Generating Random Hyperbolic Graphs in Subquadratic Time*

Moritz von Looz, Henning Meyerhenke, and Roman Prutkin

{moritz.looz-corswarem, meyerhenke, roman.prutkin}@kit.edu
Karlsruhe Institute of Technology (KIT), Germany

Abstract. Complex networks have become increasingly popular for modeling various real-world phenomena. Realistic generative network models are important in this context as they simplify complex network research regarding data sharing, reproducibility, and scalability studies. *Random hyperbolic graphs* are a very promising family of geometric graphs with unit-disk neighborhood in the hyperbolic plane. Previous work provided empirical and theoretical evidence that this generative graph model creates networks with many realistic features.
In this work we provide the first generation algorithm for random hyperbolic graphs with subquadratic running time. We prove a time complexity of $O((n^{3/2} + m) \log n)$ with high probability for the generation process. This running time is confirmed by experimental data with our implementation. The acceleration stems primarily from the reduction of pairwise distance computations through a polar quadtree, which we adapt to hyperbolic space for this purpose and which can be of independent interest. In practice we improve the running time of a previous implementation (which allows more general neighborhoods than the unit disk) by at least two orders of magnitude this way. Networks with billions of edges can now be generated in a few minutes.

Keywords: complex networks, hyperbolic geometry, efficient range query, polar quadtree, generative graph model

1 Introduction

The algorithmic analysis of *complex networks* is a highly active research area since complex networks are increasingly used to represent phenomena as varied as the WWW, social relations, protein interactions, and brain topology [20]. Complex networks have several non-trivial topological features: They are usually *scale-free*, which refers to the presence of a few high-degree vertices (hubs) among many low-degree vertices. A heavy-tail degree distribution that occurs frequently in practice follows a power law [20, Chap. 8.4], i. e. the number of vertices with degree k is proportional to $k^{-\gamma}$, for a fixed exponent $\gamma > 0$. Moreover, complex networks often have the *small-world property*, i. e. the distance between any two vertices is surprisingly small, regardless of network size.

Generative network models play a central role in many complex network studies for several reasons: Real data often contains confidential information; it is then desirable

* This work is partially supported by SPP 1736 *Algorithms for Big Data* of the German Research Foundation (DFG) and by the Ministry of Science, Research and the Arts Baden-Württemberg (MWK) via project *Parallel Analysis of Dynamic Networks*.

to work on similar synthetic networks instead. Quick testing of algorithms requires small test cases, while benchmarks and scalability studies need bigger graphs. Graph generators can provide data at different user-defined scales for this purpose. Also, transmitting and storing a generative model and its parameters is much easier than doing the same with a gigabyte-sized network. A central goal for generative models is to produce networks which replicate relevant structural features of real-world networks [10]. Finally, generative models are an important theoretical part of network science, as they can improve our understanding of network formation. The most widely used graph-based system benchmark in high-performance computing, Graph500 [6], is based on R-MAT [11]. This model is efficiently computable, but has important drawbacks concerning realism and preservation of properties over different graph sizes [15].

Random hyperbolic graphs (RHGs), introduced by Krioukov et al. [16], are a very promising graph family in this context: They yield a provably high clustering coefficient (a measure for the frequency of triangles) [13], small diameter [9] and a power-law degree distribution with adjustable exponent. They are based on *hyperbolic geometry*, which has negative curvature and is the basis for one of the three isotropic spaces. (The other two are Euclidean (flat) and spherical geometry (positive curvature).) In the generative model, vertices are distributed randomly on a hyperbolic disk of radius R and edges are inserted for every vertex pair whose distance is below R.[1] This family of graphs has been analyzed well theoretically [8,13,9,14] and Krioukov et al. [16] showed that complex networks have a natural embedding in hyperbolic geometry. Calculating all pairwise distances in the generation process has quadratic time complexity. This impedes the creation of massive networks and is likely the reason previously published networks based on hyperbolic geometry have been in the range of at most 10^5 vertices. A faster generator is necessary to use this promising model for networks of interesting scales.

Outline and contribution. We develop, analyze, and implement a fast, subquadratic generation algorithm for random hyperbolic graphs.

To lay the foundation, Section 2 discusses other generative network models and introduces fundamentals of hyperbolic geometry. The main technical part starts with Section 3, in which we use the Poincaré disk model to relate hyperbolic to Euclidean geometry. This allows the use of a new spatial data structure, namely a polar quadtree adapted to hyperbolic space, to reduce both asymptotic complexity and running time of the generation. We further prove the time complexity of our generation process to be $O((n^{3/2} + m) \log n)$ with high probability (whp, i. e. $\geq 1 - 1/n$) for a graph with n vertices, m edges, and sufficiently large n. Our experimental results in Section 4 confirm our theoretical bound for the running time. A graph with 10^7 vertices and 10^9 edges can be generated with our shared-memory parallel implementation in about 8 minutes. The generator code is available for review and will be open-sourced after acceptance in a future version of the network analysis toolkit NetworKit [25].

[1] We consider the name "hyperbolic unit-disk graphs" as more precise, but we use "random hyperbolic graphs" to be consistent with the literature. More general neighborhoods are possible [16] but not considered here since most theoretical works [8,13,9] are for unit-disk neighborhoods.

2 Related Work and Preliminaries

To make the following discussions clearer, we first introduce some network terminology. The *clustering coefficient* is the fraction of closed triangles to triads (paths of length 2) and measures how likely two vertices with a common neighbor are to be connected. Many real networks have multiple *connected components*, yet one large component is usually dominant. The *diameter* is the longest shortest path in the graph, which is often surprisingly small in complex networks. Complex networks also often exhibit a *community structure*, i. e. dense subgraphs with sparse connections between them.

2.1 Existing Graph Generators

The Barabasi-Albert model [2] is a preferential attachment model, designed to replicate the growth of real complex networks. The probability that a new vertex will be attached to an existing vertex v is proportional to v's degree, which results in a power-law degree distribution. While the distribution's exponent is constant for the basic model, generalizations for arbitrary exponents exist (see e. g. [21, Chap. 14]). Preferential attachment processes can be implemented with a running time in $O(n + m)$ [7].

The Dorogovtsev-Mendes model [12] is designed to model network growth with a fixed average degree. It is very fast in theory ($\Theta(n)$) and practice, but accepts only the vertex count as parameter and is thus inflexible.

The Recursive Matrix (R-MAT) model [11] was proposed to recreate properties of complex networks including a power-law degree distribution, the small-world property and self-similarity. Design goals also include few parameters and high generation speed. The R-MAT generator recursively subdivides the initially empty adjacency matrix into quadrants and drops edges into it according to given probabilities. It has $\Theta(m \log n)$ asymptotic complexity and is fast in practice. However, at least the R-MAT parameters used by the Graph500 system benchmark [6] lead to an insignificant community structure and clustering coefficients, as no incentive to close triangles exists.

Given a degree sequence seq, the Chung-Lu (CL) model [1] adds edges (u, v) with a probability of $p(u, v) = \frac{seq(u)seq(v)}{\sum_k seq(k)}$, recreating seq in expectation. The model can be conceived as a weighted version of the well-known Erdős-Rényi (ER) model and has similar capabilities as the R-MAT model [24]. Implementations exist with $\Theta(n + m)$ time complexity [18]. LFR [17], in turn, was designed as a benchmark generator for community detection algorithms. Usually the user specifies vertex degrees and community sizes. However, the implementation is also able to handle other parameters.

BTER [15] is a two-stage structure-driven model. It uses the standard ER model to form relatively dense subgraphs and thus distinct communities. Afterwards, the CL model is used to add edges, matching the desired degree distribution in expectation [23]. This is done in $\Theta(n + m \log d_{\max})$, where $d_{\max}$ is the maximum vertex degree.

To summarize, all these models have their characteristics and also deficiencies. While some seem more preferable than others, no model is widely accepted as suitable for all (or at least most) possible scenarios or applications. Thus, the promising previous results described next motivate a deeper investigation of RHGs.

