Network Dynamics and Learning
Homework III

Alessandro Casella
Student id: s306081
Politecnico di Torino
s306081@studenti.polito.it

Exchanged ideas with Elisa Feraud (student id: s295573) and Francesco Grandi (student id: s306272)
Note: since some results change for different runs of the code, not all the numerical values are displayed in this report.

1. Influenza H1N1 2009 Pandemic in Sweden

The first exercise asks to simulate pandemic scenarios, taking the pandemic in Sweden in 2009 as an example. Specifically, four different scenarios are 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's structure characteristics and dynamics parameters of disease for the pandemic in Sweden during the fall of 2009.

1.1. Preliminary parts

1.1.1. Epidemic on a known graph - Problem 1.1

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 $\mathcal{V}$). The epidemic is simulated on a graph of $n = 500$ nodes. For simplicity, the disease propagation model is a discrete-time version of the SIR epidemic model where each node can assume a state $X_i(t)$ at each time t. The possible values of a state $X_i(t)$ are listed here with the correspondent numerical encoding:

- $S = 0$: the node is susceptible;
- $I = 1$: the node is infected;
- $R = 2$: the node is recovered.

Two different values of probability model the epidemic spread:

- $\beta \in [0, 1]$: it defines the probability that a susceptible node becomes infected in a one-time step (if a link connects it with an infected vertex). Thus, the probability for a node not to become infected (even if it is connected to an infected one) is $(1 - \beta)^m$ in a one-time step, where m is the number of infected neighbours for the considered node;
- $\rho \in [0, 1]$: it is the probability for an infected node to recover in a one-time step.

Firstly, it is required to define a symmetric k-regular graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and to simulate an epidemic on this known graph. Secondly, it is asked to plot the average of people that become infected each week and the total number of susceptible, infected, and recovered nodes each week on average. The parameters used are reported in Table 1 with their values:

TABLE 1: Epidemic on a known graph

Parameters	Values
β	0.3
ρ	0.7
n	500
k	4
number_weeks	15
initial_infected	10
N	100

where *number_weeks* is the duration of the pandemic in weeks, *initial_infected* is the number of initial infected people and N are the number of different simulations made, over which the requested averages are computed.
Three methods are defined:

- *get_known_graph(n, k)*: given the number of nodes n and the number of directed links k for each node, it builds an undirected cyclic graph $\mathcal{G}$ on which the pandemic is spread through the following function;
- *epidemics_without_vacc($\mathcal{G}$, β, ρ, number_weeks, initial_infected)*: given the previously generated graph and the parameters specified in Table 1, it computes a simulation of the epidemic;
- *getAvgsAndPlotKnownGraph(n, k, β, ρ, number_weeks, initial_infected, N)*: for each of the 100 simulations, it creates the known graph using the *get_known_graph(n, k)* function and uses it to simulate the epidemic with the *epidemics_without_vacc(...)* method. At each step of

Figure 1: Average number of new I individuals each week

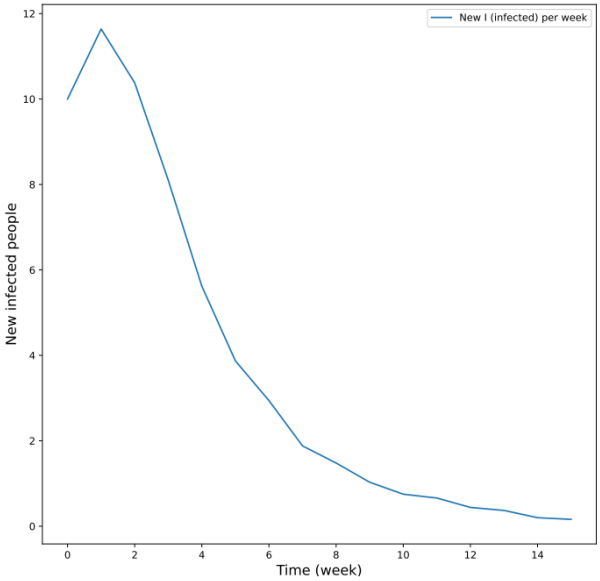
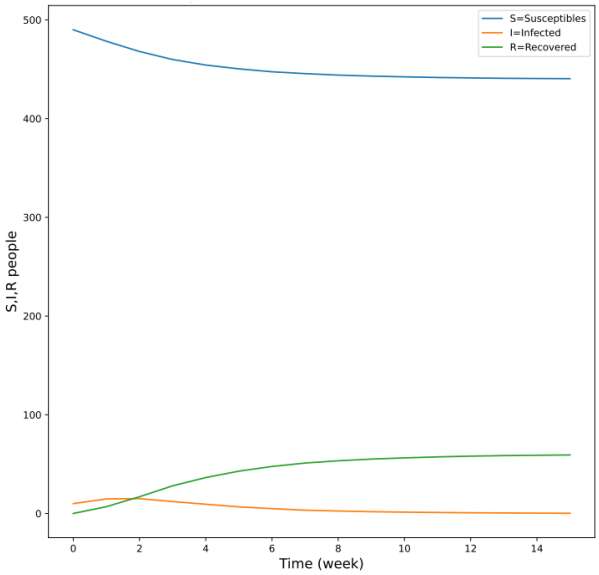


Figure 2: Average number of S,I,R people each week



Epidemic without vaccination on a known graph with parameters defined in Table 1

the loop, the matrix *SIR* is updated with the current states of the nodes (according to the previously defined encoding) for each week, as well as the number of newly infected people. Once all the simulations are done, it computes the total number of susceptible, infected, and recovered individuals each week on average. Eventually, the function plots the Figures 1-2.

The crux of the algorithm for epidemic simulation is in the second method of the list, where the following steps are repeated for each week: the people who become infected during the past week are used to record the nodes linked to, which become classified as *susceptible*. Thus, for each susceptible node, it is computed the probability to become *infected* in the current week. The same thing is done in order to find which infected nodes can be considered *recovered*.
By observing the results obtained in Figures 1-2, 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 $\beta < \rho$, which means it is more likely to recover than to be infected.

