Homework III

M.Sc. Data Science and Engineering - AY 2022-2023

Network Dynamics and Learning

Professors: F. Fagnani, G. Como, L. Cianfanelli

Pietro Cagnasso
s300801
pietro.cagnasso@studenti.polito.it

Nicolò Vergaro[1]
s295633
nicolo.vergaro@studenti.polito.it

Exercise 1 Influenza H1N1 in Sweden (2009)

This first exercise is a journey that ends with a simplified analysis of Influenza H1N1 pandemic in Sweden, 2009.

1.a Pandemic on a known graph

We start this journey studying a SIR pandemic on a symmetric k-regular graph. To start we need to define these two components:

Definition 1 (SIR pandemic) *In a SIR pandemic each node can be in one of three possible states: Susceptible, Infected, Recovered (or Removed). So we can define the set of possible states as* $\mathcal{A} = \{S, I, R\} = \{0, 1, 2\}$. *In this kind of context a susceptible node can be infected with probability* β *by one of its infected neighbors, an infected node recovers with probability* ρ. *Once recovered a node cannot become infected anymore.*

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *state* $X \in \mathcal{A}^{|\mathcal{V}|}$ *we can formalize in mathematical terms these behaviors as:*

$$P(X_i(t+1) = I \mid X_i(t) = S, \sum_{j \in \mathcal{N}_i} \delta^I_{X_j(t)} = m) = 1 - (1-\beta)^m,$$

$$P(X_i(t+1) = R \mid X_i(t) = I) = \rho.$$

As a consequence a node will remain susceptible with probability $(1-\beta)^m$ *and will remain infected with probability* $1 - \rho$.

Definition 2 (Symmetric k-regular graph) *This kind of graph is a graph that starts from a ring with* $\mathcal{V}$ *nodes, than adds to each of these* $(k-2)/2$ *links towards the first preceding and following nodes. As a result each node in the graph has as degree* k.

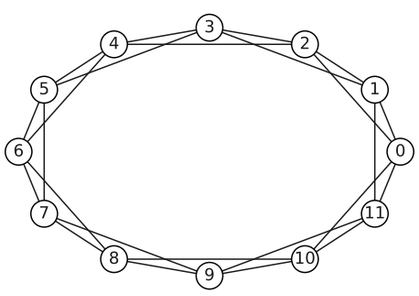

Figure 1: Symmetric 4-regular graph with 12 nodes.

To create a symmetric k-regular graph we defined a function `symmetric_k_regular_graph` to which we can pass the number of nodes `n_nodes` and the desired degree `k`. This function first creates a circle with the given number of nodes, then adds an edge to the first $(k-2)/2$ nodes on the left

[1] With whom I collaborated on the entire homework.

and the right starting from the second (the first is already connected since it is a circle). We created a graph with 500 nodes and average degree 4.

On the obtained graph we simulated a 15 weeks SIR pandemic for 100 times, using the parameters $\beta = 0.3$ and $\rho = 0.7$ starting with 10 randomly chosen infected. To simulate the pandemic we defined an iterative approach:

- for each simulation i
  - initialize the state $X_{i,j,0} = S \ \forall j \in \mathcal{V}$
  - randomly choose 10 infected, and set their state $X_{i,j,0} = I \ \forall j \in \ infected$
  - for each week w, for each node n
    - if $X_{i,n,w-1} = S$ with probability $p = 1 - (1 - \beta)^m$, where m is the number of its neighbors infected in $w - 1$, it becomes infected ($X_{i,n,w} = I$)
    - else if $X_{i,n,w-1} = I$ with probability $p = \rho$ it becomes recovered ($X_{i,n,w} = R$)

While running this algorithm we kept track of the number of new infected each week for each simulation and the number of people in each state in each week for each simulation. Averaging these statistics we got the plots in Fig. 2. As clearly visible in the plots the pandemic quickly comes to an end. This behavior can be explained by the large difference between the contagion and recovery rate combined with very small amount of infected at the beginning.

