UNIVERSITY OF ROVIRA I VIRGILI
Department of Computer Engineering and Mathematics
March 30, 2021

CN-MAI Assignment 2

Complex Networks
Spring 2021, 5 Credits

Models Of Complex Networks

Name Bartosz Paulewicz
Betty Törnkvist

E-mail `bartosz.paulewicz@student.put.poznan.pl`
`betty.tornkvist@gmail.com`

Teacher
Alejandro Arenas Moreno

Contents

1 Erdös-Rényi (ER) Networks

The ER network comes in two variants - one with defined number of edges m to generate (GNM), and the other with defined probability p of generating each edge in the network (GNP). The two implementations follow a Poisson distribution and have a fixed number of nodes. Both were implemented and further explained below.

1.1 GNM

GNM variant takes as an inputs the number of nodes and number of edges to be generated, and randomly generates edges preventing creation of self-loops and multi-edges. For a simplified overview of the algorithm, see Algorithm 1.

Algorithm 1 GNM

```
    Input: n - number of nodes, m - number of edges
    Output: g - graph
1: Create graph $g$ with $n$ nodes
2: for i in range 1..$m$ do
3:     source = random g node
4:     target = random g node (not a neighbour to source and not source itself)
5:     Add edge between source and target to graph $g$
6: Return graph $g$
```

A GNM network of size n=50 and m=100 was plotted, see Figure 1.

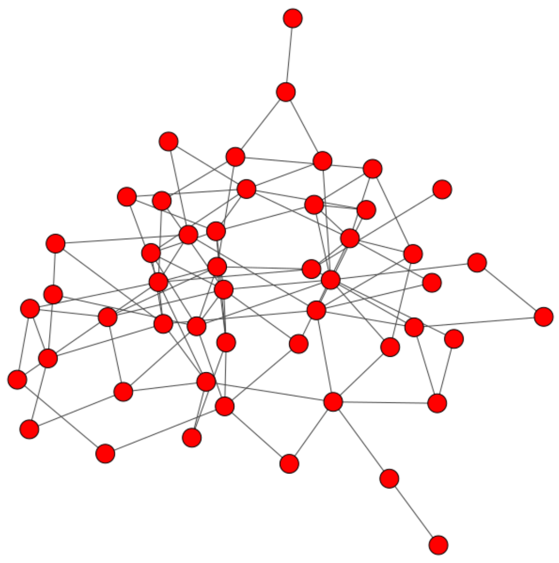

Figure 1: GNM network with 50 nodes and 100 edges.

The degree for a GNM with 10000 nodes and 20000 edges is plotted and compared to a Poisson distribution with the same size of degree, see Figure 2. It can be observed that the degree distribution of the GNM follows the Poisson distribution.

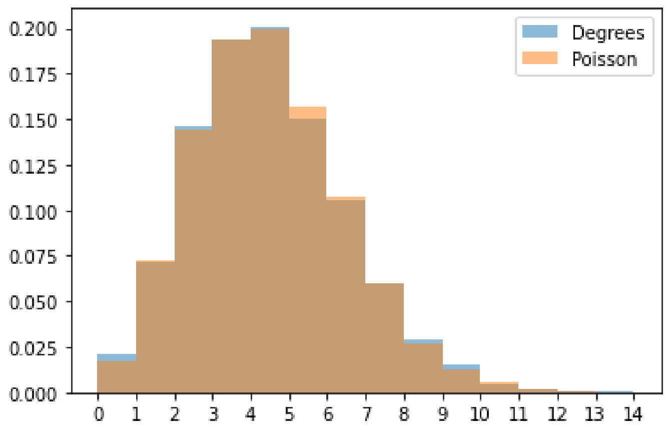

Figure 2: Poisson- and GNM distribution.

1.2 GNP

Instead of taking the number of edges, the GNP variant takes the number of nodes and the probability of creating an edge. The simplified algorithm can be seen in Algorithm 2

Algorithm 2 GNP

```
    Input: n - number of nodes, p - probability of creating an edge
    Output: g - graph
1: possible edges = all possible edges between vertices $n$
2: r = random number generator in range [0, 1)
3: Generate graph $g$ with $n$ nodes
4: for edge in possible edges do
5:     if $r()$ is below probability $p$ then
6:         Add edge to graph $g$
7: Return graph $g$
```

A GNP network of size n=50 and p=0.1 was plotted, see Figure 3.

