Generating Geometric Random Graphs with Boolean Distance Functions

Bachelor’s Thesis of

Maxime Christophe Rambaud

At the Department of Informatics
Institute of Theoretical Informatics (ITI)

Reviewer:	T.T.-Prof. Dr. Thomas Bläsius
Second reviewer:	PD Dr. Torsten Ueckerdt
Advisors:	Jean-Pierre von der Heydt
	Marcus Wilhelm

01.06.2024 – 01.10.2024

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself. I have not used any other than the aids that I have mentioned. I have marked all parts of the thesis that I have included from referenced literature, either in their original wording or paraphrasing their contents. I have followed the by-laws to implement scientific integrity at KIT.

Karlsruhe, 01.10.2024

[illegible]

. . . .
(Maxime Christophe Rambaud)

Abstract

Geometric Inhomogeneous Random Graphs (GIRGs) have proven particularly valuable for the realistic analysis of graph algorithms as they exhibit many real-world properties. However, connections between vertices are mostly limited to the case of vertices being close in every dimension. *BDF-GIRGs*, a recent extension, use distance functions based on arbitrary combinations of minima and maxima of the component-wise distances, potentially modeling even more realistic graphs.
We consider the threshold model of BDF-GIRGs and present the first sampling algorithm with linear runtime in the number of vertices. We characterize the runtime depending on the chosen distance function by identifying two properties: the *computational length*, which affects the runtime by a linear factor, and the *computational depth*, which has an exponential effect. We then further optimize the algorithm by reducing the computational length and depth, potentially leading to significant improvements of the runtime. The implementation we then provide is based on an existing sampling algorithm for GIRGs. We also enable the generation of graphs with a desired average degree by extending an existing algorithm. Additionally, we further investigate the effect of the length and depth for arbitrary BDF-GIRG algorithms. Based on the *Orthogonal Vectors Hypothesis (OVH)* we show a lower bound with matching dependency on the length and depth, under the assumption of arbitrary distributed positions. Finally, we empirically analyse our implementation and some sampled graphs. The results reveal that especially in the case of a more homogeneous weight distribution, the properties of BDF-GIRGs strongly depend on the distance function used.

Zusammenfassung

Geometric Inhomogeneous Random Graphs (GIRGs) haben sich bei der realistischen Analyse von Graphenalgorithmen als nützlich herausgestellt, da sie viele Eigenschaften von Echtweltgraphen aufweisen. Die Verbindung zwischen Knoten ist jedoch darauf beschränkt das diese in jeder Dimension nah beieinander liegen. *BDF-GIRGs*, eine kürzlich vorgestellte Erweiterung, verwenden beliebige Kombinationen von Minima und Maxima der komponentenweisen Distanz zwischen Knoten als Distanzfunktion und ermöglichen damit möglicherweise die Modellierung von noch realistischeren Graphen.
In dieser Arbeit betrachten wir die threshold Variante von BDF-GIRGs und präsentieren den ersten Generierungsalgorithmus mit linearer Laufzeit in der Anzahl an Knoten. Darüber hinaus beschreiben wir die Laufzeit in Abhängigkeit von der gewählten Distanzfunktion, indem wir zwei ihrer Eigenschaften charakterisieren: die *Länge*, welche einen linearen Einfluss auf die Laufzeit hat und die *Tiefe*, welche einen exponentiellen Einfluss hat. Anschließend optimieren wir den Algorithmus weiter, indem wir die Länge und Tiefe reduzieren, was teilweise zu erheblichen Laufzeitverbesserungen führt. Die Implementierung, die wir dann bereitstellen basiert, auf einem bestehenden Generierungsalgorithmus für GIRGs. Wir ermöglichen ebenfalls die Generierung von Graphen mit einem gewünschten durchschnittlichen Knotengrad Grad, indem wir einen bereits existierenden Algorithmus erweitern. Zusätzlich untersuchen wir die Auswirkungen der Länge und Tiefe auf die Laufzeit von BDF-GIRG Algorithmen. Basierend auf der *Orthogonal Vectors Hypothesis (OVH)* zeigen wir eine untere Schranke mit passenden Abhängigkeiten zu Länge und Tiefe für beliebig verteilte Positionen. Abschließend analysieren

wir die Laufzeit unserer Implementierung sowie einige Eigenschaft von generierten Graphen. Die Ergebnisse zeigen, dass insbesondere im Fall einer homogeneren Verteilung der Gewichte, die Eigenschaften von BDF-GIRG stark von der verwendeten Distanzfunktion abhängen.

Contents

1 Introduction

Many real-world structures can be modeled as graphs, where the vertices represent entities and the edges represent relationships between them. Applications range from social networks and internet infrastructure to road networks. Analyses performed on those graphs ranges from routing algorithms to complex analyses of social relations. While some computer scientists focus on developing algorithms for those analyses, others are interested in their theoretical examination. An important aspect of it is understanding the structures of the underlying graphs [CF06]. These structures can often explain why, e.g., worst-case analysis tends to be overly pessimistic. A promising approach is the study of random graphs, which are generated based on mathematical models that aim at emulating real-world scenarios [BF24]. This enables the possibility of average-case analyses and simplifies the generation of test instances. However, it is crucial that those random graphs have similar properties to real-world graphs, otherwise making their analysis obsolete.
Although the number of properties one can analyse is vast, many real-world graphs have been observed to exhibit four key characteristics: *scale-freeness* [FFF99], high *clustering* [SAK04], the *small-world phenomenon* [TM69], and the presence of a *giant component*. A network is said to be scale-free if the degree distribution follows a power-law. Intuitively, this property describes how most vertices are connected to only a small number of other vertices, while just a few have a large degree. Clustering describes the probability of two vertices being connected if they share a common neighbor. The natural interpretation is that two random persons having a friend in common are more likely to be friends, in contrast to two persons not related through an intermediary. The cluster coefficient is therefore known to be high in real-world graphs. A graph is said to exhibit the small-world phenomenon if its *diameter* (i.e., the maximal distance between any pair of vertices) is at most poly-logarithmic. A graph with a giant component is one where almost every vertex is reachable by any other.

One of the first and most popular models for generating random graphs is the *Erdős-Rényi model* [KR13]. Given n vertices and a probability p, each possible edge is generated with probability p. Erdős-Rényi graphs are particularly simple to generate and analyse, feature a giant component and also exhibit the small-world phenomenon. Their main drawback is that, due to the random distribution of edges, they lack any clustering. A model that counteracts this issue is that of *Random Geometric Graphs* (RGGs) [DC02]. Roughly speaking, the model works by assigning each vertex a position in an underlying space, and edges are then sampled if and only if the distance between two vertices is below a connection threshold. This procedure is motivated by the fact that most real-world structures are assumed to also be induced by some kind of geometry. This geometry induces high clustering and, for a sufficiently high connection threshold, also leads to a giant component. Although the classic model of RGGs does not exhibit the small-world phenomenon, it can be achieved by adding a few long-range edges [Erc11]. However, both of the models just presented lack the crucial property of scale-freeness. In both Erdős-Rényi graphs and RGGs, the degree distribution is homogeneous. A promising model to achieve scale-freeness is the *Hyperbolic Random Graph* (HRG) [Kri+10]. In HRGs, the positions of vertices are considered in the hyperbolic

plane rather than in the Euclidean space. HRGs exhibit all four desirable properties but have the disadvantage of being quite complex. A simpler and more general model is the one of *Geometric Inhomogeneous Random Graphs* (GIRGs) [BKL19]. It can be seen as an extension of RGGs, where vertices are equipped with weights following a power-law. The connection threshold between a pair of vertices then depends from the product of their weights. As a distance function for both RGGs and GIRGs, it is common to use the metric induced by the L_∞-norm (max-norm). While the L_∞-norm induces high clustering, it has been argued that for some types of networks this does not reflect the underlying structure well. For instance if we think of social networks, where each dimension represents some kind of attribute (*age, hobby, workplace*), two people are likely to know each other if the share only some properties and not necessarily all, like it would be implied by the L_∞-norm.
With this motivation, Kaufmann et al. [KRS24] introduced a large class of GIRG extensions called *BDF-GIRGs*, where the underlying distance is induced by a *Boolean Distance Function* (BDF) (see Definition 2.1). This family of distance functions consist of arbitrary combinations of minima and maxima of the component-wise distance between two vertices. Consider this example with three dimensions, where each dimension encodes one of the properties *age, hobby* and *workplace*. Let us assume that two persons know each other if they have a similar hobby and age, or if they have a common workplace. We can then use the BDF $\min\{dist_{workplace}, \max\{dist_{age}, dist_{hobby}\}\}$ to model this assumption and might expect graphs whose properties fit social networks better. Kaufmann et al. [KRS24] showed that the four desirable properties of being scale-free, high clustering, the existence of a giant component and a logarithmic diameter (small-world phenomenon) are also exhibited by BDF-GIRGs, no matter the concrete BDF, for power-law exponents $2 < \beta < 3$. However, unlike for HRGs and GIRGs, there are no efficient algorithms available to generate this very new class of graphs. For GIRGs, Bringmann et al. [BKL19] have shown that they can be generated in expected linear time for a fixed dimension, but with an exponential runtime in the dimensions. An efficient implementation of this algorithm has been provided by Bläsius et al. [Blä+22], enabling the generation of large GIRG instances, if the dimension is small.

The contribution of this thesis is to develop a simple and efficient generation algorithm for BDF-GIRGs. This does not only facilitate their generation but also enables their empirical analysis, as the influence of different BDFs remains widely unexplored. The algorithm we design works by transforming BDFs into a normal form, which we refer to as the *min-max form.* Based on it, BDF-GIRGs can be generated as a union of several GIRGs. The runtime is then linear in the number of vertices. The runtime in dependence of the BDF is influenced by two of its properties. The first is the amount of GIRGs that need to be sampled to obtain the BDF-GIRG and has a linear impact on the runtime. We call this property the *computational length.* The second is the largest dimension in which a GIRG needs to be sampled and impacts the runtime by an exponential factor. We refer to it as the *computational depth.* We then further optimize our algorithm by reducing the computational length and depth, potentially leading to significant improvements of the runtime. The result is then a simple extension that, based on the efficient implementation of Bläsius et al. [Blä+22], is capable of quickly sampling BDF-GIRGs. Additionally, we extend the provided algorithm, which estimates the threshold based on a desired average degree, to handle this more general class of distance functions. We also address the question of why the runtime for the generation of GIRGs grows exponentially in the number of dimensions. For this, we consider a key-assumption from fine-grained complexity theory, the *Orthogonal Vectors Hypothesis* (OVH). We deduce that for a more general class of BDF-GIRGs, with no assumptions about the positions of the vertices, no algorithm can run polynomially in the number of dimensions unless its runtime is quadratic in the number of edges (being the trivial $O(n^2 d)$ algorithm), unless OVH fails. Finally, we empirically analyse some of the graph properties mentioned earlier for a wide range of different BDF-GIRGs. Besides validating the earlier theoretical results, we take a broader look at the case of a power-law exponent $\beta > 3$, which leads to a more homogeneous degree distribution. We find out that in this case, BDF-GIRGs exhibit very different behaviors, depending on the chosen BDF. Based on those empirical observations, we classify BDFs into three categories and attempt to explain the differences.

1.1 Outline

We first introduce the formal definitions of BDFs, BDF-GIRGs and some their properties in Chapter 2. We also briefly discuss the idea behind the linear time algorithm for GIRGs. In Chapter 3 we show some lower bounds for the sampling of BDF-GIRGs. In Chapter 4 we then present the basic idea of the sampling algorithm for BDF-GIRGs as well as the algorithm used to estimate estimate the average degree. Afterwards, in Chapter 5 we discuss an important optimisation that we make to our algorithm which can potentially drastically increase the performance. In Chapter 6 we discuss implementation details as well as some properties of the generated BDF-GIRGs.

2 Preliminaries

In this chapter, we define Boolean Distance Functions as Kaufmann et al. [KRS24] did and introduce some properties to characterize them. We then discuss how their volume, an important property of BDFs, can be characterized. We then define BDF-GIRGs and show that they are a special case of the random graph model defined by Bringmann et al. [BKL18], from which a variety of properties follow directly. At the end, we discuss how GIRGs (we refer to them as L_∞-GIRGs) can be generated in linear time for a fixed dimension.

2.1 Underlying Space and Boolean Distance Functions

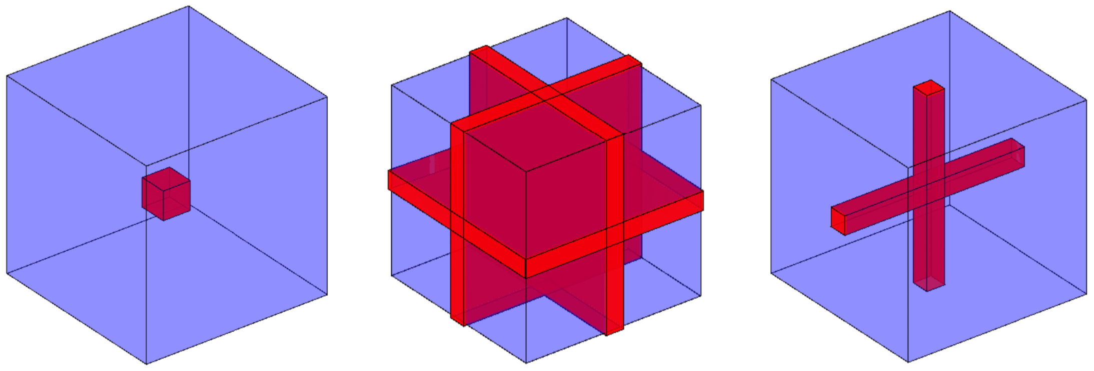
In this section, we first discuss the underlying space we use for BDF-GIRGs and then define Boolean Distance Functions.

Underlying space. A variety of different underlying spaces are used for geometrical graphs. In the case of GIRGs it is common to use the d-dimensional torus, which is written as $\mathbb{T}^d$. The component-wise distance between two vertices is then defined as:

$$|x_i - y_i|_T := \min\{|x_i - y_i|, 1 - |x_i - y_i|\} \text{ for } x, y \in \mathbb{T}^d.$$

A torus can be identified with the interval $[0, 1)^d$ where the opposite boundaries are identified. In one dimension, one can think of a torus as the interval $[0, 1)$ where the ends wrap around. Note that because of that, the component-wise distance can not exceed $\frac{1}{2}$. In the following we write $|x|$ denoting $|x|_T$.

Distance function. The distance between two elements $x, y \in \mathbb{T}^d$ can be defined as combination of the component-wise distance of x and y in different dimensions. The most common type of distance functions are those induced by a norm. Commonly, either the L_∞-norm, defined as $\max_{i\in[d]} |x_i| = |x|_\infty$, or the euclidean L_2-norm ($(\sum_{i=1}^{d} |x_i|^2)^{1/2}$) are used. As discussed earlier, we consider a wider range a distance functions, that also include non-metrics. An example of such a non-metric is the minimum component distance (MCD) defined as $\min_{i\in[d]}(|x_i - y_i|)$. In contrast to the L_∞-norm, which considers the dimension in which two positions differ the most, the MCD instead considers the dimension in which they differ the least. In the context of social-networks, where each dimension stands for a property, using the MCD can be seen as assuming that two people are friends if they have at least one property in common. In the same context, one could argue that a similarity in just one dimension for two people to know each other is somewhat of an overstatement. Therefore, we consider arbitrary combinations of min and max terms, called Boolean Distance Function (BDF), which allow encoding solutions that lay in-between the L_∞-norm and the MCD. A good intuition is to think about BDFs as a nested statement of **AND** and **OR** operators, where a maximum is an **AND** and a minimum is an **OR**. If we want to express that two people know each other if they share a hobby **OR** a job **AND** live next to each other, we can write: $\max\{dist_{live}, \min\{dist_{job}, dist_{hobby}\}\}$. The formal definition work by defining a BDF as a binary tree.



(a) $\max(x_0, x_1, x_2)$ **(b)** $\min(x_0, x_1, x_2)$ **(c)** $\max(x_0, \min(x_1, x_2))$

Figure 2.1: Comparison between three BDFs in $[0,1)^3$. The red area represents positions which are closer than 0.05 to the center of the cube.

Definition 2.1 (Def. 4 in [KRS24]): *Let $d \in \mathbb{N}$ be a positive integer, and let $\kappa : \mathbb{T}^d \to \mathbb{R}_{\geq 0}$ and $x = (x_1, x_2, .., x_d) \in \mathbb{T}^d$ be an arbitrary point. Then κ is a Boolean Distance Function (BDF) if:*

- *For $d = 1$: $\kappa(x) = |x|$.*
- *For $d \geq 2$: there exists a non-empty proper subset $S \subsetneq [d]$ of coordinates such that:*
  $$\kappa(x) = max(\kappa_1((x_i)_{i\in S}), \kappa_2((x_i)_{i\notin S})) \text{ or } \kappa(x) = min(\kappa_1((x_i)_{i\in S}), \kappa_2((x_i)_{i\notin S}))$$
  With $\kappa_1 : \mathbb{T}^{|S|} \to \mathbb{R}_{\geq 0}$ and $\kappa_2 : \mathbb{T}^{d-|S|} \to \mathbb{R}_{\geq 0}$ being BDFs.

A BDF is called outer-max (or outer-min) if it is defined as maximum (or minimum) of two BDFs. κ_1 and κ_2 are called comprising functions of κ.

To get a better sense of what a BDF might look like in practice, consider Figure 2.1, where the red area shows the area “next to” the center with respect to different BDFs. Observe that both the L_∞-norm and MCD are BDFs. For better readability, we do not write the BDFs in the formally defined manner but simplify the notation by omitting minima of minima or maxima of maxima. For example, we write $\max(0, \max(1, 2))$ as $\max(0, 1, 2)$. Also note that BDFs are generally non-metrics because they do not satisfy the triangle inequality.

We now introduce two properties of BDFs. They are used to describe the runtime of our BDF-GIRG algorithm with respect to the BDF. We recall that the idea of the algorithm is to sample BDF-GIRGs as a union of multiple GIRGs. While doing so two factors influence the runtime. The first is the maximal dimension a single GIRG can have and therefore has an exponential impact on the runtime. We describe this property as the *computational depth* of a BDF. The second factor, which we call the *computational length*, impacts the runtime by a linear factor and stands for the number of GIRGs that have to be generated to obtain a BDF-GIRG.

Definition 2.2 (Computational length and depth): *Let $\kappa : \mathbb{T}^d \to \mathbb{R}_{\geq 0}$ be a BDF. Then the computational length $L_c(\kappa)$ and computational depth $D_c(\kappa)$ of κ are recursively defined as:*

- *For $d = 1$, $L_c(\kappa) = D_c(\kappa) = 1$*

- *For $d \geq 2$ let κ_1 and κ_2 be the comprising functions of κ:*
  ***If** κ **is outer-max**:* $L_c(\kappa) := L_c(\kappa_1) \cdot L_c(\kappa_2)$ $D_c(\kappa) := D_c(\kappa_1) + D_c(\kappa_2)$
  ***If** κ **is outer-min**:* $L_c(\kappa) := L_c(\kappa_1) + L_c(\kappa_2)$ $D_c(\kappa) := \max(D_c(\kappa_1), D_c(\kappa_2))$

Observe, that for the special case of $\kappa(x) = \max_{i \in [d]} |x_i|$ (L_∞-norm) the depth is $D_c(\kappa) = d$ and the length is $L_c(\kappa) = 1$. This is intuitive as it is equivalent to just computing a single GIRG. For the MCD $\kappa(x) = \min_{i \in [d]} |x_i|$ the depth is $D_c(\kappa) = 1$ and the length is $L_c(\kappa) = d$. We see in Chapter 4 that its computation just consists of considering each dimension separately.

