated in the previous section) of the nodes. The matrix SIR, will be used to define the average total number of susceptible, infected, and recovered individuals at each week.
Then, the following steps are iterated in a cycle for each week.
The number of infected people of the past week is computed and used to find the nodes linked to infected vertices. Thus, for each susceptible node the probability to become infected in the current week is computed. The same thing is done in order to find which infected nodes can be considered recovered. In this way, the state of each node in the current week is computed.
At the end of the cycle, after the simulation for each week, the matrix SIR and array new_I, that contains the number of newly infected individuals during each week, are returned.
This function is used in the method getAvgsAndPlotKnownGraph(number_nodes,k,beta,ro,number_weeks, initial_infected,N), that calls it iteratively for each time step. Specifically, for each time step epidemics_without_vacc() is called and the values of each simulation are stored.
At the end of cycle, the average number of newly infected individuals each week and average total number of susceptible, infected, and recovered individuals at each week are computed, by dividing the results obtained in the cycle for the number of time steps.
Thus, the two means are plotted in two graphics, reported in Figure 1 and 2.

By observing the results obtained, it is possible to notice that the epidemic does not spread much among the individuals, since the number of new weekly infected nodes tends to decrease by the first weeks. Thus, the peak is reached in the first weeks. This behaviour could be explained by the fact that β is less than ρ, therefore, it is more likely to recover than to be infected.

1.1.2. Generate a random graph

At this point, it is required to generate a random graph according to the preferential attachment model, in order to obtain a randomly generated graph with average degree close to k.
Starting with an initial complete graph G1 (with k+1 nodes), at every time t $\geq$ 2, a new graph G_t is defined, by adding a new node to $G_{(t-1)}$ and connect it to some nodes already existing in $G_{(t-1)}$. This connection is defined by some stochastic rule. Specifically, according to the rule, at each time step (t $\geq$ 2) every vertex added at time t will have a degree c=k/2. The connections are established based on some probability. In particular, the probability is proportional to the current degree of the node it is connecting to.

1.1.2.1 Problem 1.2

The purpose of this point is to define a random graph with at least 900 nodes and average degree k $\in \mathbb{R}^+$, according to the rule explained in the previous section. The initial graph has k+1 nodes.

To do this, the method get_random_graph(number_nodes, k) is defines. With this function it is possible to create a random graph, given the number of nodes and the average degree k. In particular, by alternating between adding $\lfloor k/2 \rfloor$ and $\lceil k/2 \rceil$ links when adding a new node to the graph, it is possible to obtain an average degree equals to k also in the case in which k is odd.
Thus, the number of nodes is defined in a random way:
int(np.random.rand()*rng.integers(900)+900).
The average degree k is defined as:
rng.integers(1,high=10).
In this way it is possible to randomize the number of nodes and k, in such a way that the constraint (number of nodes $\geq$900) is respected.

1.2. Simulate a pandemic without vaccination

In this part it is required to simulate an epidemic on the random graph generated in Problem 1.2. Specifically, the epidemic is simulated by using the model defined in Problem 1.1.

1.2.1. Problem 2

The values of the parameters used in this point are reported in table 2.
In order to simulate the epidemic scenario described, three main methods are used:

- get_random_graph(number_nodes,k): to create a random graph.
- epidemics_without_vacc(graph,beta, ro,number_weeks, initial_infected): to simulate the epidemic.
- getAvgsAndPlotRandomGraph(number_nodes,k,beta, ro,number_weeks, initial_infected,N): to compute the averages requested and to plot them.

Specifically, the first two functions are the ones described in the previous sections. The new method is the third one which computes the average of the newly infected individuals each week and the average of the total number of susceptible, infected, and recovered individuals at each week. For each simulation, it creates a random graph using the function get_random_graph() and use it to simulate the epidemic with the method epidemics_without_vacc(). Then, at the end of the cycle, the two averages are computed and plotted in the two graphs reported in Figure 3 and 4.

By observing the reported results in Figure 3 and 4, it is possible to notice that the disease propagation has a different behaviour with respect the one observed in Problem 1.1. In this case, the peak is reached after than in the previous problem and the epidemic spreads more. Since the parameters are the same, the main reason is due to the different considered graph. The complete graph has an average degree equals to 6, thus there are more connections among nodes than the case in which the degree is equal to 4.

TABLE 2: Problem 2

	Problem 2
β	0.3
ρ	0.7
number of nodes (n)	500
k	6
number of weeks	15
initial infected	10
time steps (N)	100

1.3. Simulate a pandemic with vaccination

At this point it is required to simulate an epidemic as done before, but considering also the vaccinations, used to slow down the disease propagation.
In this scenario, during each week, some individuals will be vaccinated in such a way to not be able to be infected in the future. Specifically, individuals vaccinated in a certain week, are not more susceptible from the same week of the vaccination. The percentages of people that has to be vaccinated by each week are reported in the following array:
Vacc(t) = [0, 5, 15, 25, 35, 45, 55, 60, 60, 60, 60, 60, 60, 60, 60].
Respectively, the percentages of people to vaccinate in each week are:

[0, 5, 10, 10, 10, 10, 10, 5, 0, 0, 0, 0, 0, 0, 0]

1.3.1. Problem 3

The values of the parameters used in this problem are reported in table 3. To simulate the epidemic with vaccinations, firstly it is necessary to define a random graph, using the method get_random_graph(number_nodes,k), introduced in Problem 1.2.
Then, two new methods are defined:

- epidemics_with_vacc(graph,beta, ro,number_weeks, initial_infected, vacc): to simulate the epidemic with vaccinations.
- getAvgsAndPlotRandomGraphVacc(number_nodes, k, beta, ro, number_weeks, initial_infected, N, vacc1, vacc2, vacc): to compute the averages requested and to plot them.

With the first method, the disease propagation is simulated, considering the case in which vaccinations are used to mitigate the epidemic. This method is similar to epidemics_without_vacc() with the difference that, a new array is defined, NV, related to the non vaccinated nodes. After a proper initialization of NV, the SIR matrix, the past and current week vectors and new_I, which contains the number of newly infected individuals during each week, the states of the nodes are computed, for each week. Thus, the following steps are iterated in a cycle for each week.
Firslty, it is defined the number of nodes to vaccinate in the current week, according to the percentages reported before. In this way it is possible to randomly obtain the nodes to vaccinate. The array NV is updated: vaccinated nodes assume value equal to 0. The same thing is done for the

