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

II. Proposed approach

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

This report examines two methods of generating samples of random graphs:

- **Generating directly an Erdős–Rényi graph**: this method involves iterating over all the nodes and for each one iterating over all the nodes not already seen. This method then create an edge between a pair of nodes based on the *edge probability*. This algorithm has complexity $O(n^2)$ but it tends to $O(nlog(n))$ because the second for loop is decreasing over time.

Algorithm 1 Erdős–Rényi method

```
Input: n = number of nodes, p = edge probability
for $i$ in range($n$) do
    for $j$ in range($i+1, n$) do
        if random() $< p$ then
            create edge(i,j)
        end if
    end for
end for
```

- **Take advantage in case of small edge probability**: when p can be considered small in front of the number of nodes, it can be calculated the number of expected edges (*m*) that the graph will have through the formula:

$$\frac{n*(n-1)*p}{2}$$

Then create a for loop that for each edge picks two random nodes and connects them. This algorithm has complexity $O(m)$ that for small values of p tends to $O(n)$.

Algorithm 2 Taking advantage of small edge probability

```
Input: n = number of nodes, p = edge probability
$m = n*(n-1)*p/2$
for $i$ in range($m$) do
    while True do
        i,j = generate two nodes index
        if $i == j$ or $jinnode[i][neighbors]$ then
            continue
        else
            $createedge(i,j)$
            break
        end if
    end while
end for
```

B. Data structure

I used a **dictionary** to handle the graph. Each key of the dict is an index of a node and each node is a nested dictionary with a list of neighbors and a state variable.

C. How the FES is handled

The FES handle the wake-up events and at the beginning is filled up with only one event, the wake-up for a random node. Then after each wake-up event, it's filled with a new of such events associated with a new random node. The wake-up process is supposed to follow a Poisson process so the inter-times are generated from a exponential distribution with $\lambda = 1$

I modelled the **Voter Model** so for each event is updated the state variable of the waked up node (v) with the value of a random neighbor (w), following the formula:

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

III. Results

IV. Conclusion

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