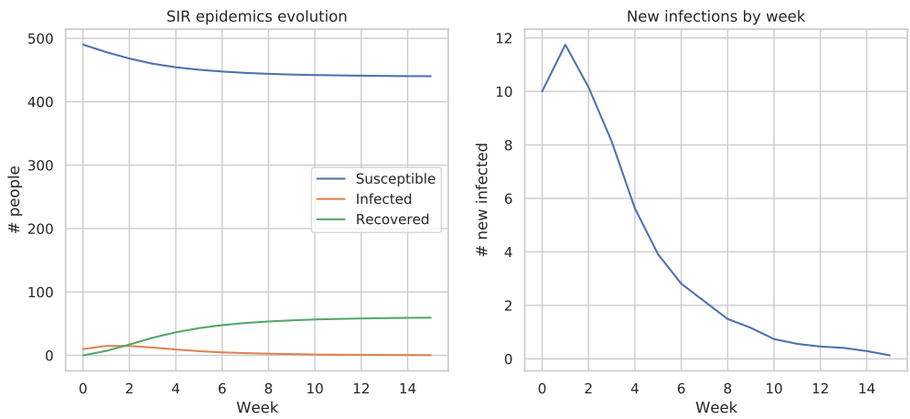

Figure 2: Evolution of the SIR pandemic on the symmetric 4-regular graph and amount of new infected each week, results averaged over 100 simulations.

1.b Random graph

In the second step of this journey we define a preferential attachment random graph that will be the network on which we will study the Influenza H1N1 pandemic in Sweden.

Definition 3 (Random graph) *A random graph is a graph in which some, if not all, the edges takes place following a probability distribution. In the past decades a lot of models to create this kind of graphs were proposed, among those we can remember: Erdös-Rényi random graph, configuration model, preferential-attachment model, and small-world model.*

Definition 4 (Preferential attachment model) *This kind of graph is defined starting from a complete graph with $k + 1$ nodes (so that each node has degree k). Than at each time stamp $t > 1$ we add a new node to the graph $\mathcal{G}^t$ with $c = \lfloor k/2 \rfloor$ edges towards preexisting nodes. The nodes in $\mathcal{G}^t$ are chosen accordingly to a probability proportional to their degree in a way that no two new links point to the same node. Notice that if k is odd we should alternatively add and remove 1 from c so that on average the graph will maintain a degree k.*

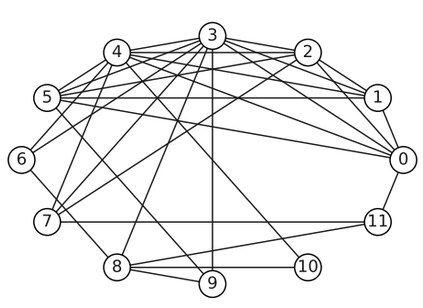

Figure 3: Preferential attachment graph with 12 nodes and average degree 5.

To create a preferential attachment random graph we defined a function `pa_random_graph` to which we can pass the number of desired nodes `n_nodes` and the desired degree `k`. Additionally, for reproducibility purposes, we added to the function the parameter `seed` that allows us to produce always the same graph so that we can reproduce each time the pandemic on the same graph.

To test this function we created a graph with 900 nodes and degree 20. Due to its large size and high edge density an image of this graph would be a black spot.

1.c Pandemic without vaccination

To simulate a pandemic without vaccination we constructed a preferential attachment random graph using the function `pa_random_graph` with 500 nodes and average degree $k = 6$. Then we simulated a SIR pandemic using the same algorithm defined in Sec. 1.a once again simulating it 100 times over 15 weeks, with parameters $\beta = 0.3$ and $\rho = 0.7$ starting with 10 randomly chosen infected. We kept track of the number of new infected each week for each simulation and the number of people in each state in each week for each simulation. Averaging these statistics we got the plots in Fig. 4.

As already noticed above the pandemic vanishes quickly due, probably, to the very high recovery rate with respect to the infection rate.

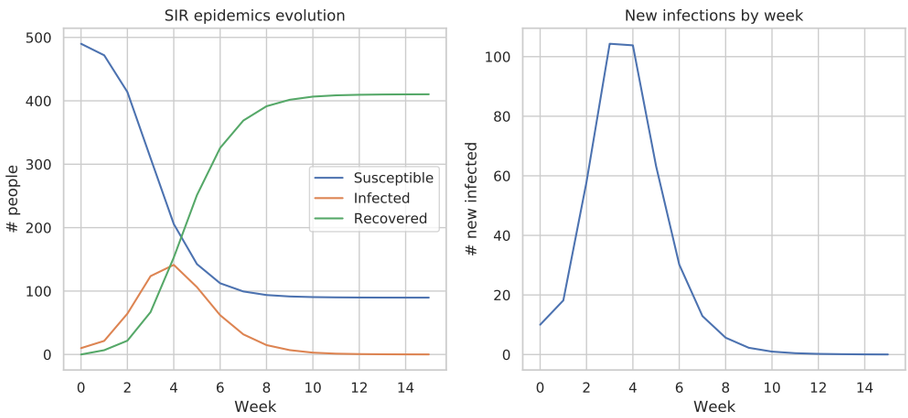

Figure 4: Evolution of the SIR pandemic on the 500 nodes preferential attachment random graph and amount of new infected each week, results averaged over 100 simulations.

