The Robustness and Predictability of Networks

Efe Ali Görgüner

September 5, 2022
ORIS Project - supervised by Prof. Ginestra Bianconi

Abstract

Networks can be used to model many different systems. Through simulations, I investigated the robustness and predictability of synthetically generated and real world networks. I verified already known theoretical results and analysed real world data, as well as exploring some boolean network models, following my own research questions.

1 Introduction

Complex systems are all around us - ranging from flocks of birds in the sky forming complicated patterns to a traffic jam suddenly appearing out of nowhere and snowflakes made of countless identical water molecules making beautiful shapes. The one thing they share in common is that the whole is not the sum of the parts. The many interactions between the individual parts results in emergent behaviour. And one way to study these systems is through networks - a set of nodes connected by links.

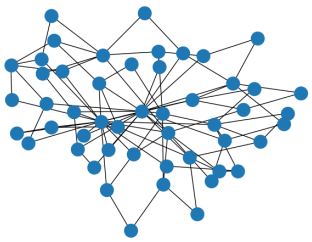


Figure 1: An example of a network - the nodes can really be anything: genes, power stations, animals, airports, etc.

The presence of many interactions, organised in an irregular complex way, makes the dynamics of networks hard to predict. However, by studying models of complex networks, we can try to gain new insights in order to better understand and predict real world networks.

2 Percolation Theory and Robustness

One way to study the robustness of networks is through studying what effect the removal of nodes or links has on the network structure. We can use percolation theory to describe this process. The major property that percolation models is the network connectivity. As more and more nodes or links are removed, the network might suddenly fragment into disconnected components. So, we can measure the connectivity of a network through the size of the giant component - the largest connected component in the network (including a finite fraction of the nodes of the network) where any nodes within this component can be reached from any other node in the component[3].

To be able to numerically study the effect of percolation on different networks, we need to have a way to generate networks. One way is using the Erdös-Rényi random network model [4]. In this model, two parameters are considered - the size of the network and the probability of including a link. Considering a complete network of size N (where every node is connected to every other node), each link in this network remains with the probability given in the parameter.

Using the NetworkX library in Python [5], I generated a random Erdős-Rényi network. Then by randomly removing a certain fraction of the nodes or links and measuring the size of the giant component, the simulation generated the graph in Figure 2.

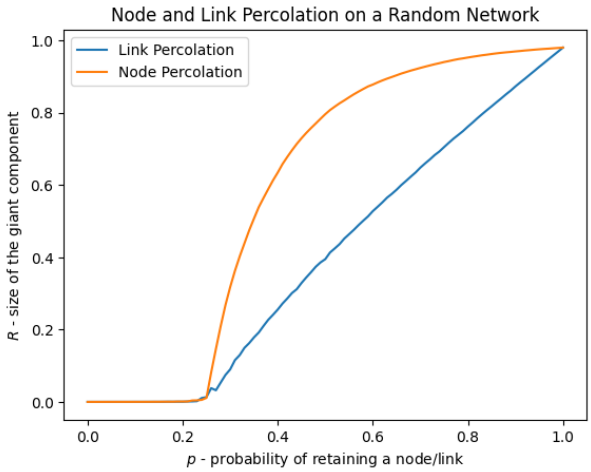


Figure 2: Percolation on a random network with $N = 10,000$ and $\langle k \rangle = 4$.

From the graph we can see that the size of the giant component remains near zero until the probability of retaining a link goes up to 0.25. At this critical probability, there is a phase transition. For both link percolation and node percolation, the giant component only appears if more than 25% of the nodes or links remain. However, for any probability higher than the critical probability, the size of the giant component is larger for the network with link percolation. This is because when a node is removed in this network, on average 4 links are removed which results in more damage and a smaller giant component.

