



Karlsruhe Reports in Informatics 2014,14

Edited by Karlsruhe Institute of Technology,
Faculty of Informatics
ISSN 2190-4782

Fast generation of dynamic complex networks with underlying hyperbolic geometry

Moritz von Looz, Christian L. Staudt,
Henning Meyerhenke, Roman Prutkin

2014

KIT – University of the State of Baden-Wuerttemberg and National
Research Center of the Helmholtz Association

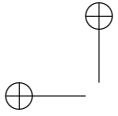


Fakultät für Informatik

Please note:

This Report has been published on the Internet under the following
Creative Commons License:

<http://creativecommons.org/licenses/by-nc-nd/3.0/de>.



Fast generation of dynamic complex networks with underlying hyperbolic geometry*

Moritz von Looz Christian L. Staudt
 Henning Meyerhenke Roman Prutkin†

Abstract

Complex networks have become increasingly popular for modeling real-world phenomena, ranging from web hyperlinks to interactions between people. Realistic generative network models are important in this context as they avoid privacy concerns of real data and simplify complex network research regarding data sharing, reproducibility, and scalability studies. We study a geometric model creating unit-disk graphs in hyperbolic space. Previous work provided empirical and theoretical evidence that this model creates networks with a hierarchical structure and other realistic features. However, the investigated networks were small, possibly due to a quadratic running time of a straightforward implementation. We provide a faster generator for a representative subset of these networks. Our experiments indicate a time complexity of $O((n+m)\log n)$ for our implementation and thus confirm our theoretical considerations. To our knowledge our implementation is the first one with subquadratic running time. The acceleration stems primarily from the reduction of pairwise distance computations through a polar quadtree newly adapted to hyperbolic space. We also extend the generator to an alternative dynamic model which preserves graph properties in expectation. Finally, we generate and evaluate the largest networks of this model published so far. Our parallel implementation computes networks with billions of edges on a shared-memory server in a matter of few minutes. A comprehensive network analysis shows that important features of complex networks, such as a low diameter, power-law degree distribution and a high clustering coefficient, are retained over different graph sizes and densities.

Keywords: complex networks, hyperbolic geometry, generative model, network analysis, polar quadtree

1 Introduction

The algorithmic analysis of *complex networks* has become a highly active research area recently since complex networks are increasingly used to represent phe-

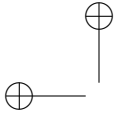
nomena as varied as the WWW, social relations, protein interactions, and brain topology [16, 21]. Complex networks have several non-trivial topological features: They are usually *scale-free*, which refers to the presence of a few high-degree nodes (hubs) among many low-degree nodes. A degree distribution of networks that occurs frequently in practice follows a power law [17, p. 247], i. e. the number of nodes 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. typical distance between two nodes is surprisingly small, regardless of network size and growth.

Generative network models play a central role in many complex network studies for several reasons: Real data, such as social networks, might contain confidential information. It is often desirable to be able to work on similar synthetic networks. Quick testing of algorithms requires small test cases, while projection of future growth and scalability studies need bigger graphs. Graph generators can provide data at different user-defined scales. Moreover, real networks might be impractical to transmit and store. When using a generative model, however, only the model parameters and the generator need to be stored or transmitted. A central goal for generative models is to produce networks with realistic features: Realism is understood as the ability to replicate relevant structural features of real-world networks such as degree distribution, spectral properties, community structure, and frequency of triangles [6]. Moreover, the formulation of generative models is an important theoretical step in network science, since realistic models can improve our understanding of phenomena that guide the formation of complex networks.

Motivation. One generative network model that has been suggested previously as fairly realistic by Krioukov et al. [14] creates networks with underlying *hyperbolic geometry*. Among the many interesting properties of hyperbolic geometry, most relevant is the *exponential expansion of space*: The area of a hyperbolic circle of radius r is $2\pi(\cosh(r) - 1) \simeq e^r$, allowing a natural embedding of trees and tree-like graphs. In recent years, the link between hyperbolic geometry and graphs with power-law degree distributions has been studied with respect to routing applications [5]. The generative model by Krioukov et al. has a proven high clustering coefficient [10], small diameter and a power-law degree distribution with adjustable exponent [14]. The generator creates unit-disk graphs based on hyperbolic geometry. Nodes are distributed randomly on a hyperbolic disk of radius R and edges are inserted for every node pair whose hyperbolic distance is below a threshold. Calculating the hyperbolic distance between each pair of coordinates has quadratic time complexity. This im-

*This work is partially supported by SPP 1736 *Algorithms for Big Data* of the German Research Foundation (DFG) and by the Juniorprofessor programme of the Ministry of Science, Research and the Arts Baden-Württemberg (MWK).

†All authors are affiliated with Karlsruhe Institute of Technology (KIT), Germany; email addresses: {moritz.looz-corswarem, christian.staudt, meyerhenke, roman.prutkin}@kit.edu.



pedes 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 10^4 nodes. A faster generator is necessary to enable a use of this promising model for networks of interesting scales. Additionally, to judge the realism of these networks, more detailed parameter studies and comparisons from a network analysis point of view are necessary.

Outline and Contribution. We implement and study the *hyperbolic unit-disk graph model* and address deficiencies in terms of generation speed and network analysis. Section 2 discusses other generative network models and introduces fundamentals of hyperbolic geometry. The main technical part starts with Section 3, in which we show how we relate hyperbolic to Euclidean geometry during the generation process. This allows us to employ a new space-partitioning data structure, more precisely a polar quadtree within the Poincaré disk model, to improve the running time of the naive generation process. We proceed by proposing an alternative dynamic model. Instead of deleting and reinserting nodes as in [20], we let nodes move gradually in the hyperbolic plane. This results in a smoother change of the network, so that we believe it to be more realistic for some applications. We also analyze the time complexity of our static and dynamic generation process in Section 3, resulting in an expected static running time in $O((n+m)\log n)$ and an expected dynamic running time in $O((k+l)\log n)$ when moving k nodes with l edges under a reasonable assumption.

In Section 4 we add to previous studies a comprehensive network analytic evaluation of the generative model based on hyperbolic geometry. The experimental results confirm the theoretical expected running time of $O((n+m)\log n)$. In practice, a graph with 10^7 nodes and 10^9 edges can be generated in less than 5 minutes on our test machine. Network analysis shows a consistently high clustering coefficient and power-law degree distribution over a wide parameter range. Proofs omitted due to space constraints can be found in the supplementary material. The generator will be made available in a future version of NetworKit [24], our open-source framework for large-scale network analysis.

2 Related Work and Preliminaries

2.1 Related Generative Network Models. The *Erdős-Rényi Model* is the earliest attempt to create a formal method for generating graphs [19]. Edges are created among n nodes with a uniform probability of p for each of the $\{u, v\}$ pairs. The *Barabasi-Albert model* [2] was intended to model the growth of real complex networks and implements a preferential attachment process which results in a power-law degree distribution. The

probability that a new node will be attached to an existing node v is proportional to its degree. The *Recursive Matrix (R-MAT)* model [7] 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. Given a degree sequence, the *Chung-Lu model* [1] creates edges (u, v) with a probability of $p(u, v) = \frac{\deg(u)\deg(v)}{\sum_k \deg(k)}$, which recreates the degree sequence in expectation. The model can be conceived as a weighted version of the Erdős-Rényi model, and has been shown to have similar capabilities as the R-MAT model [23]. The *Dorogovtsev-Mendes* generator is designed to model network growth and focuses on speed over flexibility [8]. *BTER* [13] is a two-stage structure-driven model, which combines aspects of the ER and CL model. It uses the standard ER model to form relatively dense subgraphs, thus forming distinct communities. Afterwards, the CL model is used to add edges, allowing to match the expected degree distribution [22]. Further generative network models include the *Havel-Hakimi* generator [11] (which also generates a graph from a degree sequence but tends to close triangles) and the *PubWeb* generator [9] (which is designed to model P2P networks based on Euclidean geometry). In Section 4.3, we compare our generator to the models closest in spirit: Barabasi-Albert, BTER, Chung-Lu, Dorogovtsev-Mendes and R-MAT.