1.d Pandemic with vaccination

Proceeding with our journey we introduce the vaccination as a system to control the spread of the pandemic. To do so we can define:

Definition 5 (SIRV pandemic) *In a SIRV pandemic each node can be in one of the states of a SIR pandemic, but we add a new state: Vaccinated. So, we have to redefine the the set of possible states as* $\mathcal{A} = \{S, I, R, V\} = \{0, 1, 2, 3\}$. *As in the SIR pandemic a susceptible node can be infected with probability* β *by one of its infected neighbors, an infected node recovers with probability* ρ *and once recovered a node cannot become infected anymore. In addition to this process we introduce the vaccination: each week a portion of the population is vaccinated. The nodes to be vaccinated are chosen at random among those that are not yet vaccinated, no matter if they are susceptible, infected or recovered. Once a node is vaccinated it can no longer become infected or infect. The mathematical formulation of probabilities remains the same given in Def. 1.*

The graph on which we initially test this kind of pandemic is the same 500 nodes preferential attachment random graph with average degree 6 used in the previous step.

The vaccination happens uniformly around the population with a cumulative percentage coverage given by the vector

$$vac = [0, 5, 15, 25, 35, 45, 55, 60, 60, 60, 60, 60, 60, 60, 60].$$

The value in position w means that arrived at the w^{th} week that percentage of the entire population was vaccinated. The percentage vaccinated during the w^{th} week is given by $vac_{w+1} - vac_w$.

To simulate the pandemic we have to modify the algorithm defined in Sec. 1.a to include the vaccination:

- for each simulation i
  - initialize the state $X_{i,j,0} = S \ \forall j \in \mathcal{V}$
  - randomly choose 10 infected, and set their state $X_{i,j,0} = I \ \forall j \in \ infected$
  - for each week w
    - compute the amount of vaccinated nodes as $v = \#nodes \cdot (vac_{w+1} - vac_w)/100$
    - choose at random v nodes among those that are not already vaccinated, set their state $X_{i,j,w} = V \ \forall j \in vaccinated$
    - for each node n
      - if $X_{i,n,w-1} = S$ with probability $p = 1-(1-\beta)^m$, where m is the number of its neighbors infected in $w-1$ not even vaccinated in week w, it becomes infected ($X_{i,n,w} = I$)
      - else if $X_{i,n,w-1} = I$ with probability $p = \rho$ it becomes recovered ($X_{i,n,w} = R$)

We kept track of the number of new infected each week for each simulation and the number of people in each state in each week for each simulation. Averaging these statistics we got the plots in Fig. 5.

Comparing these results with the results we obtained in Sec. 1.c without vaccination we can immediately see the positive effect of the vaccination. The peak of new infected reaches just 55 while before it reached 104, the week with most infected people has 67 currently infected nodes while before this value reached 141. The amount of people that made contact with the pandemic, sum of new infected, with the vaccination is just 194 while previously it overcame 410.

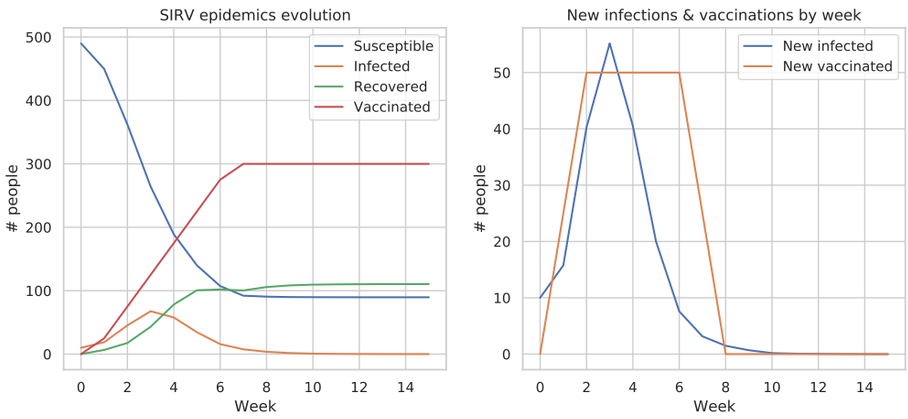

Figure 5: Evolution of the SIRV pandemic on the 500 nodes preferential attachment random graph and amount of new infected each week, results averaged over 100 simulations.

