Network Dynamics and Learning

Homework 3

2023-2024

Carena Simone, s309521
Carmone Francesco Paolo, s308126
Mazzucco Ludovica, s315093

This document has been produced jointly by the students mentioned above, who have tightly collaborated throughout the whole homework.

Exercise 1

In this task we simulate the H1N1 pandemic that plagued Sweden in 2009, with the goal of estimating the network-structure characteristics and the disease-dynamics parameters of the model. The disease propagation model used to simulate the epidemic is a discrete-time simplified version of the SIR model. In particular, during each time step there is a probability β that the infection is spread form an infected individual to a susceptible one, and a probability ρ that each infected individual recovers.

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

$$\mathbb{P}\left(X_i(t+1) = R | X_i(t) = I\right) = \rho$$

If a vaccination action is considered, once someone receives the vaccines, they immediately become vaccinated, regardless of their current status. A vaccinated individual cannot infect nor be infected.

The network used to simulate the pandemic consists of a random graph with an average degree of k. The three parameters to be estimated for the Swedish pandemic are then k, β and ρ. Before simulating the real systems, other simulations are made on different graph structures and in different conditions: in particular with and without considering the vaccination campaign.

Epidemic on a Known Graph

The first simulation is performed on a on a symmetric k-regular undirected graph with $n = |\mathcal{V}|$ nodes. In particular the following parameters are considered: $k = 4$, $n = 500$, $\beta = 0.3$ and $\rho = 0.7$. The simulation is performed for 15 weeks and considering a starting infected population of 10 individuals. To take into account the stochasticity of the process, the epidemic diffusion is simulated $N = 100$ times, and the final results presented in (Fig. 1) are the average over all of the simulations.

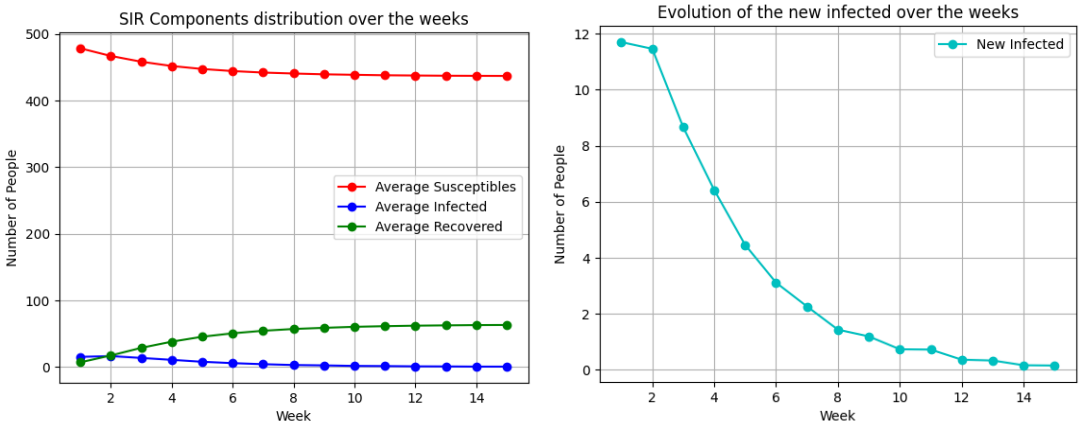


Figure Fig. 1: (Symmetric Regular Graph) Evolution of the Pandemic over the 15 Weeks (on the left). Evolution of the Number of New Infected Over the 15 Weeks (on the right)

The pandemic was simulated using algorithm (Alg. 1).

Epidemic on a Random Graph (Without Vaccines)

The simulation is now performed on a random graph generated with *preferential attachment*, with 500 nodes and an average degree $k = 6$. The epidemics parameters are $\beta = 0.3$ and $\rho = 0.7$, and the initial condition comprises of 10 infected individuals. As before the simulation is repeated 100 times and the averaged results are presented in (Fig. 2).

