Homework 3

Giuseppe Galilei s295620

15 January 2022

Preface

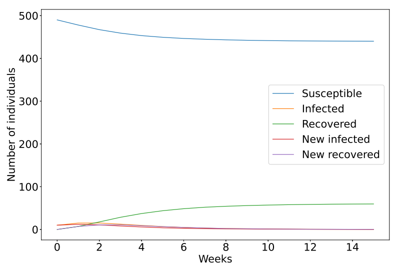
The following homework is submitted alongside a Jupyter notebook. In the submitted zip archive are present a folder 'images' which is empty and a folder 'results' which includes some empty folders, the notebook uses these folders to save data and will be necessary in case of execution.

Ideas have been exchanged with Beatrice Alessandra Motetti (s287618), although all code implementations have been carried out individually.

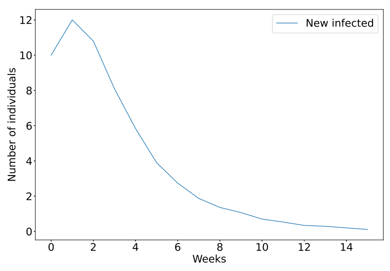
Problem 1.1

It is asked to simulate a SIR epidemic on symmetry k-regular graph $G = (V, \epsilon)$ with $|V| = 500$ nodes and $k = 4$, the graph has been constructed using the function *generate_symmetryK_regular_graph*: it takes as input the values n and k, computes a list of offsets based on k and then calls the function *circulant_graph* from the Networkx library, this function returns a circulant graph of n nodes such that each node i is connected to nodes $(i+x)$ mod n and $(i-x)$ mod n, being x the offset value.

Note that, to easily refer to the status of agents in the algorithms, a convention has been adopted to associate a certain status to an integer number:



(a)



(b)

Figure 1: SIR without vaccination for problem 1.1

- Susceptible: 0
- Infected: 1
- Recovered: 2
- Vaccinated: 3 (used later)

To simulate the SIR epidemic the function *simulate_SIR* is used, it takes as input the graph G, the number of weeks the simulation should run, the parameters β and ρ and the initial number of infected. An array *state* is used to store the current status of agents in the network, the initial configuration is constructed by randomly placing elements of value 1 (infected) in an array of zeros (susceptible), respecting the chosen number of initial infected.

Then, iteratively for each week:

1. The current state is stored as *state_old* to represent the state at the beginning of the week
2. For each agent, based on its state value stored in *state_old*, infection or recover is simulated respectively if the agent was in a susceptible or infected state, following the given transition probabilities:

   $$P\left(X_i(t+1) = \mathrm{I} \mid X_i(t) = \mathrm{S}, \sum_{j\in\mathcal{V}} W_{ij}\delta^1_{X_j(t)} = m\right) = 1-(1-\beta)^m$$
   $$P\left(X_i(t+1) = \mathrm{R} \mid X_i(t) = \mathrm{I}\right) = \rho$$

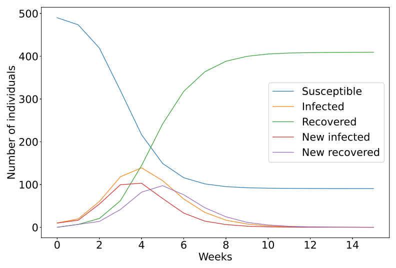
   where $\sum_{j\in\mathcal{V}} W_{ij}\delta^I_{X_j(t)}$ is the number of infected neighbors for node i.

   Then its current state is updated in the *state* array and the change in the number of infected or recovered individuals is recorded in the *changes* array.
3. At the end of the week the count of agents in each state and the count of state changes is stored in *week_stats* using the auxiliary function *compute_week_stats*.

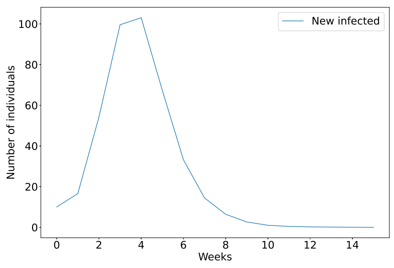
By simulating the SIR epidemic on the symmetry k-regular graph G with 500 nodes, $k = 4$, $\beta = 0.3$, $\rho = 0.7$, for 15 weeks and with 10 initial infected, on average over 100 iterations, it is obtained the behaviour showed in figure 1.