2.2 Graphs in Hyperbolic Geometry

Kriokouv et al. [16] introduced the family of random hyperbolic graphs and showed how they naturally develop a power-law degree distribution and other properties of complex networks. In the generative model, vertices are generated as points in polar coordinates (ϕ, r) on a disk of radius R in the hyperbolic plane with curvature $-\zeta^2$. We denote this disk with $\mathbb{D}_R$. The angular coordinate ϕ is drawn from a uniform distribution over $[0, 2\pi]$. The probability density for the radial coordinate r is given by [16, Eq. (17)] and controlled by a growth parameter α:

$$f(r) = \alpha \frac{\sinh(\alpha r)}{\cosh(\alpha R) - 1} \tag{1}$$

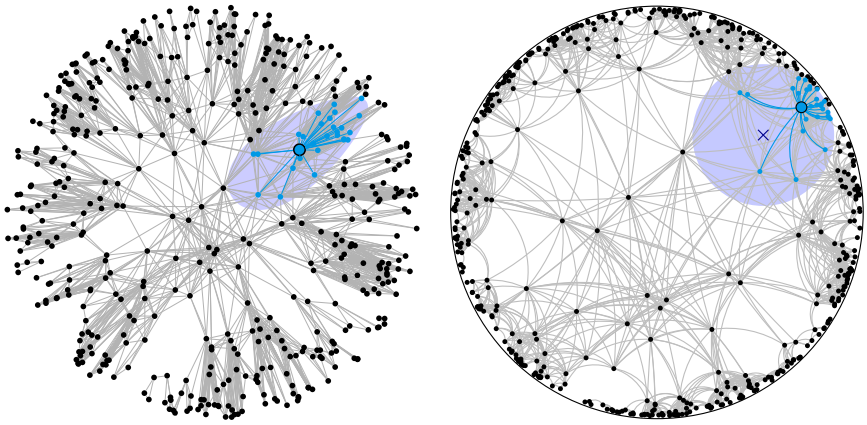
For $\alpha = 1$, this yields a uniform distribution on hyperbolic space within $\mathbb{D}_R$. For lower values of α, vertices are more likely to be in the center, for higher values more likely at the border of $\mathbb{D}_R$.

We denote the hyperbolic distance between two points p_1 and p_2 with $\mathrm{dist}_{\mathcal{H}}(p_1, p_2)$. In the model, any two vertices u and v are connected by an edge if their hyperbolic distance $\mathrm{dist}_{\mathcal{H}}(u, v)$ is below R. The neighborhood of a point (= vertex) thus consists of the points lying in a hyperbolic circle around it. (Krioukov et al. also present a more general model in which edges are inserted with a probability depending on hyperbolic distance. For this purpose, they define a family of monotonically falling functions parametrized by a temperature T. As noted earlier, we are in line with the theoretical works [8,13,9] and discuss unit-disk neighborhoods only. The latter can be considered as the special case $T = 0$.) Several works have analyzed the properties of the resulting graphs theoretically. Krioukov et al. show that for $\alpha/\zeta > \frac{1}{2}$, the degree distribution follows a power law with exponent $2 \cdot \alpha/\zeta + 1$ [16, Eq. (29)]. Gugelmann et al. [13] prove non-vanishing clustering and a low variation of the clustering coefficient. Bode et al. [9] discuss the size of the giant component and the probability that the graph is connected [8]. They also show [8] that the curvature parameter ζ can be fixed while retaining all degrees of freedom, we thus assume $\zeta = 1$ from now on. Kiwi and Mitsche [14] bound the diameter asymptotically almost surely for $\frac{1}{2} < \alpha < 1$. The average degree $\overline{k}$ of a random hyperbolic graph is controlled with the radius R, using an approximation given by [16, Eq. (22)]. An example graph with 500 vertices, $R \approx 5.08$ and $\alpha = 0.8$ is shown in Figure 1a. For the purpose of illustration in the figure, we choose a vertex u (the bold blue vertex) and add edges (u, v) for all vertices v where $\mathrm{dist}_{\mathcal{H}}(u, v) \leq 0.2 \cdot R$. The neighborhood of u then consists of vertices within a hyperbolic circle (marked in blue).

A previous generator implementing the extended model and with quadratic complexity is available [3]. We show in Section 4.1 that for the random hyperbolic graphs described above, our implementation is at least two orders of magnitude faster in practice.

2.3 Poincaré Disk Model

The Poincaré disk model is one of several representations of hyperbolic space within Euclidean geometry and maps the hyperbolic plane onto the Euclidean unit disk $D_1(0)$. The hyperbolic distance between two points $p_E, q_E \in D_1(0)$ is then given by the



(a) Native representation as specified in [16].

(b) Poincaré disk model.

Fig. 1: Comparison of geometries. Neighbors of the bold blue vertex are in the hyperbolic respective Euclidean circle. The center of the Euclidean circle is marked with $\times$.

Poincaré metric [4]:

$$\operatorname{dist}_{\mathcal{H}}(p_E, q_E) = \operatorname{acosh}\left(1 + 2\frac{||p_E - q_E||^2}{(1 - ||p_E||^2)(1 - ||q_E||^2)}\right). \tag{2}$$

Figure 1b shows the same graph as in Figure 1a, but translated into the Poincaré model. This model is conformal, i. e. it preserves angles. More importantly for us, it maps hyperbolic circles onto Euclidean circles.

3 Fast Generation of Graphs in Hyperbolic Geometry

We proceed by showing how to relate hyperbolic to Euclidean circles. Using this transformation, we are able to partition the Poincaré disk with a polar quadtree that supports efficient range queries. We adapt the network generation algorithm to use this quadtree and prove subsequently that it achieves subquadratic generation time.

3.1 Generation Algorithm

Transformation from hyperbolic geometry. Neighbors of a query point $u = (\phi_h, r_h)$ lie in a hyperbolic circle around u with radius R. This circle, which we denote as H, corresponds to a Euclidean circle E in the Poincaré disk. The center E_c and radius rad_E of E are in general different from u and R. All points on the boundary of E in the Poincaré disk are also on the boundary of H and thus have hyperbolic distance R from u. Among these points, the two on the ray from the origin through u are straightforward to construct by keeping the angular coordinate fixed and choosing the radial coordinates to match the hyperbolic distance: (ϕ_h, r_{e_1}) and (ϕ_h, r_{e_2}), with $r_{e_1}, r_{e_2} \in [0, 1)$, $r_{e_1} \neq r_{e_2}$ and $\operatorname{dist}_{\mathcal{H}}(E_c, (\phi_h, r_e)) = R$ for $r_e \in \{r_{e_1}, r_{e_2}\}$. It follows (for details see Appendix A):

Algorithm 1: Graph Generation

Input: $n, \overline{k}, \alpha$
Output: $G = (V, E)$

1 R = getTargetRadius($n, \overline{k}, \alpha$); /* Eq. (22) [16] */
2 V = n vertices;
3 T = empty polar quadtree of radius mapToPoincare(R);
4 **for** *vertex* $v \in V$ **do**
5 draw $\phi[v]$ from $\mathcal{U}[0, 2\pi)$;
6 draw $r_{\mathcal{H}}[v]$ with density $f(r) = \alpha \sinh(\alpha r)/(\cosh(\alpha R) - 1)$; /* Eq. (1) */
7 $r_E[v]$ = mapToPoincare($r_{\mathcal{H}}[v]$);
8 insert v into T at $(\phi[v], r_E[v])$;
9 **for** *vertex* $v \in V$ **do in parallel**
10 $C_{\mathcal{H}}$ = circle around $(\phi[v], r_{\mathcal{H}}[v])$ with radius R;
11 C_E = transformCircleToEuclidean($C_{\mathcal{H}}$); /* Prop. 1 */
12 **for** *vertex* $w \in T$.*getVerticesInCircle*(C_E) **do**
13 add (v, w) to E;
14 **return** G;

Proposition 1. *E_c is at $(\phi_h, \frac{2r_h}{ab+2})$ and* rad_E *is* $\sqrt{\left(\frac{2r_h}{ab+2}\right)^2 - \frac{2r_h^2 - ab}{ab+2}}$, *with $a = \cosh(R) - 1$ and $b = (1 - r_h^2)$.*

Algorithm. The generation of $G = (V, E)$ with n vertices and average degree k is shown in Algorithm 1. As in previous efforts [3], vertex positions are generated randomly (lines 5 and 6). We then map these positions into the Poincaré disk (line 7) and, as a new feature, store them in a polar quadtree (line 8). For each vertex u the hyperbolic circle defining the neighborhood is mapped into the Poincaré disk according to Proposition 1 (lines 10-11) – also see Figure 1b, where the neighborhood of u consists of exactly the vertices in the light blue Euclidean circle. Edges are then created by executing a Euclidean range query with the resulting circle in the polar quadtree (lines 12-13). The functions used by Algorithm 1 are discussed in Appendix B. We use the same probability distribution for the node positions and add an edge (u, v) exactly if the hyperbolic distance between u and v is less than R. This leads to the following proposition:

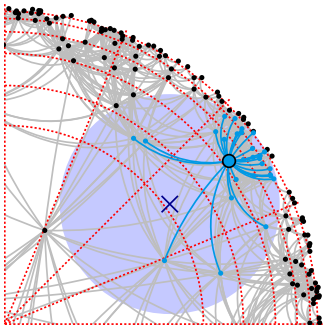


Fig. 2: Polar quadtree

Proposition 2. *Algorithm 1 generates random hyperbolic graphs as defined in Section 2.2.*

Data structure. As mentioned above, our central data structure is a polar quadtree on the Poincaré disk. While Euclidean quadtrees are common [22], we are not aware of previous adaptations to hyperbolic space. A node in the quadtree is defined as a tuple

$(\min_\phi, \max_\phi, \min_r, \max_r)$ with $\min_\phi \leq \max_\phi$ and $\min_r \leq \max_r$. It is responsible for a point $p = (\phi_p, r_p) \in D_1(0)$ iff $(\min_\phi \leq \phi_p < \max_\phi)$ and $(\min_r \leq r_p < \max_r)$. Figure 2 shows a section of a polar quadtree, where quadtree nodes are marked by dotted red lines. When a leaf cell is full, it is split into four children. Splitting in the angular direction is straightforward as the angle range is halved: $\text{mid}_\phi := \frac{\max_\phi + \min_\phi}{2}$. For the radial direction, we choose the splitting radius to result in an equal division of probability mass:

$$\text{mid}_{r\mathcal{H}} := \text{acosh}\left(\frac{\cosh(\alpha \max_{r\mathcal{H}}) + \cosh(\alpha \min_{r\mathcal{H}})}{2}\right)/\alpha \tag{3}$$

(Note that Eq. (3) uses radial coordinates in the native representation, which are converted back to coordinates in the Poincaré disk.) This leads to two lemmas useful for establishing the time complexity of the main quadtree operations:

Lemma 1. *Let $\mathbb{D}_R$ be a hyperbolic disk of radius R, $\alpha \in \mathbb{R}$, p a point in $\mathbb{D}_R$ which is chosen according to the distribution discussed in Section 2.2, and T be a polar quadtree on $\mathbb{D}_R$. Let C be a quadtree cell at depth i. Then, the probability that p is in C is 4^{-i}.*

Proof. See Appendix C.

Lemma 2. *Let R and $\mathbb{D}_R$ be as in Lemma 1. Let T be a polar quadtree on $\mathbb{D}_R$ containing n points distributed according to Section 2.2. Then, for n sufficiently large,* $\text{height}(T) \in O(\log n)$ *whp.*

Proof. See Appendix D.

3.2 Time Complexity

The time complexity of the generator is determined by the quadtree operations.

Quadtree Insertion. For the amortized analysis, we consider each element's initial and final position during the insertion of n elements. Let $h(T)$ be the final height of quadtree T, let $h(i)$ be the final depth of element i and let $t(i)$ be the depth of i when it was inserted. During insertion of element i, $t(i)$ quadtree nodes are visited until the correct leaf for insertion is found, the cost for this is linear in $t(i)$. When a leaf cell is full, it splits into four children and the depth of each element in the leaf increases by one. Over the course of inserting all n elements, element i thus moves $h(i) - t(i)$ times due to leaf splits. To reach its final position at depth $h(i)$, element i accrues cost of $O(t(i) + h(i) - t(i)) = O(h(i)) \subseteq O(h(T))$, which is $O(\log n)$ whp due to Lemma 2. The amortized time complexity for a node insertion is then: $T(\text{Insertion}) \in O(\log n)$ whp.

Quadtree Range Query. Neighbors of a vertex u are the vertices within a Euclidean circle constructed according to Proposition 1. Let $\mathcal{N}(u)$ be this neighborhood set in the final graph, thus $\deg(u) := |\mathcal{N}(u)|$. We denote leaf cells that do not have non-leaf siblings as *bottom leaf cells*. A visualization can be found in Figure 4 in Appendix E.

Lemma 3. *Let T and n be as in Lemma 2. A range query on T returning a point set A will examine at most $O(\sqrt{n} + |A|)$ bottom leaf cells whp.*

Proof. See Appendix E.

Due to Lemma 3, the number of examined bottom leaf cells for a range query around u is in $O(\sqrt{n} + \deg(u))$ whp. The query algorithm traverses T from the root downward. For each bottom leaf cell b, $O(h(T))$ inner nodes and non-bottom leaf cells are examined on the path from the root to b. Due to Lemma 2, $h(T)$ is in $O(\log n)$ whp. The time complexity to gather the neighborhood of a vertex u with degree $\deg(u)$ is thus: $T(\mathrm{RQ}(u)) \in O\left((\sqrt{n} + \deg(u)) \cdot \log n\right)$ whp.

Graph Generation. To generate a graph G from n points, the n positions need to be generated and inserted into the quadtree. The time complexity of this is $O(n) + n \cdot O(\log n) = O(n \log n)$ whp. In the next step, neighbors for all points are extracted. This has a complexity of

$$T(\mathrm{Edges}) = \sum_{v} O\left(\left(\sqrt{n} + \deg(v)\right) \cdot \log n\right) = O\left(\left(n^{3/2} + m\right) \log n\right) \text{ whp.} \quad (4)$$

This dominates the quadtree operations and thus total running time. We conclude:

Theorem 1. *Generating random hyperbolic graphs can be done in $O((n^{3/2} + m) \log n)$ time whp for sufficiently large n, i. e. with probability $\geq 1 - 1/n$.*

4 Experimental Evaluation

We compare the output and running times of our implementation and the implementation of Aldecoa et al. [3] for the same parameters. Please note that the implementation at [3] supports the generalized model of Krioukov et al. described in Section 2.2. In practice, their implementation allows a variable clustering coefficient.

Implementation. Our implementation uses the NetworKit toolkit [25] and is written in C++ 11. The code is compiled with GCC 4.8 and parallelized with OpenMP. The parallelization over the range queries is straightforward as they are independent.

Several optimizations improve performance. Sorting the points by angular coordinates before generating the graph improves cache locality, since points close to each other also have similar neighbors and thus similar access patterns to the quadtree data structure. The number of memory reallocations while constructing the neighborhood of a vertex v can be reduced by pre-allocating memory according to the expected degree of v, which is approximated by [16, Eq. (12)]. This is especially useful in a parallel setting with a global lock for memory allocations. The effect of these optimizations depend on the density of the generated graph. For reviewing, our implementation is available at `https://algohub.iti.kit.edu/parco/NetworKit/NetworKit-Hyperbolic/` on the branch "hyperbolic_generator" with the username "rev-isaac15" and the password "F33db@ck?".

Experimental Setup. In the comparison with the implementation of [3], the generated graphs could not be compared directly as both implementations sample random graphs. In its output files, the implementation of [3] does provide the hyperbolic coordinates of the generated points. Yet, since the distance threshold R is computed non-deterministically with a Monte Carlo process and not written to the log file, we do not have all necessary information to recreate the graphs exactly. Instead, we generate series of graphs with 10000 nodes, average degree $\overline{k}$ between 4 and 256 and degree distribution exponent γ between 2.2 and 7. We then compare the average properties of the generated graphs.

Running time measurements were made on a server with 256 GB RAM and 2x8 Intel Xeon E5-2680 cores at 2.7 GHz. With hyperthreading, we use 32 virtual threads for our parallel implementation. The implementation of [3] is sequential.

4.1 Results

Qualitative Comparison. The Figures 5, 6 and 7 in Appendix F show properties of the generated graphs, averaged over 10 runs. Plots showing graphs created with the implementation of [3] are on the left, plots created with our implementation are on the right. Some random fluctuations are visible, but for almost all properties the averages of our implementation are very similar to the implementation of [3]. The measured values of γ for thin graphs and various target γs differs from the previous implementation, but the fluctuation within the measurements of each implementations are sufficiently strong that it leads us to assume some measurement noise. The differences between the implementations are smaller than the variations within one implementation. Both generators create graphs with at times high diameters, about 600 for graphs with 10000 nodes, $\overline{k} = 16$ and $\gamma = 7$ (Figure 5). At first glance, this seems to contradict the theoretical bound in [14]. However, their result is only for $\frac{1}{2} < \alpha < 1$, while $\gamma = 7$ corresponds to a value of 3 for α.

