Third homework Network Dynamics and Learning

[redacted]

[redacted]

Luca Pesce
Politecnico di Torino
[redacted]

***Abstract*—The first part of this homework explores network dynamics and learning by simulating the Influenza H1N1 2009 pandemic in Sweden. Divided into four parts, it involves simulating a pandemic on known and random graphs, estimating parameters, and comparing simulated and real pandemic data. The second one explores distributed learning in graph coloring, progressing from a line graph to wi-fi channel assignment for routers.**

I. Influenza H1N1 2009 Pandemic in Sweden

A. Epidemic on a Known Graph

The homework begun by creating a symmetric k-regular graph with 500 nodes and k=4. This graph was then converted into a sparse adjacency matrix to optimize operations during simulation. In particular, this type of matrix stores only non-zero values within it, in which case it is very convenient because the adjacency matrix has most of the values equal to zero. Thus when we calculate the number of infected neighbors for each node as Lambda @ (states == 1), where "Lambda" is the sparse adjacency matrix and "states" is a vector representing the state of each node (Infected = 1), the calculation will be very smooth. To simulate the pandemic we employed a simplified SIR model (one of the simplest compartmental models) for the simulation, where nodes could be in one of these states: Susceptible (S), Infected (I), or Recovered (R). The parameters β and ρ were set to 0.3 and 0.7, respectively. The simulation was conducted considering 15 weeks with 100 iterations. The following results were plotted:

- Average number of newly infected individuals each week. Figure 1
- Average total number of susceptible, infected, and recovered individuals each week. Figure 2

Through this simulation, we could visualize and analyze the progression of an epidemic on a symmetric k-regular graph. Utilizing sparse matrices allowed for an efficient execution of the simulation while ensuring accurate results. Such analyses can be immensely valuable in understanding the spread of diseases in complex networks and devising effective mitigation strategies.

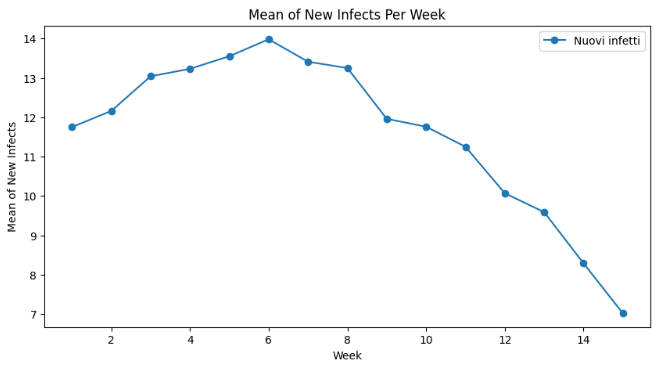

Fig. 1. amplitude over time and frequency

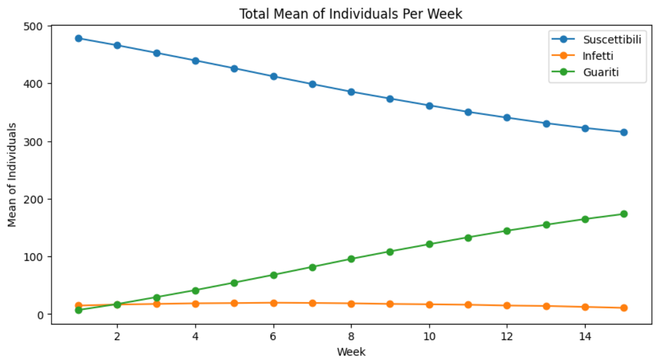

Fig. 2. amplitude over time and frequency

B. Generate a Random Graph

The objective of this exercise is to generate a large random graph (with at least 900 nodes) using the preferential attachment model. This model facilitates creating a network with an average degree close to k. The process commences by creating a complete graph **G** with k+1 nodes. Subsequently, the degree sequence **w** is computed for each node in the graph. After initializing the starting graph, additional nodes are added until reaching a total of 900 nodes. During this phase, each new node connects to k/2 existing nodes in the graph. In instances where k is odd, a specific logic is employed to ensure that the average degree of the graph approximates to the requested value. This is achieved by alternating the number of connections for new nodes between

