Complex Networks Report: Project Part I

Group 12
75657 - Paulo Gouveia, 75694 - Daniel Figueira, 79758 - Luisa Santo
Monday, 15:30h, Lab11

October 30, 2017

Abstract

This project provided the opportunity to learn how complex networks work by getting acquainted with NetworkX, a Python library for creation, manipulation, and study of complex networks. It describes the most important models for random, small world, and scale-free networks, and discusses their properties.

With the Python programming language, we analyzed and compared different graph generation models, along with algorithms we implemented ourselves.

Measures such as the clustering coefficient and the degree distribution were also defined.

1 Introduction

Networks have been gaining more attention by the research communities. By analyzing complex networks in various ways, many important properties of the subjects can be inferred: for military reasons – networks need to be robust against enemy attacks and component failures and designed so that information can be spread as fast as possible – or for other reasons – the Internet and World Wide Web (WWW), social networks, transportation networks, food webs and biological networks.

It has been established in several papers that the random graph models, introduced by Erdös and Rényi, do not accurately capture many of the key properties of networks that occur in the real world. This reason led us to analyze other network models including the small-world model of Watts and Strogatz (WS) and the scale-free models of Barabási and Albert (BA).

1.1 Random Networks

As mentioned before, in the absence of real data people tried to recreate the phenomena observed in real networks using random networks. As a result, a large variety of graph models has been established.

Around the 1950s, the only graphs ever known were the "regular" graph. The focus was afterward extended with the Hungarians Paul Erdös and Alfréd Réyni on large–scale networks with no apparent design principles, the so–called "random graphs". Since their famous paper from 1959, random graph theory has become one of the main areas of interest in modern discrete mathematics, producing many and fascinating results. Recently, other models have been created, namely the one by Watts and Strogatz, and the one by Barabási and Albert.

But what is a random graph? A random graph consists of N nodes where each node pair is connected with probability p. To construct a random network we follow these steps:

1. Start with N isolated nodes.
2. Select a node pair and generate a random number between 0 and 1. If the number exceeds p, connect the selected node pair with a link, otherwise leave them disconnected.
3. Repeat step (2) for each of the $N(N-1)/2$ node pairs.

The network obtained after this procedure is called a random graph.

1.1.1 The Watts–Strogatz and "small–world" Model

The idea is to start with order and then randomize. Starting off with order allows for the relatively high clustering, yet randomizing by rewiring some edges creates "long distance shortcuts" which are responsible for the relatively low average path length.

Watts and Strogatz chose a one-dimensional lattice with a periodic bounding condition as starting point: take n nodes, lay them out to form a "ring" and connect each node with its k_0 neighbors. In order for that lattice to be symmetrical, k_0 must be even (for example two nodes to the "left" and two nodes

to the "right" would mean $k_0 = 4$). Once this regular lattice is generated, edges are randomly rewired – random both in terms of whether or not to be rewired in the first place, and, if so, where to connect the free end to.

1.1.2 The Barabási–Albert and "scale–free" Model

This model allowed for the first time to describe the inherent ordering principle observed in real-world networks. Its simple but elegant generating algorithm is the following:

- *Growth*: Start with a very small but fully connected graph on n_0 nodes ($1 \leq n_0 \ll n$, n being the desired final number of nodes). At each iteration step t, add one node with m edges that link the new node to m nodes that already are in the system.
- *Preferential attachment*: The "targets" for the m new edges to be placed are not chosen with equal probability, but with probability proportional to their respective degrees.

So in order to get a network with n nodes, this algorithm needs to be run for $t = n - n_0$ steps, resulting in a total of $(n - n_0)m$ edges.

1.1.3 Node Duplication

In this model, the network is represented by a non-directed, unweighted graph. Let t_0 be a constant and G_{t0} be a graph on t_0 vertices. For t $> t_0$, G_t is constructed by partial duplication from G_{t1} as follows: A random vertex, u, of G_{t1}, the sampling vertex, is selected. Then a new vertex v is added to G_{t1} in such a way that for each neighbour w of u, with probability p, a new edge $v - w$ is added and with a probability q of being connected to the original node. The complete, or full duplication of the node results from setting p and q to 1.