1.e H1N1 pandemic in 2009, Sweden

By now we have all the elements we need to simulate the pandemic on the population of Sweden in 2009. To simplify the model we divide by 10^4 the population, so that each node in our graph will represent 10000 people, we assume the preferential attachment graph correctly represents the way the society is connected and we limit the examination between week 42, 2009 and week 5, 2010.

Given these assumptions we create a preferential attachment random graph with 934 nodes. The cumulative percentage of vaccinated people in this case is given by the vector:

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

For this part of the exercise the amount of new infected in each week is given and equal to:

$$inf = [1, 1, 3, 5, 9, 17, 32, 32, 17, 5, 2, 1, 0, 0, 0, 0]$$

Knowing this infection distribution we can try to approximate all the parameters we miss: k, β, and ρ.

To approximate the real value of those parameters we implemented the following algorithm:
Given the initial values (k_0, β_0, ρ_0) and $(\Delta k, \Delta\beta, \Delta\rho)$

- define the parameters (k, β, ρ) with $k \in \{k_0 - \Delta k, k_0, k_0 + \Delta k\}$, $\beta \in \{\beta_0 - \Delta\beta, \beta_0, \beta_0 + \Delta\beta\}$, and $\{\rho_0 - \Delta\rho, \rho_0, \rho_0 + \Delta\rho\}$

- for each triplet (k, β, ρ)
  - generate a preferential attachment random graph with the function `pa_random_graph` with 934 nodes and degree k
  - simulate for 10 iterations the SIRV pandemic on the graph
  - average the vector of new infected by week over the 10 iterations $\hat{inf}$
  - compute the RMSE between the simulate infected vector and the correct one as:

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{inf}_t - inf_t)^2}$$

    where $T = 15$ the number of weeks over which we study the pandemic
- update k_0, β_0 and ρ_0 with the values of the optimal configuration
- if the optimal configuration is the same in 2 consecutive iterations
  - divide by 2 the value of Δk, $\Delta\beta$ and $\Delta\rho$
  - if you have done this operation less than 4 times repeat the algorithm, else exit.

With this algorithm we have tested some initial configurations as reported in Tab. 1

(k_0, β_0, ρ_0)	$(\Delta k, \Delta\beta, \Delta\rho)$	Best parameters' set found	Best RMSE
$(10, 0.3, 0.6)$	$(1, 0.1, 0.1)$	$(10, 0.2, 0.45)$	4.841
$(10, 0.3, 0.6)$	$(2, 0.1, 0.1)$	$(11, 0.125, 0.625)$	4.968
$(10, 0.5, 0.5)$	$(2, 0.1, 0.1)$	$(4, 0.472, 0.325)$	5.247

Table 1: Some test of the algorithm to find the best parameters to fit the actual contagion statistics in Sweden, 2009.

Using the parameters found in the best case (first row Tab. 1) we could plot the graphs in Fig. 6. We can clearly notice that the approximated curve describing new infections does not fit very well the real curve, but this can be explained since we are using just 10 simulations to reduce the time required. Using an higher number of simulations could give us a better approximation, but since preferential attachment is a very simple model the results could also be much worse. All the consideration made in Sec. 1.d about the positive effect of vaccination still hold here.

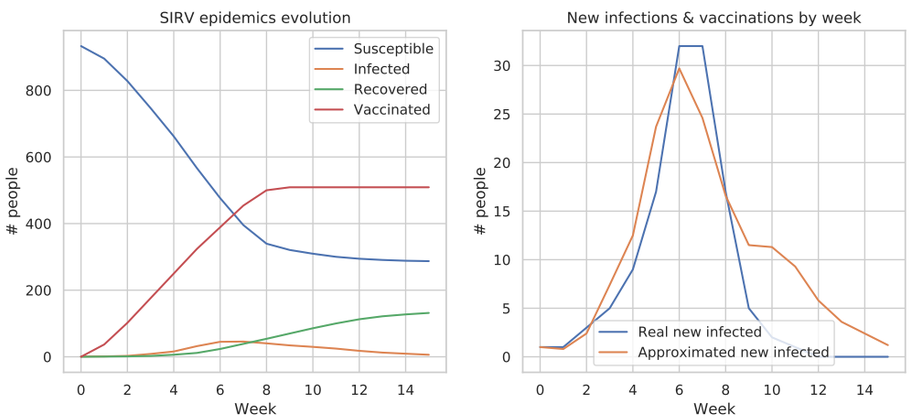

Figure 6: SIRV evolution on the 934 nodes population modeling Sweden.

