Robustness of Kronecker Graphs

- exploring erroneous networks of varying sizes -

by

Grant Conybear

Gururaj Desai

JohnBordrick Nnadozie

Lena Schwertmann

Module: Analysing Networks (# 82006000)

Examiner: Prof. Peter Niemeyer

Date of Submission: August 15, 2019

Management & Data Science

Master of Science

Leuphana University of Lüneburg

Abstract

The main objective of this paper is to ascertain the robustness in erroneous networks. In this paper, we have generated Kronecker graphs using kronfit and krongen algorithms based on the real-world networks, Dolphins and Jazz. We have used error mechanisms add edge, remove edge, add node, remove node to create erroneous networks using different centrality measures like degree, eigenvector and pagerank. We have calculated robustness for the generated Kronecker graphs along with original networks and we used these robustness to compare the robustness of the generated kronecker graphs of different sizes. Robustness is calculated using Kendall's Tau correlation by comparing order list of centrality values of hidden and measure kronecker graphs. Throughout the paper, we have implemented and discussed robustness by generating error networks using 10% and 30% error levels with error mechanisims. From the observations, we have found that Dolphins network robustness is not as responsive to network size compared to Jazz network. We observed that removing nodes error mechanism has proven to produce robustness values closer to the original network.

Contents

1 Introduction

Social networks presently have a fast increasing impact especially considering the continuing growth of web-based services like Facebook and Twitter. The fundamental challenge for any social network analysis is the identification of the key component within the social network, hence social network analysis offers a lot of measures for quantifying a member's interconnectedness within social networks (Landherr et al., 2010). This is achieved by using centrality measures, which offer different ways to identify the nodes which are considered being the most "important" ones (Newman, 2010). What is exactly considered as important, depends on the respective measure, meaning that they each highlight different characteristics of the same network.
But in the process of collecting data to create real networks, measurement errors happen. As Wang et al. (2012) (amongst others) summarized, there are various possibilities for these, e.g. too many edges or missing nodes. In case of these erroneous networks, the general understanding of robustness as "resisting changes" can also be applied: Networks can be robust in the sense that basic characteristics do not change even if the networks are altered through errors. Martin and Niemeyer (2019) first illustrated the importance of using robustness to quantify the changes that erroneous networks inflict on centrality measures.
This approach extends their work by investigating the effect network size has on robustness. To achieve this, Kronecker graphs based on the real-world networks Dolphins and Jazz are used. It is then discussed, how the robustness of the Kronecker graphs differs from the original networks that were used to create them.
Chapter 2 discusses the characteristics of the two networks Dolphins and Jazz. In chapter 3 we then introduce the error mechanisms and the robustness measure after Martin and Niemeyer (2019). Then, chapter 4 introduces the Kronecker theory and the algorithm used to create the graphs. The implementation of the simulations are explained in chapter 5, the results presented in chapter 6 and discussed in chapter 7, finishing with a short conclusion in chapter 8.

2 Description of Underlying Networks

2.1 Dolphins Network

The study captures the behavior of different bottlenose dolphin populations between 1994 and 2001 across regions and climates. The primary objective of the study is to observe the social network of bottlenose dolphins – in Fjords system where survival may entail a greater level of co-operation, hence community structure and stability (Lusseau et al., 2003). Furthermore, it observes that members within the community are relatively closely associated and are characterized by fluid and dynamic interactions and some degree of

dispersal from the natural group by both sexes.
This undirected, unweighted network consists of 62 nodes - which are the bottlenose dolphins and 159 edges which indicates a frequent association. The diameter of the graph is 8 edges and it's density is equal to 0.084.

2.2 Jazz Network

Similar to the Dolphins Network, this work studies the categorization and the community structure of the collaboration network of jazz musicians to create a social network (Gleiser and Danon, 2003). The analysis reveals the presence of communities which have a strong correlation with the recording locations of the bands. Here, the nodes corresponds to Jazz bands. Two nodes are connected by an edge, if they have at least one musician in common.
The network consists of 198 nodes and 2,742 edges. It has a diameter of 6 edges and a density of 0.141.

3 Robustness of Networks

In order to quantify robustness, the approach of Martin and Niemeyer (2019) is used. They introduced the concept to quantify the changes that erroneous networks inflict on centrality measures. This is of interest for research on networks, because the robustness thus represents a way of assessing whether drawing conclusions from erroneous networks is valid despite the errors or not. Some networks and some centrality measures might show a high robustness, despite containing a lot of errors, some might not. To be able to calculate this measure, the respective error mechanism needs to be known, or assumed in a reasonable way. The error mechanisms which are considered here are the same ones that Martin and Niemeyer (2019) used:

1. Randomly **missing nodes with uniform probability**. A quantity of nodes that are missing is specified, which are picked at random from all nodes with the same probability. The edges connected to these nodes are also missing.

2. Randomly **missing edges with uniform probability**. The mechanism is the same as for the missing nodes. All edges have the same probability to be missing.

3. **Edges are missing proportionally** to the sum of the degree centrality of the endpoints of an edge. That means that the higher this degree sum, the higher the probability that this edge is missing.

4. Randomly **added edges with uniform probability**. Every edge that does not already exist in the respective network, has the same probability to be added to the network.

These error mechanism are applied to the real-world networks Jazz and Dolphins that are described in chapter 2. Furthermore, Kronecker graphs of varying sizes are used which are

discussed in theory and practical implementation in chapter 4. The centrality measures that are considered here are degree centrality, eigenvector centrality and pagerank.

For the derivation of the robustness measure, we consider an undirected, unweighted graph G with the vertex set $V(G)$ and the edge set $E(G)$. Any of the centrality measures mentioned above is represented by c, where $c(G)$ is a vector that contains the value of the respective centrality measure for each node of G. The value for a specific node $u \in V(G)$ is given by $c_G(u)$. In order to measure the effect of an error mechanism on a graph with respect to a centrality measure c, the correlation between the original graph G and the respective erroneous graph G' is calculated. But treating $c(G)$ and $c(G')$ like continuous variables and calculating the "normal" Pearson correlation, does not lead to meaningful results. According to the approach of Martin and Niemeyer (2019), robustness is not about how much the values of the centrality measure changes, but whether the pattern of centrality measures is changed noticeably. This is why rank correlation, more specifically the Kendall's tau ("tau-b" version, Kendall (1945)) rank coefficient is used. Also, this approach reduces the influence of outliers (Martin and Niemeyer, 2019; Wang et al., 2012). To calculate Kendall's tau, the concept of concordant and discordant node pairs is essential. This means that the order of the centrality measure values matter, not the values themselves. Kendall tau correlation is high, if the order of values is very similar, also including ties where the values are the same. In a more formal way, a node pair $u, v \in V(G) \cap V(G')$ with $u \neq v$ is defined as concordant, if the nodes have different values for c and the order in the unchanged network G is the same as in the erroneous network G'. This means that either $c_G(u) < c_G(v) \wedge c_{G'}(u) < c_{G'}(v)$ or $c_G(u) > c_G(v) \wedge c_{G'}(u) > c_{G'}(v)$ is true. Accordingly, a node pair is considered discordant, if the order of the respective node pair is not the same in the two networks. Ties are also considered, meaning that the values for c are the same. These measures are then used to calculate the robustness ρ_c for two given networks G and G':

$$\rho_c(G, G') = \frac{n_c - n_d}{\sqrt{(n_c + n_d + n_t) * (n_c + n_d + n_{t'})}} \tag{1}$$

where n_c is the number of concordant and n_d the number of discordant pairs. The ties are represented by n_t for ties that occur in $c(G)$ and $n_{t'}$ for ties that occur in $c(G')$.
Equivalent to the "normal" Pearson correlation coefficient between continuous variables, the range of Kendall's tau is $[-1, +1]$.