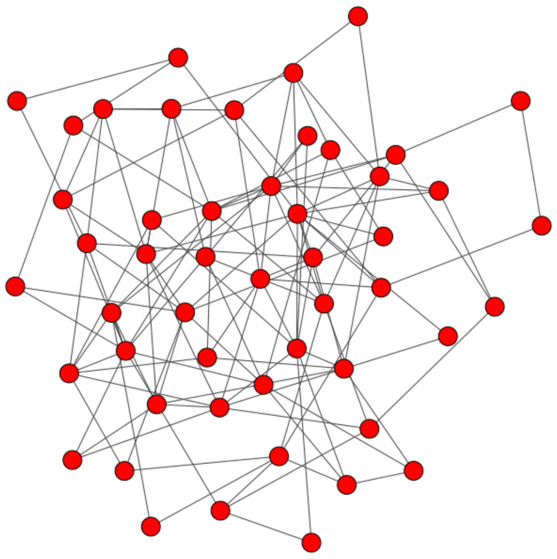

Figure 3: GNP network with 50 nodes and probability 0.1.

The degree for a GNP with 10000 nodes and probability 0.001 is plotted and compared to a Poisson distribution with the same size of degree, see Figure 2. Since the degree is drawn from a binomial distribution, the degree distribution almost perfectly overlaps with the Poisson distribution. The reason being that both the binomial and Poisson distribution measure the number of random events, in this case the number of edges created.

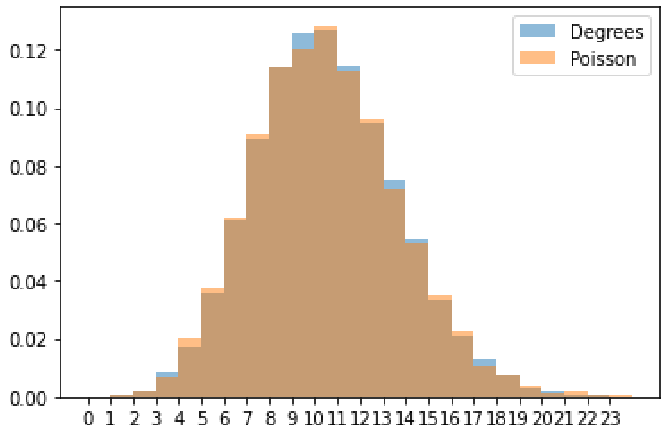

Figure 4: Poisson- and GNP degree distribution.

2 Watts-Strogatz (WS) Small-World Model

The small-world model is built to output a model with a small average shortest path, and a large clustering coefficient. The model is initialised by making edges between each nodes four closest neighbours, resulting in a network with a degree of four for every node. Depending on the given probability, random nodes are then re-wired with new edges. Like the GNM and GNP, the WS network has a fixed number of nodes and is therefore limited in the way that it can not be used to model network growth. For a simplified overview of the algorithm, see Algorithm 6.

Algorithm 3 WS

```
    Input: n - number of nodes, p - probability of re-wiring an edge
    Output: g - graph
 1: r = random number generator in range [0, 1)
 2: Generate graph $g$ with $n$ nodes
 3: for node in g nodes do
 4:     Connect node to its four closest neighbours
 5: for edge in g edges do
 6:     if $r()$ is below probability $p$ then
 7:         source = edge source node
 8:         target = random g node (not a neighbour to source and not source itself)
 9:         Add edge between source and target to graph $g$
10:         Remove edge from graph g
11: Return graph $g$
```

A WS network of size n=50 and p=0.1 was plotted, see Figure 5. It can be observed that many nodes still have their initial degree of four, while others have lost or added a few nodes. However, none of the nodes has decreased to a degree of 1 and the highest degree is 7.

The degree distribution for a WS with 10000 nodes and probability 0.75 is plotted, see Figure 6. Here we can observe that the amount of nodes changes drastically between degree 1 and 2 and then gradually starts to decrease between degree 4 to 10.

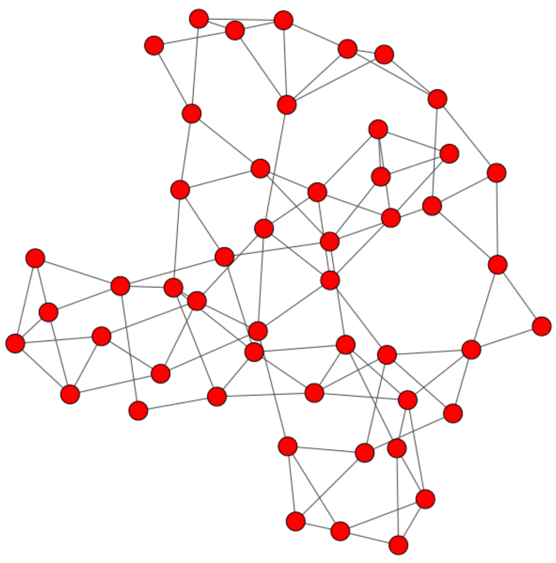

Figure 5: WS network with 50 nodes and probability 0.1.