1.f Challenge

Following the consideration above we tried a new kind of random graph aiming at improving the underlying model of the society. To do so we tried the small world random graph, here its definition:

Definition 6 (Small world model) *This model starts with a symmetric k-regular random graph to model the close connections among neighbors. Than with a probability called α adds to each node a long-distance connection. This kind of model was introduce to have small diameter and high clustering.*

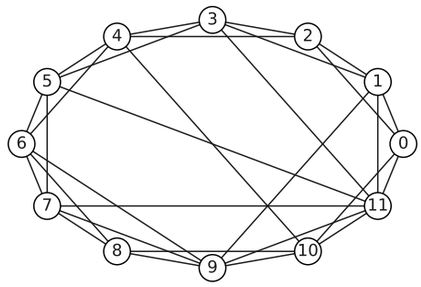

Figure 7: Small-world random graph with 12 nodes average degree 5 and $\alpha = 0.5$.

We have tested this new model with the same parameter selection technique defined in Sec. 1.e just adding to the algorithm a parameter α_0 and $\Delta\alpha$ behaving as the others. Since the addition of this new parameter would increase the number of tests per iteration to $3^4 = 81$ we reduced the number of Δx reductions to 2.

In Tab. 2 we report the results obtained with some starting configuration we tested. Despite the new model the best result is worse than before achieving no less than 6.324.

$(\alpha_0, k_0, \beta_0, \rho_0)$	$(\Delta\alpha, \Delta k, \Delta\beta, \Delta\rho)$	Best parameters' set found	Best RMSE
$(0.7, 10, 0.3, 0.6)$	$(0.1, 1, 0.1, 0.1)$	$(0.7, 8, 0.35, 0.7)$	7.291
$(0.3, 10, 0.3, 0.6)$	$(0.1, 1, 0.1, 0.1)$	$(0.3, 11, 0.3, 0.6)$	7.477
$(0.5, 10, 0.3, 0.6)$	$(0.2, 2, 0.2, 0.2)$	$(0.5, 4, 1.0, 0.1)$	6.324

Table 2: Some test of the algorithm to find the best parameters to fit the actual contagion statistics in Sweden, 2009.

Since the change of model did not perform as expected we focused our attention on the parameters' selection algorithm. A possible alternative we came up with is the random search of parameters, this allows us to explore much more deeply the space. The algorithm we used is the following:

- for 1000 iterations
  - draw $k \in \mathcal{D}_k$, $\beta \in \mathcal{D}_\beta$, and $\rho \in \mathcal{D}_\rho$
  - generate with `pa_random_graph` the graph with 934 nodes and average degree k
  - simulate for 10 iterations the SIRV pandemic on the graph
  - average the vector of new infected by week over the 10 iterations $\hat{inf}$
  - compute the RMSE as:

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{inf}_t - inf_t)^2}$$

    where $T = 15$
  - if the RMSE is the lowest ever reached save the triplet (k, β, ρ).

We can decide the distribution over which to draw the parameters for the simulation to indicate a prior knowledge about the distribution. We tested both the uniform distribution and the Normal distribution, here the results:

- k drawn from $U(5, 15)$, β drawn from $U(0.01, 0.99)$, and ρ drawn from $U(0.01, 0.99)$: best configuration found $(k, \beta, \rho) = (15, 0.1, 0.67)$ with $RMSE = 3.970$
- k drawn from $\mathcal{N}(\mu = 10, \sigma = 1), \in [5, 15]$, β drawn from $\mathcal{N}(0.3, 0.1) \in [0.01, 0.99]$, and ρ drawn from $\mathcal{N}(0.6, 0.1) \in [0.01, 0.99]$: best configuration found $(k, \beta, \rho) = (11, 0.170, 0.733)$ with $RMSE = 4.554$.

In Fig. 8 we can compare the results obtained with the different algorithm tested. We can clearly see that none of the curves fits very well the original one, this may be due to the very low number of simulation that we used (10) or to the wrong underlying model described by the graph.

