Lab6 - Dynamic processes on graphs

Martina Martini s306163
Politecnico di Torino

Assignment The goal of Lab L6 is to study properties of the dynamic processes over graphs.

Consider a voter model over a G(n,p) with n chosen in the range $[10^3, 10^4]$ (in such a way that simulation last a reasonable time i.e. 5/10 min at most) and p_g= 10/n. According to the initial condition, each node has a probability p_1 of being in state +1 with p_1 in $\{0.51, 0.55, 0.6, 0.7\}$. Evaluate the probability of reaching a +1-consensus (if the graph is not connected consider only the giant component). Evaluate, as well, the time needed to reach consensus.

Then consider a voter model over finite portion of Z^2 and Z^3. Fix p_1=0.51 and, for 2/3 values of n in $[10^2, 10^4]$, estimate, as before, the probability of reaching a +1-consensus and the time needed to each consensus.

Deliver the code along with a brief report, in which you present and comment your results. Are the obtained results in line with theoretical predictions?

1 Random elements, inputs and main assumptions

1.1 Introduction

The G(n, p) graph is a typology of graphs that allows us to handle dynamical processes. This type of graph is the result of a random construction process, whose parameters are the number of nodes n and the probability of adding an edge p or not adding it *1 - p*. Moreover, the degree of each node (i.e., the number of edges) follows a binomial distribution, as explained in this report.

The k-dimensional grid is, instead, a type of graph in which each central node has 2^k neighbors: each neighbor's coordinates is computed by adding or subtracting +1 or -1 to one of the coordinates of the central node. The structure of the graph is a regular grid, so that the size is retrieved by taking the integer part of $grid\_size = n^{\frac{1}{k}}$.

With this simulation, I am going to show how to generate a G(n, p) graph, a Z^2 grid and a Z^3 grid intuitively - without any help of predefined libraries - and how the Future Event Set works in a dynamic context. Finally, I will evaluate the time and the probability to reach the +1-consensus state (all the nodes voting +1) and I will add comments on the comparison between the empirical and the analytical degree distribution, the q-q plot and the χ^2 test results.

1.2 Stochastic elements

To simulate such a dynamic process on a graph, some stochastic elements must be considered. Firstly, the inter-arrival times for scheduling the events when the FES is working are exponentially distributed with parameter $\frac{1}{\lambda}$. Secondly, the choice of the state variable between +1 and -1 for each node at the first stage is depending on the probability $p1 \in \{0.51, 0.55, 0.6, 0.7\}$. Thirdly, in the G(n, p) case, the number of edges k is Poisson distributed as follows: $k \sim Poisson(np)$. This result derives from the probability that the degrees (i.e., the number of edges) of the node v equal to k is $\binom{n-1}{k} p^k (1-p)^{n-1-k}$, so we can say that

$$Pr(deg(v) = k) \sim \frac{(np)^k e^{-np}}{k!}.$$

Another stochastic parameter is introduced in the *update* function when the current node selects in a uniform way one of its neighbors and copies its opinion. This leads to the implementation of the Voter Model and the FES scheduling:

activation when this type of event is triggered from the FES, the node taken under consideration is woken up and associated with an opinion (+1 or -1) based on p1; after that, the next *update* event is scheduled,

update instead, when this type of event is triggered, the node chooses randomly one of its neighbors and copies the opinion; then the graph is updated applying these changes.

In addition, in the G(n, p) case, the choice of the k nodes linked with each node (i.e., the neighbors) is randomly executed, while for the other two graphs is constrained by the regular structure. Finally, a seed is introduced for reproducibility purposes.