$$(k-1)//2 \tag{1}$$

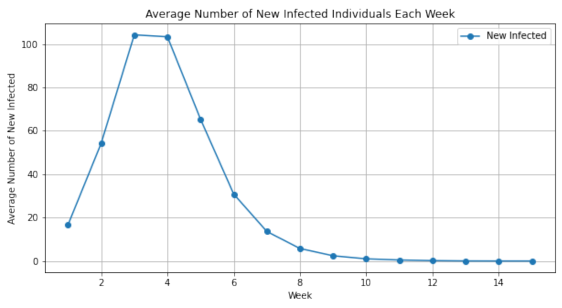

Fig. 3. amplitude over time and frequency

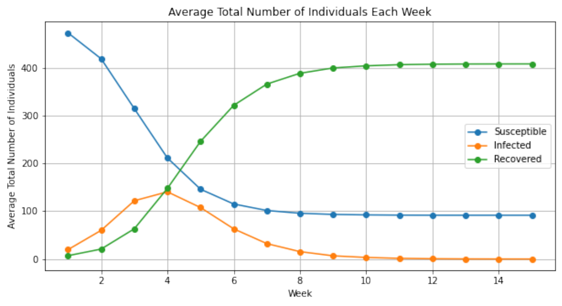

Fig. 4. amplitude over time and frequency

and

$$(k+1)//2 \tag{2}$$

. The verification process confirmed that the calculated degrees matched the degree sums in the adjacency matrix and the average degree aligned with expectations.

C. Simulate a Pandemic Without Vaccination

The objective of this simulation exercise is to study the spread of an epidemic on a preferential attachment random graph, leveraging the discrete-time version of the SIR epidemic model. A preferential attachment random graph with 500 nodes and an average degree k=6 is generated. This graph construction ensures that nodes joining the network establish connections based on existing nodes' degrees. Using the SIR model, the epidemic's spread is simulated over 15 weeks. The parameters β=0.3 and ρ=0.7 govern the infection probability and recovery rate, respectively. A total of 10 nodes are initially infected randomly from the node set. The simulation is executed 100 times to capture variability and trends across different runs. The results are visualized by plotting the average number of newly infected individuals weekly and tracking the total counts of susceptible, infected, and recovered individuals over the 15-week period Figure 4. By leveraging the SIR model and specific parameters, the study provided valuable insights into disease propagation, emphasizing the network structure's influence on the epidemic's spread and control.

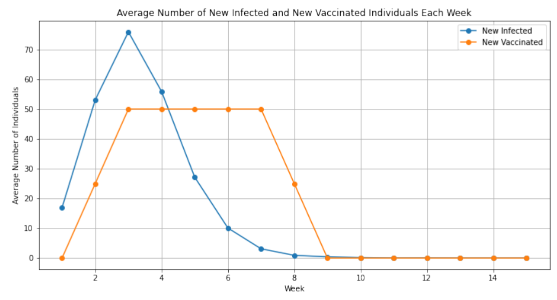

Fig. 5. amplitude over time and frequency

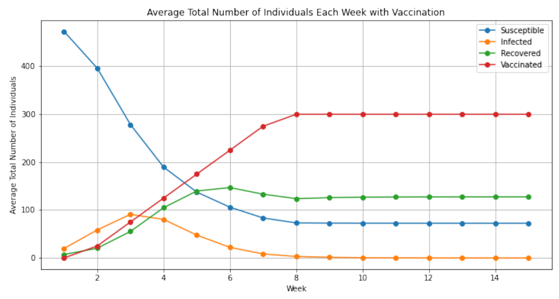

Fig. 6. amplitude over time and frequency

D. Simulate a Pandemic with Vaccination

In this simulation, a vaccination program is implemented with a vaccination rate that varies weekly. The goal is to show up the impact of vaccination on spreading and recovering time. The epidemic situation is the one described before. We used a k-regular symmetric graph G with 500 nodes and average degree k=6, converting the adjacency matrix to a sparse matrix for efficient computation. Individuals are selected for vaccination uniformly at random from the unvaccinated. Once vaccinated, an individual is immune for the rest of the simulation. The plots show the average number of new infections and vaccinations per week over 100 simulations. This provides insights into how the vaccination rate impacts the spread of the epidemic. Figure 5. The total number of individuals in each state (Susceptible, Infected, Recovered, and Vaccinated) is visualized over the 15-week period. Figure 6.

