Learning Dynamics / Computational Game Theory Assignment 2: Networks and Games

Horatiu Luci
Year 1 - M-SECU-C @ ULB
VUB Student ID: 0582214
ULB Student ID: 000516512

25 November 2020

Part 1 - Complex Networks

1) Generating Erdos-Renye networks (Random networks)

For this exercise, I have generated a network of 10000 nodes. On each generation step, two **different** random nodes were selected and an edge was applied between them if one did not already exist, until the network has 10000 nodes, 20000 edges and an average node degree of 4.

Algorithm 1 Generate 10 000 node random network with a node degree average of 4

Require: $N = 10000$
Require: $e = 20000$
  $G \leftarrow$ Empty graph with 10000 nodes
  $edge\_number \leftarrow 0$
  while $edge\_number < e$ **do**
    $x, y = random.randint(0, n-1)$
    while y = x **do**
      $y = random.randint(0, n-1)$
    end while
    if not $(G.has\_edge(x, y))$ **then**
      $G.add\_edge(x, y)$
      $edge\_number++$
    end if
  end while

Note: For this academical purposes, I used the NetworkX Python package in order to visualize and gather the network information, however, **no NetworkX algorithms were used when generating the network.**

2) Degree distribution for the generated network

The degree distribution for the random network that was generated has been plotted as histogram using Python matplotlib module. Then, the mean and standard deviation were used to compute the normal distribution. [Figures 1 and 2]

Figure 1: Degree distribution plot of a random network

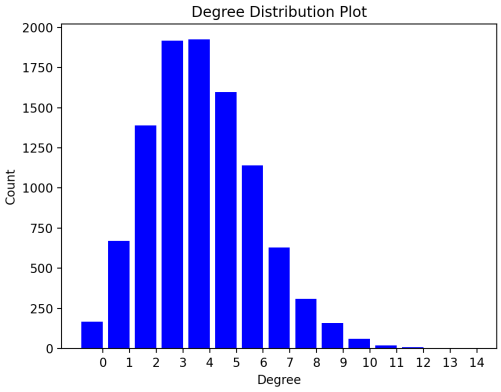
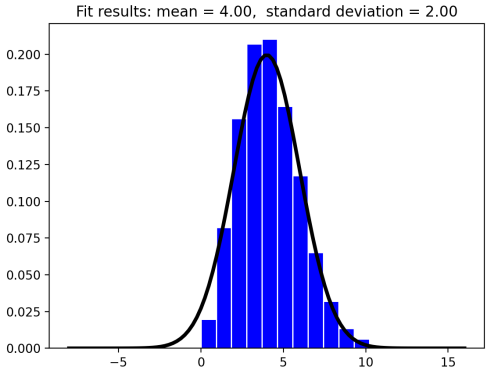


Figure 2: Degree distribution plot of another random network fitted with a Gaussian



3) Generate a Barabasi-Albert network (Scale Free network)

For this part, I have generated a network of 10000 nodes. Starting from a configuration of 4 fully connected nodes. At each step, a node with preferential attachment to nodes with many links is added.

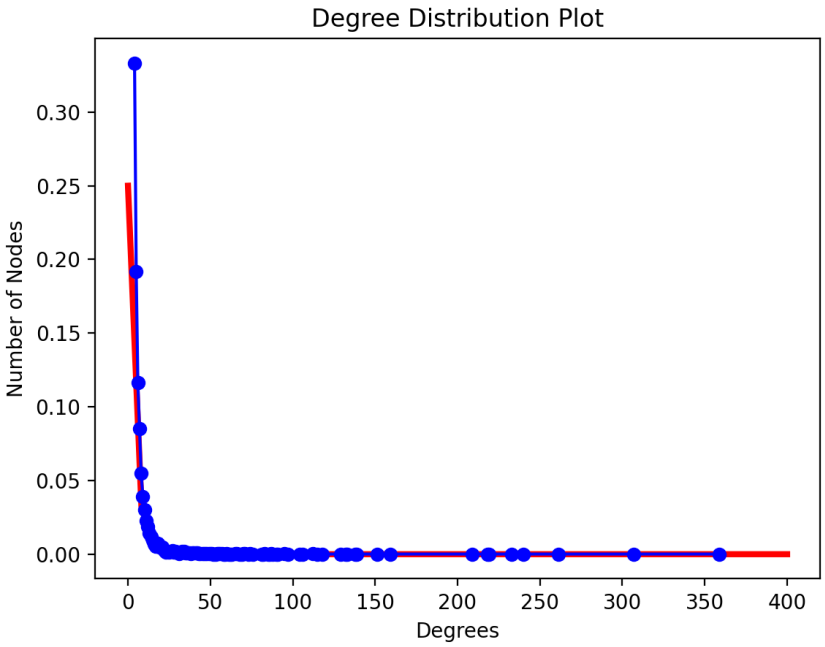
Algorithm 2 Generate 10 000 node random network with a node degree average of 4