4 Kronecker Theory

4.1 Motivation

In the field of graph theory and network analysis, real networks are a prized possession. However, the construction of a real network can be incredibly tedious and time consuming due to the requirement of a rich source of data. Even more, these real networks are absolutely essential to the progression of the field due to their specific realistic network structures, which have been proven difficult to replicate by certain random graph methods (Leskovec et al., 2010).

Particularly, natural graphs must uphold certain properties. One of these properties calls for natural graphs to possess a power-law degree distribution. The degree of a node represents the amount of connections that specific node has with other nodes. Therefore, a graph with a power-law degree distribution is one where in the distribution of all nodes degrees, a small number of nodes have a high degree, and a large number of nodes have a low degree. This phenomena is present in most natural graphs spanning across a variety of disciplines. Another property commonly found in real networks (as well as the study of fractals (Ravasz and Barabasi, 2003)) is that of self-similarity, which simply says that the structure of the connections between certain sections of a network are similar to the connections within those sections of the network.

In an effort to maintain these properties and produce structurally natural graphs, the Kronecker product can be incorporated to an underlying real network to generate structurally-similar graphs without going through the hassle of gathering copious amounts of data.

4.2 Mathematical Formulation

4.2.1 Kronecker Product

As mentioned previously, Kronecker graphs are derived from the concept of a matrix operation analogously named the Kronecker product. The implementation of the Kronecker product will allow for a recursive construction of self-similar graphs (Leskovec et al., 2010). The Kronecker product acts on two matrices, denoted as A and B, where A is of dimension $m \times n$ and B is of dimension $p \times q$. The resulting Kronecker product of these matrices will have a dimensionality of $(mp) \times (nq)$. More formally, for any matrices A and B with dimesions $m \times n$ and $p \times q$ respectively, this relationship can be represented as:

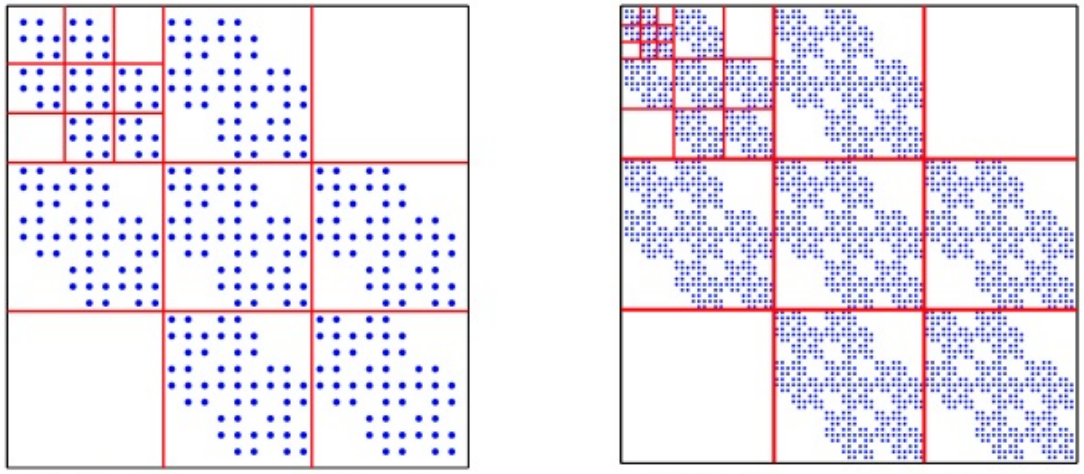
$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix}$$

Since networks can be represented by adjacency matrices, the concept of the Kronecker product naturally extends to graphs as well. Formally, if R and S are graphs with cor-

responding adjacency matrices $A(R)$ and $A(S)$, then the Kronecker product of the two graphs is the graph with the adjacency matrix C, where $C = A(R) \otimes A(S)$.

4.2.2 Kronecker Power

Kronecker graphs do not only have to be the result of a Kronecker product between two distinct graphs, but can also be the result of a Kronecker product of a graph with itself. As previously mentioned, this process is done iteratively to create self-similar graphs. The notion of recursively taking the Kronecker product of a matrix by itself yields to a generalized representation where the k^{th} power of some matrix K is the Kronecker product of K taken k times ($K \otimes K \otimes K....K$), henceforth denoted as $K^{[k]}$.
The figure below depicts the expansion of the adjacency matrix of a 3-node graph using the Kronecker power.



(a) K_3 adjacency matrix (27×27) (b) K_4 adjacency matrix (81×81)

Figure 1: Kronecker Power (Leskovec et al., 2010)

4.3 Stochastic Kronecker Graphs

While self-similar graphs are useful in certain scenarios, there is much more potential if random graphs are able to be generated. It turns out that structurally-natural random graphs derived from the Kronecker power can be realized with a small tweak to the adjacency, or "initiator", matrix of the underlying network. Normally, the elements of an adjacency matrix will take on binary values, where a 1 indicates a link between two nodes, and a 0 indicates no link. Instead, stochasticity can be introduced into the Kronecker graph by allowing for probability values in the adjacency matrix instead of the usual binary (Leskovec et al., 2010). Therefore, the elements in the initiator matrix will fall

within the interval of $[0, 1]$, where a value closer to 1 suggests that an edge is likely between the two nodes, and a value closer to 0 suggests the opposite.
To generate a Kronecker graph from this newly-defined method of representing the initiator matrix, the Kronecker power can be taken, which will yield a large matrix of probabilities. From this resulting matrix, each element can be sampled to discover whether or not an edge exists between the corresponding nodes.

4.3.1 Naïve Algorithm

Using the previously outlined format, the most rudimentary method to obtain Kronecker graphs would be to use the Kronecker power to generate a probabilistic adjacency matrix, and then sample each value to determine whether an edge is present or not. This process can be summarized in the following steps.