2.2 Graphs in Hyperbolic Geometry. A geometric foundation for complex networks can be found in hyperbolic geometry. Kriokou et al. [14] show how unit-disk graphs generated with underlying hyperbolic geometry naturally develop a power-law degree distribution. An alternative implementation with an extended model and quadratic time complexity is available [18]. Boguñá et al. [5] use node embedding in hyperbolic space to obtain virtual coordinates enabling greedy routing on internet topology networks, and Kleinberg showed that every graph can be embedded in the hyperbolic space such that greedy routing always succeeds [12]. Papadopoulos et al. [20] extend the generator of [14] with a dynamic growth model. In the hyperbolic unit-disk graph model, nodes are distributed randomly over a disk of radius R in the hyperbolic plane. Node positions are generated in polar coordinates (ϕ, r) . The angular coordinate ϕ is drawn from a uniform distribution over $[0, 2\pi]$, while the probability density function for the radial coordinate r is proportional to the circumference of

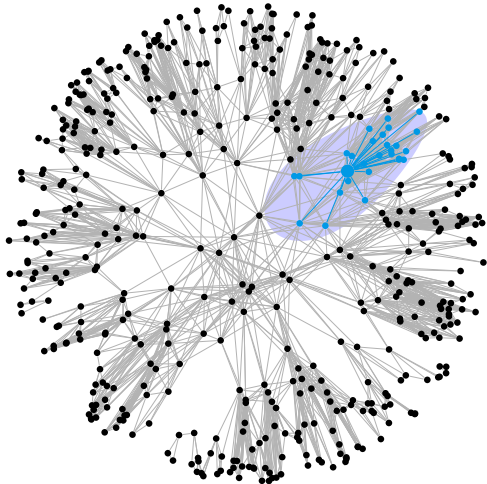
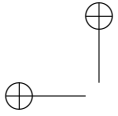


Figure 1: Native representation of graph in hyperbolic geometry. Blue nodes are in a hyperbolic circle around the bold blue node.

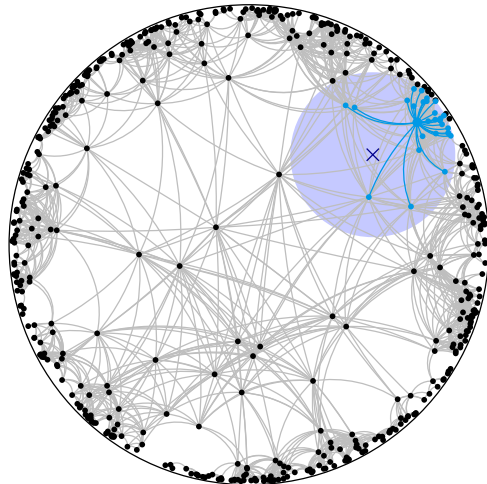


Figure 2: Poincaré disk model. Neighborhood of marked node lies in a Euclidean circle with moved center.

a hyperbolic circle with radius r :
(2.1)

$$f(r) = \frac{2\pi \sinh(r)}{2\pi(\cosh(R) - 1)} = \frac{\sinh(r)}{\cosh(R) - 1} \propto \sinh(r)$$

Integrating and normalizing the probability density function yields the cumulative distribution $F_Q(r) = (\cosh(r) - 1)/(\cosh(R) - 1)$. The probability mass is thus equally spread over the hyperbolic space within the base disk. In the basic model, an edge is inserted between two points $p = (\phi_p, r_p)$ and $q = (\phi_q, r_q)$ if their hyperbolic distance $\text{dist}_{\mathcal{H}}(p, q)$ is below a threshold. The neighborhood of a node thus consists of the nodes lying in a hyperbolic circle around it. Gugelmann et al. [10] analyzed this model theoretically and proved a low variation of the clustering coefficient for fixed parameters. The basic model can be extended with a number of parameters, of which we focus on three: The *stretch* parameter s determines the disk radius: $R = s \cdot \text{acosh}(n/(2\pi) + 1)$. The *dispersion* parameter α determines whether nodes tend to occur in the center or at the border of the hyperbolic disk. For this purpose, Equation 2.1 is changed to $f(r) \propto \alpha \cdot \sinh(\alpha r)$ and the cumulative distribution function becomes $(\cosh(\alpha r) - 1)/(\cosh(\alpha R) - 1)$. The third parameter t determines the distance threshold for edge insertion. Two nodes are connected with an edge if their hyperbolic distance is below tR .

An example graph with 500 nodes, $s = 1$, $\alpha = 0.8$ and $t = 0.2$ is shown in Figure 1. The neighborhood of node u (the bold blue node) consists of nodes v where $\text{dist}_{\mathcal{H}}(u, v) \leq 0.2 \cdot R$ (marked in blue).

2.3 Poincaré Model. The Poincaré disk model is one of several representations of hyperbolic space within Euclidean geometry. An n -dimensional hyperbolic space is represented by a hypersphere of dimension n , in our case the hyperbolic plane is mapped onto the Euclidean unit disk $D_1(0)$. The hyperbolic distance between two points $p_{\mathcal{E}}, q_{\mathcal{E}} \in D_1(0)$ is given by the Poincaré metric [3]:

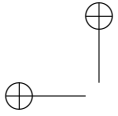
$$(2.2) \quad \text{dist}_{\mathcal{H}}(p_{\mathcal{E}}, q_{\mathcal{E}}) = \text{acosh} \left(1 + 2 \frac{\|p_{\mathcal{E}} - q_{\mathcal{E}}\|^2}{(1 - r_{p_{\mathcal{E}}}^2)(1 - r_{q_{\mathcal{E}}}^2)} \right).$$

This model is conformal, i.e. it preserves angles. Hyperbolic circles are mapped onto Euclidean circles in the Poincaré disk model, which is of importance to our generation algorithm. Figure 2 shows the same graph as in Figure 1, but translated into the Poincaré model. The neighborhood of v – a hyperbolic circle – consists of exactly those nodes which are within the light blue Euclidean circle. Radius and position of the Euclidean query circle are discussed in the supplementary materials.

3 Fast Generation of Graphs in Hyperbolic Geometry

Next we present our new generation algorithm using the adapted quadtree, prove the quadtree’s balance in expectation, and proceed with the complexity analysis of the generator and the dynamic extension.

3.1 Generation Algorithm. The generation of a graph in our model is described in Algorithm 3.1. Node positions are generated (lines 5 and 6), mapped into



the Poincaré disk (line 7) and stored in the quadtree (line 8). For each node the hyperbolic circle defining the neighborhood is generated and mapped into the Poincaré disk (lines 10 and 11). Edges are created by executing a Euclidean range query with the mapped circle in the polar quadtree. Details of the used functions are discussed in the supplementary materials.