Problem 1.2

It is asked to generate a random graph using preferential attachment, to do so the function *generate_RandomGraph_PA* is proposed. This function takes as input the number of nodes and the average degree k wanted for the graph. An initial graph G, which is complete and composed of $k+1$ nodes, is constructed using the *complete_graph* function from NetworkX. A flag is used to tell if k is odd, in such case for the following iterations the value of the variable c alternates between $int(k/2)$ and $int(k/2)+1$. So, iteratively for each new node i that it is required to add, the following steps are performed:



(a)



(b)

Figure 2: SIR without vaccination for problem 2

1. an array containing the degree of each node in G is constructed
2. the degree array is normalized to obtain a probability distribution
3. c nodes are selected among the nodes of G, based on such distribution
4. c edges are added to G, linking the node i to each of the selected nodes

Using the algorithm, it is tried to generate a graph with 500 nodes and average degree equal to 6, by computing the average degree of the resulting graph, the function is confirmed to work.

Problem 2

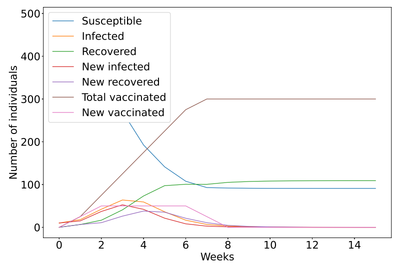
Using the functions created for the previous points, it is required to simulate a SIR epidemic on a graph G with 500 nodes and average degree 6. The simulation uses as parameters $\beta = 0.3$, $\rho = 0.7$, a number of initial infected equal to 10 and runs for 15 weeks. The average behaviour obtained over 100 simulations is shown in figure 2.

Problem 3

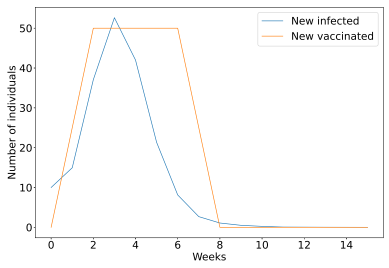
It is now required to simulate a SIR epidemic with the introduction of vaccinations, under the same conditions introduced for problem 2. To model such situation the original function *simulate_SIR* has been slightly modified with the introduction of a vaccination phase for each weekly iteration. Vaccinations need to be performed according to the values in

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

which in the code is referred to as *weekly_vax_percentage*, this array represents for each week what is the percentage of population that has received



(a)



(b)

Figure 3: SIR with vaccination for problem 3

vaccination. So the initial configuration is constructed with a certain number of infected people and a certain number of vaccinated ones, according to the first value in *weekly_vax_percentage*, in this case the 0% of the population. (Note that in the code it is avoided to vaccinate the initial infected individuals, such choice will be explained in the solution of problem 4 and does not influence simulations for this problem)

Then, at the beginning of each week i, vaccination is performed on a percentage of population equal to the difference between the percentages of vaccinated people in weeks $i+1$ and i. A corresponding number of people is chosen randomly among the ones who have yet to receive vaccination, and their state is changed to 3 (which represents the "vaccinated" state). Then the array *people_to_vaccinate* is updated.

Details about the evolution of the epidemic and of the vaccination campaign are, as before, stored in the *changes* array and then in *week_stats*.

The simulation results are shown in figure 3. It can be seen how the vaccination curve respects the requirements and is effective in slowing down the epidemics, less people get infected compared to the simulation without vaccination, shown in figure 2.

Problem 4

It is asked to perform a gradient based search on parameters k, β and ρ to improve the estimation of the SIR model with vaccination, with regards to the actual number of newly infected individuals in the H1N1 pandemic in Sweden, in 2009.

It is given a new *weekly_vax_percentage* array.

In the algorithm, the different phases of the search are orchestrated by a *stage* value, which is initially set to 0 and during execution will assume values $0, 1, 2$, each activating a different scenario.