1.1.2. Generate a random graph - Problem 1.2

It is required to generate a random graph with n nodes and an average degree close to k, according to the preferential attachment model. At the starting time $t = 1$, a complete graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ where $|\mathcal{V}_1| = k+1$ is defined. At every $t \geq 2$, a new graph $\mathcal{G}_t$ is obtained by connecting a new node to some already existing nodes in $\mathcal{G}_{t-1}$. Specifically, every vertex added at time $t \geq 2$ has degree $c = k/2$. The connections between the newly added node and the $k/2$ already existing ones are based on some probability, proportional to the degree distribution of all the nodes in $\mathcal{G}_{t-1}$.
The purpose is to generate a random graph, given the number of nodes $|\mathcal{V}| \geq 900$ and average degree $k \in \mathbb{Z}^+$, according to the rule explained above. To do this, the method *get_random_graph(n, k)* is defined. 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 equal to k, even in the case in which k is odd. The number of nodes is defined in such a way that the constraint (≥ 900) is respected.

1.2. Simulate a pandemic without vaccination - Problem 2

The objective of this part is to simulate an epidemic on the random graph generated in Problem 1.2 with the pandemic model without vaccination defined in Problem 1.1. The values of the parameters used in this point are reported in Table 2.

TABLE 2: Epidemic on a random graph

Parameters	Values
β	0.3
ρ	0.7
n	500
k	6
number_weeks	15
initial_infected	10
N	100

In order to satisfy the requests, the same logic of the methods in Problem 1.1 is proposed again, but now taking into account the random graph, instead of the known one.

Figure 3: Average number of new I individuals each week

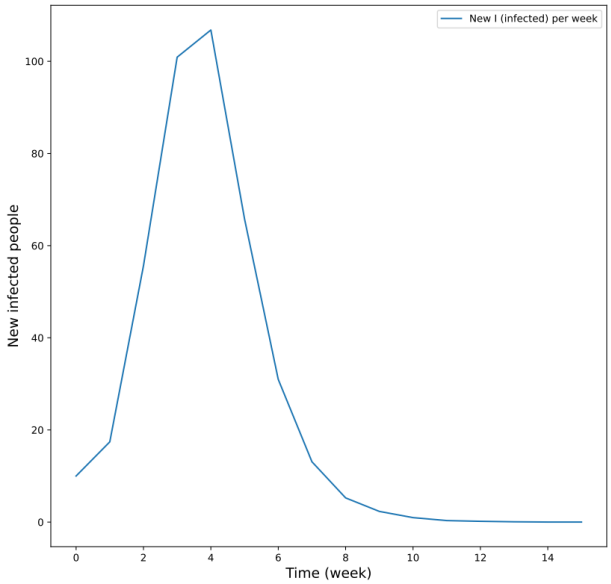
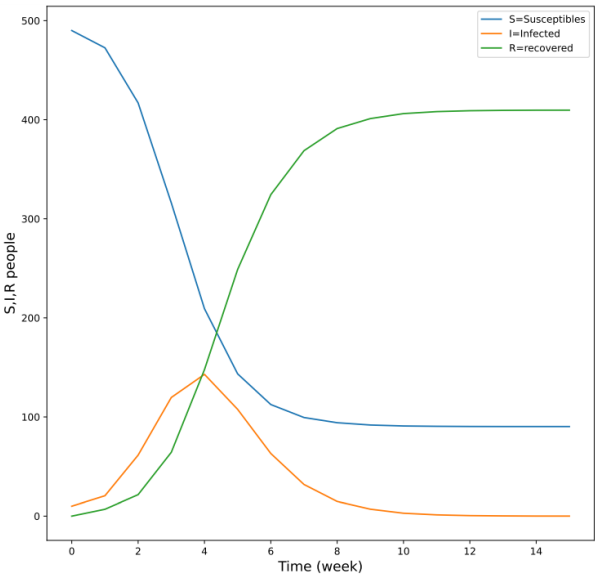


Figure 4: Average number of S,I,R people each week



Epidemic without vaccination on a random graph with parameters defined in Table 2

The two requested averages are computed and plotted in Figures 3-4. By observing them, it is possible to notice that the disease propagation has a different behaviour with respect to the one observed in Figures 1-2. In this case, the peak is reached later and the epidemic's spread is stronger. Since the parameters have the same value as in Table 1 except k, the main reason is due to the different structure of the network. Since the complete graph has a higher average degree ($k = 6$ vs $k = 4$), the nodes have more connections among themselves, so there is more probability that the epidemic spreads among the population.

1.3. Simulate a pandemic with vaccination - Problem 3

At this point, it is required to simulate an epidemic as done before, but also considering the vaccinations, which have the effect of slowing down the disease propagation. During each week, some individuals are vaccinated in such a way that they cannot be infected in the future. Specifically, individuals vaccinated in a certain week are no more susceptible starting from the same week of the vaccination. The cumulative percentages of people who have received the vaccine by each week are reported in the following array:

$$vacc_{\%} = [0, 5, 15, 25, 35, 45, 55, 60, 60, 60, 60, 60, 60, 60, 60]$$

from which the exact percentages of vaccinated people each week are intuitively obtained and stored in the $vacc$ array. Two other arrays are derived from it not to consider percentages, $vacc1$ and $vacc2$, which respectively define the real number of nodes which become infected each week and the real number of nodes which has become infected since the start of the epidemic. The parameters with relative values are exactly the same as in Table 2.
In order to simulate the vaccinations, it is necessary to define two new methods:

- *epidemics_with_vacc($\mathcal{G}$, β, ρ, number_weeks, initial_infected, vacc)*: given the previously generated graph through the *get_random_graph(n, k)* method and the parameters specified in Table 2, it computes a simulation of the epidemic;
- *getAvgsAndPlotRandomGraphVacc(n, k, β, ρ, number_weeks, initial_infected, N, vacc1, vacc2, vacc)*: for each of the 100 simulations, it creates the random graph by using the *get_random_graph(n, k)* function and uses it to simulate the epidemic with the *epidemics_with_vacc(...)* method. At each step of the loop, the matrix *SIR* is updated with the current states of the nodes (according to the previously defined encoding) for each week, as well as the number of newly infected people. Once all the simulations are done, it computes the total number of susceptible, infected and recovered individuals each week on average. Eventually, the function plots the Figures 5-6, also by taking into account the evolution of the established vaccinations.