Running Time. Figure 3 shows the running times for networks with 10^4-10^7 vertices and up to $1.2 \cdot 10^9$ edges. We achieve a throughput of up to 13 million edges/s. Even at only 10^4 vertices, our implementation is two orders of magnitude faster than the quadratic-time implementation of [3] for the same parameters – where only one order of magnitude stems from parallelization (the typical parallel speedup values at this scale range between 8 and 12). For graphs with 10^5 vertices, we already see an improvement of three orders of magnitude and, due to our algorithm's smaller asymptotic complexity of $O((n^{3/2} + m) \log n)$, this gap grows with increasing graph sizes. Note that the proof of this complexity bound (stated in Section 3) is supported by the measurements, as illustrated by the lines for the theoretical fit in Figure 3.

5 Conclusions

In this work we have provided efficient range queries in the hyperbolic plane – based on a polar quadtree, which we have adapted to hyperbolic space and which can thus be of independent interest. We have further shown that the fast range queries facilitate the generation of random hyperbolic graphs (RHGs) in running time $O((n^{3/2} + m) \log n)$

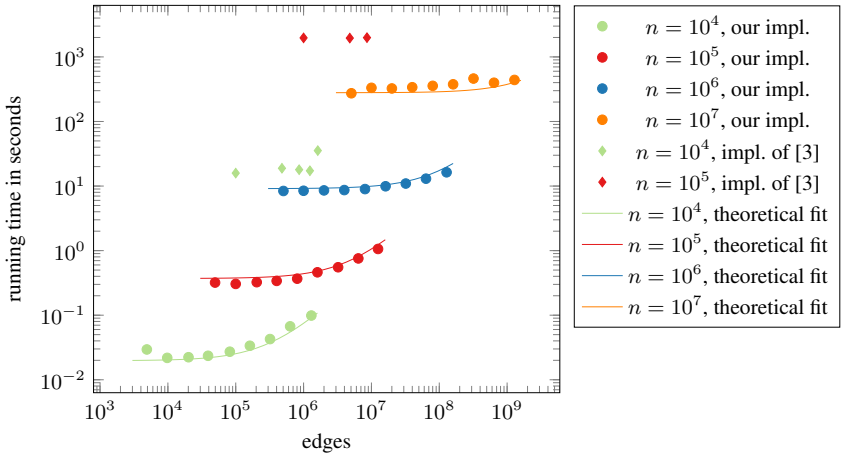


Fig. 3: Comparison of running times to generate networks with 10^4-10^7 vertices, $\alpha = 1$ and varying $\overline{k}$. Circles represent running times of our implementation, diamonds the running times of the implementation of [3]. Our running times are fitted with the equation $T(n,m) = \left(\left(3.8 \cdot 10^{-7} n + 1.14 \cdot 10^{-9} n^{3/2} + 1.38 \cdot 10^{-8} m\right) \log\ n\right)$ seconds.

time whp. In practice our parallel generator constructs RHGs with billions of edges in a few minutes, about two orders of magnitude faster than the quadratic-time algorithm.

Previous work (both theoretical and empirical) has already shown that the generated graphs are complex networks with many properties also found in real-world instances [16, Section 4]. Thus, RHGs constitute a promising model for complex network research that deserve even further attention. In the context of our paper, future work will investigate the incremental quadtree construction in order to admit a dynamic model with vertex movement; this deserves a more thorough treatment than possible here given the space constraints.

References

1. William Aiello, Fan Chung, and Linyuan Lu. A random graph model for massive graphs. In *Proc. 32nd ACM Symp. on Theory of computing*, pages 171–180. Acm, 2000.
2. R. Albert and A.L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.
3. Rodrigo Aldecoa, Chiara Orsini, and Dmitri Krioukov. Hyperbolic graph generator. *arXiv preprint arXiv:1503.05180*, 2015.
4. James W Anderson. *Hyperbolic geometry; 2nd ed.* Springer undergraduate mathematics series. Springer, Berlin, 2005.
5. R. Arratia and L. Gordon. Tutorial on large deviations for the binomial distribution. *Bulletin of Mathematical Biology*, 51(1):125–131, 1989.
6. D. A. Bader, Jonathan Berry, Simon Kahan, Richard Murphy, E. Jason Riedy, and Jeremiah Willcock. Graph 500 benchmark 1 (”search”), version 1.1. Technical report, Graph 500, 2010.
7. Vladimir Batagelj and Ulrik Brandes. Efficient generation of large random networks. *Physical Review E*, 71(3):036113, 2005.

8. Michel Bode, Nikolaos Fountoulakis, and Tobias Müller. The probability that the hyperbolic random graph is connected. 2014. Preprint available at `http://www.staff.science.uu.nl/~muell001/Papers/BFM.pdf`.
9. Michel Bode, Nikolaos Fountoulakis, and Tobias Müller. On the giant component of random hyperbolic graphs. In Jaroslav Nešetřil and Marco Pellegrini, editors, *The Seventh European Conference on Combinatorics, Graph Theory and Applications*, volume 16 of *CRM Series*, pages 425–429. Scuola Normale Superiore, 2013.
10. Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys (CSUR)*, 38(1):2, 2006.
11. Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-MAT: A recursive model for graph mining. In *Proc. 4th SIAM Intl. Conf. on Data Mining (SDM)*, Orlando, FL, April 2004. SIAM.
12. Sergei N Dorogovtsev and José FF Mendes. *Evolution of networks: From biological nets to the Internet and WWW*. Oxford University Press, 2003.
13. Luca Gugelmann, Konstantinos Panagiotou, and Ueli Peter. Random hyperbolic graphs: Degree sequence and clustering - (extended abstract). In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Proceedings, Part II*, 2012.
14. Marcos Kiwi and Dieter Mitsche. A bound for the diameter of random hyperbolic graphs. *preprint available at http://arxiv. org/abs/1408.2947*, 2015.
15. Tamara G. Kolda, Ali Pinar, Todd Plantenga, and C. Seshadhri. A scalable generative graph model with community structure. *SIAM J. Scientific Computing*, 36(5):C424–C452, Sep 2014.
16. Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *Physical Review E*, 82, Sep 2010.
17. Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.
18. Joel C Miller and Aric Hagberg. Efficient generation of networks with given expected degrees. In *Algorithms and Models for the Web Graph*, pages 115–126. Springer, 2011.
19. Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.
20. Mark Newman. *Networks: an introduction*. Oxford University Press, 2010.
21. Mark Newman. *Networks: An Introduction*. 2010.
22. Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
23. C Seshadhri, Tamara G Kolda, and Ali Pinar. Community structure and scale-free collections of Erdős-Rényi graphs. *Physical Review E*, 85(5):056109, 2012.
24. C. Seshadhri, Ali Pinar, and Tamara G. Kolda. The similarity between stochastic kronecker and chung-lu graph models. In *Proc. 2012 SIAM Intl. Conf. on Data Mining (SDM)*, pages 1071–1082, 2012.
25. Christian L Staudt, Aleksejs Sazonovs, and Henning Meyerhenke. Networkit: An interactive tool suite for high-performance network analysis. *arXiv preprint arXiv:1403.3005*, 2014.

