Dynamical Processes on Graphs

Computer-Aided Simulations Lab - Lab L5

Henrique Pedro Roschel
Politecnico di Torino
Student id: S306718

I. Problem overview

The goal of the present activity is to generate a random graph based on the Erdős-Rényi model, analyze the distribution of its degree and simulate a dynamic process on the graph.

II. Proposed approach

A. Graph data structures and generation

First, classes `Node` and `Graph` were implemented. The first containing the node id, degree, current state and a counter of the times the node woke up. The latter, for a given pair (n, p), we were able to generate a graph according to the Erdős-Rényi model of a $G(n, p)$ graph, where the probability of each edge between the n nodes to be present is p. One instance of each `Node` is stored in an array, while the edges between them are stored in a set of tuples. After the initialization, isolated nodes are discarded.

B. Dynamical processes

Two dynamical models are implemented for the produced $G(n, p)$ graph: linear threshold and averaging model. The first one has a discrete set as domain for the nodes states ($X_v(t) \in \{0, 1\}$) and each node v wake up at rate $\lambda_v = 1$, while the latter has a continuous domain ($X_v(t) \in [0, 1]$) and each edge (v, w) wake up at rate $\lambda_{(v,w)} = 1$.

In the averaging model, when an edge (v, w) wakes up, the state of both vertices is updated according to the rule:

$$X_v(t^+) = \alpha X_v(t^-) + (1 - \alpha) X_w(t^-)$$
$$X_w(t^+) = \alpha X_w(t^-) + (1 - \alpha) X_v(t^-)$$

for $\alpha \in [0, 1]$.

As for the linear threshold model, when a node v wakes up, its next state will be 1 if the number of neighbors at state 1 is greater than r, and 0 otherwise, where $r \in \mathbb{N}$ is an input parameter.

Since these models relies on Poisson processes, we can use its properties for implementing the Future Event Set more efficiently. For the linear threshold model for example, instead of considering n nodes waking up at rate $\lambda_v = 1$, we consider that the event of any node to wake occurs also as a Poisson process, with rate $\lambda_n = \lambda_v \cdot n$. The same for averaging model, this time, if there are m edges, any edge wakes up with rate $\lambda_m = \lambda_{(v,w)} \cdot m$.

In both case, we generate the next wake up time based on these properties and select uniformly a random node/edge for the next event. By doing that, we will not store an event for each of the nodes/edges, but rather only the next event.

Finally, for each dynamic model, we impose a stop condition for the simulation, based on either, the total number of events or the number of times that each node has waken up.

III. Experiments and Results

A. Degrees distribution

For the considered graph $G(n, p)$, with large n (e.g. 100k) and np = constant, the degree distribution of the graph's nodes is Poisson with parameter $\lambda = np$. Given that, we can obtain the Q-Q plot (Figure 1) for the generated graph comparing its values to the theoretical values for the Poisson distribution.

B. Linear threshold model

After the graph is generated, we are able to simulate a linear threshold model, after setting the initial states of the nodes. In this case, the nodes are initialized with states 0 or 1, where the probability of it being 1 is given by the input parameter $p_{1,\text{init}} = P(X_v^{(0)} = 1)$. For this simulation, we may compare the initial and final distributions by looking at this probability, i.e. the average state of the nodes, at the beginning ($p_{1,\text{init}}$) and at the end of the simulation ($p_{1,\text{final}}$), after 100k iterations. For a range of inputs parameters r and $p_{1,\text{init}}$, the results obtained are shown Figure 2.

In this case, we see a linear correlation between the considered inputs. As expected the, for a low value for r, such as 1, the average state of the nodes tends increase during the simulation, while higher values of r for the linear threshold model cause the proportion of nodes at state 1 to decrease after the simulation.

C. Averaging model

With the same graph, we are also able to test the averaging model. This time, we consider two possible initial distributions for the nodes state: uniform and beta. For the beta distribution we consider two pairs of parameters: (i) $\alpha = 4.0$ and $\beta = 1.5$ and (ii) $\alpha = 1.5$ and $\beta = 4.0$. We run the model for $n = 100$k wake up events and compare the initial and final distribution of the nodes' states on the histograms in Figure 3.

In the histograms, we see that, after 100k iterations, the trend is to bring the vertices' states towards the initial distribution mean, which is 0.5 for the first two scenarios and, on the third and fourth cases, is shifted to the right and to left, respectively. The parameter α controls the speed to which the process occurs, since for α closer to 0.5, the states reach the average value faster.

