BACHELOR

Efficiently generating Geometric Inhomogeneous Random Graphs to test the influence of parameters on Degree-Degree Correlation in Geometric Graphs

Eltink, Gijs A.W.

Award date:
2023

Eindhoven University of Technology
Department of Applied Mathematics

Efficiently generating Geometric Inhomogeneous Random Graphs to test the influence of parameters on Degree-Degree Correlation in Geometric Graphs

2WH40 - Bachelor Final Project

Gijs Eltink 1452924

Supervisors:
Dr. W.L.F (Pim) van der Hoorn

Bachelor Thesis

Eindhoven, Saturday 20th May, 2023

Contents

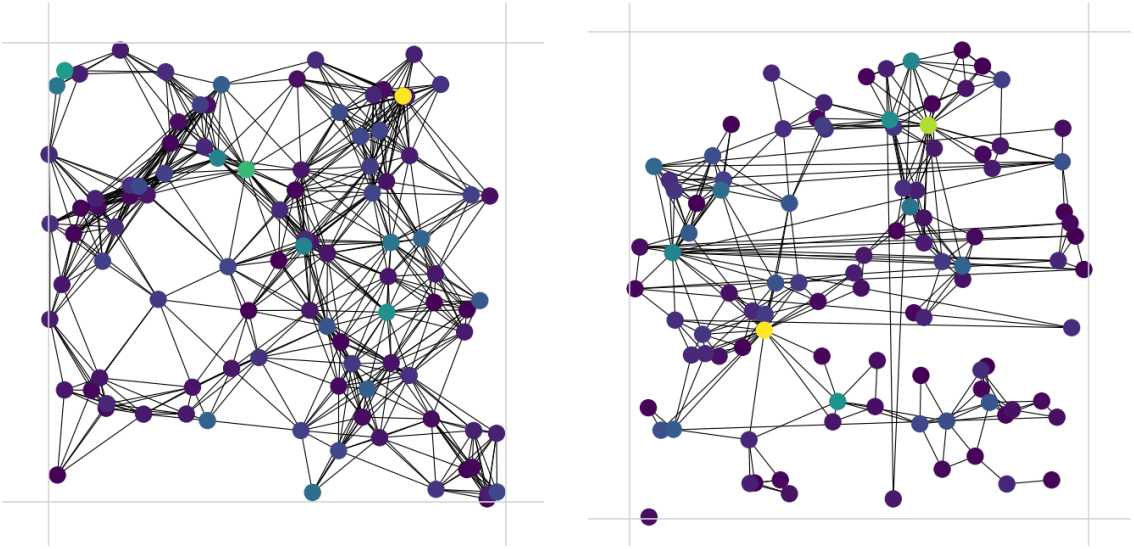
Chapter 1

Introduction

Complex networks are becoming more and more prevalent in today's society, as many real-world phenomena can be modeled by complex networks. Whether it is to study the internet (Broder et al., 2000), social networks (Buford et al., 2009), or even the phenomena that celebrities marry other celebrities (Barabási & Pósfai, 2016), everywhere networks appear. These networks can be very simple or majorly complex with millions of nodes. To determine how to study these networks, the structure of these networks can be exploited. For complex systems, the methods to determine these structures are not always simple however. In mathematical models, a network, or a graph, is a collection of vertices connected through edges in some ground space, with these vertices and edges assigned according to some stochastic mechanism. Some of these complex networks are influenced by a geometric feature, in which the distance of vertices in the ground space has an effect of the formed edges, a prime example of this is the small world phenomenon (Buford et al., 2009). By analyzing these geometric networks using some mathematical measure, more can be known about its complex structure.

We will investigate one of these measures, namely degree-degree correlations. In a network, the degree of a node refers to the number of connections or links it has with other nodes. Degree-degree correlation examines the association between the degrees of connected nodes (Barabási & Pósfai, 2016). It helps to determine whether nodes with similar or different degrees tend to connect to each other more often than expected by chance.

This correlation is a way to describe the structure of a network without having to view the intricate details of the network such as triangles, a phenomenon where three vertices are all connected to one another. To investigate the effects of geometric aspects on the structure of a network this thesis looks to study the influence of generative parameters on the Degree-Degree Correlation in geometric networks. This thesis will discuss d-dimensional Geometric Inhomogeneous Random Graphs (GIRGs). GIRGs generate networks by placing nodes randomly in a d-dimensional metric space and connecting them according to a distance-dependent probability function. This probability function also accounts for the 'weight' of nodes, for which a weight sequence is given. A common interpretation is that the weight of a node represents its capability to connect with other nodes in the network (Bringmann et al., 2015). In this case, a higher weight indicates a greater likelihood for a node to establish connections. The influence of this dimension d and distribution of the sequence of weights, sampled from some power-law distribution with exponent β (van der Hofstad & Litvak, 2012), have not been researched thoroughly, but are likely to influence the structure of these GIRGs. These GIRGs will be formalized in section 2.1. This thesis will thus explore the influence of these variables d and β on the structure of GIRGs, measured through Degree-Degree Correlation. The numerical experiments will be used to gather intuition into what theoretical analysis on the complex formulas could later prove. To begin this theoretical analysis the numerical implications shown in this thesis should help in finding a direction in creating proofs and hypotheses.



(a) GIRG on a Euclidean Space

(b) GIRG on a Torus

Figure 1.1: An example of two GIRGs with 100 nodes, on a Euclidean space and on a Torus. Here both the geometric- as well as the weight- (shown by the colour of the vertices) influence can be noticed.

GIRGs are a relatively new concept, discussed recently in research such as (Bringmann et al., 2015) and (Bläsius et al., 2019a). GIRGs were first mentioned in Bringmann et al., 2015 and were designed to model the behaviour of Hyperbolic Random Graphs (HRG), "GIRGs are technically simpler ... while preserving the qualitative behavior of hyperbolic random graphs" (Bringmann et al., 2015). Why this behaviour modeling of HRGs is useful will be elaborated in section 2.1.

The generation of these GIRGs is computationally slow, as every vertex pair will need to be sampled to calculate whether the vertices in question form an edge. As the number of vertices increases, the number of pairs of vertices increases quadratically. Luckily a study by Bringmann et al., 2015 introduced the theory behind a linear algorithm, capable of generating GIRGs. Later (Bläsius et al., 2019a) wrote an implementation of this algorithm in C++ and discussed more implementation details to show the practical usage of this algorithm. As this implementation is still very complex, and written fairly user-unfriendly, a new implementation of this linear algorithm is written in Python based on the existing code written in C++. This implementation will be thoroughly discussed in chapter 3.

On these "large" generated GIRGs multiple Degree-Degree Correlation measures, such as Pearson's Correlation Coefficient and Spearman's Rho (Van Der Hoorn & Litvak, 2015), will then be calculated. More details and theory about these measures will be given in section 2.2. To investigate the influence of the parameters, these numerically computed measures will then be compared for multiple values of d and β in chapter 4 to try and infer ideas for theoretical results for which intuitive reasoning will be given. This is done so that further research can be done to formalize theoretical results on the influence of these parameters on the Degree-Degree Correlation, and thus the structural analysis, of GIRGs.

Chapter 2

Theoretical Background

In this chapter we lay out the theory behind the model we use and the correlation measures we apply to it. We concretely explain what GIRGs are and how their structure differs from other networks. We then explain the concept of degree-degree correlation in a formal manner.

2.1 GIRGs

The model we shall use is a specific geometric weighted network, a Geometric Inhomogeneous Random Graph (GIRG), of which the structure is altered through its various parameters. These parameters give it the flexibility to closely represent network structures that can also be found in real-world complex networks such as social and technological networks (Lengler et al., 2016).

GIRGs are a concept first defined in Bringmann et al., 2015, and further discussed in Bläsius et al., 2019a as GIRGs approximate HRGs.

"The processes of generating a HRG and a GIRG can be coupled such that it suffices to decrease and increase the average degree of the GIRG by only a constant factor to obtain a subgraph and a supergraph of the corresponding HRG, respectively." (Bläsius et al., 2019a).

This qualitative behavior regards the capability of both models for generating complex networks that exhibit features such as scale-free degree distributions and clustering. The simplicity of GIRGs therefore became one of the major reasons why these GIRGs are a central research topic as of recent years (Bringmann et al., 2015).

Definition 1: GIRG
A Geometric Inhomogeneous Random Graph (GIRG) is a geometric network of n vertices, based on a data set (w_i, x_i) for $i \in \{1, \dots n\}$. Each vertex i is then assigned weight w_i and d-dimensional position x_i. Whether the vertices i and j are connected with an edge is independently determined using a probability p_{ij}, proportional to their respective weights, and inversely correlated to the distance between i and j.

In this paper we discuss GIRGs with data w_i sampled from a power law distribution W with exponent $\beta > 2$, and x_i sampled uniformly in $\mathbb{T}^d$, the $d-$dimensional torus with volume n. The exact probability p_{uv} is defined, derived from (Bringmann et al., 2015) and (Bläsius et al., 2019a), as follows:

$$p_{uv} = \min(1, c\left(\frac{w_u \cdot w_v}{||x_u - x_v||^d}\right)^\alpha). \tag{2.1}$$

Here the $||\cdot||$ norm is the distance norm in the d-dimensional torus, defined as

$$||x_u - x_v|| = \sqrt{\sum_{i=1}^{d} \min(|x_u^i - x_v^i|, 1 - |x_u^i - x_v^i|)^2}.$$

where x_u^i is the i'th position of the $d-$dimensional coordinate of x_u. Furthermore the constants $c \in (0, \infty)$ and $\alpha \in (1, \infty]$ are used to, respectively, alter the *average degree* and the *clustering coefficient*. The average degree of a network is a measure that displays to how many other nodes

a random node u is connected via an edge. The clustering coefficient measures the proportion of the number of triangles in a network including a node versus the possible triangles including that node.

Note that $\alpha = \infty$ is allowed, which leads to a threshold model, in which case the following relaxation of p_{uv} is sufficient:

$$p_{uv} = \begin{cases} 1, & \text{if} \quad ||x_u - x_v|| \leqslant (w_u \cdot w_v)^{\frac{1}{d}}, \\ 0, & \text{else.} \end{cases} \tag{2.2}$$

In this paper we will focus on this exact threshold model, and thus will be assuming $\alpha = \infty$. This leads to some simplifications in calculations and coding in the practical sections later. Further research could focus on the influence of α, and its implied clustering coefficient, in degree assortativity of GIRGs.

In the experiments, further on in this paper, the power law distribution W is specifically chosen to be a Pareto Distribution, with parameter β. This implies:

$$\mathbb{P}(W > t) = t^{-\beta}.$$

Furthermore W can be sampled with p.d.f. $f_\beta(x) = \frac{\beta}{x^{\beta+1}}$. Some more statistics for a randomly sampled W_i:

$$\mathbb{E}[W_i] = \begin{cases} \infty, \beta \leqslant 1, \\ \frac{\beta}{\beta-1}, \beta > 1. \end{cases} \quad , \quad Var(W_i) = \begin{cases} \infty, \beta \in (1, 2], \\ \left(\frac{1}{\beta-1}\right)^2 \cdot \frac{\beta}{\beta-2}, \beta > 2. \end{cases}$$

This choice was made to allow the possibility to easily replicate results in further research and later experiments, as the Pareto Distribution is a well-known and well-modeled distribution.

Notation:
If we write "Let G be a GIRG, with n vertices" then this implies $G = (V, E)$ is an undirected graph, where $|V| = n$, with all edges in E sampled using the above definition of a GIRG.

For $u, v \in V$ we write $u \sim v$ if u and v are adjacent.

2.2 Degree-Degree Correlation

Degree-degree correlation is a property of a network indicating the correlation between the degrees of pairs of connected nodes in a network. Specifically, it measures the extent to which nodes with similar degrees tend to be connected to each other, and nodes with different degrees tend to be connected to each other. If nodes with similar degrees tend to be connected, the network is said to have positive degree-degree correlation, whereas if nodes with different degrees tend to be connected, the network is said to have negative degree-degree correlation.