Appendix

A Derivation of Proposition 1

When given a hyperbolic circle with center (ϕ_h, r_h) and radius rad_h as in Section 3.1, the radial coordinates r_e of points on the corresponding Euclidean circle can be derived with several transformations from the definition of the hyperbolic distance:

$$\text{rad}_h = \text{acosh}\left(1 + \frac{2(r_e - r_h)^2}{(1 - r_h^2)(1 - r_e^2)}\right) \tag{5}$$

$$\Leftrightarrow \cosh(\text{rad}_h) - 1 = \frac{2(r_e - r_h)^2}{(1 - r_h^2)(1 - r_e^2)} \tag{6}$$

$$\Leftrightarrow (\cosh(\text{rad}_h) - 1)(1 - r_e^2) = \frac{2(r_e^2 - 2r_h r_e + r_h^2)}{1 - r_h^2} \tag{7}$$

To keep the notation short, we define $a := \cosh(\text{rad}_h) - 1$ and $b := (1 - r_h^2)$. Since $\text{rad}_h > 0$ and $r_h \in [0, 1)$, both a and b are greater than 0. It follows:

$$(\cosh(\text{rad}_h) - 1)(1 - r_e^2) = \frac{2(r_e^2 - 2r_h r_e + r_h^2)}{1 - r_h^2} \tag{8}$$

$$\Leftrightarrow a - a \cdot r_e^2 = \frac{2(r_e^2 - 2r_h r_e + r_h^2)}{b} \tag{9}$$

$$\Leftrightarrow a = r_e^2 \cdot a + \frac{2(r_e^2 - 2r_h r_e + r_h^2)}{b} \tag{10}$$

$$\Leftrightarrow a = r_e^2(a + \frac{2}{b}) + r_e\frac{-4r_h}{b} + \frac{2r_h^2}{b} \tag{11}$$

$$\Leftrightarrow 0 = r_e^2(a + \frac{2}{b}) + r_e\frac{-4r_h}{b} + \frac{2r_h^2}{b} - a \tag{12}$$

$$\Leftrightarrow 0 = r_e^2 + r_e\frac{-4r_h}{b(a + \frac{2}{b})} + \frac{2r_h^2}{b(a + \frac{2}{b})} - \frac{a}{(a + \frac{2}{b})} \tag{13}$$

Solving this quadratic equation, we obtain:

$$r_{e_{1,2}} = \frac{2r_h}{ab + 2} \pm \sqrt{\left(\frac{2r_h}{ab + 2}\right)^2 - \frac{2r_h^2 - ab}{ab + 2}} \tag{14}$$

Since (ϕ_h, r_{e_1}) and (ϕ_h, r_{e_2}) are different points on the border of E, the center E_c needs to be on the perpendicular bisector. Its radial coordinate r_{E_c} is thus $(r_{e_1} + r_{e_2})/2 = \frac{2r_h}{ab+2}$. To determine the angular coordinate, we need the following lemma:

Lemma 4. *Let H be a hyperbolic circle with center at (ϕ_h, r_h) and radius rad_h. The center E_c of the corresponding Euclidean circle E is on the ray from (ϕ_h, r_h) to the origin.*

Proof. Let p be a point in H, meaning $\mathrm{dist}_{\mathcal{H}}(p, (\phi_h, r_h)) \leq \mathrm{rad}_h$. Let p' be the mirror image of p under reflection on the ray going through (ϕ_h, r_h) and p. The point (ϕ_h, r_h) is on the ray and unchanged under reflection: $(\phi_h, r_h) = (\phi_h, r_h)'$. Since reflection on the equator is an isometry in the Poincaré disk model and preserves distance, we have $\mathrm{dist}_{\mathcal{H}}(p', (\phi_h, r_h))$ = $\mathrm{dist}_{\mathcal{H}}(p, (\phi_h, r_h)') = \mathrm{dist}_{\mathcal{H}}(p, (\phi_h, r_h)) \leq \mathrm{rad}_h$ and $p' \in H$. The Euclidean circle E is then symmetric with respect to the ray and its center E_c must lie on it. □

The radius of the circle is then derived from the distance of the center to (ϕ_h, r_{e_1}) and (ϕ_h, r_{e_2}), which is $\sqrt{\left(\frac{2r_h}{ab+2}\right)^2 - \frac{2r_h^2 - ab}{ab+2}}$. With both radial and angular coordinates of E_c fixed, Proposition 1 follows.

B Methods Used in Algorithm 1

B.1 getTargetRadius

For given values of n, α and R, an approximation of the expected average degree $\overline{k}$ is given by [16, Eq. (22)] and the notation $\xi = (\alpha/\zeta)/(\alpha/\zeta - 1/2)$:

$$\overline{k} = \frac{2}{\pi}\xi^2 n \left(e^{-\zeta R/2} + e^{-\alpha R} \left(\alpha \frac{R}{2} \left(\frac{\pi}{4} \left(\frac{\zeta}{\alpha}^2 \right) - (\pi - 1)\frac{\zeta}{\alpha} + (\pi - 2) \right) - 1 \right) \right) \tag{15}$$

As mentioned in Section 2, the value of ζ can be fixed while retaining all degrees of freedom in the model and we thus assume $\zeta = 1$. We then use binary search with fixed n, α and desired $\overline{k}$ to find an R that gives us a close approximation of the desired average degree $\overline{k}$. Note that the above equation is only an approximation and might give wrong results for extreme values. Our implementation could easily be adapted to skip this step and accept the commonly used[14] parameter C, with $R = 2\ln n + C$ or even accept R directly. For increased usability, we accept the average degree $\overline{k}$ as a parameter in the default version.

B.2 mapToPoincare

In the native representation[16], the radial coordinate $r_{\mathcal{H}}$ of a point $p_{\mathcal{H}} = (\phi_{\mathcal{H}}, r_{\mathcal{H}})$ is set to the hyperbolic distance to the origin:

$$r_{\mathcal{H}} = \mathrm{dist}_{\mathcal{H}}(p_{\mathcal{H}}, (0, 0))$$

A mapping $g : \mathcal{H}^2 \to D_1(0)$ from the native representation to the Poincaré disc model needs to preserve the hyperbolic distance to the origin across models. By using the definition of the Poincaré metric, we can derive its radial coordinate r_e in the Poincaré disc model:

$$g((\phi_{\mathcal{H}}, r_{\mathcal{H}})) = \left(\phi_{\mathcal{H}}, \sqrt{\frac{\cosh(r_{\mathcal{H}}) - 1}{\cosh(r_{\mathcal{H}}) + 1}} \right) \tag{16}$$

This mapping then gives the correct hyperbolic distance with the Poincaré metric (Eq. (2):

$$\text{dist}_{\mathcal{H}}(g((\phi_{\mathcal{H}}, r_{\mathcal{H}})), (0,0)) = \text{acosh}\left(1 + 2\frac{||g((\phi_{\mathcal{H}}, r_{\mathcal{H}})) - (0,0)||^2}{(1 - ||g((\phi_{\mathcal{H}}, r_{\mathcal{H}}))||^2)(1 - ||(0,0)||^2)}\right) \tag{17}$$

$$= \text{acosh}\left(1 + 2\frac{||g((\phi_{\mathcal{H}}, r_{\mathcal{H}}))||^2}{(1 - ||g((\phi_{\mathcal{H}}, r_{\mathcal{H}}))||^2)(1)}\right) \tag{18}$$

$$= \text{acosh}\left(1 + 2\frac{||\left(\phi_{\mathcal{H}}, \sqrt{\frac{\cosh(r_{\mathcal{H}})-1}{\cosh(r_{\mathcal{H}})+1}}\right)||^2}{(1 - ||\left(\phi_{\mathcal{H}}, \sqrt{\frac{\cosh(r_{\mathcal{H}})-1}{\cosh(r_{\mathcal{H}})+1}}\right)||^2)}\right) \tag{19}$$

$$= \text{acosh}\left(1 + 2\frac{\left(\sqrt{\frac{\cosh(r_{\mathcal{H}})-1}{\cosh(r_{\mathcal{H}})+1}}\right)^2}{(1 - \left(\sqrt{\frac{\cosh(r_{\mathcal{H}})-1}{\cosh(r_{\mathcal{H}})+1}}\right)^2)}\right) \tag{20}$$

$$= \text{acosh}\left(1 + 2\frac{\left(\frac{\cosh(r_{\mathcal{H}})-1}{\cosh(r_{\mathcal{H}})+1}\right)}{1 - \left(\frac{\cosh(r_{\mathcal{H}})-1}{\cosh(r_{\mathcal{H}})+1}\right)}\right) \tag{21}$$

$$= \text{acosh}\left(1 + 2\frac{(\cosh(r_{\mathcal{H}}) - 1)}{\cosh(r_{\mathcal{H}}) + 1 - (\cosh(r_{\mathcal{H}}) - 1)}\right) \tag{22}$$

$$= \text{acosh}\left(1 + 2\frac{(\cosh(r_{\mathcal{H}}) - 1)}{2}\right) \tag{23}$$

$$= \text{acosh}\left((\cosh(r_{\mathcal{H}})\right) = r_{\mathcal{H}} \tag{24}$$

$$\tag{25}$$

B.3 transformCircleToEuclidean

The circle is constructed according to Proposition 1.

C Proof of Lemma 1

To prove Lemma 1, we use Eq. (1) and an auxiliary lemma.

