communications physics

A Nature Portfolio journal

Article

https://doi.org/10.1038/s42005-025-02122-0

Symmetry-driven embedding of networks in hyperbolic space

Simon Lizotte [1,2], Jean-Gabriel Young [1,3,4] & Antoine Allard [1,2,4]

Hyperbolic models are known to produce networks with properties observed empirically in most network datasets, including heavy-tailed degree distribution, high clustering, and hierarchical structures. As a result, several embedding algorithms have been proposed to invert these models and assign hyperbolic coordinates to network data. Current algorithms for finding these coordinates, however, do not quantify uncertainty in the inferred coordinates. We present BIGUE, a Markov chain Monte Carlo (MCMC) algorithm that samples the posterior distribution of a Bayesian hyperbolic random graph model. We show that the samples are consistent with current algorithms while providing added credible intervals for the coordinates and all network properties. We also show that some networks admit two or more plausible embeddings, a feature that an optimization algorithm can easily overlook.

Embedding enables us to comprehend abstract objects and conduct rigorous quantitative analyses of their similarities, differences, and structural characteristics. In the complex network context, embeddings usually consist of vertex coordinates in a latent space. A rich literature has shown that embedding in hyperbolic spaces[1], in particular, can visually emphasize the important parts of large networks[2], but also, when used as the basis for a random graph model, naturally reproduce many common network properties such as the community structure[3–9], a power-law degree distribution[10,11], a non-vanishing clustering coefficient[10–15], a fractal structure[16–18] and the sparsity of connections[14]. Furthermore, since hyperbolic geometry closely approximates the shortest paths of graphs via geodesics[10,19], hyperbolic embeddings have been used to design efficient routing protocols[20,21].

Considerable effort has gone into representing a given graph using latent spaces[22–26]. Many studies focus on placing the vertices such that the graph distance between the vertices is closely approximated by the hyperbolic geodesics[27–31]. From this perspective, trees[32] can be embedded in hyperbolic space up to an arbitrarily small error, enabling perfect greedy routing. This is due to the exponentially growing volume in the hyperbolic space, which can be related to the branching factor of trees[10]. The embedding task was also tackled with dimension reduction techniques from machine learning[26,33–35] and with the maximum likelihood estimation of various hyperbolic random graph models[20,36–45]. Other methods allow the embedding of directed graphs[46–48], weighted graphs[49,50] and graphs to the D-dimensional hyperbolic space with[51] or without vertex features[52].

Existing methods have, however, largely overlooked uncertainty: They return a single embedding without allowance for error or acknowledging the existence of other solutions. Yet, this information is crucial as perturbations to the embedding affect edge prediction[53] and can lead to different graph representations.

As a first step towards addressing this problem, we propose a Bayesian embedding algorithm that quantifies uncertainty rigorously. Related methods have proven useful in network science, where they have been used to reconstruct networks from noisy measurements[54,55], predict the existence of edges[56], infer higher-order interactions from network data[57] and pairwise observations[58], and perform community detection[59,60].

Our proposed algorithm, BIGUE (Bayesian Inference of a Graph's Unknown Embedding), uses Markov chain Monte Carlo (MCMC) sampling to generate embeddings compatible with a giv en graph; see Fig. 1. We show how these samples can be used to calculate credible intervals, interpreted as error bars, for vertex positions, properties of the embedding, and model parameters. We also show that many qualitatively different embeddings are often compatible with an observed graph—a phenomenon known as multimodality.

Bayesian approaches hyperbolic embeddings of networks have already been studied[61], but the results therein suggest that the proposed MCMC algorithm has poor mixing—a common problem many MCMC methods face when the posterior distribution is complex and multimodal[62–70]. We identify community structure[9] as a source of multimodality and introduce a set of cluster-based transformations to improve the exploration of the posterior distribution. Our approach draws on previous literature in which community structure was used to guide greedy embedding algorithms[44,71]. We show that these transformations drastically improve the mixing of the MCMC algorithm, even if the community structure is weak. In doing so, we

[1]Département de physique, de génie physique et d'optique, Université Laval, Québec (Québec), G1V 0A6, Canada. [2]Centre interdisciplinaire en modélisation mathématique, Université Laval, Québec (Québec), G1V 0A6, Canada. [3]Department of Mathematics and Statistics, University of Vermont, Burlington, VT 05405, USA. [4]Vermont Complex Systems Institute, University of Vermont, Burlington, VT 05405, USA. e-mail: antoine.allard@phy.ulaval.ca

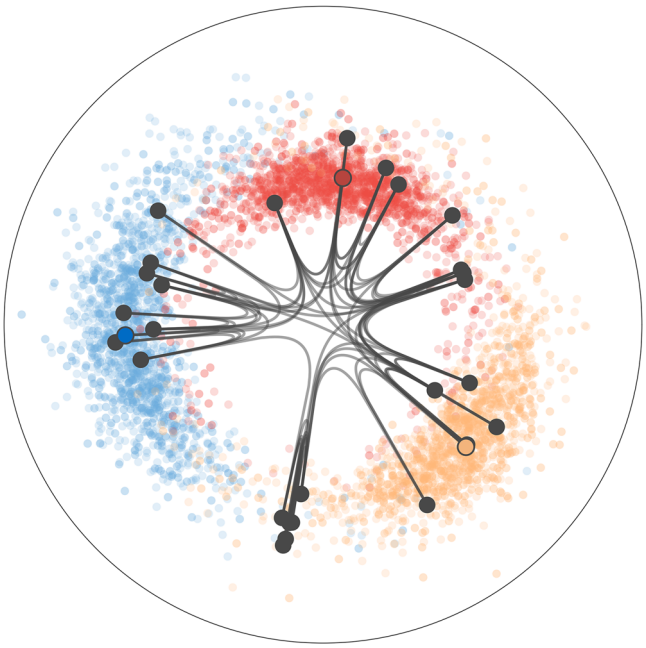


Fig. 1 | Probabilistic hyperbolic embedding of a synthetic graph with BIGUE. Hyperbolic coordinates are obtained using a transformation between the $\mathbb{H}^2$ and the $\mathbb{S}^1$ models described in Supplementary Note 2. Black points and dark-colored points are the median coordinates of each vertex. Light-colored points are the 2000 sampled positions for the three highlighted vertices. Lines are edges drawn using hyperbolic geodesics. The synthetic graph of 30 vertices is generated with the $\mathbb{S}^1$ likelihood, with angular coordinates drawn from their prior, $\beta = 2.5$ and the κ parameters drawn from a Pareto distribution of exponent -2.5 truncated over the interval (4, 10).

make a tradeoff between speed and accuracy—greedy methods find a single solution rapidly, while our MCMC methods explore a large space more slowly. In fact, our implementation of BIGUE does not scale well beyond a few hundred vertices. Thad said we show that the cluster transformations powering BIGUE improve mixing and thus make it more scalable than alternative MCMC algorithms. Furthermore, we provide evidence that the posterior distributions of hyperbolic graph models are not Gaussian in practice, thereby motivating the need for MCMC as opposed to faster but less accurate methods like variational inference.

Methods

Bayesian hyperbolic embedding

The Bayesian framework aims to infer parameters—here the embedding—from some data—the graph. In this section, we define the likelihood and prior distributions that lead to the posterior of the model, and highlight unique challenges that do not arise when searching for a single embedding.

Model definition. Let $G = (V, E)$ be the graph we aim to embed, with V and E being the sets of its vertices and of its edges, respectively. Let $\mathcal{G}$ be the random graph used to perform the embedding. We use the circular probabilistic model $\mathbb{S}^1$ to describe the graph G[15]; it is nearly equivalent to the hyperbolic plane model $\mathbb{H}^2$[10] (see Supplementary Note 2), but it will turn out to facilitate inference. In the $\mathbb{S}^1$ model, the probability that an edge (u, v) exists between vertices u and v is a function of the angular coordinates $\theta = (\theta_w)_{w\in V}$ of the vertices where each $\theta_w \in [-\pi, \pi)$, of the parameters $\kappa = (\kappa_w)_{w\in V}$ where $\kappa_w > 0$ controls the degree of vertex w, and of an inverse temperature $\beta > 1$ which controls the amount of clustering. This probability is defined as

$$\mathbb{P}[a_{uv} = 1|\theta, \kappa, \beta] = \left(1 + \left(\frac{d(\theta_u, \theta_v)}{\mu\kappa_u\kappa_v}\right)^{\beta}\right)^{-1}, \tag{1}$$

where $a_{uv} = a_{vu}$, $u \neq v$, is any element of the adjacency matrix of a graph realized by $\mathcal{G}$, $d(\theta_u, \theta_v) = R\Delta(\theta_u, \theta_v)$ is the arc length between vertices u and v, $\Delta(\theta_u, \theta_v) = \pi - |\pi - |\theta_u - \theta_v||$ is the angular separation, and $R = |V|/2\pi$ is the radius of the circle on which vertices are embedded. We fix the constant $\mu = \beta \sin(\pi/\beta)/(2\pi\mathbb{E}[\tilde{\kappa}])$ so that the parameters κ match the degrees $\mathbb{E}[K|\kappa] = \kappa$ in expectation in the limit of large graphs, where $\tilde{\kappa}$ represents the random variable for the parameter κ_w of any vertex w (they are i.i.d.) and $K|\kappa$ is the random degree of a vertex of parameter κ when all vertices are independently and uniformly distributed on the circle[10]. These choices for R and μ are without lost of generality.

The complete graph is modeled using conditional independence for the edges, leading to the likelihood

$$\begin{aligned}\mathbb{P}[\mathcal{G} = G|\theta, \kappa, \beta] &= \prod\nolimits_{(u,v)\in E} \mathbb{P}[a_{uv} = 1|\theta, \kappa, \beta] \\ &\quad \times \prod_{(u,v)\notin E} (1 - \mathbb{P}[a_{uv} = 1|\theta, \kappa, \beta]) \\ &= \prod_{(u,v)\in V2} \left(1 + \left(\frac{d(\theta_u, \theta_v)}{\mu\kappa_u\kappa_v}\right)^{\beta^{\pm}_{uv}}\right)^{-1},\end{aligned} \tag{2}$$

where $V2$ is the set of all combinations of two vertices, and $\beta^{\pm}_{uv}$ equals β if $a_{uv} = 1$ and equals $-\beta$ otherwise.