To perform the search the parameters are initially set to $k_0 = 10$, $\beta_0 = 0.3$, $\rho_0 = 0.6$, with $\Delta k = 2$, $\Delta\beta = 0.1$, $\Delta\rho = 0.1$. For each parameter x, the respective parameter space consists of $\{x - \Delta x, x, x + \Delta x\}$ and it is created by using the function *param_space* which returns the parameter space making sure constraints for each parameter are satisfied (for example k needs to be integer and non-negative, while ρ and β are defined in $[0, 1]$).

The algorithm starts by creating the parameter space for each parameter, then all possible combinations of parameters are computed and, for each one, the epidemics is simulated 10 times. The results are then averaged and the value about new weekly infected is compared, by means of RMSE, to the actual value given by the given array $I_0(t)$ (in the code referred to as $true\_newly\_infected$). If the RMSE value is lower than the previous best, the current combination of parameter values is saved as the best one and the *improving* flag is set to True. Once all combinations have been evaluated, if one of them resulted in an improvement, the value of parameters in such best configuration become the new starting parameters and the *stage* value is set to 0. Otherwise the *stage* value is incremented, if such value is less than 2 it is still possible to update the values for the *parameters delta* (in this case dividing them by half) and continue running the search, otherwise it means that even this action has been performed without obtaining an improvement, then the search ends and the best found parameters are returned.

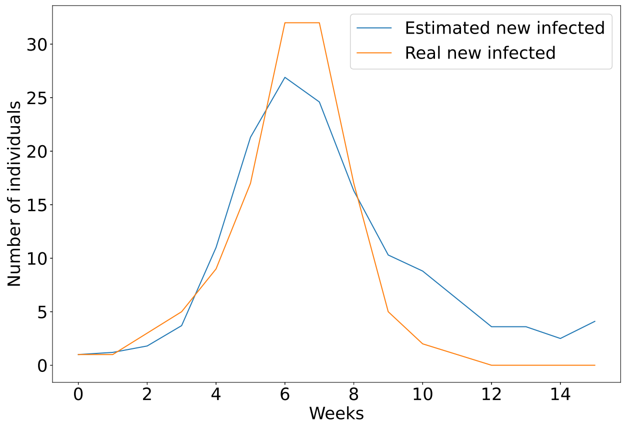
The SIR simulations have been run with one initial infected, in order to be coherent with the first value of the array $I_0(t)$. Also such first infected has been preserved from vaccination during the definition of the initial configuration, this has been done to avoid the possibility of having simulations where the epidemics does not diffuse at all due to the fortunate case of vaccinating the only infected individual, which is not coherent with reality and would potentially end ruining the results of the parameter search.

The best result, with an RMSE of 4.14, has been obtained with parameters $k = 11$, $\beta = 0.175$, $\rho = 0.425$ The behaviour is shown in figure 4.

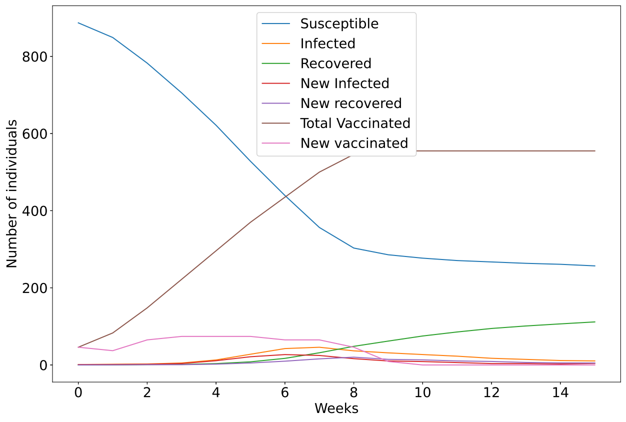
It is important to note that a certain variability has been noticed in the RMSE values, for the same set of parameters over several simulations, thus the performed search might have been flawed by not reliable RMSE results and a better parameter set could exist. This could be the consequence of the fact that simulations results have been averaged over only 10 simulations, due to computational time contraints.

Problem 5

Several alternative ways of approaching the problem have been tried.