To calculate the degree-degree correlation of a network we need to determine some measures we shall use, and some preliminaries for how we apply these measures. These measures will then, for any given graph, determine a value in $[-1, 1]$, which indicates how positively or negatively degree-degree correlated the graph is.

As we want to test the correlation of a graph $G = (V, E)$, between the degrees on both sides of an edge it is helpful to look at all undirected edges $e \in E, e = (u, v)$ as a combination of two directed edges. We define for each directed edge a left (source) and right (target) node, let $\overline{e} = u$ and $\underline{e} = v$. We denote the set of directed edges by $E' \subseteq V \times V$, where E' is the set of directed edges derived from all undirected edges of E.

Note: *due to this addition of directed edges some measures and constants are scaled by a constant factor, as there are now twice as many edges in the graph.*

Furthermore we denote by

$$L(G) = \sum_{v \in V} d_v = |E'|,$$

the sum of degrees in the undirected graph, thus twice the number of undirected edges in G.

We then define the joint degree distribution as follows:

$$h_n(k,l) := \frac{1}{L(G)} \sum_{i \to j \in E'} \mathbb{1}_{\{d_{\underline{e}}=k, d_{\overline{e}}=l\}} \tag{2.3}$$

This is a distribution of the degrees on both sides of an edge, and therefore is a joint degree distribution. This joint distribution indicates how often nodes with degrees k and l are connected. This is clearly closely connected to degree-degree correlation, as we can quickly recognize from this distribution whether nodes with similar degrees are often connected, or whether nodes with greatly different degrees are connected many times.

Notice that if this distribution is dense in areas where k and l are similar, and sparse in areas where k and l differ largely, the degree-degree correlation is strongly positive. Similarly in the opposite scenario a strongly negative degree-degree correlation would be expected.

Measures
To quantify the degree-degree correlation of a given graph we will use two distinct measures: Pearson's Correlation Coefficient (often represented by an r) and Spearman's Rank Correlation Coefficient (also denoted as Spearman's Rho or Spearman's ρ). We shall be investigating these exact measures for the following reason; Pearson's r is a general measure for the correlation of data sets, but previous research noticed that Pearson's r relies on certain circumstances to converge to a proper limit, as this measure converges to zero when the degree distribution has infinite variance (van der Hofstad & Litvak, 2012),(Yao et al., 2017). We thus also include Spearman's Rho, which is shown to converge to a proper limit as the size of the network increases (van der Hofstad & Litvak, 2012).

We shall apply the known formula to determine the coefficient on the data set of tupled degrees of directed edges, for which we shall need some new notation. We thus denote for the required data set the following:

$$(d_{\underline{e}}, d_{\overline{e}}) \; \forall \; e \in E'$$

where we recall that $e = (\underline{e}, \overline{e})$ for any $e \in E'$.

2.2.1 Pearson's Correlation Coefficient

As stated before, Pearson's r is a general measure for correlation of data sets, in this case specifically jointly observed data. Its formula for determining r on sampled data $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ is well known to be the following:

$$r_{xy} = \frac{\sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \overline{y})^2}} \tag{2.4}$$

Here N is the sample size, x_i, y_i are the sampled points and $\overline{x}, \overline{y}$ are respectively the sample means of (x_i) and (y_i).

2.2.2 Pearson's r for Degree-Degree Correlation

To measure the degree-degree correlation in a network we can apply Pearson's Correlation Coefficient 2.4 to the data. For this we need to determine the data sets, and the other required constants. Notice the set of data we wish to find the correlation of consists of degree-tuples based on the edges in E', the set of directed edges related to E. Thus $N = |E'|$, $(x_i, y_i) = (d_{\underline{e_i}}, d_{\overline{e_i}})$, and $\overline{x} = \overline{d}_{\text{left}}, \overline{y} = \overline{d}_{\text{right}}$, the average degree of the nodes on the left and right size of an edge repectively.

We then find our first measure of degree-degree correlation on a graph $G = (V, E)$ to be the following:

$$r(G) := \frac{\sum_{e \in E'} \left(d_{\underline{e}} - \overline{d}_{\text{left}}\right)\left(d_{\overline{e}} - \overline{d}_{\text{right}}\right)}{\sqrt{\sum_{e \in E'} \left(d_{\underline{e}} - \overline{d}_{\text{left}}\right)^2} \sqrt{\sum_{e \in E'} \left(d_{\overline{e}} - \overline{d}_{\text{right}}\right)^2}} \tag{2.5}$$

Now that we have a way to measure the Degree-Degree Correlation for a network we want to find out what consistency the measure has, to ensure that we report only valid results. From (van der Hofstad & Litvak, 2012) we find the following theorem:

Theorem 1 (Adapted from Theorem 3.2(b) in van der Hofstad and Litvak, 2012)**.** *Let $(G_n)_{n\geq 1}$ be a sequence of random graphs of size n, where $G_n = (V_n, E_n), |V_n| = n$. Let (X_n, Y_n) be the degrees on both sides of a uniform directed edge $e \in E'_n$. Suppose that for every bounded continuous $h : \mathbb{R}^2 \to \mathbb{R}$,*

$$\mathbb{E}_n[h(X_n, Y_n)] \xrightarrow{\mathbb{P}} \mathbb{E}[h(X, Y)], \tag{2.6}$$

where the r.h.s. is non-random. Suppose $\mathbb{E}_n[X_n^2] \xrightarrow{\mathbb{P}} \mathbb{E}[X^2] < \infty$ and $\text{Var}(X) > 0$, then also

$$r(G_n) \xrightarrow{\mathbb{P}} r = \frac{\text{Cov}(X, Y)}{\text{Var}(X)} \tag{2.7}$$

Notice that this theorem requires the second moment of the random variable X, representing the limiting distribution of degrees on one side of a uniform directed edge, to be finite. Bringmann et al., 2015 shows the following theorem:

Theorem 2 (Adapted from Theorem 2.1 in Bringmann et al., 2016)**.** *With high probability the degree sequence of a GIRG follows a power law with exponent β. With high probability here implies with probability $1 - n$*

To give some intuition into why the degree sequence follows a power law, we highlight that the expected degree of a vertex v with weight w_v is by Lemma 4.3, it is not surprising that the degree sequence of the random graph will also follow a power law.

We note that for a power-law x^{-k} has a well-defined mean over $x \in [1, \infty)$ if $k > 2$, and it has a finite variance only if $k > 3$. As the degree sequence of a GIRG whp follows a power law with exponent β, we remark that the second moment of the degree sequence is infinite on $\beta < 3$.

Further note that for $\beta \in (1, 2)$ the mean of the degree distribution is ill-defined and the variance is infinite. This issue will be addressed later in subsection 2.2.3.

It is thus evident that the divergence of the second moment of degrees implies that Pearsons $r(G_n)$ does not necessarily converge to some r as n grows, and thus displays that $r(G_n)$ is inconsistent, for $\beta \in (2, 3)$. We therefore require a second measure of Degree-Degree Correlation.

2.2.3 Spearman's Rho for Degree-Degree Correlation

Spearman's rho considers the ranks of the degrees on the left and right side of an edge. For a sequence of values $x_1, x_2, \ldots, x_n$, we order them such that

$$x_{(1)} \geq x_{(2)} \geq \cdots \geq x_{(n)}.$$

For only distinct values we find strict inequalities, but since the degree of a node is a discrete measure, we need to differentiate between nodes with the same degree. We choose to solve this tie-breaking with uniformly sampled values, and since these are chosen in a continuous space [0,1], we find with probability zero for nodes to have equality for $d_v + U_v = d_w + U_v$ for all nodes v, w.

We now wish to apply Pearson's r as defined in 2.2.1 to this ranked data set, this value of r on the ranked data set is then the value of Spearman's ρ.

We define the ranked set of degrees through $\overline{\eta}$ and $\underline{\eta}$, where $\overline{\eta}(e)$ is the rank of $d_{\overline{e}}$ and $\underline{\eta}(e)$ the rank of $d_{\underline{e}}$.

We thus define:

$$\overline{\eta}(e) := \sum_{e' \in E'} \mathbb{1}_{\{d_{\overline{e}'} + U_{\overline{e}'} \} \geq d_{\overline{e}} + U_{\overline{e}}}$$

$$\underline{\eta}(e) := \sum_{e' \in E'} \mathbb{1}_{\{d_{\underline{e}'} + U_{\underline{e}'} \} \geq d_{\underline{e}} + U_{\underline{e}}}$$

We require the mean of these ranked sets when we apply Pearson's r to this data set, and that this value is equal to the sum of ranks divided by the number of ranks:

$$m := \frac{\sum_{i=1}^{|E'|} i}{|E'|} = \frac{|E'|+1}{2}.$$

We then find our second measure of degree-degree correlation on a graph $G = (V, E)$ to be the following:

$$\rho(G) := \frac{\sum_{e \in E'} \left(\underline{\eta}(e) - m\right)\left(\overline{\eta}(e) - m\right)}{\sqrt{\sum_{e \in E'} \left(\underline{\eta}(e) - m\right)^2} \sqrt{\sum_{e \in E'} \left(\overline{\eta}(e) - m\right)^2}} \tag{2.8}$$

We can rewrite this equation to the following compact form adapted from equation 2.6 from (van der Hofstad & Litvak, 2012):

$$\rho(G) = \frac{12 \sum_{e \in E'} \underline{\eta}(e)\overline{\eta}(e) - 3|E'|(|E'|+1)^2}{|E'|(|E'|^2 - 1)} \tag{2.9}$$

Now we can study the behaviour of this value for arbitrary n. We first note that $|E'| \approx n\mathbb{E}[d_u]$, for large n, where $\mathbb{E}[d_u] =: \overline{d}$ is the average degree of a uniform random node. Note further that $\overline{\eta}(e), \underline{\eta}(e) \approx n\overline{d}F^*(...)$ where $F^* : [R] \to [0, 1]$, which maps the degrees of a node to their ranking

as a fraction of the maximal degree. We can then *approximate* $\rho(G_n)$ in terms of n.

$$\begin{aligned}
\rho(G_n) &= \frac{12\sum_{e\in E'}\underline{\eta}(e)\overline{\eta}(e) - 3|E'|(|E'|+1)^2}{|E'|(|E'|^2-1)}, \\
&\approx 12\frac{\sum_{e\in E'}\underline{\eta}(e)\overline{\eta}(e)}{n\overline{d}((n\overline{d})^2-1)} - \frac{3n\overline{d}(n\overline{d}+1)^2}{n\overline{d}((n\overline{d})^2-1)}, \\
&\approx 12\frac{\sum_{e\in E'}(n\overline{d}\cdot F^*(d_{\underline{e}}+U_{\underline{e}}))(n\overline{d}\cdot F^*(d_{\overline{e}}+U_{\overline{e}}))}{n\overline{d}((n\overline{d})^2-1)} - 3, \\
&\approx \frac{12}{n}\frac{\sum_{e\in E'}F^*(d_{\underline{e}}+U_{\underline{e}})F^*(d_{\overline{e}}+U_{\overline{e}})}{\overline{d}} - 3, \\
&= \frac{12}{\overline{d}}\frac{1}{n}\sum_{e\in E'}F^*(d_{\underline{e}}+U_{\underline{e}})F^*(d_{\overline{e}}+U_{\overline{e}}) - 3, \\
&\xrightarrow{\mathbb{P}} 12\mathbb{E}[F_X^*(X^*)F_Y^*(Y^*)] - 3.
\end{aligned}$$

This intuition can be formalized using a theorem, once again from van der Hofstad and Litvak, 2012:

Theorem 3 (Adapted from Theorem 3.2(a) in van der Hofstad and Litvak, 2012)**.** *Let $(G_n)_{n\geq 1}$ be a sequence of random graphs of size n, where $G_n = (V_n, E_n), |V_n| = n$. Let (X_n, Y_n) be the degrees on both sides of a uniform directed edge $e \in E_n'$. Suppose that for every bounded continuous $h : \mathbb{R}^2 \to \mathbb{R}$,*

$$\mathbb{E}_n[h(X_n, Y_n)] \xrightarrow{\mathbb{P}} \mathbb{E}[h(X, Y)], \tag{2.10}$$

where the r.h.s. is non-random. Then

$$\rho(G_n) \xrightarrow{\mathbb{P}} 12\mathbb{E}[F_X^*(X^*)F_X^*(Y^*)] - 3 = \rho \tag{2.11}$$

We now note that this implies that $\rho(G_n)$ converges to some ρ, independent of the second moment of X. Thus Spearmans ρ is a consistent measure for degree-degree correlations in GIRGs generated with values of $\beta \in (2, 3)$.