Applying Bayes' rule yields the following posterior density for the embedding (i.e., angular coordinates and parameters)

$$p(\theta, \kappa, \beta|G) = \frac{\mathbb{P}[\mathcal{G} = G|\theta, \kappa, \beta]p(\theta, \kappa, \beta)}{\mathbb{P}[\mathcal{G} = G]}. \tag{3}$$

The likelihood $\mathbb{P}[\mathcal{G} = G|\theta, \kappa, \beta]$ is given by Eq. (2). To complete the specification of Eq. (3), we use independent prior densities

$$p(\theta, \kappa, \beta) = p(\beta)\prod_{w\in V} p(\kappa_w)p(\theta_w), \tag{4}$$

for ease of modeling, though our sampling algorithm does not depend on this simplifying assumption. Previous literature has shown that inverse temperatures are typically small; we thus choose a truncated half-normal distribution for β,

$$p(\beta) \propto \mathbb{1}[\beta > 1] \exp\left\{-\frac{(\beta - \beta_0)^2}{2\sigma^2}\right\} \tag{5}$$

with $\beta_0 = 3$ and $\sigma = 2$, and where $\mathbb{1}[\cdot]$ is the indicator function (equals 1 is its argument is true, and 0 otherwise). Since known networks tend to have a heavy-tailed degree distribution, we use a half-Cauchy prior

$$p(\kappa_w) \propto \mathbb{1}[\kappa_w > \varepsilon] \frac{2}{\pi\gamma\left(1 + (\kappa_w/\gamma)^2\right)}, \quad \forall w \in V \tag{6}$$

with $\gamma = 4$ and $\varepsilon = 10^{-10}$; this allows for large variations in the estimated values of κ. Finally, because the angular coordinates are defined on a bounded space and we do not want to favor an embedding direction a priori, we use a uniform prior

$$p(\theta_w) = \frac{1}{2\pi} \tag{7}$$

except for the two vertices with the highest degree, whose position is restricted (the reason is technical and has to do with avoiding degeneracy in the posterior; see below). With these choices, the posterior distribution is proper and can express correlations between the parameters even if the priors are independent. The normalization constant $\mathbb{P}[\mathcal{G} = G]$ can be obtained by integration in principle, though it is not needed here; see the Sampling section below.

Symmetries and automorphisms. The $\mathbb{S}^1$ model exhibits fundamental symmetries that give rise to identifiability problems. While these symmetries are often overlooked when using maximization algorithms (which produce single solutions), they must be addressed in the Bayesian context. Indeed, our goal is to compute a distribution over embeddings, requiring us to distinguish between embeddings that differ merely by trivial reparametrizations (such as coordinate rotations) and those that represent genuinely distinct configurations. In this section, we examine the two classes of symmetries inherent in the model and present our approach for handling them.

First, the model inherits the symmetries of the circle $\mathbb{S}^1$. Since the distance d is invariant to rotations and reflections $\theta_w \mapsto -\theta_w \ \forall\, w$, the edge probabilities are preserved in the likelihood, and an infinite number of equivalent embeddings exist. Maximization algorithms break this symmetry at random, but we need to proceed with more caution when we move to a distribution over embedding. Without loss of generality, we fix the angular coordinate of the highest-degree vertex u to $\theta_u = 0$ to handle rotational symmetries. We remove the reflection symmetry by restricting the angular coordinate of the second highest-degree vertex v to $\theta_v \in [0, \pi]$.

Second, a graph is inherently labeled by its vertices, but any vertex relabelling that preserves the edges—an automorphism—leaves the likelihood invariant since the distances between connected and unconnected vertices are unchanged. For example, if a transformation $u \leftrightarrow v$ exists such that each $(u, w) \in E \Leftrightarrow (v, w) \in E$ for any vertex w other than u and v, then we can also exchange $\kappa_u \leftrightarrow \kappa_v$ and $\theta_u \leftrightarrow \theta_v$ to preserve the likelihood.

The automorphisms cannot easily be avoided in the model and while the restriction on the highest-degree vertices helps with sampling, it does not guarantee an optimal compatibility between the embeddings. For instance, fixing $\theta_u = 0$ implies no uncertainty on θ_u and restricting some v to $\theta_v \in [0, \pi]$ can affect $|V| - 2$ vertices instead of only θ_v. To have the most coherent embedding sample, we opt for a post-processing alignment where we minimize the sum of squared angular separations to some arbitrary reference embedding over the automorphisms found using Nauty[72], the two possible reflections (reflection or no reflection) and the rotations $[0, 2\pi)$.

In what follows, we will compare the embeddings obtained using this Bayesian framework with the ones obtained using Mercator[36]—a coordinate ascent algorithm that maximizes the Eq. (2). When embedding synthetic graphs generated with the $\mathbb{S}^1$ model, we will also use the original input parameters (θ, κ, β) as the "ground truth" embedding.

Sampling

With the model in place, we can now generate embeddings by calculating expectations over the posterior density of Eq. (3). For example, the expected position of a vertex can be calculated as

$$\mathbb{E}[\Theta_u | \mathcal{G} = G] = \int \theta_u \, p(\theta, \kappa, \beta | G) \, \mathrm{d}\theta \, \mathrm{d}\kappa \, \mathrm{d}\beta, \tag{8}$$

where Θ_u is the random variable of the angular position of vertex u, and similar integrals can be defined for all quantities of interests, like confidence interval for the positions. However, these integrals have, as far as we can tell, no closed-form solution, and we thus turn to sampling approximations computed using Markov Chain Monte Carlo (MCMC) algorithms. These algorithms tend to be slow (see Supplementary Note 1 for a discussion), but we show in the Multimodality section below that there is evidence for their necessity to adequately tackle this problem.

Unfortunately, we have found that standard MCMC algorithms are not well-adapted to the $\mathbb{S}^1$ model. Indeed, for most graphs G, the posterior distribution has a complicated geometry defined by several steep barriers that separate the posterior distribution in a multitude of small regions. This is the case even for a small graph of 30 vertices, as Fig. 2 shows: Changing the embedding coordinate of a single vertex reveals multiple local maxima of the likelihood (and thus posterior density). The model predicts that two vertices with nearby embedding coordinates should be connected with high probability. In the extreme case where two disconnected vertices have the same angular coordinate, the likelihood approaches 0, causing the sharp dips shown in the Figure. Furthermore, the gradients are undefined due to their dependence on the distance function in Eq. (1). Since the posterior geometry is filled with these barriers for sparse and large graphs (the number of modes increases with the number of disconnected pairs of vertices), hyperbolic embedding can be challenging for standard MCMC algorithms.

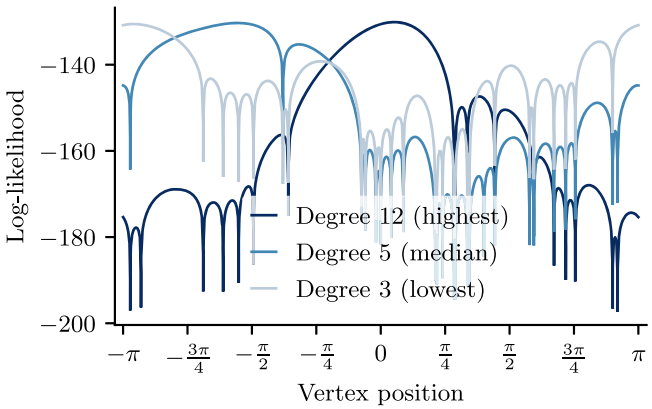


Fig. 2 | Log-likelihood of the model when a single vertex is moved along the circle. Each dip is a divergence that occurs when a vertex is positioned at the same position as a disconnected vertex. These divergences form barriers in the objective function landscape (log-likelihood, posterior distribution), which is one of the reasons why hyperbolic embedding is a difficult problem. The likelihood is computed on the ground truth embedding of the synthetic graph of Fig. 1. Colors indicate the degree of the moved vertex (lowest degree in light blue, median degree in blue and highest degree in dark blue).

In what follows, we first shows that both a simple random walk algorithm and a sophisticated gradient-based algorithm cannot adequately sample the posterior of a small graph as a result. Then, we demonstrate that our proposed algorithm, BIGUE, can avoid this problem by supplementing cluster-based transformations to a random walk.

Baseline algorithms. We first tested a naive random walk MCMC (RW) algorithm (details in Supplementary Methods 1) on a small synthetic graph with known ground truth embedding coordinates (shown in Fig. 1). We quantified our results using the normalized autocovariance, the effective number of samples S_{eff}, and $\hat{R}$—a measure of mixing (see Supplementary Methods 2 for details). Furthermore, we tested convergence by initializing the algorithm (i) at the ground truth and (ii) with a simple initialization strategy that can be applied to observational data. Figure 3 illustrates that the RW has a high normalized autocovariance and it very slowly reaches the typical set—the part of the distribution that holds a significant probability mass and where good algorithms should naturally spend most of their time[73].

For our second test, we wrote the model in Stan, a probabilistic programming language that implements a dynamic Hamiltonian Monte Carlo (HMC) algorithm[74] for sampling from arbitrary Bayesian models. Dynamic HMC runs slowly because each sampling iteration involves integrating a large system of differential equations. Nonetheless, the promise of high-quality samples makes this option worth exploring.

Unfortunately, the barriers shown in Fig. 2 are problematic for a purely gradient-based method such as HMC, and Stan indeed reports gradient divergences, which signal incorrect sampling. This issue can be partially mitigated by using an approximate likelihood (described in Supplementary Note 3) that smoothens these barriers at the cost of a slightly distorted posterior. However, even with this modification, the algorithm is unable to explore the sampling space sufficiently fast (we reach the maximum tree depth easily[74]).

Stan's default parameters can be modified to allow for better exploration (increased tree depth), but even then, a critical problem remains

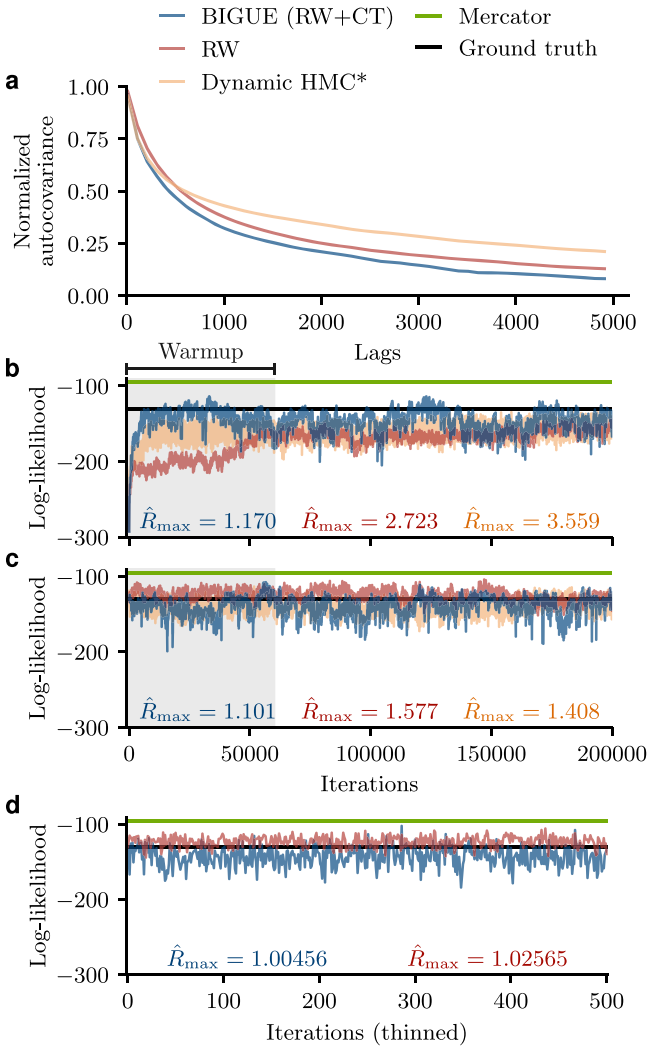


Fig. 3 | Statistics of Markov chain Monte Carlo algorithms when embedding the synthetic graph of Fig. 1. Results for the random walk algorithm (RW) and for our method (BIGUE) that uses both cluster transformations and a random walk (RW +CT) are displayed in red and blue, respectively. The orange curves show the results for dynamic Hamiltonian Monte Carlo (HMC) for the differentiable $\mathbb{S}^1$ model described in Supplementary Note 3. The green line displays the maximum log-likelihood obtained from Mercator after 10 runs, each with 10 refinements[36] and the black line is the ground truth embedding's log-likelihood. **a** Normalized autocovariance averaged over all parameters and chains at different lags. **b**, **c** Traceplot of log-likelihood of a simulated Markov chain initialized without (**b**) and with (**c**) access to ground truth information. When the ground truth is unknown, we initialize $\kappa_u = \deg(u)$ and draw the other parameters from their priors. **d** Traceplot of log-likelihood of a Markov chain after thinning the chain shown in panel b. The blue line of this panel is the sample shown in Fig. 1. In each case, we compute $\hat{R}_{\max}$, the maximum potential scale reduction factor for all parameters after the iterations removed from the warm-up (grey part of traceplots)—values close to 1 are desirable. Four independent chains were simulated to compute $\hat{R}$ (shown on each panel with the color corresponding to the sampling algorithm) and the autocovariance, but only one is displayed (representative of the four).