- Goal: Given $k \in (1, \ldots, K)$ iterations and an $n \times n$ initiator matrix denoted as P_1, compute P_1^K, which is the the Kronecker power of P_1 after all iterations. From P_1^K, generate a graph.
- Step 1: P_1^K is a matrix of probability values, where each element represents the probability that two nodes (the u^{th} row and v^{th} column) have an edge. Given P_1, the formula for the $(u, v)^{th}$ element of P_1^K, or p_{uv}, is:

$$p_{uv} = \prod_{i=0}^{k-1} P_1[(\frac{u-1}{n^i}) \pmod{n+1}, (\frac{v-1}{n^i}) \pmod{n+1}]$$

- Step 2: Now that P_1^K has been found, the actual graph can be generated. For each element in P_1^K, the occurrence of an edge between the corresponding nodes is the result of a sample drawn from a Bernoulli distribution parameterized by p_{uv}.

After k iterations of the Kronecker power, the resulting matrix will be a squared matrix with dimension n^k, which can be denoted as N. Therefore, the time complexity of the algorithm will be $O(kN^2)$.

4.3.2 `Kronfit` Algorithm

In an effort to decrease the complexity of generating Kronecker random graphs, the problem can be reformulated as an optimization problem (Leskovec et al., 2010). Like the naive algorithm, P_1 is the initiator matrix. Similarly, P_1^K represents the probability matrix of a Kronecker graph after the k^{th} Kronecker power iteration on P_1. If G is a given graph with n^k nodes, the optimization task is to find an initiator matrix P_1 with the highest probability of generating G. In more explicit terms, this can be represented as:

$$\arg\max_{P_1} P(G|P_1)$$

For mathematical convenience, the authors (Leskovec et al., 2010) substitute the symbol Θ for P_1. Thus, the task is still to search initiator matrices Θ in order to maximize $P(G|\Theta)$. To do this, the log-likelihood is maximized through iteratively updating Θ by taking the gradient of $l(\Theta)$ with respect to Θ. This is the essence of the `Kronfit` algorithm, which is represented in the sequence of steps below.

- Goal: given some starting initiator matrix $\hat{\Theta}_1$, a graph G with n^k nodes, and a learning rate λ, return the parameter matrix $\hat{\Theta}$ that maximizes $P(G|\Theta)$
- Step 1: Initialize starting parameter matrix $\hat{\Theta}_1$
- Step 2: Perform parameter matrix update $\hat{\Theta}_{t+1} = \hat{\Theta}_t + \lambda \frac{\partial}{\partial \hat{\Theta}_t} l(\hat{\Theta}_t)$ until convergence
- Step 3: Return $\hat{\Theta} = \hat{\Theta}_t$

In this shift of perspective, the calculation of the likelihood can be performed in linear time. Our analysis and generation of Kronecker graphs in the following sections of the paper were the result of the `Kronfit` algorithm.

5 Implementation

The main focus of our topic is to see how robustness varies with different sizes of Kronecker versions of the Jazz and Dolphins network. Main steps of our implementation are:

- Generate the Kronecker graphs for Jazz and Dolphins with varying number of nodes:
  - Train the `kronfit` module to find the best initiator matrix for the given graph, in our case Jazz and Dolphins.
  - Use the fitted initiator matrix as an input to the `krongen` module to generate Jazz-Kronecker, Dolphins-Kronecker graphs of varying node size.
- Calculate the robustness for Jazz-Kronecker, Dolphins-Kronecker graphs:
  - Generate the erroneous graphs using the erroneous methods specified in chapter 3
  - Calculate the robustness between the original Kronecker and erroneous Kronecker graphs
- Calculate the robustness for Jazz and Dolphins graphs
- Use percentage error as the error measure to calculate the error between original network of Jazz and Dolphins with different Kronecker graphs

5.1 Generating Kronecker Graphs with `kronfit` and `krongen`

As popular libraries like `networkx` and `igraph` do not have the methods to generate Kronecker graphs directly, we used the snap library (v. 4.1, `https://github.com/snap-stanford/snap-python`). However, the necessary functions `kronfit` and `krongen` were only fully available in the C++ version. As the project is coded in Python (v. 3.7, Python Software Foundation, `www.python.org`), we cloned the C++ repository (`https://github.com/snap-stanford/snap/tree/master/examples/kronfit`, `https://github.com/snap-stanford/snap/tree/master/examples/krongen`) and compiled the `kronfit` and `krongen` modules using the `make` command. After compilation, we were able to get the executable files. The following paragraphs describe how we used them to generate the Kronecker graphs.

First, we used the `kronfit` function to find the fitted initiator matrix for the graphs (Jazz and Dolphins). The command we used to do this was `./kronfit -i:I -n0:2 -m:M -gi:N`. Here, `I` is the edge list of Jazz and Dolphins network, `M` is the initial initiator matrix for which we used the default values "0.9 0.6; 0.6 0.1" as suggested in the ReadMe-file and `N` (= 100) is the number of iterations we ran to fit the matrix. After this, we got the fitted initiator matrix which can be used by the `krongen` program to generate Kronecker graphs.

Secondly, we used the command `./krongen -o:O -m:M -i:N` to generate the Jazz-Kronecker and Dolphins-Kronecker graphs. Here, `O` is the output file name to store the edge list of the network. `M` is the initiator matrix, where `kronfit` resulted in "0.99 0.62; 0.71 0.35" for Jazz Network and "0.93 0.53; 0.52 0.076" for the Dolphins Network. `N` is the network size, which specifies the 2^N number of nodes the generated network should have.

Finally, we ran these commands with different network sizes. The network sizes we have chosen are 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096 and 8192. We generated the edge lists as `.txt` files. These files are the input for the `.py` file that contains the robustness calculation.

5.2 Calculating the Robustness

We used `EstimatingCentralityRobustness` python module (`https://github.com/crsqq/EstimatingCentralityRobustness`) provided by Martin and Niemeyer (2019) to calculate robustness and to generate erroneous networks. Two error levels of 10% and 30%, specified by `alpha = [0.1, 0.3]`, are used. The four different error mechanisms are implemented using the `remove_node_uniform`, `remove_edges_uniform`, `remove_edges_proportional_degree` and `add_edges_random`. Degree centrality, eigenvector centrality and pagerank are calculated using the `networkx` library (`degree_centrality, pagerank_scipy, eigenvector_centrality_numpy`).

The function that is used to calculate the robustness, takes three parameters; which are (1) robustness_calculator, (2) number_of_iterations and (3) network_sizes. The robustness

calculator is the function from the library `EstimatingCentralityRobustness` to estimate the robustness and we have used 3 types of robustness as discussed earlier. The number of iterations is used to specify number of times we calculate the robustness for creating different erroneous graphs using the same error mechanism. We then take the mean and standard deviation at the end to summarize the result. We ran our calculation with 1000 iterations.