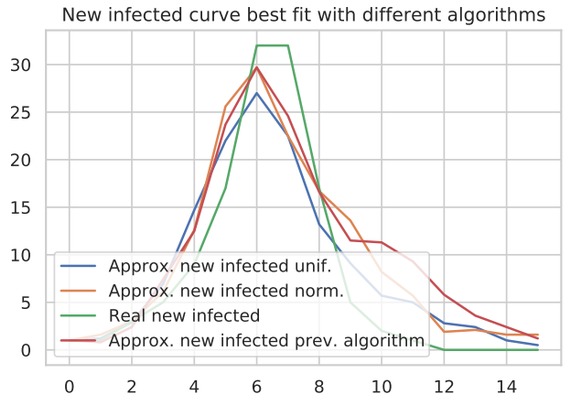

Figure 8: Comparison between the curves found minimizing the RMSE with different algorithms.

Exercise 2 Coloring

In this second exercise we study a coloring problem over 2 different graphs. First we test the algorithm on a simple line graph with 10 nodes, than we move to a more realistic scenario in which we try to use the coloring problem to reduce the interference between Wi-Fi frequencies.

2.a Coloring on a line graph

In this first step of the exercise we study a line graph with 10 nodes as shown in Fig. 9.

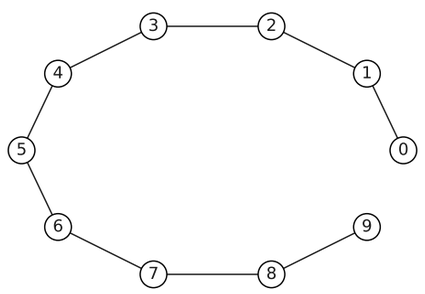

Figure 9: Line graph on which we will run coloring.

But before starting we miss one last thing:

Definition 7 (Coloring) *The graph coloring problem concerns assigning to each node of an undirected graph a color chosen from a finite set $\mathcal{A}$ so that no two neighbors have the same color. If it exists such assignment is called k-coloring and the corresponding graph is called k-colorable.*

We define the set of available colors

$$\mathcal{A} = \{\text{red},\ \text{green}\}$$

and in this context we initialize the state of each of the nodes in the graph as *red*, so the state of the system is $X_i(0) = \text{red}\ \forall i \in \mathcal{V}$. At every consecutive discrete time a node $I(t)$, chosen uniformly at random, wakes up and updates its color from a probability distribution given by:

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

where the cost function is given by $c(s, X_j(t)) = 1$ if $X_j(t) = s$, otherwise $c(s, X_j(t)) = 0$. As function $\eta(t)$ we used $\frac{t}{100}$.

To consider how close to a solution the actual state is we used the utility function

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

when its value reaches 0 a solution is found.

To implement this kind of dynamic we implemented the following algorithm: given the state $X(0)$

- choose uniformly at random a node i
- compute for each state $a \in \mathcal{A}$ its probability
- choose the future state at random following the probabilities computed above
- update the state assigning the new state to the node i
- compute the utility of the state $X(t+1)$
- if the utility is 0 stop, otherwise go to the next iteration.

Running this algorithm on the line graph defined above we obtained the dynamic described in Fig. 10. The number of iterations required by the algorithm to end is 138 as reported as title of the last graph. Since the utility reaches 0 we know that the line graph is 2-colorable, which was expected since it is a bipartite graph.

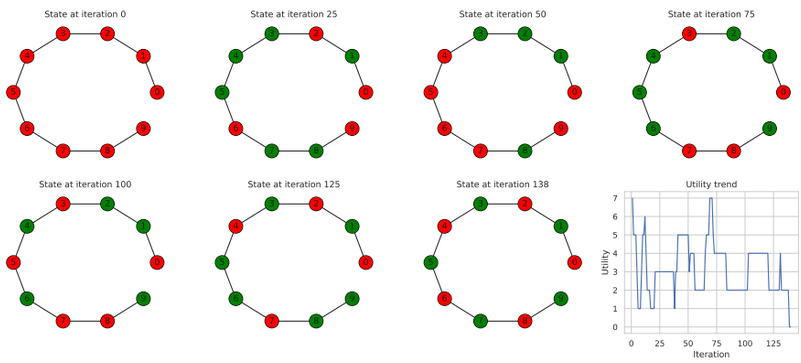

Figure 10: Initial, some of the intermediate and final state of the coloring dynamic on the line graph.

2.b Assign WiFi channels via coloring

WiFi frequency channels should not overlap to guarantee a more stable and reliable connection to routers. This context can be easily modeled as a coloring problem. We read the coordinates and the adjacency matrix from the given `.mat` files.