Now, we define a class of BDFs which Kaufmann et al. [KRS24] called *Single-Coordinate Outer-Max* (SCOM). Those are outer-max BDFs that can be written such that one of the comprising functions acts on a single coordinate. Kaufmann et al. demonstrated that the existence of sublinear cuts depend on whenever a SCOM-BDF is used or not. In Chapter 6 we see that, except a special case, this categorization also makes sense beyond the context it was introduced in.

Definition 2.3 (SCOM-BDF Definition 6. in [KRS24])**:** *Let $\kappa : \mathbb{T}^d \rightarrow \mathbb{R}_{\geq 0}$ be a BDF. We say that κ is Single-Coordinate Outer-Max (SCOM) if it can be written as:*

$$\kappa(x) = max(|x_k|, \kappa_0((x_i)_{i \neq k}))$$

for some coordinate $k \in [d]$ and some BDF $\kappa_0 : \mathbb{T}^{d-1} \rightarrow \mathbb{R}_{\geq 0}$. In dimension 1, $\kappa(x) = |x|$ is also a SCOM-BDF.

2.1.1 Volume of a BDF

In this section, we discuss how the volume of a BDF can be characterized both asymptotically and exactly as a function of the radius r. The volume is an important property, which later plays an important role in estimating the average degree, and optimizing the performance of the sampling algorithm.

By the volume of a BDF κ we mean the Lebesgue measure of $B_\kappa^r(x) = \{y \in \mathbb{T}^d | \kappa(x - y) < r\}$, being a ball with radius $r > 0$ centered around $x \in \mathbb{T}^d$ and using κ as distance function. Because the Lebesgue measure is invariant under translation, the choice x is irrelevant. We write the volume as $V_\kappa(r)$. Consider Figure 2.1 which represents $B_\kappa^{0.05}((0.5, 0.5, 0.5))$ for different κ. To describe $V_\kappa(r)$ as a function of r Kaufmann et al. [KRS24] introduced the *depth* of a BDF. In order to better differentiate it from the computational depth, we call it *volumetric depth.*

Definition 2.4 (Volumetric depth (Def. 5 in [KRS24]))**:** *Let $\kappa : \mathbb{T}^d \rightarrow \mathbb{R}_{\geq 0}$ be a BDF. Then the volumetric depth $D_v(\kappa)$ of κ is recursively defined as:*

- *For $d = 1$, $D_v(\kappa) = 1$*
- *For $d \geq 2$ let κ_1 and κ_2 be the comprising functions of κ:*
  ***If** κ **is outer-max**:* $D_v(\kappa) := D_v(\kappa_1) + D_v(\kappa_2)$
  ***If** κ **is outer-min**:* $D_v(\kappa) := \min(D_v(\kappa_1), D_v(\kappa_2))$

The only difference between the computational and volumetric depth is that, in the case of an outer-min BDF, the minimum of the depths of the two comprising functions is used, instead of the maximum. Consequently, the computational depth is greater or equal than the volumetric depth i.e., $D_c(\kappa) \geq D_v(\kappa)$ for any BDF κ. We can now use the volumetric depth to characterize the volume of BDF asymptotically for decreasing a radius.

Lemma 2.5 (Volume of BDF (Proposition 11 in [KRS24])): *Let $\kappa : \mathbb{T}^d \longrightarrow \mathbb{R}_{\geq 0}$ be a BDF, with volumetric depth $D_v(\kappa)$. Then $V_\kappa(r) = \Theta(r^{D_v(\kappa)})$ for $r \longrightarrow 0$.*

Proof. The full proof is given by Kaufmann et al. [KRS24] and requires the use of the Lebesgue integrals. However, we sketch the proof, to give a better intuition about the lemma. The proof is an induction on the dimensions d. Let $\kappa : \mathbb{T}^d \longrightarrow \mathbb{R}_{\geq 0}$ be a BDF:
For $d = 1$ and $r \in [0, \frac{1}{2}]$ the volume of is $2r = \Theta(r)$. Because $D_v(\kappa) = 1$ and therefore $\Theta(r^{D_v(\kappa)}) = \Theta(r)$, the induction hypothesis if fulfilled.
For $d \geq 2$, with κ_1, κ_2 comprising functions of κ, we differentiate between the outer-max and outer-min case. In the outer-max case, the volume is:

$$V_\kappa(r) = V_{\kappa_1}(r) \cdot V_{\kappa_2}(r) = \Theta(r^{D_v(\kappa_1)+D_v(\kappa_2)}) = \Theta(r^{D_v(\kappa)})$$

The intuition is that, for an $y \in \mathbb{T}^d$ to be in both the ball using $B^r_{\kappa_1}(x)$ and $B^r_{\kappa_2}(x)$, it is similar to when handling joint probabilities, where we consider the product of both probabilities. In the outer-min case, let WLOG $D_v(\kappa_1) \geq D_v(\kappa_2)$.

$$V_\kappa(r) = V_{\kappa_1}(r) + V_{\kappa_2}(r) - V_{\kappa_1}(r) \cdot V_{\kappa_2}(r) = \Theta(r^{D_v(\kappa_2)} + r^{D_v(\kappa_1)} - r^{D_v(\kappa_1) \cdot D_v(\kappa_2)}) = \Theta(r^{D_v(\kappa_2)})$$

The intuition this time is that we consider combined probabilities, where we sum the probability of two cases and subtract the intersection. Because $r^D_v(\kappa_2)$ dominates the asymptotic behavior for decreasing r, we can omit the remaining part of the term. ■

It is important to understand that this describes the asymptotic behavior of the volume for $r \longrightarrow 0$ by ignoring the intersections and comprising BDF with smaller volume in the outer-min case. This can help us to understand which parts of the BDF are responsible for how many edges. In other cases however, like estimating the average degree, we need to know the exact volume of a BDF. It that case, we can easily describe it as a polynomial of r.

Lemma 2.6: *Let $\kappa : \mathbb{T}^d \longrightarrow \mathbb{R}_{\geq 0}$ be a BDF. The volume $V_\kappa(r)$ for $r \in [0, \frac{1}{2}]$ can be described as a polynomial p_κ of degree d.*

Proof. The polynomial can be constructed recursively.
Let $d = 1$. For $r \in [0, \frac{1}{2}]$, $p_\kappa(r) = 2r$ and has degree 1.
For $d \geq 2$, let p_{κ_1} and p_{κ_2} be the polynomials of the two comprising BDFs of κ.
If κ is outer-max, $p_\kappa(r) = p_{\kappa_1}(r) \cdot p_{\kappa_2}(r)$. This results in a polynomial of degree $deg(p_{\kappa_1}) + deg(p_{\kappa_2})$.
If κ is outer-min, $p_\kappa(r) = p_{\kappa_1}(r) + p_{\kappa_2}(r) - p_{\kappa_1}(r) \cdot p_{\kappa_2}(r)$. Due to the intersection, the degree of the polynomial is also $deg(p_{\kappa_1}) + deg(p_{\kappa_2})$. ■

Note that for arbitrary for $r > \frac{1}{2}$ the volume is always 1. We can therefore write $V_\kappa(r) = p_\kappa(\min\{\frac{1}{2}, r\})$.

2.2 BDF-GIRGs

In this section, we first discuss what it means for weights to follow a power-law distribution. Following that, we use the previous definitions to formally define BDF-GIRGs. Afterwards, we look at some properties of BDF-GIRGs.

Power law weights. The formal definition of weights following a power-law distribution is quite technical. Therefore, we only mention two properties of power-law distributed weights that we need in this work.
(1) For an exponent $\beta > 2$, the fraction of weights that are larger or equal to w, is $w^{1-\beta}$. This specifically means that for lower power-law exponents, the weights tend to get larger. Conversely, for higher power-law exponents, the weights are expected to be lower.
(2) The sum of weights following a power-law distribution is linear. Meaning that for $n \in \mathbb{N}$ weights, their sum is in $\Theta(n)$.
For a formal definition of power-law distributed weights and a proof of (2), we refer to the work of Bringmann et al. [BKL18], specifically Section 2.2 and Lemma 4.1.

We are now able to define BDF-GIRGs as graphs where each vertex is assigned a positions and an edge exists if two vertices are closer the a certain threshold. The threshold depends on the weights of the vertices and the distance between vertices on the chosen BDFs.

Definition 2.7 (BDF-GIRG): *Let $\beta > 2, d, n \in \mathbb{N}$ and $\kappa : \mathbb{T}^d \to \mathbb{R}_{\geq 0}$ be a BDF. Let $w_1, ..., w_n \in \mathbb{R}_{>0}$ be a sequence of weights that is power-law distributed with exponent β and let $x_1, ..., x_n \in \mathbb{T}^d$ be randomly and uniformly drawn positions. The resulting κ-GIRG is then a graph $G = ([n], E)$ with:*

$$(u, v) \in E \Leftrightarrow \kappa(x_u - x_v) \leq \tau \cdot \left(\frac{w_u \cdot w_v}{n}\right)^{\frac{1}{D_V(\kappa)}}$$

For some $\tau > 0$ which controls the average degree. We call τ the threshold constant. Its precise choice is discussed in Section 4.3.

For better differentiation, we refer to GIRGs, like they are defined by Bringmann et al.[BKL19], as L_∞-GIRGs.

Note that in contrast to some literature, we only consider the *threshold variant* of BDF-GIRGs and L_∞-GIRGs. This means that the edges of our BDF-GIRG are uniquely determined by the positions and weights. A more general model is the *binomial variant*, where a temperature is used to soften the hard boundary between generating an edge and no edge. For each possible edge, a probability is then assigned, which decreases with increasing distance. This model is harder to study as an edge might be sampled between every pair of vertices. In this work we therefore only consider the threshold model.

2.2.1 Properties of BDF-GIRGs

Here, we discuss some known properties of BDF-GIRGs. They arise partly from the fact that BDF-GIRGs are a special case of a very general class of geometric random graphs defined by Bringmann et al. [BKL18], and partly from properties demonstrated by Kaufmann et al. [KRS24] who first introduced the concept of BDF-GIRGs.
First, we show that our model is a special case of the mentioned random graphs.

Lemma 2.8: *BDF-GIRGs in Definition 2.7 are included in the random graph model defined by Bringmann et al. [BKL18].*

Proof. We show the claim by using theorem 7.3 in [BKL18]. It states that random graphs with weights following a power-law distribution and positions that are drawn uniformly and at random, are a special case of the random graph model if the edge probability $p_{uv}(x_u, x_v)$ satisfies

$$p_{uv}(x_u, x_v) = \Theta\left(\min\{1, V_\kappa(\kappa(x_u - x_v))^{-\alpha} \cdot \left(\frac{w_u w_v}{W}\right)^{\alpha}\}\right)$$

where W is the sum of the weights and $\alpha > 1$. Note that α is the temperature used in the binomial variant we mentioned earlier. For $\alpha \longrightarrow \infty$ we get the threshold case we are considering here. To show the equivalence to our model, we first use $W \in \Theta(n)$ and $V_\kappa(r) = \Theta(r^{D_\nu(\kappa)})$ from Lemma 2.5. If we substitute this to the first formula, we get:

$$p_{uv}(x_u, x_v) = \Theta\left(\min\{1, \left(\frac{w_u w_v}{n \cdot \kappa((x_u - x_v)^{D_\nu(\kappa)})}\right)^{\alpha}\}\right)$$

Next we consider $\alpha \longrightarrow \infty$:

$$p_{uv}(x_u, x_v) = \begin{cases} 0 & \Theta\left(\left(\frac{w_u w_v}{n}\right)^{\frac{1}{D_\nu(\kappa)}}\right) > \kappa(x_u - x_v) \\ \Theta(1) & \Theta\left(\left(\frac{w_u w_v}{n}\right)^{\frac{1}{D_\nu(\kappa)}}\right) \leq \kappa(x_u - x_v) \end{cases}$$

Note, that the notation with Θ hides a constant for a fixed α and κ. As α changes we have to write a Θ around the threshold. We can see that Definition 2.7 matches the threshold case of the model if we replace the constant edge probability $\Theta(1)$ with 1. The Θ notation on the right side is then just hiding the threshold constant that we denoted as τ. ∎

From this lemma, we can now deduce that BDF-GIRGs possess some of the properties described in the introduction. BDF-GIRGs are scale-free, no matter which power-law exponent is used.

Theorem 2.9 (Theorem 2.1 in [BKL18])): *Let $\kappa : \mathbb{T}^d \longrightarrow \mathbb{R}_{\geq 0}$ be a BDF, and $G = (n, E)$ a κ-GIRG with weights following a power-law distribution with exponent β. With high probability $(1 - \frac{1}{n^{\omega(1)}})$ the degree-sequence follows a power-law distribution with exponent β.*

If we limit the power-law exponent to $\beta \in (2, 3)$, it was also shown that BDF-GIRGs have poly-logarithmic diameter i.e., satisfy the small-world phenomenon. Besides that, the largest component has linear size while all other are small, which satisfies the property of having a giant component.

Theorem 2.10 (Theorem 2.2 in [BKL18])): *Let $\kappa : \mathbb{T}^d \longrightarrow \mathbb{R}_{\geq 0}$ be a BDF, and G be a κ-GIRG with weights following a power-law distribution with exponent $2 < \beta < 3$. With high probability $(1 - \frac{1}{n^{\omega(1)}})$ the largest component has linear size and all others have at most poly-logarithmic size. Morever, the diameter is at most poly-logarithmic.*

From Kaufmann et al. [KRS24] we also know that the clustering coefficient is non-vanishing for BDF-GIRGs. Intuitively the clustering coefficient is the probability of two vertices being connected if they are both adjacent to a same vertex. The formal definition considers the number of triangles in which a vertex is involved, where a triangle corresponds to two adjacent vertices being connected. This count is then divided by the total number of possible vertex pairs that are adjacent to the vertex.

Definition 2.11 (Clustering Coefficient): *For a graph $G = ([n], E)$ the clustering coefficient of a vertex v is defined as:*

$$cc(v) = \begin{cases} \frac{1}{\binom{deg(v)}{2}} \cdot \#\{triangles\ in\ G\ containing\ v\} & if\ deg(v) > 2 \\ 0 & otherwise \end{cases}$$

The clustering coefficient of G is then $cc(G) = \frac{1}{n} \cdot \sum_{v \in V} cc(v)$

Theorem 2.12 (Theorem 3 in [KRS24])): *Let $\kappa : \mathbb{T}^d \to \mathbb{R}_{\geq 0}$ be a BDF, and G be a κ-GIRG with weights following a power-law distribution with exponent $2 < \beta < 3$. Then, with probability $1 - o(1)$, its clustering coefficient is constant i.e., $cc(G) = \Theta(1)$*

In Chapter 6 we see that this statement is also true for $\beta > 3$, where the graph tends to have a more homogeneous degree distribution, which favours clustering even more.

The key statement of Kaufmann et al. [KRS24] when introducing BDF-GIRGs was about the existence of sublinear cuts. More precisely, they categorize BDFs in SCOM and non-SCOM and show that only BDFs induced by a SCOM-BDF have high probability of having a sublinear cut. A sublinear cut is defined as a set of edges whose removal from the graph results in the graph being split into two parts, each of linear size.

Theorem 2.13 (Theorem 1&2 in [KRS24])): *Let $\kappa : \mathbb{T}^d \to \mathbb{R}_{\geq 0}$ be a BDF, and $G = ([n], E)$ be a κ-GIRG with weights following a power-law distribution with exponent $2 < \beta < 3$.*
If κ be a SCOM-BDF, G has a sublinear cut with probability $1 - o(1)$.
If κ is not a SCOM-BDF G has no sublinear cut with probability $1 - o(1)$.

2.3 Generation of L_∞-GIRGs

A major advantage of L_∞-GIRGs, compared to other random graph models, is that they can be generated in linear time. More precisely, for a graph with n vertices and m edges, the runtime is $O(n+m)$. Since for real-world graphs, one often assumes $m \in O(n)$, this results in a runtime of $O(n)$. This runtime is a significant benefit when generating graphs with millions of vertices, where the naive $O(n^2)$ algorithm *(checking every possible edge)* would be impractical. In the following, we outline the general approach of the algorithm first introduced by Bringmann et al. [BKL19]. The intuition given here is similar to that given by Bläsius at al. [Blä+22] who implemented the algorithm and provided a more intuitive explanation of it. However, we place a greater focus on the influence of the dimension that is usually assumed to be constant. The idea of the algorithm is that instead of considering every possible edge, only the pairs of vertices that are close to each other are considered. This is done with the use of a geometrical data structure that allow to query only vertices in the proximity of a certain position. To better understand the idea of this data structure, we first assume all weights to be equal and therefore also, the threshold between each pair of vertices.

Constant weights. Recall, that in L_∞-GIRGs the distance function induced by the L_∞-norm is used. When we are interested in finding the neighbors of v, we are interested in all vertices located within the ball centered around v with the radius equal to the threshold. This ball can then be thought of as a cube with side length of two times the threshold. Let us now divide the ground space into a grid of cells with a diameter as large as the threshold, and insert each vertex in the corresponding cell. We can observe that the union of the cell containing v and its neighboring cells always forms a superset of the previously mentioned ball around v. The edges are therefore sampled by considering only pairs of vertices that are in the same or in neighboring cells. This results in a linear runtime with respect to the number of vertices. Two factors are crucial for the runtime with respect to the dimension: the proportion of checked vertex pairs that form an edge and the number of neighboring cells that must be examined. If the cells have side lengths exactly equal to the threshold, the considered vertices are in a ball with a radius $\frac{3}{2}$ times larger than the threshold. The proportion we search for is then $(\frac{3}{2})^d$. In each dimension, we consider the cell in the center, as well as the neighbors on each side, leading two 3 cells per dimension. In total, this leads to 3^d neighboring cells. We see that the dimension affects the runtime by an exponential factor.

Power-law weights For arbitrary weights, the approach requires some modifications, as the threshold between each pair of vertices might be different. This is done by discretizing the number of weights and using grids of different size. The weights are discretized by assigning each one to a weight bucket, where the minimal and maximal weight inside the bucket only differ by a factor of two. Due to the power-law distribution, this leads to $log(n)$ many buckets. The ground space is recursively divided into a grid of cells with side length 2^{-l} for each recursion level l. This is done till each cell contains a constant number of vertices. Then, each pair of weight buckets is assigned the minimal level l such that the cell diameter is still larger than the largest possible threshold. The edges are then sampled by first iterating over each cell and determining which pairs of weight buckets are compared at the cells level. For each pair of weight buckets, the appropriate vertices are then queried.
By efficiently sorting the vertices in each weight bucket, access can be made in constant time per vertex. Since only a few weight bucket pairs at the deepest levels need to be compared, the overall runtime remains linear in the number of vertices. As with the case of constant weights, the number of neighboring cells increases exponentially with the dimension.

3 Computational Boundaries

In Section 2.3, we saw that L_∞-GIRGs can be generated in linear time with respect to the number of vertices, although the runtime grows exponentially with the number of dimensions. In this chapter, we investigate whether it is possible to generate L_∞-GIRGs and BDF-GIRGs in polynomial time with respect to the dimension. More precisely: We want to know if it is possible to achieve a sub-quadratic runtime with respect to the number of vertices while having a polynomial runtime in the number of dimensions. Formally, we ask if there exists an $\varepsilon > 0$, such that the runtime of our algorithm is in $O(n^{2-\varepsilon} \cdot poly(d))$. Intuitively speaking, we aim to determine if an algorithm exists that is more efficient than the trivial one, while avoiding an exponential runtime in the dimensions.
To address this, we first introduce a key assumption from fine-grained complexity theory, the *Orthogonal Vectors Hypothesis* (OVH). We then show a reduction from OVH to a more general version that we call *L-OVH*. Later, we discuss how the BDF-GIRG sampling can be formulated as a decision problem we call BOOLEAN DISTANCE FUNCTION - NEIGHBORS (BDF-NB), upon which we base the subsequent reductions. We then give a reduction for the special case of L_∞-NB, which helps to understand the idea. Afterwards, we generalize it for all BDFs and show how the depth and length can impact the generation of BDF-GIRGs.