We do not discuss another measure that is consistent at $\beta \in (1, 2)$, however further research could very well investigate the influences of other measures. Average Nearest Neighbour Degrees (Yao et al., 2017) could for example be a good measure to investigate, both on consistency and on the results after numerical experiments.

Chapter 3

Sampling GIRGs

To actually apply the measures we studied above to GIRGs we need to generate these GIRGs first. When we wish to generate GIRGs we find a difficult problem in the implementation, namely the speed at which these GIRGs can be generated. Generating GIRGs involves selecting a pair of nodes u, v, determining p_{uv}, sampling this probability to determine whether an edge should be generated and then constructing the resulting graph by sampling all pairs of nodes in the graph. As the number of vertices in the graph increases, the number of potential edges between vertices also increases rapidly. It is intuitive to see that this implies that for n nodes an algorithm would need to sample $O(n^2)$ edges. As the process of sampling a singular edge takes a non-negligible amount of time (see Figure 3.4, for 1000 nodes the algorithm takes longer than a second) a quadratic order algorithm takes considerable time to generate a large graph. For accurate results on graphs we want to sample multiple GIRGs with a large number of nodes, as we desire to investigate the asymptotic behaviour of these GIRGs. Due to this large number of graphs that need to be generated, of considerable size, the quadratic algorithm will not suffice in generating all these networks in acceptable time conditions and thus we require a more efficient way to generate these GIRGs for this thesis.

Luckily in 2015 Bringmann et al. published a paper theoretically proving that GIRGs can be generated in linear time. This paper did not involve any concrete program and left out many implementation details, but did showcase the structure of an algorithm that would be capable of sampling edges in linear time, with linear precomputation. Only the next year Bläsius et al., 2019b published a paper elaborating on their, slightly altered, implementation of the algorithm in C++ with more implementation details and explaining the algorithm in a more intuitive matter.

To investigate GIRGs and the influence of parameters on the value of the Degree-Degree Correlation in these GIRGs we will use a Python implementation based on Bläsius et al., 2019b. The code by Bläsius et al., 2019a, written in C++, can be found at the GitHub page of (Blasiüs et al., 2019). We wrote this new implementation to supply a usable algorithm for those that have little knowledge of C++ and wish to use a clearly annotated program to generate GIRGs.

We shall first introduce the structure of the algorithm on a broad level, after which we shall elaborate on the main features on a deeper level. This paper is not meant to explain all fine implementation details of the algorithm that were already discussed by Bläsius et al., 2019b, but it will lay out a general understanding of how the algorithm functions and some details that come with the implementation process.

3.1 Linear Algorithm

To understand why the algorithm works and to comprehend its efficiency a simple explanation of the algorithm is given first. We realise that nodes form an edge purely on their respective weights and distances. We further notice that this implies that we can divide nodes into weight buckets and grid cells, on which calculations can be made to figure out which weight buckets and grid cells need to be compared.

Let $B = \{B_0, B_1, \ldots, B_k\}$ be a set of *weight buckets*. A weight bucket B_i contains all nodes v with weights $w_v \in [w_{min} * a^i \ , \ w_{min} * a^{i+1})$, where a is constant. Thus each vertex of V is contained

in a weight bucket, of which the maximum and minimum weight are at most a constant factor a apart. Let k such that B_k contains the vertex with the largest weight in the sample of W. The algorithm described in this section will now use these weight buckets to figure out which nodes can be connected. This will be done by comparing the nodes of a certain weight bucket B_i to some weight bucket B_j. We shall now highlight how for two weight buckets B_i, B_j this comparison is done.

For the heaviest node u in B_i with weight w_u, and v, heaviest node in B_j with weight w_v, we can determine how far u and v can be from each other before their distance is too far to connect. This distance can be computed using the formula of p_{uv} 2.2. We call this distance $T(i, j)$.

Note that all nodes in B_i, when compared with any node in B_j, have a similar threshold distance whether the nodes are connected. Note that this threshold distance is always less or equal $T(i, j)$, as the weights of two vertices in B_i and B_j are less or equal than w_u, w_v respectively. Thus we find that when comparing two weight buckets we can easily compute a threshold distance $T(i, j)$.

This threshold distance $T(i, j)$ will then be used to partition the geometric space for a set of weight buckets B_i, B_j. We tile the geometric space into cells of which the side lengths are at least $T(i, j)$, then any $u \in B_i$ can only be connected to $v \in B_j$ if u and v are in *neighbouring* cells. If u and v are not in neighbouring cells, their distance is greater than $T(i, j)$ and the nodes are thus too far to connect.

A cell A *neighbours* cell B if B is directly adjacent or diagonally adjacent to A, or is A itself. A simple image to display an example in $d = 2$ for this can be seen below.

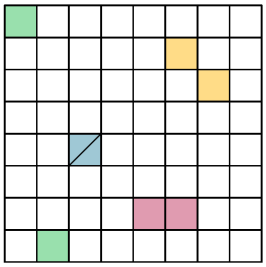


Figure 3.1: An example of a tiled $\mathbb{T}^2$. Each pair of colored cells represent neighboring cells. Note that the green cells are neighbours as the ground space is a torus, and these cells are thus diagonally adjacent to another. Adapted from (Bläsius et al., 2019a)

Now for the comparison of B_i and B_j, any node $u \in B_i$ in cell A, only needs to be compared to the nodes $v \in B_j$ that are in a neighbouring cell to A. This reduces the number of nodes we have to compare drastically, as in often many grid cells can be ignored in the comparisons. In the example above we notice that the nodes in a cell A only need to be compared to the nodes in the 9 neighbours of A, out of the total 64 grid cells, which gives a clear indication of the previous claim.

It can now be seen that if we compare each weight bucket B_i to all weight buckets B_j, we compare all nodes that are able to form an edge, and neglect only comparisons that would not form an edge anyways.

The explanation above is still fairly vague, and would not suffice in an attempt to implement the algorithm into a program without additional details. Furthermore while it has been demonstrated that many comparisons can be neglected, and that the algorithm is thus faster, it is not yet clear whether the time taken for the precomputations actually improves the algorithms' overall efficiency. For that reason these following subsections will formalize each of the largest parts of

the precomputations in more detail, and display how the algorithm is ran.

3.1.1 Weight Buckets and Cells

To create an efficient algorithm we need to set up two properties. Property I will ensure the correctness of the algorithm. Property II is then set up to reduce the number of unnecessary edges sampled.

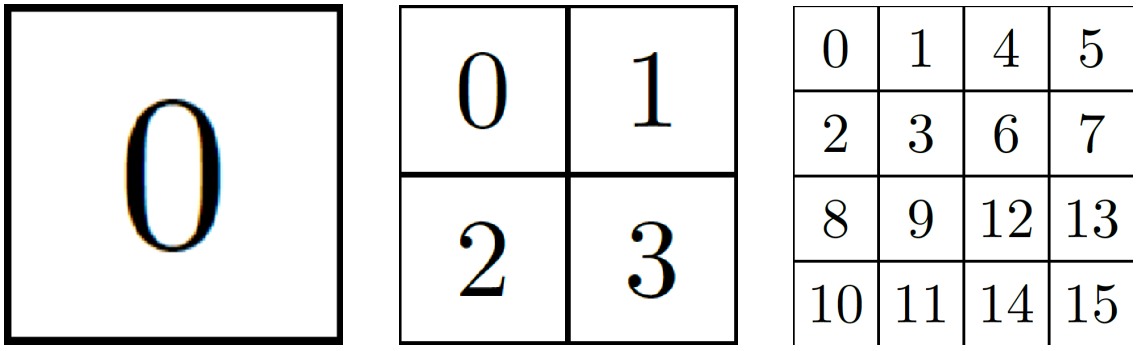
Property I: For any two weight buckets B_i, B_j, with heaviest nodes u, v respectively, the edge length of cells used to compare these weight buckets is at least equal to $T(i,j) = c(w_u \cdot w_v)^{\frac{1}{d}}$.

Property II: For any two weight buckets B_i, B_j, the edge length of cells used to compare these weight buckets is as small as possible.

As introduced before each node will be assigned to a singular weight bucket. We decide to split these weight buckets by factors of $a = 2$, implying that the weight of the heaviest node is at most a factor of two greater than the lightest node in the weight bucket.

To compare two weight buckets B_i and B_j a grid of cells is used, whose granularity is based on $T(i,j)$. To determine this granularity, the torus in which all nodes lie is divided into cells recursively, so that all side lengths of the cells are halved at each level. The grid thus starts off as a hypercube of volume n, with sidelengths $n^{\frac{1}{d}}$, we call this level 0.

The l'th level of granularity is then a set of 2^d times as many hypercubes as the level above. Thus each hypercube at level $l-1$ is filled exactly with 2^d hypercubes at level l. The volume of these hypercubes is then $\frac{n}{2^{d \cdot l}}$, where the hypercubes have side length $n^{\frac{1}{d}} \cdot 2^{-l}$.



(a) Level 0 (b) Level 1 (c) Level 2

Figure 3.2: An example of how a 2 dimensional space is granulated. Each figure shows the entire ground space of volume n. The side lengths of each cell in the figures is thus, $\sqrt{n}$ for level 0, $\frac{\sqrt{n}}{2}$ for level 1 and $\frac{\sqrt{n}}{4}$ for level 2.

For each node v, we assign it to a cell based on its weight bucket and its position x_v. This introduces the concept of the comparison level, denoted as $CL(i,j)$, of two weight buckets B_i and B_j. The comparison level represents the deepest level at which the side length of a grid cell, exceeds $T(i,j)$.

$$CL(i,j) := \max\{l \geq 0 : T(i,j) \leq n^{\frac{1}{d}} \cdot 2^{-l}\}. \tag{3.1}$$

Consequently all nodes in bucket B_i are *inserted* at the deepest level among their respective comparison levels. We define this level as the insertion level of bucket i, denoted as I(i).

$$I(i) = \max_{j \geq 0} CL(i,j) \tag{3.2}$$

This notion of *inserting* a vertex at a certain level will be used in subsection 3.1.3 to efficiently access all vertices for comparison. Note that a vertex has a fixed position in the ground space, which is thus not affected by this insertion. A node u is contained in cell A if $x_u \in A$. We denote the set of nodes of weight bucket B_i for which their position is in A by V_i^A.

By comparing all vertices of V_i^A to V_j^B, for all neighbouring cell pairs A, B at level $CL(i,j)$, Property I is ensured while simultaneously adhering to Property II to some degree.

The linear algorithm now iterates through all combinations of weight buckets B_i, B_j and determines for all nodes in V_i^A and all nodes in V_j^B whether an edge can be formed, for each neighbouring cell pair A, B at level $CL(i,j)$. This can be done recursively, for which the algorithm can be found in subsection 3.1.2.

3.1.2 The Algorithm

We now wish to lay out the full formal structure of the algorithm used to sample the GIRGs. As we now have all building blocks for the algorithm we shortly recap what we know.

All nodes have been sorted into weight buckets, and cells at a specific level. Furthermore Property I is not violated and thus the correct result will follow if the previously laid out comparisons are conducted.