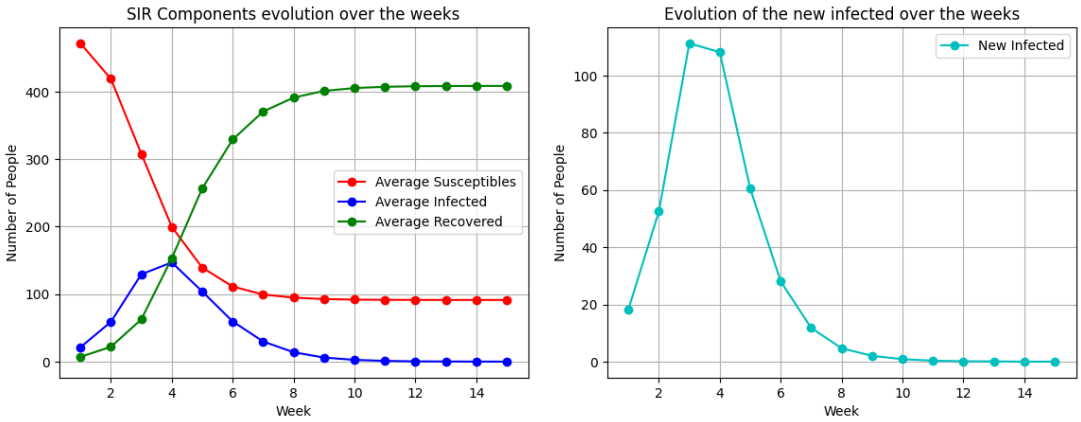


Figure Fig. 2: (Random Graph, Without Vaccines) Evolution of the Pandemic over the 15 Weeks (on the left). Evolution of the Number of New Infected Over the 15 Weeks (on the right)

Epidemic on a Random Graph (With Vaccines)

We now consider the case in which vaccines are distributed to the population over the 15 weeks of simulation. The cumulative percentage of vaccinated people is described as

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

The simulation is again performed on a random graph generated with preferential attachment, with 500 nodes and an average degree $k = 6$. The epidemics parameters are again $\beta = 0.3$ and $\rho = 0.7$, and the number of initial infected individuals is 10. The average results obtained simulating the epidemic for 100 times are reported in (Fig. 3).

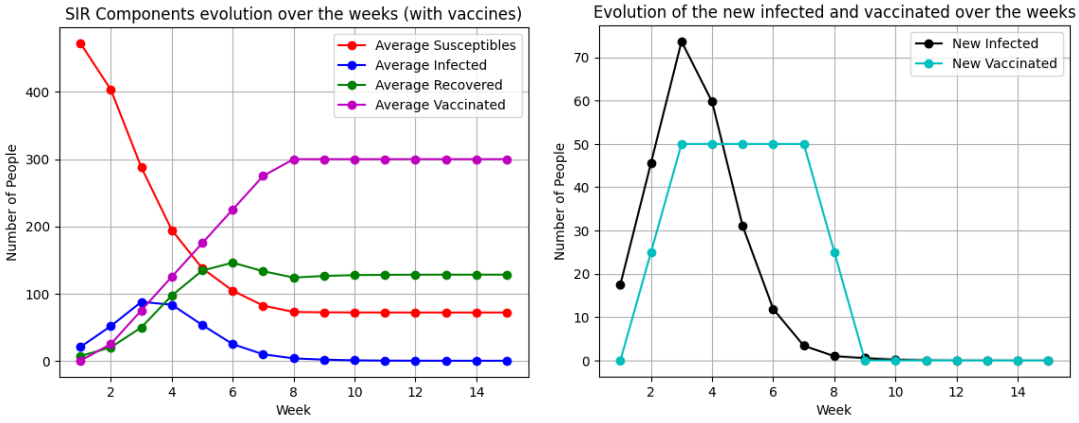


Figure Fig. 3: (Random Graph, With Vaccines) Evolution of the Pandemic over the 15 Weeks (on the left). Evolution of the Number of New Infected Over the 15 Weeks (on the right)

The H1N1 pandemic in Sweden 2009

We now simulate the pandemic on Sweden. We consider a more limited scenario, described by a network of $n = |\mathcal{V}| = 934$ nodes. During the 15 weeks of simulation the number of newly infected people weekly, scaled to match the size of the considered network, was

$$\text{TrueInfected}(t) = [\,1, 1, 3, 5, 9, 17, 32, 32, 17, 5, 2, 1, 0, 0, 0, 0\,]$$