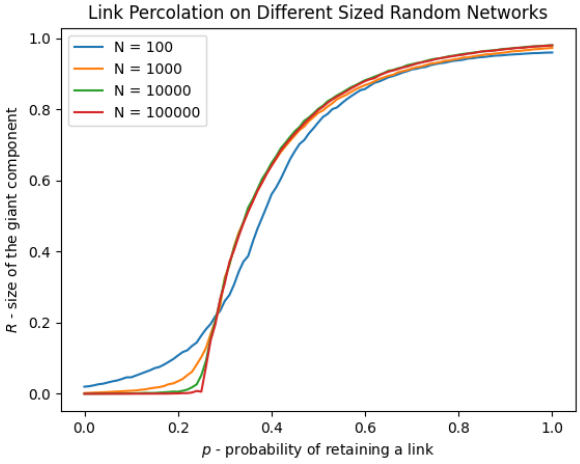


Figure 3: Link percolation on a random networks of different size with $\langle k \rangle = 4$. For each network of size N, there were $100,000/N$ graphs sampled to get smooth data.

More generally, for Erdős-Rényi networks, the critical probability, p_c, is given by $p_c = \frac{1}{\langle k \rangle}$, where $\langle k \rangle$ is the average degree of the network. So, the larger the average degree, the lower the p_c - so a more robust network. This is because the critical threshold for random networks is when the average degree of the damaged network is one[4]. So, more nodes or links need to be removed from networks with large average degree to reach this configuration. Also, as the network gets larger, the critical threshold doesn't change, but the phase transition becomes more and more abrupt.

The Erdős-Rényi random network model is a simple model, but many real world networks have a different structure to random networks. The degree distribution of a random network follows the binomial distribution. However, most real network have a power law in their degree distribution.

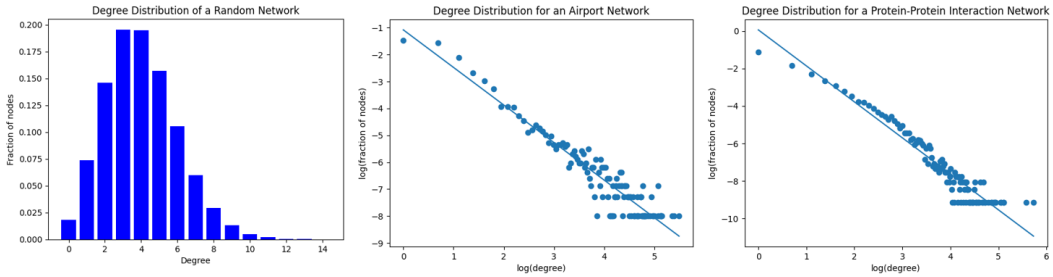


Figure 4: From left to right: The degree distribution of a random network with $N = 10,000$ and $\langle k \rangle = 4$, the degree distribution of an airport network[10][8] ($N = 2918$) and the degree distribution of a protein-protein interaction network[10] ($N = 9527$). These networks were obtained from the Network Repository[10].

The linear relationship in the log-log graphs seen in the two networks shows that their degree distribution follows a power law - these are scale free networks. This means that lots of nodes have very few links, whereas a small number of nodes have lots of links - these are the hubs.

The Barabási-Albert model was put forward to try and explain this observation [1]. In this model, the network starts of with only one node. Then, each time a new node is added which is attached to m new nodes, with the probability of attaching to any existing node being proportional to the node's degree. This processes is repeated until the total N nodes are added.

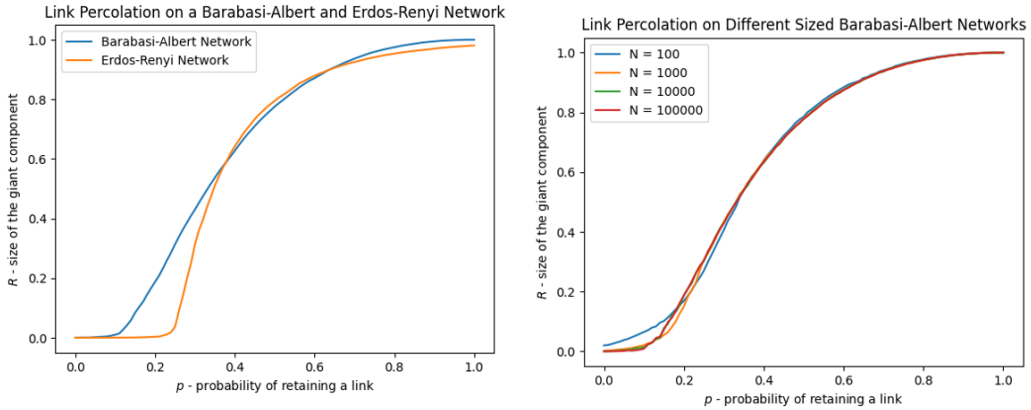


Figure 5: Percolation on Barabási-Albert Networks, comparing the robustness to random networks and considering different sized networks