1.3 Input parameters

For what concerns the input parameters, I assume to set the following fixed variables: the p1 probabilities,

as assigned, equal to $\{0.51, 0.55, 0.6, 0.7\}$, the number of nodes n equal to 1000 for the G(n, p) graph and firstly 100 then 1000 for the regular grids, the number of runs for each simulation equal to 5, the probability of adding an edge p equal to $\frac{10}{n}$, the end time of the simulation to 3000 for G(n, p) and 5000 for the grids, a seed to 42 and some arrival rates as $lambdas = [1, 1.2, 2]$. Finally, for evaluating the confidence intervals and the accuracy of the times taken to reach consensus, I set the confidence level to 0.90, the acceptance to 0.94 and the number of batches to 3.

1.4 Main assumption

Regarding the assumption I made, just one main specification needs to be explained for the simulation purposes. After working with more than 3_000 nodes, I saw that the graph takes a small amount of time to be created, but the update process took always more than one hour per graph. For this reason, I lowered the number of nodes to 1_000 and I saw similar behaviors, so the following considerations and analyses are done under this assumption.

2 Output metrics, data structures and further analyses

2.1 Methodology and data structures

For each arrival rate λ presented above, the same system structure is applied in order to compare it at the end of the simulation. Generally, at the first stage, nodes and graphs are created, then, the FES is initialized and, finally, the simulation starts activating and updating the state variables by means of the FES. Going into the details, the pseudo-codes are presented below.

2.1.1 Nodes generation

This function generates n instances of the class Node. This class is characterized by the index, the vote and the list of neighbors (i.e., the linked nodes). Furthermore, since the voter model is applied, the state variable representing the vote is set to -1 and associated with each node.

2.1.2 G(n, p) graph construction

This function handles each node of the graph generating its neighbors: the graph is structured as a dictionary whose keys are the nodes and whose values consist in a list of linked nodes.

Basically, the loop works for each node as follows:

step 1 choose k number of edges Poisson distributed

step 2 store k as degrees of the node

step 3 select all the other nodes except for the one taken under consideration

step 4 for each iteration until k is reached, choose a random node from the others and add it as a neighbor

step 5 once the first version of the graph is created, check if the node under consideration is present in the list of neighbors of some other nodes: if the node i is present in the list of the node j, then add j to the neighbors of i

step 6 finally, remove the duplicates in the neighbors list

step 7 return the dictionary-structured graph and degrees

2.1.3 k-dimensional grids construction

This function handles the generation of the 2-dimensional grid. It firstly computes the size of the grid that will be extended in two axes, then for each node the function creates the links imagining the nodes occupying a 2D space:

- if the node is not the first in a row, connect it to the neighbor on its left,
- if the node is not the first in a column, connect it to the neighbor on the top,
- if the node is not the last of the row, connect it with the neighbor on the right,
- if the node is not the last of the column connect it with the neighbor on the bottom.

Transporting this discussion in a 3D space, the Z^3 grid is created with the *generate_3d_grid_graph* function: in fact, just two conditions are added to the ones presented above:

- if the node is not in the first layer, connect it with the neighbor in front of it,
- if the node is not in the last layer, connect it with the neighbor behind it.

2.1.4 Simulation

After initializing the parameters required for the simulation, the nodes are created and the graph is constructed. Then, this function initializes the FES as a priority queue (FIFO type) and schedules the first event characterized by the tuple of time = 0 and event = *activation*. Each event *activation* handles a specific node and schedules the next event *update*. Each event of type *update* is associated with a single arrival time,