And the cumulative percentage of population that received the vaccination was

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

Since the size of such vectors is 16, the first element has been interpreted as the initial condition for the simulation. As such, the networks is initialized with

$$x_0 = \begin{cases} 887 & \text{Susceptible} \\ 1 & \text{Infected} \\ 0 & \text{Recovered} \\ 46 & \text{Vaccinated} \end{cases}$$

The objective of this task was to find the k, β and ρ parameters that best approximate the real data on newly infected people. To achieve such task, a slight variation of the algorithm described in (Alg. 2) was used. In particular the algorithm is run several times, each time using as initial guess on the parameters the best result it got from the previous iteration.

The simulation found as best parameters $k = 10$, $\beta = 0.2$, $\rho = 0.7$ corresponding to an RMSE of 5.14. The evolution of the agents' states and of the newly infected people is reported in (Fig. 4).

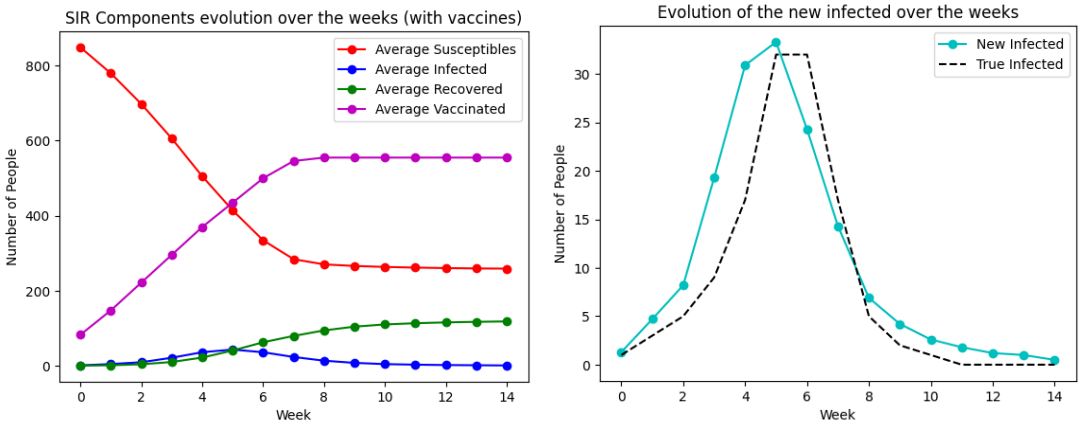


Figure Fig. 4: Evolution of the Pandemic over the 15 Weeks (on the left). Evolution of the Number of New Infected Over the 15 Weeks (on the right)

It is important to note that the RMSE reported here represents just one of the possible realizations of the algorithm used. Further run of it would result in a plethora of different values. This is due to the fact that each pandemic, for each set of parameters, is simulated only 10 times, and given the stochasticity of the process the variance on the resulting RMSE is noticeable.

Exercise 2

In this second task we inspect the properties of a graph representing wifi routers and their connections by means of its coloring, in other words, the final goal is to find a configuration in which neighboring nodes (i.e. the routers) have different colors in the sense that they do not interfere with each other. Their dynamics is driven by the minimization of a certain cost function that increases if their color is like one of the directly attached nodes, in particular, the probability to change state to a certain color c is inversely proportional to the corresponding cost. In this way, any of the agents is lead to decrease the overall potential function, defined as:

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

An additional noise function $\frac{1}{\eta}$ depending on time is inserted in the cost function in order to cope with situations in which the network falls in a local minima and none of the nodes is able to increase their utility although a lower potential can be reached. In order to set it properly, it has to decrease in time in order to allow for a harsh initial exploration and then just a loyal exploitation of the given probabilities. During the simulations η is set as:

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

Further in the report we will focus on the influence on performance of the η shape.

Point A

We experience the environment disclosed above starting with a simpler setting, in which the network is represented by a line graph with 10 nodes and just two colors, red or green. We already know that a line graph is bipartite, thus we expect that the potential function will easily reach zero in finite time. The following figure is showing the obtained simulation results.