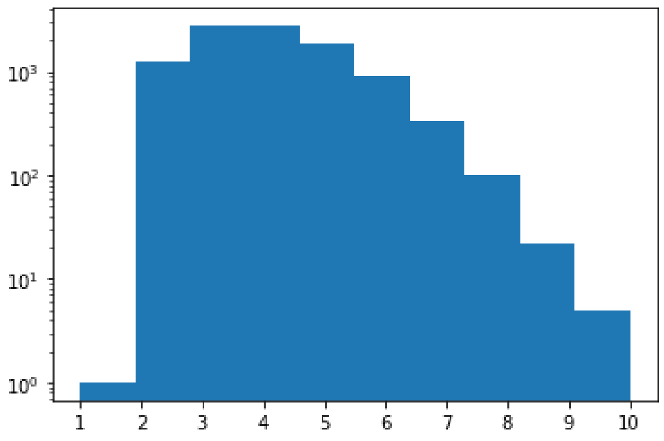

Figure 6: WS degree distribution.

3 Barabási-Albert (BA) Preferential Attachment Model

The BA network is built on the concept that nodes will have a higher probability to connect to popular nodes of a high degree. The popular nodes will be preferred to less popular ones. This is called preferential attachment and is implemented by letting a new arbitrarily chosen node connect to any node, but a probabilistic mechanism makes it more probable to connect to a node with a higher degree. It is a scale-free network, which means it contains its original structure but grows with one node per time-step. It therefore also follows a power-law, meaning that a larger amount of nodes have more edges, and often a few of them are highly connected. For a simplified overview of the algorithm, see Algorithm 4.

Algorithm 4 BA

```
    Input:
    n - number of nodes,
    m - number of edges per new node,
    m0 - number of fully connected nodes
    Output: g - graph
1: Generate complete graph g with m0 nodes
2: for i in range 1..n-m0 do
3:     Add new source node to graph g
4:     for j in range 1..m do
5:         target = random g node (not a neighbour to source and not source itself) with a probability of being chosen proportional to its degree
6:         Add edge between source and target to graph g
7: Return graph g
```

A BA network of size n=100, m=1 and m0=1 was plotted, see Figure 7. It is an edge case, which clearly shows the implementation works by creating a network that has one edge per new node (initial nodes can have more), and one central node that is fully connected.

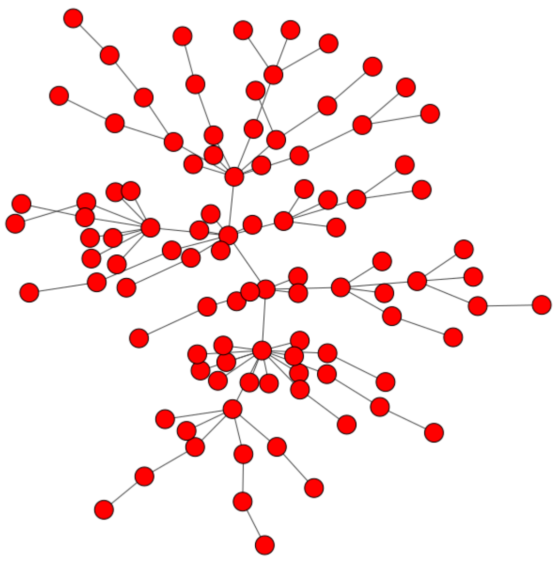

Figure 7: BA network with 50 nodes and probability 0.1.

The distribution for a BA network with n=1000, m=5 and m0=5, plotted in logarithmic scale follows a fairly straight decaying line, which suggests that it follows the power-law, see Figure 8.When examining the exponent of the powerlaw from the probability distribution of the same network, we get a value of α = 2.9 which results in a skewed and long-tailed plot, see Figure 9. This corresponds quite well to the typical degree distribution exponent of a BA network (γ=3)[1].

Further, the complementary cumulative distribution function (CCDF) and the probability density function (PDF) is plotted for the same network, see Figure 10. They are both probability functions, where they in this case measure the probability of how many edges a node has. More precisely, the CCDF measures the probability that a nodes degree has a greater value than x, while the PDF measures the probability that the degree is exactly x.[2] It can be observed that the data follows the theoretical powerlaw CCDF quite well, while the PDF is slightly worse towards the tail.