ALGORITHM 3.1. (GRAPH GENERATION)

Input: n, t, α, s . **Output:** $G = (V, E)$

1. $R = s \cdot \operatorname{acosh}(n/(2\pi) + 1)$
2. $V = n$ nodes
3. $T =$ empty polar quadtree
4. **For each** node $v \in V$:
5. draw $\phi[v]$ from $\mathcal{U}[0, 2\pi)$.
6. draw $r_{\mathcal{H}}[v]$ with density $f(r) \propto \alpha \sinh(\alpha r)$
7. $r_{\mathcal{E}}[v] = \operatorname{hyperbolicToEuclidean}(r_{\mathcal{H}}[v])$
8. insert v into T at $(\phi[v], r_{\mathcal{E}}[v])$
9. **For each** node $v \in V$:
10. $C_{\mathcal{H}} =$ circle around $(\phi[v], r_{\mathcal{H}}[v])$ with radius tR
11. $C_{\mathcal{E}} = \operatorname{transformCircleToEuclidean}(C_{\mathcal{H}})$
12. **For each** node $w \in T.\operatorname{getNodesInCircle}(C_{\mathcal{E}})$
13. add (v, w) to E
14. **Return** G

3.2 Data Structure. Our central data structure is a polar quadtree on the Poincaré disk. 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 3 shows a section of a polar quadtree where quadtree nodes are marked by dotted red lines. When a leaf node is full, it is split into four children, once in the angular and once in the radial direction. Splitting in the angular direction is straightforward as the angle range is halved: $\operatorname{mid}_{\phi} = \frac{\max_{\phi} + \min_{\phi}}{2}$. For the radial direction, we choose the splitting radius to result in an equal division of space.

$$(3.3) \quad \operatorname{mid}_r = \operatorname{acosh}\left(\frac{\cosh(\max_r) + \cosh(\min_r)}{2}\right)$$

Theorem 3.1 *Let R be the hyperbolic radius of the disk covered. A node at depth i of the quadtree covers an area of $(2\pi(\cosh(R) - 1))/4^i$.*

When the graph generator is called with $\alpha = 1$, the points are generated uniformly within the hyperbolic disk. In this case, the expected height of the tree is logarithmically bounded in the number of nodes.

Theorem 3.2 *Let T be a polar quadtree constructed as defined in Eq. 3.3 with n points distributed uniformly in hyperbolic space. Then $\mathbb{E}(\operatorname{height}(T)) \in O(\log n)$.*

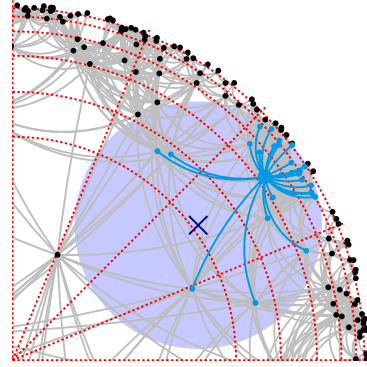


Figure 3: Polar Quadtree

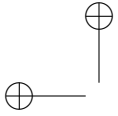
In case of $\alpha \neq 1$, the nodes are not distributed uniformly in space. By setting the splitting radius in Equation 3.3 to $\operatorname{acosh}(\cosh(\alpha \max_r) + \cosh(\alpha \min_r)/2)/\alpha$, we still have an equal division of probability mass and the balance argument holds.

3.3 Time Complexity. The time complexity of the generator is determined by the operations of the polar quadtree.

Quadtree Insertion. The quadtree is constructed one element at a time. The time required for finding the appropriate leaf node for insertion is linear in the quadtree’s height. Due to Theorem 3.2, the expected height is in $O(\log n)$. If the leaf with capacity c is full, it needs to be split up and c move operations occur. Every c th element induces a split, each element is moved one additional time in amortization. The amortized time complexity is:

$$(3.4) \quad T(\operatorname{Insertion}) \in O(\log n) + O(1) = O(\log n)$$

Quadtree Range Query. The neighborhood of a node consists of the nodes within a hyperbolic circle of radius r around it. For each neighbor found, at most c nodes have to be examined in the leaf node. At most 4-height inner nodes need to be visited, with $\mathbb{E}(\operatorname{height}) \in O(\log n)$, cf. Theorem 3.2. It is possible that leaf nodes have to be examined that do not contain suitable points. These leaf nodes are cut by the boundary of the query circle, but do not have points within the query circle. The number of unnecessarily examined leaf nodes depends on the circumference of the query circle. Due to Theorem 3.1, each leaf on a given level manages the same area of hyperbolic space. We work with the assumption that the number of equally-sized cells cut is linear in the circumference of the circle. This holds in Euclidean geometry and seems to hold here as well, but is yet unproven. Since the area of a hyperbolic circle is in $O(\cosh(r))$ and the circumference in $O(\sinh(r))$ [3],



the amortized number of unnecessary leaf examinations lies in $O(\sinh(r)/\cosh(r)) = O(\tanh(r)) = O(1)$ per extracted edge. The amortized time complexity for a node v with degree $\deg(v)$ is thus:

$$(3.5) \quad T(\text{RQ}(v)) \in O(1 + \deg(v) \cdot \log n).$$

Since the quadtree is not modified during this step, it can be parallelized easily with up to n threads.

Graph Generation. To generate a graph G with n nodes, the n points are distributed in a disc of radius $s \cdot \text{acosh}(n/(2\pi) + 1)$ and inserted into the quadtree. The expected time complexity of this is $n \cdot O(\log n) = O(n \log n)$. In the next step, neighbors for all points are extracted. This has a complexity of

$$(3.6) \quad \sum_v O(1 + \deg(v) \cdot \log n) = O((n + m) \log n).$$

The total running time is dominated by the range queries. We can therefore conclude:

Theorem 3.3 *Under the assumption of fast range queries (Eq. 3.5), generating networks with hyperbolic geometry can be done in $O((n + m) \log n)$ time in expectation.*

3.4 Dynamic Model. To model gradual change in networks, we design and implement a dynamic version with node movement. While deleting nodes or inserting them at random positions is a suitable dynamic for modeling internet infrastructure with sudden site failures or additions, change in for example social networks happens more gradually. A suitable node movement model needs to be *consistent*: After moving a node, the network may change, but properties should stay the same *in expectation*. Since the properties emerge from the node positions, the probability distribution of node positions needs to be preserved. In our implementation, movement happens in discrete time steps. We choose the movement to be *directed*: If a node i moves in a certain direction at time t , it will move in the same direction at $t + 1$. We implement this movement in two phases: In the initialization, step values τ_ϕ and τ_r are assigned to each node according to the desired movement. Each movement step of a node then consists of a rotation and a radial movement. The rotation step is a straightforward addition of angular coordinates: $\text{rotated}(\phi, r, \tau_\phi) = (\phi + \tau_\phi/r) \bmod 2\pi$. The radial movement is described in Algorithm 3.2, a visualization can be found in the supplementary materials.

ALGORITHM 3.2. (RADIAL MOVEMENT)

Input: r, τ_r, R . **Output:** r_{new}

1. $x = \cosh(r)$
2. $y = x + \tau_r$
3. $z = \text{acosh}(y)$
4. **Return** z

If the new node position would be outside the boundary ($r > R$) or below the origin ($r < 0$), the movement is reflected and τ_r set to $-\tau_r$.

Theorem 3.4 *Node movement preserves distribution of angular and radial coordinates: $F_X(r) = F_X(\text{scale}(r))$ for $0 \leq r \leq R$ and $F_\Phi(\text{rotated}((\phi, r))) = F_\Phi(\phi)$ for $0 \leq \phi \leq 2\pi$.*

Time Complexity. When a node is moved far enough to change the quadtree cells, it is deleted from the old cell and inserted into the new one. Affected edges need to be recomputed. Deletion and insertion are in amortized $O(\log n)$, thus we get:

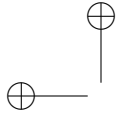
$$(3.7) \quad T(\text{move}(v)) \in O(\log n + \deg(v) \log n).$$

Details are discussed in the supplementary materials.