3.1 Orthogonal Vectors Hypothesis

The decision problem ORTHGONAL VECTORS (OV) consists of deciding whether, in two sets of vectors, there exist two orthogonal vectors or not.

Definition 3.1 (ORTHOGONAL VECTOR): *We define the decision problem as:*

Given: $A, B \subseteq \{0,1\}^d$ *with* $|A| = |B| = n$.

Decide: *Does there exist* $a \in A$ *and* $b \in B$ *such that* a *and* b *are orthogonal i.e.,* $a \cdot b = 0$.

Note that OV assumes the vectors to be from $\{0,1\}^d$ i.e., the entries can only be 0 or 1 in each dimension. Two vectors are then orthogonal if and only if there is no dimension in which both have the entry 1. Two basic approaches exist to solve OV. The naive approach would be to check for every pair of vectors if, in any dimension, both have an entry 1. The resulting runtime is then $O(n^2 \cdot d)$. Another approach consists of enumerating every possible vector in $\{0,1\}^d$ which leads to a runtime of $O(nd \cdot 2^d)$. The runtimes of both approaches are similar to those required for generating L_∞-GIRGs. In both cases, solving the problem in linear time with respect to the number of vertices (or vectors) requires exponential runtime in the dimensions. The *Orthogonal Vectors Hypothesis* (OVH) states that no algorithm can achieve a significantly better runtime.

Definition 3.2 (Orthogonal Vectors Hypothesis): *Given an OV instance* $A, B \subseteq \{0,1\}^d$ *with* $|A| = |B| = n$ *and* $d \in \mathbb{N}$. *For no* $\varepsilon > 0$ *there is a* $O(n^{2-\varepsilon} \cdot poly(d))$ *OV-algorithm.*

Note that the statement only holds for arbitrary n and d. If d is assumed to be constant, for instance, the $O(nd \cdot 2^d)$ algorithm solves OV in linear time. This also generally holds for $d < \log(n)$. A helpful approach is to think of d as being in $O(\log^2(n))$. It is also worth noticing that the OVH follows from the *Strong Exponential Time Hypothesis* (SETH). For further algorithms and details on OVH and fine-grained complexity theory in general, we refer to the lecture of Kühnemann [Kün24].

3.1.1 L-OVH

We now take a look at a slightly modified version of OV. It consists of L instances of OV, each having the same dimension and amount of vectors. We want to decide if at least one of these instances is a yes-instance i.e., has a pair of orthogonal vectors.

Definition 3.3 (L-Orthogonal Vectors): *We define the decision problem as:*

Given:	$A_i, B_i \subset \{0,1\}^d$ *and* $\lvert A_i\rvert = \lvert B_i\rvert = n$ *for* $i \in \{1, ..., L\}$.
Decide:	*Does there exist an* $i \in \{1, ..., L\}$ *with an* $a \in A_i$, $b \in B_i$ *such that* $a \cdot b = 0$.

This modification is inspired by the way our BDF-GIRG algorithm, we discuss in Chapter 4, works. In this algorithm an edge is part of the graph if it has been generated in at least one of the L_∞-GIRGs the BDF-GIRG consists of. This is similar to L-OV which is a yes-instance when one of the OV instances is a yes-instance. We show this follow-up hypothesis of OVH that we call L-OVH. For no $\varepsilon > 0$ an algorithm can decide L-OV in $O(n^{2-\varepsilon} \cdot L \cdot poly(d))$. This bound is intuitive since solving L OV instances should naturally increase the runtime by a factor of L.

Lemma 3.4 (OVH implies L-OVH): *Given an L-OV instance $A_i, B_i \subseteq \{0,1\}^d$ with $\lvert A_i\rvert = \lvert B_i\rvert = n, L \le n^c$ for a $c > 0$ and $d \in \mathbb{N}$ for every $i \in \{1, ..., L\}$. For no $\varepsilon > 0$ there is a $O(n^{2-\varepsilon} \cdot L \cdot poly(d))$ L-OV algorithm, unless OVH fails.*

Proof. We give following reduction. Assume for now that L is a square number. Let $A, B \subseteq \{0,1\}^d$ be an instance of OV with $\lvert A\rvert = \lvert B\rvert = n$. Partition the sets A, B into $\sqrt{L}$ equally sized sets of size $\lceil \frac{n}{\sqrt{L}} \rceil$. For all partitions to have equal size, we use a dummy vector having only ones as entry. For each possible combination of those partitions from A and B, create one OV instance. This results in $\sqrt{L} \cdot \sqrt{L} = L$ instances that can be joined to one L-OV instance. If L is not a square number, we use the largest square number $L' < L$. Since $L' \in \Theta(L)$, the partition sizes remain in $\Theta(\frac{n}{\sqrt{L}})$. We then create a L'-OV instance and convert it to a L-OV instance by adding $L - L'$ no-instances of OV.

We show that the reduction is valid. Assume the OV instance to be a yes-instance and $a \in A, b \in B$ to be the pair of orthogonal vectors. Both must be in one of the $\sqrt{L}$ partitions of A and B, respectively. As for each combination of partitions, a OV instance has been created, at least one of the created OV instances is going to be a yes-instance and therefore also the L-OV instance. Now assume that the constructed L-OV instance is a yes-instance i.e., one of the constructed OV instances is a yes-instance. Let a, b be the orthogonal vectors of this instance. If one of the vectors is a dummy vector, the other must be a zero vector, which is orthogonal to any other vector. This makes the OV instance a yes-instance. If a, b are not dummy vectors a must be in A and b in B of the given OV instance. The original instance is therefore a yes-instance.

We analyse the runtime of the reduction. Partitioning A and B into partitions of size $\Theta(\frac{n}{\sqrt{L}})$ and creating L OV instances can be done in $O(n \cdot \sqrt{L} \cdot d)$. The created L-OV instance has the size $d \cdot \frac{n}{\sqrt{L}} =: n'$. Let us now assume that we can solve this instance in $O(n'^{2-\varepsilon} \cdot L \cdot poly(d))$. The runtime with respect to the size n of the OV instance is then:

$$O((\frac{n}{\sqrt{L}})^{2-\varepsilon} \cdot L \cdot poly(d)) = O(n^{2-\varepsilon} \cdot L^{(2-\varepsilon)/2} L \cdot poly(d)) = O(n^{2-\varepsilon} \cdot L^{\varepsilon/2} \cdot poly(d)).$$

Now recall that $L \leq n'^c$. The size of L with respect to the size of the OV instance is then: $L \leq \frac{n^c}{L^{c/2}} \Leftrightarrow L \leq n^{\frac{c}{1+c/2}}$. We use this upper-estimate of L and get $O(n^{2-\varepsilon} \cdot n^{\frac{c}{2+c}\varepsilon} \cdot poly(d))$. Observe that $2 - \varepsilon + \frac{c}{2+c}\varepsilon < 2$. We therefore have solved the OV instance in sub-quadratic time, a contradiction to OVH. ■

3.2 Boolean Distance Function - Neighbors

As discussed earlier, to show a lower bound for the time complexity of a BDF-GIRG algorithm, we must formulate it as a decision problem. Our decision problem looks as follows: Given a set of positions $C \subseteq \mathbb{R}^d$, do there exist two elements $c, c' \in C$ which are closer than 1, with respect to the given BDF. We see that this problem is very similar to sampling a BDF-GIRG. A BDF-GIRG sampling algorithm can indeed solve this decision problem by scaling down all elements in C to fit in $\mathbb{T}^d$ and adjusting the threshold accordingly. The instance is then a yes-instance if and only if at least one edge has been sampled.

Definition 3.5 (BDF - Neighbors): *We define the problem κ-NB as follows:*

Given: $C \subseteq \mathbb{R}^d$ *with* $|C| = n$.

Determine: *Does there exist* $c, c' \in C$ *with* $c \neq c'$ *and* $\kappa(c - c') \leq 1$.

Where $\kappa : \mathbb{R}^d \longrightarrow \mathbb{R}_{\geq 0}$ *is a BDF defined equivalently to Definition 2.1, but using the absolute value* $|x|$ *on* $\mathbb{R}$.

Note that the re-definition of κ is necessary for formal reasons, as it has been originally defined on a torus.

3.2.1 L_∞-NB

We first consider the case of the BDF κ being the L_∞-norm. It is crucial to understand the concept as both reductions for arbitrary κ follow the same idea. We show that OVH implies that no sub-quadratic algorithm exists for L_∞-NB.

Lemma 3.6: *Given an L_∞-NB instance $C \subseteq \mathbb{R}^d$ with $|C| = n$ and $d \in \mathbb{N}$. For no $\varepsilon > 0$ there is a $O(n^{2-\varepsilon} \cdot poly(d))$ L_∞-NB algorithm, unless OVH fails.*

Proof. We give following reduction. Let $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$ be an instance of OV. For each $a = (a_1, ..., a_n) \in A$ and $b = (b_1, ..., b_n) \in B$ we define $a' = (a'_1, ..., a'_n)$ and $b' = (b'_1, ..., b'_n)$. The components a'_i and b'_i are defined as follows:

$$a'_i := \begin{cases} \frac{1}{2} & \text{if } a_i = 0 \\ -1 & \text{if } a_i = 1 \end{cases}, \qquad b'_i := \begin{cases} -\frac{1}{2} & \text{if } b_i = 0 \\ 1 & \text{if } b_i = 1 \end{cases}$$

The L_∞-NB instance is then the set C containing a' and b' for every $a \in A$ and $b \in B$.

Table 3.1: Mapping of: a_i and b_i to a'_i and b'_i. We can see all possible cases that can happen in a dimension. Only in the last case the distance between the mapped values is larger then 1.

a_i	b_i	$a_i \cdot b_i$		a'_i	b'_i	$\lvert a'_i$-$b'_i\rvert_T$
0	0	0	$\rightarrow$	$\frac{1}{2}$	-$\frac{1}{2}$	1
0	1	0	$\rightarrow$	$\frac{1}{2}$	1	$\frac{1}{2}$
1	0	0	$\rightarrow$	-1	-$\frac{1}{2}$	$\frac{1}{2}$
1	1	1	$\rightarrow$	-1	1	2

We show that the reduction is valid. First, observe that the distance between two positions $c, c' \in C$, if constructed from vectors in the same set, is always $\frac{3}{2}$ unless $c = c'$. Let the OV instance be a yes-instance and $a \in A, b \in B$ the orthogonal vectors. Consider the corresponding vectors $a', b' \in C$. In each dimension their distance is equal to either 1 or $\frac{1}{2}$. The constructed L_∞-NB instance is therefore also a yes-instance. Now, let the constructed L_∞-NB instance be a yes-instance and $c, c' \in C$ positions close to each other. We know that the vectors they have been constructed from must be in different sets, namely A and B. From the construction, we know that in no dimension both have an entry 1. That makes the original instance a yes-instance. Consider also Table 3.1 for better comprehension.

We analyse the runtime of the reduction. Every vector can be transformed in $O(d)$. C can therefore be constructed in $O(n \cdot d)$. If we now assume that there is a sub-quadratic time algorithm for L_∞-NB, then for some $\varepsilon > 0$ the constructed instance can be solved in $O(n^{2-\varepsilon} \cdot poly(d))$. The total runtime for transforming and solving the instance is then $O(n^{2-\varepsilon} \cdot poly(d))$. This contradicts the OVH because we would have therefore solved the original instance in this runtime. ■

3.2.2 κ-NB

We now aim to extend the result from the previous lemma to arbitrary BDFs. We show the same lower

bound for BDFs, except that instead of the dimension, we consider the computational depth of the BDF. This is somehow intuitive, as the depth denotes the highest dimension of all L_∞-GIRGs we need to sample to obtain a BDF-GIRG. At first glance, it is not clear how this L_∞-GIRG is related to a BDF. Therefore, we introduce a lemma, whose proof follows directly from the construction of the BDF-GIRG algorithm in Chapter 4. It states that for each BDF, we can find a set S of dimensions, which are those of the largest L_∞-GIRG we would need to generated to obtain a BDF-GIRG.

Lemma 4.3: *Let $\kappa : \mathbb{R}^d \rightarrow \mathbb{R}_{\geq 0}$ be a BDF acting on $\mathbb{R}^d$. There exist a subset $S \subseteq [d]$ with $|S| = D_c(\kappa)$, such that $\kappa(x) \leq \max_{i \in [S]} |x_i|$ for every $x \in \mathbb{R}^d$. Moreover, if $\min_{i \in [d] \setminus S} |x_i| \geq \kappa(x)$, then $\kappa(x) = \max_{i \in [S]} |x_i|$. S can be computed in $O(d^2)$.*

Note that for the reduction, we must assume the dimension to be polynomial in the depth, otherwise we are not able to construct an κ-NB instance in $O(n^{2-\varepsilon} \cdot poly(d))$ and the reduction fails.

Lemma 3.7: *Given an κ-NB instance $C \subseteq \mathbb{R}^d$ with $|C| = n$ and $d \in \mathbb{N}$ with $D_c(\kappa) \in poly(d)$. For no $\varepsilon > 0$ there is a $O(n^{2-\varepsilon} \cdot poly(D_c(\kappa)))$ κ-NB algorithm, unless OVH fails.*

Proof. We give following reduction. Let $C \subseteq \mathbb{R}^{D_c(\kappa)}$, $|C| = n$ be an instance of L_∞-NB. There exists a permutation of the d dimensions κ is acting one, such that the S from Lemma 4.3 has the form $S = [1, ..., D_c(\kappa)]$. We call those permuted positions C_p. We assign a fixed order $\{c^1, ..., c^n\}$ for the elements in C_p. For every $D_c(\kappa)$-dimensional position $c^j \in C_p$ with $j \in \{1, ..., n\}$ we construct a d-dimensional position c'^j where the first $D_c(\kappa)$ dimensions are equal to those of c^j and all remaining dimensions have the entry $2j$. The constructed κ-NP instance then consists of $C' = \{c'^1, ..., c'^n\}$ containing each constructed position.

We show that the reduction is valid. Assume that the original L_∞-NB instance is a yes-instance and $x, y \in C$ with $|x - y|_\infty \leq 1$ are positions close to each other. Let x', y' be the constructed positions in C'. From the inequality in Lemma 4.3 it follows that $\kappa(x - y) \leq |x - y|_\infty \leq 1$. C' is therefore a yes instance. Now assume C' to be a yes-instance and $x', y' \in C'$ with $\kappa(x - y) \leq 1$ to be positions close to each other. For each dimension, not in S i.e., the dimensions $i \in [D_c(\kappa)+1, d]$, the distance between two positions is at least 2. From Lemma 4.3, it follows that $\kappa(x) = \max_{i \in S} |x_i| \leq 1$. The original instance is therefore a yes-instance.

We analyse the runtime of the reduction. First, note that $d \in poly(D_c(\kappa))$. S can be computed in $d^2 \in poly(D_c(\kappa))$ and C can be permuted to C_p in $O(n \cdot poly(D_c(\kappa))$. Assigning each element of fixed order takes $O(n)$ and constructing C' takes $O(n \cdot poly(D_c(\kappa)))$. We now assume that for some $\varepsilon > 0$ the constructed instance can be solved in
$O(n^{2-\varepsilon} \cdot poly(D_c(\kappa)))$. As the size of the original and the constructed instance differ by only $O(poly(D_c(\kappa)))$, we can solve the given L_∞-NB instance in $O(n^{2-\varepsilon} \cdot poly(D_c(\kappa)))$. This contradicts the Lemma 3.6. ■

3.2.3 Taking Length into account

Finally, we want to use the L-OVH, which we showed to follow from the OVH, to demonstrate that there is a class of BDFs where not only the computational depth but also the length contributes to the lower bound. Specifically, we show that for any length $L \in \mathbb{N}$ and depth $D \in \mathbb{N}$, there exists a BDF κ such that no algorithm can solve κ-NB in $O(n^{2-\varepsilon} \cdot L_c(\kappa) \cdot poly(D_c(\kappa)))$ for a $\varepsilon > 0$. Note, that this bound is not valid for every BDF. In Chapter 4 we see that we can construct BDFs such that just one L_∞-GIRG is high dimensional and therefore dominates the runtime, while the remaining L_∞-GIRGs are low-dimensional.

Lemma 3.8: *For any $L, D \in \mathbb{N}$, there exists a BDF κ with $L_c(\kappa) = L$ and $D_c(\kappa) = D$ such that for no $\varepsilon > 0$ a $O(n^{2-\varepsilon} \cdot L_c(\kappa) \cdot poly(D_c(\kappa)))$ κ-NB algorithm exists, unless OVH fails.*

Proof. We construct a BDF that meets the requirements outlined above and acts on $L \cdot D$ dimensions. For every $i \in \{1, ..., L\}$ we define a κ_i that is the L_∞-norm on D dimensions and ignores the remaining dimensions.

$$\kappa_i(x) = \max_{i \in [(i-1) \cdot D+1, i \cdot D]} |x_i| \text{ for } x \in \mathbb{R}^{D \cdot L} \text{ and } i \in \{1, ..., L\}$$

We then combine those max-terms into a BDF

$$\kappa(x) = \min\{\kappa_0(x), ..., \kappa_L(x)\} \text{ for } x \in \mathbb{R}^{D \cdot L}$$

Observe that κ is a BDF, and both the computational depth and length satisfy our requirements.

Next, we give following reduction from L-OVH. Let $A_i, B_i \subseteq \{0,1\}^{D_c(\kappa)}$ with $|A| = |B| = n$ for $i \in \{1, ..., L_c(\kappa)\}$ be a L-OV instance. For each $i \in \{1, ..., L_c(\kappa)\}$ we construct a L_∞-NB instance C_i, in the same way as in Lemma 3.6. We then define an order for the positions in C_i to be $\{c_i^1, ..., c_i^n\}$. We can now construct a κ-NB instance by combining L vectors, one from each C_i, to a single vector c':

$$c'^j = \begin{pmatrix} c_1^j \\ \vdots \\ c_L^j \end{pmatrix} \in \mathbb{R}^{D_c(\kappa) \cdot L_c(\kappa)} \text{ for every } j \in \{1, ..., n\}.$$

The resulting κ-NB instance is then $C' = \{c'^1, ..., c'^n\}$.

We show that the reduction is valid. Assume that the original L-OV instance is a yes-instance. Then there exists an $i \in \{1, ..., L\}$ and $a \in A_i, b \in B_i$ such that $a \cdot b = 0$. From the proof of Lemma 3.6 we know that the constructed L_∞-NB instance C_i is a yes-instance. Therefore, there exist $x, y \in C_i$ with $\max_{j \in [D_c(\kappa)]} |x_j - y_j| \leq 1$. Let $x', y' \in C'$ be the positions constructed from x, y. Then $\kappa(x' - y') \leq 1$, because the max-term $\kappa_i(x' - y') \leq 1$. The constructed κ-NB instance is therefore a yes-instance. Now assume that the constructed instance is a yes-instance with $x', y' \in C' : \kappa(x' - y') \leq 1$. By the form of the BDF, there exists an i such that $\kappa_i(x' - y') \leq 1$. Then in the L_∞-NB instance C_i there must be $x, y \in C_i$ with $\max_{i \in [D_c(\kappa)]} |x_i - y_i| \leq 1$. This instance is therefore a yes-instance. It follows that the L-OV instance must also be a yes-instance.