In this second part of the exercise the possible states/colors available are

$$\mathcal{A} = \{\text{red}, \text{green}, \text{blue}, \text{yellow}, \text{magenta}, \text{cyan}, \text{white}, \text{black}\} = \{1, 2, 3, 4, 5, 6, 7, 8\}.$$

Also the cost function was modified as follows

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

This cost function weights not just the fact that two close router use the same frequency, but also the fact that two close routers use frequencies that are not so far apart.

We let run, in this new context, the same algorithm defined above with the same function $\eta(t) = t/100$. On this new graph the algorithm was unable to find an exact solution with utility 0, over 5000 iterations the algorithm reached its lowest point with utility 4 at iteration 981. We can conclude that this graph is not 8-colorable. In Fig. 12 you can see the initial state and the best configuration reached, in Fig. 11 you can see a plot of the utility across the 5000 iterations.

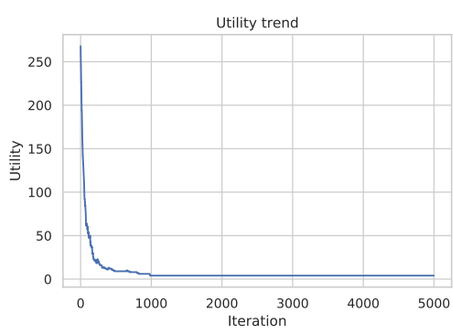

Figure 11: Utility trend running 5000 iteration of the algorithm on the WiFi graph.

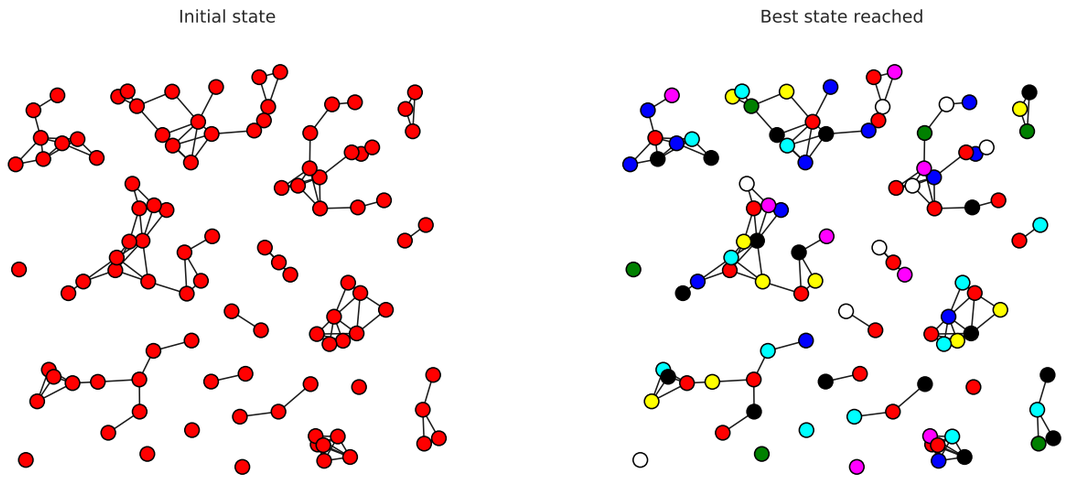

Figure 12: Initial configuration and best configuration reached running the algorithm.

2.c Challenge

In this final part of the exercise we try to modify the function $\eta(t)$ looking to the effects this function has on the utility trend. We have done this process on both the graph used in Sec. 2.a and Sec. 2.b.