4 Evaluation

In this section we first discuss several structural properties of networks and use them to analyze the graphs generated by the model under different parameters. Comparisons to real-world networks and existing generators as well as an evaluation of the running time follow.

4.1 Network Properties. A complex network is a graph with non-trivial structural properties characterized by certain key figures. The *degree distribution* of many complex networks follows a *power law* [17, p. 247]. The *clustering coefficient* is the fraction of closed triangles to triads (paths of length 2) and measures how likely two nodes with a common neighbor are to be connected. *Degree assortativity* describes whether nodes have neighbors of similar degree. A value near 1 signifies subgraphs with equal degree, a value of -1 star-like structures. While many real network have multiple *connected components*, one large component is usually dominant. *k-Cores* are a generalization of components, result from iteratively peeling away nodes of degree k and assigning to each node the *core number* of the innermost core it is contained in. The maximum distance between two nodes is called the *diameter*. A small diameter despite being sparse is defining for complex networks. Complex networks also often exhibit a *community structure*, i. e. dense subgraphs with sparse connections between them. *Modularity* is a measure that quantifies how well a partition of the node set corresponds to the dense subgraphs. We use a modularity-maximizing algorithm [24] to detect communities.



The plots in Figure 4 illustrate the relationship between the model parameters *dispersion* α and *stretch* s and a structural property of the resulting network. We focus on α and s instead of the threshold t , since the theoretical analysis about power-law degree distributions was previously only done for these parameters [14]. The effect of t is very similar to the effect of s , only inverted. Plots with a variation in t instead of s can be found in the supplementary materials. The node dispersion α starts from 0.8 (lower values tend to result in a complete graph) and goes up to 3, which is sufficient to reveal the relationships. The stretch s is in the range (0.5, 2), yielding graphs that are not too dense nor too sparse. As Figure 4a shows, the network becomes sparser with rising dispersion and stretch. The density ranges from 10^{-5} at ($\alpha = 3, s = 2$) to 0.1 at ($\alpha = 0.8, s = 0.5$). The stretch parameter s has a stronger influence on the density than α . The generated graphs are connected unless the stretch is very high and the average degree below ≈ 20 . Through parameter studies, we confirm the theoretical result from [20] that the exponent of the power-law degree distribution is $\gamma = 2\alpha + 1$ (Figure 4b). When the stretch parameter is too high and the graph very sparse, the degree distribution no longer follows a power law (resulting in an outlier in Figure 4b).

The generator is capable of producing degree assortative and disassortative networks. The degree assortativity rises with α and s , above $\alpha = 1.2$ it is positive for some, above $\alpha = 1.8$ positive for all values of s (Fig. 4c). Diameter rises with thinning graphs, until the graph becomes disconnected and only the diameter of the largest component is measured (Figure 4d). At ($\alpha = 1, s = 1$) the diameter of a network with 10^6 nodes is three, the maximum diameter is obtained at ($\alpha = 3, s = 1.6$) with 5.2k. It is one of the few properties which are influenced by both parameters equally. The clustering coefficient (Fig. 4e) is above 0.75 for values of s below 1.9, significantly higher than in random graphs. The geometric graph model inherently promotes the formation of triangles, since two nodes connected to a third node are also likely to be close to each other. This value is mostly stable at around 0.8 and independent from the value of α , but decreases in cases of high s . Determining the maximum core number of a node through core decomposition, we see that high density leads to a higher degree of connectedness and hence the emergence of a dense core (Fig. 4f). The maximum core number corresponds closely to the overall density seen in Figure 4a.

Another revealing aspect of a network is its community structure. We quantify it by applying a modularity-driven algorithm [24] and measuring the modularity and average size of the resulting communities. The size of communities decreases with the sparsity of the network.

Dense graphs with very few communities have a relatively low modularity. Modularity is not independent of graph size and not an absolute measure for the strength of community structure, but nonetheless indicates that the graphs have community structure. Finally, Figure 4i shows how well a power law function fits the degree distribution. Except for cases of high s , a power-law fit is much more likely than an exponential fit.

4.2 Comparison with Real-world Networks.

To judge the realism of the generative model, we list the properties of a diverse set of real complex networks (Table 1): `PGPgiantcompo` describes the largest connected component in the PGP web of trust, `caidaRouterLevel` and `as-Skitter` represent internet topology, while `citationCiteseer` and `coPapersDBLP` are scientific collaboration networks, `soc-LiveJournal` and `fb-Texas84` are social networks and `uk-2002` and `wiki_link_en` result from web hyperlinks. The columns show the number of nodes and edges, the clustering coefficient, the maximum core number derived by core decomposition, the log likelihood of a power-law degree distribution, the exponent of an optimal power-law fit, the degree assortativity, the diameter, average size of communities and modularity of the community structure. For the study, we tried to match typical properties of this set of real networks by selecting parameters of our generator. We found that the graphs generated by the hyperbolic unit-disk model share important properties with the real-world networks, but differ in others. The power law exponent γ can be easily matched through the formula $\gamma = 2\alpha + 1$. The average degree can be influenced independently by varying s , with a higher s leading to a sparser graph. If we match density and degree distribution to a real network, the clustering coefficient tends to be higher (in the range of 0.75-0.85) than usually found in those networks, with the exception of `coPapersDBLP` (which contains many cliques). The diameter is right when matching the facebook graph, but higher by a factor of about 100 otherwise, since the generator produces fewer long-range edges that are responsible for small-world network diameters. The degree assortativity of the real networks varies from slightly negative to strongly positive, and our generator can represent this spectrum. Generated dense subgraphs tend to be a tenth as large as communities typically found in real networks of the same density, and are not independent of total graph size. Real networks mostly admit high-modularity partitions, which is also true for our synthetic graphs. When targeting an average degree below ≈ 20 , the generated graphs are no longer connected and decompose into many small connected components.