Through 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 its equivalent *epidemics_without_vacc(...)*, but with the addition of a new binary array NV, related to the non-vaccinated nodes, which is initialized with all 1s (all the nodes are non-vaccinated). In this function, the following steps are repeated for the entire pandemic simulation: at each week t, a number of nodes equal to $vacc1(t)$ is randomly chosen

Figure 5: Average number of new I and V people each week

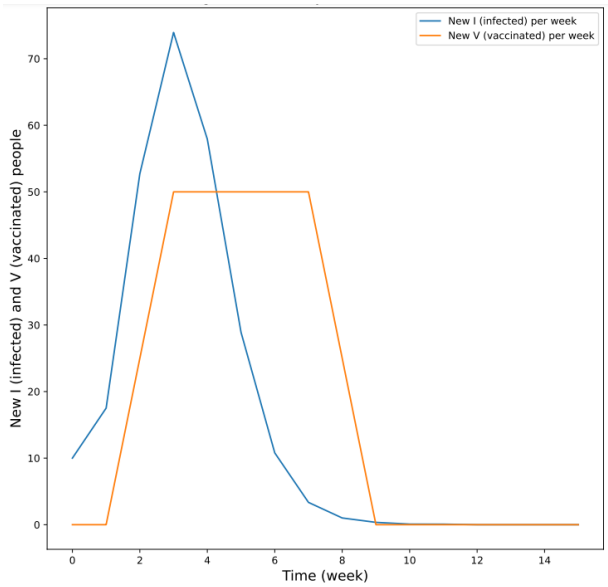
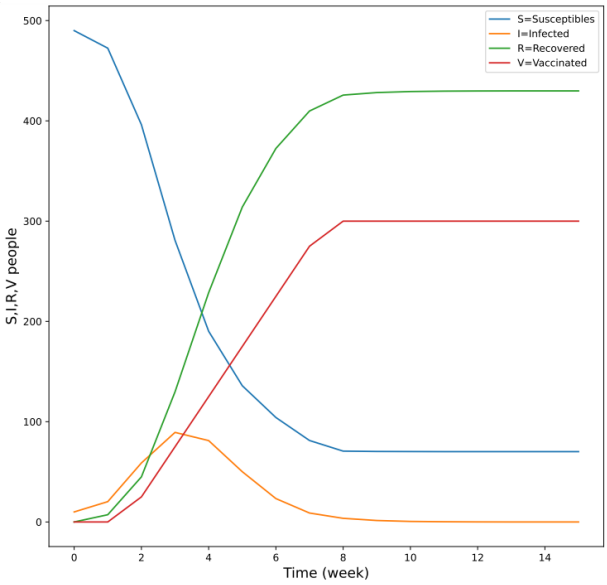


Figure 6: Average number of S,I,R,V people each week



Epidemic with vaccination on a random graph with parameters defined in Table 2

to be vaccinated. Thereafter, the NV array is updated by assigning a 0-value to the vaccinated nodes. The same update is done for the current and past week arrays, in which the new vaccinated nodes assume the *recovered* status.
The people who become infected during the past week are used to record the nodes linked to, which become classified as *susceptible*. Thus, for each susceptible node, it is computed the probability to become *infected* in the current week. The same thing is done in order to find which infected nodes can be considered *recovered*. This is done under the following hypothesis:

- people who are in susceptible, recovered or infected status can be vaccinated;
- whenever a person gets vaccinated, it is immediately considered as recovered.

As a consequence, among the recovered people there are also considered the vaccinated ones, so the number of recovered people is always greater or equal then the number of vaccinated ones, as shown in Figure 6. Specifically, in Figure 5 the newly vaccinated people are reported for each week in comparison to the newly infected people, while in Figure 6 the total number of nodes in all the possible states is shown for each week. By observing the results, it is possible to notice how the disease propagation changes behaviour if vaccinations are distributed to individuals: its main effect is to slow down the epidemic, which now grows slower than in the case where no preventive measures are taken.

1.4. The H1N1 pandemic in Sweden 2009 - Problem 4

All the previous parts are now used to simulate the real case of H1N1 pandemic that 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 newly infected and vaccinated equal to 15 weeks, during which the cumulative fraction of the population vaccinated evolves as below:

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

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) it creates a random preferential attachment graph with 934 nodes and degree = k;

Figure 7: Average number of new infected: predicted vs true

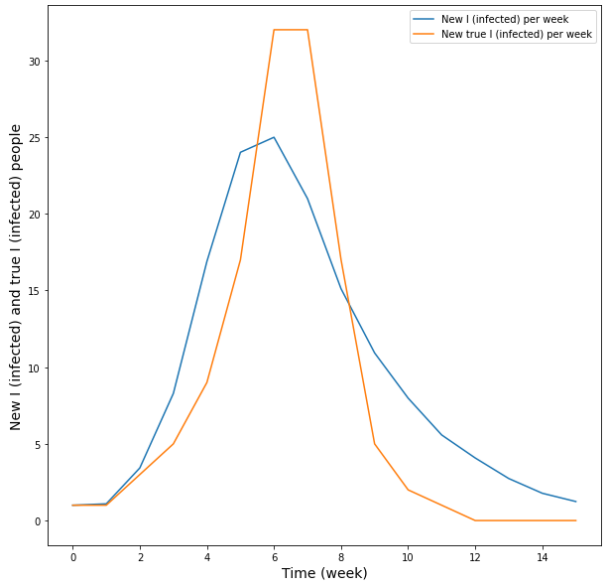
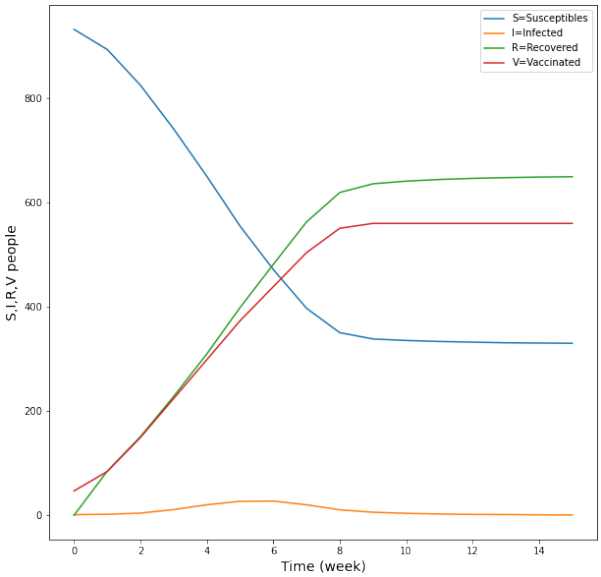


Figure 8: Average number of S,I,R,V people each week



$RMSE \approx 4.778$: 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: Average number of new infected: predicted vs true