We analyse the runtime of the reduction. Constructing a L_∞-NB instance can be done in $O(n \cdot d)$. In total $O(L_c(\kappa) \cdot n \cdot d)$ is needed to construct all $L_c(\kappa)$ L_∞-NB instances. Enumerating each positions and merging the positions can be done in $O(L_c(\kappa) \cdot n \cdot d)$. The resulting instance then has the size n and $D_c(\kappa) \cdot L_c(\kappa)$ dimensions. We now assume that some $\varepsilon > 0$ the constructed instance can be solved in $O(n^{2-\varepsilon} \cdot L_c(\kappa) \cdot poly(D_c(\kappa)))$. The total runtime for transforming and solving the instance would be $O(n^{2-\varepsilon} \cdot L_c(\kappa) \cdot poly(D_c(\kappa)))$. This would be a contradiction to L-OVH.

∎

Note that this result is interesting for $L_c(\kappa) \in \omega(poly(d))$, as otherwise the result already follows directly from the previous lemma. In conclusion, we have shown through these reductions that it is unlikely that an algorithm exists which has both linear runtime in the number of vertices and polynomial runtime in the dimensions. However, it is important to highlight that the reductions presented ignore an important assumption about the input values for the BDF-GIRG sampling algorithm: the uniform distribution of positions. Thus, it is still not ruled out that such an algorithm may exist.

4 Generating BDF-GIRGs

In this chapter, we first focus on the generation of BDF-GIRGs. We begin by examining how a BDF can be transformed into a normal form, which we refer to as the *min-max form.* After that, we discuss how the actual generation algorithm works and analyse its runtime. Next, we aim to provide some intuition for the volume $V_\kappa(r)$ and how it can help us estimate the number of edges generated. We also discuss why the threshold considers the weights with an exponent of $\frac{1}{D_\nu(\kappa)}$. Finally, we discuss how to choose the threshold constant based on the desired average degree and the challenges that arise in doing so.

4.1 Generation

In this section we discuss the main idea of the generation algorithm for BDF-GIRGs. We start by giving some intuition about BDFs.

Consider the BDF $\kappa(x) = \min_{i\in[d]}\{|x_i|\}$ with $x \in \mathbb{T}^d, d \in \mathbb{N}$, which is the minimal component distance. A pair of vertices is connected if their distance is lower than the threshold in at least one dimension. We can therefore consider each dimension separately i.e., compute a one-dimensional L_∞-GIRG for each dimension. An edge is then included between two vertices if in at least one of the computed L_∞-GIRGs the vertices are connected. This results in a $O(n \cdot d)$ algorithm for MCD-GIRGs, first described by Lengler and Todorovic [LT17]. This approach, however, can be extended beyond MCD-GIRGs. In fact, for any BDF that can be expressed as the minimum of other BDFs, we can compute the edge set each of those BDF-GIRGs separately. The desired edge set is then the union of all generated edges.

Lemma 4.1: *Let $\kappa_i : \mathbb{T}^{|S_i|} \to \mathbb{R}_{\geq 0}$ be BDFs acting on the dimensions $S_i \subseteq [d]$ for $i \in \{1, ..., L\}, d, L \in \mathbb{N}$. Furthermore, let $x_1, ..., x_n \in \mathbb{T}^d$ be positions and $w_1, ..., w_n \in \mathbb{R}_{\geq 0}$ weights following a power-law distribution for $n \in \mathbb{N}$. Let $G_i = ([n], E_i)$ be the resulting κ_i-GIRGs. For*

$$\kappa(x) = \min_{i\in\{1,..,L\}}\{\kappa_i((x_j)_{j\in S_i})\} \text{ with } x \in \mathbb{T}^d \qquad \text{and} \qquad E := \bigcup_{i\in\{1,..,L\}} E_i$$

If κ is a BDF, $G = ([n], E)$ is the resulting κ-GIRG.

Proof. First, observe that the threshold between two vertices u, v is the same in each graph as it does not depend on the positions. We write $k_{(u,v)} := \tau \cdot \left(\frac{w_u w_v}{n}\right)^{D_\nu(\kappa)}$ for the threshold between two vertices u, v.
Let $(u, v) \in E$ now be an arbitrary edge of the κ-GIRG. Because κ is a defined as the minimum of BDFs, there must be an $i \in \{1, ..., L\}$ such that $k_{(v,u)} \geq \kappa(x_u - x_v) = \kappa_i(x_u - x_v)$. Therefore the edge (u, v) is in E_i.
Let $(u, v) \in E_i$ be an arbitrary edge of a κ_i-GIRG. Because κ is defined as the minimum κ_i and the other BDFs, $k_{(v,u)} \geq \kappa_i(x_u - x_v) \geq \kappa_i(x_u - x_v)$. The edge must therefore also be in the according κ-GIRG. ■

We can now use of this property, along with the fact that algorithms for computing L_∞-GIRGs exist. Recall that a BDF can be seen as a nesting of **AND** and **OR** operators. We can therefore also think about a BDF as a logical formula. We are then looking for a form that consists of **ORs** of **ANDs**. This form is called *disjunctive normal form* in the context of logical formulas. For BDFs this form then looks like:

$$\min(\max((x_i)_{i\in S_1}), ..., \max((x_i)_{i\in S_k}) \text{ for } S_i \subseteq [d] \text{ with } i \in \{1, ..., k\}\ k \in \mathbb{N}.$$

We can compute this form using the distributive property of minima and maxima, similar to how it is done with logical formulas. The number of max-terms is then going to be the length of the BDF and the depth is going to be the size of the largest max-term. Similar to the definition of BDFs, the construction is recursive. We describe the construction and prove its correctness.

Lemma 4.2: *For every BDF $\kappa : \mathbb{T}^d \to \mathbb{R}_{\geq 0}$ there exist and a set $\mathcal{S} = \{S_i \subseteq [d] | 1 \leq i \leq L_c(\kappa)\}$ such that: $\kappa(x) = \min\{max((|x_j|)_{j\in S_i}) | S_i \in \mathcal{S}\}$ for every $x \in \mathbb{T}^d$. Moreover the largest set in $\mathcal{S}$ has size $D_c(\kappa)$.*

Proof. We define $\mathcal{S}$ recursively and show by induction that the constructed set is valid. For better readability, we write $\max(S_i)$ instead of $\max\big((x_j)_{j\in S_i}\big)$ and κ instead of $\kappa(x)$ for $x \in \mathbb{T}^d$.
Let $d = 1$, i.e. κ is acting on a single dimension $i \in [d]$. We chose $\mathcal{S} = \{\{i\}\}$ with just one set, containing the one dimension it is acting on. $\mathcal{S}$ is then a valid set as:

$$\min(\max(|x_i|)) = |x_i| = \kappa(x) \text{ for any } x \in \mathbb{T}.$$

The size of $\mathcal{S}$ is then $|\mathcal{S}| = 1 = L_c(\kappa)$ and the size of the largest set is $1 = D_c(\kappa)$. The induction hypothesis is therefore fulfilled.
For $d \geq 2$, let $\mathcal{S}^1$ and $\mathcal{S}^2$ be the sets of the comprising functions κ_1 and κ_2:
If the BDF is an outer-min, we chose $\mathcal{S} := \mathcal{S}^1 \cup \mathcal{S}^2$. We see that it is a valid set:

$$\kappa = \min(\kappa_1, \kappa_2) = \min\left[\min_{S_i\in\mathcal{S}^1}\big(\max(S_i)\big), \min_{S_i\in\mathcal{S}^2}\big(\max(S_i)\big)\right] = \min\big(\max_{S_i\in\mathcal{S}^1\cup\mathcal{S}^2}(S_i)\big)$$

By induction hypothesis $|\mathcal{S}^1| = L_c(\kappa_1)$ and $|\mathcal{S}^2| = L_c(\kappa_2)$. Because κ_1 and κ_2 act on different dimensions, $\mathcal{S}^1$ and $\mathcal{S}^2$ are disjoint. We get $|\mathcal{S}^1 \cup \mathcal{S}^2| = |\mathcal{S}^1| + |\mathcal{S}^2| = L_c(\kappa_1) + L_c(\kappa_2) = L_c(\kappa)$. The largest set in $\mathcal{S}^1 \cup \mathcal{S}^2$ is the maximum of the largest set in $\mathcal{S}^1$ and $\mathcal{S}^2$. We therefore get $\max_{S\in\mathcal{S}^1\cup\mathcal{S}^2}(|S|) = \max(D_c(\kappa_1), D_c(\kappa_2)) = D_c(\kappa)$.
If the BDF is on outer-max, we choose $\mathcal{S} := \mathcal{S}^1 \times \mathcal{S}^2$ a min-max set. We see that:

$$\kappa = \min(\kappa_1, \kappa_2) = \max\left[\min\big(\max_{S_i\in\mathcal{S}^1}(S_i)\big), \min\big(\max_{S_i\in\mathcal{S}^2}(S_i)\big)\right] = \min\big(\max_{S_i\in\mathcal{S}^1\times\mathcal{S}^2}(S_i)\big)$$

By induction hypothesis $|\mathcal{S}^1| = L_c(\kappa_1)$ and $|\mathcal{S}^2| = L_c(\kappa_2)$. We get $|\mathcal{S}^1 \times \mathcal{S}^2| = |\mathcal{S}^1| \cdot |\mathcal{S}^2| = L_c(\kappa)$ for the length. The largest set in $\mathcal{S}^1 \times \mathcal{S}^2$ is the union of the two largest sets in $\mathcal{S}^1$ and $\mathcal{S}^2$. We get $\max_{S\in\mathcal{S}^1\times\mathcal{S}^2}(|S|) = \max_{S\in\mathcal{S}^1}(|S|) + \max_{S\in\mathcal{S}^2}(|S|) = D_c(\kappa_1) + D_c(\kappa_2) = D_c(\kappa)$. ∎

We call $\mathcal{S}$ the *min-max set* of κ and refer to a set $S \in \mathcal{S}$ as a max set.

With this knowledge we now describe the process of sampling BDF-GIRGs as following three steps: Compute the min-max form $\mathcal{S}$ of the BDF. For each $S \in \mathcal{S}$ compute a L_∞-GIRG with the positions adapted to the dimension in S. The edges of the BDF-GIRG are then the union of all generated edges. If we assume the runtime of the L_∞-GIRG sampling algorithm to be $O((n+m) \cdot \rho^d)$ for some constant $\rho > 1$, the runtime of the sampling algorithm described is $O(L_c(\kappa) \cdot (n+m) \cdot \rho^{D_c(\kappa)})$.

We can also easily prove Lemma 4.3 that we introduced in the previous chapter. We searched for the dimensions that correspond to the largest L_∞-GIRG that needs to be sampled. As each max-set in the min-max set represents the dimensions of a L_∞-GIRG, those dimensions are simply the one in the largest max-set of the min-max set. We must, however, prove how it can be determined in $O(d^2)$.

Lemma 4.3: *Let $\kappa : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ be a BDF acting on $\mathbb{R}^d$. There exist a subset $S \subseteq [d]$ with $|S| = D_c(\kappa)$, such that $\kappa(x) \leq \max_{i\in[S]} |x_i|$ for every $x \in \mathbb{R}^d$. Moreover, if $\min_{i\in[d]\setminus S} |x_i| \geq \kappa(x)$, then $\kappa(x) = \max_{i\in[S]} |x_i|$. S can be computed in $O(d^2)$.*

Proof. Let $\mathcal{S}$ be the min-max set of κ and S_{max} the largest max set. We show that S has the described properties. First, $|S_{max}| = D_c(\kappa)$ by definition. As κ is the minimum of multiple max terms $\kappa(x) = \min_{S\in\mathcal{S}}(\max((x_i)_{i\in S})) \leq \max((x_i)_{i\in S_{max}})$ for any $x \in \mathbb{R}^d$. Moreover, if for some $x \in \mathbb{R}^d$, in any dimension not in S_{max} the component-wise distance is larger or equal to $\kappa(x)$, $\max((x_i)_{i\in S_{max}}) = \kappa(x)$. To compute S_j in $O(d^2)$ we modify the construction of Lemma 4.2. In each recursion step, instead of considering all sets we only return S_{max}. If it is not unique, we select an arbitrary one. In each step we therefore only consider two sets of a size at most d. As each BDF consists of at most $2d - 1$ BDFs, the resulting runtime is $O(d^2)$. ∎

4.2 Volume & Edge Probability

In this section, we give some intuition about the volume and how it can help us to estimate the average degree.

First, recall that we are considering a torus as ground space, which has a volume of 1. Therefore for any BDF κ and $r > \frac{1}{2}$, $V_\kappa(r) = 1$. Also the ratio of some volume $V_\kappa(r)$ to the total volume is always $V_\kappa(r)$. This property is particularly useful if we recall a second property, which is that the positions of the vertices of a BDF-GIRG are distributed uniformly. This means that given a subspace of $\mathbb{T}^d$, the probability of a randomly drawn position to be inside this subspace is equal to its volume. With that we can describe the expected number of randomly drawn positions to be in a subspace. Let $\mathcal{P} \subseteq \mathbb{T}^d$ be a set of randomly and uniformly drawn positions with $|\mathcal{P}| = n$. For a (subspace) $A \subseteq \mathbb{T}^d$ the expected number of $p \in \mathcal{P}$ to be in A are:

$$\mathbb{E}[|A \cap \mathcal{P}|] = n \cdot \lambda(A) \qquad \text{where } \lambda(A) \text{ is the Lebesgue volume of } A.$$

We use this property to describe the expected degree of a vertex in a κ-GIRG, for a BDF κ. Let v, u be arbitrary vertices. The probability of them being adjacent i.e., for an edge (u, v) to exist, is $V_\kappa(k_{(u,v)})$, where $k_{(u,v)}$ is the threshold between u and v. The expected degree of a vertex v is then the summed probability of each possible edge adjacent to v:

$$\mathbb{E}[deg(v)] = \sum_{\substack{u\in V\\ v\neq u}} V_\kappa(k_{(u,v)})$$

We can now use this knowledge to understand why the weights are considered with exponent $\frac{1}{D_v(\kappa)}$. As we want the choice of the weight distribution to reflect on the degree distribution, we want each vertex to have an expected degree that depends linearly on its weight. From Lemma 2.5 we know that $V_\kappa(r) = \Theta(r^{D_v(\kappa)})$ for decreasing r. If we now insert this formula and the threshold formula for $k_{(u,v)}$, we get:

$$\mathbb{E}[deg(v)] = \sum_{\substack{u\in V\\ v\neq u}} V_\kappa\Big(\tau \cdot (\frac{w_v \cdot w_u}{n})^{\frac{1}{D_v(\kappa)}}\Big) = \sum_{\substack{u\in V\\ v\neq u}} \Theta\Big(\tau^{D_v(\kappa)} \cdot \frac{w_v \cdot w_u}{n}\Big).$$

We use the threshold formula from Definition 2.7 in the first step and the volume formula in the second step. We can now use that $\tau^{D_v(\kappa)}$ is constant and that the sum of the weights is $\Theta(n)$.

$$\sum_{\substack{u \in V \\ v \neq u}} \Theta\Big(\tau^{D_v(\kappa)} \cdot \frac{w_v \cdot w_u}{n}\Big) = w_v \cdot \sum_{\substack{u \in V \\ v \neq u}} \Theta\Big(\frac{w_u}{n}\Big) = w_v \cdot \Theta(1) = \Theta(w_v).$$

Note that for another exponent than $\frac{1}{D_v(\kappa)}$ the expected degree would not depend linearly from the weights of the vertex.

4.3 Choice of the threshold constant

In this section we discuss how the threshold constant, we denote as τ in Definition 2.7, can be chosen such that the sampled graph has a desired average degree. We can of course not predict the average degree exactly, as it depends on the positions used, but we can compute the expected average degree. We start by showing that there can be no closed formula for the threshold constant, even for equal weights. We then consider arbitrary weights, and how we can compute the expected average degree, given the threshold constant, in linear time. The resulting algorithm is then a modified version of the one designed by Bläsius et al. [Blä+22] performing a binary search over the threshold constant.

Constant weights. To better understand how we are estimating the average degree, let us first consider a L_∞-GIRG in $d \in \mathbb{N}$ dimensions and weights all equal to 1. The threshold is then $k_{(u,v)} = \frac{\tau}{n}$ between every pair of vertices u and v. From the previous chapter we know that expected degree of a vertex v can be written as the sum the volumes $V_\kappa(k_{(v,u)})$ for every possible edge adjacent to v. For L_∞-GIRGs using Lemma 2.6 we know that $V_{L_\infty}(r) = \min(1, (2r)^d)$. The minimum is necessary as the volume cannot grow larger than 1. However, we can for now assume that $k_{(u,v)} < 1/2$, as otherwise this would lead to a complete graph. With this assumption we do not need the minimum and get:

$$\mathbb{E}[deg(v)] = \sum_{\substack{u \in V \\ v \neq u}} V_{L_\infty}(k_{(u,v)}) = (n-1) \cdot V_{L_\infty}(\frac{\tau}{n}) = (n-1) \cdot \left(\frac{2\tau}{n}\right)^d.$$

We used the formula from the last chapter in the first step, the fact that $k_{(v,u)}$ is constant in the second and inserted the formula for the volume in the last step. This formula can be solved for τ and we obtain $\tau = (\frac{\mathbb{E}[deg(v)]}{n-1})^{1/d} \cdot (n/2)$.
Let us now consider a κ-GIRG for an arbitrary BDF κ but still with weights all equal to 1. We repeat the same procedure as for L_∞-GIRGs, where we denote p_κ as the polynomial describing the volume for a radius in $[0, \frac{1}{2}]$ and again assume that $k_{(u,v)} < 1/2$.

$$\mathbb{E}[deg(v)] = \sum_{\substack{u \in V \\ v \neq u}} V_\kappa(k_{(u,v)}) = (n-1) \cdot V_\kappa(\frac{\tau}{n}) = (n-1) \cdot p_\kappa(\frac{\tau}{n}).$$

Solving this formula for τ is equivalent to finding the roots of $p_\kappa(\frac{\tau}{n}) - \frac{\mathbb{E}[deg(v)]}{(n-1)}$. From Lemma 2.6 we know that p_κ is of degree d. For BDFs acting on at least five dimensions, we therefore cannot give a closed formula for τ, as this would contradict the theorem of Abel-Ruffini [Ayo80]. Still we can use numerical methods to get an approximation of τ, which in practice is completely sufficient.

Arbitrary weights: Let us now assume weights following a power-law distribution. Because each vertex has a different weight, we cannot consider a single vertex anymore. Instead, we need to consider average of all expected degrees. We write $\mathbb{E}[deg(G)]$ to denote the expected average degree of a graph G and get:

$$\mathbb{E}[deg(G)] = \frac{1}{n} \cdot \sum_{v \in V} \mathbb{E}[deg(v)] = \frac{1}{n} \cdot \sum_{v \in V} \sum_{\substack{u \in V \\ v \neq u}} V_\kappa(k_{(u,v)}).$$

The difference to the previous case where we assume constant weights is that now for some vertex pairs $k_{(u,v)} > 1/2$. This means the volume is $V_\kappa(k_{(u,v)}) = p_\kappa(\min(1/2, k_{(u,v)}))$. The minimum prevents us from simplifying the double sum, meaning that we cannot simply approximate the roots like we can in the constant case. The same problem has already been observed by Bläsius et al. [Blä+22] for L_∞-GIRGs. The solution they came up with consisted in computing the average degree for some threshold constant τ, by first ignoring the minima and then subtracting the error by considering each pair of vertices (u, v) with $k_{(u,v)} \geq \frac{1}{2}$. Because $\mathbb{E}[deg(G)]$ is monotone in τ, a binary search can then be performed on τ to approximate it.