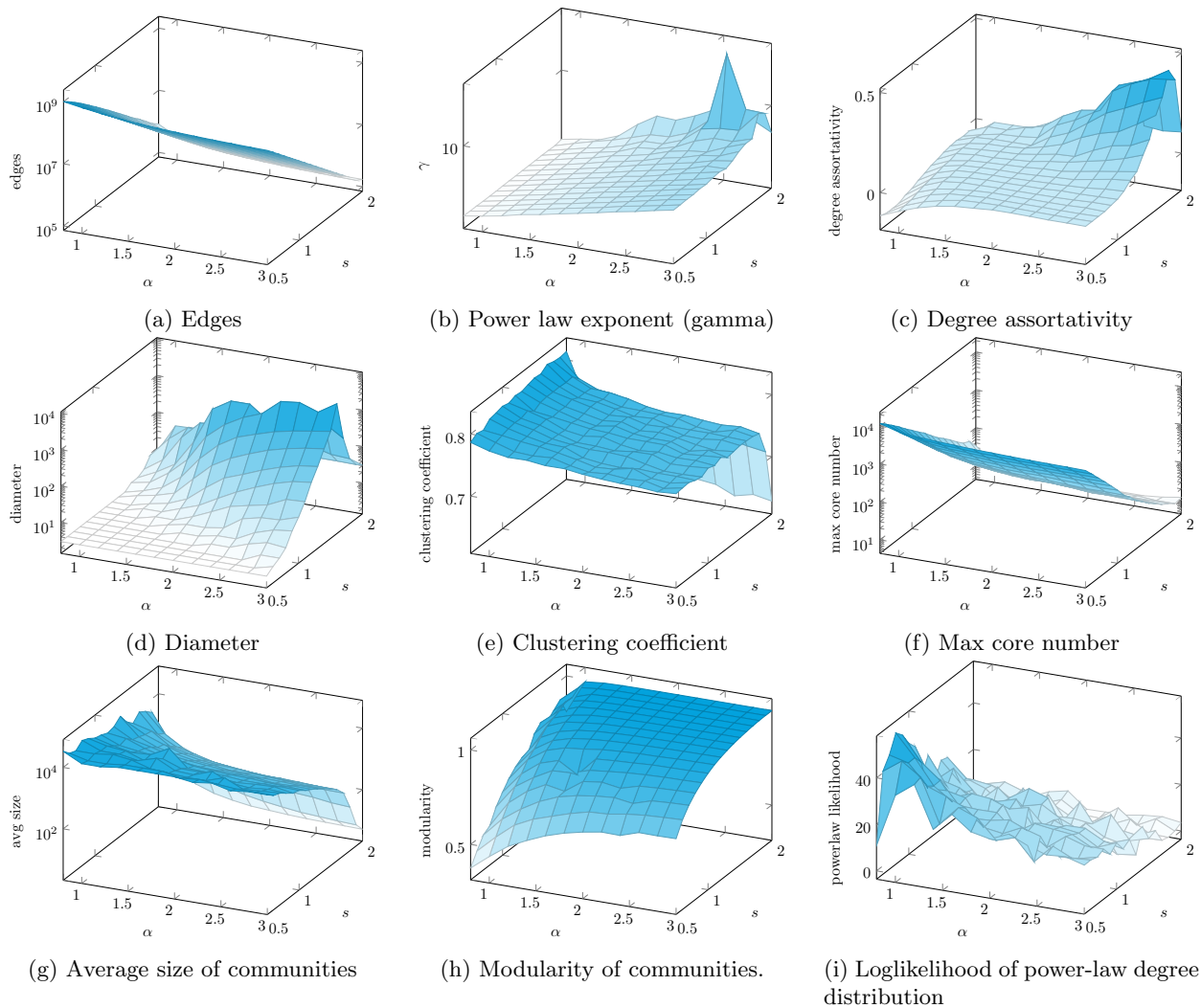
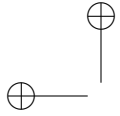
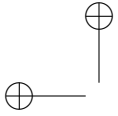


Figure 4: Properties of graphs generated with the hyperbolic generator for 100,000 nodes and different values for the parameters α and s . Higher values of α cause fewer edges to be generated, as do higher values of s . With too few edges, the graphs become disconnected and noise of other property measurements increases.

	n	m	cc	max core	power law	γ	deg.ass.	diameter	comm. size	mod.
PGPgiantcompo	10k	24k	0.44	31	2.04	4.41	0.23	24	101	0.88
fb-Texas84	36k	1.6m	0.19	81	1.54	4.8	0	7-8	1894	0.38
caidaRouterLevel	192k	609k	0.19	32	6.73	3.46	0.02	26-30	365	0.85
citationCiteseer	268k	1m	0.21	15	9.6	3.0	-0.05	36-40	1861	0.80
coPapersDBLP	540k	15m	0.81	336	4.04	5.95	0.50	23-24	2842	0.84
as-Skitter	1.7m	11m	0.3	111	20.3	2.35	-0.08	31-40	1349	0.83
soc-LiveJournal	4.8m	43m	0.36	373	6.94	3.34	0.02	19-24	632	0.75
uk-2002	18.5m	261m	0.69	943	331	2.45	-0.02	45-48	441	0.98
wiki.link_en[15]	27m	547m	0.10	-	26	3.41	-0.05	-	21.6	0.67

Table 1: Properties of various real networks.



4.3 Comparison with Existing Generators.

When comparing the hyperbolic generator and its implementation to existing generators, we consider realism, flexibility and performance. The *Barabasi-Albert* generator produces networks with a power-law degree distribution and a fixed exponent of 3, which is roughly in the range of real-world networks. However, the degree assortativity is negative and the clustering coefficient low. The running time is in $\Theta(n^2)$, rendering the creation of massive networks infeasible. Accepting only the node count n as parameter, the *Dorogovtsev-Mendes* model is very fast at the expense of flexibility. Clustering coefficient, degree assortativity and power law exponent of generated graphs are roughly similar to those of real-world networks. The *R-MAT* generator has a fast running time in practice and $O(m \log m \log n)$ asymptotic complexity; it is also more flexible and subsumes several other models. At least the Graph500 benchmark parameters lead to an insignificant community structure and clustering coefficients, as no incentive to close triangles exists.

The *Chung-Lu* model recreates a graph to a given degree sequence, which is matched in expectation. When called with degree sequences from the first four graphs in Table 1 as input, the degree distributions are matched but in all results both clustering coefficient and degree assortativity are near zero and the diameter too small. The *BTER* generator matches both degree distribution and clustering coefficient per degree of existing graphs. We test it with the PGPgiantcompo, caidaRouterLevel, citationCiteseer and coPapersDBLP networks. The degree distributions and clustering coefficients are matched with a difference of $\approx 5\%$. Similarly to the hyperbolic unit disk generator, it has difficulties recreating connected but sparse networks. Generated communities have a size of 5-45 on average, which is smaller than typical real communities and those of the hyperbolic unit disk model.

In conclusion, the hyperbolic unit disk generator can match a degree distribution exponent while having stronger clustering than the Chung-Lu and R-MAT generator and being more scalable and flexible than the Barabasi-Albert generator. Diameter and number of connected components are less realistic than those produced by BTER, but community structure is closer to typical real communities.

4.4 Performance Measurements. Figure 5 shows the running times for networks with $10^5 - 10^7$ nodes and different values for α and s . Our test platform for this is a shared-memory server with 256 GB RAM and 2x8 Intel(R) Xeon(R) E5-2680 cores (32 threads due to hyperthreading) at 2.7 GHz. Apart from outliers, the

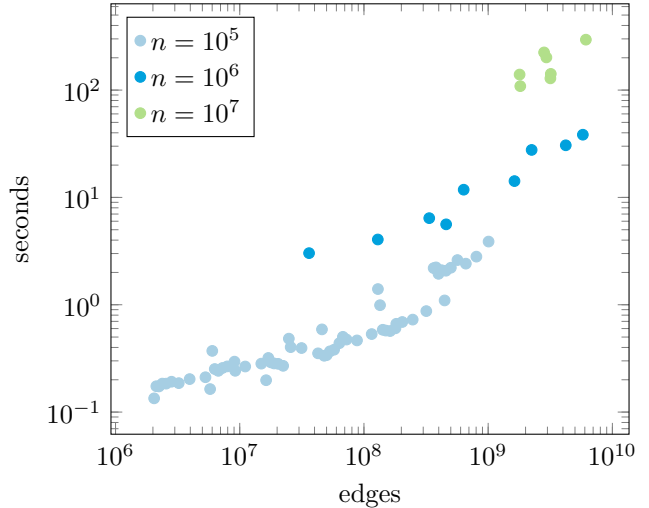


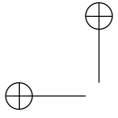
Figure 5: Runtime for generating networks with $10^5 - 10^7$ nodes and up to 6×10^9 edges. With 32 threads we achieve a throughput of up to 400 million edges/s.

running time grows linearly in the number of edges if n remains fixed. Random differences in the node positions influence the quadtree height and running time. For graphs with 10^7 nodes, system memory became limiting and small fluctuations in the memory consumption had large effects on the amount of swapping and thus running time.