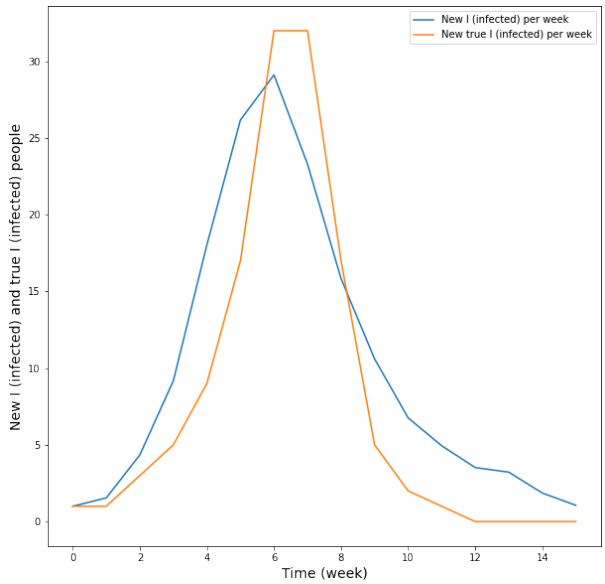
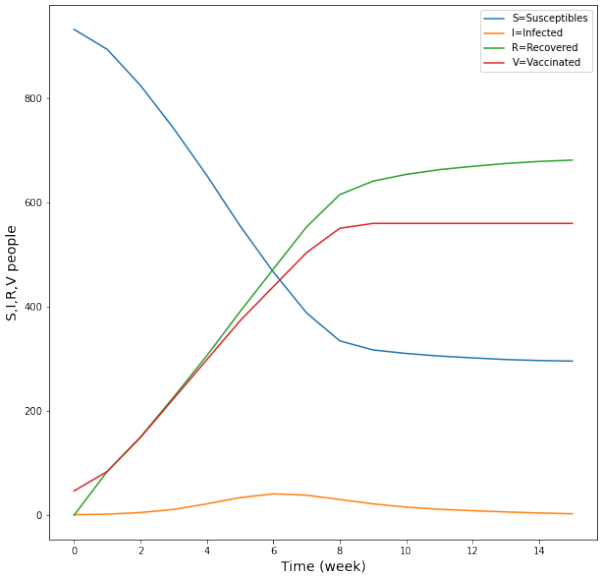


Figure 10: Average number of S,I,R,V people each week



$RMSE \approx 4.952$: Average evolution on a randomly generated preferential attachment graph with 934 nodes and parameters space $\{k = 8,\ \beta = 0.206,\ \rho = 0.6\}$

2) it simulates the pandemic with the same function with vaccinations, used in Problem 3;
3) it computes RMSE as in (1);
4) it 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 newly 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 parameters 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:
  $\{k = 8,\ \beta = 0.206,\ \rho = 0.6\}$.

Since the lowest RMSE is obtained from a fixed number of simulations, it can vary from time to time, and then as a consequence, the best parameters can change too. In order to plot a reliable epidemic trend with these parameters, is run a simulation made by another algorithm with the same 1), 2) and 3) points as 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 people related to the minimum RMSE has been stored, it is compared to the number of real infected individuals in Figures 7-9, while in Figures 8-10 they are shown susceptible, infected and recovered people for each week.
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 is presented other 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 tends to be 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 people seems to be excessively anticipated in the rise and excessively postponed in the descent, without reaching the real peak.

1.5. Challenge - Small World Graph

The challenge consists in trying a different random graph that does not use the preferential attachment to represent the network for the pandemic, with the aim of obtaining a better estimation. In this analysis, the Small World model plays the role of a random graph for representing the structure of the Swedish population. In real life, an epidemic's spread is proportional to the probability that two individuals interact, not only directly but also through contact with a third individual who shares 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 on 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 Problem 1.1, and then a new link is added between every other possible couple of nodes with a 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 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. Figures 11-12 present another parameters space close to the optimum, which highlights the ability of the 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 variations of the parameters. The different trend of newly 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 (3rd week vs 4th week), in addition to obviously terminate with more recovered (about 700 vs

Figure 11: Average number of new infected: predicted vs true

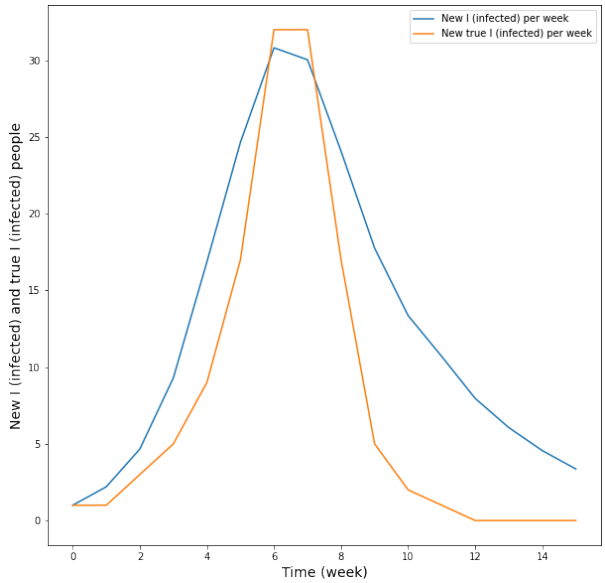
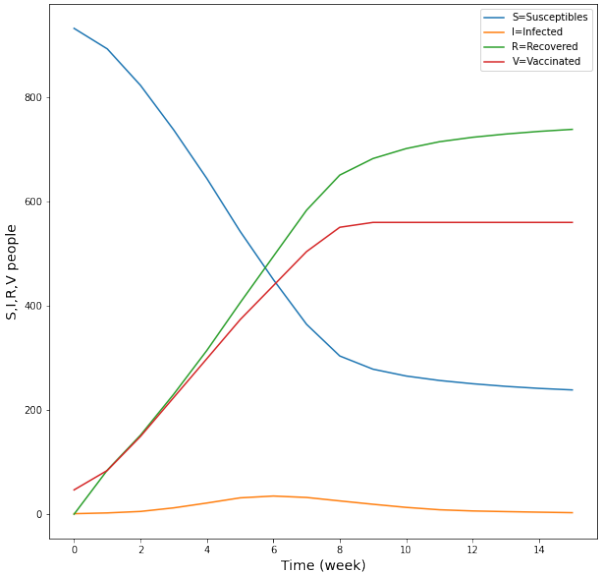


Figure 12: Average number of S,I,R,V people each week



$RMSE \approx 6.912$: 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: Average number of new infected: predicted vs true