Require: $number\_of\_edges \leftarrow 0$
  for all $nodes\_i \leftarrow 1 \dots m_0$ **do**
    for all $nodes\_j \leftarrow i+1 \dots m_0$ **do**
      $graph.add\_node(i, j)$
      $number\_of\_edges++$
    end for
  end for
  for all $nodes\_i \leftarrow m_0 + 1 \dots N$ **do**
    $current\_degree = 0$
    while $current\_degre < m$ **do**
      node_j = randomly uniformly picked node except node_i and those adjacent to node_i
      b = (number of nodes adjacent to node j) / number_of_edges
      Chance = a uniform random number between 0 and 1
      if $b > Chance$ **then**
        $graph.add\_edge(i, j)$
        $number\_of\_edges++$
      end if
    end while
  end for

4) Degree distribution of the Barabasi-Albert network on a linear scale

In this exercise, I have plotted the degree distribution of the generated network, after normalising the data by dividing each degree count with N (the total amount of nodes). The exponential distribution was generated using the stats.expon methods from scipy python module.

Figure 3: Degree distribution plot of a Barabasi-Albert and an exponential distribution

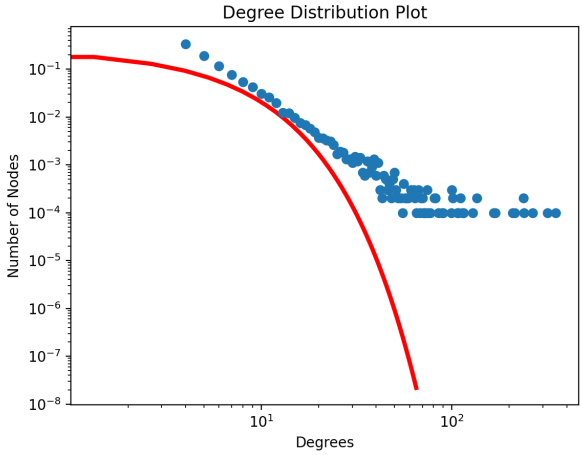


Even in this plot, it can be slightly seen that the degree distribution plot drops faster than the exponential one, this indicates that we are encountering a power-law.

5) Degree distribution of the Barabasi-Albert network on a log-log scale

The degree distribution of the generated network was again plotted but this time on the log-log scale. On the log-log scale, a power-law looks like a line (apart from the variations in the tail due to the randomness i.e.: events with big impact that happen with small probability). Contrary to this, the exponential distribution is curved. As predicted in the previous exercise, this fit is not good for the distribution of this network.

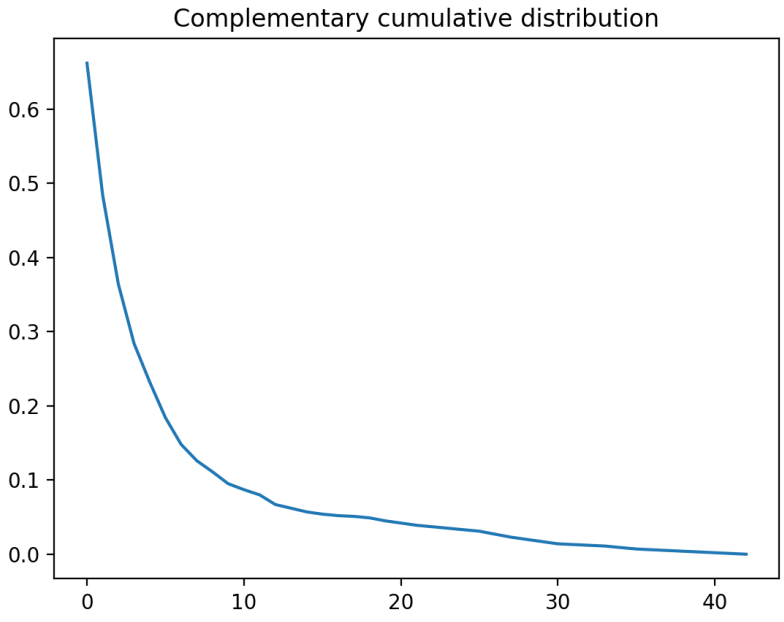
Figure 4: Degree distribution plot of a Barabasi-Albert and an exponential distribution on log-log scale



The least square fitting procedure is the most simple and commonly used form of linear regression, thus providing a solution to the common issue of figuring out which the best fitting straight line through a points. A way to probe for power-law behaviour with this procedure is to plot a histogram representing the frequency distribution of the quantity in some data along with a least-square linear regression on a log-log scale.