6 Results

6.1 True robustness

As we have discussed earlier, we have calculated the robustness for each of the Kronecker graphs we have generated for Dolphins and Jazz networks. We have used degree, eigenvector and pagerank as centrality measure.
In specific, when we look at true robustness values for degree centrality in figure 2, we can see that for the 10% error level, there is no much difference in values between different error measures. But for 30% error level, we can see true robustness varies initially but until network size of 64 nodes and then it also behaves similarly as compared to 10% error level. Robustness looks random initially for 30% error level for both Dolphins and Jazz,

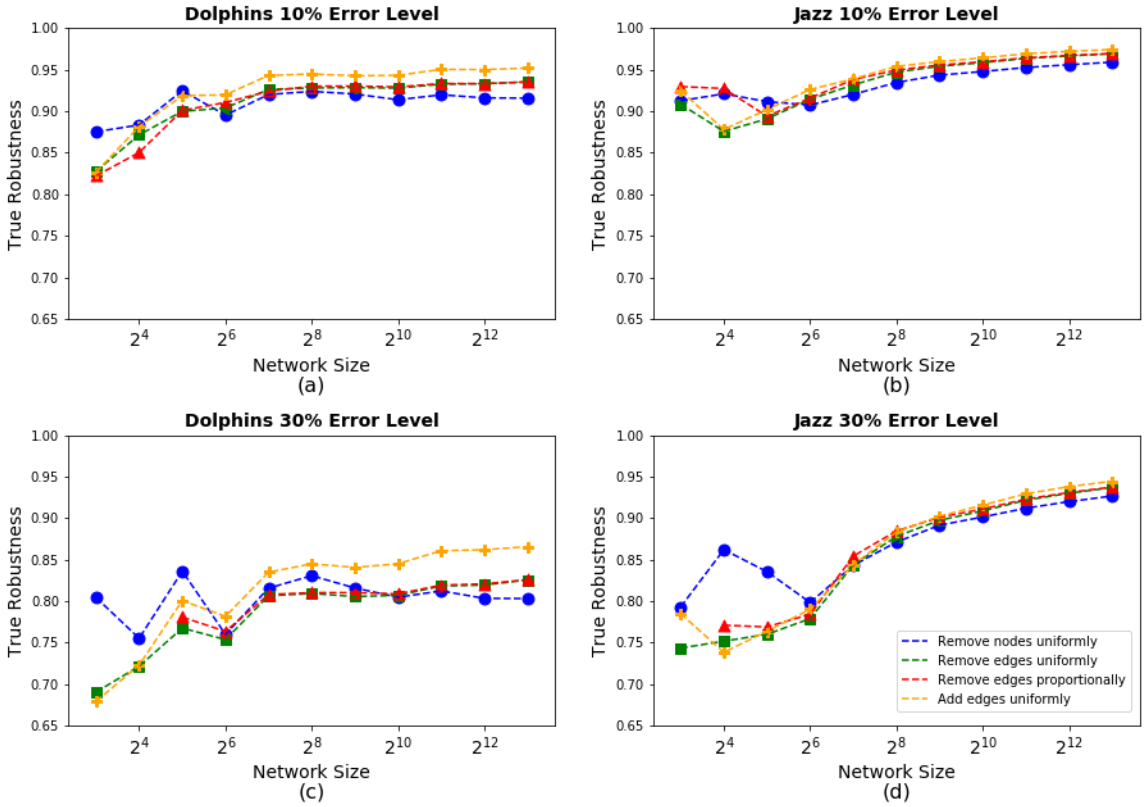


Figure 2: True robustness in regard to **degree centrality** plotted over the network size of Kronecker graphs based on the Jazz and Dolphins network.

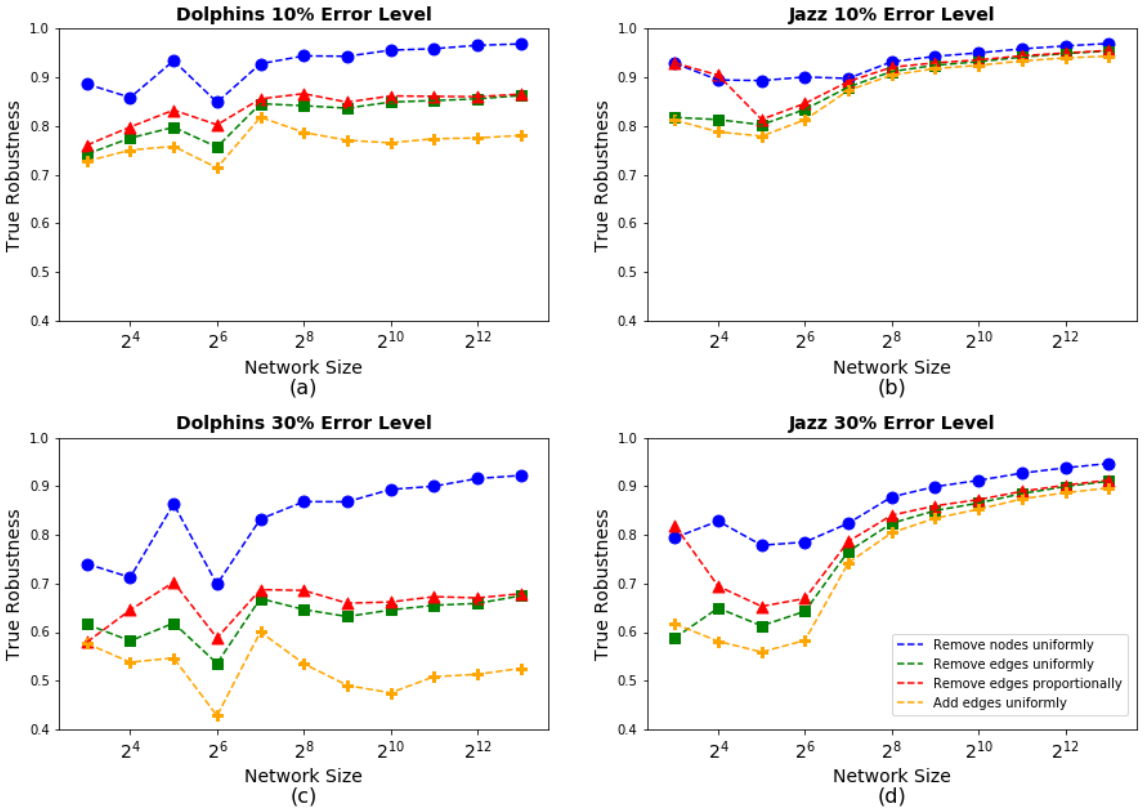


Figure 3: True robustness in regard to **eigenvector centrality** plotted over the network size of Kronecker graphs based on the Jazz and Dolphins network.