Lemma 5. *Let p be a point in $\mathbb{D}_R$ and C a quadtree cell delimited by $\min_r$, $\max_r$, $\min_\phi$ and $\max_\phi$. The probability of p being in C is given by the following equation:*

$$\Pr(p \in C) = \frac{\max_\phi - \min_\phi}{2\pi} \cdot \frac{\cosh(\alpha \max_r) - \cosh(\alpha \min_r)}{\cosh(\alpha R) - 1} \tag{26}$$

Proof. Let $g((\phi, r))$ be the probability density for a point p at (ϕ, r). In Section 2.2, we used a uniform distribution over $[0, 2\pi)$ for the angles and defined $f(r)$ as the probability density for radial coordinate r. Since the two parameters are independent, we write:

$$g : [0, 2\pi) \times [0, R) \to [0, 1] \tag{27}$$

$$g((\phi, r)) = \frac{1}{2\pi} \cdot f(r) = \frac{1}{2\pi} \cdot \alpha \frac{\sinh(\alpha r)}{\cosh(\alpha R) - 1} \tag{28}$$

With C delimited by $\min_r$, $\max_r$, $\min_\phi$ and $\max_\phi$ and g being the product of two independent functions, we write:

$$\Pr(p \in C) = \int_{\min_r}^{\max_r} \int_{\min_\phi}^{\max_\phi} \frac{1}{2\pi} \cdot \alpha \frac{\sinh(\alpha r)}{\cosh(\alpha R) - 1} d\phi dr$$

Constant factors can be moved out of the integral:

$$\Pr(p \in C) = \frac{1}{2\pi} \cdot \frac{1}{\cosh(\alpha R) - 1} \cdot \int_{\min_r}^{\max_r} \int_{\min_\phi}^{\max_\phi} \alpha \sinh(\alpha r) d\phi dr$$

The integrand is independent of ϕ:

$$\Pr(p \in C) = \frac{\max_\phi - \min_\phi}{2\pi} \cdot \frac{1}{\cosh(\alpha R) - 1} \cdot \int_{\min_r}^{\max_r} \alpha \sinh(\alpha r) dr$$

Finally, we get:

$$\Pr(p \in C) = \frac{\max_\phi - \min_\phi}{2\pi} \cdot \frac{\cosh(\alpha \max_r) - \cosh(\alpha \min_r)}{\cosh(\alpha R) - 1}$$

□

We proceed by proving Lemma 1 by induction.

Proof. Start of induction (i = 0): At depth 0, only the root cell exists and covers the whole disk. Since $C = \mathbb{D}_R$, $\Pr(p \in C) = 1 = 4^{-0}$.

Inductive step ($i \to i + 1$): Let C_i be a node at depth i. C_i is delimited by the radial boundaries $\min_r$ and $\max_r$, as well as the angular boundaries $\min_\phi$ and $\max_\phi$. It has four children at depth $i + 1$, separated by mid_r and mid_ϕ. Let SW be the south west child of C_i. With Lemma 5, the probability of $p \in SW$ is:

$$\Pr(p \in SW) = \frac{\mathrm{mid}_\phi - \min_\phi}{2\pi} \cdot \frac{\cosh(\alpha \mathrm{mid}_r) - \cosh(\alpha \min_r)}{\cosh(\alpha R) - 1} \tag{29}$$

The angular range is halved ($\mathrm{mid}_\phi := \frac{\max_\phi + \min_\phi}{2}$) and mid_r is selected according to Eq. (3):

$$\mathrm{mid}_r := \operatorname{acosh}\left(\frac{\cosh(\alpha \max_r) + \cosh(\alpha \min_r)}{2}\right) / \alpha$$

This results in a probability of

$$\frac{\frac{\max_\phi + \min_\phi}{2} - \min_\phi}{2\pi} \cdot \frac{\cosh(\alpha \cdot \operatorname{acosh}\left(\frac{\cosh(\alpha \max_r)+\cosh(\alpha \min_r)}{2}\right)/\alpha) - \cosh(\alpha \min_r)}{\cosh(\alpha R) - 1} \tag{30}$$

$$= \frac{\max_\phi + \min_\phi - 2\min_\phi}{4\pi} \cdot \frac{\cosh(\operatorname{acosh}\left(\frac{\cosh(\alpha \max_r)+\cosh(\alpha \min_r)}{2}\right)) - \cosh(\alpha \min_r)}{\cosh(\alpha R) - 1} \tag{31}$$

$$= \frac{\max_\phi - \min_\phi}{4\pi} \cdot \frac{\frac{\cosh(\alpha \max_r)+\cosh(\alpha \min_r)}{2} - \cosh(\alpha \min_r)}{\cosh(\alpha R) - 1} \tag{32}$$

$$= \frac{\max_\phi - \min_\phi}{4\pi} \cdot \frac{\cosh(\alpha \max_r) + \cosh(\alpha \min_r) - 2\cosh(\alpha \min_r)}{2(\cosh(\alpha R) - 1)} \tag{33}$$

$$= \frac{1}{4}\frac{\max_\phi - \min_\phi}{2\pi} \cdot \frac{\cosh(\alpha \max_r) - \cosh(\alpha \min_r)}{\cosh(\alpha R) - 1} \tag{34}$$

$$= \frac{1}{4}\Pr(p \in C_i) \tag{35}$$

As per the induction hypothesis, $\Pr(p \in C_i)$ is 4^{-i} and $\Pr(p \in SW)$ is thus $\frac{1}{4} \cdot 4^{-i} = 4^{-(i+1)}$. Due to symmetry when selecting mid_ϕ, the same holds for the south east child of C_i. Together, they contain half of the probability mass of C_i. Again due to symmetry, the same proof then holds for the northern children as well. □

D Proof of Lemma 2

First we motivate and introduce an auxiliary lemma which is used in the proof. We say "with high probability" (whp) when referring to a probability of at least $1 - 1/n$ (for sufficiently large n). While previous results exist for the height and cost of two-dimensional quadtrees [22], these quadtrees differ from our polar hyperbolic approach in important properties and the results are not easily transferable. For example, we adjust the size of our quadtree cells to result in an equal division of probability mass when taking the hyperbolic geometry into account, see Lemma 5. We thus make use of a lemma from the theory of *balls into bins* [19] instead.

Lemma 6. *When n balls are thrown independently and uniformly at random into n^3 bins, the probability that any bin receives more than one ball is less than $1/n$.*

Proof. Each ball has a probability of $1/\left(n^3\right)$ to land in a given bin and among n balls, $\binom{n}{2}$ pairs exist. Using a union bound, we can thus upper bound the probability that bin 1 receives at least two balls to:

$$\binom{n}{2}\left(\frac{1}{n^3}\right)^2.$$

Again with a union bound, it follows that the probability that any of the n^3 bins receives at least two balls is at most

$$n^3 \binom{n}{2} \left(\frac{1}{n^3}\right)^2 = \binom{n}{2} \left(\frac{1}{n^3}\right).$$

It follows:

$$\binom{n}{2} \left(\frac{1}{n^3}\right) \tag{36}$$

$$= \frac{n(n-1)}{2} \cdot \frac{1}{n^3} \tag{37}$$

$$< n^2 \cdot \frac{1}{n^3} = \frac{1}{n} \tag{38}$$

□

Proof (of Lemma 2). In a complete quadtree, 4^i cells exist at depth i. For analysis purposes only, we construct such a complete but initially empty quadtree of height $k = 3 \cdot \lceil \log_4(n) \rceil$, which has at least n^3 leaf cells. As seen in Lemma 1, a given point has an equal chance to land in each leaf cell. Hence, we can apply Lemma 6 with each leaf cell being a bin and a point being a ball. (The fact that we can have more than n^3 leaf cells only helps in reducing the average load.) From this we can conclude that, for n sufficiently large, no leaf cell of the current tree contains more than 1 point with high probability (whp). Consequently, the total quadtree height does not exceed $k = 3 \cdot \lceil \log_4(n) \rceil \in O(\log n)$ whp.

Let T' be the quadtree as constructed in the previous paragraph, starting with a complete quadtree of height k and splitting leaves when their capacity is exceeded. Let T be the quadtree created in our algorithm, starting with a root node, inserting points and also splitting leaves when necessary, growing the tree downward.

Since both trees grow downward as necessary to accommodate all points, but T does not start with a complete quadtree of height k, the set of quadtree nodes in T is a subset of the quadtree nodes in T'. Consequently, the height of T is bounded by $O(\log n)$ whp as well. □

E Proof of Lemma 3

Proof. As done previously, we denote leaf cells that do not have non-leaf siblings as *bottom leaf cells*, see Figure 4 for an example. The following proof is done for a leaf capacity of one. Since a larger leaf capacity does not increase the tree height and adds only a constant factor to the cost of examining a leaf, this choice does not result in a loss of generality.