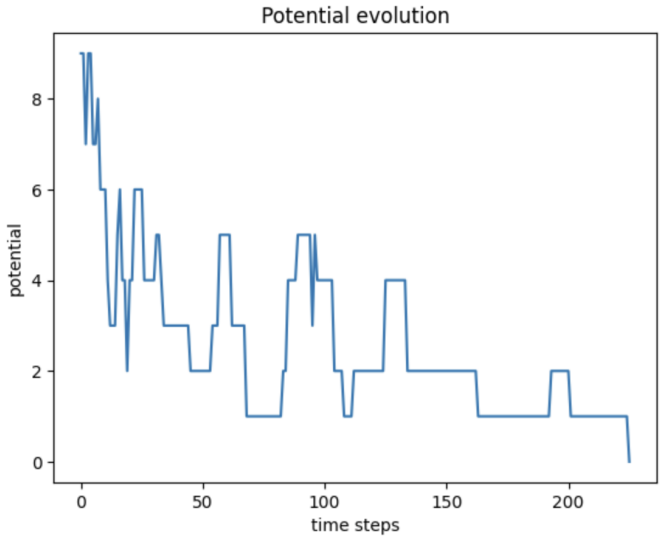


Figure Fig. 5: Potential function evolution for the line graph with 10 nodes

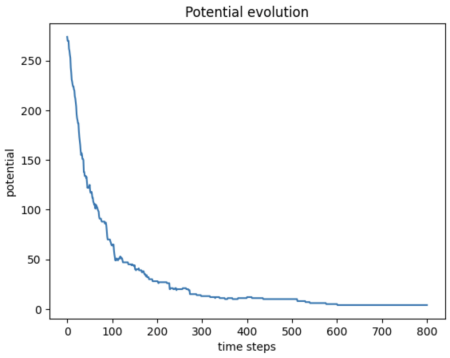
As it can be noticed, at first the potential oscillates due to the high value of the noise, nonetheless it quite immediately starts to acquire a decreasing trend converging to zero in less than 250 steps and confirming what we have anticipated.

Point B

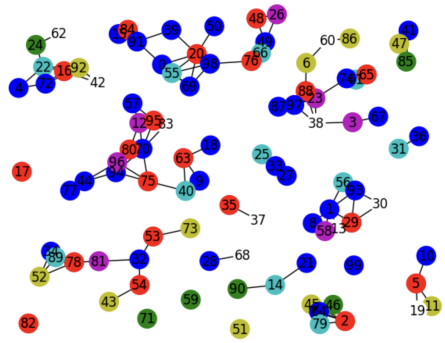
We approach now a more complicated situation in which there are 100 access points that need to be coordinated and, instead of just two channels, in this case there are eight different ones. Not less important, the graph is no more a line but it assumes a more complex shape with multiple isolated components. To take into account those differences with respect to the previous case, the cost function has to be modified as well and it is defined as:

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

In this way we are penalizing the nodes even if they assume a color differing by one with respect to neighboring agents one.
Running the simulation we noticed an increased inertia to converge, plus the lower bound of the potential function is not null as before but the function approaches the value of 4 in more time (600 time steps ca.) and does not decrease any more, assuming the shape of a perfect hyperbole, as it can be seen in the figure below, together with a visual representation of the final configuration of the network.



(a) Evolution of potential function



(b) Final configuration of the graph

Unlike the previous case, where we predicted the potential would have converged to zero since the graph was bipartite (hence 2-colorable), here the same reasoning does not apply anymore. In fact, even though we calculated that the given graph is 6-colorable (with the NetworkX function `nx.greedy_color`) the simulation highlights that the potential stabilizes close but not exactly at zero. This is due to the fact that the cost function penalizes the case in which the color of neighbors differs by one as well, while in the coloring problem what matters is just that close nodes have different colors. In fact, if the simulation is run with the cost function given in Point A then the potential succeeds to converge to zero, as the plot below demonstrates.