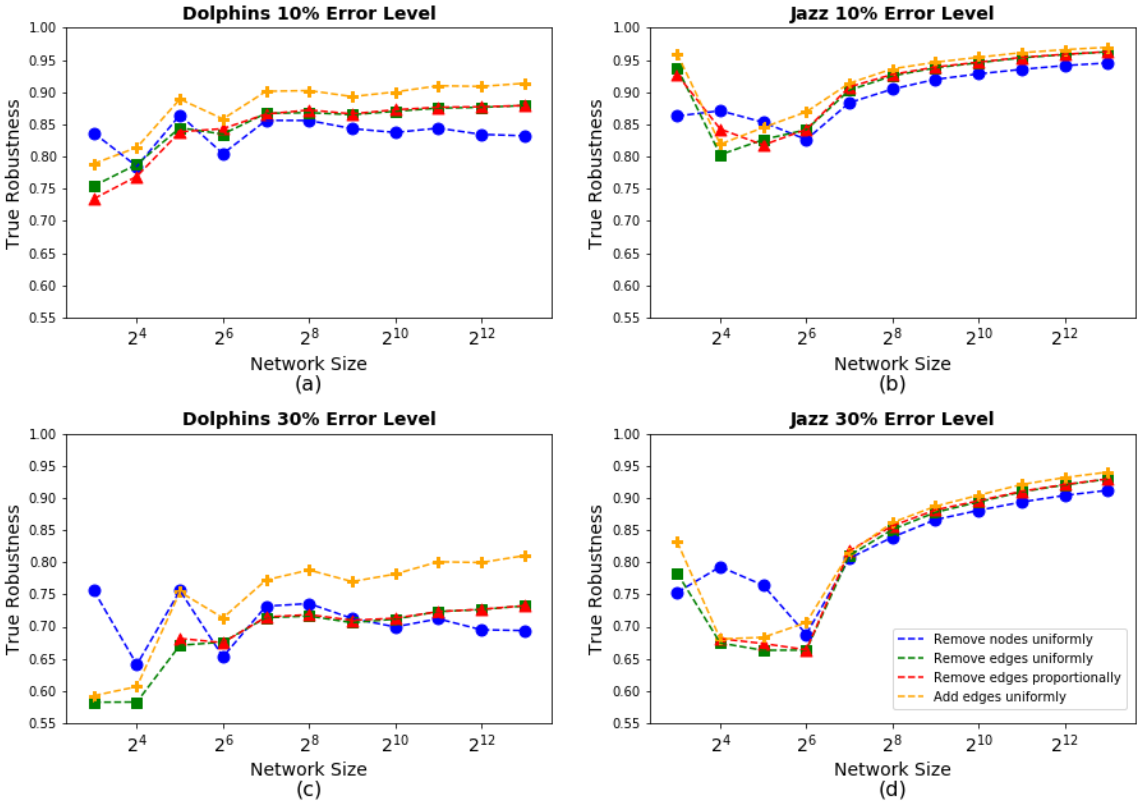


Figure 4: True robustness in regard to **pagerank** plotted over the network size of Kronecker graphs based on the Jazz and Dolphins network.

but does not change much as network size increases.
When we look at true robustness values of eigenvector centrality in figure 3, we can see that the robustness values do not converge to the same value for all error mechanisms for the Dolphins-Kronecker graphs. .For the Jazz network, this is the case, despite some more initial variation than for degree centrality. The Dolphins network specifically has a very low robustness, around 0.5, for the error measure "add edges". Generally, the results of the Jazz network are more consistent.
At last, regarding the pagerank centrality in figure 4, the overall trend of true robustness has same behavior as the other centrality measures, which is it that it increases over network size. When we compare the 10% error level robustness values for both networks, the values are similar but for 30% error level true robustness values change. Overall, true robustness for 30% Error level has less value compared to 10% error level and thus behaves similarly, but less extreme than for eigenvector centrality.
As we have discussed earlier, our implementation resulted in getting both mean and standard deviation values of robustness. We can see from table 1 that the standard deviation values are small in general. From the table, we can see that standard deviation values vary between 0.0003 to 0.51. We could observe that as network size increases, standard deviation values decreases. From this we can say that as the network size increases, our confidence bound for the mean true robustness values also increases and true robustness values become consistent.

Table 1: Results for the standard deviation of the Kronecker graphs based on the Jazz and Dolphins network. Size: Network Size; c: centrality measure with D = degree centrality, E = eigenvector centrality, P = pagerank; Netw.: underlying network with D = Dolphins, J = Jazz

Size	c	Netw.	Rem.Nodes Uni.		Rem.Edges Uni.		Rem.Edges Pro.		Add Edges Uni.	
			10%	30%	10%	30%	10%	30%	110%	30%
8	D	J	0.1063		0.0577	0.1769	0.0556	0.1427	0.0545	0.1228
16			0.0387	0.1575	0.0455	0.0891	0.0457	0.0926	0.0487	0.0684
32			0.0834	0.1368	0.0321	0.0597	0.0320	0.0560	0.0354	0.0491
64			0.0445	0.0787	0.0183	0.0357	0.0180	0.0348	0.0205	0.0332
128			0.0108	0.0213	0.0066	0.0139	0.0060	0.0135	0.0080	0.0145
256			0.0068	0.0128	0.0035	0.0075	0.0032	0.0072	0.0045	0.0080
512			0.0042	0.0077	0.0021	0.0043	0.0018	0.0039	0.0025	0.0045
1,024			0.0029	0.0050	0.0013	0.0026	0.0011	0.0024	0.0016	0.0030
2,048			0.0020	0.0032	0.0008	0.0015	0.0007	0.0014	0.0010	0.0017
4,096			0.0012	0.0020	0.0005	0.0010	0.0004	0.0008	0.0007	0.0011
8,192			0.0008	0.0013	0.0003	0.0006	0.0003	0.0005	0.0004	0.0008
8	E	J	0.1042	0.2822	0.1462	0.2492	0.1630	0.2490	0.0574	0.1385
16			0.1229	0.3881	0.1043	0.1663	0.0993	0.1854	0.0801	0.0831
32			0.1938	0.2542	0.0599	0.1232	0.0715	0.1496	0.0507	0.0674