Performing percolation on this network shows that Barabási-Albert networks are more robust to random failure than random networks. As seen in Figure 5, for almost all p values, the size of the giant component is larger in the Barabási-Albert network, and the critical threshold is lower. This is because when most nodes are removed in a Barabási-Albert network, this has limited effect on the network structure. Most of these nodes have small degree. So, as long as the hubs aren't affected, which has a very low probability, the effect of removing a node is much less for a Barabási-Albert network. Furthermore, as the network size increases, the critical threshold goes to zero for a Barabási-Albert network. This means that the giant component always exists and the network is very robust to failure. However, on the converse, for epidemic spreading this means that even a disease with very low infectivity can lead to an outbreak in a scale free network.

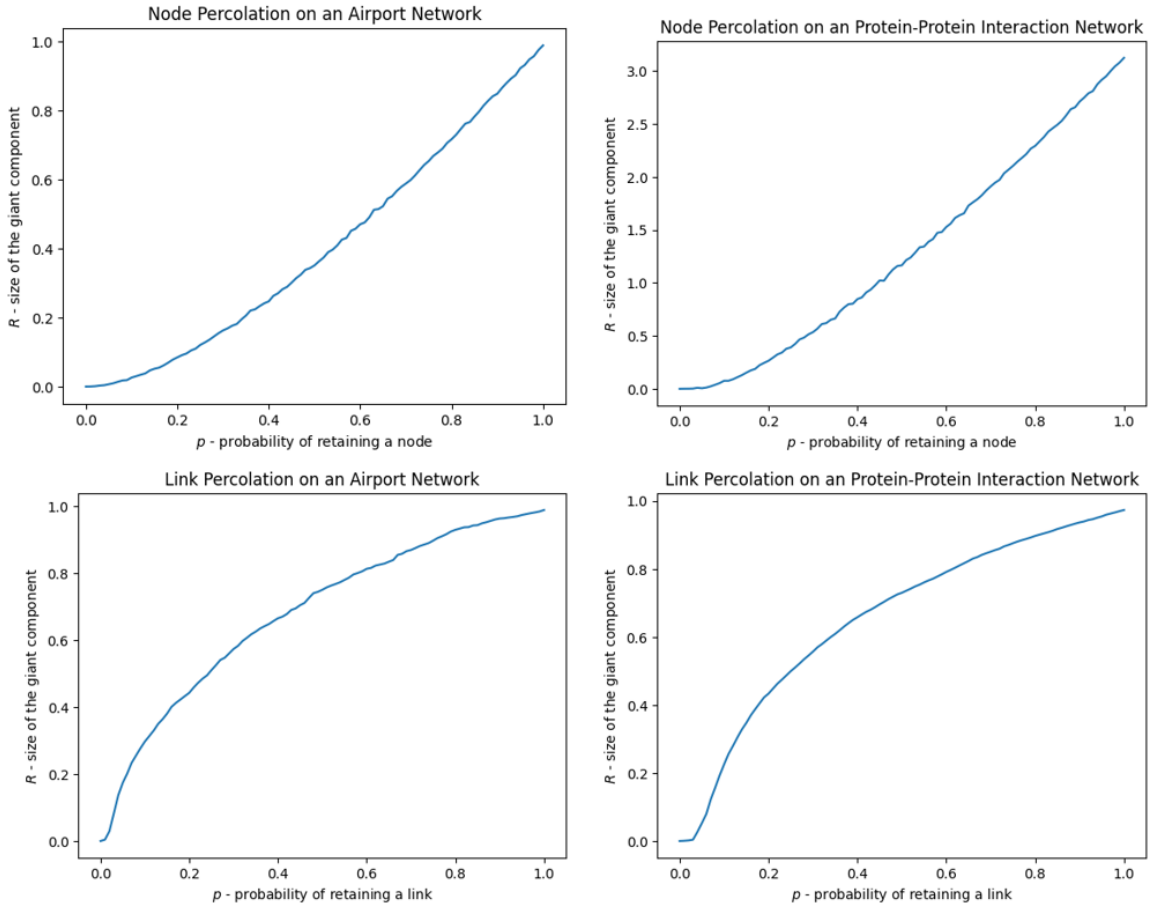


Figure 6: Percolation on an Airport Network and Protein-Protein Interaction Network (The percolation was repeated 20 times on the airport network and 5 times for the protein-protein interaction network)