and seems insurmountable: after a large number of iterations, the sampled distribution differs depending on the initialization, a signal that the mixing is poor; see Fig. 3. Initializations at the ground truth yield samples of higher quality than the RW, but random initializations do not converge to the typical set even after 200,000 iterations. This mixing issue is likely caused by the complicated and non-convex geometry of the posterior distribution, which motivates the modifications we propose next.

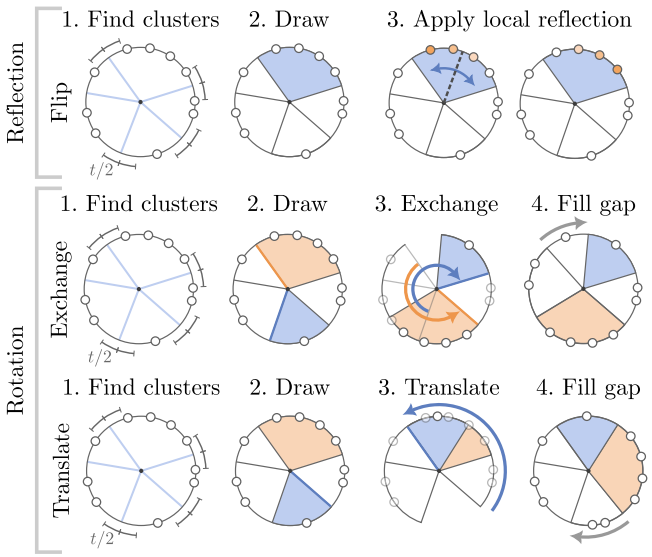


Fig. 4 | Summary of the cluster transformations used in BIGUE. The flip transformation targets the reflection symmetries, and the exchange and translate transformations target the rotation symmetries.

Cluster transformation MCMC. We can improve upon dynamic HMC and RW by noticing that the connection probability is a decreasing function of the angular separation, such that groups of vertices at a large enough distance from one another are essentially independent. As a result, each region is akin to a smaller and simpler embedding problem, which is the basis for the approach of Wang et al.[44,71]. (This is also the basis for the idea of renormalization in the $\mathbb{H}^2$ hyperbolic model in which groups of vertices are coarse-grained while partially preserving the structural integrity of the graph[1,17].) Recalling that the likelihood is invariant to reflections and rotations, we propose to apply these transformations at the level of groups or clusters of vertices; c.f. Fig. 4.

The first step for these transformations is to partition vertices into roughly independent clusters given the angular positions. This is done by grouping vertices with an angular separation below some randomly sampled threshold in clusters. We use this naive partitioning scheme instead of an optimization procedure (e.g. modularity maximization[75]) for technical reasons discussed in Supplementary Methods 1.

Once we have selected clusters, we apply one of the following transformations. The first transformation, named flip, applies a reflection on the vertices of the selected cluster. The second transformation, exchange, swaps the relative positions of two selected clusters. The third transformation, translate, moves the first selected cluster to the relative position of the second cluster. The first transformation explores local reflection symmetries, while the two others explore rotation symmetries. In each case we select the involved cluster(s) uniformly at random from all clusters.

Our proposed algorithm, BIGUE, combines these random cluster transformations (CT) with the random walk (RW) baseline. Cluster transformations explore the mesoscale structure of the embedding, while the random walk fine-tunes the embedding coordinates, including the parameters κ. Each sampling iteration is either a randomly selected CT or a RW, and we calibrate the transition probabilities to the posterior distribution using the Metropolis-Hastings algorithm; see Supplementary Methods 1 for the technical details. (One could also combine random cluster transformations with dynamic HMC, but the high computational cost of dynamic HMC turns out to outweigh its benefits. A naive random walk provides cheaper and sufficient fine-tuning in our experiments).

Figure 3 shows that BIGUE has the lowest autocovariance and reaches the typical set faster the other algorithms. The fluctuations at equilibrium are also larger, which suggests that it explores the posterior more efficiently. Furthermore, BIGUE's maximum potential reduction factor $\hat{R}_{\max}$ is the

lowest, suggesting that it has better mixing. However, the three algorithms yield $\hat{R}$ values above the recommended 1.01 threshold[76].

A direct fix to high normalized autocovariance and potential reduction factor is to thin the chains, that is, only use every k-th sample point of the chain. Guided by Fig. 3a, we set $k = 10,000$ for BIGUE and the RW and obtain the chains of panel d. We now see that BIGUE's chains have $\hat{R}_{\max}<1.01$ while those of the RW do not. Moreover, the fluctuations of the RW are still smaller, indicating, again, that the sampling space is better explored by BIGUE.

From the samples of Fig. 3, we also compute the effective sample size of the chains, which is related to the estimation error in the Markov Chain central limit theorem. The median effective sample sizes S_{eff} across all parameters were, for BIGUE: 486, 462, and 1814 (in b–d respectively). For the random walk MCMC, we obtained medians of $S_{\text{eff}} = 248$, 384, and 1390, and for dynamic HMC, they were equal to 593 and 347. This provides further evidence that cluster transformation benefit the random walk algorithm. Autocovariance, $\hat{R}$ and S_{eff} and their adaptation to periodic random variables are explained in Supplementary Methods 2.

In light of these simulations, we conclude that BIGUE outperforms both dynamic HMC and the RW in all metrics, and that in general, we expect these algorithms to perform equivalently to BIGUE at best. For this reason, the remainder of the paper uses BIGUE to sample the posterior distribution.

It may seem surprising that BIGUE relies on clusters in the embedding while the vertices are uniformly distributed according to prior density of the generative model. However, the clusters found need not be well separated to be useful in sampling. This is due to the locality of connections: If some pairs of vertices are sufficiently distant, then we can reasonably attempt to separate these vertices as they might not be connected anyway. Conversely, vertices that are closeby in the best embedding will tend to be connected and can thus be moved as a unit instead of one at a time.

We note that Fig. 3 highlights one counterintuitive aspect of continuous random variables[73]: The neighborhood of the likelihood maximum accounts for a small fraction of the probability mass because probabilities are obtained through integration with respect to the Lebesgue measure. While the log-likelihood maximum (found by Mercator) has a high probability density, its surroundings cover a larger volume (the typical set) and hold a much larger probability mass. Hence, our MCMC sample concentrates away from the maximum likelihood, as expected. (That said, we note that cluster transformations could easily be added to maximization routines if a point estimate was the goal, something we leave for future work).

Results

Having a reliable algorithm to sample embeddings from the posterior distribution, we can now estimate embeddings and integrals like Eq. (8). This section illustrates how the resulting Bayesian approach compares to maximum likelihood estimation both on a synthetic graph and various empirical networks.

Embedding and network properties

We first compare our Bayesian model to Mercator on the synthetic graph used in Fig. 3. Figure 5 shows what such an analysis might look like, using credible intervals for the embedding coordinates and their transformations, computed with samples drawn from the chains depicted in Fig. 3d.

We see in Fig. 5a that Mercator's estimation generally lies within the estimated credible intervals, revealing some leeway in the coordinates compatible with the data G. The ground truth value is sometimes not within the interval because the graph is small (see Supplementary Fig. 1, where the graph contains 100 vertices, and the coordinates are recovered almost perfectly). As expected, for most vertices u, the κ_u parameters concentrate on the vertices' original degrees (Fig. 5b), but there are discrepancies. Note that the relationship $\mathbb{E}[K|\kappa_u] = \deg(u)$, mentioned in the Model definition section, is valid assuming an infinite number of vertices. This is why, for a high-degree vertex u in a small graph, κ_u is larger to compensate for the fact that there is a finite number of vertices. Figure 5c also shows that the posterior covers the original β value, although it is in a somewhat low probability region. The posterior distribution agrees with Mercator and places a higher likelihood than the ground truth on β being a bit under 2—

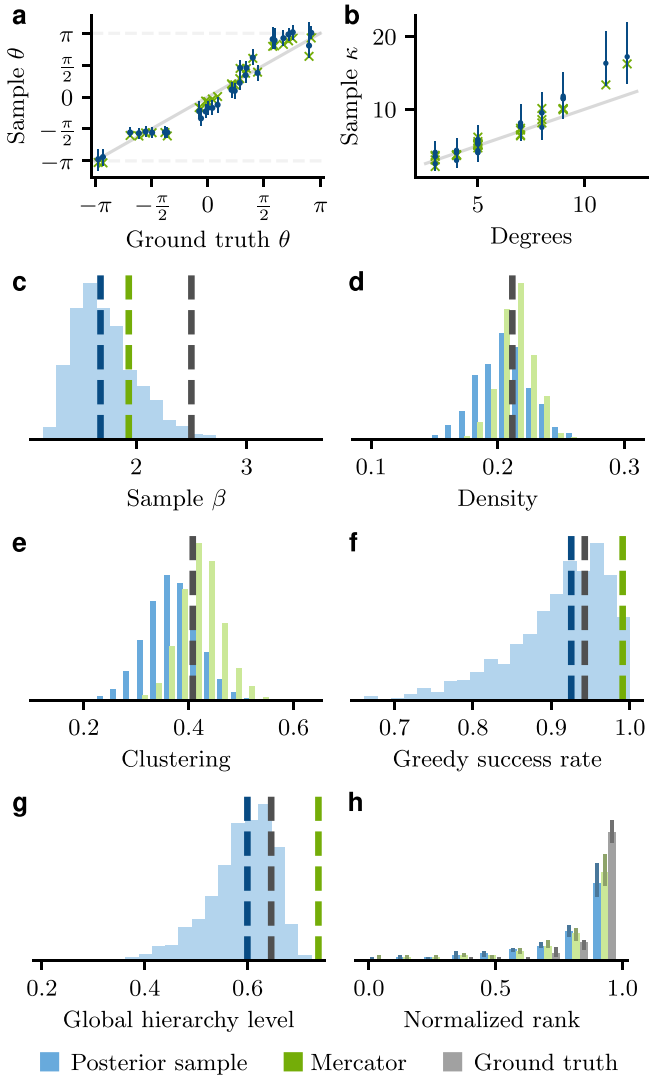


Fig. 5 | Posterior estimates and posterior predictive distribution for the synthetic graph used in Fig. 3. a Angular coordinates θ. **b** Parameters κ. **c** Inverse temperature β. **d** Density. **e** Clustering (the transitivity). **f** Greedy routing success rate. **g** Global hierarchy level[77]. **h** Normalized rank of the removed edges' existence probability among all the unconnected pairs (see below). In each case, 2000 total embeddings were sampled from four independent chains. All shades of blue, green and gray display the values for the posterior sample, Mercator and the ground truth, respectively. In panels (**a**, **b**), points denote the median, and the error bars cover the interquartile range. In panels (**c**–**g**), the vertical dotted lines show point estimates computed with the Mercator and ground truth embeddings. In panels (**d**, **e**), we generated graph samples for Mercator and the ground truth by conditioning the model's likelihood on point estimates of the embedding. In panels (**c**, **f**, **g**), the blue dashed line is the median of the posterior sample. In (**h**), we test the algorithm on a link prediction task[91], and report the rank of removed edges as predicted by the likelihood. Pairs of embedding and graphs of 30 vertices were sampled from the prior. For each graph, 5% of the edges were then randomly removed (4 to 5 edges), and the normalized ranks of removed edges were calculated and binned. (Disconnected graphs were rejected for compatibility with Mercator). The figure reports the median and interquartile range of the normalized rank frequencies calculated across graphs, removals, and samples. The AUC values are shown in Supplementary Note 5.