Algorithm 1 Sample Edges by Recursive Iteration of Cell Pairs

Input: cell pair (A, B), level l
1: **for all** *bucket pairs* (B_i, B_j) **do**
2: **if** $CL(i,j) =$ level of A **then**
3: **if** *A and B are neighbors* **then**
4: emit each edge $(u,v) \in V_i^A \times V_j^B$ with probability p_{uv}
5: **if** *A and B are neighbors* **and** *level* $< \max_{i \le k}(I(i))$ **then**
6: **for all** *children X of A* **do**
7: **for all** *children Y of B* **do**
8: Run Algorithm 1 with input (X,Y) at level $l+1$

Algorithm 1 now shows pseudo code to demonstrate how the linear algorithm samples all edges in a threshold model GIRG. Notice the recursive structure of this algorithm. This structure ensures that all neighbouring cell pairs will be visited for any combination of weight buckets i and j. Thus by the result of Property I, every pair of nodes that can form an edge have been sampled, and the edges have been emitted. This implies that this algorithm samples exactly all edges of a GIRG generated from a set of given nodes with weights and positions.

The full implementation of the code to efficiently generate a GIRG will then be as follows:

Algorithm 2 Generate a GIRG

Input: n, β, d
Output: $G(V,E)$
1: Sample n values from W dependent on β
2: Sample n uniformly random coordinates in $[0, n^{\frac{1}{d}}]^d$ and create V
3: Insert vertices into weight buckets dependent on their weights
4: Determine all $CL(i,j), I(i)$ and sidelengths of cells at level $\le \max_{i \le k}(I(i))$
5: Generate all edges using Algorithm 1, with input $((0,0),0)$, save them in E

3.1.3 Efficiently Accessing Cell Pairs

While we now have a functioning algorithm, this relies on efficiently being able to access all nodes in V_i^A. To efficiently access those without having to test whether a node of B_i is in A we first precompute, in linear time, all these V_i^A for all $i \in 0, 1, \ldots, k$ and for all cells A at levels $l \in 0, 1, \ldots, \max_{i \leq k}(I(i))$.

When we consider a set V_i^A, we need to also be able to easily access the nodes that are in the children of A, one level deeper, as the algorithm relies on recursively iterating through the cell pairs at deepening levels. To do this we linearize the grid at every level, so that all children of a cell A appear consecutively in this ordering. To do this we use "Morton code" (Morton, 1966), also known as "z-order curve". An example of how this encoding works can be seen in Figure 3.3, and can then also be recognized inFigure 3.2.

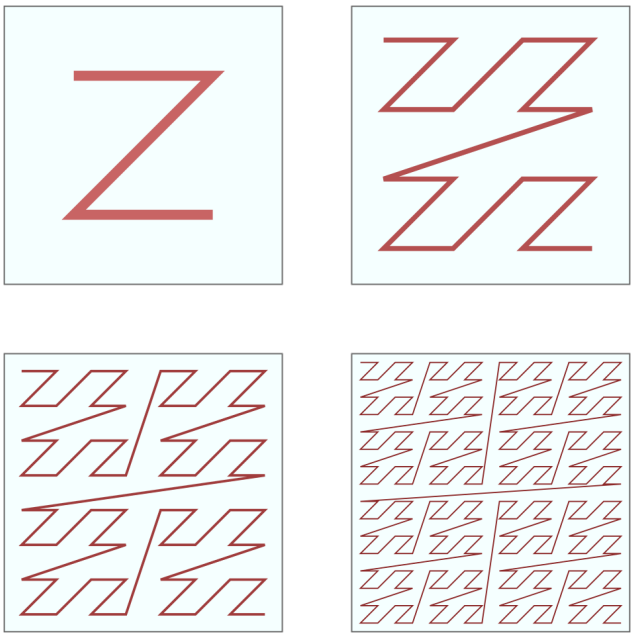


Figure 3.3: Four levels of Morton Code (Eppstein, 2008)

This encoding has exactly the property that for each cell at level l, its children at level $l' > l$ in the grid appear numbered consecutively. A nice extra property of Morton code is that it is computationally easy to convert between a cells position in the Morton (linear) order and its d-dimensional coordinates, by interleaving the binary representations of coordinate values. This conversion is already built into existing Python modules, and thus is practical for our Python implementation.

Now that the children of a cell appear in consecutive order, we can sort the nodes in a weight bucket i by the Morton code of the cell in which they were *inserted* (at level $I(i)$). This is also precisely the reason why nodes where inserted, as this leads to them being easily accessible whenever necessary. Two weight buckets will be compared at a specific grid level $CL(i, j)$, so it is important to be able to access all nodes of B_i in a specific cell at grid level $CL(i, j)$. Since the nodes of B_i were inserted at level $I(i) \geq CL(i, j) : \ \forall j \leq k$, the nodes of B_i are inserted at a level equal to or deeper then the required comparison level. If all nodes in the cell are now sorted consecutively, the nodes of V_i^A will be easily accessible.

The usage of the Morton encoding then leads to the following Lemma1:

Lemma 1. *[Adapted from Lemma 1 in (Bläsius et al., 2019a)] For any cell A at level $l' \leq I(i)$, the vertices of V_i^A appear consecutive.*

Lemma 1 will be used, as it implies that it suffices to know for each cell A the index of the first node in V_i^A. Using this index all nodes of A can easily be accessed, without having to view the stored data of any other node. Furthermore this index can easily be computed with a single loop through all nodes of i. This precomputation takes linear time to determine and leads to the following lemma.

Lemma 2 (Adapted from, and proven in Bläsius et al., 2019a)**.** *After linear preprocessing, for all cells A and weight buckets i with $level(A) \leq I(i)$, vertices in the set V_i^A can be enumerated in $O(|V_i^A|)$.*

3.1.4 Sanity Checks

It is essential to implement sanity checks into our code to ensure that our programs are working correctly and producing the expected outputs. Sanity checks are a set of tests that verify that the outputs of a program are within the expected range and make sense in the context of the problem at hand. They help catch errors early in the development cycle, preventing larger problems downstream. One way to implement sanity checks is by calculating statistical measures that can provide insight into the correctness of the results.

In our specific case, we use the average degree of the GIRG, dependent on c to determine whether the results of our program are correct. We use the basic quadratic order algorithm to generate small GIRGs, and test whether the results of our implementation of Algorithm 1 output the same results for: Average degree, Pearsons r and Spearmans ρ. If both algorithms receive the same input, and output the same results, we trust our implementation of Algorithm 1. We also show that our implementation of Pearsons r and Spearmans ρ is correct, this can be found in section 6.2.

We are now able to test the validity of the algorithm through the methods discussed before. We have run both algorithms for multiple different combinations of parameters β, d and c numerous times and with that verified the validity.

For the combinations found in Table 3.1 of values simulations were run, each for at least 10 runs at $n = 1000$, and 3 runs at $n = 10,000$.

We first generated n vertices using the combinations in Table 3.1, and then used these vertices as input for both algorithms. This method was used as both processes generate the nodes in the exact same way, to remove any influence of stochastic processes. This way the only part tested is the sampling of the edges given a set of vertices.
We found that for each combinations of parameters in Table 3.1, both algorithms output the same values for number of edges (thus also for average degree), r, and, within very fine margin, ρ. Further testing showed that these edges were the exact same, thus both algorithms generated the same networks for all inputs.

d	β	c
1	2.5	0.5
1	5	0.5
1	20	0.5
2	2.5	0.5
2	5	0.5
2	20	0.5
3	2.5	0.5
1	2.5	5
1	5	5
1	20	5
2	2.5	5
2	5	5
2	20	5
3	2.5	5
1	2.5	1
2	2.5	1
3	2.5	1

Table 3.1: List of combinations of parameters tested to verify validity of Algorithm 1

3.1.5 Speed test of Algorithm

While running the tests for subsection 3.1.4 we also generated numerous graphs of sizes $n = 1000$ and $n = 10,000$, and stored the time taken to generate these graphs. The figures Figure 3.4 and Figure 3.5 show how 200 generations of both algorithms for GIRGs of size $n = 1000$ and 10 generations of both algorithms for GIRGs of size $n = 10,000$ compare.

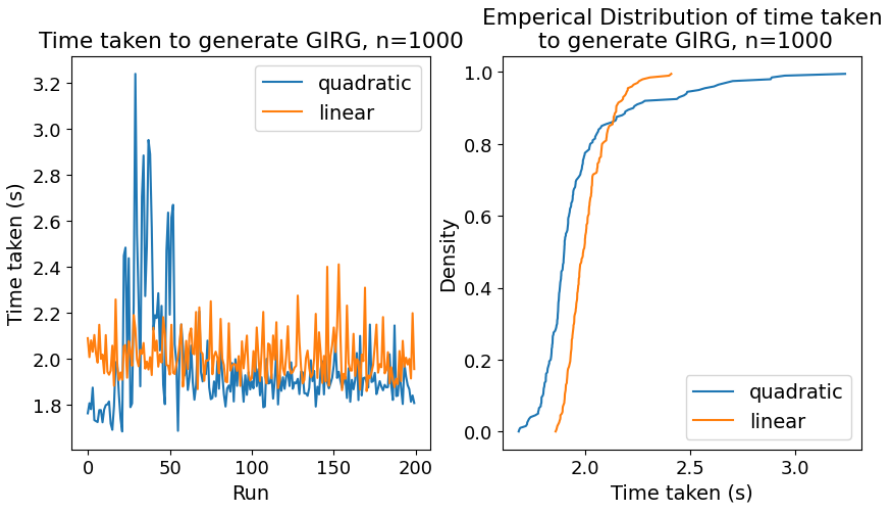


Figure 3.4: Comparison of computation speed of linear and quadratic algorithm for $n = 1000$, with 200 data points.

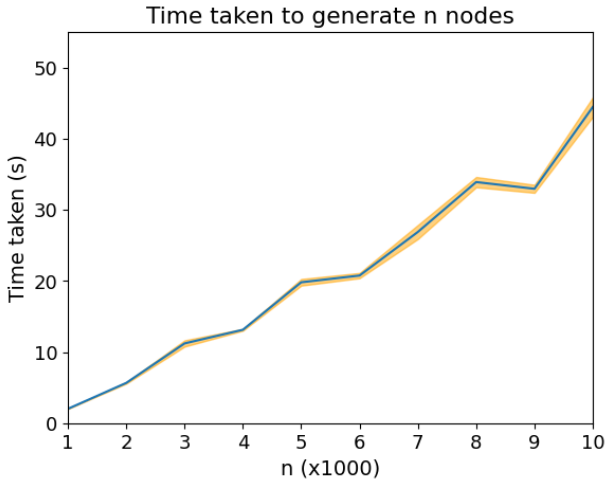


Figure 3.6: Comparison of mean computation speed of Algorithm 1 for all values of n between 1000 and 10,000, with steps of 1000, measured on at least 10 data points.

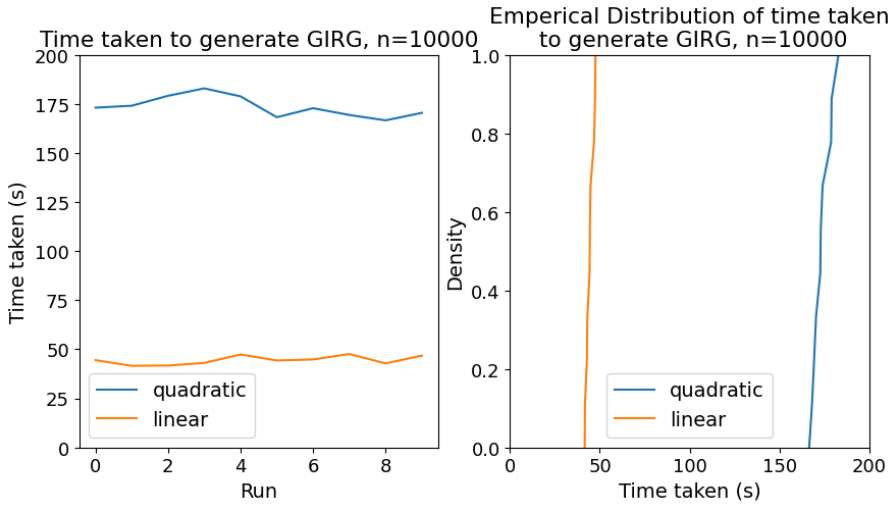


Figure 3.5: Comparison of computation speed of linear and quadratic algorithm for $n = 10,000$, with 10 data points.