As seen above, for both the protein-protein interaction network and the airport network, the critical threshold is very close to 0. So, both these networks are very robust compared to random networks of the same size and average degree. This means that even if mutations occur and some proteins malfunction, the cell can continue to operate normally. And, even if there are delays in some airports, air traffic can still operate mostly as normal.

Taking the idea of hubs to the extreme, if we want to design the most robust network to random failures for a certain number of nodes and links, it turns out to have a bimodal degree distribution[9]. In this network there is one central hub which has very large degree, and all

the rest of the nodes have the same small degree. This network structure is also surprsingly tolerant to attacks to the network as well, since even when the hub is removed there is still an almost uniform network remaining.

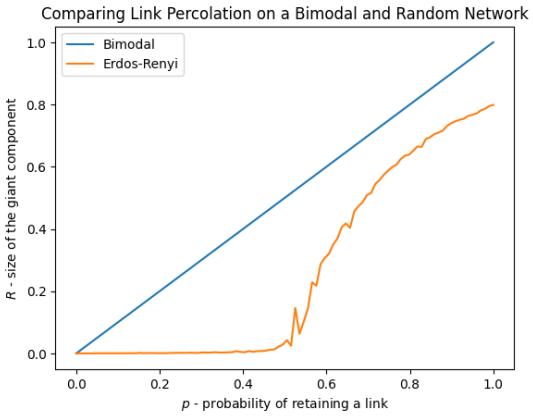


Figure 7: Link percolation on a random network and Bimodal network, both with $N = 10,000$ and $\langle k \rangle \approx 1$.

3 Cascading Failures

Networks make systems robust, but the interconnectivity also leaves them vulnerable to huge cascading failures. Even one small failure in the network can affect the whole network's structure. Power grid blackouts, trophic cascades and Twitter information cascades are all examples of cascading failures. In 1996, failure in a single transmission line caused a huge blackout in 11 US states and 2 Canadian provinces [6]. In Yellowstone National Park, the introduction of only 10 wolves completely changed the ecosystem, even changing the shape of the rivers. In all of these systems, the failure of a node is dependent on the failure of its neighbours - this is the cause of cascading failures.

One model for cascading failures is the failure propagation model [2]. Here, every node in the network is either active or inactive. All nodes start as active at the start. Then, one node is set to be inactive. Consequently, at every time step, each node follows the threshold rule:
• If the node is inactive, then it remains inactive.
• If the node is active and the fraction of its neighbours which are inactive is above the threshold, it becomes inactive. Otherwise, it stays active.

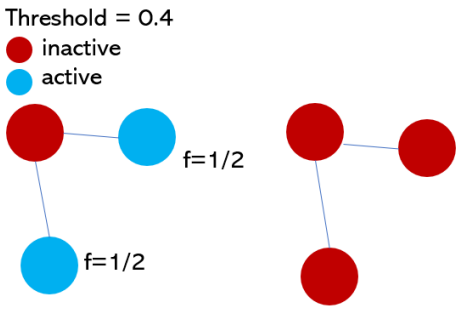


Figure 8: An example network following the rules of the failure propagation model at one time step and then the next.

Performing a simulation of this model, we can see that most failures are very small in size, but a few result in huge cascades. As seen in Figure 9 the frequency of the failure against its size follows a power law, with the exponent being -3/2 for random networks. However, the behaviour of the failure propagation model also depends on the average degree of the network. If the average degree is small, many failures will result in large cascades (since most nodes have few neighbours, even a single failure can result in many nodes passing the threshold). However, for networks with larger average degree, cascades are much less likely to occur.