1.2 Metrics

1.2.1 Degree Distributions

The degree distribution lets us identify the type of network (scale-free, small-world, etc) as it gives us an idea of its structure.

However, this metric does not take into account how the nodes are connected to each other and therefore can never give a comprehensive description of the structure. In this case, we will also measure the clustering coefficient and average shortest path length.

1.2.2 Clustering Coefficient

Many types of networks have some inherent tendency to form clusters. For instance, people tend to have friends who are also friends with each other.

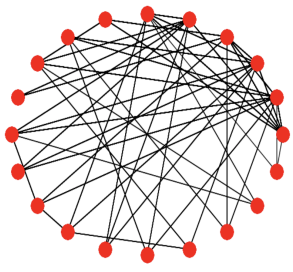
A high clustering coefficient means sets of nodes among which many edges exist, compared to a set made from randomly chosen nodes that would have a much smaller number of edges between them.

1.2.3 Average Shortest Path Length

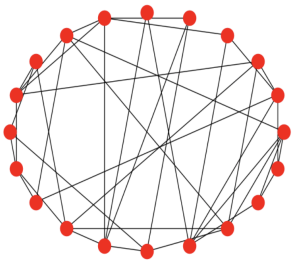
Lastly, the average shortest path length is a measure of the efficiency at which information is transported through the network.

2 Code Architecture

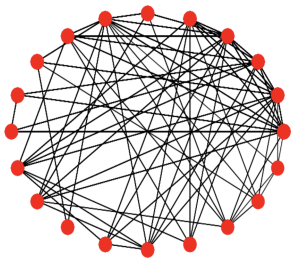
Since the goal of our project is not to achieve the best performance possible, we chose to write our program in Python due to its clean code and existing libraries, specifically NetworkX, which we use to generate and get metrics on graphs.



(a) Barabasi-Albert with 20 nodes and 4 edges



(b) Watts-Strogatz 20 nodes, 3 edges and 0.5 probability of swap



(c) Node Duplication with 20 final nodes, 4 initial nodes, 0.5 probability of connecting to neighbors and 1 probability of connecting to duplicated node

Figure 1: Visual representation of each network model

Our program works in a simple guided shell like the format in which is possible to select the generation model as well as its parameters. Our code is able to generate graphs based on three different models: Barabasi-Albert, Watts-Strogatz and Node Duplication, as well as extract metrics from such graphs so that we can then compare them.

3 Results

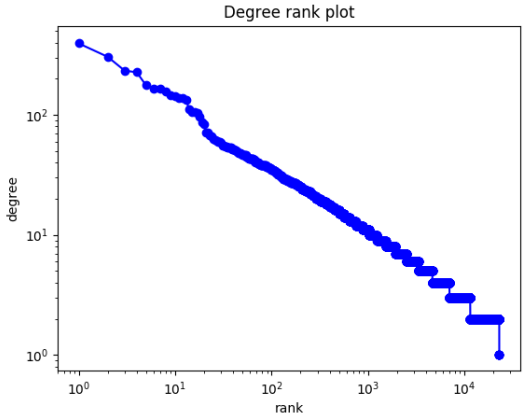
Creating one small graph per model (with 20 nodes) we can have a visual representation of the network and verify that the generation is correct, as shown in Figure 1

3.1 Comparison of the models

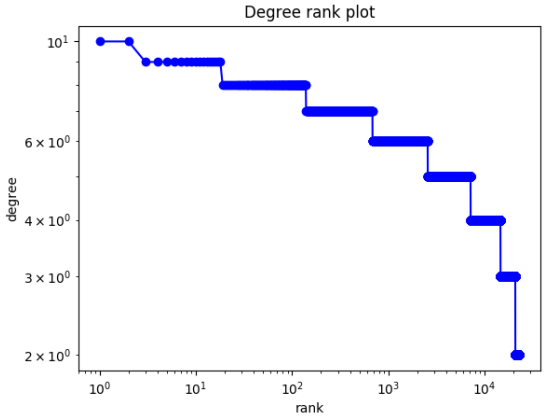
As expected from Figure 2 and Table 1, the Barabasi-Albert model creates a network which degree distribution follows a power-law. This can be explained by the use of preferential attachment where nodes give preference to connecting to nodes with a higher degree, making the degree of the target node even higher and thus creating hubs.