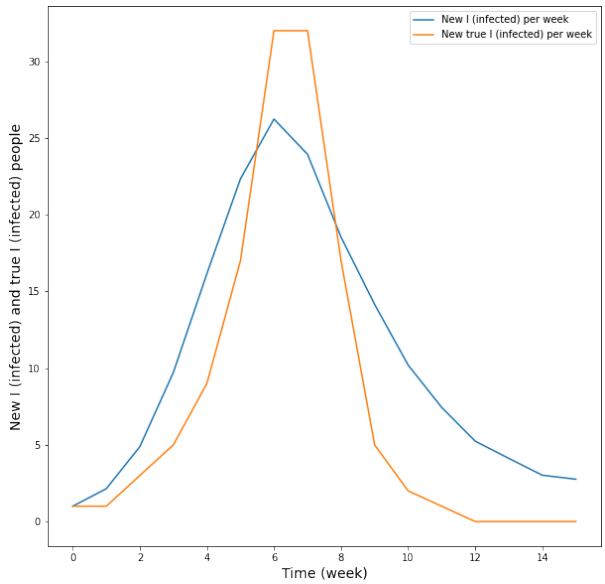
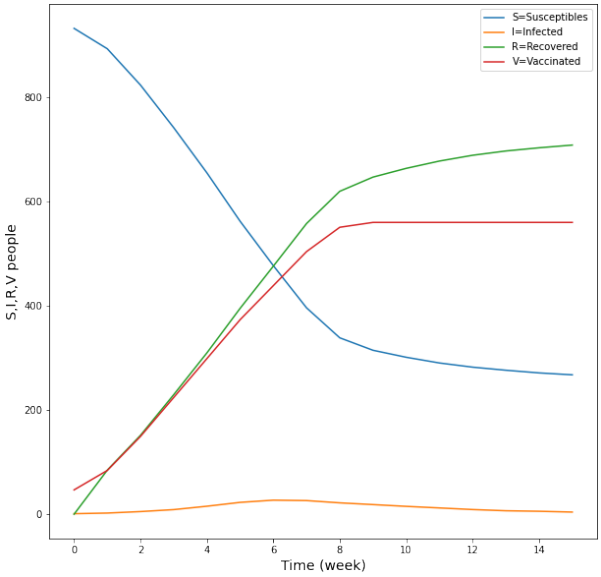


Figure 14: Average number of S,I,R,V people each week



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

650). Moreover, the best epidemic evolution of newly infected people is reached in Figure 11 visually: it seems there is no time lag anymore, which was the lack seen in all the simulations with the preferential attachment graph. Indeed, the predicted maximum number of newly infected individuals is reached during the 6th week and kept 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 contribution to increasing the RMSE. A solution could consist of a modification of the parameters' search in the algorithm. This work is not done due to computational time, but an improvement could be reached by doing more simulations to have a more reliable

estimation of RMSE for each configuration tried. This would avoid getting stuck in a sub-optimal space, so consider other parameters' space.

2. Coloring

This problem deals with distributed learning in potential games applied in graph coloring, which consists in assigning a colour to each node of an undirected graph, with the constraint that the colour assigned to a node is different from the colours of all its neighbours. 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.

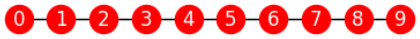


Figure 15: Simple line graph with 10 nodes

2.1. Line graph

At initialization, each node in the graph of Figure 15 is red. During the simulation, the nodes can update their colour 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 colour, 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 that a new action a 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 the denominator assigned to $\eta(t)$, the more 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 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 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 with more configurations. 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 a higher configuration value is reached, the transition probability for the algorithm to go back to the previous low potential state is very high.

2.2. General example graph

Similarly to the first coloring problem, this exercise requires assigning eight colours 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 colour in the set $C = \{red,\ green,\ blue,\ yellow,\ magenta,\ cyan,\ white,\ black\}$

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

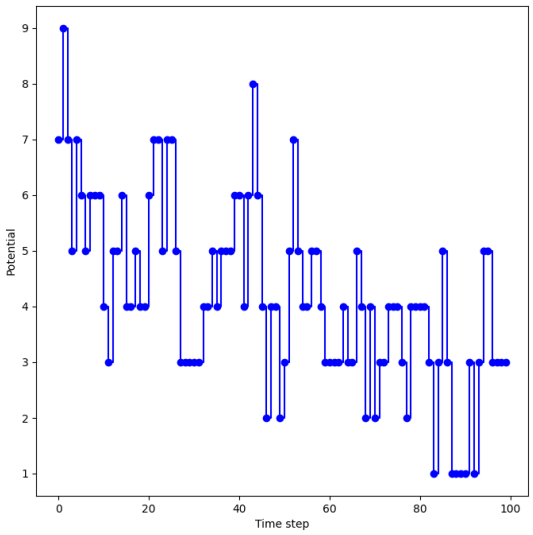
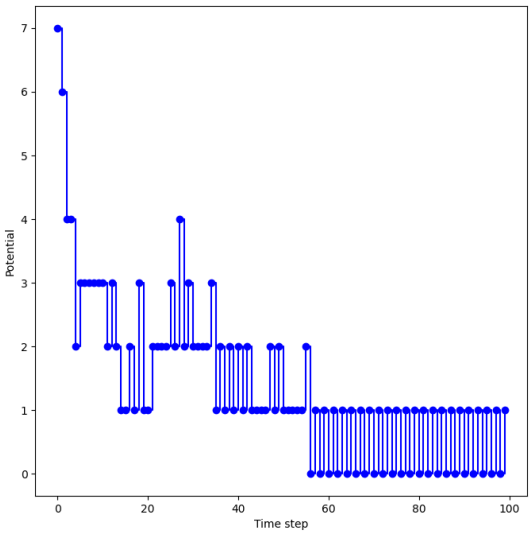