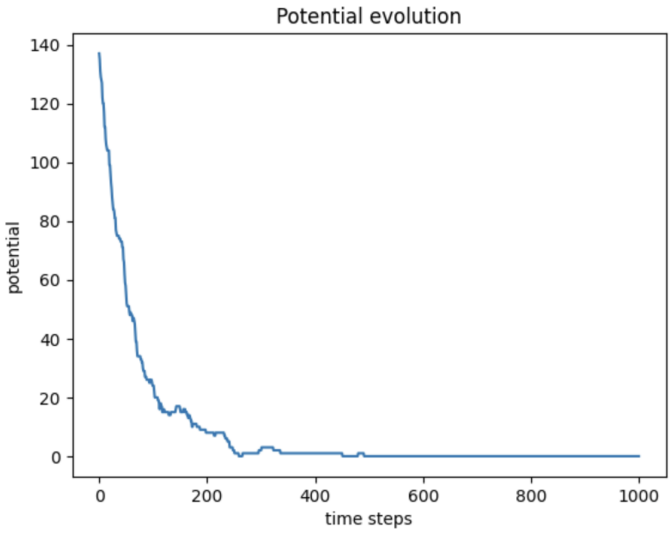


Figure Fig. 7: Potential function evolution with a different cost function

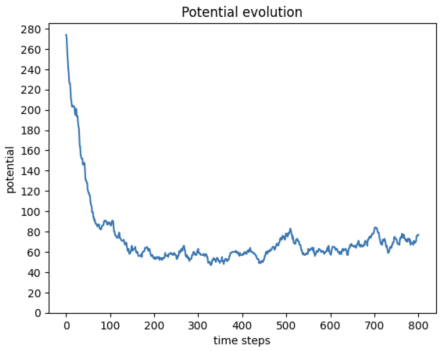
In order to further inspect this case and firmly confirm the correctness of our results, we implemented an additional algorithm which computes the dynamics of the color modification like the previous one but in a slightly different way, the pseudo-code is given in Alg. 3. This time, instead of choosing uniformly the node to trigger the transition from, we divide the whole graph in all its connected components, at each time step we take into account the one yielding the greater potential and we randomly choose a node in there hoping the potential will decrease. To make it more efficient, we avoid the situation in which the same node is chosen twice consequently. Moreover, if the algorithm chooses the same connected component for too many iterations, meaning that its potential is the highest in the graph but it cannot be decreased anymore, than we exclude it from the next computations. Eventually, we end up in a setting in which every connected component potential can be either null or a positive value that cannot be decreased, as the corresponding subgraph has reached an "equilibrium". Even in this case, we finally get an overall potential of 4, besides for every trial the same two subgraphs remain with a positive potential value, as it can be noticed from the simulation output reported below:

```
POTENTIAL:  4.0
[({64, 2, 7, 45, 46, 79}, 3.0), ({1, 8, 13, 29, 56, 58, 93, 30}, 1.0)]
```

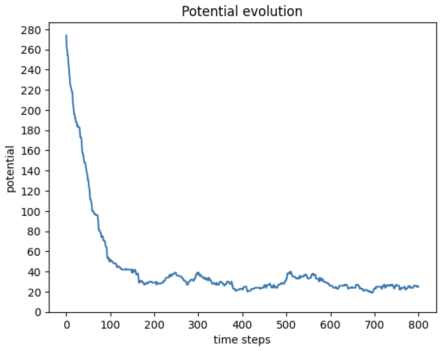
We infer that this happens due to the intrinsic structure of those subgraphs and the inner connections between the nodes they are made up with, since at this stage the noise has been dramatically decreased.

Point C

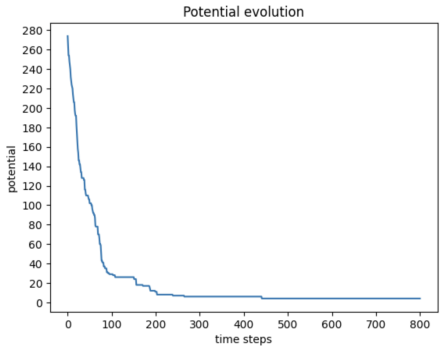
In this section we are going to focus on the effect of the noise in the dynamics of the potential. As we have said, the general rule to design η is to choose for it an increasing function since it is the inverse of noise and we want it to be grater at the beginning to allow nodes to explore more thus hinder the fall in local minima.
We tried first what happens setting η to different constant values: 0.1, 1 and 10; in this way it is possible to view the impact of noise at various orders of magnitude. The figures below are reporting the simulation results.