Figure 3

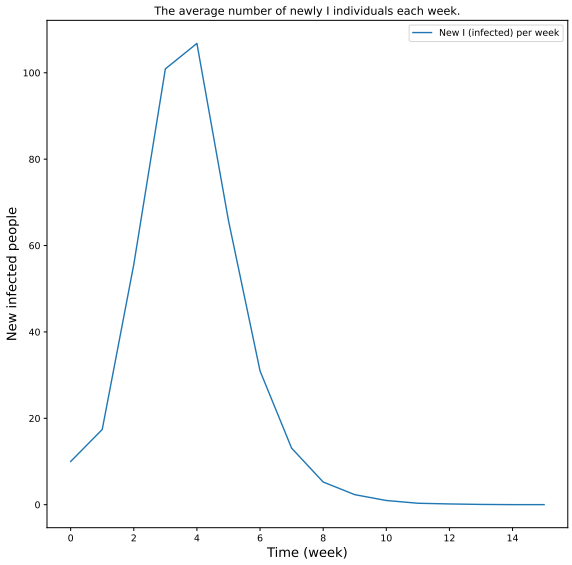


Figure 4

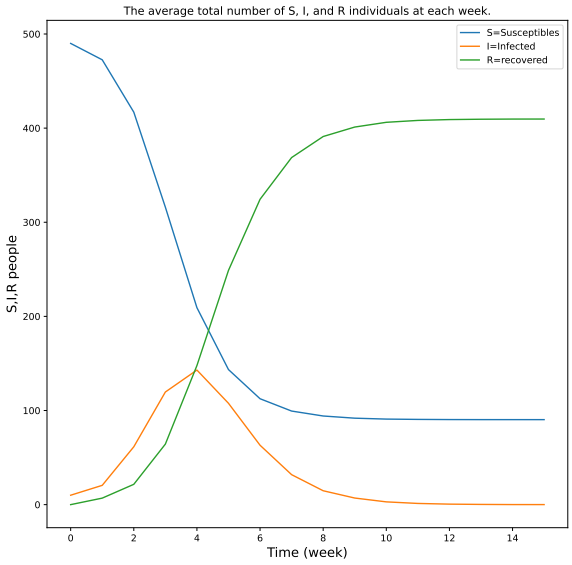


Figure 5

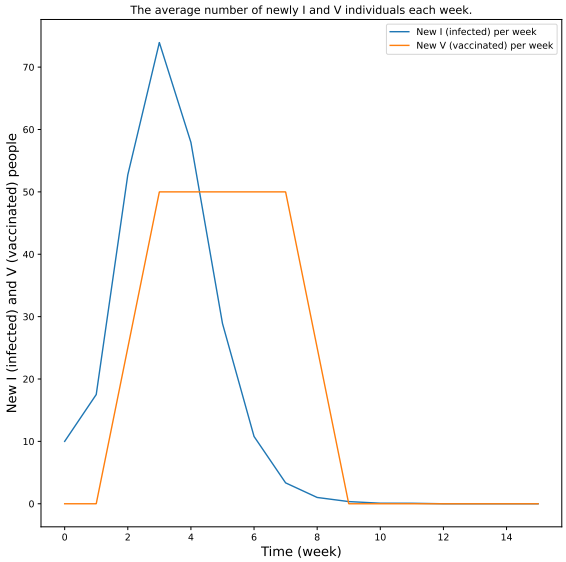
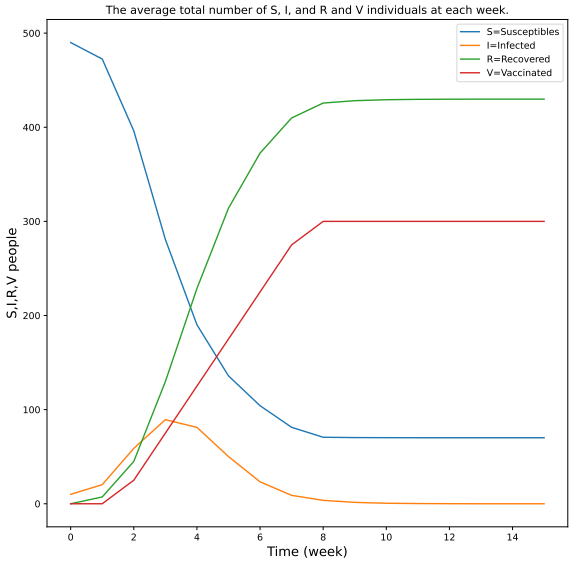


Figure 6



current and past week arrays, in which the new vaccinated nodes assumed status ”Recovered”, thus equal to 2.
Then, the number of infected people of the past week is computed and used to select the nodes linked to infected vertices. Thus, for each susceptible vertex is computed the probability to become infected in the current week. Similarly, the infected nodes that can be considered recovered are found. In this way, the state of each node in the current week is computed.
At the end of the cycle, after the simulation for each week, the matrix SIR and and array new_I, that contains the number of newly infected individuals during each week, are returned.
This method is used in getAvgsAndPlotRandomGraph-

Vacc(). This function computes the averages requested by the problem, and plot them in two distinct graphics. The main difference with the graphics plotted in the previous points, is that now also the vaccinated nodes are plotted. Specifically, in the 5 are reported the new vaccinated people for each week, while in the 6 the total number of vaccinated node for each week is shown. By observing the results, it is possible to notice how the disease propagation changes behaviour if vaccinations are distributed to individuals. The main effect of the vaccinations is to slow down the epidemic which grows less than in the case where no preventive measures are taken.

TABLE 3: Problem 3

	Problem 3
β	0.3
ρ	0.7
number of nodes (n)	500
k	6
number of weeks	15
initial infected	10
time steps (N)	100

1.4. The H1N1 pandemic in Sweden 2009

All the previous parts are now used to simulate the real-case of H1N1 pandemic happened in Sweden in 2009. The population size of Sweden is scaled down to 934 people, due to computational power. The pandemic simulation covers the most critical period in terms of new infected and vaccinated equal to 15 weeks, during which the cumulative fraction of population vaccinated evolves as below:

$$vacc = [5, 9, 16, 24, 32, 40, 47, 54, 59, 60, 60, 60, 60, 60, 60]$$

and the vector of newly infected people is:

$$real\_infected = [1, 1, 3, 5, 9, 17, 32, 32, 17, 5, 2, 1, 0, 0, 0, 0]$$

In order to match the best set of parameters (k, β, ρ) that fits the real pandemic, it is defined an algorithm which, starting from user-defined initial conditions (k_0, β_0, ρ_0) and $(\Delta k, \Delta\beta, \Delta\rho)$, does a gradient-based search with the scope of minimizing the RMSE indicator, computed considering the number of infected individuals each week in the simulation and in the real pandemic:

$$RMSE = \sqrt{\frac{1}{15}\sum_{t=1}^{15}(real\_infected(t) - tot\_new\_I(t))^2} \quad (1)$$

where $tot\_new\_I(t)$ is the number of infected people each week of the simulation and $real\_infected(t)$ is the average number of newly infected people each week in the real pandemic.
Starting from a random guess of the initial parameters, the algorithm works for each configuration in the parameter spaces $k \in \{k_0 - \Delta k,\ k_0,\ k_0 + \Delta k\}$, $\beta \in \{\beta_0 - \Delta\beta,\ \beta_0,\ \beta_0 + \Delta\beta\}$, and $\rho \in \{\rho_0 - \Delta\rho,\ \rho_0,\ \rho_0 + \Delta\rho\}$ in the following way:

1) creates a random preferential attachment graph with 934 nodes and degree = k;
2) simulates the pandemic with the same function with vaccinations, used in the previous problem;
3) computes RMSE as in (1);
4) updates the best parameter spaces if the RMSE found is the minimum one among all the epochs;
5) if a new best set of parameters cannot be found in an epoch, then $\Delta\beta$ and $\Delta\rho$ are divided by 2 and Δk is approximated to the nearest integer of its value divided by 2.

The execution of points 1) and 2) is repeated $n\_sim = 50$ times. The algorithm runs the five steps for 20 epochs, unless the new found variation of parameters k and ρ is very small ($\Delta < 0.003$): in this case the algorithm stops ahead of schedule. More simulations with different parameters spaces have been tried, here are reported two of the ones that give the lowest RMSE values with the relative best parameter space:

- Initial conditions:
  $\{k_0 = 10,\ \beta_0 = 0.5,\ \rho_0 = 0.5\}$,
  $\{\Delta k = 9,\ \Delta\beta = 0.4,\ \Delta\rho = 0.4\}$.
  Best parameters:
  $\{k = 18,\ \beta = 0.094,\ \rho = 0.644\}$.
- Initial conditions:
  $\{k_0 = 8,\ \beta_0 = 0.2,\ \rho_0 = 0.6\}$,
  $\{\Delta k = 2,\ \Delta\beta = 0.2,\ \Delta\rho = 0.2\}$.
  Best parameters:
  $\{k = 8,\ \beta = 0.188,\ \rho = 0.588\}$.