(a)



(b)

Figure 4: Best estimated behaviour for the Sweden H1N1 pandemic, problem 4

Small world network

A first attempt has been made by substituting the graph which was previously generated by using preferential attachment, with one adopting the small world model. This choice has been done because of, usually, the better capability of small world networks to model real networks, with a small diameter and a high clustering. The new graph has been created using the NetworkX function *newman_watts_strogatz graph*, which takes as arguments the number of nodes n, a parameter k and a probability p. Such function starts by creating a ring of n nodes, then each node is connected to its k nearest neighbors ($k-1$ if k is odd). Then, for each node i, a shortcut (i.e. edge) connecting it to a randomly chosen node j, is created with a probability p. This graph is a variant of the *Watts-Strogatz* graph, with the difference that here new shortcut edges do not substitute any pre-existing edge.

The parameter p, along with the usual k, β and ρ, has been optimized in a gradient based search, as before. To perform the search the parameters are initially set to $k_0 = 10$, $\beta_0 = 0.3$, $\rho_0 = 0.6$,$p = 0.5$, with $\Delta k = 2$, $\Delta\beta = 0.1$, $\Delta\rho = 0.1$, $\Delta p = 0.1$.

The best combination, with an RMSE of 5.77, has been obtained with $k = 10$, $\beta = 0.425$, $\rho = 0.575$ and $p = 0.525$. The estimated behaviour can be seen in figure 5.

MAE

Another attempt has been made by substituting the RMSE metric with the MAE (mean absolute error) to estimate the number of newly infected each week. This time using the graph generated via preferential attachment. To perform the search the parameters are initially set to $k_0 = 10$, $\beta_0 = 0.3$, $\rho_0 = 0.6$ with $\Delta k = 2$, $\Delta\beta = 0.1$, $\Delta\rho = 0.1$.

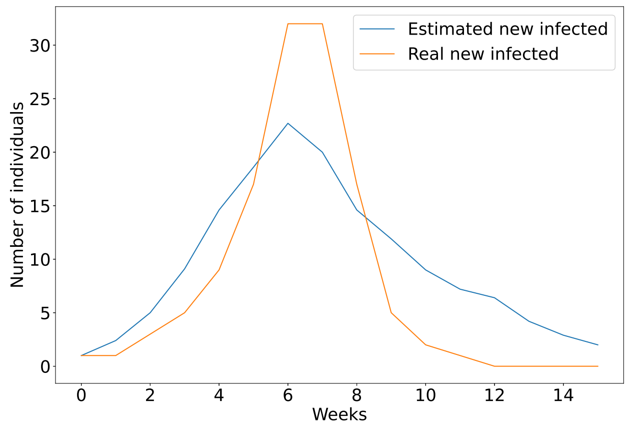
The best combination, with an MAE of 2.9, has been obtained with $k = 10$, $\beta = 0.2$, $\rho = 0.75$. The estimated behaviour can be seen in figure 6.

Random search

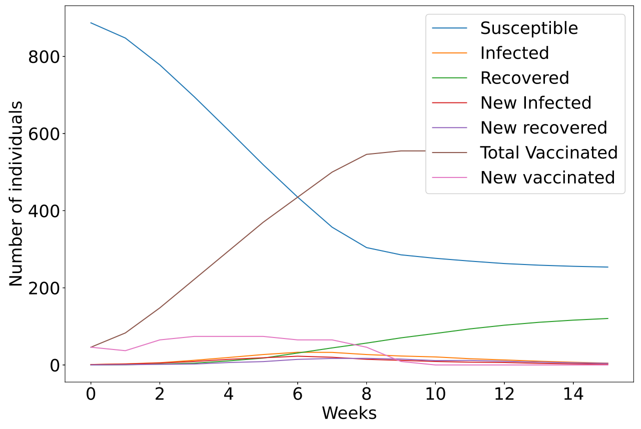
In the end it was tried to change the parameter search approach and adopt a random based search. The idea is that given a parameter x and a value Δx, a value is sampled randomly in the range $\{x - \Delta x, x + \Delta x\}$ (accordingly to parameter constraints). This is done for all parameters (k, β and ρ in this case), obtaining in the end a combination which is used for simulations, as before. The code is fairly similar to the original presented for problem 4, the main difference is that now, because there is not a list of combinations on which to iterate, it is given a fixed number of iterations (in this case 30 has been chosen) and in each one a combination is generated as described before.

This approach did not provide better results than the original gradient based search, however proved to be generally faster in providing fairly good results.