Binary search on $\mathbb{E}[deg(G)]$. We first look at how, given a threshold constant τ, we can compute the expected average degree $\mathbb{E}[deg(G)]$ in linear time. Let $p_\kappa : \mathbb{R} \to \mathbb{R}$ be the polynomial describing the volume V_κ for a radius in $[0, \frac{1}{2}]$. Using p_κ we compute the expected average degree by using $p_\kappa(k_{(v,u)})$ also for $k_{(v,u)} > \frac{1}{2}$ and then subtract the error:

$$\mathbb{E}[deg(G)] = \frac{1}{n} \cdot \underbrace{\sum_{v \in V} \sum_{\substack{u \in V \\ v \neq u}} p_\kappa(k_{(v,u)})}_{=: Q_{over}} - \underbrace{\sum_{(u,v) \in E_{err}} p_\kappa(k_{(v,u)}) - 1}_{=: Q_{err}}$$

with $E_{err} = \{(u, v) \in V^2 | u \neq v \text{ and } k_{(v,u)} > \frac{1}{2}\}$ being the pairs of vertices having a threshold larger then $\frac{1}{2}$. Note that the subtracting 1 in Q_{error} is necessary as the probability of those edges to be sampled is 1.
With the minimum eliminated, the double sum Q_{over} can now be simplified. First, we consider the double sum over the threshold $k_{(u,v)}$ to the power of an exponent $i \in \mathbb{N}$:

$$\begin{aligned}
\sum_{v \in V} \sum_{\substack{u \in V \\ v \neq u}} k_{(v,u)}^i &= \sum_{v \in V} \sum_{\substack{u \in V \\ v \neq u}} \tau^i \left(\frac{w_u w_v}{n}\right)^{\frac{i}{D_v(\kappa)}} = \sum_{v \in V} \tau^i \left(\frac{w_v}{n}\right)^{\frac{i}{D_v(\kappa)}} \cdot (W^{(i/D_v(\kappa))} - w_v) \\
&= \tau^i \frac{1}{n^{i/D_v(\kappa)}} \cdot \sum_{v \in V} w_v^{\frac{i}{D_v(\kappa)}} \cdot (W^{(i/D_v(\kappa))} - w_v) \\
&= \tau^i \frac{1}{n^{i/D_v(\kappa)}} \cdot (W^{(i/D_v(\kappa))^2} - W^{(2i/D_v(\kappa))})
\end{aligned}$$

where we write $W^{(a)}$ to denote $\sum_{u \in V} w_u^a$ for better readability. Given the coefficient representation of p_κ as $p_\kappa(\tau) = \sum_{i=0}^{d} a_i \cdot \tau^i$ we get:

$$\begin{aligned} Q_{over} = \frac{1}{n} \cdot \sum_{v \in V} \sum_{\substack{u \in V \\ v \neq u}} p_\kappa(k_{(v,u)}) &= \frac{1}{n} \cdot \sum_{v \in V} \sum_{\substack{u \in V \\ v \neq u}} \sum_{i=0}^{d} a_i \cdot k_{(v,u)}^i \\ &= \frac{1}{n} \cdot \sum_{i=0}^{d} a_i \sum_{v \in V} \sum_{\substack{u \in V \\ v \neq u}} k_{(v,u)}^i \\ &= \frac{1}{n} \cdot \sum_{i=0}^{d} \tau^i \cdot \underbrace{a_i \cdot \frac{1}{n^{i/D_v(\kappa)}} \cdot (W^{(i/D_v(\kappa))^2} - W^{(2i/D_v(\kappa))}.)}_{\text{independent from } \tau} \end{aligned}$$

In the second step, we substituted the coefficient notation of p_k, in the third step, we brought the double sum into the equation, and in the final step, we applied the previously stated equation. We see that Q_{over} is a polynomial of τ. The coefficients can therefore be computed in $O(|V| \cdot d)$ once, and we can then evaluate Q_{over} for any threshold constant in $O(d)$. Next, we need to compute Q_{err}. This can be done by first sorting the weights and using the two pointer technique to find all vertex pairs with a threshold larger than $\frac{1}{2}$. As the weights follow a power-law distribution we can sort the weights in expected $O(|V|)$ using a modified version of bucket sort. After pre-computation we can then get query E_{err} and compute Q_{err} in $O(1 + |E_{err}|)$.

The total runtime of the binary search is then $O(|V|)$ for the pre-computation and $O(d + |E_{err}|)$ per step. The runtime of the binary at each step therefore heavily depends on the size of E_{err}, which grows for a larger threshold constant. Consequently, we are interested in a lower bound estimate for τ to be used as a starting value of the binary search. We obtain this by considering an upper estimate of the expected average degree that can be solved for τ.

Lower bound of the threshold constant. We get an upper estimate of the expected average degree of a κ-GIRG by considering the min-max set $\mathcal{S}$ of the BDF κ. More precisely, we consider the volume of each max set separately. We then get an upper estimate of the expected average degree by using the fact that the volume of a d-dimensional L_∞-BDF is $V_\infty(r) = \min(1, (2r)^d)$:

$$V_\kappa(r) \leq \sum_{S \in \mathcal{S}} \min(1, (2r)^{|S|}) \leq \sum_{S \in \mathcal{S}} (2r)^{D_v(\kappa)} = L_c(\kappa) \cdot (2r)^{D_v(\kappa)} \qquad r \in \mathbb{R}_{\geq 0}.$$

Where we use the equation for the volume in the first step, in the second step we can upper estimate the volume using the fact that that all max sets have a size of at least $D_v(\kappa)$ a remove the minimum as $(2r)^{D_v(\kappa)}$ is larger 1 for any $r > 1/2$. In the last step we then use the fact that $|\mathcal{S}| = L_c(\kappa)$. Using this upper estimate, and the known equation for $k_{(u,v)}$ in the double sum, we get:

$$\mathbb{E}[deg(G)] \leq \frac{1}{n} \cdot \sum_{v \in V} \sum_{\substack{u \in V \\ v \neq u}} L_c(\kappa) \cdot (2 \cdot k_{(u,v)})^{D_v(\kappa)} \leq \frac{1}{n^2} L_c(\kappa) \cdot (2\tau)^{D_v(\kappa)} \cdot (W^2 - W^{(2)}).$$

By solving this for τ we get $\tau \geq \frac{1}{2} \left(\frac{\mathbb{E}[deg(G)] \cdot n^2}{L_c(\kappa) \cdot (W^2 - W^{(2)})} \right)^{1/D_v(\kappa)}$.

5 Optimizing the Computation

In the previous chapter, we saw how BDF-GIRGs can be sampled as a combination of multiple L_∞-GIRGs where the runtime is affected by the computational depth with an exponential factor. In Chapter 3, we also observed that this exponential factor is likely unavoidable for arbitrary distributions of positions. However, in this chapter, we introduce an approach that, based on the assumption of a uniform distribution of the positions, can significantly reduce the computational effort. This is achieved by simplifying the BDF such that the graph we try to sample is a sub-graph of the graph sampled using the simplified BDF.

We first show an example to clarify the idea of the approach. Then, we discuss some properties the simplified BDF must have. Based on that, we give a dynamic programming to construct such a simplified BDF and discuss the difficulties that come with that.

5.1 Example

Consider the following BDF $\kappa : \mathbb{T}^d \to \mathbb{R}_{\geq 0}$ for $d \in \mathbb{N}$:

$$\kappa(x) = \min(x_1, \max((x_i)_{i \in [2,d]}))$$

Using the known algorithm, computing a κ-GIRG would take $O(n \cdot \rho^d)$, assuming a linear amount of edges and ρ is the base of the exponential time caused by the dimension. This is due to the computation of $\max((x_i)_{i \in [2,d]})$ which dominates the runtime. However, after a closer look at the sampled L_∞-GIRGs, we call them G_1 and G_2, for the first and second max-term, respectively, we notice that:
(1) Most edges are sampled in G_1. The intuitive reason is that two vertices have to be close only in one dimension to be adjacent in G_1 while in G_2, this must be the case in $d-1$ dimensions.
(2) The computational effort is dominated by the sampling of G_2, as G_1 can be sampled in linear time.
Now consider following BDF $\kappa' : \mathbb{T}^2 \to \mathbb{R}_{\geq 0}$:

$$\kappa'(x) = \min(x_1, x_2)$$

The sampling of a κ'-GIRG is both linear in the number of vertices and dimensions as κ' is the 2-dimensional minimum component distance. This is, unsurprisingly, significantly faster than for sampling a κ-GIRG. If we now compare κ with κ' we notice that $\kappa'((x_1, x_2)) \leq \kappa(x)$ for any $x \in \mathbb{T}^d$. This means that if for the same weights and positions we sample a κ-GIRG and a κ'-GIRG, where we use only the two first dimensions of the positions for the κ'-GIRG, the edge set of the κ'-GIRG is going to be a superset of the edge set of the κ-GIRG. Ultimately, we ask ourselves how much more edges have been sampled in the κ'-GIRG. This can be approximated as twice the number of edges sampled in the κ-GIRG as both generated the same number of edges in the L_∞-GIRG acting on x_1. We can therefore sample a $kappa'$-GIRG instead of a $kappa$-GIRG, potentially need to check twice the number of edges, but saving the crucial factor ρ^d and therefore improving the runtime significantly.

5.2 Conditions for a simplified BDF

We now formalize the described idea and look at the conditions a simplified $\kappa' : \mathbb{T}^{d'} \to \mathbb{R}_{\geq 0}$ of a BDF $\kappa : \mathbb{T}^d \to \mathbb{R}_{\geq 0}$ for $d, d' \in \mathbb{N}, d' \leq d$ must satisfy to be used like in the example.

1. For every $x \in \mathbb{T}^d$, $\kappa'((x_1, ..., x_{d'})) \leq \kappa(x)$. This condition makes sure that the sampled edge set for a κ'-GIRG is a superset of the edges sampled for an according κ-GIRG.

2. While it should sample a super-set of edges, there shouldn't be "too much" more edges. I.e., there must exist a constant $c(\kappa) > 0$ such that for the edges sets E of the κ-GIRG and E' of the κ'-GIRG: $\mathbb{E}[|E'|] \leq c(\kappa) \cdot \mathbb{E}[|E|]$.

3. While we do not require from κ' to be a BDF, we want it to have a min-max set representation, so that we can use it for the sampling of edges in the same way we do with BDFs.

4. Finally, we require κ' to be optimal i.e., for any possible simplifications of κ aim to find the one with both the shortest min-max set and max-sets of minimal size. We latter show that for a constant average degree those conditions do not contradict i.e., we do not need to chose between a lower length or a lower depth.

At first glance, it may not be clear how those conditions can be combined to obtain a formal construction rule for κ'. We start by considering the first and second condition. Like before we consider the volume of the κ', which must always be larger than the one of κ, but only by a constant factor. Because we do not require κ' to be a BDF, but simply require it to have a min-max set representation, we first examine the asymptotic behavior of the volume of a min-max set. This can be done quite intuitively and makes clear how the construction of κ' is going to be done.

5.3 Volume of a min-max set

From Lemma 2.5 we know that the volume of a BDF mainly depends on its volumetric depth. If we think of a BDF in the representation given by the min-max i.e., max-terms enclosed by a min-term, the volumetric depth corresponds to the size of the shortest max-term. That the volume mainly depends on this max-term also seems to be quite logical as it needs the fewest conditions to be satisfied to have an edge sampled. We show that this is also valid for any min-max set even if it is not induced by a BDF. Let $\kappa' : \mathbb{T}^d \to \mathbb{R}_{\geq 0}$ have min-max set $\mathcal{S} = \{S_i \subseteq [d] | i \in \{1, ..., L_c(\kappa')\}\}$ for dimension $d \in \mathbb{N}$. Although κ' is not BDF we use $L_c(\kappa')$ for the size of the min-max set $|\mathcal{S}|$ and $D_v(\kappa)$ to describe the size of the smallest max-set. This facilitates the intuition as the length and depth have a very similar influence on the volume. The volume of κ' can be described as the sum of the volumes of every max-term induced by $S \in \mathcal{S}$ minus the intersections that some max-terms might have. We show that the volume of those intersections is negligible for decreasing radius, by considering the intersection between every pair of max-term:

$$\sum_{i=1}^{L_c(\kappa')} \sum_{j=i+1}^{L_c(\kappa')} (2r)^{|S_i|+1} \leq \sum_{i=1}^{L_c(\kappa')} \sum_{j=i+1}^{L_c(\kappa')} (2r)^{D_v(\kappa')+1} \leq (2r)^{D_v(\kappa')+1} \cdot L_c(\kappa')^2 \qquad \text{for } r > \frac{1}{2}, S_i \in \mathcal{S}$$

We used the fact that the intersection of two max-sets S_i and S_j is the max-set $S_i \cup S_j$. Because two max-sets always differ in at least on dimension we can estimate its size with $|S_i \cup S_j| \leq |S_i| + 1$. Due to the exponent $D_v(\kappa) + 1$, the volume of κ' is asymptotically dominated by the smallest max-term. This shows, that the simplified version κ' of a BDF κ must have the same volumetric depth, otherwise the ratio between their volumes can grow arbitrary for decreasing radius.

We finish by giving an estimate by which factor the volume of κ' and κ with equal volumetric depth differs:

$$\frac{V_{\kappa'}(r)}{V_\kappa(r)} \leq \frac{\sum_{i=1}^{L_c(\kappa')} (2r)^{|S_i|}}{(2r)^{D_v(\kappa)}} \leq \frac{L_c(\kappa)(2r)^{D_v(\kappa)}}{(2r)^{D_v(\kappa)}} = L_c(\kappa').$$

We have estimated $V_{\kappa'}(r)$ from above by ignoring the intersections and then using $|S_i| \geq D_v(\kappa')$. $V_\kappa(r)$ is estimated as the volume of the smallest max-term, ignoring all others.

5.4 Simplifying a min-max set

From the previous section we know that the min-max set of a simplified κ' must have the same volumetric depth as the original min-max set of κ, to not sample too much more edges. However, as the computational depth and length are the factors which are affecting the runtime, we can minimize those as long as we let the volumetric depth unchanged. This means that we can remove dimensions from all max-sets larger than the smallest, till all max sets have the same size. We can do this in different ways. The naive one is to just remove arbitrary dimensions from each max-set till all have size $D_v(\kappa)$. This approach can already reduce the runtime needed for sampling significantly, as $D_c(\kappa)$ might be much larger then $D_v(\kappa)$. The downside is that the length, which also might grow exponentially, is not necessarily reduced as much as it could be.

Example: To better understand consider this BDF, with dimension $d \geq 4$:

$$\kappa(x) = \min(x_0, \max(x_1, \min((x_i)_{i \in [2,d)})))$$

Having the min-max set:

$$\{\{0\}, \{1\} \times [2, d)\} \qquad \text{with length } L_c(\kappa) = d - 1$$

We can simplify this BDF in two ways, among others. The first is to remove every dimension but the first two. In the second we leave one of the dimension 0 to $d - 2$, in each set:

$$1.\ \{\{0\}, \{1\}\} \qquad \text{or} \qquad 2.\ \{[0, d-1)\}$$

We see that depending on the approach chosen, the runtime varies by a factor of d. It is therefore a desirable property to shorten the max-sets such that their length is reduced as much as possible, in contrast to removing arbitrary dimensions. Unfortunately, the problem of removing dimensions from the max-sets such that the resulting min-max set has minimal length, is NP-Complete. We show a reduction from Hitting Set which is known to be NP-Complete [Kar10].

Definition 5.1 (Hitting Set): *The decision problem is defined as:*

Given: $S_1, ..., S_n \subseteq T$, *and* $k \in \mathbb{N}$ *for any set* T *with* $n \in \mathbb{N}$.

Determine: *Does there exists a* $H \subseteq T$ *with* $|H| \le k$, *such that* $H \cap S_i \neq \emptyset$ *for every* $i \in \{1, ..., n\}$.

H *is then called the hitting set.*

Lemma 5.2: *Let* $\mathcal{S} := \{S_1, ..., S_n\}$ *be a min-max set, and* $k \in \mathbb{N}$ *arbitrary. Deciding whether there is a way to shorten* $\mathcal{S}$ *to have size less or equal to* k', *is NP-Complete.*

Proof. For a reduced set $\mathcal{S}'$ we can verify if for each $S' \in \mathcal{S}'$ there exist a $S \in \mathcal{S}$ such that $S' \subseteq S$. This can be done in polynomial time.

We show a reduction from Hitting set. Let $S_1, ..., S_n \subseteq T$ and $k \in \mathbb{N}$ be an instance of Hitting set. WLOG let $T \subseteq \mathbb{N}$. We define $\mathcal{S} := \{\{0\}, S_1, ..., S_n\}$ and $k' = k + 1$.
Let the hitting set instance be a yes-instance, and H the hitting set of size k or less.
$\mathcal{S}' := \{\{h\} | h \in H\} \cup \{\{0\}\}$ is a valid min-max set with $|\mathcal{S}| = |H| + 1 \le k$.
Let $\mathcal{S}'$ be shortened set, with $|\mathcal{S}| \le k + 1$. Because of $\{0\} \in \mathcal{S}$, every set in $\mathcal{S}'$ contains one element $H := \{s | \{s\} \in \mathcal{S}\} \backslash \{0\}$ is a hitting set. ∎

In practise, we can afford the resulting exponential runtime if the BDF is small. For larger BDFs, however, we use an a dynamic program to approximate a shortening.

5.4.1 Approximating the optimal shortening

Recall that a BDF is recursively defined as a binary tree where each leaf is a dimension and every inner node is either a max or min node. The idea of this dynamic program is to remove parts of this tree, such that the volumetric and computational depth are identical and the length is minimized as much as possible. While doing so, we must make sure that the BDF we obtain is smaller i.e., $\kappa'(x) \le \kappa(x)$ for any $x \in \mathbb{T}^d$. We refer to that as the first condition (discussed in Section 5.2).

First, we need to understand what it means to remove a part of the binary tree and in which cases we can do so without violating the first condition. For a BDF κ, with comprising BDFs κ_1 and κ_2, we say that we remove the sub-tree of κ_1, if we replace κ by κ_2. All dimensions κ_1 is acting one, are then removed. We can only remove a sub-tree from a maximum node, otherwise violating the first condition. For instance, removing κ_1 from $\kappa(x) = \max(\kappa_1, \kappa_2)$ can be done as $\kappa(x) \ge \kappa_2(x)$ because of the maximum. For $\kappa(x) = \min(\kappa_1, \kappa_2)$ removing κ_1, however, would violate the first condition. This is due to $\kappa(x) \le \kappa_2(x)$ for any $x \in \mathbb{T}^d$. We can now describe our problem as DP. For each recursively defined BDF we consider the minimal length that can be achieved when reducing the volumetric depth by $n \in \{0, ..., D_v(\kappa)\}$ and the computational depth by $n + D_c(\kappa) - D_v(\kappa)$. We obtain the shortening of a BDF by choosing $n = 0$, resulting in the same volumetric, but reduced computational depth. We denote the length of κ shortened by n as $C_\kappa[n]$.
For a one-dimensional BDF κ removing it leads to a length of 0 and keeping it to 1.

$$C_\kappa[0] = 1 \qquad C_\kappa[1] = 0$$

For an outer-min BDF κ with comprising BDFs κ_1 and κ_2, let WLOG $D_v(\kappa_1) \ge D_v(\kappa_2)$. As we want the result to have equal volumetric and computational depth, we consider the optimal solution for a depth of $D_v(\kappa_2) - n$:

$$C_\kappa[n] = C_{\kappa_1}[n + D_v(\kappa_1) - D_v(\kappa_2)] + C_{\kappa_2}[n]$$

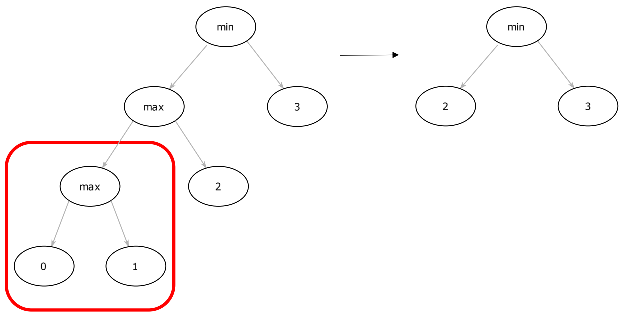


Figure 5.1: A representation of $\min(\max(1,2,3),0)$ as binary tree where we remove the dimensions 1 and 2. We obtain $\min(2,3)$.