We notice that in the case where the number of nodes is small (smaller than $n = 1000$ at least), the linear algorithm has no significant speed advantage over the quadratic algorithm. We notice however that as n grows, we see that the time taken by the quadratic algorithm grows fast, we notice an increase of approximately two orders of magnitude in time taken, while n increased by a single order of magnitude. We thus recognize that the quadratic algorithm indeed requires quadratic time to be ran. For the linear algorithm we notice that the time taken grows less rapidly, and indication of how fast this grows can be found in Figure 3.6. This behaviour appears linear, as was hoped. We note that in these comparisons the precomputation times required for the linear algorithm were included, so this gives an accurate representation of the time taken by the program, not just the time Algorithm 1 takes.

Chapter 4

Numerical Experiments

To discover the effects of the parameters β and d on the Degree-Degree Correlation of a GIRG, we can now generate GIRGs using our efficient program and generate results. We shall investigate values of β and d where we expect to find a change in structure of GIRGs, and thus of the values of the Degree-Degree Correlation measures. As we have shown in subsection 2.2.2, we wish to test at least for values of $\beta \in (2,3)$ and $\beta \in (3,\infty)$. We choose not to discuss $\beta \in (1,2)$, as resulting values for r and ρ for these experiments are not expected to be consistent, and thus numerical testing with large graphs will not be executed. We thus choose to handle the following cases for β: $\beta \in \{2.5, 3.5, 4.5\}$. Note that we expect for $\beta = 2.5$, that r converges to zero, while we do not know yet what results to expect for all other simulations. Furthermore we choose to include $\beta = 4.5$ to be able to recognize how Pearsons r is influenced by an increase in β on a domain where r is consistent.

For values of d we expect the biggest change in structure to happen between the shift from $d = 1$ to $d = 2$ due to the differening behaviour and existence of giant components in such graphs. We expect less change when moving to higher dimensions, though this is an interesting topic for further research.

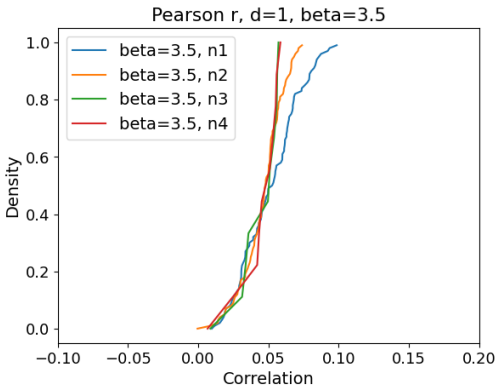
Furthermore we want to find out whether changing the value of n influences the mean and/or variance of the measures over multiple runs. We will model this by testing all $n \in \{10^4, 10^{4\frac{1}{3}}, 10^{4\frac{2}{3}}, 10^5\}$. As this shows we have to run many simulations we will keep the number of runs limited, but will test on their *confidence intervals* to allow ourselves certainty that our results remain valid. For $n = 10^4$ and $n = 10^{4\frac{1}{3}}$, we shall run the simulation 100 times, while for $n = 10^{4\frac{2}{3}}$ and $n = 10^5$ we shall work with 10 runs, to keep the time taken by the simulations feasible for the scope of this paper.

The confidence intervals were generated using a value of $\alpha = 0.025$, on a t-distribution, with the number of runs minus one as degrees of freedom. This led to a 95% confidence interval of a sample. We used $t_{df=99,\alpha=0.025} = 1.9842$ for the tests with 100 runs, and $t_{df=9,\alpha=0.025} = 2.2622$ for the experiments with 10 runs.

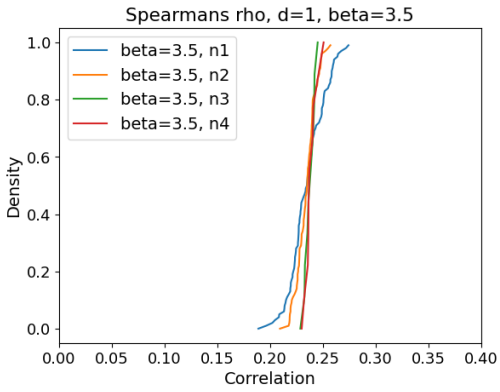
The influence of n and β will be handled in separate sections, the influence of the value of d in these scenarios will be discussed there.

4.1 Influence of n

As discussed, we will investigate the influence of different values of $n \in \{10^4, 10^{4\frac{1}{3}}, 10^{4\frac{2}{3}}, 10^5\}$ on the measures of Degree-Degree Correlation of GIRGs with n nodes. For this we fix a value of β and d, and generate a number of GIRGs of size n, and analyze the results. When fixing $\beta = 3.5$ and $d = 1$ the results in Figure 4.1a and Figure 4.1b arise.



(a) Cumulative distribution of Pearsons r for $n \in \{10^4, 10^{4\frac{1}{3}}, 10^{4\frac{2}{3}}, 10^5\}$. $\beta = 3.5$, $d = 1$



(b) Cumulative distribution of Spearmans ρ for $n \in \{10^4, 10^{4\frac{1}{3}}, 10^{4\frac{2}{3}}, 10^5\}$. $\beta = 3.5$, $d = 1$

Figure 4.1

We notice that as the value of n increases, that the cumulative distribution seems to stay centered around the same empirical mean, for both Pearsons r and Spearmans ρ. Furthermore, it seems that GIRGs with a larger number of nodes have less variance in their correlation than GIRGs of smaller size. As these figures are generated for a very limited number of runs, we plot the empirical mean of these simulations, and determine how much confidence we have that this empirical mean is indeed this value for GIRGs of size n.

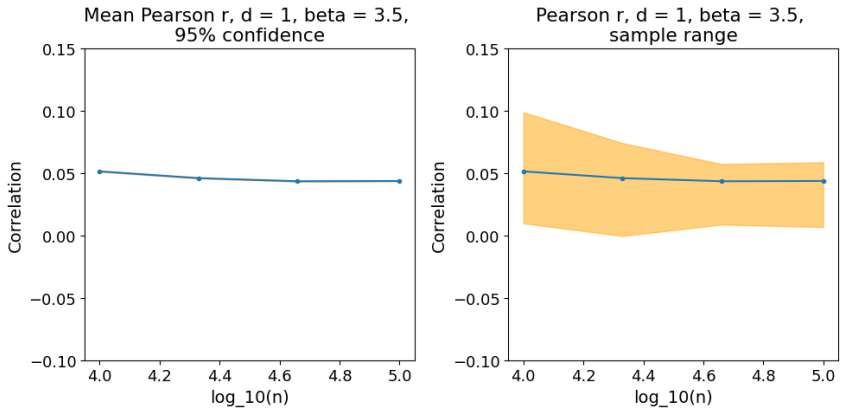


Figure 4.2: Mean values of Pearsons r depending on the size n of the GIRG. $d = 1, \beta = 3.5$. Figure (a) shows a 95% confidence interval, figure (b) highlights the area containing all recorded values

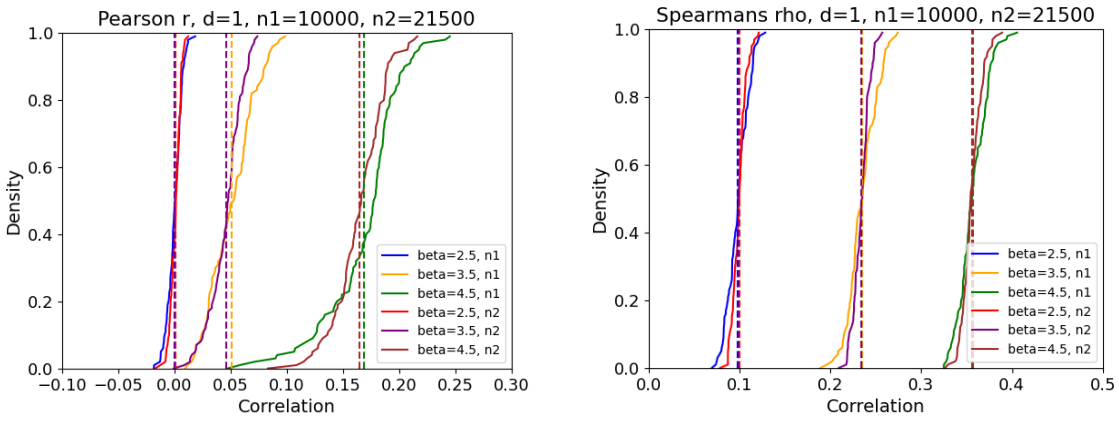
From Figure 4.2 it becomes apparent that, with high certainty, the value of n has little influence on the correlation of the GIRG. This result seems to show that the, from literature expected and shown in subsection 2.2.2, convergence occurs at fairly small networks already, as for $n = 10^4$ this value seems to already have basically stabilized. It is important to note the scale on the y-axis of these plots, as the difference between simulations might seem large, but this is due to the small steps on the y-axis.

Figure 4.2 only shows Pearsons r, the same figure for Spearmans ρ can be found in the appendices at Figure 6.2, and gives similar results. Furthermore results for $d = 2$ can also be found in the appendices in section 6.3, as we find these to have the same conclusions as for $d = 1$.

4.2 Influence of β

Now that we have shown that large increases in n only alter the variance of the Degree-Degree Correlation measures on GIRGs, we shall now investigate the influence of β on these measures. As β influences the weight distribution for the sampled nodes, we expect to see larger differences in the values measured than we saw previously with n.

We fix $d = 1$ and $n \in \{10^4, 10^{\frac{13}{3}}\}$, and will generate 100 GIRGs, for which we will plot the empirical distribution of the correlation, and indicate the mean for each run.



(a) Pearsons r

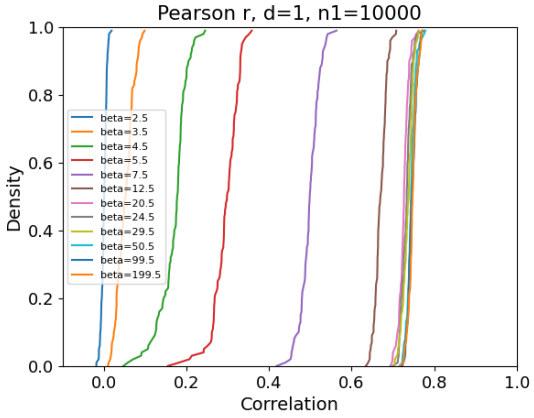
(b) Spearmans ρ

Figure 4.3: Cumulative distributions of Pearsons r and Spearmans ρ of different GIRGs with $\beta \in \{2.5, 3.5, 4.5\}$, $d = 1$ and $n \in \{10^4, 10^{\frac{13}{3}}\}$. The vertical dashed lines represent the mean values of the indicated data.

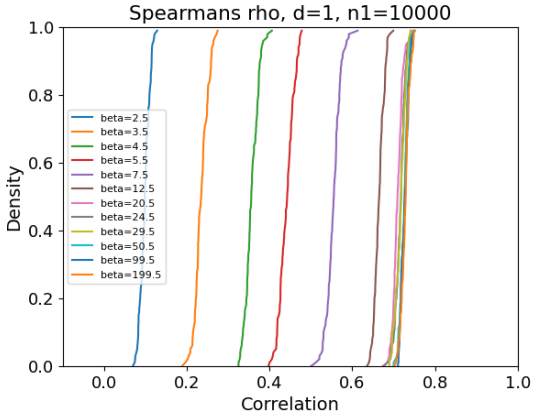
From Figure 4.3a and Figure 4.3b one can see that for these values of β, generally positive correlations arise, where a trend occurs: as β increases, the Degree-Degree Correlations of the GIRG increases as well. We notice this happens for the entire cumulative distribution, as well as the empirical mean. As a side note we once again notice the same results as found in section 4.1, that for higher values of n the variance around the mean decreases.