Since the lowest RMSE is obtained from a fixed number of simulations, it can vary from time to time, then as a consequence the best parameters can change too. In order to plot a reliable epidemics trend with these parameters, it is run a simulation made by another algorithm with the same 1), 2) and 3) points of before, but now with the scope of catching the minimum RMSE by keeping fixed the best parameters previously found. Once the average number of newly infected related to the minimum RMSE has been stored, it is compared to the number of real infected in Figures 7-9, while in Figures 8-10 they are shown susceptible, infected and recovered people.

In Figures 7-8 it is shown the overall best estimate that coincides with the lowest RMSE found in the analysis, while in Figures 9-10 it is presented another parameters space close to the optimum. In both cases, the algorithm proves to be more effective mainly at the start of the epidemics, contrary to the peak of the curve which is always underestimated. For the last weeks, the predictions' trend returns to be nearer to the real one, but in general the temporal evolution of newly infected seems to be excessively anticipated in the rise and excessively postponed in the descent, without reaching the real peak.

1.5. Challenge

The challenge consists in trying a different random graph that does not use preferential attachment to represent the

Figure 7: Number of newly infected: predicted vs true

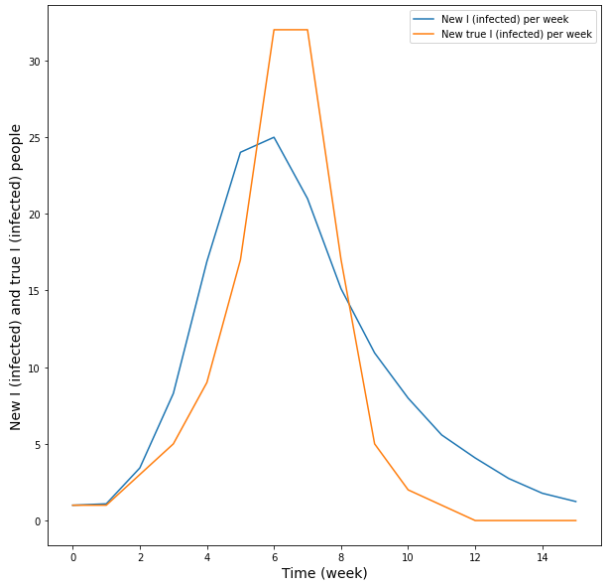
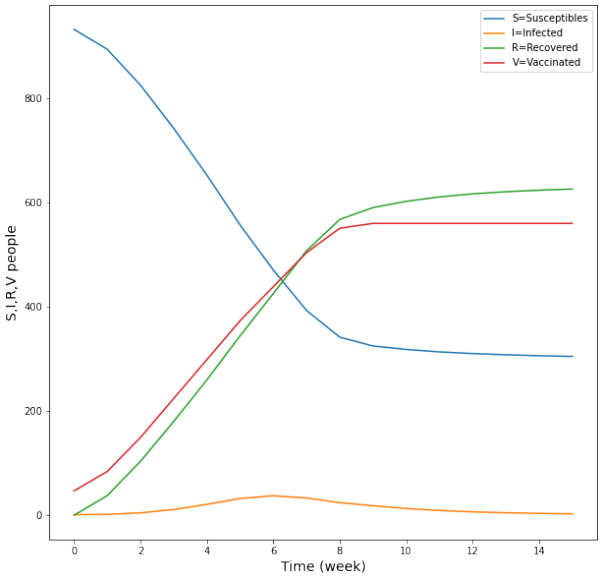


Figure 8: S,I,R,V people evolution



$RMSE \approx 4.778$: Average evolution on a randomly generated preferential attachment graph with 934 nodes and parameters space $\{k = 18,\ \beta = 0.094,\ \rho = 0.644\}$

Figure 9: Number of newly infected: predicted vs true

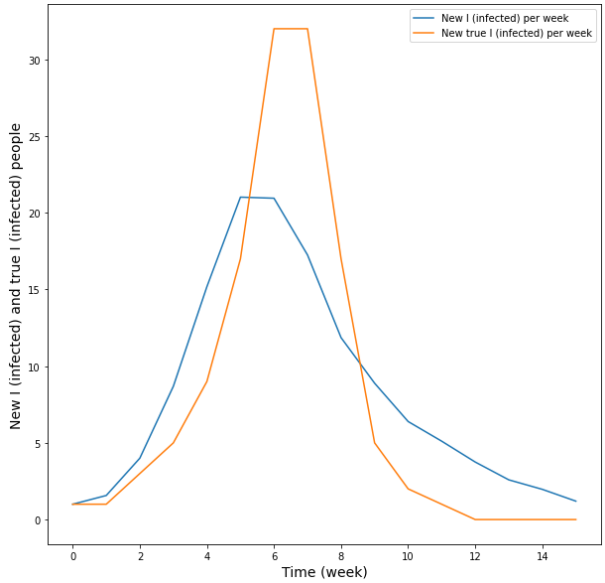
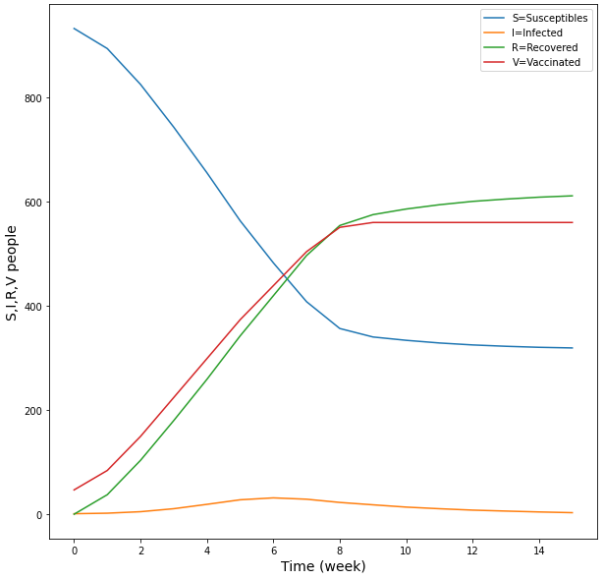


Figure 10: S,I,R,V people evolution



$RMSE \approx 5.809$: Average evolution on a randomly generated preferential attachment graph with 934 nodes and parameters space $\{k = 8,\ \beta = 0.188,\ \rho = 0.588\}$

network for the pandemic, with the aim of obtaining a better estimation. In this analysis, the Small World model plays the role of random graph for representing the structure of the Swedish population. In real life, epidemics' spread is proportional to the probability that two individuals interact, not only directly but also through the contact with a third individual who share the contact with both two and thus acts as a bridge. In terms of network theory, this concept can be translated by taking into consideration that even though nodes i and j are not directly connected to each other but they both directly reach another node k, then there exists a probability that the state of i is influenced by the state of j and vice-versa. For this reason, Small World models assign a certain probability that a new link is shared between a couple of nodes $(i,\ j)$. While the preferential attachment graph gives the advantage of establishing the exact degree

of the nodes a priori, it cannot handle this kind of contact explained above.
The new Small World random graph is created in dependence of the parameters k and p, respectively defined as the value to which the average degree of the graph is close and the probability of adding a link between a random couple of nodes. Firstly, it is generated a regular graph with 934 nodes in the same way as in the known graph of Section 1, then a new link is added between every other possible couple of nodes with probability equal to p. From a practical point of view, the initial k-regular graph represents the close interactions that happen between people who keep in touch very frequently, while the randomly added links represent the random encounters that can happen sometimes between people who do not have near contacts in common.
The experiments of the previous section are run again, but now generating a Small World random graph, instead of the preferential attachment one, and considering the new parameter p in the parameters search. Due to computational time, the number of simulations is now restricted to $n\_sim = 30$, while the number of epochs as well as the other settings are kept as before. If a certain epoch does not return a better parameters space, then Δp is halved, as it happens for the other parameters. The initial values assigned to the parameters at the simulation's starting equal the ones used in the first simulation of the previous section, but the best parameter space is now different, influenced by the probability of adding links in the random graph:

- Initial conditions:
  $\{k_0 = 10,\ \beta_0 = 0.5,\ \rho_0 = 0.5,\ p = 0.02\}$,
  $\{\Delta k = 9,\ \Delta\beta = 0.4,\ \Delta\rho = 0.4,\ \Delta p = 0.01\}$.
- Best parameters:
  $\{k = 5,\ \beta = 0.103,\ \rho = 0.978,\ p = 0.012\}$.

In Figures 13-14 it is shown the overall best estimate that coincides with the lowest $RMSE \approx 5.545$ found in the analysis. In Figures 11-12 it is presented another parameters space close to the optimum, which highlights the ability of Small World graph to almost reach the real peak of the curve, even though the value of $RMSE \approx 6.912$ is higher than before. Some significant considerations can be made by comparing the new results to the epidemics simulated with the preferential attachment graph as the model to represent the Swedish population. First of all, the best ρ value found is very different from the previous section; now, being very close to 1 means there is a very high probability that an infected individual will recover during only one time step. This result can be explained by the two random graphs' different ways of working according to the parameters variations. The different trend of new recovered people with respect to the previous section can be seen in Figures 12-14, where the number of recovered individuals grows with more strength and in fact overcomes the curve of vaccinated people before (6th week vs 8th week), in addition to obviously terminate with more recovered (about 650 vs 600). Moreover, the best epidemics evolution of new infected people is reached in Figure 11 visually: it seems there is no time lag anymore, that was the lack seen in all the simulations with preferential attachment graph. Indeed, the predicted maximum number of new infected individuals is reached during the 6th week and keep approximately constant until the 7th week, after that it starts decreasing as in the ground truth. Apart from this, there are too many new infected individuals registered before and especially after the peak: this fact brings a crucial contribute to increase the RMSE.

2. Coloring

This problem deals with distributed learning in potential games applied in graph coloring, which consists in assigning a color to each node of an undirected graph, with the constraint that the color assigned to a node is different from the colors of all its neighbors. Two different situations are taken into account in the following parts:

1) a simple line graph;
2) a more complex network that represents WiFi channels among different routers.

2.1. Line graph

At initialization, each node in the graph of Figure 15 is red. During the simulation, the nodes can update their color multiple times, so the set of their possible states is $\mathcal{C} = \{red, green\}$. At every discrete time instant t, one node is randomly chosen to change its color, according to all the transition probabilities computed in the following way:

$$P(X_i(t+1) = a \mid X(t), I(t) = i) = \frac{e^{-\eta(t)\sum_j u_i(a, X_j(t))}}{\sum_{s\in\mathcal{C}} e^{-\eta(t)\sum_j u_i(s, X_j(t))}} \tag{2}$$

where $X_i(t)$ is the state of node i at time t, $I(t)$ is the node chosen to change its color, $\eta(t)$ is the inverse of the noise and $u_i(s, X_j(t))$ is the utility function for node i to change its color in s with respect to the state of its neighbors $X_j(t)$, defined as follows:

$$u_i(s, X_j(t)) = W_{ij}\phi(s, X_j(t)) \tag{3}$$

where W_{ij} is the entry corresponding to the nodes i and j of the weight matrix and $\phi(s, X_j(t))$ is the cost function given by

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

The task is to simulate the learning dynamics described above, taking care of the potential function (5), which indicates how close to a solution the algorithm is.

$$U(t) = \frac{1}{2}\sum_{i,j} W_{ij}\phi(X_i(t), X_j(t)) \tag{5}$$

A zero-potential $U(t) = 0$ implies the existence of a solution because of the absence of conflicting nodes. The interpretation of (2) is that the probability with which a new action a

Figure 11: Number of newly infected: predicted vs true

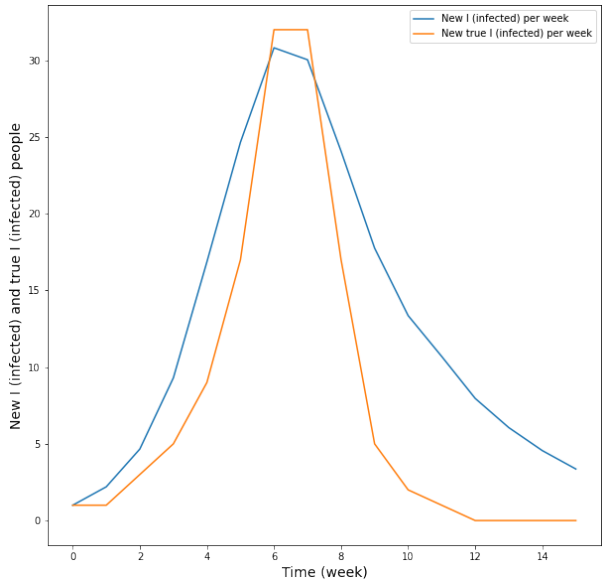
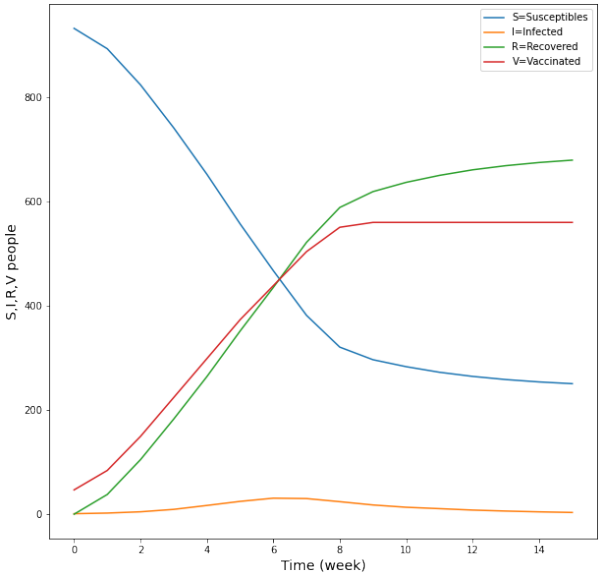


Figure 12: S,I,R,V people evolution



$RMSE \approx 6.912$: Average evolution on a randomly generated Small World graph with 934 nodes and parameters space $\{k = 6,\ \beta = 0.2,\ \rho = 1,\ p = 0.005\}$