E. Simulate the Pandemic in Sweden

In this study, we aimed to estimate the social structure of the Swedish population and the parameters governing the spread of the H1N1 pandemic during the fall of 2009. The objective was to find the set of parameters that best matched the real pandemic, considering the number of newly infected and vaccinated individuals each week.

We utilized a gradient-based search algorithm to explore the parameter space of k, β, and ρ. The algorithm started with an initial guess of parameters, k_0, β_0, and ρ_0. We set $\Delta k = 1$, $\Delta\beta = 0.1$, and $\Delta\rho = 0.1$. The search was conducted over

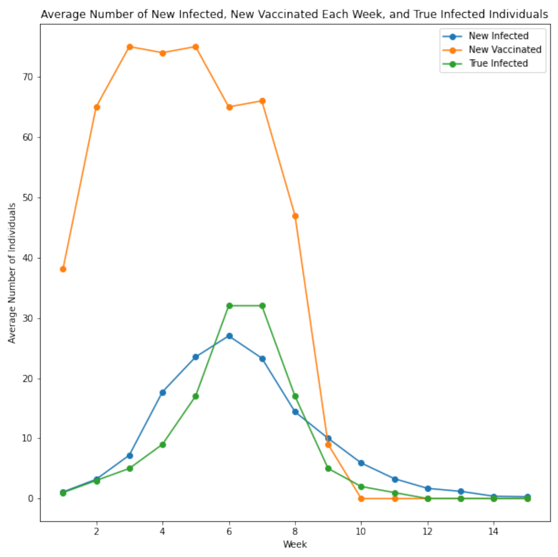

Fig. 7. Comparison of actual and predicted infected

the parameter space defined by $k \in \{k_0 - \Delta k, k_0, k_0 + \Delta k\}$, $\beta \in \{\beta_0 - \Delta\beta, \beta_0, \beta_0 + \Delta\beta\}$, and $\rho \in \{\rho_0 - \Delta\rho, \rho_0, \rho_0 + \Delta\rho\}$.

For each set of parameters, the algorithm performed the following steps:

1) Generated a random graph $G = (V, E)$ using the preferential attachment model with an average degree of k and $|V| = 934$ nodes.
2) Simulated the pandemic for 15 weeks on G, starting from week 42, using the method developed before with the specified vaccination scheme. This process was repeated $N = 10$ times, and the average number of newly infected individuals each week, $I(t)$, was computed.
3) Computed the root-mean-square error (RMSE) between the simulation and the real pandemic using the formula:

$$\text{RMSE} = \sqrt{\frac{1}{15}\sum_{t=1}^{15}(I(t) - I_0(t))^2}$$

The parameters k_0, β_0, and ρ_0 were updated to the set of parameters yielding the lowest RMSE.

After running the algorithm, we found the set of parameters that resulted in the minimum RMSE. The optimal configuration was $k = 8$, $\beta = 0.2$, and $\rho = 0.7$, which produced an RMSE of 4.1. Figure 7.

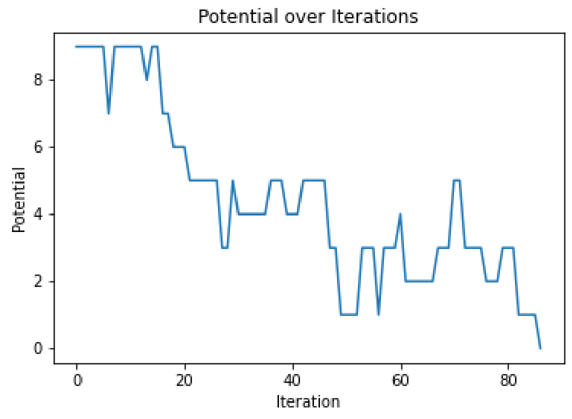

Fig. 8. Potential function - Line Graph

II. Coloring

Introduction

Graph coloring is a classic problem with various applications, including distributed learning in potential games. In this report, we explore a distributed learning algorithm for graph coloring on a line graph and its application to assigning wifi channels to routers in a network.