64			0.1175	0.1489	0.0316	0.0518	0.0352	0.0662	0.0285	0.0400
128			0.0343	0.0434	0.0094	0.0175	0.0101	0.0200	0.0082	0.0144
256			0.0223	0.0272	0.0046	0.0091	0.0048	0.0098	0.0039	0.0066
512			0.0151	0.0181	0.0028	0.0053	0.0030	0.0057	0.0021	0.0038
1,024			0.0114	0.0137	0.0017	0.0033	0.0019	0.0036	0.0014	0.0024
2,048			0.0088	0.0097	0.0011	0.0019	0.0012	0.0021	0.0008	0.0013
4,096			0.0066	0.0070	0.0006	0.0013	0.0008	0.0013	0.0005	0.0008
8,192			0.0053	0.0055	0.0004	0.0008	0.0005	0.0009	0.0003	0.0005
8	P	J	0.1076		0.0628	0.1707	0.0511	0.1326	0.0965	0.1617
16			0.0826	0.1501	0.0826	0.1034	0.0802	0.0988	0.0800	0.0956
32			0.1267	0.1472	0.0470	0.0630	0.0437	0.0636	0.0512	0.0554
64			0.0666	0.0823	0.0436	0.0545	0.0320	0.0444	0.0356	0.0449
128			0.0104	0.0203	0.0079	0.0138	0.0071	0.0129	0.0094	0.0147
256			0.0066	0.0129	0.0043	0.0076	0.0036	0.0070	0.0050	0.0085
512			0.0040	0.0075	0.0022	0.0042	0.0020	0.0040	0.0028	0.0047
1,024			0.0026	0.0047	0.0013	0.0027	0.0012	0.0023	0.0018	0.0031
2,048			0.0017	0.0030	0.0009	0.0015	0.0007	0.0014	0.0012	0.0019
4,096			0.0011	0.0019	0.0005	0.0010	0.0004	0.0008	0.0007	0.0012
8,192			0.0007	0.0013	0.0003	0.0006	0.0003	0.0005	0.0005	0.0008
8	D	D	0.1284		0.0740	0.1252	0.0699	0.1122	0.0720	0.1339
16			0.0920		0.0448	0.1032	0.0453	0.1028	0.0473	0.0886
32			0.0773	0.1299	0.0337	0.0615	0.0307	0.0568	0.0331	0.0479
64			0.0478	0.0761	0.0218	0.0428	0.0199	0.0364	0.0249	0.0405
128			0.0269	0.0489	0.0126	0.0248	0.0117	0.0219	0.0142	0.0235
256			0.0246	0.0401	0.0096	0.0177	0.0076	0.0145	0.0103	0.0146
512			0.0220	0.0318	0.0067	0.0132	0.0056	0.0103	0.0075	0.0111
1,024			0.0190	0.0251	0.0050	0.0093	0.0042	0.0075	0.0057	0.0086
2,048			0.0118	0.0167	0.0035	0.0066	0.0029	0.0050	0.0038	0.0056
4,096			0.0103	0.0138	0.0026	0.0047	0.0020	0.0034	0.0029	0.0041
8,192			0.0072	0.0101	0.0018	0.0032	0.0015	0.0024	0.0021	0.0029
8	E	D	0.3873	0.5101	0.1464	0.2346	0.1365	0.2420	0.1104	0.1764
16			0.1551	0.2583	0.0837	0.1399	0.0846	0.1630	0.0839	0.1090
32			0.1996	0.2483	0.0656	0.0872	0.0610	0.0889	0.0433	0.0544
64			0.1543	0.2126	0.0523	0.0726	0.0542	0.0837	0.0375	0.0468
128			0.1061	0.1365	0.0233	0.0362	0.0269	0.0418	0.0194	0.0255
256			0.1177	0.1620	0.0214	0.0268	0.0238	0.0327	0.0113	0.0156
512			0.1160	0.1553	0.0156	0.0209	0.0229	0.0329	0.0086	0.0107
1,024			0.1005	0.1406	0.0112	0.0150	0.0161	0.0202	0.0051	0.0071
2,048			0.0728	0.0963	0.0077	0.0105	0.0124	0.0131	0.0037	0.0050

4,096			0.0640	0.0790	0.0057	0.0071	0.0091	0.0085	0.0022	0.0032
8,192			0.0462	0.0577	0.0040	0.0052	0.0064	0.0057	0.0016	0.0023
8	P	D	0.1513		0.1711	0.1183	0.1603	0.1266	0.1443	0.1787
16			0.1165		0.0740	0.1064	0.0754	0.1147	0.0853	0.1095
32			0.0741	0.1103	0.0636	0.0766	0.0499	0.0696	0.0602	0.0655
64			0.0604	0.0745	0.0318	0.0449	0.0288	0.0428	0.0364	0.0447
128			0.0312	0.0461	0.0237	0.0325	0.0180	0.0280	0.0208	0.0264
256			0.0309	0.0397	0.0182	0.0246	0.0135	0.0192	0.0162	0.0191
512			0.0281	0.0316	0.0133	0.0173	0.0099	0.0128	0.0129	0.0143
1,024			0.0246	0.0258	0.0101	0.0129	0.0063	0.0083	0.0096	0.0104
2,048			0.0155	0.0171	0.0064	0.0085	0.0047	0.0060	0.0064	0.0070
4,096			0.0129	0.0136	0.0049	0.0060	0.0033	0.0042	0.0048	0.0053
8,192			0.0092	0.0099	0.0036	0.0045	0.0022	0.0030	0.0036	0.0037

6.2 Percentage error between original Kronecker graphs

The figures below depict the effect of the generated Kronecker graphs robustness of varying network size to the true robustness of the respective underlying networks.

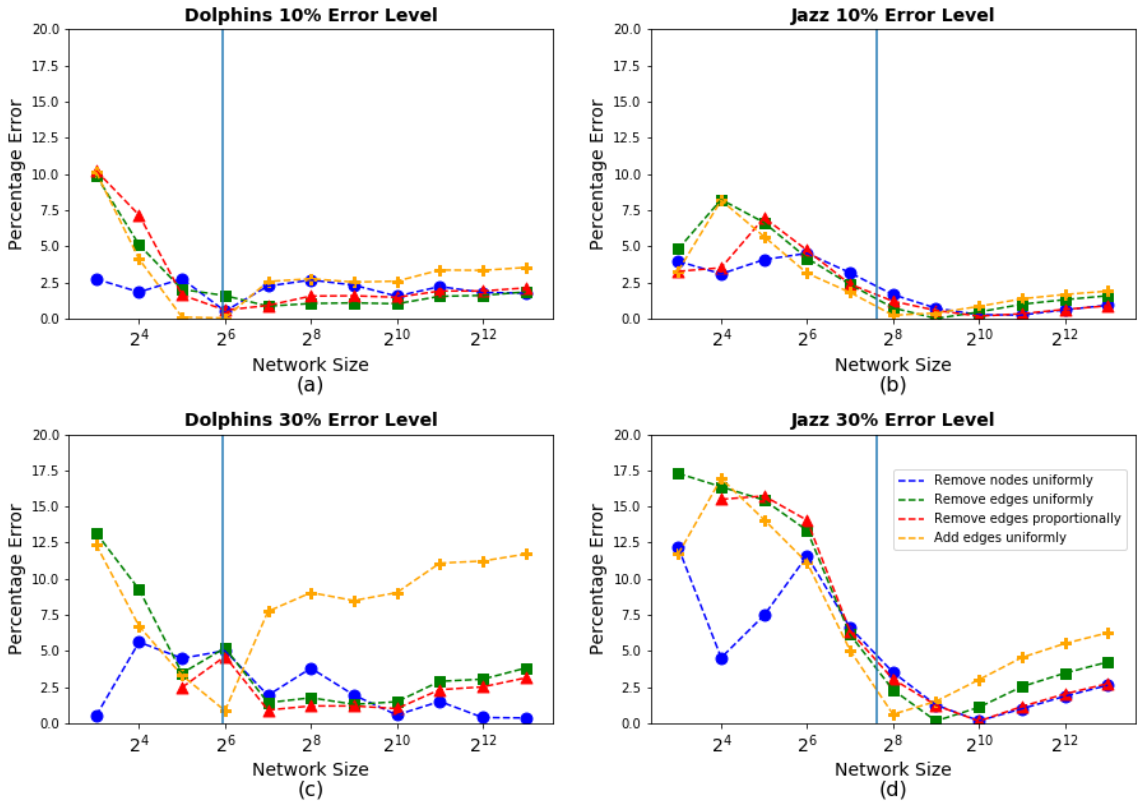


Figure 5: Percentage error of the mean robustness using **degree centrality** calculated in respect to the underlying Dolphins and Jazz networks. The vertical blue line indicates the network size of the respective network.