5 Conclusion

We provide a scalable implementation of a generative graph model based on hyperbolic geometry. The model provides provably high clustering [10], a power-law degree distribution with an adjustable exponent and a good theoretical connection to the ongoing research about hyperbolic embedding. Under the assumption of a fast range query, the running time of our implementation is proven to be $O((n + m \log n))$ in expectation, which is asymptotically faster than previous implementations we know of. This time complexity was supported by experiments. Since all coordinates are projected onto the unit disk, floating point precision might become a problem at a certain graph size. Such problems could be circumvented by using floating point data types with arbitrary precision. In addition to the static model, we provide a dynamic model for gradual movement, which is yet without data but consistent to the static model.

We also provide a comprehensive network analysis of the generated graphs, which are the largest published networks of their kind.



Name	param	m	cc	deg.ass.	power law	γ	diam.
Barabasi-Albert	k, n0	$k \cdot n$	0-0.68	< 0	for $n0 < 0.3n$	$\simeq 3$	< 30
BTER	dd,ccd	matched	matched	\approx matched	possible	\approx matched	
Dorogovtsev-Mendes		$2 \cdot n$	0.7	0.02-0.05	yes	5-6	15-40
Chung-Lu	seq	$\approx \sum \text{seq}/2$	$< 10^{-2}$	$< 10^{-2}$	possible	varies	8-12
Hyperbolic	α, s, t		0.5-0.9	-0.05-0.4	yes	$2\alpha + 1$	3-5238
R-MAT	a, b, c, d, eF	$eF \cdot n$	0-1	0-0.6	yes	0-10	0-18

Table 2: Measured properties of various generative models. Parameter ranges were $n_{\text{Max}} = 10^5, n0 \in [0, 10^5), k \in [0, 10^4)$ for the Barabasi-Albert generator, PGPgiantcompo, caidaRouterLevel, citationCiteseer and coPapersDBLP for Chung-Lu and BTER and scale = 16, $eF = 10, a \in [0.4, 1.0), b \in [0, a), c \in [0, a)$ for R-MAT.

A Transformation of Neighborhood Circles to the Poincaré disk

When inserting edges for a given node $u = (\phi_h, r_h)$ in the hyperbolic model, all nodes v within a hyperbolic circle of radius radius_h need to be gathered. To avoid a quadratic amount of hyperbolic distance computations, we construct the Euclidean circle E in the Poincaré disk which corresponds to the hyperbolic circle around u . The center E_c and radius E_r of E are almost always different from u and radius_h . All points on the edge of the Euclidean circle have hyperbolic distance tR from u . Two of these points are straightforward to construct by keeping the angular coordinate fixed and choosing the radial coordinates to match the hyperbolic distance: (ϕ_{i_h}, r_{e_1}) and (ϕ_{i_h}, r_{e_2}) , with $r_{e_{1,2}} \neq r_h$ and

$$\text{radius}_h = \text{acosh}(1 + 2(r_e - r_h)^2 / ((1 - r_h^2)(1 - r_e^2)))$$

These points are directly below and above u . The radial coordinates can be derived with several transformations:

$$(1.8) \quad \cosh(\text{radius}_h) - 1 = \frac{2(r_e - r_h)^2}{(1 - r_h^2)(1 - r_e^2)}$$

$$(1.9) \quad (\cosh(\text{radius}_h) - 1)(1 - r_e^2) = \frac{2(r_e^2 - 2r_h r_e + r_h^2)}{1 - r_h^2}$$

To keep the notation short, define $a = \cosh(\text{radius}_h) - 1$ and $b = (1 - r_h^2)$. It follows:

$$(1.10) \quad (a) - r_e^2(a) = \frac{2(r_e^2 - 2r_h r_e + r_h^2)}{b}$$

$$(1.11) \quad a = r_e^2(a) + \frac{2(r_e^2 - 2r_h r_e + r_h^2)}{b}$$

$$(1.12) \quad = r_e^2(a + \frac{2}{b}) + r_e \frac{-4r_h}{b} + \frac{2r_h^2}{b}$$

$$(1.13) \quad 0 = r_e^2(a + \frac{2}{b}) + r_e \frac{-4r_h}{b} + \frac{2r_h^2}{b} - a$$

$$(1.14) \quad 0 = r_e^2 + r_e \frac{-4r_h}{b(a + \frac{2}{b})} + \frac{2r_h^2}{b(a + \frac{2}{b})} - \frac{a}{(a + \frac{2}{b})}$$

Solving this quadratic equation, we obtain:

$$(1.15) \quad r_{e_{1,2}} = \frac{2r_h}{ab + 2} \pm \sqrt{\left(\frac{2r_h}{ab + 2}\right)^2 - \frac{2r_h^2 - ab}{ab + 2}}$$

Since (ϕ_{i_h}, r_{e_1}) and (ϕ_{i_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 A.1 *Let H be a hyperbolic circle centered at (ϕ_h, r_h) and radius radius_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 $\text{dist}_{\mathcal{H}}(p, (\phi_h, r_h)) \leq \text{radius}_h$. Let p' be the mirror image of p under reflection on the ray going through (ϕ_h, r_h) and p . (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 $\text{dist}_{\mathcal{H}}(p', (\phi_h, r_h)) = \text{dist}_{\mathcal{H}}(p, (\phi_h, r_h)') = \text{dist}_{\mathcal{H}}(p, (\phi_h, r_h)) \leq \text{radius}_h$ and $p' \in H$. The Euclidean circle E is then symmetric with respect to the ray and E_c must lie on it. \square

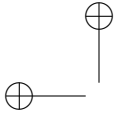
E_c is thus at $(\phi_h, \frac{2r_h}{ab+2})$ and E_r is $\sqrt{\left(\frac{2r_h}{ab+2}\right)^2 - \frac{2r_h^2 - ab}{ab+2}}$.

B Complexity Analysis and Dynamic Model

Theorem 3.1 *Let R be the hyperbolic radius of the disk covered. A node at depth i of the quadtree covers an area of $(2\pi(\cosh(R) - 1))/4^i$.*

Proof. Start of induction ($i = 0$): By definition from circle area in hyperbolic space: $\text{area}(R) = (2\pi(\cosh(R) - 1))$.

Inductive step ($i \rightarrow i + 1$): Let d be a node at level i . It has four children at level $i + 1$. Let d' be without loss of generality the left inner child of d . The area of



d' is:

$$(2.16) \quad \text{area}(d') = \frac{\text{mid}_\phi - \text{min}_\phi}{2\pi} \cdot (2\pi(\cosh(\text{mid}_r) - \cosh(\text{min}_r)))$$

This results in an area of $\frac{1}{2}(\text{max}_\phi - \text{min}_\phi) \cdot (\cosh(\text{acosh}((\cosh(\text{max}_r) + \cosh(\text{min}_r))/2)) - \cosh(\text{min}_r)) = \frac{1}{2}(\text{max}_\phi - \text{min}_\phi) \cdot (\cosh(\text{max}_r) + \cosh(\text{min}_r))/2 - \cosh(\text{min}_r) = \frac{1}{2}(\text{max}_\phi - \text{min}_\phi) \cdot \frac{1}{2}(\cosh(\text{max}_r) - \cosh(\text{min}_r)) = \frac{1}{4}\text{area}(d)$. \square