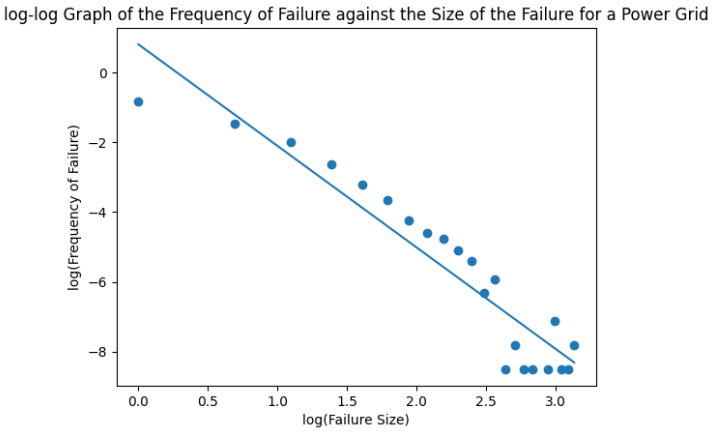


Figure 9: The distribution of the size of failures on a power grid network[10][11] from the western USA with $N = 4941$, giving an exponent of -2.9.

4 Boolean Feedback Network Model

There are lots of networks where there is feedback between the individual nodes. For example, in gene regulatory networks, one gene might produce a protein which acts as a transcription factor which promotes/inhibits the expression of another gene. Or, in foodwebs, increased numbers in the predator would result in a negative effect for the number of prey. For neurons in the brain and neural networks in machine learning, the firing of a neuron affects whether all the neurons connected to it fire as well. As a simplified model taking inspiration from situations like these, I decided to investigate a boolean feedback network model.

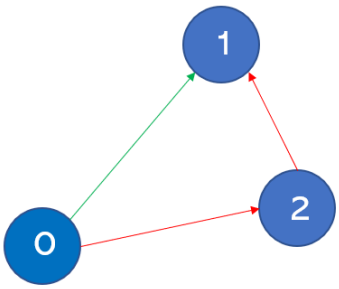
In this model, there is a directed network. And, each directed link is either positive(weight +1) or negative(-1). Then, each node is one of two states - either ON(1) or OFF(0). At the start, all nodes start of as ON(1). Then, at each consequent time step every node follows the same rule. The state of each node is calculated using the edge sum,

$$S_i = \sum_j w_{ij} S_j,$$

where the nodes j are all the nodes which have a directed edge to node i, w_{ij} is the weight o the directed edge from j to i, and S_j is the state of node j.

Then, the following rule is followed to determine the state of each node at the next time step:

$$S_i(t+1) = \begin{cases} 0 & S_i(t) < 0 \\ 1 & S_i(t) > 0 \\ S_i(t) & S_i(t) = 0 \end{cases}$$



Time step	State of node 0	State of node 1	State of node 2
t=0	1	1	1
t=1	S=0 unchanged 1	S=(+1)*1+(-1)*1=0 unchanged 1	S=(-1)*1=-1 Set to OFF 0
t=2	S=0 unchanged 1	S=(+1)*1+(-1)*0=1 Set to ON 1	S=(-1)*1=-1 Set to OFF 0
t=3	S=0 unchanged 1	S=(+1)*1+(-1)*0=1 Set to ON 1	S=(-1)*1=-1 Set to OFF 0

Figure 10: An example network following the rules of the boolean feedback model

I investigated the ratio of ON nodes across time in simulations of this model with randomly generated networks.