(a) $\eta = 0.1$



(b) $\eta = 1$



(c) $\eta = 10$

Having a constant parameter implies admitting a constant level of exploration and obtaining something always oscillating without approaching a definite value. Nonetheless, if the exploration is low like in the case of $\eta = 10$ the performance is similar to the case with varying parameter and the potential reaches 4. On the contrary, if we set a high exploration level $\eta = 0.1$ the performance drastically decreases with the potential unable to enhance further 60, since the algorithms explores too much.
Moving to variable functions, if we choose $\eta = t$ or $\eta = e^t$ the simulation soon stops since probabilities grow too fast, then a reasonable trial would be $\eta = log(t)$ (with particular attention when t is zero or 1). The simulation output is shown below:

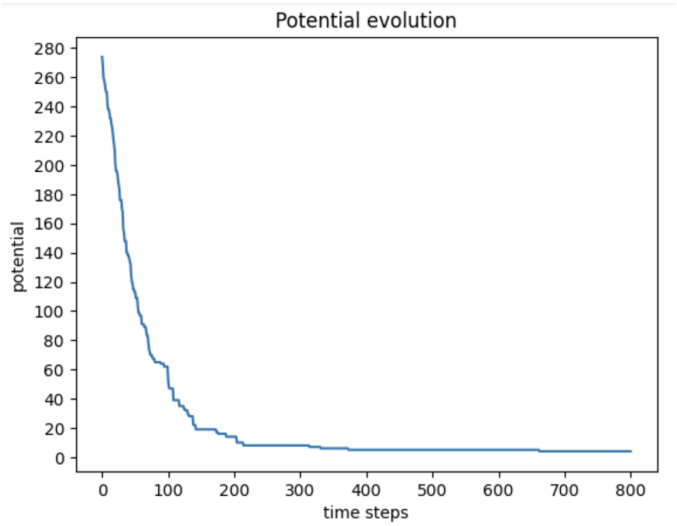


Figure Fig. 9: Potential function evolution with logaritmic noise

As it can be noticed, the potential approaches the minimum value of 4 but more slowly with respect to the case with linear noise, as the logaritmic function grows in a smoother way.

Algorithms

Algorithm Alg. 1 SIR Epidemic Simulation

Input: $\mathcal{G}, \beta, \rho, T, x_0, Vaccines$ (optional)
  $x \leftarrow x_0$
  for t in $[0, T)$ **do**
    $y = x$
    $Counter \leftarrow \mathbf{0}_4$ ▷ One counter for each state
    $NewInfected \leftarrow 0$
    if Vaccines are distributed **then**
      $N_v \leftarrow$ Number of vaccines
      $vaccinated \leftarrow N_v$ individuals among the unvaccinated
      $y[vaccinated] = 3$
    end if
    for all *agent* in *Agents* **do**
      if $x[agent] = 0$ **then** ▷ *agent* is susceptible
        $m \leftarrow$Number of infected neighbors
        $p \leftarrow (1 - \beta)^m$
        $state \leftarrow 0$ with probability p or 1 with probability $1 - p$
        $y[agent] \leftarrow state$
        if $state = 1$ **then**
          $NewInfected \leftarrow NewInfected + 1$
        end if
      else if $x[agent]$=1 **then** ▷ *agent* is infected
        $state \leftarrow$ 1 with probability ρ or 2 with probability $1 - \rho$
        $y[agent] \leftarrow state$
      else ▷ The agent is recovered or vaccinated
        $state \leftarrow x[agent]$
        $y[agent] \leftarrow state$
      end if
      $Counter[state] \leftarrow Counter[State] + 1$
    end for
    Store the statistics for the current time-step t
    $x \leftarrow y$
  end for
Output: The statistics over the simulation period T