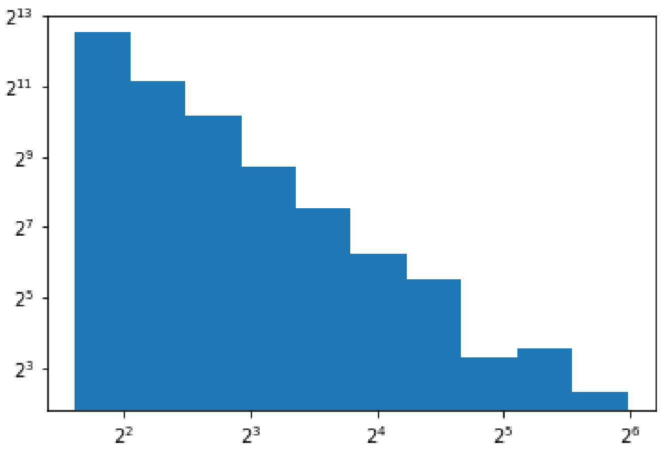

Figure 8: BA degree distribution in log-log scale, which reveals a linear relationship.

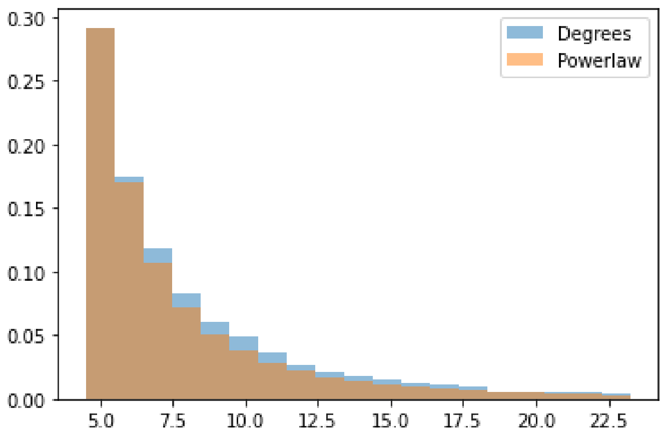

Figure 9: BA and power-law distribution, with alpha on the y-axis and degree on the x-axis.

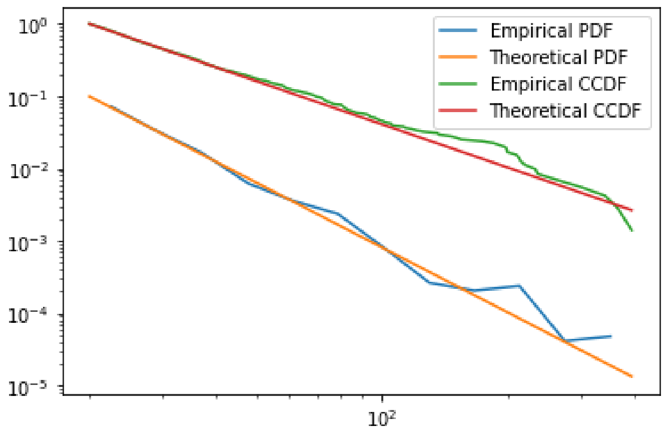

Figure 10: Theoretical/Empirical CCDF and PDF for BA.

4 Configuration Model (CM)

The CM model takes a distribution function as input and creates a network with the corresponding degree distribution. Our implementation uses a heap of nodes and their desired degree, and then assigns edges until the degree for each node is fulfilled. For a simplified overview of the algorithm, see Algorithm 5.

Algorithm 5 CM

```
    Input: n - number of nodes, distribution - distribution function
    Output: g - graph
 1: Generate graph $g$ with $n$ nodes
 2: repeat
 3:     Generate desired degrees for $n$ nodes using distribution function
 4: until sum of degrees is even
 5: Create a mapping stubs from each node to its desired degree
 6: Create a heap from stubs sorted by descending desired degree
 7: while stubs is not empty do
 8:     repeat
 9:         From heap pop source node having the highest remaining degree to satisfy
10:     until source node is in stubs
11:     target = random g node (not a neighbour to source and not source itself)
12:     Add edge between source and target to graph g
13:     for node in (source, target) do
14:         In stubs decrease remaining degree to satisfy given for node
15:         if degree to satisfy is 0 then
16:             Remove node from stubs
17:     if source still has degree to satisfy then
18:         Push source back to heap with the new degree to satisfy
19: Return graph $g$
```

Two CM networks of size n=100 where generated, one with Poisson distribution and the other with gamma distribution, see Figure 11 and Figure 12.