Figure 17: $\eta(t) = t/10$



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

Figure 18: *n_steps=100*

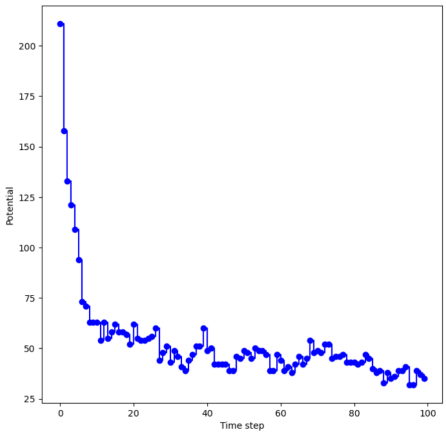


Figure 19: *n_steps=250*

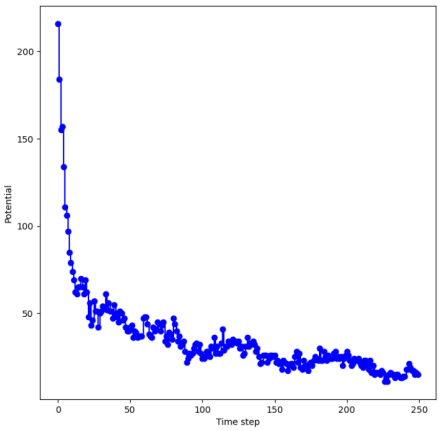
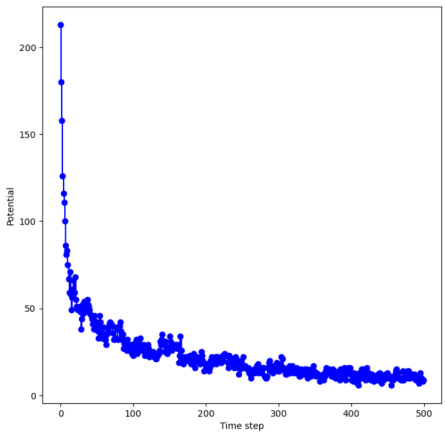


Figure 20: *n_steps=500*



Potential functions generated by different simulation periods and same $\eta(t) = t/100$ function

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 neighbour nodes have the same frequency, a 2 points penalty is assigned; instead, if they have similar frequencies (frequencies having only one frequency between them), a penalty of 1 is assigned; no penalty for any other case.
At every discrete time instant t, one node is randomly chosen to change its colour, according to all the transition probabilities computed as in (2). In addition, the utility and the potential are computed respectively as in (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 plots show a similar trend: the longer the simulation, the lower potential is obtained. Starting from a maximum

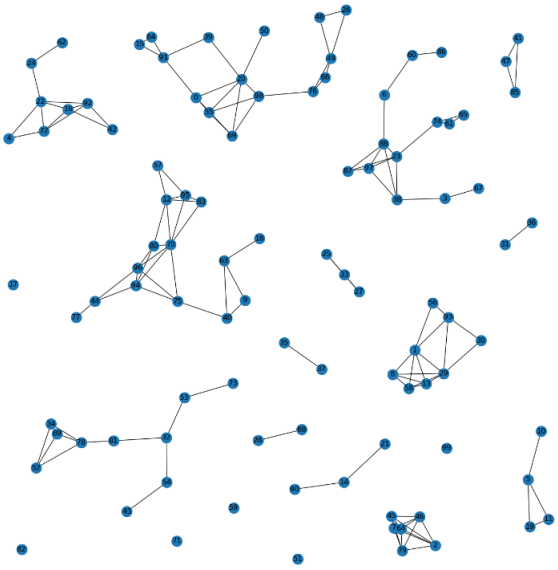


Figure 21: Network with 100 routers and their links

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 behaviour 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 colours 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 the frequencies are only 8, the cost of the graphs in Figures 22-23 are respectively 1 and 3 points.

2.3. Evaluation for different choices of $\eta(t)$

At this point, it is required to evaluate what happens for different choices of $\eta(t)$. Firstly, constant values of $\eta(t)$ are evaluated. Then, two increasing functions $\eta(t)$ are considered.
In order to obtain a better analysis, each $\eta(t)$ is evaluated with various time steps. Specifically, the numbers of time steps considered are 30, 50, 100, 250, and 500.
In Tables 3-4, the results obtained for each scenario are reported. 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 very much for different code compilations.
When $\eta(t)$ is constant, it reaches the minimum observed value of the potential function ($U(t) = 4$) when it is smaller (in the cases reported, $\eta(t) = 100, 10$) in less

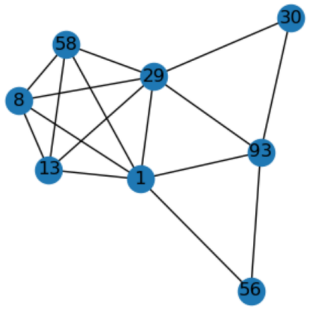


Figure 22: Graph with 5 fully connected routers

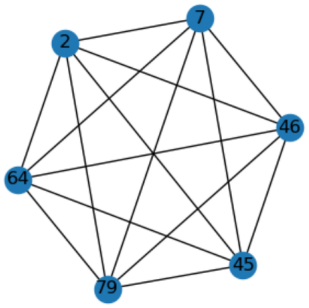


Figure 23: Fully connected network with 6 routers

TABLE 3: 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 4: Minimum values reached by the potential function

η(t)	U(t)				
	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

than 500-time units. Then, for constant values of $\eta(t)$, the potential function is better minimized for smaller $\eta(t)$. For what concerns the exponential function, it works quite well, since the potential function reaches a good minimum point ($U(t) = 5$) in less than 200 steps (in particular 67, 108 and 65 steps). Eventually, for the function $\eta(t) = 10t$, the minimum value reached is $U(t) = 4$. 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 $\eta(t)$ 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 it is noisy since they do not get stuck in a sub-optimal configuration. This does not happen in the cases in which $\eta(t)$ assumes constant values as 10000, 1000, 100 and 10: even though these dynamics risk getting stuck in a not-optimized configuration, they perform well. In particular, the small values of $\eta(t)$ (10 and 100) are a good trade-off between noisy and not noisy dynamics, as they reach the minimum potential function values. In general, the best $\eta(t)$ seems to be $10t$ and 100, because they reached the minimum potential function value ($U(t) = 4$) respectively in 97 steps and 68 steps.

In the Appendix, the graphics of potential function for $\eta(t) = 100$ and $\eta(t) = 10t$ are reported.