Let L be the set of bottom leaf cells containing a vertex in A and let Q be the set of bottom leaf cells examined by the range query. Since the contents of leaf cells are disjoint, $|L| \leq |A|$ holds. The set $Q \backslash L$ consists of leaf cells which are examined by the range query but yield no points in A. These are empty leaf cells within the query circle as well as cells cut by the circle boundary.

Empty leaf cells occur when a previous leaf cell is split since its capacity is exceeded by at least one point. Therefore an empty leaf cell a in the interior of the query circle only occurs when at least two points happened to be allocated to its parent cell b. A split caused by two points creates four leaf cells, therefore there are at most twice as many empty bottom leaf cells as points.

The number of cells cut by the boundary can be bounded with a geometric argument. At depth $k = \lceil \log_4 n \rceil$, at most 4^k cells exist, defined by at most 2^k angular and 2^k radial divisions. When following the circumference of a query circle, each newly cut leaf cell requires the crossing of an angular or radial division. Each radial and angular coordinate occurs at most twice on the circle boundary, thus each division can be crossed at most twice. With two types of divisions, the circle boundary crosses at most $2 \cdot 2 \cdot 2^k = 4 \cdot 2^{\lceil \log_4 n \rceil}$ cells at depth k. Since the value of $4 \cdot 2^{\lceil \log_4 n \rceil}$ is smaller than $4 \cdot 2^{1+\log_4 n}$, this yields $< 8 \cdot \sqrt{n}$ cut cells.

In a balanced tree, all cells at depth k are leaf cells and the bound calculated above is an upper bound for $|Q \backslash L|$. For the general case of an unbalanced tree, we use auxiliary Lemma 7, which bounds to $O(\sqrt{n})$ the number of bottom leaf cells descendant from cells cut at depth k. □

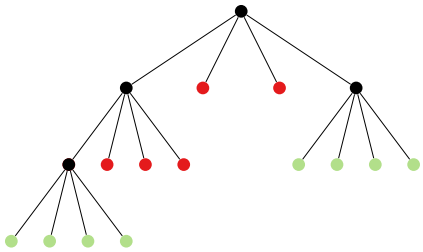


Fig. 4: Visualization of bottom leaf cells in a quadtree. Bottom leaf cells are marked in green, non-bottom leaf cells in red and interior cells in black.

Lemma 7. *Let $\mathbb{D}_R$ be a disk with radius R in hyperbolic space. Let T be a quadtree on $\mathbb{D}_R$, containing n points distributed according to Section 2.2. Let $k := \lceil \log_4 n \rceil$ and let C be a set of $\lfloor c \cdot \sqrt{n} \rfloor$ quadtree cells at depth k, for $c \geq 1$. The total number of bottom leaf cells among the descendants of cells in C is then bounded by $4c \cdot \sqrt{n}$ whp.*

Proof. New leaf cells are only created if a point is inserted in an already full cell. We argue similarly to Lemma 3 that the descendants contain at most twice as many empty bottom leaf cells as points. The number of points in the cells of C is a random variable, which we denote by X. Since each point position is drawn independently and each point is equally likely to land in each cell at a given depth (Lemma 5), X follows a binomial distribution:

$$X \sim B\left(n, \frac{\lfloor c \cdot \sqrt{n} \rfloor}{4^k}\right) \tag{39}$$

For ease of calculation, we define a slightly different binomial distribution $Y \sim B(n, \frac{c\cdot\sqrt{n}}{n})$. Since $n \leq 4^k$ and $c \cdot \sqrt{n} \geq \lfloor c \cdot \sqrt{n} \rfloor$, the tail bounds calculated for Y also hold for X.

Let $H(\frac{2c\cdot\sqrt{n}}{n}, \frac{c\cdot\sqrt{n}}{n})$ be the relative entropy (also known as Kullback-Leibler divergence) of the two Bernoulli distributions $B(\frac{2c\cdot\sqrt{n}}{n})$ and $B(\frac{c\cdot\sqrt{n}}{n})$. We can then use a tail bound from [5] to gain an upper bound for the probability that more than $2c$ points are in the cells of C:

$$\Pr(Y \geq 2c \cdot \sqrt{n}) \leq \exp\left(-nH\left(\frac{2c \cdot \sqrt{n}}{n}, \frac{c \cdot \sqrt{n}}{n}\right)\right) \tag{40}$$

For consistency with our previous definition of "with high probability", we need to show that $\Pr(Y \geq 2c\sqrt{n}) \leq 1/n$ for n sufficiently large. To do this, we interpret $\Pr(Y \geq 2c \cdot \sqrt{n})/(1/n)$ as an infinite sequence and observe its behavior for $n \to \infty$. Let $a_n := \Pr(Y \geq 2c \cdot \sqrt{n})/(1/n) = n \cdot \Pr(Y \geq 2c \cdot \sqrt{n})$ and $b_n := n \cdot \exp\left(-nH\left(\frac{2c\cdot\sqrt{n}}{n}, \frac{c\cdot\sqrt{n}}{n}\right)\right)$. From Eq. (40) we know that $a_n \leq b_n$.

Using the definition of relative entropy, we iterate over the two cases (point within C, point not in C) for both Bernoulli distributions and get:

$$b_n = n \cdot \exp\left(-nH\left(\frac{2c\sqrt{n}}{n}, \frac{c\sqrt{n}}{n}\right)\right) \tag{41}$$

$$= n \cdot \exp\left(-n\left(\left(\frac{2c}{\sqrt{n}}\right)\ln 2 + \left(1 - \frac{2c}{\sqrt{n}}\right)\ln\frac{1 - \frac{2c\sqrt{n}}{n}}{1 - \frac{c\sqrt{n}}{n}}\right)\right) \tag{42}$$

$$= n \cdot \exp\left(-n\frac{2c}{\sqrt{n}}\ln 2\right) \cdot \exp\left(-n\left(1 - \frac{2c}{\sqrt{n}}\right)\ln\frac{\sqrt{n} - 2c}{\sqrt{n} - c}\right) \tag{43}$$

$$= n \cdot \exp\left(-2c\sqrt{n}\ln 2\right) \cdot \exp\left(\left(n - 2c\sqrt{n}\right)\ln\frac{\sqrt{n} - c}{\sqrt{n} - 2c}\right) \tag{44}$$

$$= n \cdot \frac{1}{2^{2c\sqrt{n}}} \cdot \left(\frac{\sqrt{n} - c}{\sqrt{n} - 2c}\right)^{n - 2c\sqrt{n}} \tag{45}$$

$$= n \cdot \frac{1}{4^{c\sqrt{n}}} \cdot \left(1 + \frac{c}{\sqrt{n} - 2c}\right)^{n - 2c\sqrt{n}} \tag{46}$$

(While b_n is undefined for $n \in \{c^2, 4c^2\}$, we only consider *sufficiently large* n from the outset.)

We apply a variant of the root test and consider the limit $\lim_{n\to\infty}(b_n)^{\frac{1}{\sqrt{n}}}$ for an auxiliary result:

$$\lim_{n\to\infty}\left(n \cdot \frac{1}{4^{c\sqrt{n}}} \cdot \left(1 + \frac{c}{\sqrt{n} - 2c}\right)^{n - 2c\sqrt{n}}\right)^{\frac{1}{\sqrt{n}}} \tag{47}$$

$$= \lim_{n\to\infty} n^{\frac{1}{\sqrt{n}}} \cdot \frac{1}{4^c} \cdot \left(1 + \frac{c}{\sqrt{n} - 2c}\right)^{\sqrt{n}} \left(1 + \frac{c}{\sqrt{n} - 2c}\right)^{-2c} \tag{48}$$

$$= 1 \cdot \frac{1}{4^c} \cdot e^c \cdot 1 = \left(\frac{e}{4}\right)^c \tag{49}$$

From $e/4 < 0.7$, $c \geq 1$ and the limit definition, it follows that almost all elements in $(b_n)^{\frac{1}{\sqrt{n}}}$ are smaller than 0.7 and thus almost all elements in b_n are smaller than $0.7^{\sqrt{n}}$. Thus $\lim_{n\to\infty} b_n \leq \lim_{n\to\infty} 0.7^{\sqrt{n}} = 0$. Due to Eq. (40), we know that a_n is smaller than b_n for large n, and therefore that the number of points in C is smaller than $2c \cdot \sqrt{n}$ with probability at least $1 - \frac{1}{7}n$ for n sufficiently large. Again with high probability, this limits the number of non-leaf cells in C to $c \cdot \sqrt{n}$ and thus the number of bottom leaf cells to $4c \cdot \sqrt{n}$, proving the claim. □