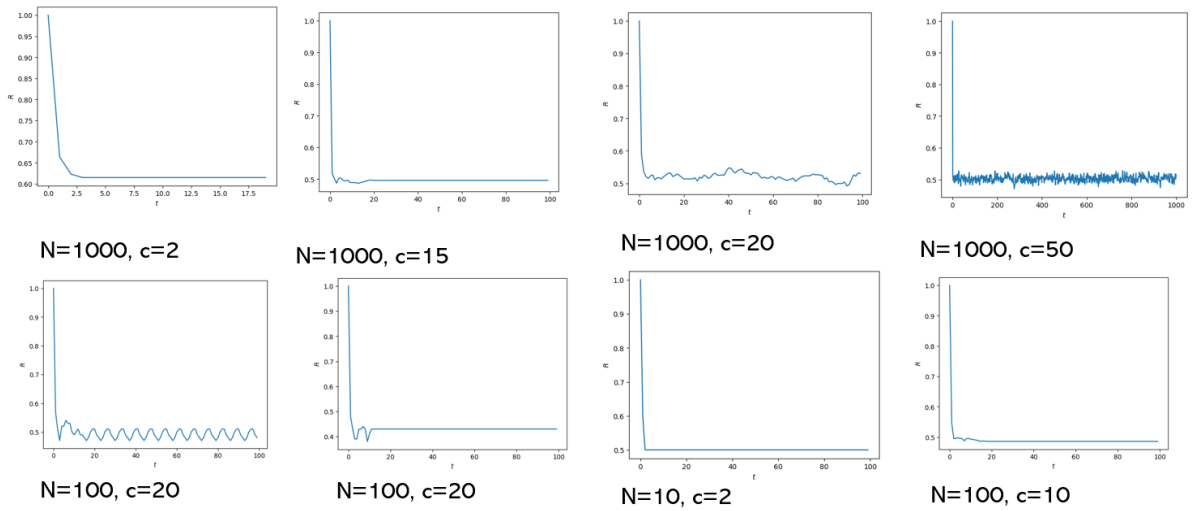


Figure 11: The time evolution of the ratio of ON nodes in many different random networks (here $\langle k \rangle = c$)

The network either stabilises to a single state, falls into a cycle or exhibits chaotic behaviour. However, in the end the system will revisit a state it was previously in (since the space of 2^N possible states of the boolean network is finite) and fall into a steady attractor. From these numerical simulations, I observed patterns in the ratio which the networks stabilised to and how much chaotic behaviour the networks showed before stabilising to a steady state. So, I performed many numerical simulations.

Testing out how long the network takes to stabilise, we can see that larger and more densely connected networks show more stochastic behaviour before stabilising. Furthermore, for small degree networks, we can see that on average the ratio which the networks stabilise to isn't 0.5, and shows the pattern seen in Figure 12. Furthermore, this ratio doesn't seem to depend on the size of the network either. The reason for this pattern is unclear - it could be an inherent part of the system or just an artefact of the specific way the model is designed.

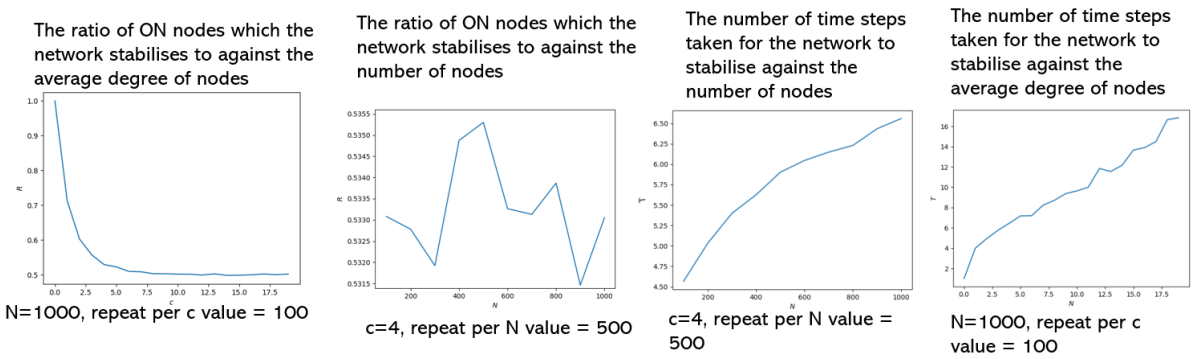
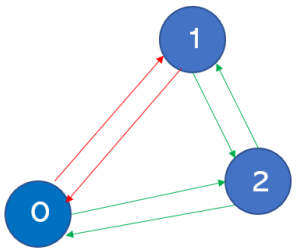


Figure 12: Graphs on how the ratio of ON nodes the graph stabilises to and how long it takes to stabilise depend on the size and average degree of the network

Furthermore, I also tested a very similar model but with the following rule for determining the state of each node:

$$S_i(t+1) = \begin{cases} 0 & S_i(t) \leq 0 \\ 1 & S_i(t) > 0 \end{cases}$$

This model showed very similar behaviour to the previous model. However, cyclic patterns appeared much more frequently.