Line graph

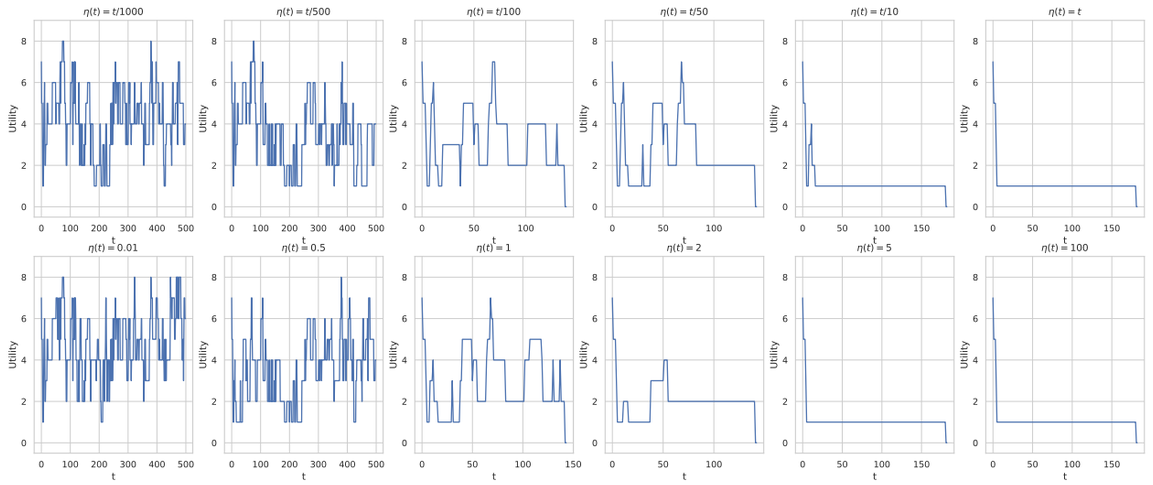

Figure 13: Utilities generated by several functions $\eta(t)$ on the line graph with 10 nodes.

In Fig. 13 we can notice that an eta function increasing too slowly (e.g., $\eta(t) = t/1000$) as well as functions that have a constant value which is too low (e.g., $\eta(t) = 0.01$) do not reach the 0 utility we know to be reachable on this graph. On the other side a function that is increasing too fast (e.g., $\eta(t) = t$) as well as a constant function that is too high (e.g., $\eta(t) = 100$) while producing a more linear trend require more iterations to reach utility 0. Tab. 3 summarizes the results.

$\eta(t)$	Lowest $U(t)$	#Iterations	$\eta(t)$	Lowest $U(t)$	#Iterations
$t/1000$	1	5	0.01	1	5
$t/500$	1	5	0.5	1	5
$t/100$	0	138	1	0	142
$t/50$	0	141	2	0	141
$t/10$	0	181	5	0	181
t	0	181	100	0	181

Table 3: Best utilities reached with different eta functions and the iterations requested to reach for the first time the lowest value.

2.c.1 WiFi channels

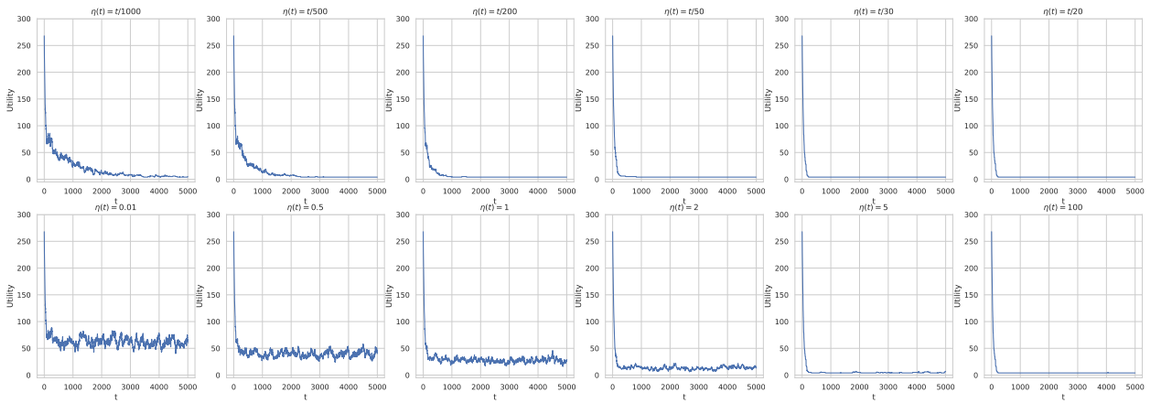

Figure 14: Utilities generated by several functions $\eta(t)$ on the WiFi graph.

Looking at Fig. 14 we can come to similar conclusions to the ones we had on the line graph. Eta functions increasing too slowly (e.g., $\eta(t) = t/1000$) or with constant value too low (e.g., $\eta(t) = 0.01$) require more iterations to reach the lowest value or do not even reach it (constant function). On the other side eta functions that are faster to increase (e.g., $\eta(t) = t/20$) or with higher constant value (e.g., $\eta(t) = 100$) require less iterations to reach the lowest utility. Tab. 4 summarizes the results.

$\eta(t)$	Lowest $U(t)$	#Iterations	$\eta(t)$	Lowest $U(t)$	#Iterations
$t/1000$	4	3456	0.01	40	4575
$t/500$	4	2329	0.5	24	3532
$t/200$	4	978	1	16	4863
$t/50$	4	821	2	6	2671
$t/30$	4	218	5	4	318
$t/20$	4	218	100	4	218

Table 4: Best utilities reached with different eta functions and the iterations requested to reach for the first time the lowest value.