For an outer-max BDF κ we need to reduce the volumetric depth of the comprising BDFs κ_1 and κ_2 by a total of n. The optimal solution consists of trying out every combination.

$$C_\kappa[D_c(\kappa)] = 0 \qquad C_\kappa[n] = \min\{ \bigcup_{i \in \{n - D_v(\kappa_2), \dots, D_v(\kappa_1)\}} \max(C_{\kappa_1}[i], 1) \cdot \max(C_{\kappa_2}[n-i], 1)\}$$

The optimal length of a shortened BDF κ is then $C_\kappa[0]$. We can reconstruct the solution by saving the optimal indices used at every outer-max BDF. Due to a BDF having $O(d)$ inner BDFs each having volumetric depth up to $O(D_c(\kappa))$, with up to $O(D_c(\kappa))$ possible combinations at every outer-max BDF, the total runtime of the DP is $O(d \cdot D_c(\kappa)^2)$.

Note that this approximation does not provide an optimal shortening for every BDF, even if we assume the shortened min-max set to be induced by a BDF. This is because it is not always optimal to remove a dimension completely. Consider the following BDF and its min-max set as an example:

$$\max(\min(\max(x_0, x_1), x_2), \min(x_3, x_4)) \quad \text{and} \quad \{\{0,1,3\}, \{0,1,4\}, \{2,3\}, \{2,4\}\}.$$

The presented DP would remove the dimensions 0 or 1, resulting in the min-max set $\{\{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}\}$. We can see that it is however better to remove the dimension 3 and 4 in the first two max sets and get $\{\{0,1\}, \{2,3\}, \{2,4\}\}$. Even thought we did not remove any dimension entirely, the resulting min-max set is smaller and also induced by BDF $\min(\max(x_0, x_1), \max(x_2, \min(x_3, x_4)))$.

5.5 The Final Sampling Algorithm

The final algorithm to sample BDF-GIRGs now consists of first computing a simplified κ' of the given BDF κ. Depending on the size, this can be done either using the dynamic program or, if the BDF is small enough, the exponential time algorithm. We then use the min-max set of κ' to compute $L_c(\kappa')$ L_∞-GIRGs, each having the dimension $D_v(\kappa) = D_c(\kappa') = D_v(\kappa')$. We then filter the set of sampled edges to obtain only those that would have been sampled in the κ-GIRG This leads to an optimized time complexity of the algorithm of $O(L_c(\kappa') \cdot n \cdot \rho^{D_v(\kappa')})$ for $\rho > 1$ being the exponential factor that affects the runtime of L_∞-GIRGs.

6 Evaluation

In this chapter, we first discuss some implementation details of the BDF-GIRG algorithm described earlier. We then investigate the quality of the threshold estimator described in Chapter 4 and verify that the degree distribution does indeed follow a power law. Later, we empirically analyse three properties of the generated graphs: the size of the largest component, the clustering coefficient and the diameter. For a better overview, in Section 6.2 we divide BDFs into three categories that have been observed to have similar properties and also discuss the experimental setup used. Finally, we briefly examine the runtime of the generator.

6.1 Implementation Details

We implemented the algorithm as described in Chapter 5. The implementation is an open-source module and comes with a basic command line interface.[1] The L_∞-GIRG sampling algorithm our algorithm is based on was implemented by Bläsius et al.[Blä+22] and provided as an open source C++ library.[2] The C++ implementation is extended to work with min-max sets. The sampling algorithm gets a shortened min-max set which is used to sample L_∞-GIRGs and the original min-max set which is used to check every edge. The threshold estimator is extended such that it gets an arbitrary polynomial describing the volume. Everything else, e.g., BDFs, their shortening and the CLI are implemented in Python to enable easy usage. The python wrapper used for the C++ code is also based on an existing python wrapper for L_∞-GIRGs.[3] Beside details of the algorithm discussed earlier, the implementation is also adjusted in three ways we have not discussed yet:

Approximation of the volume. We saw in Lemma 2.6 that the volume of a BDF can be described by a polynomial with a degree equal to the dimension. For higher dimensions, this leads to both Q_{over} and Q_{error} (see Section 4.3) to have very high values, which can result in numerical cancellation when subtracted. Therefore, it is sensible to simplify the polynomial to have a lower degree. The simplest approach is to ignore the intersections between max-sets, but it leads to a relatively high error (10-15%) between the desired and gotten average degree, even in larger graphs. A practically reasonable alternative is to consider only the intersection of the highest min-nodes in the binary tree of the BDF. For the BDFs we examined, the resulting error was lower then the one resulting from the binary search. (Which aborts if the estimated threshold differs from the desired by less then 0.1).

Find an optimal BDF shortening. As discussed in Chapter 5, given a min-max set, finding an optimal shortening is NP-Complete. For smaller BDFs, where the number of possible combinations to check do not exceed 2^{17}, we compute the optimal shortening. For any larger BDF we use the dynamic program approach presented in Section 5.4.1.

[1] https://github.com/Maxime-RA/bdf-girg
[2] https://github.com/chistopher/girgs
[3] https://github.com/gavento/girg-sampling

Increasing runtime for lower thresholds. An unforeseen behavior of the used L_∞-GIRG algorithm is that the runtime can significantly increase if it is generated with a very low average degree. The origin of this behavior lies in the recursive partitioning of the ground space, where the recursion depth depends on the smallest possible threshold. For lower average degrees, this threshold is smaller, resulting in a higher recursion depth. In these cases, it can be more efficient to terminate the partitioning at a lower depth and compare "a few more" pairs of vertices. In practice, we found out that the runtime increases if the average degree in every L_∞-GIRG is lower than five times the dimension. This is a limit that is quickly reached if the BDF-GIRG consists of multiple L_∞-GIRGs. To counteract this L_∞-GIRGs are not generated with an average degree lower than that. Instead, we compute two thresholds. One larger for the L_∞-GIRG generation and the "correct one" that we use when we filter the edges.

6.2 BDFs & Configuration

6.2.1 Used BDFs

For each property investigated, we tried various BDFs, of different length, depth and dimensions. Because sampling L_∞-GIRGs with dimensions higher than five involves considerable computational effort, we only consider the case where the depth is less or equal to five. Note that for easier notation, we write the dimensions as a number e.g., for $\kappa(x) = max(|x_0|, |x_1|)$ with $x \in \mathbb{T}^2$ we write $\max(0, 1)$. Overall, we divide BDFs into three categories that have shown to behave in similar ways. We, therefore, first look at some representatives of each of the three categories and, if necessary, examine each category in more detail.

1d-outer-min. The first category are outer-min BDFs where a single dimension is surrounded by the minimum, while all other dimensions are part of at least one outer-max BDF. Formally, we say that a BDF is 1d-outer-min if its min-max set contains exactly one max-set with size one. We can intuitively think of such BDFs as one-dimensional L_∞-GIRGs where some max-terms add extra edges. Examples of such BDFs are $min(0, max(1, 2), max(3, 4))$ or the two representatives we are using: **0**, which is just the one-dimensional L_∞-GIRG and $\mathbf{min(0, max(1, 2))}$. We can think of such graphs as one-dimensional GIRGs, where due to some max-term, a few extra edges are sampled. Note that the number of edges sampled due to the max-terms is low. Depending on the power-law exponent, the proportion ranges between 3% and less than 1% for two-dimensional max-terms. For higher dimensional max-term this value is even lower and for large enough graphs eventually reaches zero.
If we want to further investigate a behavior with more BDFs in this category, we use l max-terms each having size two or three. The BDF then has the form:

$$\underbrace{\min(0, \max(\underbrace{1, 2, \ldots}_{\text{d dimensions}}), \ldots)}_{\text{l max-terms (including 0)}} \quad \text{We write:} \quad \text{1D-OM}[D = d, L = l]$$

For instance, $\min(0, \max(1, 2))$ is then written as $\text{1D-OM}[D = 2, L = 2]$. The resulting BDFs all have volumetric depth one, computational depth d and length l.

H-SCOM. The second category is a class that is very similar to one we already know from Definition 2.3, so called SCOM-BDFs. Those are BDFs that can be written as outer-max BDFs where one of the comprising BDFs is a single dimension. By definition, SCOMs also contain

the one-dimensional BDF $|x|$. As we already assigned the one-dimensional BDF to the first category, this category contains SCOM-BDFs of dimension greater then one. We use $\mathbf{max}(\mathbf{0}, \mathbf{1})$ and $\mathbf{max}(\mathbf{0}, \mathbf{min}(\mathbf{1}, \mathbf{2}))$ as representatives of this category.
To further observe their behaviour we also define a class of BDFs having the form:

$$\max(\underbrace{0, 1, 2, \ldots,}_{d_{max}\text{ dims}} \underbrace{\min(\underbrace{3, 4, \ldots}_{l\text{ dims}}))}_{d_{min}\text{ terms}} \quad \text{We write:} \quad \text{H-SCOM}[D_{max} = d_{max}, D_{min} = d_{min}, L = l]$$

$\max(0, \min(1, 2))$ is then H-SCOM$[D_{max} = 1, D_{min} = 1, L = 2]$. The depths (volumetric & computational) are then both equal to $d_{max} + d_{min}$ and the computational length is l.

Remaining BDFs. The last category contains all remaining types of BDFs. This may sound surprising, but we observed hardly any differences in the behavior of BDFs within this category. Therefore, in the following discussions, we primarily focus on the first two categories. We use a $\mathbf{min}(\mathbf{0}, \mathbf{1})$ as representative.

6.2.2 Configuration

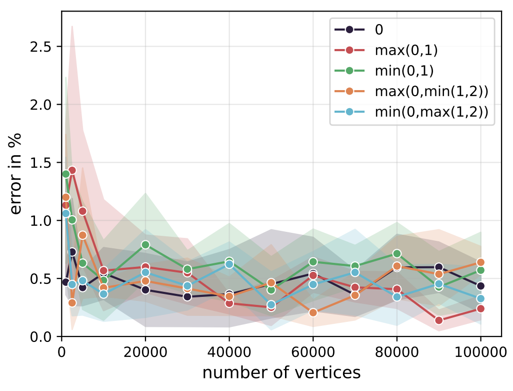
As the default configuration, we generate graphs of size $n = 2^{15}$, an average degree of 10 and a weight distribution with a power-law exponent (PLE) of 2.5. For each configuration showed, we generate five random graphs and use the average value. The variance of the values is is showed in the plots. For plots that differentiate by PLE, the average of all five BDFs (0 *(one-dimensional GIRG)*, $\max(0, 1)$, $\min(0, 1)$, $\min(0, \max(1, 2))$, $\max(0, \min(1, 2))$) is used.

6.3 Prediction of the Average Degree

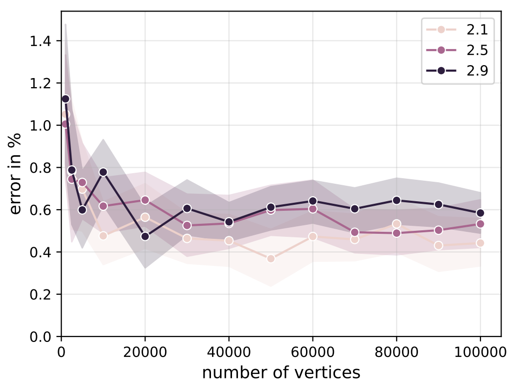
In this section, we briefly discuss the quality of the threshold estimator. For each measurement we compute the error in percent between the desired and obtained average degree (see Figure 6.1). In Figure 6.1a and Figure 6.1b, we observe that minor inaccuracies occur for smaller graphs, while for larger graphs, the error is below one percent. This can be explained by the fact that the binary search (see Section 4.3) terminates if the predicted average degree differs by less than 0.1 from the desired one. For an average degree of 10, this corresponds to the observed 1 percent. This also explains why the error decreases with an increasing average degree as it can be seen in Figure 6.1c and Figure 6.1d. In conclusion, we can state that the prediction works well, and the deviations are likely negligible for practical applications but could also be reduced by adapting the binary search.

6.4 Degree Distribution

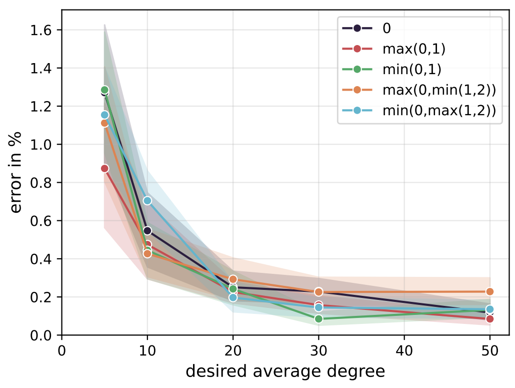
In this section, we briefly examine the degree distribution of the generated graphs. From Theorem 2.9 we know the generated graphs should be scale-free i.e., their degree distribution should follow a power-law. Moreover, the power law distribution should follow the same exponent as the weight distribution even if the exponent is greater than three, which is otherwise a restriction for some properties. As shown in Figure 6.2a, this is indeed the case, and the choice of the PLE affects the degree distribution as expected. In Figure 6.2b, we can also observe that the choice of BDF has no influence on the degree distribution.



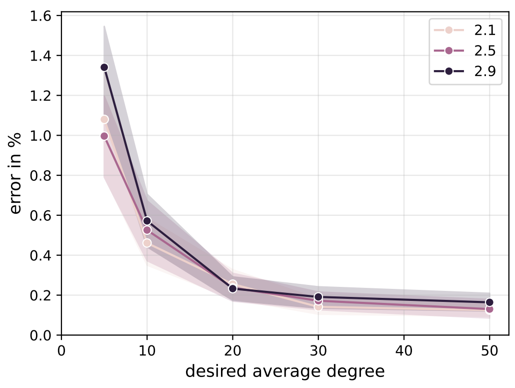
(a) for growing size (by BDF)



(b) for growing size (by PLE, avg of 5 BDFs)

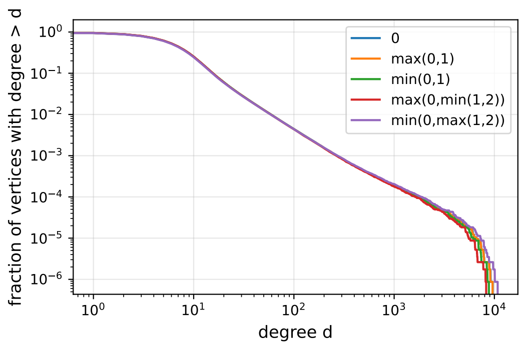


(c) for growing desired average degree (by BDF)

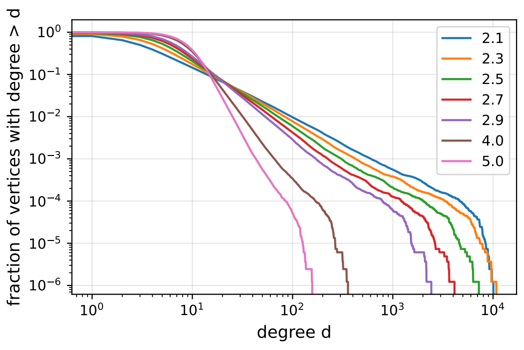


(d) for growing desired average degree (by PLE, avg of 5 BDFs)

Figure 6.1: Error in % between the desired average degree and the observed average degree.



(a) by BDF



(b) by PLE

Figure 6.2: Degree distribution of BDF-GIRGs. The distribution is plotted as a cumulative distribution function with the degree d on the x-axis and the percent of vertices having a degree larger d on the y-axis.

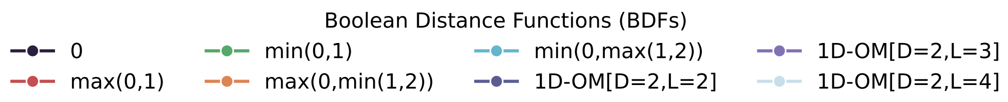
6.5 Size of the largest Component

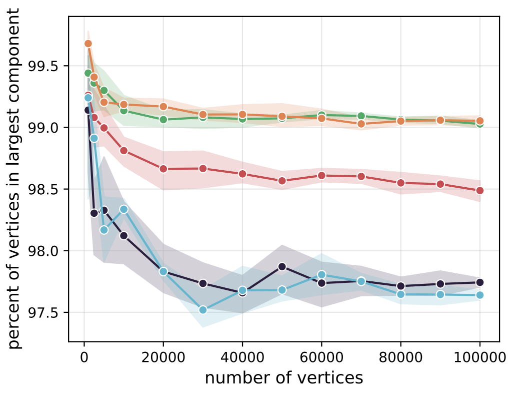
In this section, we analyse the size of the largest component for different configurations. We first take a look at the case of the PLE being between 2 and 3 and later observe the behavior for a PLE greater than 3. We test the five representatives and later look at some more results using BDF from the first category. From Theorem 2.10 we know that the largest component is of linear size if the PLE is between two and three. In Figure 6.3a and Figure 6.3b we empirically verify this theoretical result. Note that for a lower PLE the size of the largest component can still be assumed to be linear, but is significantly lower than for higher PLEs. This is due to a high number of vertices with a large weight. This, in turn, leads to fewer local connections as the high-weight vertices dominate and connect over larger distances, reducing the number of local connections. Vertices with a low weight are then more likely to have no connection at all.

Next, we analyse the behavior for an increasing PLE i.e., less vertices have a large weight and the degree distribution is therefore more homogeneous (Figure 6.3c). We can observe that BDFs in the second and third categories experience little to no influence under this kind of change. For both categories the resulting graph still features a giant component. For BDFs in the first category, however, this is not the case. We can observe a phase transition for the one-dimensional BDF as soon as the PLE gets larger then three. For other BDFs of the first category we observe a behavior that is somehow in-between (Figure 6.3d). BDFs made of max-terms with a size greater than two (recall the construction in Section 6.2) have an almost identical behavior to the one-dimensional GIRG. This is due to almost no edges being sampled due to those max-terms. Those featuring multiple max-terms with two dimensions however, interpolate well between the one-dimensional GIRG and the BDFs of the other categories. For an increasing graph size, all BDFs in the first category seem to converge against the one-dimensional case.

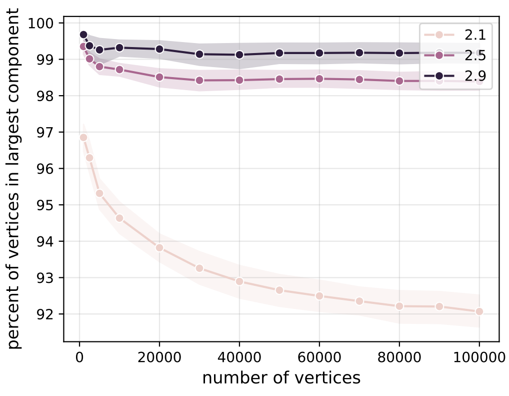
We try to explain this behavior by first noticing that for a PLE that goes towards infinity, the weights of all vertices would be equal, resulting in a random geometric graph (RGG). We think that the observed behaviour can be well explained if we think of one-dimensional GIRGs with a high PLE as one-dimensional RGGs. For those, Dall and Christensen [DC02] showed that a giant component emerges as soon as the connection threshold is greater than a certain value, which increases with decreasing dimension. If, however, the connection threshold is lower, the giant component almost immediately falls apart into multiple small components. We therefore suspect that this threshold is not yet reached at an average degree of 10. Figure 6.3f also strengthens our suspicion, as a giant component seems to emerge for an average degree exceeding 25.