Figure 13: Number of newly infected: predicted vs true

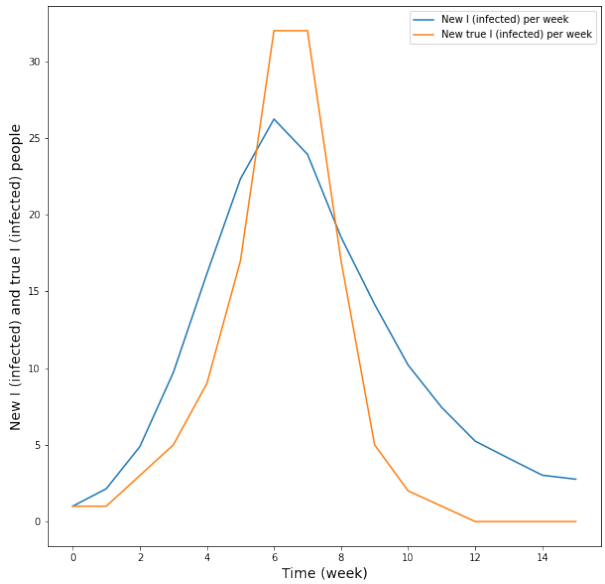
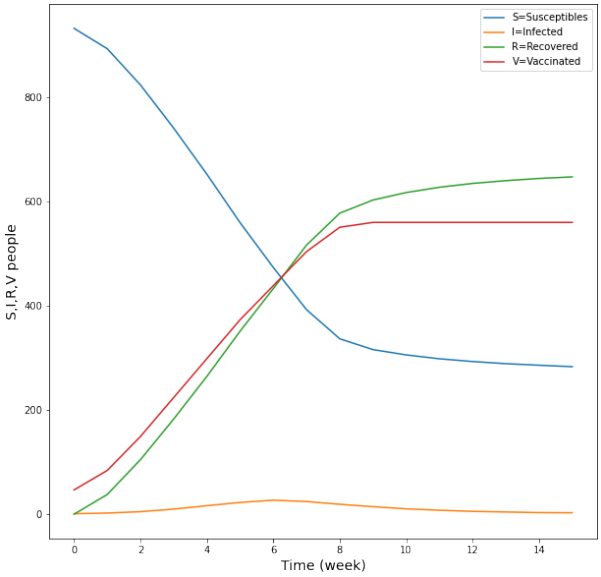


Figure 14: S,I,R,V people evolution



$RMSE \approx 5.545$: Average evolution on a randomly generated Small World graph with 934 nodes and parameters space $\{k = 5,\ \beta = 0.103,\ \rho = 0.978,\ p = 0.011875\}$

is adopted by node i is increasing with its associated utility $\sum_j u_i(a, X_j(t))$. By changing the initial value assigned to the inverse of the noise, it is possible to compare the number of steps needed to reach a null potential: the higher is the denominator assigned to $\eta(t)$, the more are the number of steps required to have $U(t) = 0$. This follows from the interpretation of the existing dependence of the transition probability on $\eta(t)$:

- as $\eta(t) \to 0$ this dependence vanishes and (2) converges to a uniform probability distribution on the action set C_i;
- as $\eta(t) \to \infty$ this dependence becomes stronger and (2) converges to a uniform probability on the best-response set, so that the noisy best response

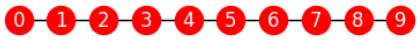


Figure 15: Simple line graph with 10 nodes

dynamics reduces to the best response dynamics.

As expected, by simulating the coloring dynamics for 100 time instants, the convergence to best response is never reached in Figure 16 for $\eta = t/100$, differently from what happens in Figure 17 for $\eta = t/10$, where instead the potential manages to settle on best response convergence. When $\eta = t/100$, the process of potential's lowering takes more time and in the first 100 steps it is only capable to reach a potential equal to 1. To better understand the coloring process, another simulation is run with 40 time-steps and $\eta = t/2$, whose Figure 24 represents the evolution of the different configurations. The first null potential is registered at the 23rd step, after which there is a continuous alternation of potential equal to 1 and 0, as it happens at a certain point in Figure 17 but at different times. This is due to the fact that unless $\eta(t)$ is very big, there is always a probability to reach another destination. In general, this phenomenon is very useful as it guarantees that the function does not settle on a local optimum, allowing the algorithm to experiment more configuration. On the contrary, if $\eta(t) \to \infty$, it is possible that the algorithm gets stuck on a local optimum since no other configurations can be considered. In fact, as soon as another state with an higher configuration value is reached, the transition probability for the algorithm to go back to the previous low potential state is very high. The process happens at the cost of having the possibility to find an even lower potential reachable only through the transition between higher potential states.

2.2. General example graph

Similarly to the first coloring problem, this exercise requires assigning eight colors to an indirect graph $\mathcal{G}$ of 100 nodes. The graph is obtained from a node-node adjacency matrix $\mathcal{A} \in \{0, +1\}$ of dimensions $\mathcal{V} \times \mathcal{V}$: if two nodes i, j are connected, then a 1 can be found at the i-th column and j-th row of the matrix. In addition, it is given a list of coordinates for each vertex in order to plot $\mathcal{G}$ in an uncluttered way. Each color in the set $C = \{red, green, blue, yellow, magenta, cyan, white, black\}$ represents a different frequency. The following cost function is implemented:

$$\phi(s, X_j(t)) = \begin{cases} 2 & \text{if } X_j(t) = s \\ 1 & \text{if } |X_j(t) - s| = 1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

whose logical interpretation can be explained as follows. If two neighbor nodes have the same frequency, a 2 points penalty is assigned; instead, if they have similar frequency-that is frequencies having only one frequency between them, a penalty of one is assigned; no penalty is present for any other case.

At every discrete time instant t, one node is randomly chosen to change its color, according to all the transition probabilities computed in Equation (2). In addition, the utility and the potential are computed respectively as in Equations (3) and (5). In order to manage a large graph as $\mathcal{G}$, a splitting operation is computed. 20 small graphs of dimensions ranging from 1 to 15 vertexes are obtained by exploiting the fact that $\mathcal{G}$ is made of various graphs without any link connecting them. Therefore, for the purpose of the simulation, they can be considered individually as there cannot be interference between them. To simulate the algorithm, a new cost matrix of dimensions $C \times C$ is defined. For every small graph, a similar algorithm to the first exercise is implemented. In order to compute the total potential, all the small graph potentials are summed. In Figures 18-19-20, three potential functions are shown. They were obtained by simulating the dynamics for three different periods of time but with the same $\eta = t/100$. The graphs show a similar trend: the longer the simulation, the lower potential is obtained. Starting from a maximum potential of 213, in Figure 18 a potential equal to 32 is reached at the 96th iteration; in Figure 19 a potential equal to 11 is reached at the 229th iteration; in Figure 20 a potential equal to 6 is reached at the 450th iteration. In Figure 18 it is possible to note an exploratory behavior of the algorithm shown by a higher variance of the potential function. This is due to the fact that η is linearly dependent on time and very small at early iterations. This translates into very noisy best response dynamics that change states nearly randomly, gradually becoming less noisy as time passes and η increases. On the contrary, the variance for big η is low, since the average time between transitions increases. In fact, it is unlikely that a vertex modifies its action from a Nash equilibrium to achieve a lower cost.

The distribution of colors corresponding to the smallest reachable potential function is shown in Figure 25. The minimum potential reached is 4 and it is usually obtained after the 500th iteration. The potential function value is justified by the presence of the two small graphs shown in Figures 22-23. The first graph has a component of five vertexes fully connected. Therefore, to not have any interference, it is easy to see that 10 frequencies are needed. Instead, for the 6 vertexes fully connected graph in Figure 23, 12 frequencies are needed to avoid any interference. Since we have only 8 frequencies, the cost of the graphs in Figures 22-23 are respectively 1 and 3 points.

2.3. Evaluation for different choices of η(t)

At this point, it is required to evaluate what happens for different choices of η(t). Firstly, constant values of η(t) are evaluated. Then, two increasing functions η(t) are considered.