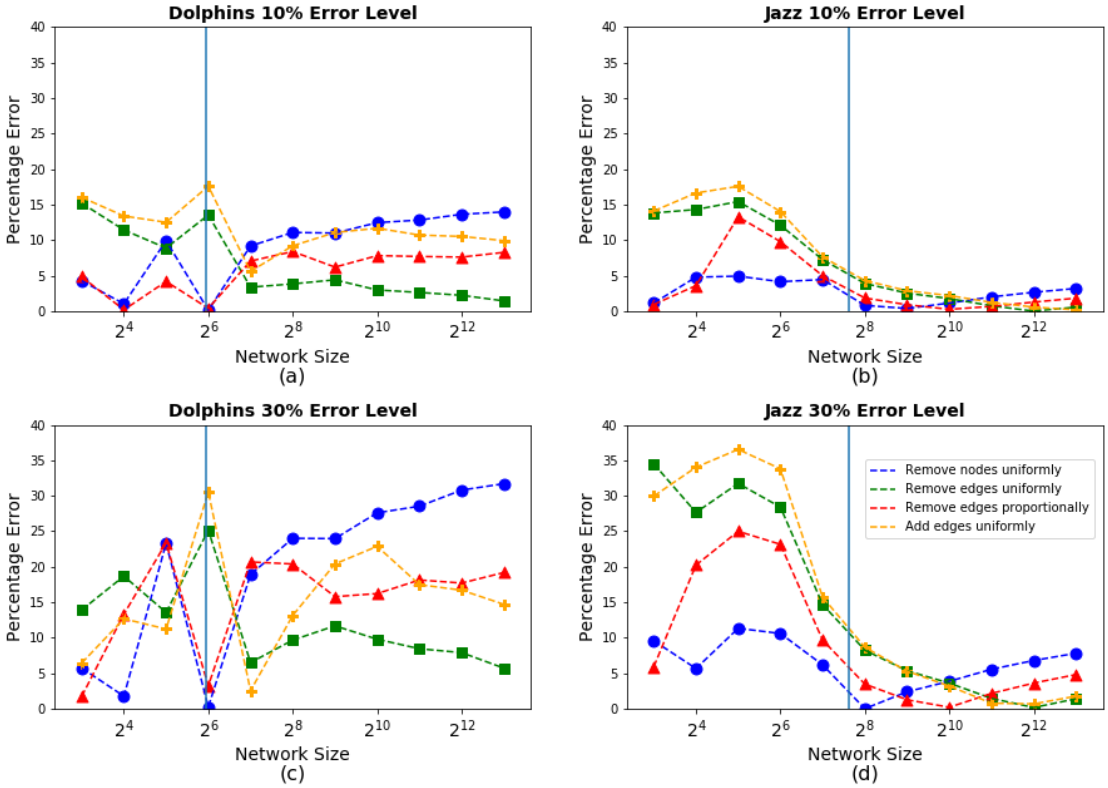


Figure 6: Percentage error of the mean robustness using **eigenvector centrality** calculated in respect to the underlying Dolphins and Jazz networks. The vertical blue line indicates the network size of the respective network.

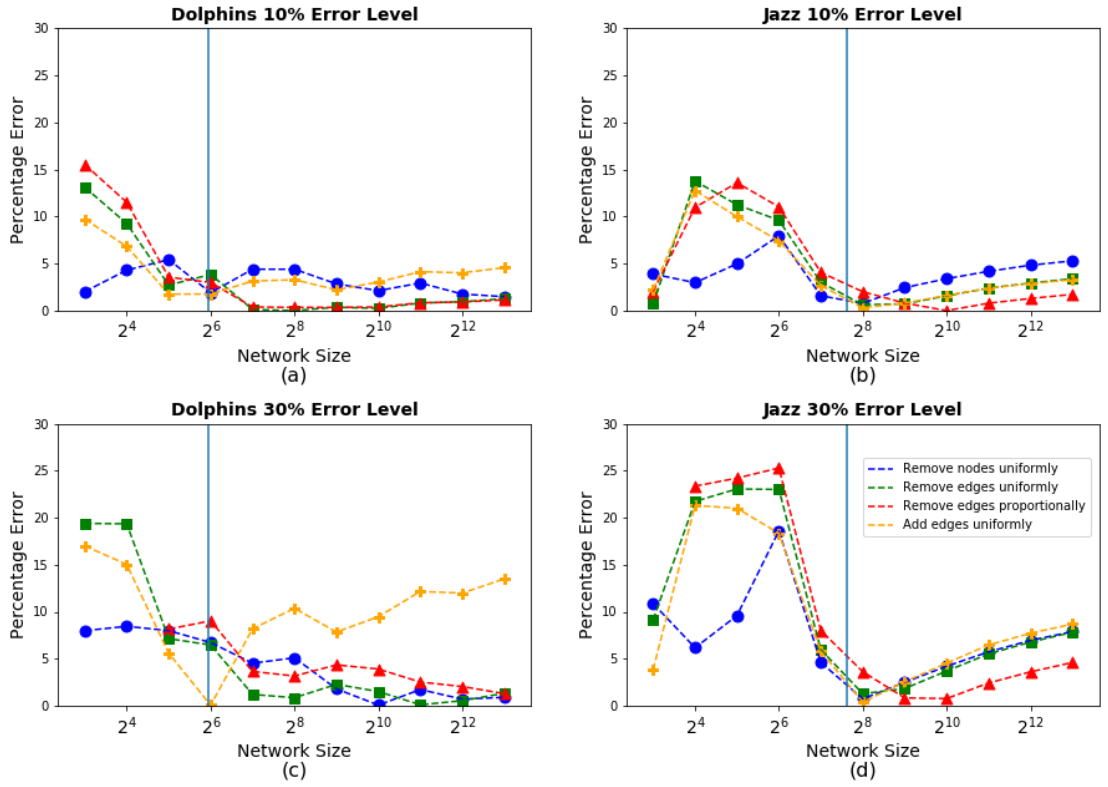


Figure 7: Percentage error of the mean robustness using **pagerank** calculated in respect to the underlying Dolphins and Jazz networks. The vertical blue line indicates the network size of the respective network.