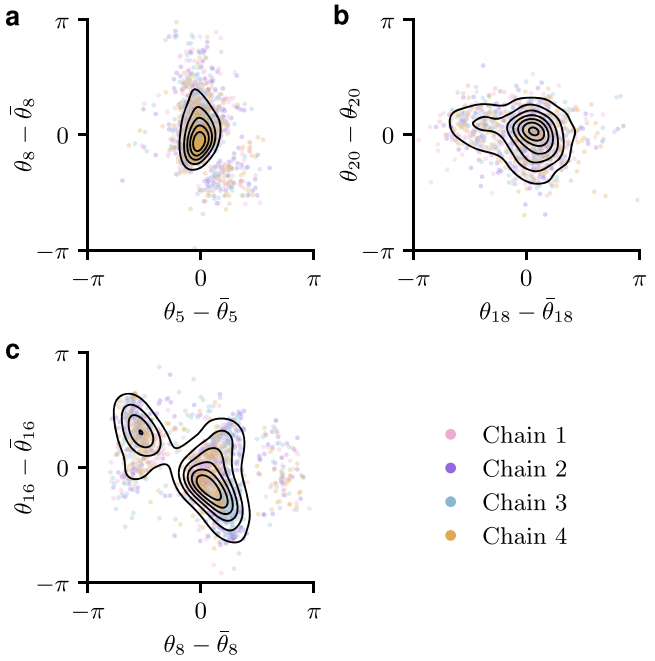


Fig. 6 | Multimodality and non-normality of the posterior distribution. Marginal posterior distributions are shown for pairs of angular coordinates in the embedding of (**a**) the synthetic graph in Fig. 1(**b**) Zachary's karate club[83](**c**) a synthetic graph of 30 vertices with two conflicting embeddings, as described in the Multimodality section (θ_0 has two different ground truth positions). Samples are obtained with the thinned BIGUE algorithm, and chains comprise 500 random embeddings. The color of each circle indicates from which chain it was sampled. The black lines delineate the density of a normal kernel density estimation of the marginal. The sample averages $\bar{\theta}_u$ are circular averages as described in Supplementary Methods 2.

this is due to random fluctuations in the instantiation of G conditioned on the ground truth coordinates.

Next, we turn to the properties of the graph. The edge probabilities of good embeddings should yield graphs with properties similar to those of the original graph. As a result, if the embedding is accurate, we expect that the graphs generated with estimated coordinates will be similar to the original graph. We first verify this with the maximum likelihood found with Mercator; the resulting distribution of graph properties is centered around the observed graph in Fig. 5d, e. We note that although Mercator yields a point-wise estimation, different graphs can be sampled from the same embedding, which explains why there are many density and clustering values for Mercator. With BIGUE, each sample point corresponds to a random graph using the likelihood, and consequently, we can marginalize the graph properties over a much larger parameter space. (Formally, we draw from the posterior predictive distribution of the model.) The resulting distributions are wider and now capture parameter uncertainty; see Fig. 5d, e.

The embeddings also capture less obvious properties. For example, hyperbolic embeddings can play a role in finding effective and efficient greedy routing on the network. In the greedy routing algorithm[10], hyperbolic coordinates act as addresses, and one attempts to reach the target v from a source u by repeatedly following the edge that leads to the neighbor closest to the destination u (in hyperbolic space). The main issue with this algorithm is that paths can sometimes devolve into "greedy loops" that lead nowhere. The greedy success rate evaluates the extent to which this is an issue as the proportion of greedy routes that successfully reach their destination. The posterior sample allows us to compute error bars on the greedy success rate, but also to comment the navigability of the ensemble of plausible embeddings. For instance, Fig. 5f shows that, while many embeddings are as navigable as the embedding inferred by Mercator, the majority of plausible embeddings are less reliable.

The hierarchical organization of graphs is another example of complex properties captured by hyperbolic embeddings. The global hierarchy level was introduced to quantify this, using a function of the angular distance between neighboring vertices with coordinates in the outer and inner parts of the embedding[77]. Figure 5g illustrates how, for the synthetic graph analyzed, most plausible embeddings have smaller global hierarchy levels than both the ground truth and the Mercator embeddings.

Finally, embeddings for the $\mathbb{S}^1$ model can be viewed as classifiers for edges since the likelihood maps potential edges (binary classes) to edge probabilities. If the embedding is a good representation of the graph, edges should have a high probability, while non-edges should have a low probability. We test the performance in the edge prediction task by generating many embeddings and synthetic graphs of 30 vertices and removing 5% of its edges. We then compute the rank of the removed edges among all the unconnected pairs, using their inferred connection probability, Eq. (1). If the removed edges still have a relative high existence probability, the embedding is a good predictor and the ranks should approach 1. Figure 5h shows that Mercator and BIGUE perform almost identically and, unsurprisingly, the original embedding performs better. This suggests that both embeddings are robust to small perturbations.

Multimodality

Next, we inspect the properties of the posterior distribution of coordinates in more detail. MCMC algorithms are computationally expensive, and one might wonder if we could do away with sampling altogether—be it HMC, BIGUE, or another algorithm—by quantifying uncertainty with Laplace's approximation or a variational approach[78] instead. These approximations are attractive because they yield error bars for all quantities nearly for free, and they have strong theoretical backing, e.g., the Bernstein-von Mises theorem that guarantees convergence to the approximated distribution in the big data regime under certain conditions. Unfortunately, for small graphs and $\mathbb{S}^1$, at the very least, we find that the posterior distributions are not multivariate normal distributions and that MCMC is truly necessary.

Treating our sample as truly representative of the posterior distribution, we computed the p-values of the Henze-Zirkler normality test[79] for all pairs of parameters. This test compares the empirical characteristic function to its pointwise limit under the null hypothesis that the data follows a multivariate normal distribution. We found the largest p-value for this test to be 1.11×10^{-25} for the example synthetic graph. This is not surprising when looking at the marginal distribution of the pair of coordinates shown in Fig. 6a: The bulk of the distribution is skewed, and there is a region of very low probability mass next to the bulk. All marginalizations of a multivariate normal yield another multivariate normal; thus, finding one non-normal marginal is enough to reject it as a model for the whole.

It could be that only synthetic graphs show non-normal posterior distributions, but it turns out that the embedding of well-known graphs not explicitly generated by the model behaves the same way. For example, the posterior of Zachary's Karate Club is also not normal: one pair of parameters has a p-value of 0.092 for the Henze-Zirkler normality test, but the complete posterior has a p-value of 0 numerically; Fig. 6b shows that joint marginals for this graph are not ellipsoids.

One of the most compelling pieces of evidence for the non-normality of hyperbolic embeddings, is the ease with which it can be induced using a mixture for the embedding coordinates. A simple procedure that achieves multimodality is as follows. First, we give random coordinates to the vertices as usual. But when we generate the edges of G, we use either the original or an updated angular coordinate for a single vertex u (e.g., $\theta_u \mapsto \theta_u + 2 \bmod 2\pi$), with probability 0.5, thereby connecting u to two incompatible sets of vertices. Figure 6c shows the marginal distribution of two angular positions for the resulting graph. Neither of these two chosen vertices is the vertex with a superposition of angular coordinates, yet the marginal is clearly multimodal. In our experiments, we also found that almost all joint marginals that include at least one angular

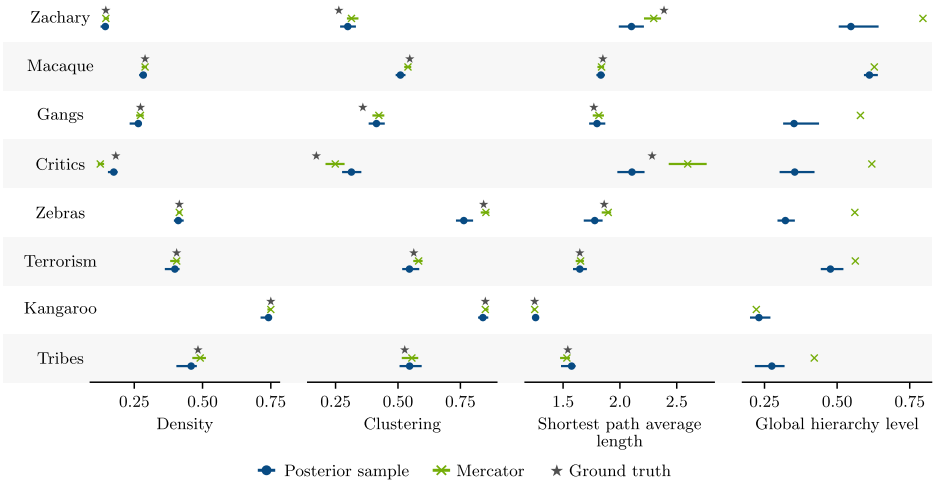


Fig. 7 | Comparison of network embeddings obtained from a maximum likelihood estimator (Mercator, green cross) and the posterior of a Bayesian model (BIGUE, blue circle) for various empirical networks. These networks are Zachary's karate club[83], cortical connectivity of the macaque[84], street gangs in Montreal[85], dutch literary critics[86], zebra social interactions[87], connections in greek terrorist group[88], kangaroo dominance relationships[89] and New Guinea tribes friendships[90]. The value for the original network are displayed in gray stars when available. Vertices outside the largest connected component were removed in order to run Mercator. The horizontal bars show the 50% highest density intervals, and points within are the median. Supplementary Table 1 reports the values for this figure.

coordinate are bimodal; a single vertex with an ambiguous position is sufficient to induce the behavior.