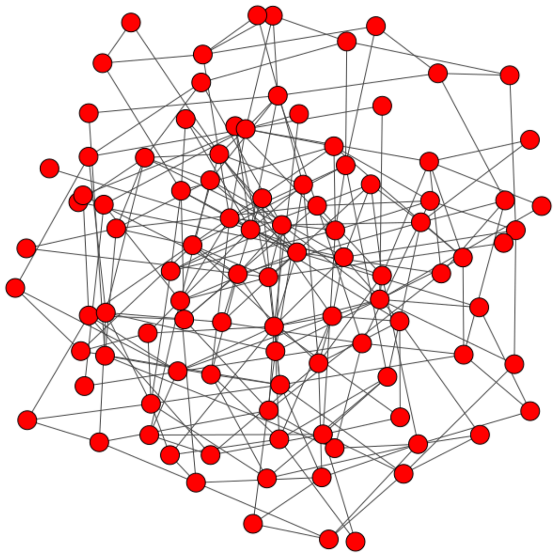

Figure 11: CM network with 100 nodes and Poisson distribution.

When plotting the degree distribution in comparison to their corresponding distribution, both can

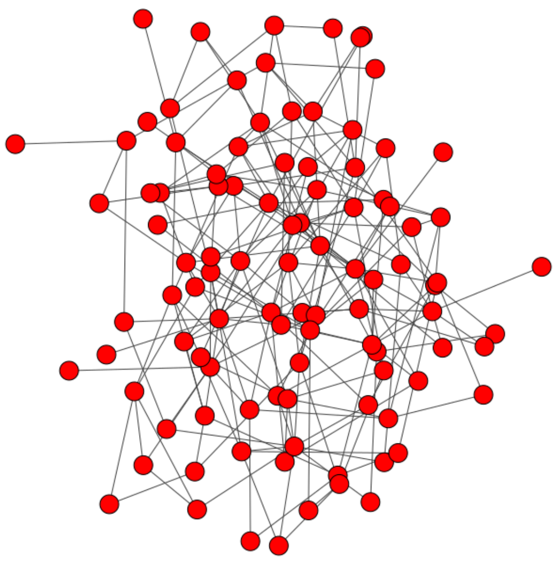

Figure 12: CM network with 100 nodes and gamma distribution.

be observed to follow the same distribution. See Figure 13 and Figure 14.

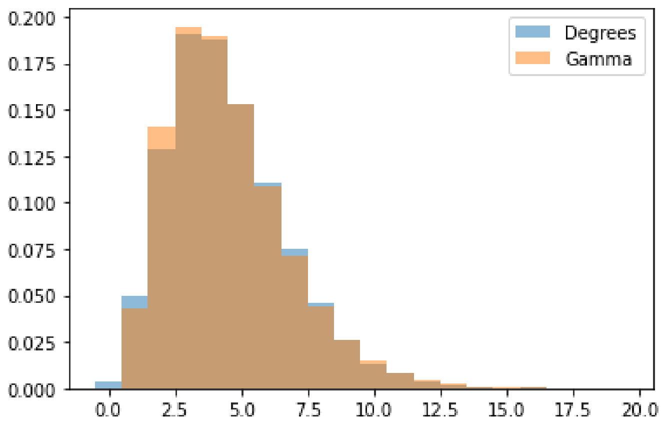

Figure 13: CM gamma distribution.

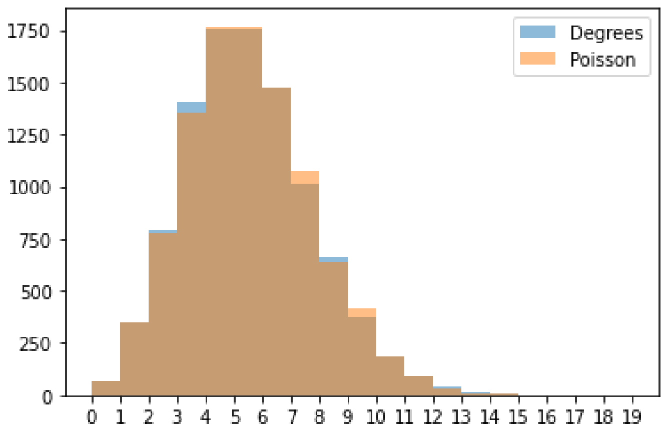

Figure 14: CM Poisson distribution.

References

[1] R. Rak and E. Rak, "The fractional preferential attachment scale-free network model," *Entropy*, vol. 22, no. 5, p. 509, 2020.

[2] boost. Non-member properties. [Online]. Available: https://www.boost.org/doc/libs/1_41_0/libs/math/doc/sf_and_dist/html/math_toolkit/dist/dist_ref/nmp.html