Due to the observation that the correlation increases, more values, namely higher, of β will be simulated, to try and figure out whether a pattern emerges. For this some values up to $\beta = 200$ will be tested, to see whether the weight distributions shape influences the Degree-Degree Correlations consistently.

From Figure 4.4a and Figure 4.4b one can now clearly see that for increasing values of β, the Degree-Degree Correlations increase too. It seems as though the correlations both converge to some positive maximum. The mean recorded value of the maxima, for the simulations with the highest β, at $\beta = 199.5$ are $\overline{r} = 0.74752$ and $\overline{\rho} = 0.72808$. In the case of $d = 1$ and $n = 10000$. This convergence seems to happen decreasingly, so a difference of 1 in small β values has more influence than for larger β values. We can investigate this behaviour by plotting the mean value of the correlation directly against the value of β. We do this in Figure 4.5 and Figure 4.6, including the same approach to showing certainty in the simulations as in Figure 4.1.



(a) Cumulative distributions of Pearsons r of different GIRGs dependent on β, with $d = 1$ and $n \in \{10^4, 10^{\frac{13}{3}}\}$.



(b) Cumulative distributions of Spearmans ρ of different GIRGs dependent on β, with $d = 1$ and $n \in \{10^4, 10^{\frac{13}{3}}\}$.

Figure 4.4

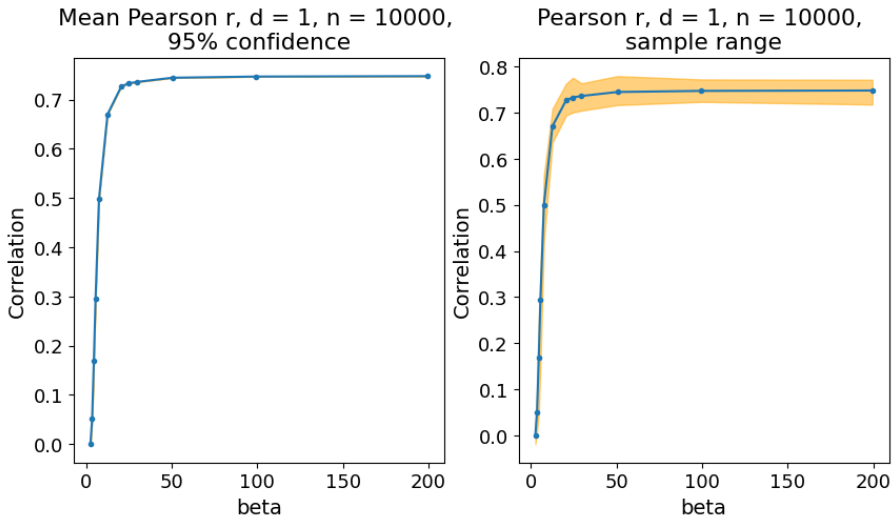


Figure 4.5: Mean values of Pearsons r depending on the value of β of the GIRG. $d = 1, n = 10000$. Figure (a) shows a 95% confidence interval, figure (b) highlights the area containing all recorded values

By analyzing Figure 4.5 and Figure 4.6 we indeed notice the structure we described before. It is visible that the Degree-Degree Correlations both approach a maximal value as β increases. We notice that these values seem very similar, so we delve further into the difference between Pearsons r and Spearmans ρ.

When investigating the left most plot of Figure 4.7, we notice that the approached values for Pearsons r and Spearmans ρ are not (exactly) the same. An interesting observation however is how the value for Pearsons r seems to outgrow Spearmans ρ around $\beta = 12$, although Spearmans ρ seems to increase faster than Pearsons r for smaller values of β. When we think about what this implies regarding the structure of the network, we might look at what difference the mapping to Spearmans ρ makes.

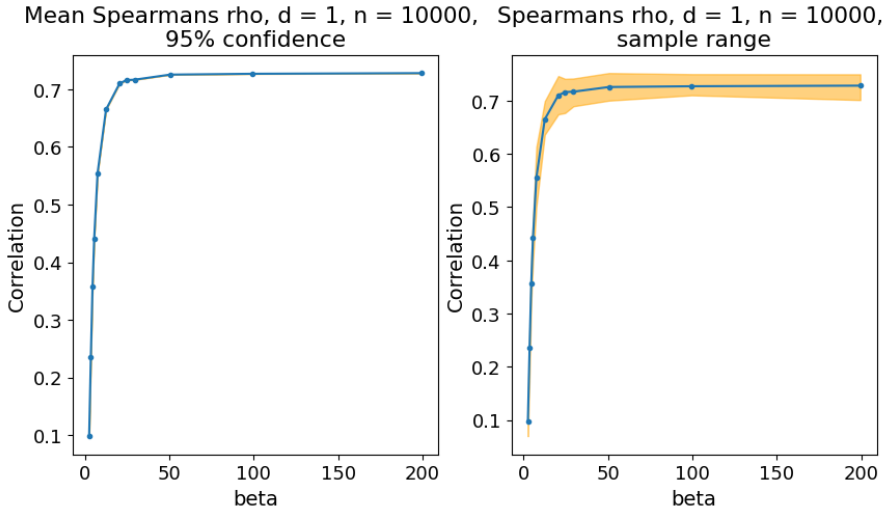


Figure 4.6: Mean values of Spearmans ρ depending on the value of β of the GIRG. $d = 1, n = 10000$. Figure (a) shows a 95% confidence interval, figure (b) highlights the area containing all recorded values

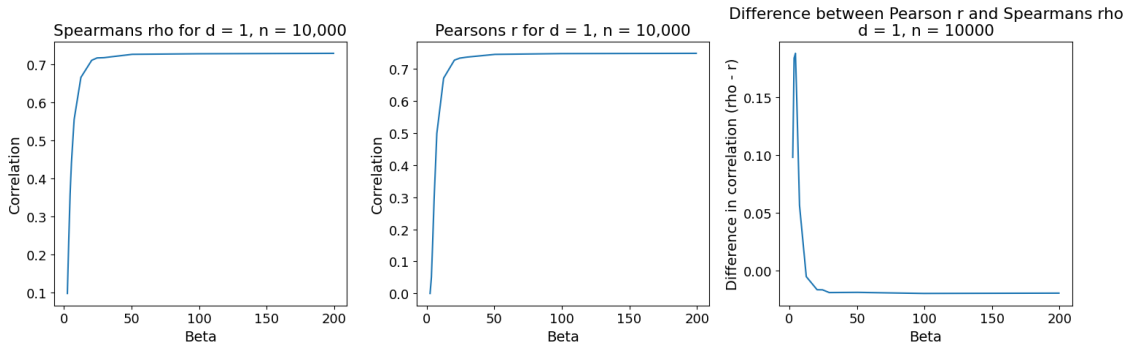


Figure 4.7: Difference between Pearsons r and Spearmans ρ, for increasing values of β

If we try to imagine why this value change occurs, we notice that in Figure 4.3 the larger variance for the plots at $\beta = 3.5$ for r than we see for ρ. These two arguments might be related, as both r and ρ are essentially a form of Pearsons Correlation. Spearmans ρ however, applies Pearsons r after a mapping, from degrees to rankings. When degrees have high variance, this mapping will reduce the variance in the data. However when the data set of degrees has low variance, the mapping to rankings increases the variance in the data. We notice that as β increases, the tail of the weight distribution slinks, and thus the variance in the weights of all nodes decreases. This might influence the variance and the relative value of r and ρ and would be an interesting point for further research, including the question at what point of β this change occurs, and if this is consistent among other parameter changes.

As we further investigate the data gathered by these simulations we discuss the behaviour of Pearsons r for values of $\beta < 3$. Here we notably compare Spearmans ρ with r at values of $\beta = 2.5$ and $\beta = 3.5$. We note that the positive correlation shown by ρ in Figure 4.8 for $\beta = 2.5$ indicates that the Degree-Degree Correlations in this setting are indeed positive. Comparing this to the value of r, we notice that the value of r is centralized around $r = 0$, which is the behaviour we expected from the inconsistent measure of Pearsons r for values of $\beta < 3$ as shown in subsection 2.2.2.

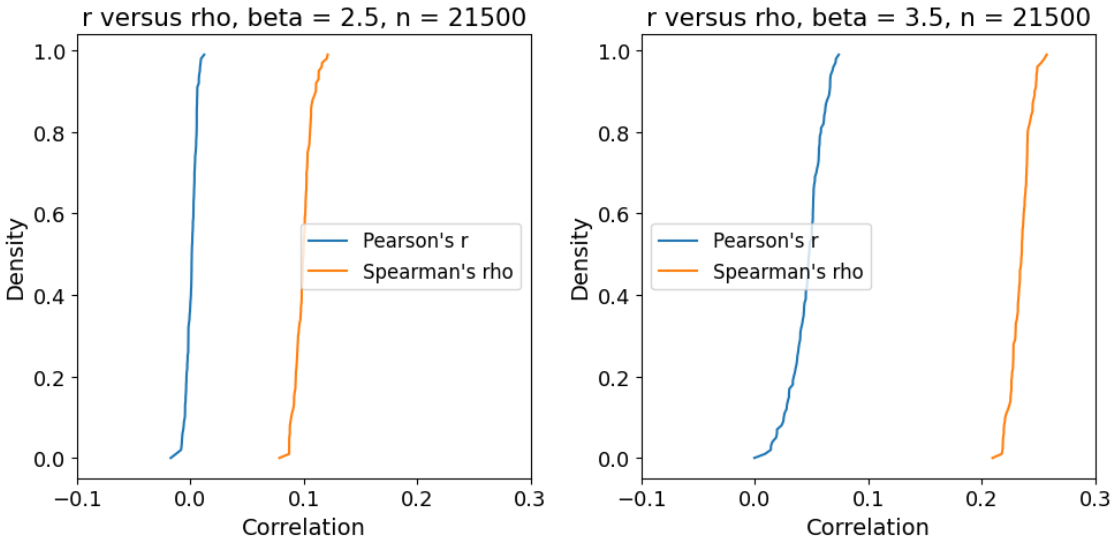
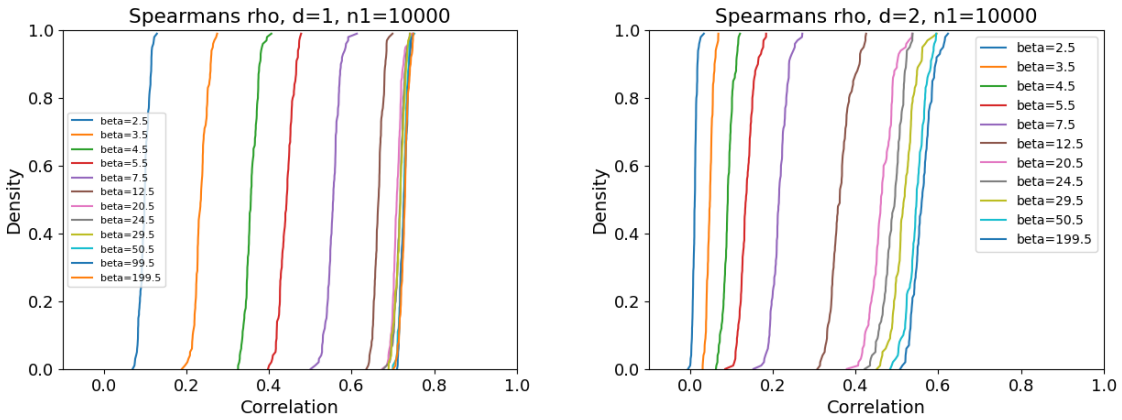


Figure 4.8: Values of r and ρ for small values of β

4.3 Influence of d

As mentioned before we would also like to discuss the influence of d on the Degree-Degree correlations of the network. As results of previous sections show we do not see any systematically different results for different values of d, Figure 4.9 and Figure 4.10 show this again.