In order to obtain a better analysis, each η(t) is evaluated

Figure 16: $\eta(t) = t/100$

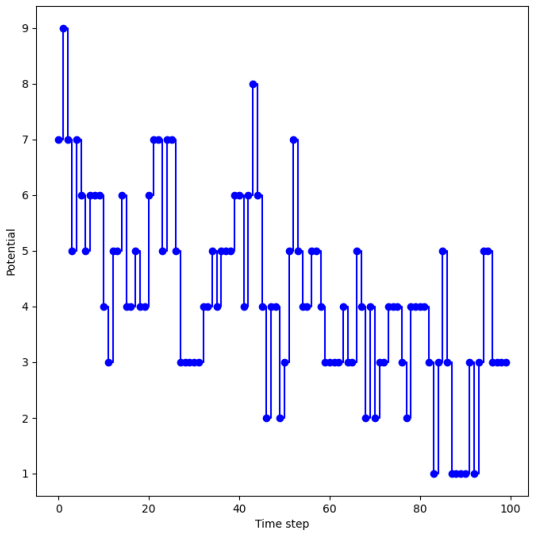
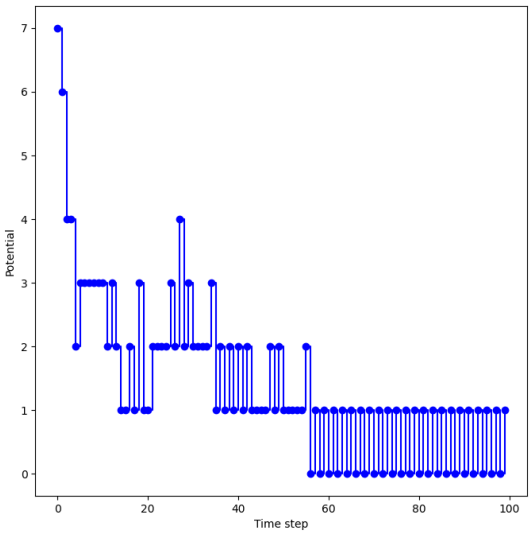


Figure 17: $\eta(t) = t/10$



Potential functions generated by 100 steps of coloring algorithm for different $\eta(t)$ functions

Figure 18: n_steps=100

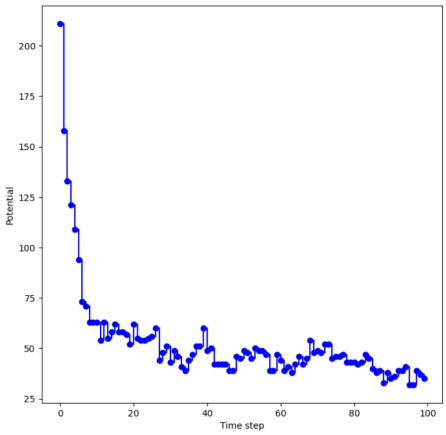


Figure 19: n_steps=250

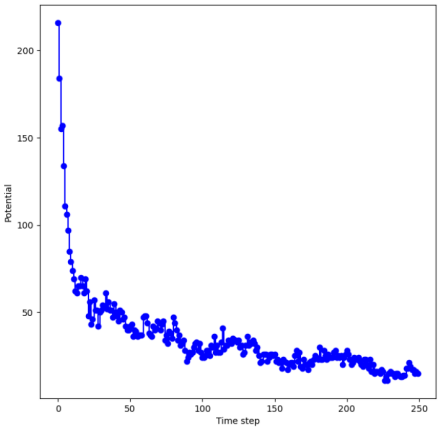
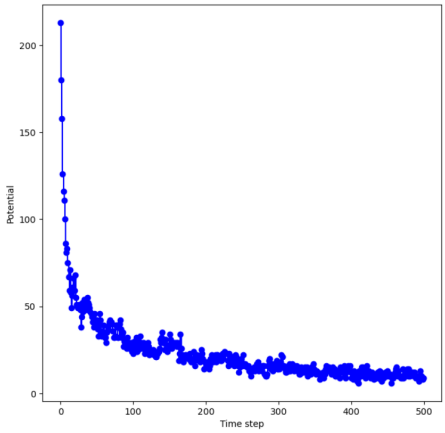


Figure 20: n_steps=500



Potential functions generated by different simulation period and same $\eta(t)$ function

TABLE 4: Problem 2.c: number of steps required to reach the minimum value of the potential function.

η(t)	n_steps				
	30	50	100	250	500
10000	24	37	44	70	103
1000	17	36	45	133	46
100	23	25	68	64	282
10	28	48	93	247	283
e^t	22	48	67	108	65
10t	24	40	84	221	97

TABLE 5: Problem 2.c: minimum reached values of the potential function.

η(t)	n_steps				
	30	50	100	250	500
10000	24.0	16.0	10.0	8.0	10.0
1000	28.0	18.0	8.0	8.0	8.0
100	14.0	13.0	4.0	5.0	4.0
10	21.0	12.0	7.0	5.0	4.0
e^t	7.0	7.0	5.0	5.0	5.0
10t	17.0	8.0	7.0	4.0	4.0

with various time steps. Specifically, the numbers of time steps considered are 30, 50, 100, 250, and 500.
In Tables 4, 5 are reported the results obtained for each scenario. It is important to underline the fact that these values are random, thus they might change if the code is run again. Still, they are good exemplifications of the dynamics performances as the values obtained do not vary a lot every time the code is compiled.
From Tables 4, 5 it is possible to notice that, in the case in which η(t) assumes constant value, it reaches the minimum

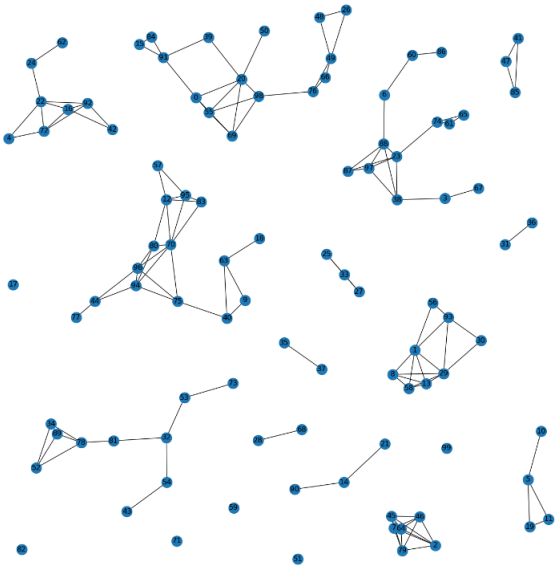


Figure 21: Simple graph with 100 nodes

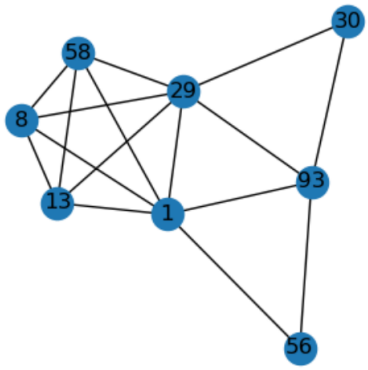


Figure 22: Graph with a 5-vertexes-fully-connected component

observed value of the potential function (4.0) when it is smaller (in the cases reported, for η(t)=100 and 10) in less than 500-time units. By observing the result obtained for constant values of η(t), it is possible to notice that for smaller η(t), the potential function is better minimized. For what concerns the exponential function, it works quite well, since the potential function reaches a good minimum point (5.0) in less than 200 steps (in particular, 67,108, and 65 steps). Finally, for the function 10t, the minimum value reached is 4. It is possible to observe that in both functions (e^t and 10t), the longer the simulation is, then the lower potential is reached, as said in the previous point. This is due to the fact that η(t) is dependent on time and the two considered functions are increasing, allowing the dynamics to be less noisy when an optimum is near. In fact, in the beginning, the dynamics work well if noisy since they do not get stuck in a sub-optimal configuration. This does not happen in the cases in which η(t) assumes constant values as 10000, 1000, 100, and 10. In fact, even though these dynamics risk getting stuck in a not-optimized configuration, they performed well. In particular, the small values of η(t) (10 and 100) are a good trade-off between noisy and not noisy dynamics as they are reaching the minimum potential function values. In general, the best η(t) seems to be 10t and 100, because they reached the minimum potential function value (4.0) respectively in 97 steps and 68 steps.
In Appendix the graphics for $\eta(t)$=100 and $\eta(t)$=10t are reported.