Theorem 3.2 *Let T be a polar quadtree constructed as defined in Equation 3.4 of the paper with n points distributed uniformly in hyperbolic space. Then $\mathbb{E}(\text{height}(T)) \in O(\log n)$.*

Proof. Due to Theorem 3.1, each tree cell at a given height covers an equally-sized area of hyperbolic space. In a complete quadtree 4^k cells exist at height k . For analysis, we construct a complete quadtree of height $k = \lceil \log_4(n) \rceil$, which has at least n leaf cells. As the points are distributed uniformly in hyperbolic space, each point has an equal chance to land in a given leaf cell. Let C be an arbitrary but fixed leaf cell and c the leaf capacity. The leaf C is split into child nodes when more than c points land in the region managed by C . An abstract representation of this probability is an urn with balls of 4^k different types. Drawing n balls with replacement, what is the probability more than c balls show the same type? This follows a binomial distribution:

$$(2.17) \quad P(\text{split}) = 1 - \sum_{i=0}^{[c]} \binom{n}{i} (1/(4^k))^i (1 - 1/(4^k))^{n-i}.$$

Generalizing this, a tree T exceeds height k if more than c points fall in any region managed by a cell at level k . The splitting probabilities for different cells are unfortunately not independent, as can be seen with the pigeonhole principle: If $n = 5, k = 1$ and $c = 1$, a split needs to happen somewhere, but all individual non-splitting probabilities are above 0 and so is their product.

However, an argument borrowed from hashing helps: Let $c = 1$ to allow only a single point per leaf cell. The number of excess points in a leaf cell is then equivalent to the number of hash collisions with n elements and n buckets. For this scenario it is shown that the expected maximum number of collisions in any bucket is $O(\log n / \log \log n)$ [4]. This will result in at most $O(\log n / \log \log n)$ additional levels and an expected total height of $O(\log n)$. \square

C Dynamic Model

Theorem 3.4 *Node movement preserves distribution of angular and radial coordinates: $F_X(r) = F_X(\text{scale}(r))$ for $0 \leq r \leq R$ and $F_\Phi(\text{rotated}((\phi, r))) = F_\Phi(\phi)$ for $0 \leq \phi \leq 2\pi$.*

Proof. As introduced in Section 2.2 of the paper, the radial coordinate r is sampled from a distribution with density $\sinh(r)/(\cosh(R) - 1)$ and τ_r is the unscaled radial movement parameter. We introduce additional random variables X, Y, Z for each step in Algorithm 3.2 of the paper, each is denoted with the upper case letter of its equivalent. We have an additional random variable Q denoting the pre-movement radial coordinate. The other variables are defined as $X = \cosh(Q)$, $Y = X + \tau_r$ and $Z = \text{acosh}(Y) = \text{scale}(Q, \tau_r)$. With $F_Q(r) = (\cosh(r) - 1)/(\cosh(R) - 1)$, we get a uniform distribution over x : $F_X(x) = F_Q(\text{acosh}(x)) = (\cosh(\text{acosh}(x)) - 1)/(\cosh(R) - 1) = (x - 1)/(\cosh(R) - 1)$. The cumulative density functions are $F_Y(r) = F_X(r - \tau_r) = (r - \tau_r - 1)/(\cosh(R) - 1)$ and $F_Z(r) = F_Y(\cosh(r)) = (\cosh(r) - \tau_r - 1)/(\cosh(R) - 1)$.

The distributions of Q and Z only differ in the constant addition of $\tau_r/(\cosh(R) - 1)$. Every $(\cosh(R) - 1)/\tau_r$ steps, the radial movement reaches a limit (0 or R) and is reflected, causing τ_r to be multiplied with -1 . On average, τ_r is thus zero and $F_Q(r) = F_Z(r)$.

A similar argument works for the rotational step: While the rotational direction is unchanged, the change in coordinates is balanced by the addition or subtraction of 2π whenever the interval $[0, 2\pi)$ is left, leading to an average of zero in terms of change. \square

C.1 Time Complexity We first discuss the movement of a single node with j edges and then proceed to move k nodes with l edges. Movement of a single node v consists of the following steps:

1. $S_{\text{before}} = \text{neighborhood of } v \text{ at old coordinates}$
2. update coordinates
3. IF new coordinates are in different quadtree cell
4. delete v from old cell
5. insert v into new cell
6. $S_{\text{after}} = \text{neighborhood of } v \text{ at new coordinates}$
7. removed edges = $S_{\text{before}} \setminus S_{\text{after}}$
8. new edges = $S_{\text{after}} \setminus S_{\text{before}}$

As seen in Equation 3.5 of the paper, a range query has a time complexity of $O((1 + j) \log n)$. The deletion of a point from the quadtree requires height = $\log_4(n)$ steps to find the responsible leaf node, then at most one *coarsening* operation if the number of elements in this leaf node falls under a threshold. The coarsening operation copies at most c elements, thus we have:

$$(3.18) \quad T(\text{Deletion}) \in O(\log_4(n/c) + c) = O(\log n).$$

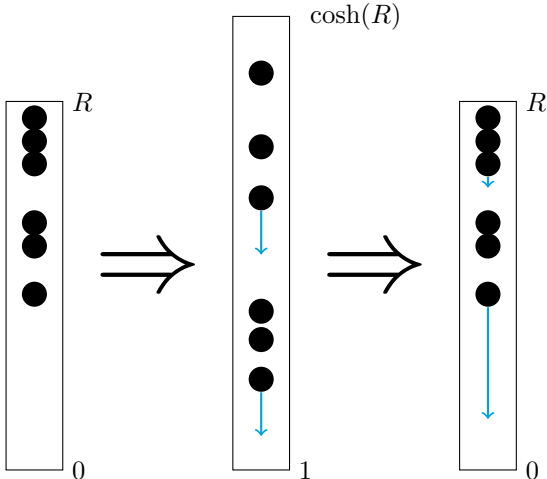
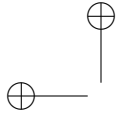


Figure 6: For each movement step, radial coordinates are mapped into the interval $[1, \cosh(R))$, where the coordinate distribution is uniform. Adding τ_r and transforming the coordinates back results in correctly scaled movements.

This results in a total time complexity of $2 \cdot T(\text{RQ}(v)) + T(\text{Deletion}(v)) + T(\text{Insertion}(v)) + T(\text{SetDifference}(j))$. The neighborhood set difference for j edges can be obtained with an amortized complexity of $O(j)$ using a hash set. With the $O(\log n)$ time complexity for insertion (Equation 3.4 of the paper) and $O((1+j) \log n)$ for the range query, we have

$$(3.19) \quad T(\text{Movement}) = O((1+j) \log n).$$

When moving k nodes with l edges, first all neighborhood sets are computed at the old positions, then all quadtree updates are applied and finally the neighborhood sets at new positions are computed. This reordering does not influence the asymptotical running time, which is $O((k+l) \log n)$.

D Parameter studies with varying threshold factor t

The node dispersion α starts from 0.8 (lower values tend to result in a complete graph) and goes up to 3. The threshold factor t is in the range $(0.1, 1)$. Choosing values over 1 results in very dense graphs quickly, a complete graph is generated for $t = 2$.¹ The results are visualized in Figure 7.

References

¹Each node has a hyperbolic distance to the origin of at most R . Due to the triangle inequality, the maximal distance between two nodes is at most $2R$.