Observational study

To conclude our analysis, we embed a collection of empirical networks with BIGUE and contrast our results with Mercator's; the experiments are summarized in Fig. 7 (see Supplementary Note 4 for a table giving numerical values). Both algorithms reproduce the observed density and clustering (transitivity) of the graph. The shortest paths average length is also reproduced by both algorithms except for the Zachary's karate club and the Dutch literary critics networks, where both algorithms fail. In these cases, it is likely that the $\mathbb{S}^1$ model cannot offer a good representation altogether. And, as we have argued above, Mercator yields a single embedding and thus ignores parameter uncertainty, leading to tighter estimated intervals.

Properties that only depend on coordinates, like the global hierarchy level, have no associated ground truth in observational data and can only be compared across different embedding algorithms. Except for the macaque neural network, we find that Mercator systematically yields higher values of the hierarchy than BIGUE's, usually by a large margin. This suggests that graphs are less hierarchical than they might seem if we only had considered the Mercator embedding.

Conclusions

Hyperbolic random graphs are unquestionably useful and can explain many properties observed in empirical networks, but existing estimation algorithms neglect the significant uncertainty and multimodality that can be present in these models, while off-the-shelf uncertainty quantification methods fall short. We provided evidence that the dynamic HMC is unable to sample the posterior of a Bayesian hyperbolic embedding model, and that supplementing random cluster transformations to a generic random walk is sufficient to sample embeddings of small graphs. While maximum likelihood estimators such as Mercator overfit the coordinates and do not provide error bars on embedding, our algorithm BIGUE provides an accurate posterior sample for the error bars on the graph and embedding properties.

As future work, we believe that generalizing cluster transformation on the $D+1$-sphere could be a promising avenue[52]. While each additional dimension would increase the computational cost by adding another parameter per vertex, this tradeoff may be worthwhile. The higher-dimensional space would allow vertices to navigate around each other more freely, potentially avoiding the barriers illustrated in Fig. 2.

Another challenge we leave for future work is to improve the algorithm's overall computational efficiency. Currently, BIGUE's main limitation is its poor scaling with vertex count—it struggles to properly sample graphs with more than 100 vertices in a reasonable timeframe. While basic optimizations like GPU-accelerated likelihood calculations and a more efficient programming language would help, the most promising speedups may lie in relying on HMC for the adjustment of positions. Naively using external implementations turned out to be too costly, but an integrated solution may work well. Other potential speed improvements include using negative sampling to approximate the likelihood[80] instead of computing the likelihood in $O(|V|^2)$ operations; sampling vertices individually; incorporating graph symmetries during sampling—by swapping affected vertex positions—or even approximate symmetries[81].

Beyond sampling applications, the success of cluster transformations suggests they could enhance existing likelihood maximization algorithms like Mercator.

Data availability

The data were obtained from an online repository[82] or can be fetched from the original papers[83–90]. All the data are also available with the code at https://doi.org/10.5281/zenodo.15272626.

Code availability

A Python implementation of BIGUE is available at https://doi.org/10.5281/zenodo.15189658. The Python code for the numerical analysis of this paper is available at https://doi.org/10.5281/zenodo.15272626.

Received: 28 November 2024; Accepted: 2 May 2025;
Published online: 15 May 2025

References

1. Boguñá, M. et al. Network geometry. *Nat. Rev. Phys.* **3**, 114 (2021).
2. Munzner, T. Exploring large graphs in 3D hyperbolic space. *IEEE Comput. Graph. Appl.* **18**, 18 (1998).
3. Kovács, B. & Palla, G. The inherent community structure of hyperbolic networks. *Sci. Rep.* **11**, 16050 (2021).
4. Foster, D. V., Foster, J. G., Grassberger, P. & Paczuski, M. Clustering drives assortativity and community structure in ensembles of networks. *Phys. Rev. E* **84**, 066117 (2011).
5. Balogh, S. G., Kovács, B. & Palla, G. Maximally modular structure of growing hyperbolic networks. *Commun. Phys.* **6**, 76 (2023).

6. Balogh, S. G. & Palla, G. Intra-community link formation and modularity in ultracold growing hyperbolic networks. *Physica A* **642**, 129784 (2024).
7. Zuev, K., Boguñá, M., Bianconi, G. & Krioukov, D. Emergence of soft communities from geometric preferential attachment. *Sci. Rep.* **5**, 9421 (2015).
8. García-Pérez, G., Serrano, M. Á. & Boguñá, M. Soft communities in similarity space. *J. Stat. Phys.* **173**, 775 (2018).
9. Faqeeh, A., Osat, S. & Radicchi, F. Characterizing the analogy between hyperbolic embedding and community structure of complex networks. *Phys. Rev. Lett.* **121**, 098301 (2018).
10. Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A. & Boguñá, M. Hyperbolic geometry of complex networks. *Phys. Rev. E* **82**, 036106 (2010).
11. Gugelmann, L., Panagiotou, K. & Peter, U. Random hyperbolic graphs: degree sequence and clustering. in *Automata, Languages, and Programming* pp.573–585. (2012)
12. Krioukov, D. Clustering implies geometry in networks. *Phys. Rev. Lett.* **116**, 208302 (2016).
13. Candellero, E. & Fountoulakis, N. Clustering and the hyperbolic geometry of complex networks. in *Algorithms and Models for the Web Graph* pp.1–12. (2014)
14. Boguñá, M., Krioukov, D., Almagro, P. & Serrano, M. Á. Small worlds and clustering in spatial networks. *Phys. Rev. Res.* **2**, 023040 (2020).
15. Serrano, M. Á., Krioukov, D. & Boguñá, M. Self-similarity of complex networks and hidden metric spaces. *Phys. Rev. Lett.* **100**, 078701 (2008).
16. Zheng, M., Allard, A., Hagmann, P., Alemán-Gómez, Y. & Serrano, M. Á. Geometric renormalization unravels self-similarity of the multiscale human connectome. *Proc. Natl. Acad. Sci. USA* **117**, 20244 (2020).
17. García-Pérez, G., Boguñá, M. & Serrano, M. Á. Multiscale unfolding of real networks by geometric renormalization. *Nat. Phys.* **14**, 583 (2018).
18. Song, C., Havlin, S. & Makse, H. A. Self-similarity of complex networks. *Nature* **433**, 392 (2005).
19. Boguñá, M., Krioukov, D. & Claffy, K. C. Navigability of complex networks. *Nat. Phys.* **5**, 74 (2009).
20. Boguñá, M., Papadopoulos, F. & Krioukov, D. Sustaining the Internet with hyperbolic mapping. *Nat. Commun.* **1**, 62 (2010).
21. Papadopoulos, F., Krioukov, D., Boguna, M. & Vahdat, A. Greedy Forwarding in Dynamic Scale-Free Networks Embedded in Hyperbolic Metric Spaces, in *2010 Proceedings IEEE INFOCOM* (2010) pp. 1–9.
22. Baptista, A., Sánchez-García, R. J., Baudot, A. & Bianconi, G. Zoo guide to network embedding. *J. Phys. Complex.* **4**, 042001 (2023).
23. Cai, H., Zheng, V. W. & Chang, K. C.-C. A comprehensive survey of graph embedding: problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.* **30**, 1616 (2018).
24. Zhang, Y.-J., Yang, K.-C. & Radicchi, F. Systematic comparison of graph embedding methods in practical tasks. *Phys. Rev. E* **104**, 044315 (2021).
25. McDonald, D. *The Hierarchical Organisation and Dynamics of Complex Networks*, Ph.D. thesis, University of Birmingham (2021).
26. Goyal, P. & Ferrara, E. Graph embedding techniques, applications, and performance: a survey. *Knowl.-Based Syst.* **151**, 78 (2018).
27. Chowdhary, K. & Kolda, T. G. An improved hyperbolic embedding algorithm. *J. Complex Netw.* **6**, 321 (2018).
28. Keller-Ressel, M. & Nargang, S. Hydra: A method for strain-minimizing hyperbolic embedding of network- and distance-based data. *J. Complex Netw.* **8**, cnaa002 (2020).
29. Sala, F., Sa, C. D., Gu, A. & Re, C. Representation Tradeoffs for Hyperbolic Embeddings, in *Proceedings of the 35th International Conference on Machine Learning* pp.4460–4469. (2018)
30. Verbeek, K. & Suri, S. Metric embedding, hyperbolic space, and social networks. *Comput. Geom.* **59**, 1 (2016).
31. Clough, J. R. & Evans, T. S. Embedding graphs in Lorentzian spacetime. *PLoS ONE* **12**, e0187301 (2017).
32. Sarkar, R. Low Distortion Delaunay Embedding of Trees in Hyperbolic Plane. in *Graph Drawing* pp.355–366. (2012)
33. Chamberlain, B. P., Clough, J. & Deisenroth, M. P. *Neural Embeddings of Graphs in Hyperbolic Space*, Preprint arXiv:1705.10359 (2017).
34. Cannistraci, C. V., Alanis-Lobato, G. & Ravasi, T. Minimum curvilinearity to enhance topological prediction of protein interactions by network embedding. *Bioinformatics* **29**, i199 (2013).
35. Muscoloni, A., Thomas, J. M., Ciucci, S., Bianconi, G. & Cannistraci, C. V. Machine learning meets complex networks via coalescent embedding in the hyperbolic space. *Nat. Commun.* **8**, 1615 (2017).
36. García-Pérez, G., Allard, A., Serrano, M. Á. & Boguñá, M. Mercator: uncovering faithful hyperbolic embeddings of complex networks. *New J. Phys.* **21**, 123033 (2019).
37. Papadopoulos, F., Psomas, C. & Krioukov, D. Network mapping by replaying hyperbolic growth. *IEEE/ACM Trans. Netw.* **23**, 198 (2015).
38. Papadopoulos, F., Aldecoa, R. & Krioukov, D. Network geometry inference using common neighbors. *Phys. Rev. E* **92**, 022807 (2015).
39. Alanis-Lobato, G., Mier, P. & Andrade-Navarro, M. A. Manifold learning and maximum likelihood estimation for hyperbolic network embedding. *Appl. Netw. Sci.* **1**, 10 (2016).
40. Alanis-Lobato, G., Mier, P. & Andrade-Navarro, M. A. Efficient embedding of complex networks to hyperbolic space via their Laplacian. *Sci. Rep.* **6**, 30108 (2016).
41. Wang, Z., Li, Q., Xiong, W., Jin, F. & Wu, Y. Fast community detection based on sector edge aggregation metric model in hyperbolic space. *Physica A* **452**, 178 (2016).
42. Wang, Z., Wu, Y., Li, Q., Jin, F. & Xiong, W. Link prediction based on hyperbolic mapping with community structure for complex networks. *Physica A* **450**, 609 (2016).
43. Bläsius, T., Friedrich, T., Krohmer, A. & Laue, S. Efficient Embedding of Scale-Free Graphs in the Hyperbolic Plane. *IEEE/ACM Trans. Netw.* **26**, 920 (2018).
44. Wang, Z., Sun, L., Cai, M. & Xie, P. Fast hyperbolic mapping based on the hierarchical community structure in complex networks. *J. Stat. Mech.* **2019**, 123401 (2019).
45. Ye, D., Jiang, H., Jiang, Y., Wang, Q. & Hu, Y. Community preserving mapping for network hyperbolic embedding. *Knowl.-Based Syst.* **246**, 108699 (2022).
46. Wu, Z., Di, Z. & Fan, Y. An asymmetric popularity-similarity optimization method for embedding directed networks into hyperbolic space. *Complexity* **2020**, e8372928 (2020).
47. Kovács, B. & Palla, G. Model-independent embedding of directed networks into Euclidean and hyperbolic spaces. *Commun. Phys.* **6**, 1 (2023).
48. Allard, A., Serrano, M. Á. & Boguñá, M. Geometric description of clustering in directed networks. *Nat. Phys.* **20**, 150 (2024).
49. Yi, S., Jiang, H., Jiang, Y., Zhou, P. & Wang, Q. A hyperbolic embedding method for weighted networks. *IEEE Trans. Netw. Sci. Eng.* **8**, 599 (2021).
50. Allard, A., Serrano, M. Á., García-Pérez, G. & Boguñá, M. The geometric nature of weights in real complex networks. *Nat. Commun.* **8**, 14103 (2017).
51. Jankowski, R., Hozhabrierdi, P., Boguñá, M. & Serrano, M. Á. Feature-aware ultra-low dimensional reduction of real networks. *npj Complex.* **1**, 1 (2024).
52. Jankowski, R., Allard, A., Boguñá, M. & Serrano, M. Á. The D-Mercator method for the multidimensional hyperbolic embedding of real networks. *Nat. Commun.* **14**, 7585 (2023).
53. Kitsak, M., Voitalov, I. & Krioukov, D. Link prediction with hyperbolic geometry. *Phys. Rev. Research* **2**, 043113 (2020).
54. Newman, M. E. J. Network structure from rich but noisy data. *Nat. Phys.* **14**, 542 (2018).