Appendix

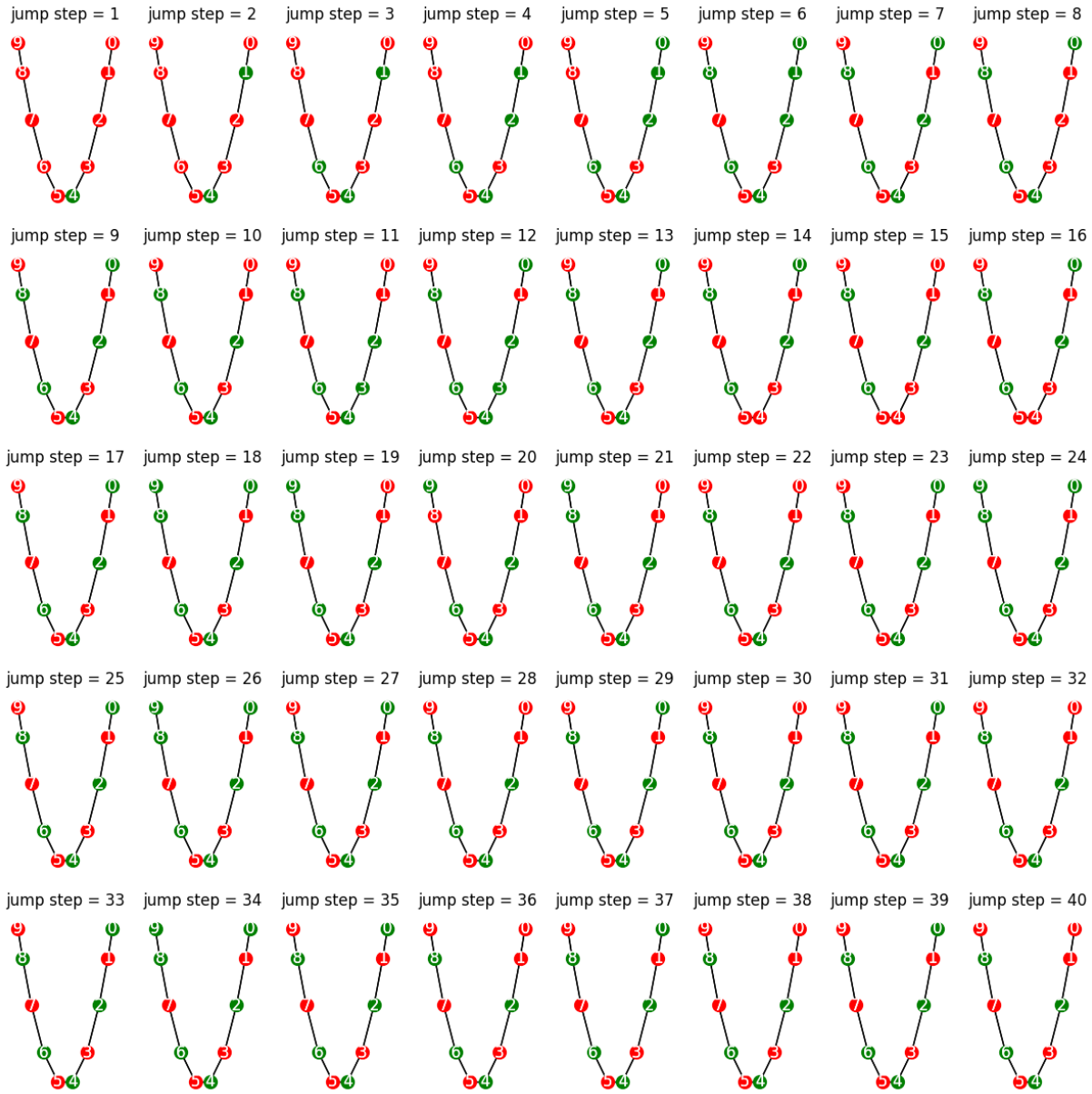


Figure 24: Learning dynamics simulated by coloring algorithm: $\eta(t) = t/2$

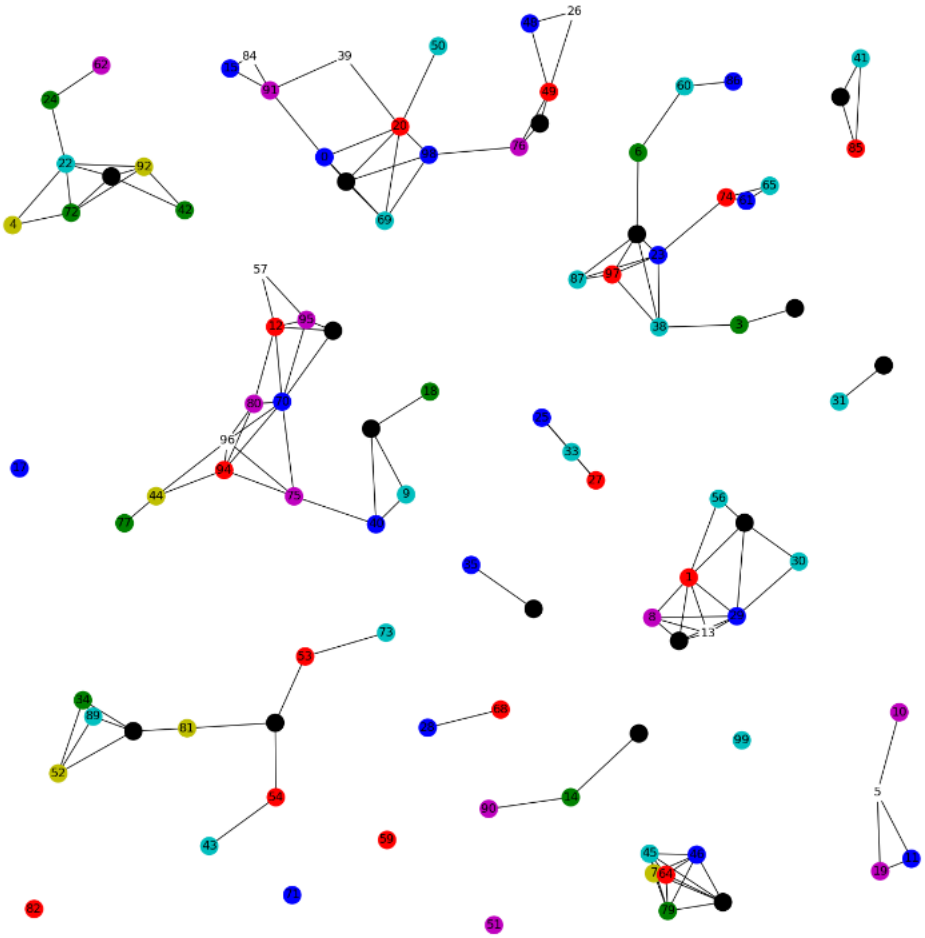


Figure 25: Final coloring of the graph

Figure 26: *n_steps=30*

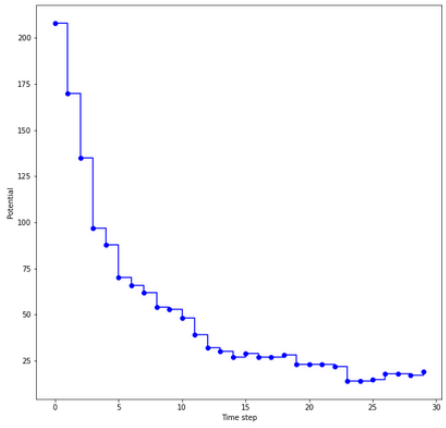


Figure 27: *n_steps=50*

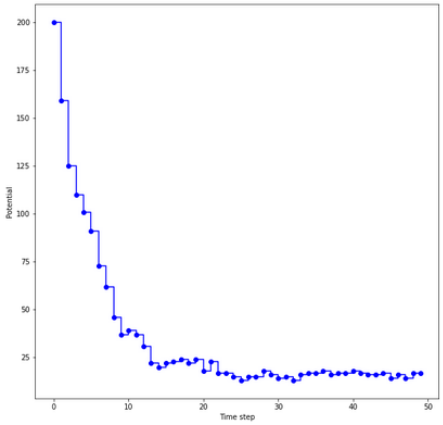
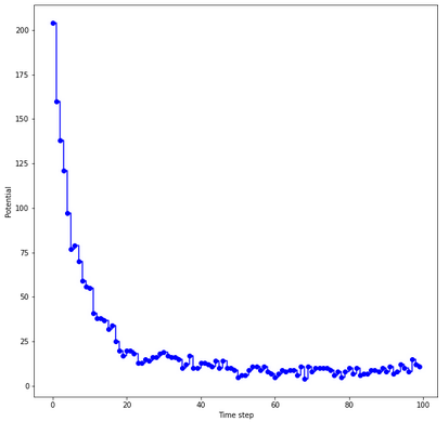


Figure 28: *n_steps=100*



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

Figure 29: *n_steps=250*

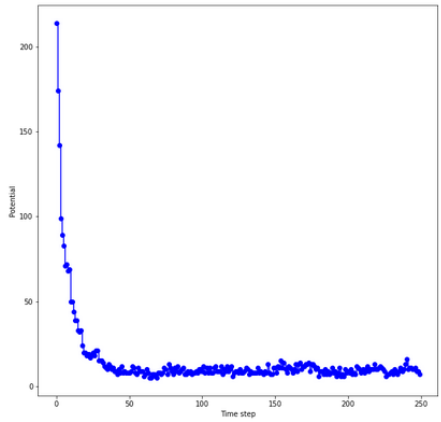