However, because of the same preferential attachment, the networks generated by this model have an unrealistically low global clustering coefficient since nodes prioritize connecting to hubs over connecting among themselves.

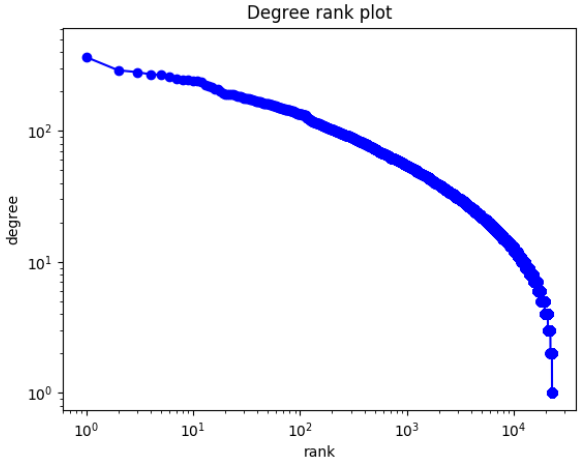
In contrast, the Watts-Strogatz model leads to networks with a higher, more realistic, global clustering coefficient as well as a low average shortest path length. However, because of its random nature, this generation model leads to an unrealistic degree distribution.



(a) Barabasi-Albert with 23000 nodes and 4 edges (BA 23000 4)



(b) Watts-Strogatz 23000 nodes, 4 edges and 0.5 probability of swap (WS 23000 4 0.5)



(c) Node Duplication with 23000 final nodes, 2 initial nodes, 0.5 probability of connecting to neighbors and 1 probability of connecting to duplicated node (ND 23000 2 0.5 1)

Figure 2: Degree distribution of nodes in the network

	BA 23000 4	WS 23000 4 0.5	ND 23000 2 0.5 1
Nodes	23000	23000	23000
Edges	91896	46000	193460
Average degree	7.991	4	16.8226
Global clustering coefficient	0.00336	0.06514	0.47540
Average shortest path length	4.14284	8.33522	6.57400
Run time	51m39s	33m11s	90m24s

Table 1: Gathered metrics on the generated networks

It is worth mentioning that because this model relies on a fixed number of nodes (that exist from the beginning) and edges (swapped with each other), metrics such as the average degree of the network never change.

The Node Duplication model is very dependant on the chosen probabilities. With a probability of 1 to connect to the duplicated node (to make sure the graph is connected) any meaningful probability of connecting with its neighbors will generate a high clustering coefficient.

3.2 Computational issues

During the development of our project, we came across two significant computational issues.

The first one is that certain metrics can only be computed on certain types of graphs, which we resolved this by generating a new graph every time we got one that did not meet the desired criteria.

The second is the computation of the average shortest path length in particular. The algorithm from the chosen library uses a breadth-first-search to find the shortest path from each node to every other node leading to a time complexity of $O(N(N+E))$.

4 Conclusion

Topics on networks grow in complexity and relevance every day and thus, this project was a great learning experience. We learned about complex network algorithms and in particular, we learned about NetworkX, its advantages and limitations. It was an opportunity to apply concepts studied in class as well by implementing certain graph generation algorithms.

Regarding the computational issues, future work can be done in order to improve this first part of the project by adapting the computation of the average shortest path length to a multithreaded algorithm and changing the graph generation algorithms so that it only generates valid graphs.