The behaving of the other BDFs can be explained by separately considering the edges sampled due to the additional max-terms. For max-terms greater than two, this number is negligible and thus has little impact. In the case of max-terms with two dimensions, a small proportion (<1% per max-term for PLE=5.0 and $n = 2^{15}$) of edges is generated by these max-terms. This is sufficient to connect enough of the components that felt apart to a giant component. The more max-terms are used, the more such edges are generated and the more of the small components are connected. This proportion of edges generated due the those additional max-terms, however, is decreasing for larger graphs as their volume of the max-terms decreases faster (recall Lemma 2.5 where we saw that the volume is in $\Theta(r^{D_\nu(\kappa)})$). This explains the behavior in Figure 6.3e.

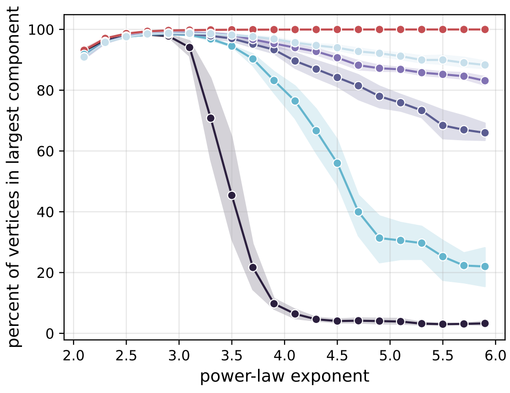




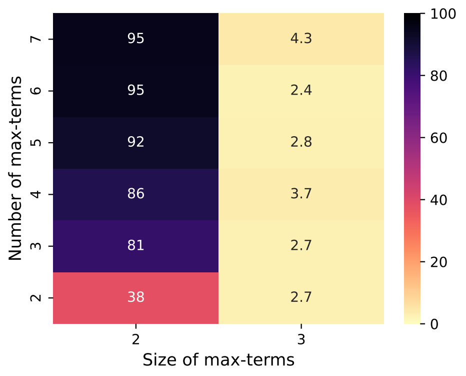
(a) for growing size (by BDF)



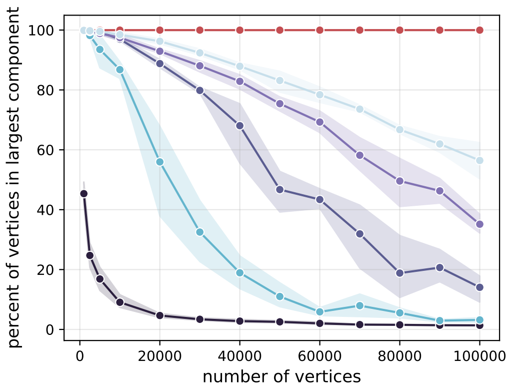
(b) for growing size (by PLE, avg of 5 BDFs)



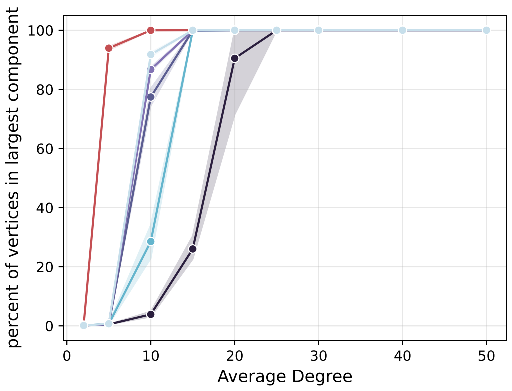
(c) for growing PLE (by BDF)



(d) 1D-OM[D=d,L=l] for $d \in \{2,3\}$ and $l \in \{2,3,4,5\}$. ($PLE = 5.0$)



(e) for growing size, by 1d-outer-min BDF and $\max(0,1)$ for comparison.($PLE = 5.0$)



(f) for growing PLE, by 1d-outer-min BDF and $\max(0,1)$ for comparison.($PLE = 5.0$)

Figure 6.3: Proportion of vertices in the largest component in %.

6.6 Clustering coefficient

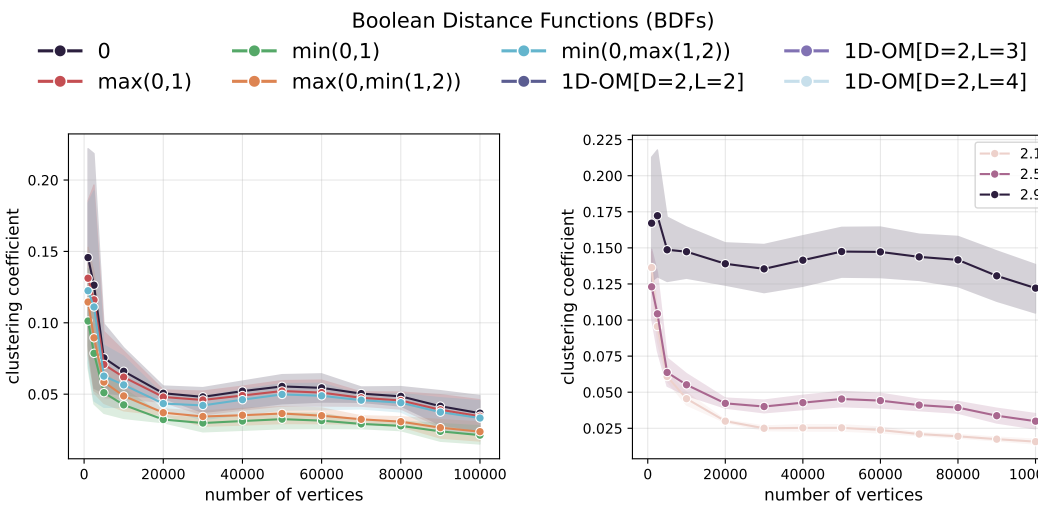
Next, we analyse the (global) clustering coefficient of BDF-GIRGs as defined in Definition 2.11. Recall that it describes the probability that two vertices with a common neighbor are also connected. In contrast to the other properties examined, this property is significantly more stable with respect to the addition or removal of a small number of edges. Thus, a few edges can substantially reduce the diameter or greatly increase the size of the giant component, while the clustering coefficient remains relatively unaffected under such changes. In Figures Figure 6.4a and Figure 6.4c, we observe that the clustering coefficient appears to depend little on the BDF, but is much more influenced by the chosen PLE, with an increase in PLE (Figure 6.4b and Figure 6.4d) also leading to an increase in the clustering coefficient. This behavior can be explained by the fact that a higher PLE leads to fewer large weights, which leads to more local connections.
This behavior can be further observed in Figure 6.4e where the clustering coefficient rises for an increasing PLE. However, in this case, a distinction between the different BDFs is evident. Unlike for the largest component and the diameter, however, the categories we previously established cannot be applied in this context. Instead, we figured out that it is more useful to think of the BDFs in the form induced by their min-max set. More precisely, we just consider the smallest max-terms, as those are the ones that lead to the most edges to be sampled. We can ignore all others as the amount of edges sampled by them is too low to influence the global clustering coefficient. We then think of each L_∞-GIRG sampled separately. The vertices that are then close to each other (using the L_∞-norm) with respect to the dimensions of one max-term have just a little probability of being close with respect to dimensions in other max-terms. If we think of vertices close to each other as “neighborhoods”, in each max-term a vertex has another neighborhood. Since the average degree remains constant, the connections of a vertex are then evenly distributed over those “neighborhoods”, effectively dividing its clustering coefficient by length of the BDF. We can observe this in Figure 6.4e by considering $\min(0, 1)$, which has a clustering coefficient half as large as the one-dimensional GIRG.
Beside this influence we can also observe the clustering coefficient to decrease for increasing dimensions. Consider the one- and two-dimensional GIRG as an example. This is consistent with the theoretical findings of Friedrich et al. [FGKS24]. They show that the clustering coefficient of GIRGs is in $\Theta((\frac{3}{4})^d)$ for PLEs greater than three. We can nicely observe this phenomenon for both the length and depth of the BDFs in Figure 6.4, where we used a BDF that has the form:

$$\underbrace{\min(\max(\underbrace{1, 2, \dots}_{\text{x dimensions}}), \dots)}_{\text{y max-terms}}$$

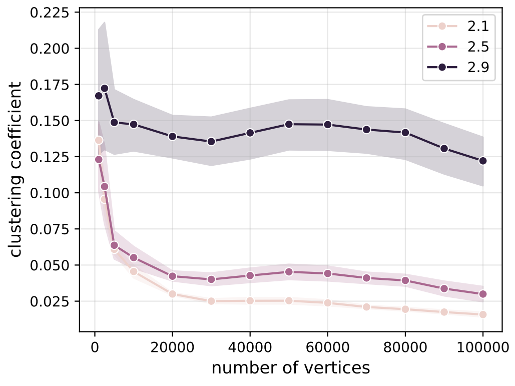
leading to a volumetric depth of x and length of y.

6.7 Diameter

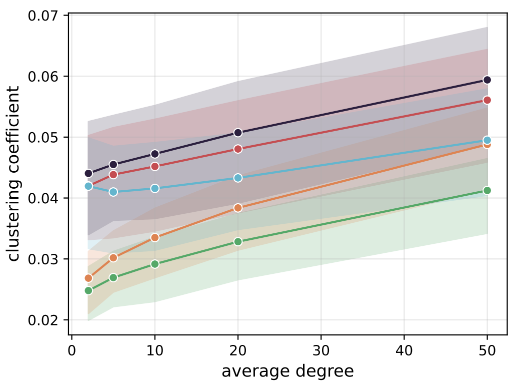
Lastly, we analyse the diameter of BDF-GIRGs. The diameter is the maximal length of all shortest paths. In a real-world setting, we can think of the diameter as the maximal number of intermediates that two arbitrary people need to be connected. If the graph features more than one connected component, we consider the largest diameter of all connected components. We know from Theorem 2.10 that if the PLE is between two and three, the diameter is at most



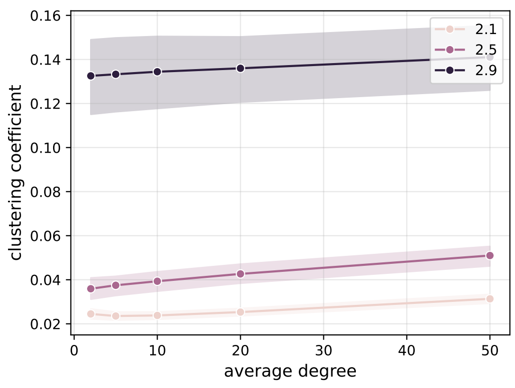
(a) for growing size (by PLE)



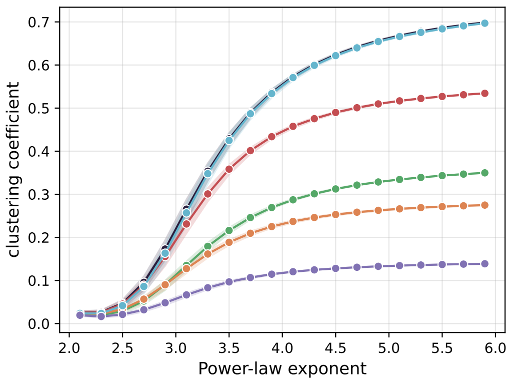
(b) for growing size (by PLE, avg of 5 BDFs)



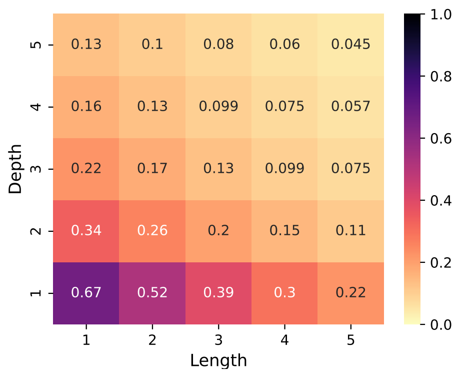
(c) for growing average degree (by BDF)



(d) for growing average degree (by PLE, avg of 5 BDFs)



(e) for growing PLE (by BDF)



(f) for BDFs constructed as described in Section 6.6 ($PLE = 5.0$)

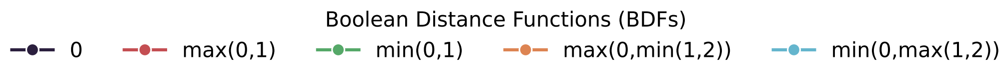
Figure 6.4: Comparison of the (global) clustering coefficient.

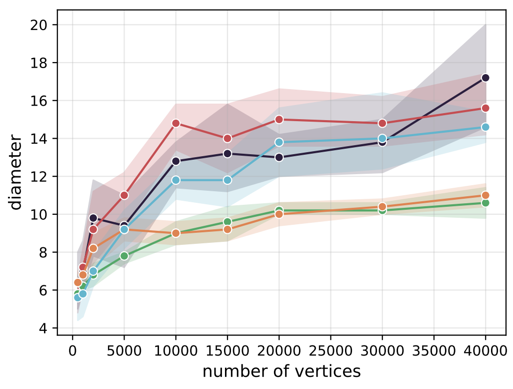
poly-logarithmic. It is difficult to verify this property empirically since the computation of the diameter, especially if the underlying space is a torus, takes much effort ([BF24], section 4.2) and the obtained values are quite low. However, it seems to hold for smaller graphs, as can be seen in Figure 6.5a and Figure 6.5b. In a similar way as with the previously examined properties, we can also observe that for a PLE between two and three, the choice of the BDF affects the diameter significantly less than the choice of the PLE. So we again look at the case of a PLE that is greater than three (Figure 6.5e and Figure 6.5f). Three distinct behaviors can clearly be observed. The first category exhibits a significantly increasing diameter, the second category appears to have a moderately growing diameter, and lastly, the third category shows little to no increase of the diameter, regardless of the PLE or graph size. It is challenging to explain those behaviors based on existing studies, as the BDFs appear to have a significant impact, but prior research has been limited to the case of distance functions induced by the Euclidean or L_∞-norm, leaving the influence of BDFs relatively unexplored. We still attempt to explain the behavior of BDFs from the first and second categories using existing knowledge about RGGs. However, it remains unclear why BDFs from the third category exhibit the observed behavior.

For BDFs in the first category, the behavior varies depending on the graph size. Initially, the one-dimensional GIRG exhibits a more rapidly increasing diameter, but for larger graphs, BDFs with several max-terms show a higher diameter (Figure 6.6b). We believe this behavior can be explained by the absence of a giant component in the one-dimensional GIRG. While the one-dimensional GIRG breaks into small components with a low diameter, the few edges generated by the max-terms in the other BDFs connect these components, significantly increasing the overall diameter. If the number of max-term further increases, however, the diameter starts decreasing again as those “extra edges” can be thought as random long-range edges in the one-dimensional GIRG which lead to a decreasing diameter (Figure 6.6a and Figure 6.6e).
For BDFs in the second category, which we know have a giant component for higher PLEs, the behavior seems to be more monotone. In Figure 6.6f we can nicely see how increasing the (both volumetric and computational) depth reduces the diameter while increasing the length also increases the diameter. The decreasing diameter with increasing depth correlates with theoretical findings made about RGGs. Friedrich et al. [FSS13] showed that for RGGs having a giant component, the diameter is in $\Theta(n^{1/d})$. This suggests that the diameter is likely not poly-logarithmic for all BDF-GIRGs for if the PLE exceeds three. However, it remains unclear why the length leads to an increasing diameter. If we think of BDF-GIRGs as a union of multiple max-GIRGs, we would expect the opposite behavior as additional edges, as seen for the first category, reduce the diameter if the graph already has a giant component. Instead, the increase in diameter is puzzling, since extra edges usually lead to a reduction in diameter.

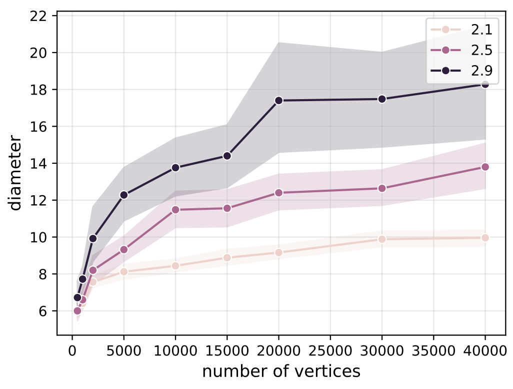
6.8 Performance

In this section, we briefly discuss the performance of our implementation. We test the runtime exclusively in single-core mode, meaning we do not use parallelization. This allows us to get more consistent measurements that depend less on background tasks. We use a system with an Intel i5-1035G4 (1.1 GHz), 8GB RAM and represent the sampled graph as an edge list. The runtime is plotted as the average sampling time per edge. We first examine the runtime behavior for an increasing number of vertices and later for an increasing average degree. Specifically, for the BDF $min(0, max(1, 2))$, we analyse the runtime of each generation

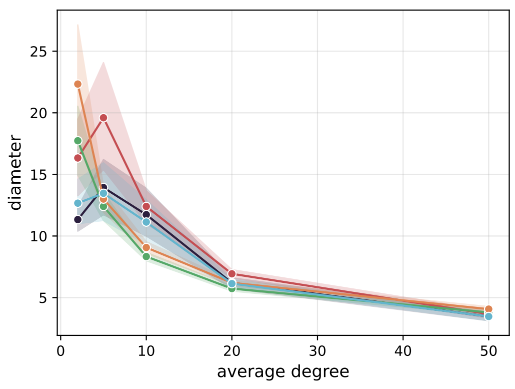




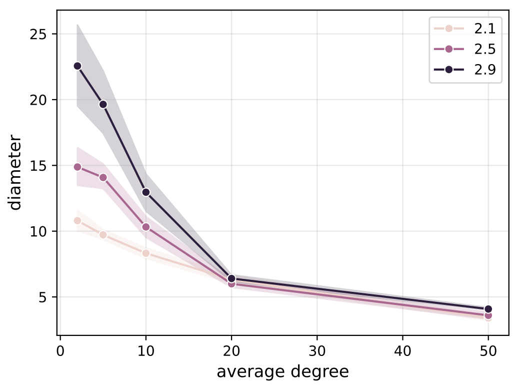
(a) for growing size (by BDF)



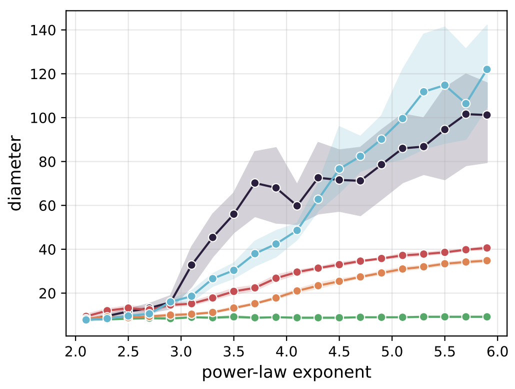
(b) for growing size (by PLE, avg of 5 BDFs)



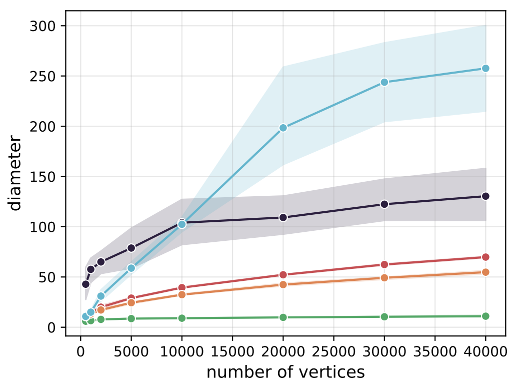
(c) for growing average degree (by BDF)



(d) for growing average degree (by PLE, avg of 5 BDFs)