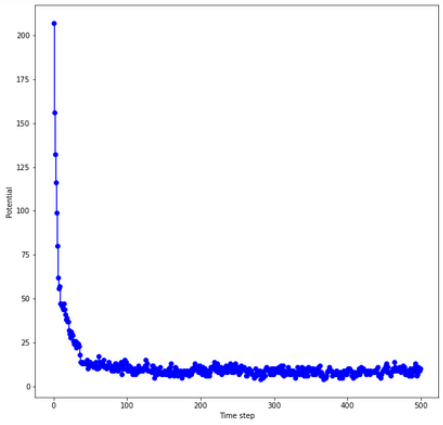


Figure 30: *n_steps=500*



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

Figure 31: n_steps=30

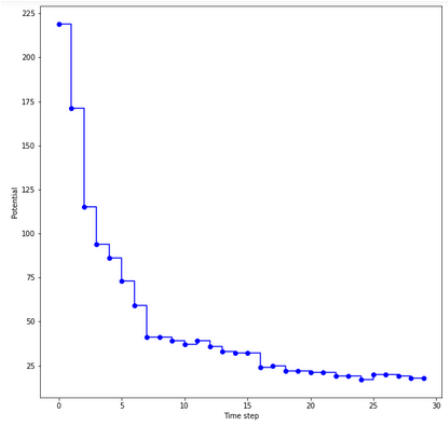


Figure 32: *n_steps=50*

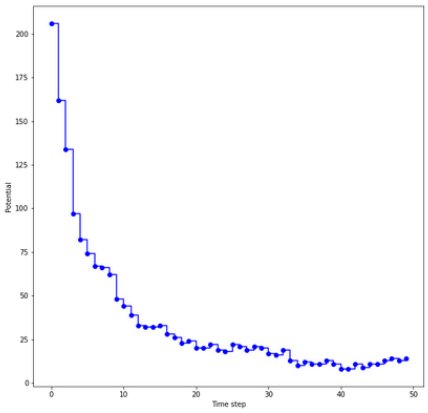
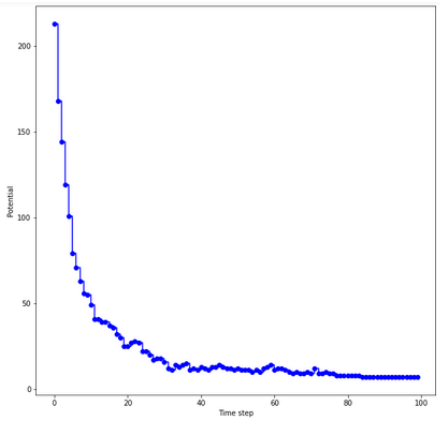


Figure 33: *n_steps=100*



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

Figure 34: *n_steps=250*

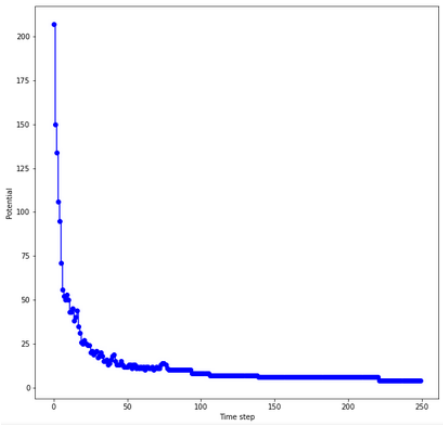
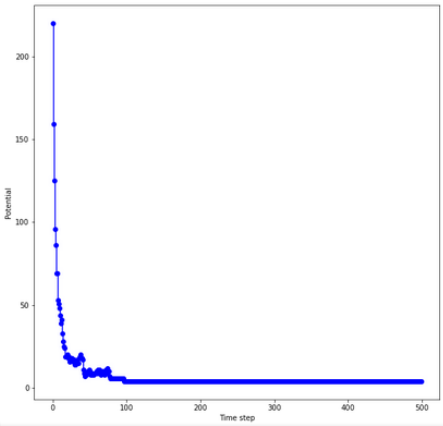


Figure 35: *n_steps=500*



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