Algorithm Alg. 2 Parameters Search

Input: $n, T, N, Vaccines, TrueInfected, P_0$

$[\,k, \beta, \rho\,] \leftarrow P_0$
$\Delta k \leftarrow 1$
$\Delta \beta \leftarrow 0.1$
$\Delta \rho \leftarrow 0.1$
$\mathbb{S}_k \leftarrow [\,k - \Delta k, k, k + \Delta k\,]$
$\mathbb{S}_\beta \leftarrow [\,\beta - \Delta \beta, \beta, \beta + \Delta \beta\,]$
$\mathbb{S}_\rho \leftarrow [\,\rho - \Delta \rho, \rho, \rho + \Delta \rho\,]$
$\mathbb{S} \leftarrow \mathbb{S}_k \times \mathbb{S}_\beta \times \mathbb{S}_\rho$
for all $parameters$ in $\mathbb{S}$ **do**
  $[\,k, \beta, \rho\,] \leftarrow parameters$
  $\mathcal{G} \leftarrow$ Random graph of average degree k with n nodes
  $x_0 \leftarrow$ Initial state of the network agents
  $[\,S_{tot}, I_{tot}, R_{tot}, V_{tot}, NewlyInfected_{tot}\,] \leftarrow \mathbf{0}$
  for i in $[0, N)$ **do**
    $[\,S, I, R, V, NewlyInfected\,] \leftarrow$ Simulate the SIR dynamics of graph $\mathcal{G}$ with β and ρ
    $S_{tot} \leftarrow S_{tot} + S$
    $I_{tot} \leftarrow I_{tot} + I$
    $R_{tot} \leftarrow R_{tot} + R$
    $V_{tot} \leftarrow V_{tot} + V$
    $NewlyInfected_{tot} \leftarrow NewlyInfected_{tot} + NewlyInfected$
  end for
  Average the values of $S, I, R, V, New; yInfected$ over the N iterations
  $RMSE \leftarrow \sqrt{\frac{1}{T} \sum (TrueInfected - NewlyInfected)^2}$
  Save the epidemic statistics and the RMSE associated to the current parameters
end for

Output: Best parameters and associated RMSE and SIR statistics

Algorithm Alg. 3 Variant of the Coloring Problem with Connected Components

Input: W, G

$x_0 \leftarrow \begin{bmatrix} 1 & 1 & ... & 1 \end{bmatrix}$ ▷ initialize all agents with color 1(red)
$states \leftarrow []$
$states[-1] \leftarrow x_0$
$t \leftarrow 0$
$u \leftarrow$ compute potential at states[t]
$conn\_comp, dict\_cc \leftarrow$ list all connected components and calculate their potential
$prev\_node \leftarrow -1$
$prev\_set \leftarrow set()$
$same\_set \leftarrow 0$
$non\_decreasing\_pot \leftarrow$
while dict_cc is not empty **do**
    $to\_minimize \leftarrow$ subgraph with highest potential
    $index, pot \leftarrow to\_minimize[0], to\_minimize[1]$
    $subgraph \leftarrow conn\_comp[index]$
    if pot == 0 **then**
        delete subgraph from dict_cc
        continue
    end if
    if subgraph == prev_set **then**
        $same\_set \leftarrow same\_set + 1$
        if same_set == 500 **then**
            $same\_set \leftarrow 0$
            delete subgraph from dict_cc
            $non\_decreasing\_pot[index] \leftarrow pot$
            continue
        else
            $same\_set \leftarrow 0$
        end if
    end if
    $prev\_set \leftarrow subgraph$
    $x \leftarrow states[t]$
    $I \leftarrow$ choose uniformly a node in subgraph
    while I == prev_node **do**
        $I \leftarrow$ choose uniformly a node in subgraph
    end while
    $prev\_node \leftarrow I$
    $color \leftarrow$ choose color for node I
    $x[I] \leftarrow color$
    $states[-1] \leftarrow x$
    $t \leftarrow t + 1$
    $new\_pot \leftarrow$ compute new potential of subgraph
    $dict\_cc[index] \leftarrow new\_pot$
    $u \leftarrow$ compute the overall potential at states[t]
end while
$residual\_components \leftarrow$ list of (subgraph,potential) tuples for all subgraphs left inside non_decreasing_pot

Output: $u, residual\_components$