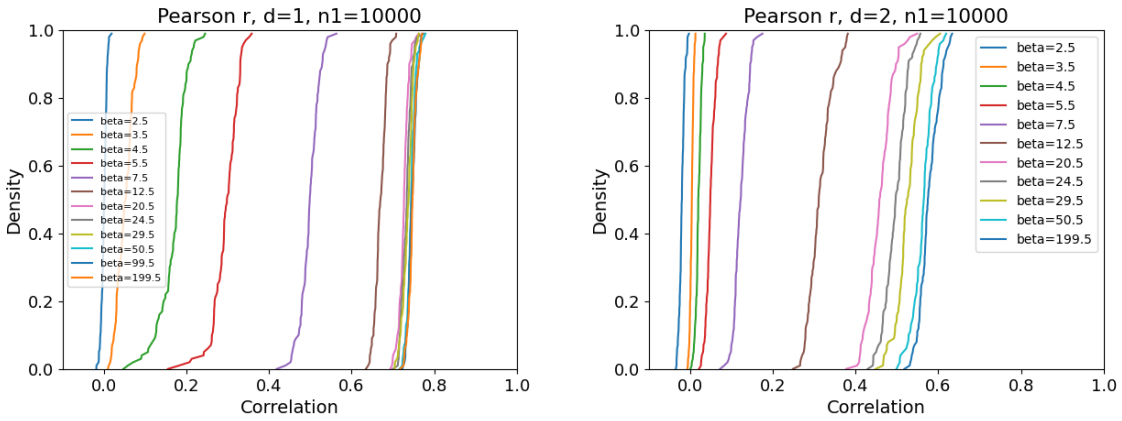
(a) Spearmans ρ for $d = 1$

(b) Spearmans ρ for $d = 2$

Figure 4.9: Values of Spearmans ρ for different values of β and d.

As seen in Figure 4.9 and Figure 4.10, the value of d seems to alter the value of the Degree-Degree Correlations. However the results follow similar patterns, as the correlations approach a maximal value for both values of d. We notice this maximal value seems consistently lower for both correlations for $d = 2$, and again seems to be very similar for both r and ρ, this time centering its values around 0.589 and 0.583 respectively.

When considering the structural implications of this finding we imagine this discrepancy occurs as nodes lie on average further away in a higher dimension, and thus vertices might have less common neighbours. This decrease in common neighbours would lead to a decrease in connections between



(a) Pearsons r for $d = 1$

(b) Pearsons r for $d = 2$

Figure 4.10: Values of Pearsons r for different values of β and d.

nodes with similar degree as these neighbours have high probability to have similar degrees. This would lead to lower degree-degree correlations.

Further research could include more research into this area, by for example testing whether higher values of d consistently lower the value of the correlation measures maxima. For these tests it would be interesting to investigate whether this increase of d leads to a minimal value of the maximal value of the degree correlations with respect to β, and finding out whether this minimum is positive, slightly negative, very negative or possibly zero.

Chapter 5

Conclusion

In the end we can conclude that the values of the parameters have clear influence on the value of the Degree-Degree Correlations in a threshold model GIRG. Furthermore we wrote a, user friendly, implementation of an efficient algorithm for generating GIRGs in Python, and tested its speed and reliability.

The implementation of the efficient algorithm in Python was found to act linearly, as theory proven in literature shows would be the case. The implementation was significantly faster for large values of n. We found that for $n = 1000$, the time taken per simulation was similar for the intuitive quadratic algorithm and the linear algorithm. For $n = 10000$ the different algorithms showed clear differences in speed, with a factor of approximately 3.5. Runs of the linear algorithm took less than 50 seconds, while the quadratic algorithm produced the same output after roughly 175 seconds. It was found that the implementation of the linear algorithm was correctly programmed, as the results were recorded for multiple different combinations of the parameters n, β, d and c, which all output the same results for the quadratic and linear algorithm.

To investigate the Degree-Degree Correlations we applied two measures of correlation, Pearsons Correlation Coefficient (r) and Spearmans Rank Correlation Coefficient (ρ). We recalled a proof that Pearsons r is not consistent for networks where the 3rd moment of the degrees is not finite. We then showed that for $\beta \in (2, 3)$, Pearsons r was not consistent, while Spearmans ρ was consistent. This consistency led to the usage of both measures for all other experiments so as to be able to compare them both. We noticed that for larger values of β the difference in values of r and ρ decreased until a certain point after which it increased to a small upper bound.

We noticed that the parameter of the weight distribution, β as exponent of the CDF of a Pareto Distribution, positively correlated with an increase in Degree-Degree Correlations. We might expect this to be due to higher values of β leading to a thinner tail in the power law distribution, thus nodes will have similar weights. Nodes with similar weights, in a not fully connected network, will only be connected with nearby nodes, and thus these nodes will cluster with their common neighbours. This possibly leads to "clouds" of nodes forming, of which the nodes have similar degrees, and thus positively influence the Degree-Degree Correlations. Furthermore we notice that an increase in β seems to lead to a maximal value of these Degree-Degree Correlations, less than one, implying that a GIRG can have high positive Degree-Degree Correlations, but does not quickly form a graph of only complete components, as these are the only graphs with Degree-Degree Correlation 1.

We tested the influence of the size $|V|$ of a GIRG G with $G = (V, E)$, and found that $n := |V|$ did not seems to alter the mean value of the Degree-Degree Correlations in GIRGs generated with the same parameters β and d. We did notice that the variance of the Degree-Degree Correlation for sampled GIRGs decreased as $n \to \infty$.

As this thesis was focused on exploring the systematic changes in Degree-Degree Correlations in GIRGs upon the altering of generation parameters of the network few rigorous proofs were conducted. A more rigor mathematical analysis can now be conducted in further research to formalize our findings with regard to the increase of n and β in the values of the Degree-Degree Correlations. Furthermore the influence of the dimension of a network were only explored slightly, it would be interesting to find out how the further increase in dimension would alter the Degree-Degree

Correlations, and whether these would become negative for some combination of parameters.

We wrote a user friendly implementation of the linear algorithm in Python, which is a relatively easy programming language to learn, but has the downside of being computationally rather slow. The existing implementation of this algorithm in C++ used by (Bläsius et al., 2019a) was significantly faster, but harder to comprehend for a non-experienced user of C++. Due to the computational speed of Python this thesis was limited in its research towards the influence of large values of n, and as many different combinations of parameters had to be attempted the number of runs for each combination was kept low. Even though the results were found to have high confidence, an increased number of runs for these simulations would be beneficial to reduce random bias in the results of the simulations.

For further research the influence of clustering coefficient α can be researched. One could speculate about the results of the addition of this clustering coefficient, which would change the model from a threshold model to a binomial model where edges can be formed between nodes that lie farther apart. This addition of α into the model could influence the generation of these "clouds" mentioned before, as these clouds would likely become more interconnected. As this is only speculation, and sparks the mind already, further research delving deeper into this would be interesting.

Bibliography

Barabási, A.-L., & Pósfai, M. (2016). Degree correlations. In *Network science*. Cambride University Press. http://physicstoday.scitation.org/doi/10.1063/PT.3.3526

Bläsius, T., Friedrich, T., Katzmann, M., Meyer, U., Penschuck, M., & Weyand, C. (2019a). Efficiently Generating Geometric Inhomogeneous and Hyperbolic Random Graphs. http://arxiv.org/abs/1905.06706

Bläsius, T., Friedrich, T., Katzmann, M., Meyer, U., Penschuck, M., & Weyand, C. (2019b). Efficiently generating geometric inhomogeneous and hyperbolic random graphs. *Leibniz International Proceedings in Informatics, LIPIcs*, *144*. https://doi.org/10.4230/LIPIcs.ESA.2019.21

Blasiüs, T., Weyand, C., & Penschuck, M. (2019). GitHub - chistopher/girgs: Linear-time sampling algorithm for geometric inhomogeneous random graphs with a special case implementation for hyperbolic random graphs. https://github.com/chistopher/girgs

Bringmann, K., Keusch, R., & Lengler, J. (2015). Geometric inhomogeneous random graphs. *Theoretical Computer Science*, *760*, 35–54. https://doi.org/10.1016/j.tcs.2018.08.014

Bringmann, K., Keusch, R., & Lengler, J. (2016). Average Distance in a General Class of Scale-Free Networks with Underlying Geometry. http://arxiv.org/abs/1602.05712

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., & Wiener, J. (2000). Graph structure in the Web. *Computer Networks*, *33*(1-6), 309–320. https://doi.org/10.1016/S1389-1286(00)00083-9

Buford, J. F., Yu, H., & Lua, E. K. (2009). Unstructured Overlays. *P2P Networking and Applications*, 45–73. https://doi.org/10.1016/B978-0-12-374214-8.00003-9

Eppstein, D. (2008). Z-order curve. https://upload.wikimedia.org/wikipedia/commons/c/cd/Four-level_Z.svg

Lengler, J., Bringmann, K., Keusch, R., & Koch, C. (2016). *Geometric Inhomogeneous Random Graphs (GIRGs) joint work with K Motivation: Network Models* (tech. rep.).

Morton, G. M. (1966). A Computer Oriented Geodetic Data Base; and a New Technique in File Sequencing. *Morton1966*. https://books.google.nl/books?id=9FFdHAAACAAJ

Van Der Hoorn, P., & Litvak, N. (2015). Degree-degree dependencies in directed networks with heavy-tailed degrees. *Internet Mathematics*, *11*(2), 155–179. https://doi.org/10.1080/15427951.2014.927038

van der Hofstad, R., & Litvak, N. (2012). Degree-degree correlations in random graphs with heavy-tailed degrees. https://doi.org/10.1103/PhysRevE.87.022801

Yao, D., van der Hoorn, P., & Litvak, N. (2017). Average nearest neighbor degrees in scale-free networks. http://arxiv.org/abs/1704.05707

Chapter 6

Appendix

6.1 Calculation of $r(G)$

Notice $\overline{d}_{\text{right}} = \overline{d}_{\text{left}} = \frac{1}{|E'|} \sum_{e \in E'} d_{\underline{e}} = \frac{1}{12}(2+2+2+8+8+8+8+6+6+6+2+2) = \frac{60}{12} = 5.$

Edge	$(d_{\overline{e}}, d_{\underline{e}})$	$\left(d_{\underline{e}} - \overline{d}_{\text{left}}\right)\left(d_{\overline{e}} - \overline{d}_{\text{right}}\right)$
(1,4)	(2,8)	-9
(2,4)	(2,8)	-9
(3,4)	(2,8)	-9
(4,1)	(8,2)	-9
(4,2)	(8,2)	-9
(4,3)	(8,2)	-9
(4,5)	(8,6)	3
(5,4)	(6,8)	3
(5,6)	(6,2)	-3
(5,7)	(6,2)	-3
(6,5)	(2,6)	-3
(7,5)	(2,6)	-3

Table 6.1: Table used for the calculation of $r(G)$

$$\begin{aligned}
\sqrt{\sum_{e \in E'} \left(d_{\overline{e}} - \overline{d}_{\text{right}}\right)^2} &= \sqrt{\sum_{e \in E'} \left(d_{\overline{e}} - \overline{d}_{\text{left}}\right)^2}, \\
&= \sqrt{5 \cdot (2-5)^2 + 4 \cdot (8-5)^2 + 3 \cdot (6-5)^2}, \\
&= \sqrt{5 \cdot 9 + 4 \cdot 9 + 3 \cdot 1}, \\
&= \sqrt{84}.
\end{aligned}$$

$$\begin{aligned}
r(G) &= \frac{\sum_{e \in E'} \left(d_{\underline{e}} - \overline{d}_{\text{left}}\right)\left(d_{\overline{e}} - \overline{d}_{\text{right}}\right)}{\sqrt{\sum_{e \in E'} \left(d_{\underline{e}} - \overline{d}_{\text{left}}\right)^2} \sqrt{\sum_{e \in E'} \left(d_{\overline{e}} - \overline{d}_{\text{right}}\right)^2}}, \\
&= \frac{6 \cdot -9 + 2 \cdot 3 + 4 \cdot -3}{\sqrt{84} \cdot \sqrt{84}}, \\
&= \frac{-60}{84}, \\
&= -\frac{5}{7}.
\end{aligned}$$