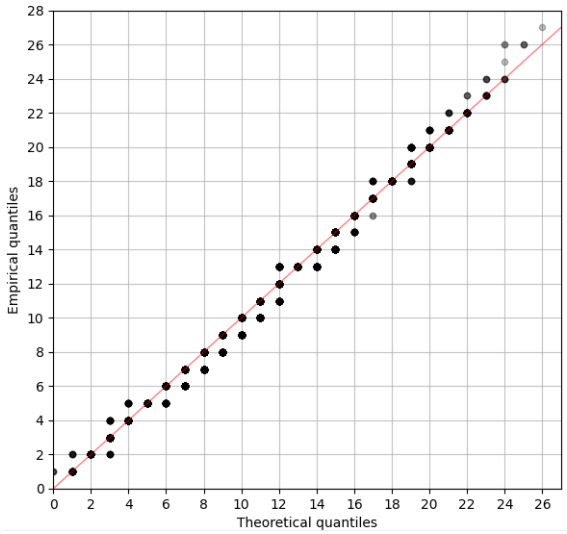

Fig. 1: Q-Q plot for the graph degree distribution.

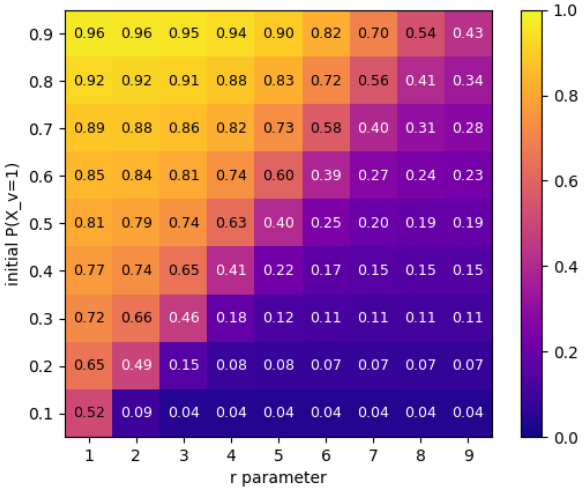

Fig. 2: Average final state of nodes for different inputs $p_{1,\text{init}}$ and r

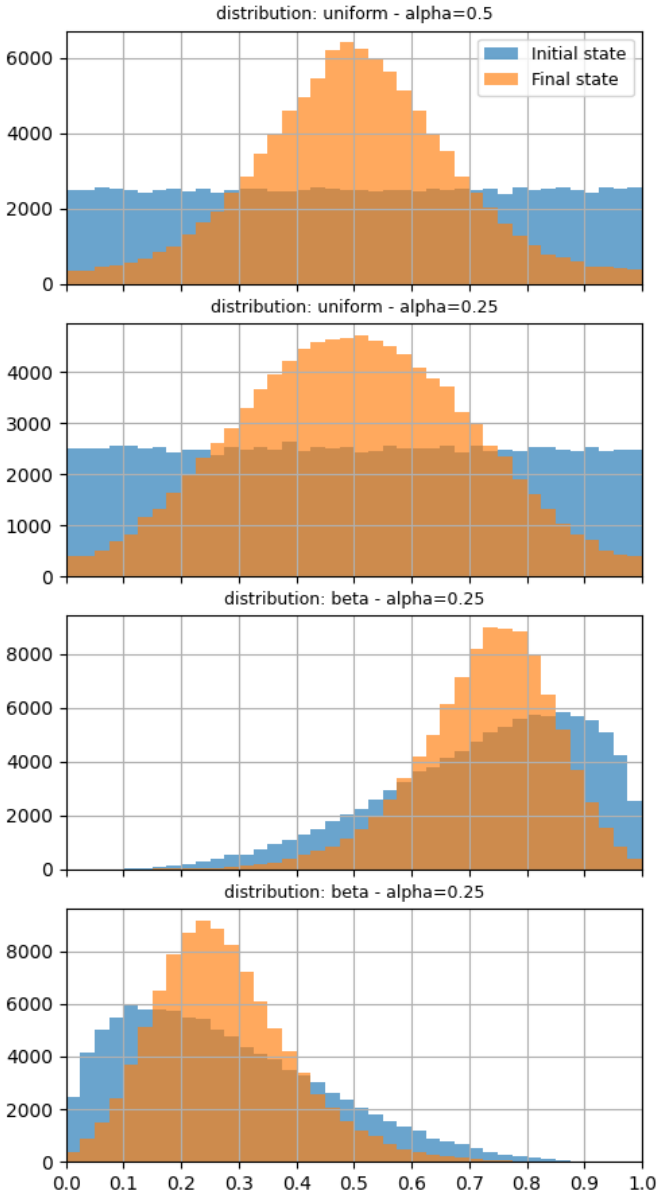

Fig. 3: Nodes state distribution before and after the averaging dynamic process for different α and initial distribution.