55. Young, J.-G., Cantwell, G. T. & Newman, M. E. J. Bayesian inference of network structure from unreliable data. *J. Complex Netw.* **8**, cnaa046 (2021).
56. Peixoto, T. P. Reconstructing networks with unknown and heterogeneous errors. *Phys. Rev. X* **8**, 041011 (2018).
57. Young, J.-G., Petri, G. & Peixoto, T. P. Hypergraph reconstruction from network data. *Commun. Phys.* **4**, 135 (2021).
58. Lizotte, S., Young, J.-G. & Allard, A. Hypergraph reconstruction from uncertain pairwise observations. *Sci. Rep.* **13**, 21364 (2023).
59. Peixoto, T. P. Bayesian Stochastic Blockmodeling, in *Advances in Network Clustering and Blockmodeling*, edited by Doreian, P., Batagelj, V. & Ferligoj A. (John Wiley & Sons, Ltd, 2019) pp. 289–332.
60. van der Pas, S. L. & van der Vaart, A. W. Bayesian community detection. *Bayesian Anal.* **13**, 767 (2018).
61. Papamichalis, M., Turnbull, K., Lunagomez, S. & Airoldi, E. *Latent Space Network Modelling with Hyperbolic and Spherical Geometries*, Preprint arXiv:2109.03343 (2022).
62. Yao, Y., Vehtari, A. & Gelman, A. Stacking for Non-mixing Bayesian Computations: The Curse and Blessing of Multimodal Posteriors. *J. Mach. Learn. Res.* **23**, 79 (2022).
63. Yao, Y., Cademartori, C., Vehtari, A. & Gelman, A. *Adaptive Path Sampling in Metastable Posterior Distributions*, Preprint arXiv:2009.00471 (2020).
64. West, G., Sinkala, Z. & Wallin, J. A kernel mixing strategy for use in adaptive Markov chain Monte Carlo and stochastic optimization contexts. *Front. Appl. Math. Stat.* **8**, 915294 (2022).
65. Tjelmeland, H. & Hegstad, B. K. Mode jumping proposals in MCMC. *Scand. J. Stat.* **28**, 205 (2001).
66. Pompe, E., Holmes, C. & Łatuszyński, K. A framework for adaptive MCMC targeting multimodal distributions. *Ann. Stat.* **48**, 2930 (2020).
67. Park, J. *Sampling from Multimodal Distributions Using Tempered Hamiltonian Transitions*, Preprint arXiv:2111.06871 (2021).
68. Neal, R. M. Sampling from multimodal distributions using tempered transitions. *Stat. Comput.* **6**, 353 (1996).
69. Lan, S., Streets, J. & Shahbaba, B. Wormhole Hamiltonian Monte Carlo. *Proc. Innov. Appl. Artif. Intell. Conf.* **2014**, 1953 (2014).
70. Graham, M. M. & Storkey, A. J. *Continuously Tempered Hamiltonian Monte Carlo*, Preprint arXiv:1704.03338 (2017).
71. Wang, Z., Li, Q., Jin, F., Xiong, W. & Wu, Y. Hyperbolic mapping of complex networks based on community information. *Physica A* **455**, 104 (2016).
72. McKay, B. D. & Piperno, A. Practical graph isomorphism, II. *J. Symb. Comput.* **60**, 94 (2014).
73. Betancourt, M. *A Conceptual Introduction to Hamiltonian Monte Carlo*, Preprint arXiv:1701.02434 (2018).
74. Carpenter, B. et al. *Stan* : a probabilistic programming language. *J. Stat. Softw.* **76**, 1 (2017).
75. Patania, A., Allard, A. & Young, J.-G. Exact and rapid linear clustering of networks with dynamic programming. *Proc. R. Soc. A* **479**, 20230159 (2023).
76. Vehtari, A., Gelman, A., Simpson, D., Carpenter, B. & Bürkner, P.-C. Rank-normalization, folding, and localization: an improved R^ for assessing convergence of MCMC (with Discussion). *Bayesian Anal.* **16**, 667 (2021).
77. García-Pérez, G., Boguñá, M., Allard, A. & Serrano, M. Á. The hidden hyperbolic geometry of international trade: World Trade Atlas 1870–2013. *Sci. Rep.* **6**, 33441 (2016).
78. Blei, D. M., Kucukelbir, A. & McAuliffe, J. D. Variational inference: a review for statisticians. *J. Am. Stat. Assoc.* **112**, 859 (2017).
79. Henze, N. & Zirkler, B. A class of invariant consistent tests for multivariate normality. *Commun. Stat. - Theory Methods* **19**, 3595 (1990).
80. Liu, D., Seshadri, A., Eliassi-Rad, T. & Ugander, J. *Re-Visiting Skip-Gram Negative Sampling: Dimension Regularization for More Efficient Dissimilarity Preservation in Graph Embeddings*, Preprint arXiv:2405.00172 (2024).
81. Pidnebesna, A., Hartman, D., Pokorná, A., Straka, M. & Hlinka, J. Computing approximate global symmetry of complex networks with application to brain lateral symmetry. *Inf. Syst. Front.* https://doi.org/10.1007/s10796-025-10585-3 (2025).
82. Peixoto, T. P. The netzschleuder network catalogue and repository (2020).
83. Zachary, W. W. An information flow model for conflict and fission in small groups. *J. Anthropol. Res.* **33**, 452 (1977).
84. Young, M. P. The organization of neural systems in the primate cerebral cortex. *Proc. R. Soc. B* **252**, 13 (1997).
85. Descormiers, K. & Morselli, C. Alliances, conflicts, and contradictions in Montreal's street gang landscape. *Int. Crim. Justice Rev.* **21**, 297 (2011).
86. de Nooy, W. A literary playground: Literary criticism and balance theory. *Poetics* **26**, 385 (1999).
87. Sundaresan, S. R., Fischhoff, I. R., Dushoff, J. & Rubenstein, D. I. Network metrics reveal differences in social organization between two fission–fusion species, Grevy's zebra and onager. *Oecologia* **151**, 140 (2007).
88. Rhodes, C. J. & Jones, P. Inferring missing links in partially observed social networks. *J. Oper. Res. Soc.* **60**, 1373 (2009).
89. Grant, T. R. Dominance and association among members of a captive and a free-ranging group of grey kangaroos (*Macropus giganteus*). *Anim. Behav.* **21**, 449 (1973).
90. Read, K. E. Cultures of the Central Highlands, New Guinea. *Southwest. J. Anthropol.* **10**, 1 (1954).
91. Kumar, A., Singh, S. S., Singh, K. & Biswas, B. Link prediction techniques, applications, and performance: A survey. *Physica A* **553**, 124289 (2020).

Acknowledgements

This work was supported by Conseil de recherches en sciences naturelles et en génie du Canada (A.A.), Sentinelle Nord (S.L., A.A.) and the Fonds de recherche du Québec (S.L.), and the Vermont Complex Systems Institute (J.G.Y.). We acknowledge Calcul Québec and Alliance de recherche numérique du Canada for their technical support and computing infrastructures.

Author contributions

S.L. designed the algorithm, implemented the method and wrote the paper. A.A. and J.G.Y. supervised the work, and contributed to the research design, result analysis, and paper writing.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42005-025-02122-0.

Correspondence and requests for materials should be addressed to Antoine Allard.

Peer review information *Communications Physics* thanks Saeed Osat, Fragkiskos Papadopoulos and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. A peer review file is available.

Supplementary Information for Symmetry-driven embedding of networks in hyperbolic space

Simon Lizotte[1,2], Jean-Gabriel Young[1,3,4], Antoine Allard[1,2,4]

1. Département de physique, de génie physique et d'optique, Université Laval, Québec (Québec), Canada G1V 0A6
2. Centre interdisciplinaire en modélisation mathématique, Université Laval, Québec (Québec), Canada G1V 0A6
3. Department of Mathematics and Statistics, University of Vermont, Burlington, VT 05405, USA
4. Vermont Complex Systems Institute, University of Vermont, Burlington, VT 05405, USA

Supplementary Methods 1: Metropolis-Hastings sampling

The Metropolis-Hastings algorithm generates Markov chains with arbitrary target invariant measures by (i) sampling transitions between states from a proposal distribution and (ii) accepting or rejecting new states with a carefully crafted acceptance probability. More specifically, the acceptance probability of new state x^* given the previous state x for a continuous random variable is

$$\alpha(x^*, x) = \frac{p(x^*)q(x|x^*)}{p(x)q(x^*|x)}, \tag{S1}$$

where $q(y^*|y)$ is the density of the transition kernel, that is, the density of sampling new state y^* from state y. In this section, the target p is always the posterior distribution of the Bayesian model.

As explained in the text, we define a global transition kernel, which selects sub-kernels randomly, proportional to their weight: 0.4 for a random walk and 0.2 for each cluster transformation (*flip*, *exchange*, *translate*; see the main text and Fig. 4. We will design all sub-kernels with the target p as their stationary measures, meaning that the proposed state of a subkernel can be accepted with probability 1. The weights are canceled in the acceptance probability and their sole purpose is to help the chain mixing.

Random walk

For the Markov chain to be irreducible, the sampling space must be accessible from every neighborhood. This is easily satisfied using a random walk algorithm. In this algorithm, a new state x^* is sampled using a normal distribution centered around the previous state x. When the random walk subkernel is selected by the global kernel, the set of parameters to explore—that is θ, κ or β—is chosen randomly with equal probability. The main challenge of a random walk Metropolis-Hastings is tuning the variance of the transition kernel. This becomes increasingly difficult in high dimensions: A large variance leads to most proposals being rejected, and a small variance leaves the chain in the same region, increasing the autocovariance and the mixing time in both cases. Hence, we move vertices without significantly altering their angular ordering.