The vertical blue line in each of the plots indicates the size of the underlying network. Instead of plotting the true robustness on the y-axis, the percentage error is plotted, which is calculated as follows:

$$\frac{|\text{Mean Robustness Kronecker graph} - \text{Mean Robustness Underlying Network}|}{\text{Mean Robustness Underlying Network}} * 100$$

In figure 5, the general trend of percent error is downward sloping and fairly consistent for both error levels in both the Dolphins and Jazz network with the degree robustness calculator. The only exception to this seems to be adding edges uniformly in the dolphins network with a 10% error level, which correlates with an increasing percentage error for larger network sizes.
In figure 6, which represents robustness estimates calculated from the eigenvector centrality measure, we can observe a similar trend for all plots except for the dolphins network at a 0.3 alpha level. The Jazz network in particular finds more accurate robustness estimates with increasing network sizes. In comparison to degree centrality, the results are diverging more for lower network sizes.
Figure 7 (which implements pagerank centrality estimates) demonstrates a similar relationship to Figure 5 in that increasing network size seems to correlate with more accurate robustness estimates except in the case of adding edges uniformly in the Dolphins network at an error level of 30%. The other plots of the Dolphins network exhibit a downward sloping behavior with larger graph sizes. For the Jazz-Kronecker graphs, the error slightly increases for networks bigger than the Jazz network itsself.

7 Discussion

To wrap up the results, the Kronecker graphs based on the Jazz network generally lead to more consistent results for all of the four error mechanisms. Also, the Jazz-Kronecker graphs overall reach higher robustness values, mostly around 0.9-0.95 for all centrality measures. The values mostly increase with network size. As the graph does not flatten out completely, the upwards trend is likely to continue for even larger networks.
The Dolphins-Kronecker graph generally deliver more varying results: Sometimes they do not converge to the same value for all error mechanisms, this is especially the case for the 10% and 30% error level of eigenvector centrality and the 30% error level of pagerank centrality. Also
Regarding the standard deviations, the values significantly and consistently decrease with increasing network size. That means that larger networks deliver more reliable results. As Kronecker graphs are a way
Although initial results look promising, there are several points that need to be addressed in order to solidify a thorough understanding of the task at hand. Particularly, in comparison to the Jazz network, we wish to look further into why the true and estimated

robustness in the Dolphins network is not as responsive to increasing network size.

Another area of interest to us is in potential time complexity improvements. One thing we have observed is that we ran 1000 iterations for each method, each network and with all centrality measures, it takes a lot of time to compute the results. We can use distributed computing concepts like MapReduce to speed these things up by computing a different set of iterations (100 in each mappers for example) in parallel time and take the mean values for all these iterations to get the final results.

In addition to that, there is potential in developing a method to find the optimal network size for a given underlying network. Namely, the "Kneedle" algorithm (Satopaa et al., 2011) could be implemented to find the knee-point in a plot of increasing network sizes. This knee-point would indicate the Kronecker network size that produces accurate robustness estimates.
Another focus for us would be to work on implementing Kronecker algorithm. While generating Kronecker graphs, we have used snap library to generate them but not in a programmatic way. As we faced issues initially regarding this, it would be good if we contribute towards popular open source libraries to have these algorithms available in python, so that other researchers wont find difficulties regarding this.
Lastly, upon further analysis of the aforementioned plots, different error mechanisms seem to have a significant impact on the robustness value but it is not the same case for Jazz. Also, as demonstrated in the plots, it would seem as if removing edges uniformly is the optimal error mechanism. Overall among centrality measures degree, pagerank, and eigenvector, robustness calculation using pagerank takes a longer amount of time compared to others. We have also observed that robustness calculation for Jazz networks takes longer time compared to Dolphins network. This is potentially due to the Jazz network size being far larger compared to Dolphins network.It is good to optimize robustness calculations in such a way that increasing network size shouldn't be a problem to get the robustness values.

8 Conclusion

Overall, the initial results of the study look promising, and a solid foundation has been laid for further analysis of Kronecker graphs. First, we calculated robustness of the original Jazz and Dolphins network. Secondly, we generated Kronecker graphs using the `kronfit` and `krongen` algorithms. Then, we applied error mechanisms to create erroneous Kronecker graphs and calcualated robustness using different centrality measures.
Conclusively, we analyzed robustness results and found out that error mechanisms play a vital role while calculating robustness of different network sizes. We have seen from the Jazz network that even though network size differs, robustness value converged at some

point. We can see from the research and results that there is a potential in researching to find the optimal number of network sizes.
Furthermore, we believe there's potential in further modifications and improvements, including a parallelized Kronecker graph generation framework, contributions to open-source Python packages, and programmatic-ally identifying optimal graph sizes.
Ultimately, and with the implementation of Kronecker theory, our results illustrate that the size of a generated Kronecker network seems to be related to the accuracy of the respective robustness measures of the underlying network. We believe that these outcomes pave the way for future work in the field of random graph generation.

References

Gleiser, P. M. and L. Danon (2003). Community Structure in Jazz. *Advances in Complex Systems* 6.4, pp. 565–573. arXiv: 0307434v2 [arXiv:cond-mat].

Kendall, M. G. (1945). The treatment of ties in ranking problems. *Biometrika* 33, pp. 239–251.

Landherr, A., B. Friedl, and J. Heidemann (2010). A critical review of centrality measures in social networks. *Business & Information Systems Engineering* 2.6, pp. 371–385.

Leskovec, J., D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani (2010). Kronecker Graphs: An Approach to Modeling Networks. *Journal of Machine Learning Research* 11, pp. 985–1042. arXiv: 0812.4905. URL: http://arxiv.org/abs/0812.4905.

Lusseau, D., K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson (2003). The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology* 54.4, pp. 396–405.

Martin, C. and P. Niemeyer (2019). Influence of measurement errors on networks: Estimating the robustness of centrality measures. *Network Science* 7.2, pp. 180–195. arXiv: 1704.01045.

Newman, M. (2010). *Networks - An Introduction*. New York: Oxford University Press.

Ravasz, E. and A.-L. Barabasi (2003). Hierarchical organization in complex networks. *Physical Review E*.

Satopaa, V., J. Albrecht, D. Irwin, and B. Raghavan (2011). Finding a Kneedle in a Haystack: Detecting Knee Points in System Behavior. *31st International Conference on Distributed Computing Systems*.

Wang, D. J., X. Shi, D. A. McFarland, and J. Leskovec (2012). Measurement error in network data: A re-classification. *Social Networks* 34.4, pp. 396–409. URL: http://dx.doi.org/10.1016/j.socnet.2012.01.003.