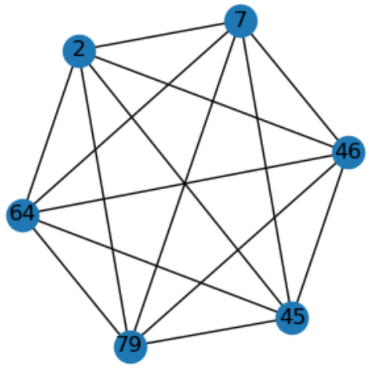


Figure 23: Fully connected graph with 6 vertexes

Appendix

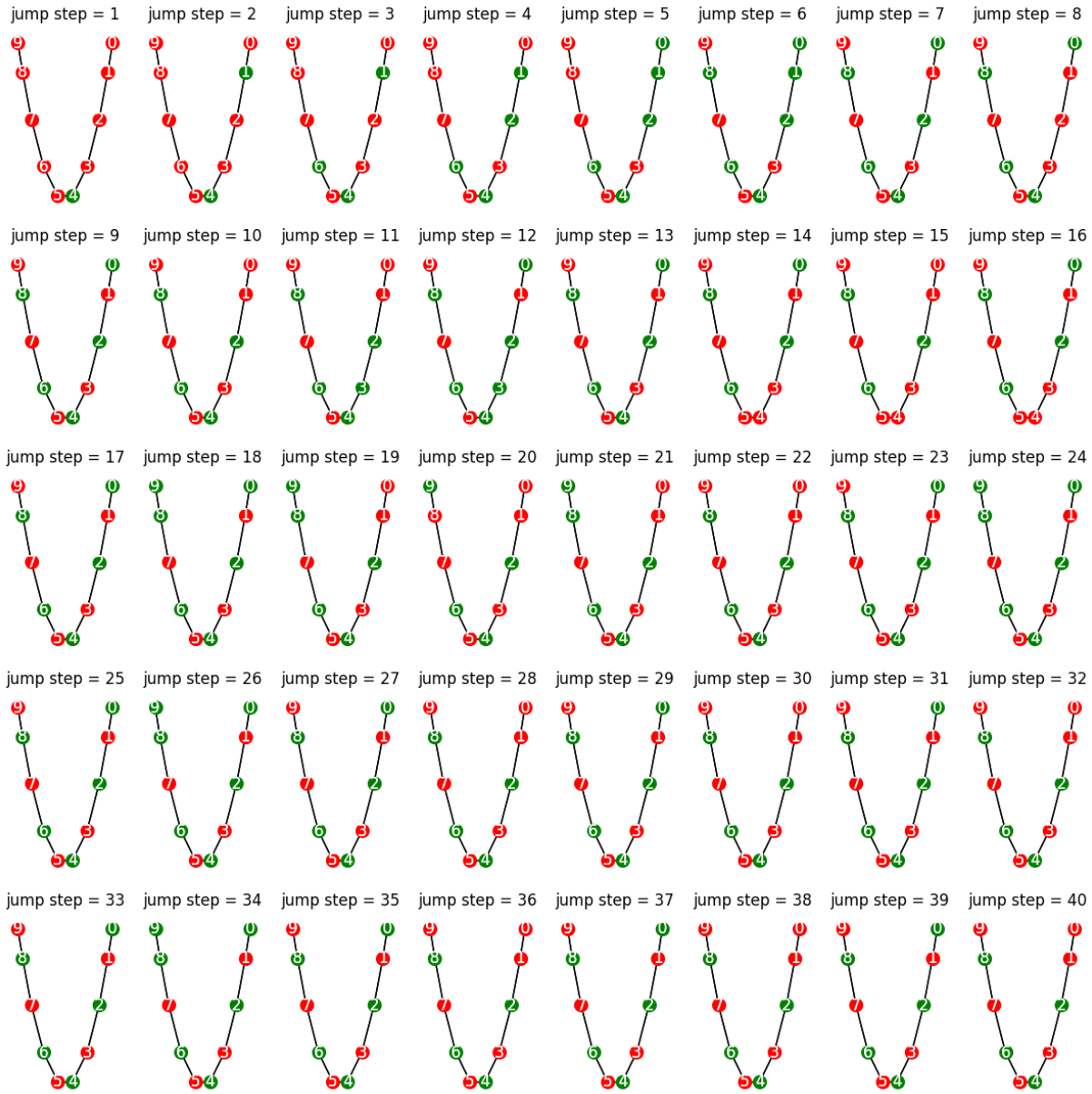


Figure 24: Learning dynamics simulated by coloring algorithm: $\eta(t) = t/2$

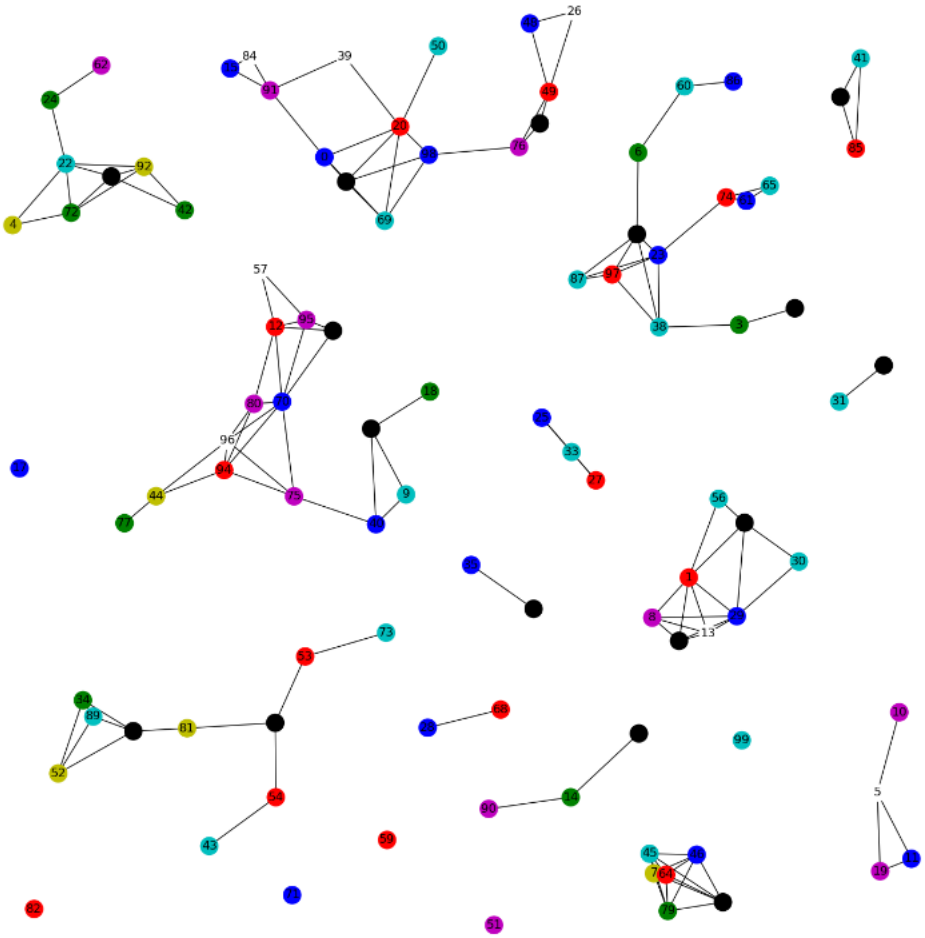
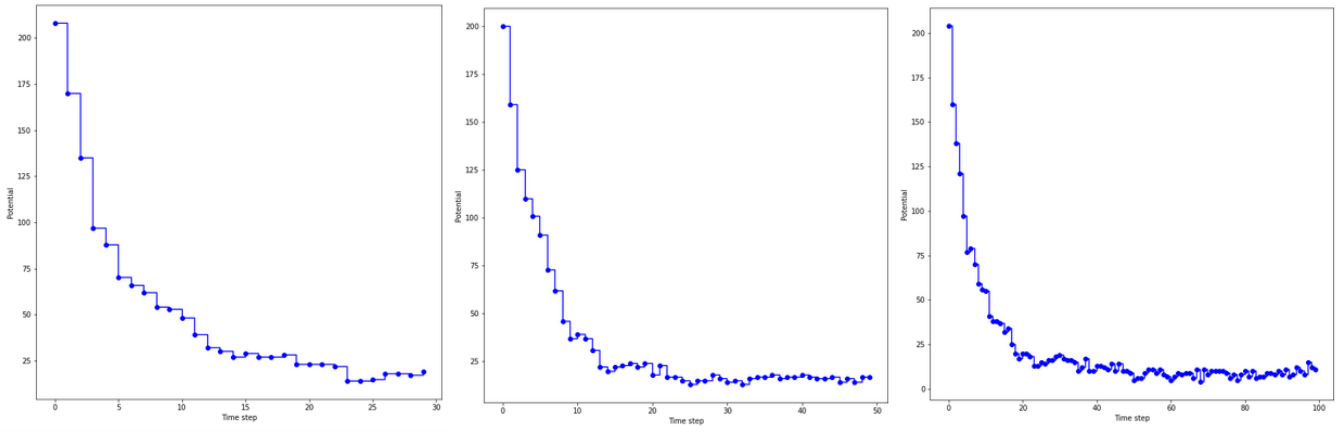


Figure 25: Final coloring of the graph

Figure 26: n_steps=30

Figure 27: n_steps=50

Figure 28: n_steps=100



Potential functions generated by different simulation period and $\eta(t)$=100

Figure 29: n_steps=250

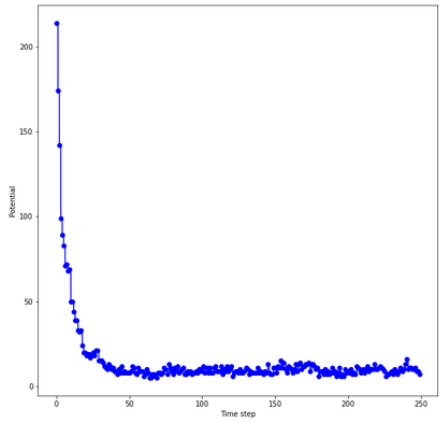
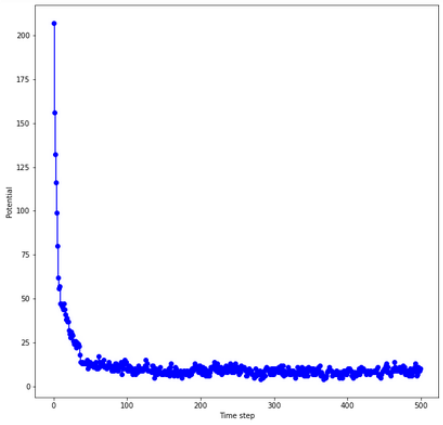