For uniformly distributed vertices on the circle, the distribution of angular separation between two geometrically adjacent vertices is approximately (because of the periodicity on the circle) the exponential distribution of parameter $2\pi/|V|$ (i.e. it is a Poisson point process when ignoring periodicity). To preserve the angular order of vertices on average, we sample each θ_u from

$$\Theta_u^* = \theta_u + \varepsilon \mod 2\pi, \quad \varepsilon \sim \mathcal{N}_{[-\pi,\pi)}\left(0, \left(\frac{\pi}{2|V|}\right)^2\right), \tag{S2}$$

where $\mathcal{N}_{[-\pi,\pi)}$ is the truncated normal distribution on the interval $[-\pi, \pi)$ and where $x \mod 2\pi$ adjusts x such that it lies in $[-\pi, \pi)$. Since the normal distribution is symmetric, there is no bias induced from this transformation $q(\theta^*|\theta) = q(\theta|\theta^*)$.

The transition kernel for κ and β is a lower truncated normal kernel

$$x^* \sim \mathcal{N}_{(x_{\min},\infty)}(x, \sigma_x^2), \tag{S3}$$

where $x_{\min}$ is the smallest value for the parameter x. When applying the random walk onto κ, each individual κ_u^* is independently sampled from Eq. (S3). The normalization for Eq. (S3) cannot be omitted when computing the ratio of q in Eq. (S1) because it depends on x. We use $\sigma_\kappa = 0.5$ and $\sigma_\beta = 0.3$ to avoid large random walk steps, since large variations would cause too large fluctuations in the connection probabilities. We use $\kappa_{\min} = \epsilon$, $\beta_{\min} = 1$ for consistency with the priors.

Cluster transformations

For given angular positions θ, we partition the vertices into clusters $\mathcal{C}$ such that every pair of angularly adjacent θ_i and θ_j in the same cluster has an angular separation $\Delta(\theta_i, \theta_j)$ below a threshold t. The two geometric neighbors of a vertex are the closest vertices in the clockwise and counter-clockwise direction of the circle. The partition of the vertex set is converted to a partition of the circle by defining the boundaries of clusters as the midpoints between each cluster's endpoints.

Because the cluster transformations are discrete, we define a subkernel's probability *mass* function $Q[\Theta^* = \theta^*|\theta]$, which gives the probability of sampling a new set of positions θ^* when the coordinates are currently equal to θ. For the cluster transformation, the acceptance probability becomes

$$\alpha(\theta^*, \theta) = \frac{p(\theta^*|G)Q[\Theta^* = \theta|\theta^*]}{p(\theta|G)Q[\Theta^* = \theta^*|\theta]}. \tag{S4}$$

Each cluster transformation operates on a subset Λ of the partition, $\Lambda \subseteq \mathcal{C}$, containing a predetermined number of clusters. For instance, when we apply a translation and move cluster $C_1 \in \mathcal{C}$ to the position of cluster $C_2 \in \mathcal{C}$, then $\Lambda = \{C_1, C_2\}$. Since Λ is not ordered, we choose to translate C_1 to C_2 or C_2 to C_1 with equal probability: $\mathbb{P}[\Theta^* = \theta^*|\Lambda, \theta] = 1/2$. (This situation does not arise for the *flip* transformation because it involves a single cluster, nor for the *exchange* transformation because it is symmetric.)

The probability Q of sampling the new state is found by marginalizing over all admissible partitions $\mathcal{C}$ (i.e., supersets of Λ)

$$Q[\Theta^* = \theta^*|\theta] = \sum_{\mathcal{C} \supseteq \Lambda} \mathbb{P}[\Theta^* = \theta^*|\Lambda, \theta]\mathbb{P}[\mathbf{\Lambda} = \Lambda|\theta], \tag{S5}$$

where $\mathbb{P}[\mathbf{\Lambda} = \Lambda|\theta]$ is the probability mass function of proposed clusters Λ (random variables are typeset in bold to distinguish them from their values). This equation holds if the cluster transformation only yields θ^* when operating on that specific Λ. This is the case for our cluster transformations except for very specific θ that have measure zero.

To define and calculate $\mathbb{P}[\mathbf{\Lambda} = \Lambda|\theta]$, we first note that a partition $\mathcal{C}$ usually contains more clusters than what is strictly required by the transformation, and so we choose Λ randomly from $\mathcal{C}$. When there isn't a sufficient number of clusters in the sampled partition $\mathcal{C}$, the proposed θ^* is automatically rejected. For the sake of simplicity and efficiency, we use a uniform distribution corresponding to

$$\mathbb{P}[\mathbf{\Lambda} = \Lambda|\mathcal{C}, \theta] = \binom{|C|}{|\Lambda|}^{-1}. \tag{S6}$$

(We note that targeted cluster selection could be an interesting future research direction.)

Since the hyperbolic space has a hierarchical structure, natural clusters can emerge at different scales, and we found it helpful to use a random threshold T to sample the partitions. This adds a bit of mathematical complexity, as we will now need to think of the cluster proposal probability $\mathbb{P}[\mathbf{\Lambda} = \Lambda|\theta]$ as the marginal of $\mathbb{P}[\mathbf{\Lambda} = \Lambda, \mathcal{C}|\theta]$ over all partitions. To compute this probability, we note that there exists an interval of threshold values $T \in [t, t')$ that yields the same partition $\mathcal{C}$. Since our partitioning algorithm is otherwise deterministic, we can convert the probability of sampling a particular partition $\mathcal{C}$ to the probability of sampling the threshold in that interval,

$$\mathbb{P}[\boldsymbol{\mathcal{C}} = \mathcal{C}|\theta] = \mathbb{P}[T \in [t, t') \mid \theta], \tag{S7}$$

where $\boldsymbol{\mathcal{C}}$ is the random variable of the partition. A uniform distribution over $[0, \pi]$ might seem reasonable for T, but it turns out to be a poor choice because the density of vertices per radian depends on the number of vertices. In other words, for the same threshold value, increasing the number of vertices would eventually lead us to propose a single cluster with probability close to 1 (unless there are gaps that never contain any vertices).

To take this into consideration, we parameterize the distribution of T using the probability ξ that at least one vertex u is within the separation threshold t of another vertex v

$$\xi := \mathbb{P}[\exists u : |\theta_u| < t/2] = 1 - \left(1 - \frac{t}{\pi}\right)^{|V|-1}, \tag{S8}$$

where we assume that the angles are uniformly distributed and set $\theta_v = 0$ without loss of generality. By rearranging this equation, we get

$$t(|V|, \xi) = \pi(1 - (1 - \xi)^{1/(|V|-1)}). \tag{S9}$$

In the algorithm, we draw T, by analogy to the random walk subkernel for the embedding coordinates:

$$T \sim \mathcal{N}_{[0,\pi)}\left(t(|V|, 0.9), \left(\frac{\pi}{2|V|}\right)^2\right). \tag{S10}$$

In order to compute Eq. (S5), we need an efficient way to find every partition $\mathcal{C} \supseteq \Lambda$ and another to find the intervals $[t, t')$ of Eq. (S7). Luckily, our partitioning algorithm makes this feasible: A cluster can only shrink as T decreases and can only grow as T increases. This means that given Λ and θ, all valid partitions $\mathcal{C} \subseteq \Lambda$ are obtained within an interval $T \in [t_{\min}, t_{\max})$. The values of $t_{\min}(\Lambda)$ and $t_{\max}(\Lambda)$ are the largest in-cluster and the smallest inter-cluster angular separations found for any cluster of Λ respectively

$$t_{\min}(\Lambda) = \max_{C_j \in \Lambda} \max_{\substack{u \in C_j \\ v \in \mathcal{A}(u) \cap C_j}} \Delta(\theta_u, \theta_v), \tag{S11}$$

$$t_{\max}(\Lambda) = \min_{C_j \in \Lambda} \min_{\substack{u \in C_j \\ v \in \mathcal{A}(u) \setminus C_j}} \Delta(\theta_u, \theta_v), \tag{S12}$$

where $\mathcal{A}(u)$ is the set of the two angularly adjacent vertices of u. The inner maximum of Eq. (S11) is set to 0 when C_j contains only u and the inner minimum of Eq. (S12) is set to π when C_j contains all the vertices (they are ill-defined otherwise).

We now determine the thresholds within $(t_{\min}, t_{\max})$ for which the partition changes, which, here, is when the number of clusters changes. These intermediary thresholds are in fact the angular separations of geometrically adjacent vertices outside the required clusters (noted $\{u \notin \Lambda\}$)

$$\{\Delta(\theta_u, \theta_v) | u \notin \Lambda, v \in \mathcal{A}(u)\} \cap (t_{\min}, t_{\max}). \tag{S13}$$

These are used to partition the admissible thresholds

$$[t_{\min}, t_{\max}) = [t_{\min}, t_1) \cup [t_1, t_2) \cup \cdots \cup [t_k, t_{\max}), \tag{S14}$$

where k is the number of intermediary thresholds and t_i is the ith intermediary threshold in ascending order. Hence, denoting $I(\Lambda)$ the set of these subintervals,

$$\mathbb{P}[\mathbf{\Lambda} = \Lambda | \theta] = \sum_{[t,t') \in I(\Lambda)} \mathbb{P}[\mathbf{\Lambda} = \Lambda | \mathcal{C}, \theta] \, \mathbb{P}[T \in [t, t') \mid \theta], \tag{S15}$$

where, with a slight abuse of notation, we have used the fact $\mathcal{C}$ is the partition obtained for thresholds $T \in [t, t')$.

We note that Eq. (S4) is derived from a detailed balance, and thus requires that $Q[\Theta^* = \theta | \theta^*] > 0$ if and only if $Q[\Theta^* = \theta | \theta] > 0$, meaning that each transformation should be reversed by a single Markov transition. This is always the case for our cluster-based transformations because each one can be undone when they operate on the same Λ. Our partitioning technique also ensures that Λ can be a subset of the partition of the transformed angular coordinates θ^*: each cluster transformation cannot place a vertex closer than $t/2$ to the cluster boundary, which means that there exists an interval of threshold $[t, t + \delta)$ that yields a partition of θ^* containing obtaining Λ for θ^* ($\delta = 0$ if two vertices exactly have a distance of $t/2$ to a cluster boundary, an event of measure zero).

Finally, it's important to note that the angular separation between vertices and cluster boundaries changes. This means that the intermediary thresholds are not the same for θ and θ^* and that the interval $[t_{\min}, t_{\max})$ and the intermediary thresholds for θ^* must also be computed in $Q[\Theta^* = \theta | \theta]$.