- [1] William Aiello, Fan Chung, and Linyuan Lu. A random graph model for massive graphs. In *Proceedings of the thirty-second annual ACM symposium 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] James W Anderson. *Hyperbolic geometry; 2nd ed.* Springer undergraduate mathematics series. Springer, Berlin, 2005.
- [4] Ken Bogart and Cliff Stein. Discrete mathematics in computer science. https://math.dartmouth.edu/archive/m19w03/public_html/Section6-5.pdf.
- [5] Marián Boguñá, Fragkiskos Papadopoulos, and Dmitri Krioukov. Sustaining the internet with hyperbolic mapping. *Nature Communications*, (62), September 2010.
- [6] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys (CSUR)*, 38(1):2, 2006.
- [7] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-mat: A recursive model for graph mining. In *SDM*, volume 4, pages 442–446. SIAM, 2004.
- [8] Sergei N Dorogovtsev and José FF Mendes. *Evolution of networks: From biological nets to the Internet and WWW*. Oxford University Press, 2003.
- [9] Joachim Gehweiler and Henning Meyerhenke. A distributed diffusive heuristic for clustering a virtual P2P supercomputer. In *Proc. 7th High-Performance Grid Computing Workshop (HGCW'10) in conjunction with 24th Intl. Parallel and Distributed Processing Symposium (IPDPS'10)*. IEEE Computer Society, 2010.
- [10] Luca Gugelmann, Konstantinos Panagiotou, and Ueli Peter. Random hyperbolic graphs: Degree sequence and clustering - (extended abstract). In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, volume 7392 of *Lecture Notes in Computer Science*, pages 573–585. Springer, 2012.
- [11] S Louis Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. i. *Journal of the Society for Industrial & Applied Mathematics*, 10(3):496–506, 1962.
- [12] Robert Kleinberg. Geographic routing using hyperbolic space. In *INFOCOM*, pages 1902–1909, 2007.
- [13] Tamara G Kolda, Ali Pinar, Todd Plantenga, and C Seshadhri. A scalable generative graph model with community structure. *arXiv preprint arXiv:1302.6636*, 2013.
- [14] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *Physical Review E*, 82:036106, Sep 2010.
- [15] Jérôme Kunegis. Wikipedia links, en network dataset – KONECT, July 2014. (http://konect.uni-koblenz.de/networks/wikipedia_link_en).

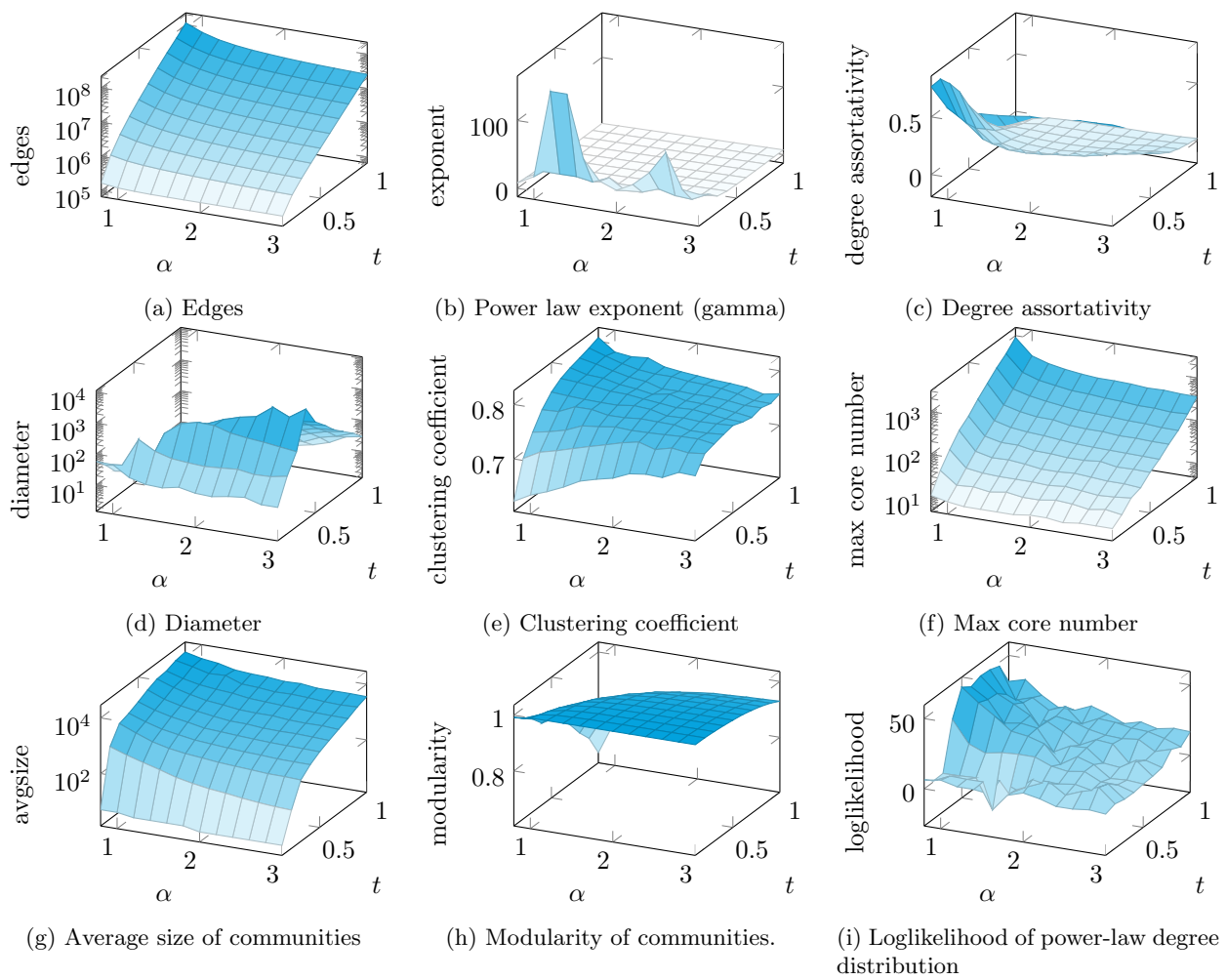
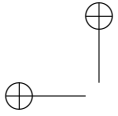
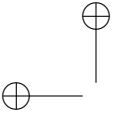


Figure 7: Properties of graphs generated with the hyperbolic generator for 100,000 nodes and different values for the parameters α and t . Higher values of α cause fewer edges to be generated, as do lower values of t . With too few edges, the graphs become disconnected and noise of other property measurements increases.

- 
- [16] M. E. J. Newman. The Structure and Function of Complex Networks. *SIAM Review*, 45:167–256, January 2003.
 - [17] Mark Newman. *Networks: an introduction*. Oxford University Press, 2010.
 - [18] Chiara Orsini and Rodrigo Aldecoa. Hyperbolic generator. (<https://github.com/named-data/Hyperbolic-Graph-Generator/releases/tag/v.1.0.1>).
 - [19] A Rényi P. Erdős. On the Evolution of Random Graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, 1960.
 - [20] Fragkiskos Papadopoulos, Dmitri Krioukov, Marián Boguñá, and Amin Vahdat. Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.
 - [21] David Papo, Javier M. Buldú, Stefano Boccaletti, and Edward T. Bullmore. Complex network theory and the brain. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1653), 2014.
 - [22] 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.
 - [23] C. Seshadhri, Ali Pinar, and Tamara G. Kolda. The similarity between stochastic kronecker and chun-gu graph models. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 1071–1082, 2011.
 - [24] Christian L. Staudt and Henning Meyerhenke. Engineering high-performance community detection heuristics for massive graphs. In *proceedings of the 2013 International Conference on Parallel Processing*. Conference Publishing Services (CPS), 2013. Updated full version at <http://arxiv.org/abs/1304.4453>.