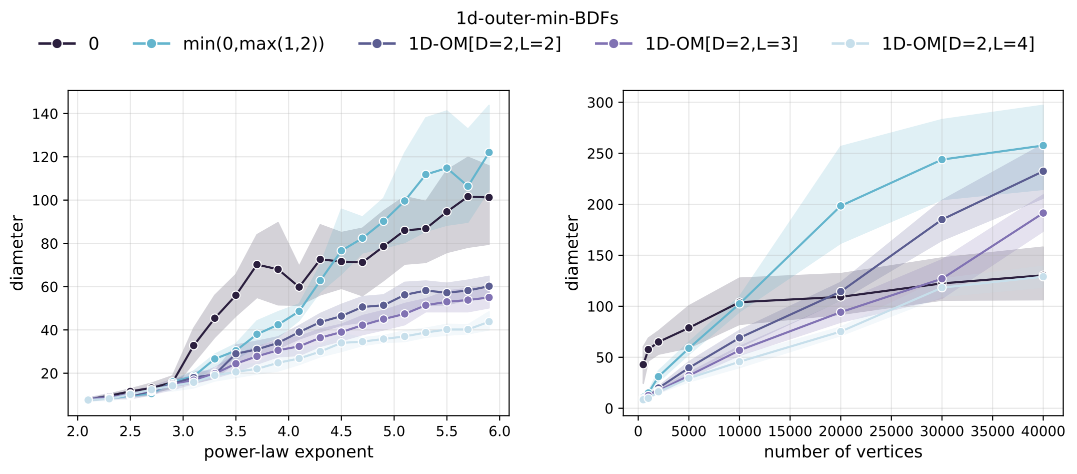


(e) for growing PLE



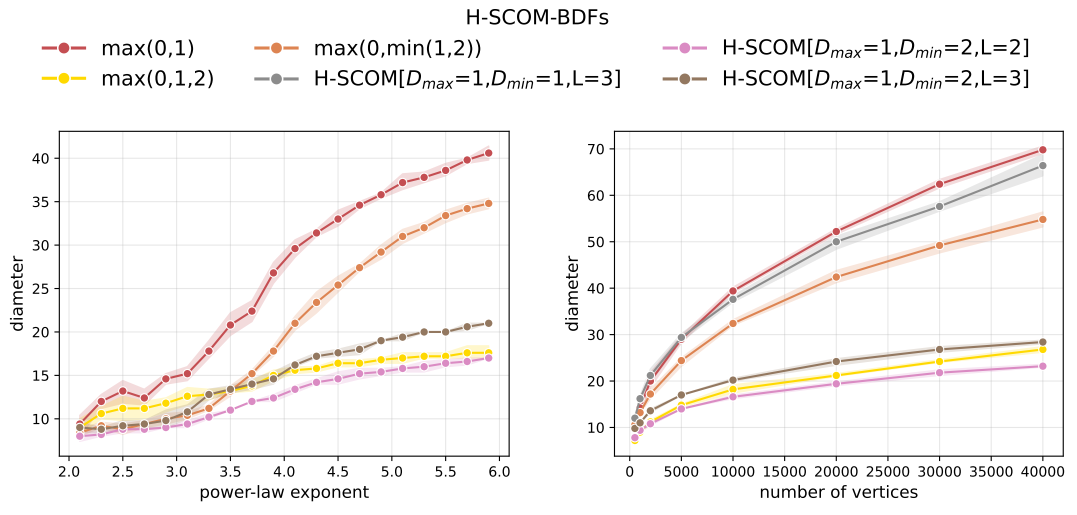
(f) for growing size (PLE = 5.0)

Figure 6.5: Comparison of the diameter for different configurations.



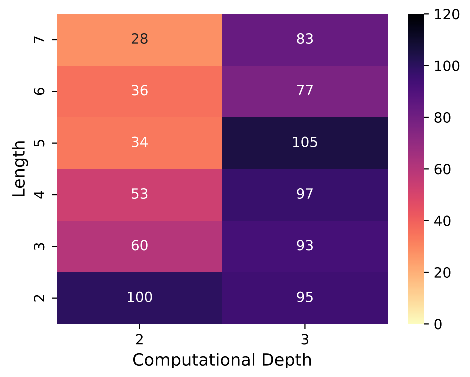
(a) for growing PLE (1d-outer-min BDFs)

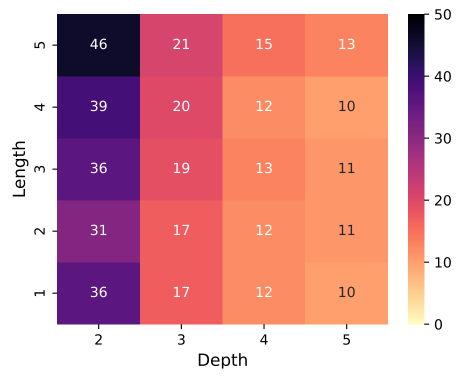
(b) for growing size (1D-OM BDFs)($PLE = 5.0$)



(c) for growing PLE (H-SCOM-BDFs)

(d) for growing size (H-SCOM-BDFs) ($PLE = 5.0$)





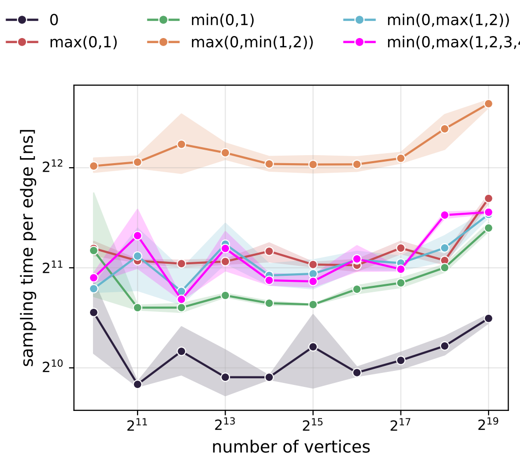
(e) 1D-OM[D=d,L=l] for $d \in \{2,3\}$ and $l \in \{2,3,4,5\}$. ($PLE = 5.0$)

(f) H-SCOM[$D_{max} = d, D_{min} = 1, L = l$] for $d \in \{2,3,4,5\}$ and $l \in \{1,2,3,4,5\}$. ($PLE = 5.0$)

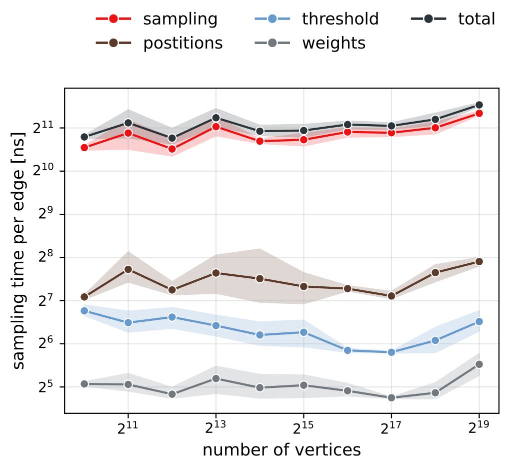
Figure 6.6: Diameter of BDFs from the first egree of ten.

steps (*generating positions & weights, estimating the threshold constant and sampling the edges*) and the total runtime of all four steps for the five BDFs used before. Additionally, we use $min(0, max(1, 2, 3, 4))$ to evaluate the optimisations done in Chapter 5.

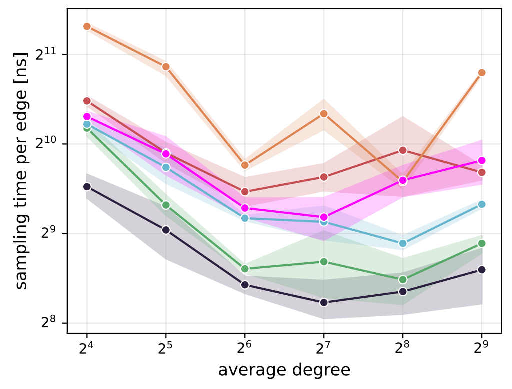
In Figure 6.7b we can see that the runtime is linear as the sampling time per edge is constant. Two interesting observations can also be made: generating positions is computationally more intensive than generating weights, despite the latter requiring the use of expensive exponential functions, which contrasts with the observations made by Bläsius et al. [Blä+22]. This suggests that the main computational overhead in generating weights and positions comes from the communication between the Python and C++ implementations. Additionally, an increase of the total runtime is noticeable for graphs with more than 2^{18} vertices in both Figure 6.7a and Figure 6.7b. Since this increase is also observed in the generation of weights and edges, it can be assumed that the system's limits are being reached at this point. In Figure 6.7b one can observe the effect of the length and depth discussed earlier. While the one-dimensional L_∞-GIRG has the lowest runtime, sampling $min(0, 1)$ takes approximately twice as long. The BDFs $min(0, max(1, 2))$ and $min(0, max(1, 2, 3, 4))$ both have similar runtime to $min(0, 1)$, although it is slightly higher due to more potential edges being checked. The highest runtime is needed for $max(0, min(1, 2))$ as it cannot be simplified and therefore two two-dimensional GIRGs must be sampled.
For an increasing average degree, we see in Figure 6.7d that the sampling time per edge first decreases and remains approximately constant for an average degree larger than 2^6. This is due to the fact that for a low average degree, the runtime is dominated by the number of vertices, therefore increasing the sampling time per edge. As generating positions and weights does not depend on the average degree, their sampling time decreases. The runtime of the threshold estimation increases due to a growing size of E_{err} (see Section 4.3). In Figure 6.7c this behavior can also be observed, although the variance of the runtime makes it more difficult. However, another interesting behavior can be observed when comparing the one-dimensional GIRG and $min(0, 1)$. For an increasing average degree, where the runtime is dominated by the number of edges sampled, their runtime gets more similar, indicating that for higher degrees the length affects the runtime less.



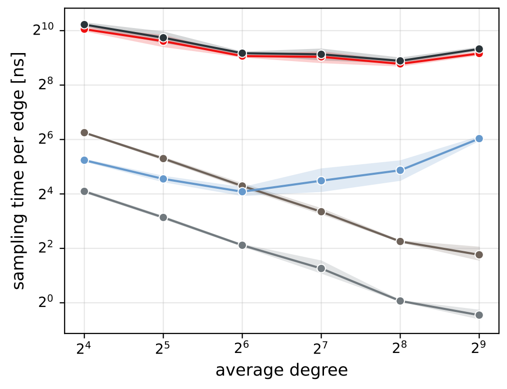
(a) for growing size (by BDF)



(b) for growing size (by step,$min(0, max(1, 2))$



(c) for growing average degree (by BDF)



(d) for growing average degree (by step, $min(0, max(1, 2))$

Figure 6.7: Performance of the BDF-GIRG sampling algorithm. We measure the sampling time per edge.

7 Conclusion

In Chapter 2 we consider a recently introduced extension of GIRGs, called BDF-GIRGs. Unlike GIRGs, BDF-GIRGs use of a wide range of distance functions consisting of an arbitrary nesting of minima and maxima of the component wise distance, referred to as BDFs. They potentially enable the modeling of more realistic graphs by better reflecting the structure of real-world graphs. Motivated by their potential use case and the lack of any (efficient) sampling algorithm for this type of graph, in Chapter 4 we develop an expected linear time algorithm to sample them. The algorithm works by bringing the BDF into a normal form, we call *min-max form*. This form represents a BDF-GIRG as a union of multiple classical GIRGs, we call L_∞-GIRGs. The main advantage of this algorithm is therefore that there is no need for a new complex data structure for its generation, as an efficient algorithm for the sampling of L_∞-GIRGs already exists. In Section 4.3 we discus why their can be no closed formula to determine the threshold constant based on a desired average degree, even for constant weights. We then modify an existing approach for L_∞-GIRGs, which consists of estimating the average degree based on a given threshold constant and then performing a binary search on it. In Chapter 5 we optimize the algorithm described by instead of sampling a very small number of edges in high-dimensional L_∞-GIRGs, sampling a superset of edges in low-dimensional L_∞-GIRGs and then filtering out any edges that were wrongly generated. This drastically reduces the runtime since the dimension affects the runtime of L_∞-GIRGs by an exponential factor. In Chapter 3 we also address the question why the runtime has this unpleasant effect. We show that if we do not require the positions of a BDF-GIRGs to be distributed evenly, under a key-assumption of the fine-grained complexity theory, known as *Orthogonal Vectors Hypothesis (OVH)*, there can be no algorithm which avoids this exponential factor, unless it has a quadratic runtime in the number of vertices (being the trivial $O(n^2)$ algorithm). Lastly, we evaluate the graphs generated by our algorithm in Chapter 6. We show that the prediction of the average degree works well, and the deviations are negligible. We also look at the degree distribution that, unsurprisingly, follows a power-law. We then further examine three key-properties of the generated graphs: the size of the largest component, the clustering coefficient and the diameter. We focus on higher power-law exponents, which yield a more homogeneous degree distribution and observe that the choice of the BDF has a significant influence on the behavior of the resulting BDF-GIRG. Based on those observation we are able to categorize BDFs into three categories that behave similarly.

7.1 Future Work

Based on the developments and findings made in this work, we roughly classify three directions that could be interesting for further research. The first concerns the theoretical foundation of our algorithm, the second focuses on its practical implementation, and the third relates to the empirically obtained results discussed in Chapter 6.

The algorithm we presented only features the threshold case of BDF-GIRGs. It is, however, realistic to assume that in real-world networks, the connection between two entities is not purely deterministic. Extending the algorithm to handle the binomial version of BDF-GIRGs is a bit more challenging, as potentially every pair of vertices might be adjacent. However, as the L_∞-GIRG algorithm of Bringmann et al. [BKL19] does support the binomial version, we assume that it can also be achieved for BDF-GIRGs. It also remains an open question whether a more efficient data structure could handle nested BDFs more effectively, as in our algorithm this might potentially lead to an exponentially increasing length.
A second, more practical question concerns the optimization of our implementation. The focus of this work was not to provide a highly efficient implementation of the developed algorithm. However, if BDF-GIRGs prove to have practical significance, an optimized implementation would be desirable.
Finally, from a theoretical perspective, the most intriguing question concerns the further exploration of geometric random graphs with a more homogeneous degree distribution that use BDFs as a distance function. For instance GIRGs with a high PLE or RGGs. We have observed that the behavior of the generated graphs can strongly depend on the chosen distance function. Explaining these differences theoretically would be desirable. It would also be interesting to investigate whether some desirable properties, such as having a giant component and the small-world phenomenon, are exhibited also for higher power-law exponents, when using certain BDFs.

Bibliography

[Ayo80] Raymond G. Ayoub. “Paolo Ruffini’s contributions to the quintic”. en. In: *Archive for History of Exact Sciences* Volume 23 (Sept. 1980), pp. 253–277. ISSN: 1432-0657. DOI: *10.1007/BF00357046*.

[BF24] Thomas Bläsius and Philipp Fischbeck. “On the External Validity of Average-Case Analyses of Graph Algorithms”. en. In: *ACM Transactions on Algorithms* Volume 20 (Jan. 2024), pp. 1–42. ISSN: 1549-6325, 1549-6333. DOI: *10.1145/3633778*.

[BKL18] Karl Bringmann, Ralph Keusch, and Johannes Lengler. *Average Distance in a General Class of Scale-Free Networks with Underlying Geometry*. Nov. 2018. DOI: *10.48550/arXiv.1602.05712*.

[BKL19] Karl Bringmann, Ralph Keusch, and Johannes Lengler. “Geometric inhomogeneous random graphs”. In: *Theoretical Computer Science* Volume 760 (Feb. 2019), pp. 35–54. ISSN: 0304-3975. DOI: *10.1016/j.tcs.2018.08.014*.

[Blä+22] Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, Ulrich Meyer, Manuel Penschuck, and Christopher Weyand. “Efficiently generating geometric inhomogeneous and hyperbolic random graphs”. en. In: *Network Science* Volume 10 (Dec. 2022), pp. 361–380. ISSN: 2050-1242, 2050-1250. DOI: *10.1017/nws.2022.32*.

[CF06] Deepayan Chakrabarti and Christos Faloutsos. “Graph mining: Laws, generators, and algorithms”. In: *ACM Computing Surveys* Volume 38 (June 2006), 2–es. ISSN: 0360-0300. DOI: *10.1145/1132952.1132954*.

[DC02] Jesper Dall and Michael Christensen. “Random geometric graphs”. en. In: *Physical Review E* Volume 66 (July 2002), p. 016121. ISSN: 1063-651X, 1095-3787. DOI: *10.1103/PhysRevE.66.016121*.

[Erc11] Gunes Ercal. “Small Worlds and Rapid Mixing with a Little More Randomness on Random Geometric Graphs”. en. In: *NETWORKING 2011*. Edited by Jordi Domingo-Pascual, Pietro Manzoni, Sergio Palazzo, Ana Pont, and Caterina Scoglio. Berlin, Heidelberg: Springer, 2011, pp. 281–293. ISBN: 978-3-642-20757-0. DOI: *10.1007/978-3-642-20757-0_22*.

[FFF99] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. “On power-law relationships of the Internet topology”. In: *SIGCOMM Comput. Commun. Rev.* Volume 29 (Aug. 1999), pp. 251–262. ISSN: 0146-4833. DOI: *10.1145/316194.316229*.

[FGKS24] Tobias Friedrich, Andreas Göbel, Maximilian Katzmann, and Leon Schiller. “Real-World Networks are Low-Dimensional: Theoretical and Practical Assessment”. en. In: *Proceedings of the Thirty-ThirdInternational Joint Conference on Artificial Intelligence*. Aug. 2024, pp. 2036–2044. DOI: *10.24963/ijcai.2024/225*.

[FSS13] Tobias Friedrich, Thomas Sauerwald, and Alexandre Stauffer. “Diameter and Broadcast Time of Random Geometric Graphs in Arbitrary Dimensions”. en. In: *Algorithmica* Volume 67 (Sept. 2013), pp. 65–88. ISSN: 1432-0541. DOI: *10.1007/s00453-012-9710-y*.

[Kar10] Richard M. Karp. “Reducibility Among Combinatorial Problems”. en. In: *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art.* Edited by Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey. Berlin, Heidelberg: Springer, 2010, pp. 219–241. ISBN: 978-3-540-68279-0. DOI: *10.1007/978-3-540-68279-0_8*.

[KR13] Michał Karoński and Andrzej Ruciński. “The Origins of the Theory of Random Graphs”. en. In: *The Mathematics of Paul Erdős I.* Edited by Ronald L. Graham, Jaroslav Nešetřil, and Steve Butler. New York, NY: Springer, 2013, pp. 371–397. ISBN: 978-1-4614-7258-2. DOI: *10.1007/978-1-4614-7258-2_23*.

[Kri+10] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marian Boguna. *Hyperbolic Geometry of Complex Networks.* en. June 2010. DOI: *10.1103/PhysRevE.82.036106*.

[KRS24] Marc Kaufmann, Raghu Raman Ravi, and Ulysse Schaller. *Sublinear Cuts are the Exception in BDF-GIRGs.* en. May 2024. DOI: *10.48550/arXiv.2405.19369*.

[Kün24] Marvin Künnemann. *Fine-Grained Complexity Theory & Algorithms, Karlsruhe Institute of Technology.* 2024.

[LT17] Johannes Lengler and Lazar Todorovic. *Existence of Small Separators Depends on Geometry for Geometric Inhomogeneous Random Graphs.* Nov. 2017. DOI: *10.48550/arXiv.1711.03814*.

[SAK04] Gábor Szabó, Mikko Alava, and János Kertész. “Clustering in Complex Networks”. en. In: *Complex Networks.* Edited by Eli Ben-Naim, Hans Frauenfelder, and Zoltan Toroczkai. Berlin, Heidelberg: Springer, 2004, pp. 139–162. ISBN: 978-3-540-44485-5. DOI: *10.1007/978-3-540-44485-5_7*.

[TM69] Jeffrey Travers and Stanley Milgram. “An Experimental Study of the Small World Problem”. en. In: *Sociometry* Volume 32 (Dec. 1969), p. 425. ISSN: 00380431. DOI: *10.2307/2786545*.