p1	dimension	n	p	mean avg time (3 batches)	probability	accuracy	chi^2	p-value
0.51	1	1000	0.01	2781.2820145862947	40.0	0.9837157830861091	997.8935424553018	0.5039279196051809
0.55	1	1000	0.01	2729.760909113942	100.0	0.9499813855133092	997.893557717684	0.5039277833006433
0.6	1	1000	0.01	2606.75181648609	80.0	0.9626257570126394	997.3575457294149	0.5087152326438676
0.7	1	1000	0.01	2222.678866576143	100.0	0.9487158571275589	996.5222692204904	0.5161762565361564
0.51	2	1000	0.01	3930.6500949270544	100.0	0.9763076963484467	-	-
0.55	2	1000	0.01	3611.9924773224166	100.0	0.9838726833389471	-	-
0.6	2	1000	0.01	3345.0524229574394	100.0	0.9732451524242827	-	-
0.7	2	1000	0.01	2873.827716342162	100.0	0.977133145924118	-	-
0.51	2	100	0.1	383.7708165524578	100.0	0.9568155589617924	-	-
0.55	2	100	0.1	355.41753339847514	100.0	0.9516402496658722	-	-
0.6	2	100	0.1	330.6634458126847	100.0	0.9537036909493056	-	-
0.7	2	100	0.1	302.80101193329074	100.0	0.957215658502824	-	-
0.51	3	1000	0.01	4010.3148888901252	100.0	0.9769915932183525	-	-
0.55	3	1000	0.01	3535.7117166709695	100.0	0.9748295466265255	-	-
0.6	3	1000	0.01	3295.816234463621	100.0	0.9676853210295341	-	-
0.7	3	1000	0.01	2851.3364831072895	100.0	0.9885527571805894	-	-
0.51	3	100	0.1	406.0394327010988	100.0	0.9567757079126196	-	-
0.55	3	100	0.1	364.0227366396663	100.0	0.9408705708215096	-	-
0.6	3	100	0.1	326.844942699199	100.0	0.9452265746514896	-	-
0.7	3	100	0.1	293.39992927892666	100.0	0.9533088995474306	-	-

Figure 2: Final table containing the average time taken and the probability to reach the +1-consensus

which is defined by the exponential distribution with parameter lambda_rate. The functions *activation* and *update* are explained in paragraph 1.2. When the consensus state is reached (i.e., all the nodes are associated to the vote +1) a flag variable *all_nodes_vote_1* is set to True, the count variable is incremented and the time taken is stored. If, instead, the simulation time expired before reaching the consensus, the number of nodes voting +1 is displayed. At the end of the 5 runs, the system computes the probability of reaching the consensus state and the average time taken.

2.2 Main loop

As outlined above, after setting the input parameters, the main loop is characterized by the following steps:

Algorithm 1 main loop for G(n, p)

```
for each dimension do
    if dimension == 1 then
        for each p1 do
            while accuracy < acceptance do
                for each batch do
                    if acceptance reached then
                        store probability and time taken
                        break
                    end
                end
            end
            QQ-plot, χ² stats and p_value
        end
    end
end
store the results in a dataframe
```

The same algorithm is applied to the k-dimensional grids either for $n = 100$ and $n = 1000$ but no analysis of the degrees distributions is added, because it would be too trivial.

2.3 Output metrics and analyses

As requested, the time taken and the probability to reach the consensus state are stored and displayed for each probability p1 and each dimension (i.e., each type of graphs) in the Figure 2 above. As expected, because of the not-huge number of nodes, all of the graphs reach the consensus state always (except for just some cases) keeping the time below the *end_time*. It can be noticed that by increasing the number of nodes, the time to reach the consensus proportionally grows. Moreover, the variable of time is linearly dependent of the dimension of the graph.

Finally, for what concerns the degrees, I built the q-q plot and I computed the χ^2 test, as shown in teh following Figure 1: the points in the graph are lying as expected on the bisector, which means that the constructed degrees distribution perfectly fits with the theoretical one. Also the p-value confirms what presented above, since its value is always greater than 0.05 so that we cannot reject the null hypothesis.

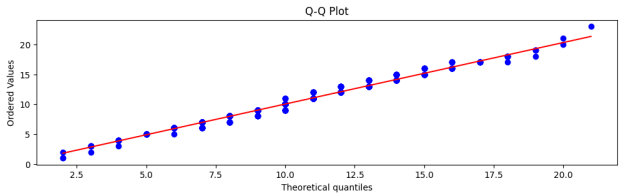

Figure 1: QQ-plot of the degrees of a G(n, p) graph created with $p1 = 0.51$, and $n = 1000$

For clearer evaluations, please refer to the code provided.