Time step	State of node 0	State of node 1	State of node 2
t=0	1	1	1
t=1	S=(-1)*1+(+1)*1=0 Set to OFF 0	S=(+1)*1+(-1)*1=0 Set to OFF 0	S=(+1)*1+(+1)*1=2 Set to ON 1
t=2	S=(+1)*1=1 Set to ON 1	S=(+1)*1=1 Set to ON 1	S=0 Set to OFF 0
t=3	S=(-1)*1+(+1)*1=0 Set to OFF 0	S=(+1)*1+(-1)*1=0 Set to OFF 0	S=(+1)*1+(+1)*1=2 Set to ON 1
t=2	S=(+1)*1=1 Set to ON 1	S=(+1)*1=1 Set to ON 1	S=0 Set to OFF 0

Figure 13: An example repeated cyclic pattern for this model.

Small networks also displayed lots of cyclic patterns, and chaotic behaviour was observed for larger, more dense networks.

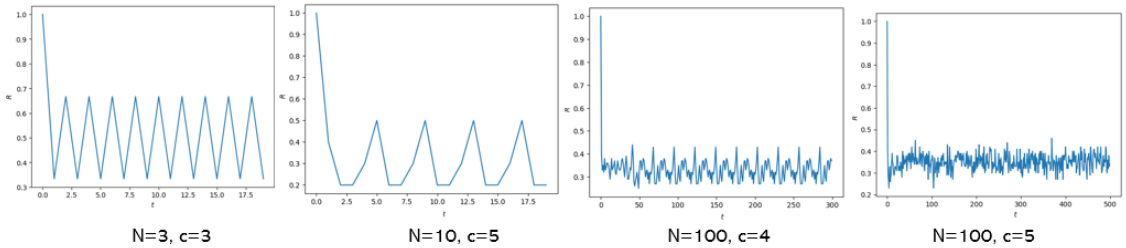


Figure 14: Graphs of how the ratio of ON nodes varies across time for four networks.

These models can provide insight into similar real world situations. Previously, almost the exact same model as the first boolean model was used to study the cell cycle regulatory network in yeast to show that the network is very stable and most initial states lead to a few stable states [7]. However, these are very simplified models, so they only provide ideas on the general properties and dynamics of real world systems.

5 Conclusion

I simulated lots of different networks to study their robustness and dynamic behaviour. I managed to see that both synthetically generated networks and real world data follows the existing theory underlying the robustness of networks. Lots more data analysis, theoretical work, and review of existing literature would be required to understand the new patterns seen in the boolean feedback network model. In the future, more accurate real world models could be constructed to analyse specific scenarios.

6 Acknowledgements

I would like to thank Prof. Ginestra Bianconi and Hanlin Sun for providing lots of help and feedback during this project, as well as Mr Wai Shun Lau for organising the ORIS project. I would also like to thank my family and friends for supporting me throughout this project.

References

[1] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[2] Albert-László Barabási. *Network Science*, chapter Network Robustness. Cambridge University Press, 2016.

[3] Ginestra Bianconi. Introduction to network theory, May 2021.

[4] P Erdös and A Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[5] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[6] Dmitry N Kosterev, Carson W Taylor, and William A Mittelstadt. Model validation for the august 10, 1996 wscc system outage. *IEEE transactions on power systems*, 14(3):967–979, 1999.

[7] Fangting Li, Tao Long, Ying Lu, Qi Ouyang, and Chao Tang. The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences*, 101(14):4781–4786, 2004.

[8] Tore Opsahl. Why anchorage is not (that) important: Binary ties and sample selection. 2011.

[9] Gerry Paul, T Tanizawa, Shlomo Havlin, and H Eugene Stanley. Optimization of robustness of complex networks. *The European Physical Journal B*, 38(2):187–191, 2004.

[10] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[11] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-world networks. *nature*, 393(6684):440–442, 1998.