F Comparison with Previous Implementation [3]

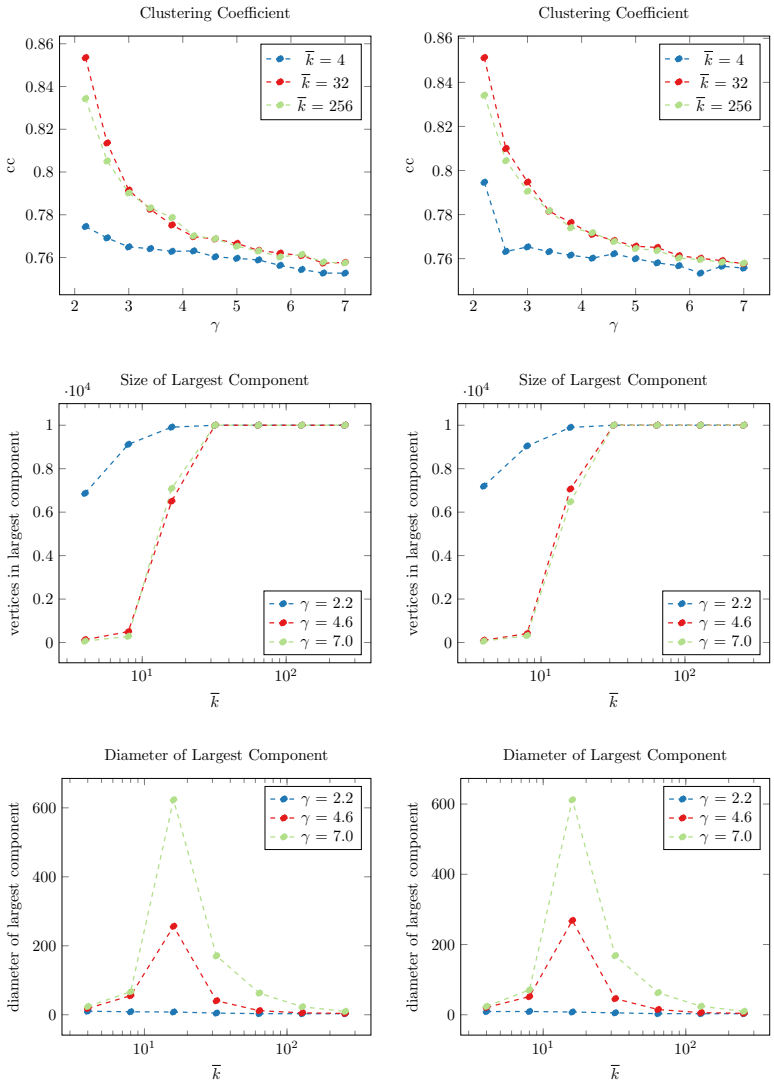


Fig. 5: Comparison of clustering coefficients, size of largest component and diameter of largest components for the implementation of [3] (left) and our implementation (right). Values are averaged over 10 runs.

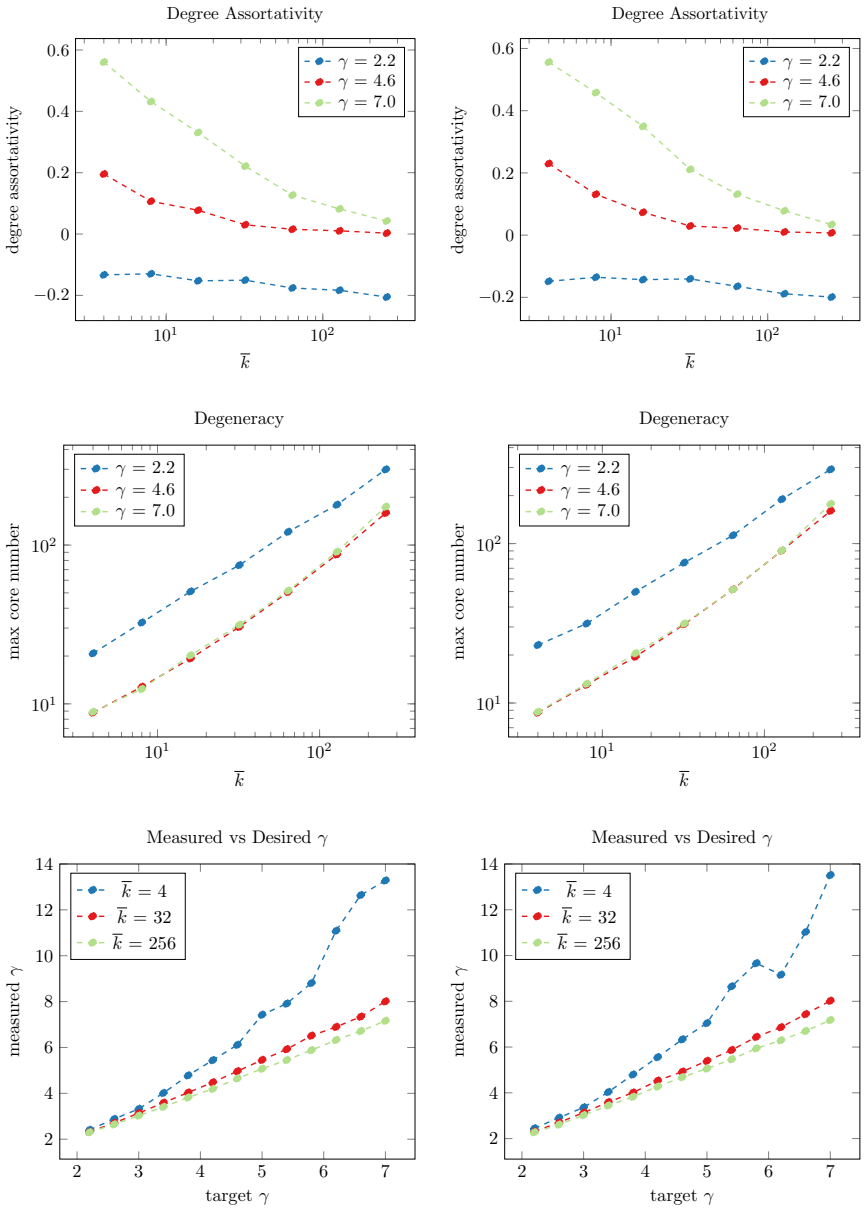


Fig. 6: Comparison of degree assortativity, degeneracy and measured vs desired γ for the implementation of [3] (left) and our implementation (right). Degree assortativity describes whether vertices have neighbors of similar degree. A value near 1 signifies subgraphs with equal degree, a value of -1 star-like structures. k-Cores, in turn, are a generalization of components and result from iteratively peeling away vertices of degree k and assigning to each vertex the core number of the innermost core it is contained in. Degeneracy refers to the largest core number. Values are averaged over 10 runs.

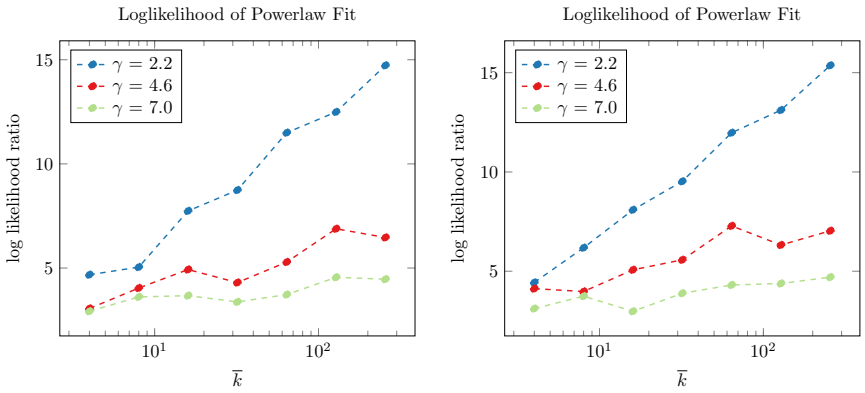


Fig. 7: Comparison of likelihood of a power-law fit to the degree distribution for the implementation of [3] (left) and our implementation (right). Values are averaged over 10 runs.