a) Line Graph with 10 Nodes

Consider a line graph with 10 nodes, each initially colored red. At each time step, a randomly chosen node updates its color based on a probability distribution. The learning dynamics involve a cost function and the inverse of noise, $\eta(t) = \frac{t}{100}$.

Learning Dynamics

The probability of a node i transitioning to color a is given by:

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

where $c(s, X_j(t))$ is the cost function and $\sum_j W_{ij} \cdot c(a, X_j(t))$ represents the number of neighbors of node i of color a.

Potential Function

The potential function $U(t)$ is defined as:

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

It represent the sum for every node of the neighbors with the same color of the node itself, divided by 2.

Simulation Results

Simulations were conducted, and the potential function over time was plotted. The results indicate the algorithm's convergence to a solution. Figure 8.

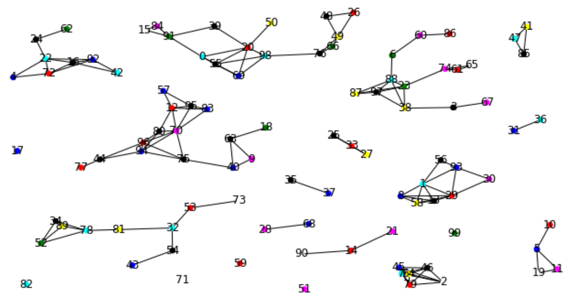
Fig. 9. Node coloring

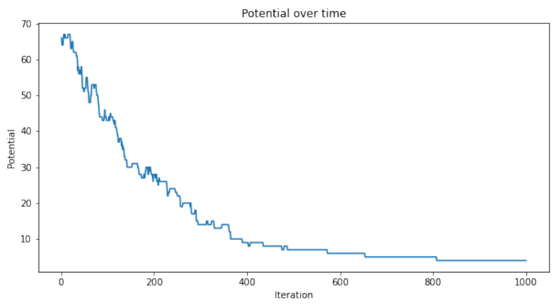

Fig. 10. Potential function - Wifi Graph

B) Wifi-Channels Assignment

Extend the graph coloring algorithm to assign wifi channels to routers in a network of 100 routers. The adjacency matrix and router coordinates are provided.

Cost Function

The cost function is designed to prevent interference between neighboring routers. It is given by:

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

In this case the potential function is the sum for every node i of two times the neighbors with the same color of node i plus the neighbors with colors adjacent to the node color i.

Simulation Results

Using $\eta(t) = \frac{t}{100}$, simulations were conducted, and potential functions for different test cases were plotted. The results demonstrate the effectiveness of the algorithm in finding near-zero potential solutions, in particular the minimum of the potential function we found is 4. Figure 9. Figure 10.

C) Optional: Experimenting with Different $\eta(t)$

Optional experiments involved testing different choices of $\eta(t)$, including constant values and other increasing functions. Figure 11.

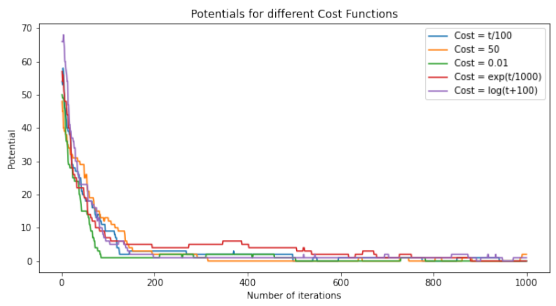

Fig. 11. Different Cost Functions

Conclusion

In conclusion, the distributed learning algorithm for graph coloring proved effective in both the line graph and wifi-channels assignment scenarios. The potential functions provided valuable insights into the convergence and optimality of the solutions.