Figure 30: n_steps=500



Potential functions generated by different simulation period and $\eta(t)$=100

Figure 31: n_steps=30

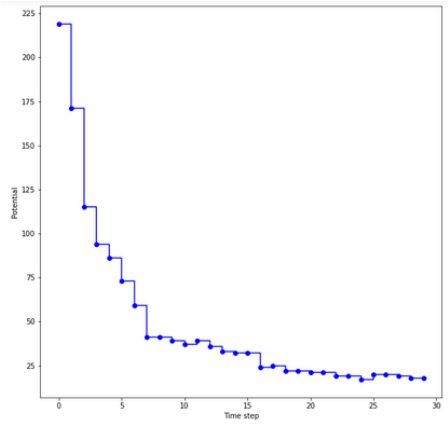


Figure 32: n_steps=50

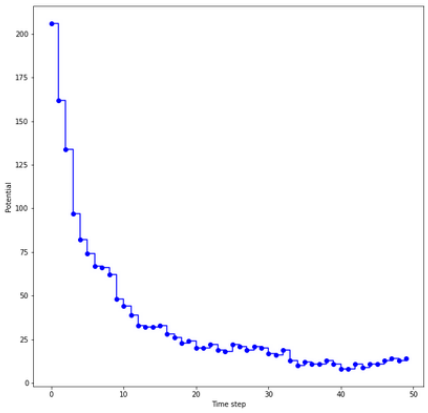
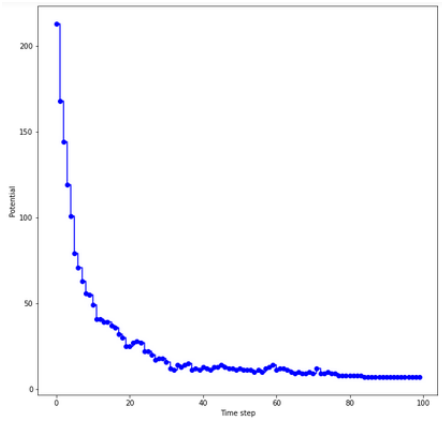


Figure 33: n_steps=100



Potential functions generated by different simulation periods and $\eta(t)$=10t

Figure 34: n_steps=250

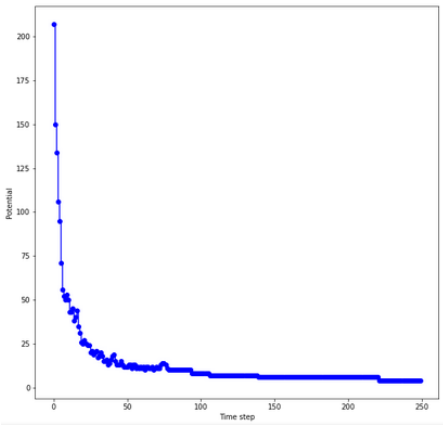
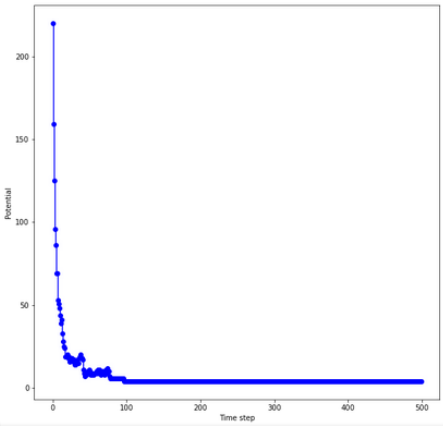


Figure 35: n_steps=500



Potential functions generated by different simulation periods and $\eta(t)$=10t