Therefore the misconception is that the scaling parameter α of a power-law is given by the absolute slope of the straight line. This method has the fault of generating significant systematic errors under relatively common conditions such as, during validation, when some non power law distributions appear to follow a power law for some small samples.[1]

6) Complementary Cumulative Distribution Plot



Part 2 - Game Theory on Networks

Simulating Prisoner's Dilemma on the generated networks

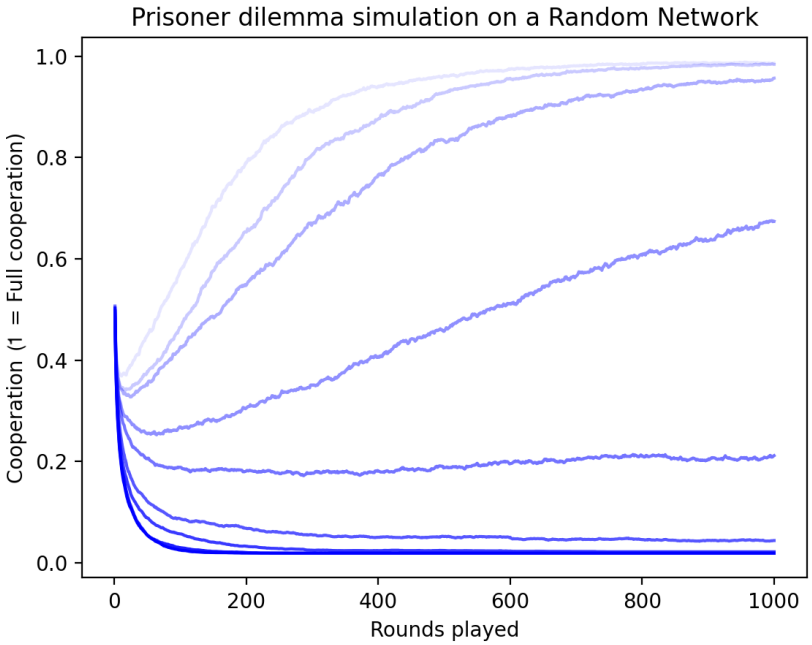
For the generated networks, the simulation of agents playing prisoner's dilemma has been run for 10000 rounds. Starting from a probability of 50% each individual will choose on each round to either play Cooperate (1) or Defect (0) and will update their beliefs according to the given probability formula.

This formula makes sense because when an agent will choose a new strategy, they will copy the agents that are doing better than them, thus the bigger the difference, the bigger the probability of behaviour replication. Furthermore, in order to keep the probability normalised, the payoff difference was divided by the biggest degree between nodes times the difference between the biggest and smallest possible payoff.

Plotting the cooperation over time

Random Network

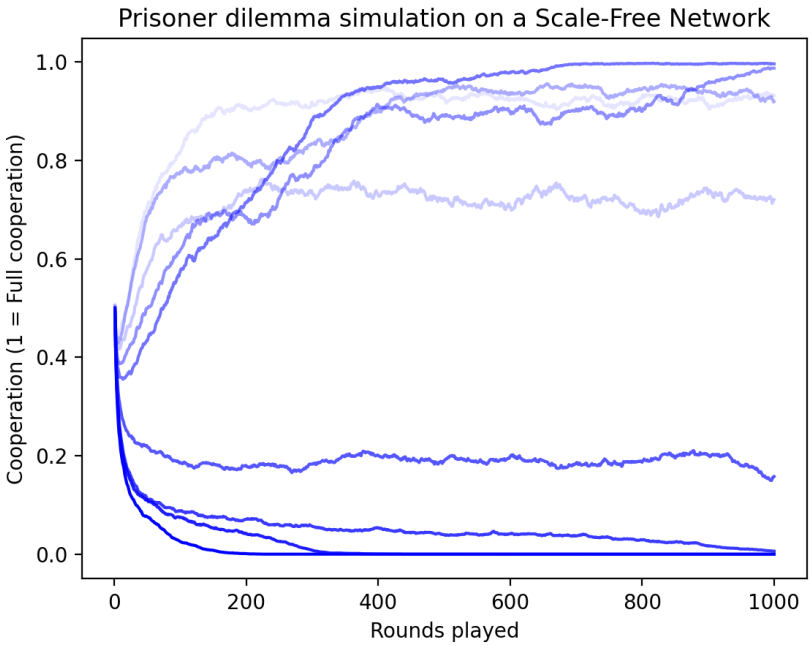
Figure 5: cooperation in prisoner dilemma simulation on a random network level over time as T increases



For the random network, for low values of T, the agents will reach a stationary state in just under 400 rounds playing Cooperate strategy (even reaching full Cooperation). In contrast, for the high values of T, the agents reached a stationary state in just under 200 rounds (for the highest values even under 100 rounds, even reaching full Defect) playing Defect strategy.

Scale-free Network

Figure 6: cooperation in prisoner dilemma simulation on a scale-free network level over time as T increases



In the scale-free network, for the high values of T, the agents reached a stationary state playing Defect in just under 200 rounds (for the highest values even under 100 rounds, and with full Defect). In contrast, for the smaller values of T, the agents will reach a stationary state playing Cooperate, apart from the smallest value of T which is still showing significant changes even after 1000 rounds.

Final comments

After plotting the cooperation levels over time on both networks, we can safely say that the cooperation levels are higher on average on a scale-free network than on a random network, since five populations in the scale-free network reached high cooperation levels, compared to only three from the random network.

Even though the average stationary cooperation level over 20 simulations (for each T and both networks) was not completed due to processing power limitations, we can still safely assume that, for the random network that we had, further simulation would have lead to 4 populations (those with the lower T) full cooperation and for the scale-free network, 5 populations will achieve full cooperation.

References

[1] https://arxiv.org/pdf/0706.1062.pdf