Lab L6: Dynamic processes on graphs

Emanuele Pietropaolo
Politecnico di Torino
Student id: s319501
emanuele.pietropaolo@studenti.polito.it

***Abstract*—A dynamical process is a mathematical model that describes the evolution of a system. When run on a graph, it can model complex real-world scenarios such as pandemics or voter preferences. In this context, the nodes represent the state of the system and the edges represent the possible transitions that the system can make. Simulating how such a system evolves over time can provide useful insights into how the modelled system behaves, more than analytical analyses can.**

This paper presents different simulations of the *Voter Model* on random graphs (*Erdős-Rényi model*) and on Z^2 and Z^3 graphs to see how these types of graphs behave in different conditions, how the initial state and the size influence the time to reach consensus and the +1 consensus probability.

I. Problem overview

The Voter Model is a mathematical model that describes a dynamic process. It models the evolution of opinions in a group of people. As we simulate the Voter Model on to a graph, each node of the graph represents an individual and the edges between nodes represent the connection that each person has with the others.

This report analyses some key questions about the dynamic of the process. It analyses whether the dynamics converge to a final state, often referred to as consensus, and what are its characteristics. It also analyses the influence of the initial conditions and the underlying graph structure on the dynamics of the system.

II. Proposed approach

The Voter Model suits the situation where there are two different opinions and individuals can be influenced by what their connections think about them. The opinion that a person has is represented by a state variable that can take two values: $x_v(t) \in \{-1, 1\}$. The change in opinion is modelled as an event where a node wakes up and changes its state variable by copying the state variable of one of its neighbours, following the formula:

$$x_v(t_v^+) = x_w(t_v^-)$$

Simulating the voter model over a graph involves several steps. First, you need to generate the graph you want to use, and then you start modelling the events that cause the system to evolve.

A. Data structure for the graph

A **dictionary** was used to implement the graph. Each node is accessible by an index and contains a nested disctionary where the state variable and the list of neighbours are stored.

Each state variable is initialised according to the probability of being in the $+1$ state, this probability was called the *bias probability*.

B. Algorithm to generate samples of $G(n, p)$ graphs

One of the most common types of graph is the random graph. It is described by a probability distribution or random process that generates it. In fact, its number of edges depends on a stochastic process that connects the nodes based on a probability, the *edge probability*. The model that almost exclusively describes a random graph is the Erdős Rényi (ER) model.

When the ER graph is generated, the dictionary is initialised with the desired number of nodes, without first connecting them. Then, to create the connection according to the edge probability, one of two methods are used based on the difference (in terms of factor of 10) between the number of nodes and the edge probability:

- **Bernoulli experiment for each pair of nodes**: this method involves iterating over all the nodes, and for each one iterates over all the nodes not yet seen. This method then creates an edge between a pair of vertices based on the *bias probability*. This algorithm has a complexity of $O(n^2)$, but tends to be $O(nlog(n))$ because the second for loop is decreasing in time.
- **Take advantage of low edge probability**: if p can be considered small before the number of nodes, the number of expected edges (m) that the graph will have can be calculated by the formula:
$$(n * (n-1) * p)/2$$
where p is the *edge probability*. Then, for each edge, two random nodes that are not already connected are selected and connected. This algorithm has complexity $O(m)$, which tends to $O(n)$ for small values of p.

C. Algorithm to generate samples of Z^k graphs

A graph can also be a *k-dimensional lattice*, where each node has an edge with all the nearest nodes in space. In this paper are explored the Z^2 and Z^3 grids. The nodes of these graphs are arranged in such a way so that they form a 2D or 3D grid respectively, and where a node has all edges perpendicular and parallel to the others.

To generate a finite Z^k graph, is first calculated the dimension of the grid to obtain a lattice with at least the desired number of nodes. Then the dictionary is instantiated with this dimension and each node index is a tuple representing the

coordinates (a pair or a triple respectively). Finally, the edges are created to connect each node to its nearest neighbour.

D. The unfolding of events

As mentioned above, the dynamic evolves through wake-up events. Each event is associated with a particular node, which picks a random neighbour and copies its state variable. Since each wake-up event is independent and occurs with a known frequency, it is assumed that the wake-up process for each node follows a Poisson distribution with $\lambda = 1$. This implies that the wake-up process for the whole graph can be modelled as a process following a Poisson distribution with $\lambda =$ number of nodes in the graph. The time between events is generated from an exponential distribution and each time a random node is chosen to wake up.

Knowing that on finite graph the simulation always reach consensus, we can stop the simulation when it's reached.

III. Results

A. Erdős Rényi graphs

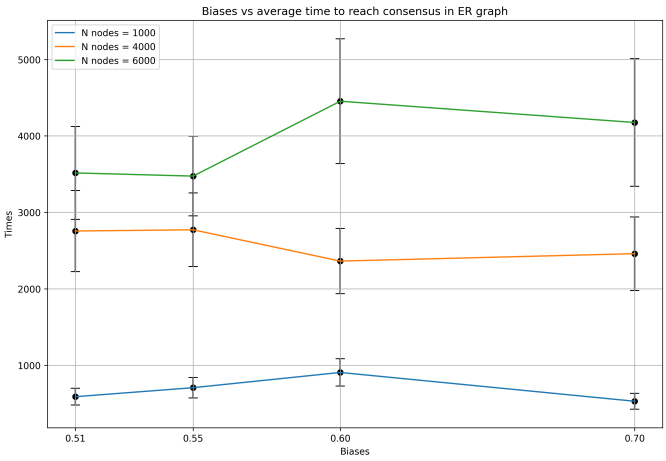

Fig. 1. Average time to reach consensus based on different bias probability

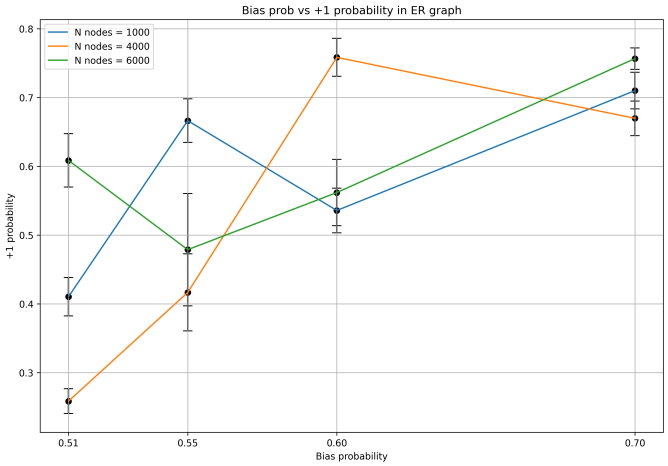

Fig. 2. +1 probability based on different bias probability

B. Z^k graphs

IV. Conclusion

In conclusion, the proposed approach demonstrated that can successfully generate random graphs.