As an example, a run of the random search with initial parameters $k_0 = 10$, $\beta_0 = 0.5$, $\rho_0 = 0.5$ and $\Delta k = 5$, $\Delta\beta = 0.4$, $\Delta\rho = 0.4$, resulted in a best configuration $k = 15$, $\beta = 0.12$, $\rho = 0.67$ with an RMSE of 5.6 and an estimated behaviour shown in figure 7.

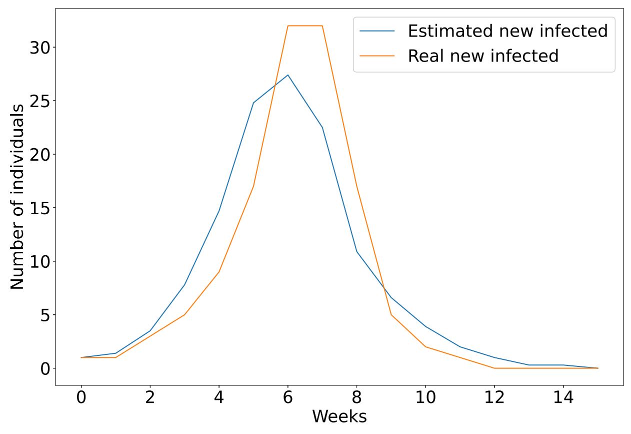


(a)

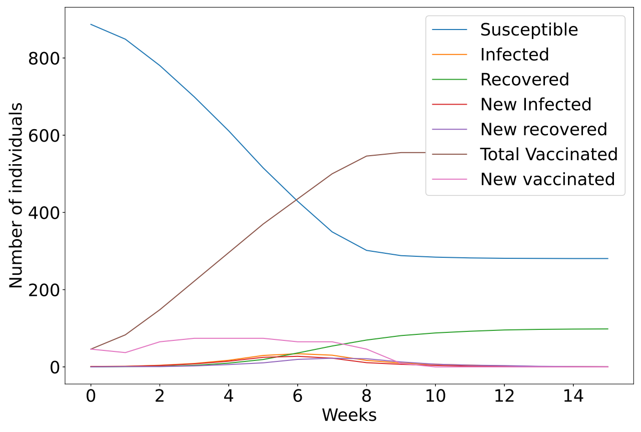


(b)

Figure 5: Best estimated behaviour with Small World, problem 5

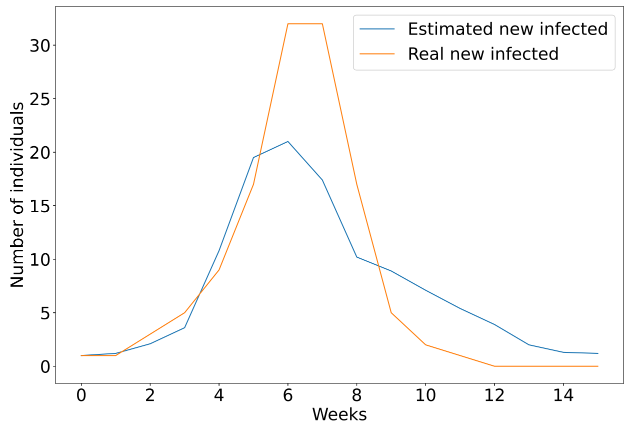


(a)

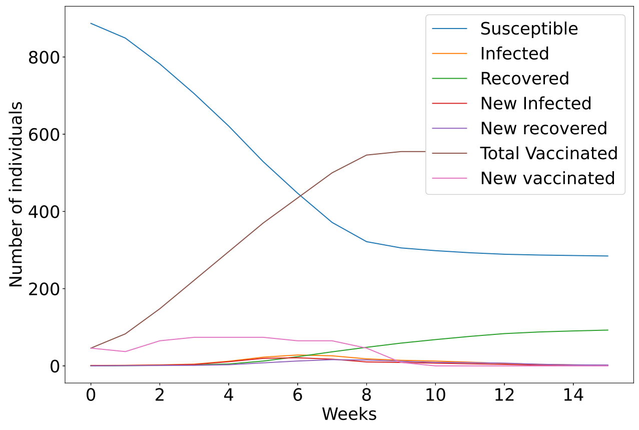


(b)

Figure 6: Best estimated behaviour with MAE as metric, problem 5



(a)



(b)

Figure 7: Best estimated behaviour obtained using random search, problem 5