6.2 Validity of Measures

We also need to test the validity of our implementation of the degree-degree correlation measures themselves, otherwise we can not trust the output used above. Therefore we first test these measures on small graphs of which the actual values of r and ρ can be calculated by hand. For this we use the following graph:

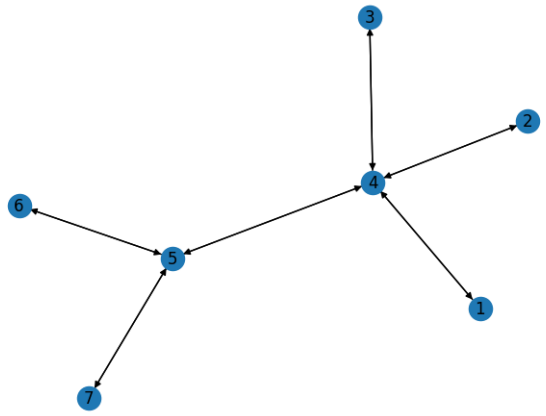


Figure 6.1: Graph used for testing measures

On this graph we determine the values of r and ρ, using Equation 2.5 and Equation 2.8.

$$\begin{aligned} r(G) &= \frac{\sum_{e \in E'} \left(d_{\underline{e}} - \overline{d}_{\text{left}}\right)\left(d_{\overline{e}} - \overline{d}_{\text{right}}\right)}{\sqrt{\sum_{e \in E'} \left(d_{\underline{e}} - \overline{d}_{\text{left}}\right)^2} \sqrt{\sum_{e \in E'} \left(d_{\overline{e}} - \overline{d}_{\text{right}}\right)^2}}, \\ &= -\frac{5}{7} \end{aligned}$$

Our Python implementation of this measure returns the value $-0.7142857142857144 \approx -\frac{5}{7}$. Thus we trust our implementation of this measure to be correct. For the exact calculation of $r(G)$, see section 6.1.

For the implementation of ρ we find that we rely on the realization of the uniform random variables used to break ties. Due to this we get some variability in the value of ρ. We notice that we can not avoid this problem as we choose to break ties at random.

6.3 Figures

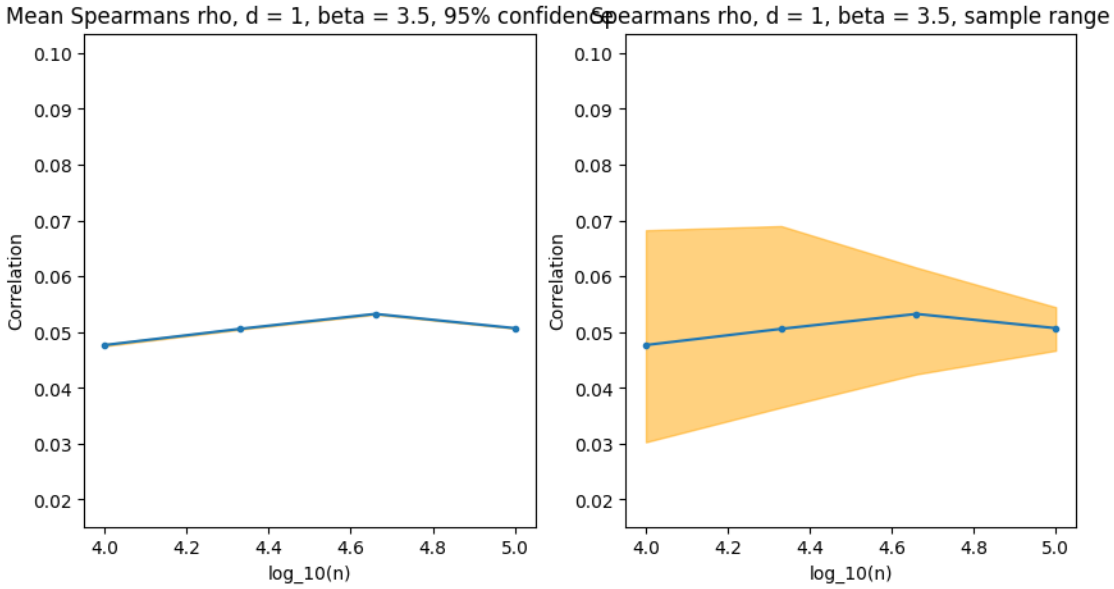
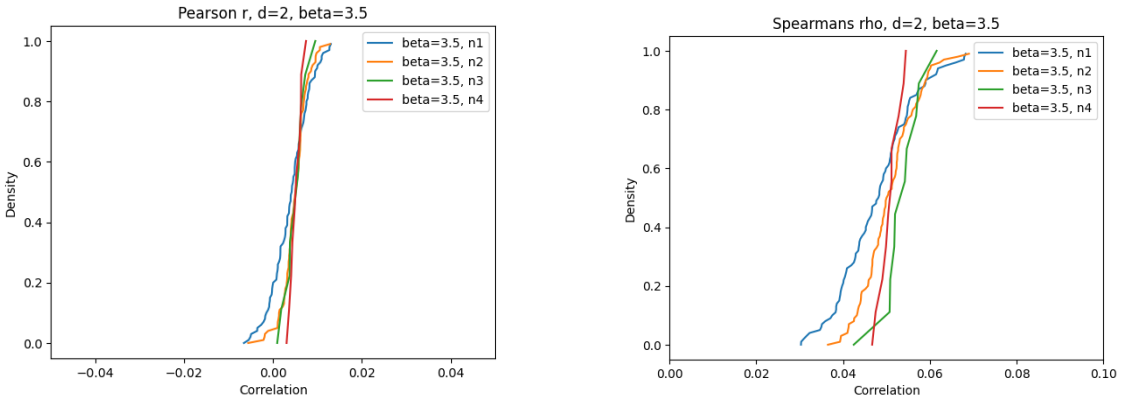


Figure 6.2: Mean values of Spearmans ρ depending on the size n of the GIRG. $d = 1, \beta = 3.5$. Figure (a) shows a 95% confidence interval, figure (b) highlights the area containing all recorded values



(a) Cumulative distribution of Pearsons r for $n \in \{10^4, 10^{4\frac{1}{3}}, 10^{4\frac{2}{3}}, 10^5\}$. $\beta = 3.5$, $d = 2$

(b) Cumulative distribution of Spearmans ρ for $n \in \{10^4, 10^{4\frac{1}{3}}, 10^{4\frac{2}{3}}, 10^5\}$. $\beta = 3.5$, $d = 2$

Figure 6.3: Values of r and ρ for different values of n in a 2 dimensional setting

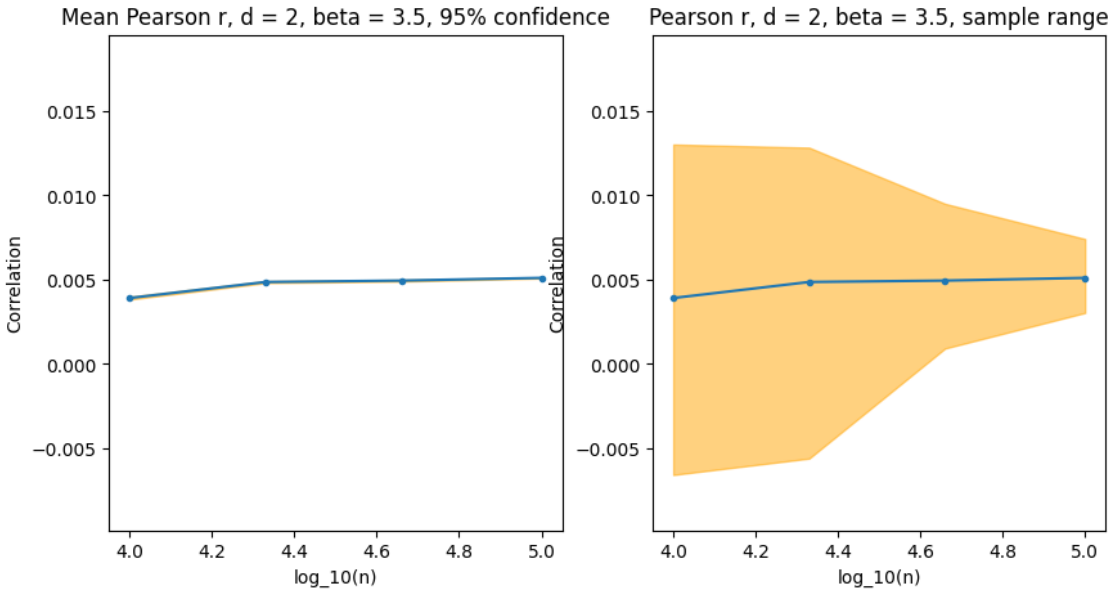


Figure 6.4: Mean values of Spearmans ρ depending on the size n of the GIRG. $d = 2, \beta = 3.5$. Figure (a) shows a 95% confidence interval, figure (b) highlights the area containing all recorded values

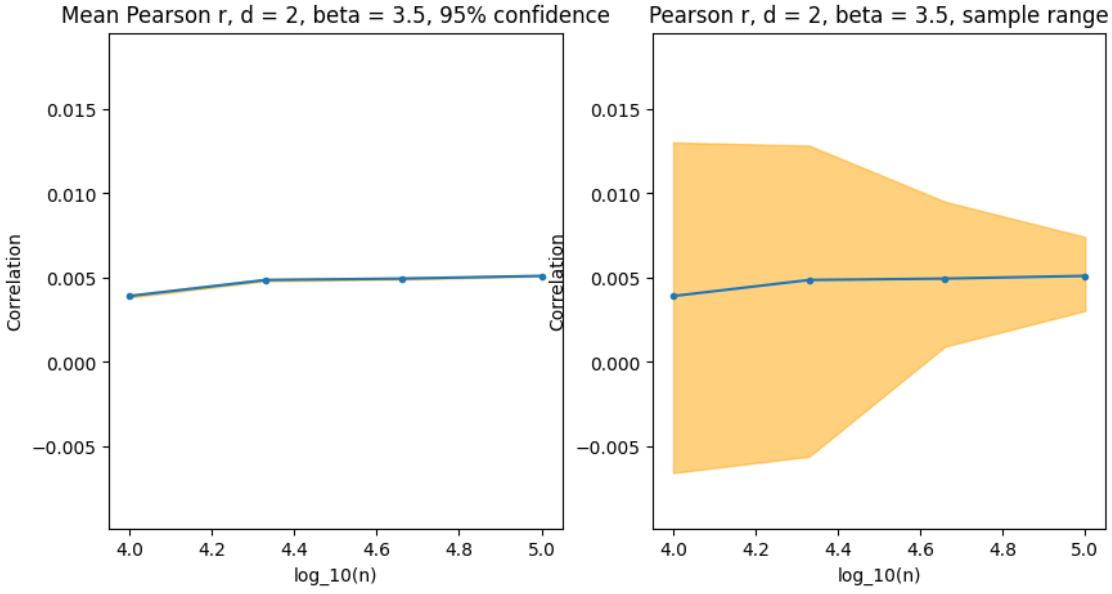


Figure 6.5: Mean values of Spearmans ρ depending on the size n of the GIRG. $d = 2, \beta = 3.5$. Figure (a) shows a 95% confidence interval, figure (b) highlights the area containing all recorded values