In summary, each cluster-based subkernel transition is done by first sampling T according to Eq. (S10), which gives a partition $\mathcal{C}$ for the current angular coordinates θ. We then sample $\Lambda \subseteq \mathcal{C}$ with $|\Lambda|$ being the required size for the transformation. We apply the cluster transformation with the outcome probabilities $\mathbb{P}[\Theta^* = \theta^* | \Lambda, \theta]$, which yields θ^*. We accept θ^* as the new state with probability $\alpha(\theta^*, \theta)$ of Eq. (S4). The marginalized probability $Q[\Theta^* = \theta | \theta^*]$ that the sampling drew θ^* is given by Eq. (S5) (and $Q[\Theta^* = \theta | \theta^*]$ is simply obtained by exchanging $\theta \leftrightarrow \theta^*$ in the equations and by using the Λ that was sampled to produce θ^* in the first place).

Supplementary Methods 2: Convergence diagnostics

Assessing the quality of an MCMC sampler can be challenging. At the very least, one should verify that (1) the autocovariance of the states decreases with the lag and (2) the stationary distribution is identical no matter the initial state. The former indicates how small the error of Monte Carlo estimators is (by virtue of the Markov chain central limit theorem), and the latter suggests that a Markov chain running long enough will lead to the correct stationary distribution. In this section, we present the effective sample size and the potential scale reduction factor, proxies of these desired properties. We also extend these statistics to random variables on the circle.

The Gelman-Rubin [1] potential scale reduction factor $\hat{R}$ indicates whether or not the different chains "agree" on the typical values of the parameters. Let $(x_1^{(m)}, x_2^{(m)}, \ldots, x_N^{(m)})$ be the m^{th} Markov chain's state, and for the sake of simplicity, let us assume that we have M chains of equal length N. The total chain variance is estimated with the weighted average

$$\widehat{\text{var}}^+ := \frac{N-1}{N} W + \frac{1}{N} B, \tag{S16}$$

where W and B/N are, respectively, the within-chain average variance and the between-chain variance

$$W = \frac{1}{M} \sum_{m=1}^{M} s_m^2, \tag{S17}$$

$$B = \frac{N}{M-1} \sum_{m=1}^{M} (\bar{x}^{(m)} - \bar{x})^2, \tag{S18}$$

and where s_m^2 and $\bar{x}^{(m)}$ are the sample variance and sample average

$$s_m^2 = \frac{1}{N-1} \sum_{j=1}^{N} (x_j^{(m)} - \bar{x}^{(m)})^2, \tag{S19}$$

$$\bar{x}^{(m)} = \frac{1}{N} \sum_{j=1}^{N} x_j^{(m)}, \tag{S20}$$

$$\bar{x} = \frac{1}{M} \sum_{m=1}^{M} \bar{x}^{(m)}. \tag{S21}$$

The potential scale reduction factor is then

$$\hat{R} := \sqrt{\frac{\widehat{\text{var}}^+}{W}}. \tag{S22}$$

When an MCMC algorithm mixes properly, the chains' $\hat{R} \to 1$ as $N \to \infty$. We use the "split-$\hat{R}$" variant [2], where each chain is split in two for the computation of $\hat{R}$. This adjustment helps detect a within-chain lack of convergence. Note that having a small $\hat{R}$ is a necessary but insufficient convergence condition.

The effective sample size quantifies how many sample points of the sample are considered independent. The (unnormalized) sample autocovariance is

$$a_m(\tau) = \sum_{j=1}^{N-\tau} (x_j^{(m)} - \bar{x}^{(m)})(x_{j+\tau}^{(m)} - \bar{x}^{(m)}). \tag{S23}$$

The effective sample size of chain m is [3]

$$n_{\text{eff}}^{(m)} \approx \frac{N}{1 + 2\sum_{\tau=1}^{N} \rho_m(\tau)} \tag{S24}$$

where $\rho_m(\tau) = a_m(\tau)/a_m(0)$ is the normalized sample autocovariance. In practice, Eq. (S23) is noisy for large τ because there are not enough sample points. We use a maximum lag of $\tau = \lfloor N/50 \rfloor$.

Autocovariance can be combined across chains using [4]

$$\rho(\tau) = 1 - \frac{W - \frac{1}{M}\sum_{m=1}^{M} s_m^2 \rho_m(\tau)}{\widehat{\text{var}}^+}, \tag{S25}$$

which yields the global effective sample size

$$S_{\text{eff}} \approx \frac{NM}{1 + 2\sum_{\tau=1}^{N} \rho(\tau)}. \tag{S26}$$

We use the same heuristic as $n_{\text{eff}}^{(m)}$ for the maximal lag value.

S_{eff} and $\hat{R}$ statistics are typically used for continuous random variables defined on the real line. Hence, we use these for the parameters κ_u and β. However, they are not appropriate for the angular coordinates because of the cyclic boundary condition. The issue stems from additions and subtractions that appear in averages, Eq. (S26) and Eq. (S22).

To extend S_{eff} and $\hat{R}$ to a Markov chain realization $(\theta_0^{(m)}, \theta_1^{(m)}, \ldots, \theta_N^{(m)})$ on the circle, we use the circular analogous of the sample average and the sample correlation coefficient, which are respectively [5]

$$\bar{\phi} := \arg\left(\sum_{j=1}^{S} \exp\{i\phi_j\}\right), \tag{S27}$$

$$r_{\Phi,\Psi} := \frac{\sum_{j=1}^{S} \sin\left(\phi_j - \bar{\phi}\right) \sin\left(\psi_j - \bar{\psi}\right)}{\sqrt{\sum_{j=1}^{N} \sin^2\left(\phi_j - \bar{\phi}\right)} \sqrt{\sum_{j=1}^{N} \sin^2\left(\psi_j - \bar{\psi}\right)}}, \tag{S28}$$

where i is the imaginary number, and $(\phi_j)_{j=1}^{S}$ and $(\psi_j)_{j=1}^{S}$ are i.i.d. samples of Φ and Ψ respectively with Φ and Ψ being random variables on the circle. Note that, to simplify notation, θ_j denotes the jth state of the Markov chain realization instead of the angular coordinate of vertex j.

The circular autocovariance at lag τ is obtained directly from Eq. (S28) with the sample points $\phi_j = \theta_j$ and their lagged values $\psi_j = \theta_{j+\tau}$. Since the numerator is analogous to the unnormalized sample covariance, we define the unnormalized sample autovariance

$$a_\circ(\tau) = \sum_{j=1}^{N-\tau} \sin\left(\theta_j - \bar{\theta}\right) \sin\left(\theta_{j+\tau} - \bar{\theta}\right) \tag{S29}$$

and the normalized autocovariance function

$$\rho_\circ(\tau) := \frac{a_\circ(\tau)}{a_\circ(0)} \tag{S30}$$

that leads to the circular effective sample size

$$n_{\text{eff}}^{\circ} \approx \frac{1}{1 + 2\sum_{\tau=1}^{N} \rho_\circ(\tau)}. \tag{S31}$$

To obtain the circular equivalents of S_{eff} and $\hat{R}$, we use the circular sample average for $\bar{x}$ and $\bar{x}^{(m)}$ and substitute the subtractions $(\phi - \psi)$ with $\Delta(\phi, \psi)$ in Eqs. (S18) and (S19).

Supplementary Note 1: Computational complexity

The main drawback of using MCMC is that it typically scales poorly with the dimension of the sampling space. In the case of the $\mathbb{S}^1$ model, this sampling space consists of $2|V|+1$ parameters: $|V|$ angular coordinates, $|V|$ expected degrees and β.

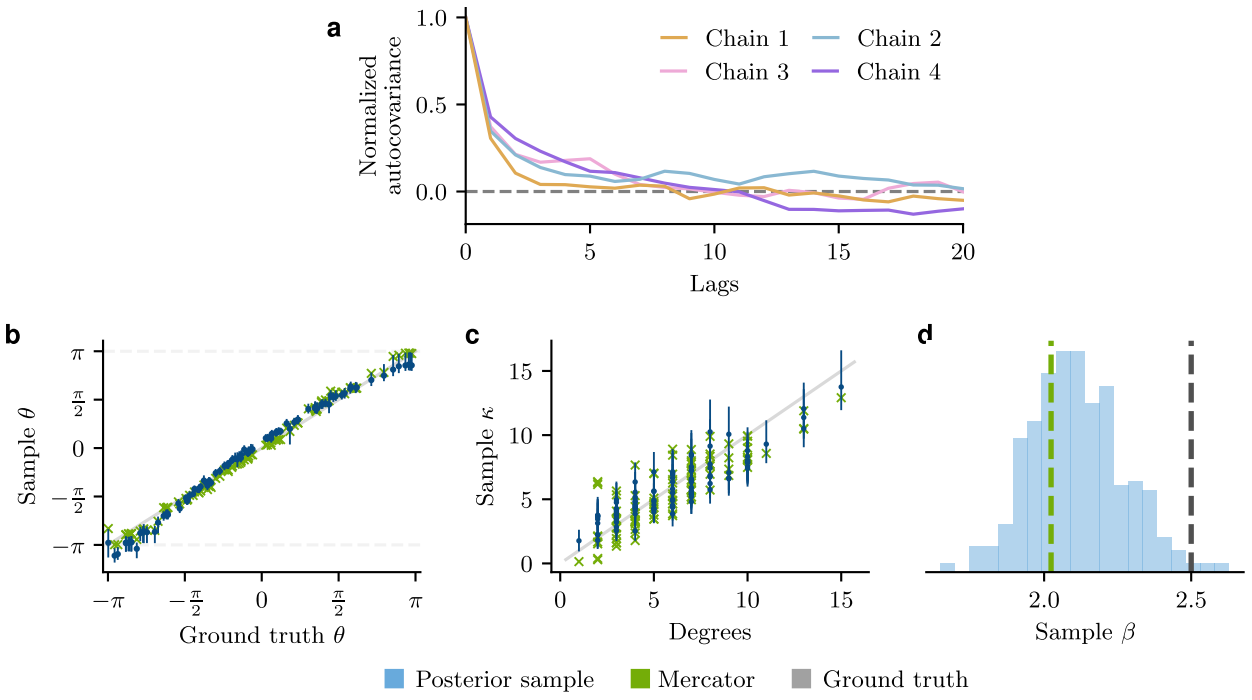
The complexity of our particular approach, BIGUE, is as follows. For the random walk described in section Random walk of Supplementary Methods 1, sampling Θ^* and computing the acceptance probability require $\mathcal{O}(|V|)$ and $\mathcal{O}(|V|^2)$ time, respectively, since the acceptance ratio contains one term for every pair of vertices. This is an unavoidable bottleneck as disconnected vertices contribute greatly to the likelihood in the $\mathbb{S}^1$ model.

For the cluster transformations discussed in section Cluster transformations of Supplementary Methods 1, the calculation of the acceptance probability also consists in a bottleneck. A number of steps have linear complexity in the number of vertices. These include partitioning vertices into clusters $\mathcal{C}$; performing the cluster transformation; and finding the thresholds $t_{\min}$, $t_{\max}$ and $(t_i)_{i=1}^k$. Other steps are quicker. Sampling t and clusters both take $\mathcal{O}(1)$ steps (at most two clusters sampled), and computing the biases induced by $(t_{\min}, t_{\max}, (t_i)_{i=1}^k)$ takes $\mathcal{O}(|\mathcal{C}|)$ time. Computing the posterior density, however, again requires $\mathcal{O}(|V|^2)$ steps.

In summary, both kinds of sampling iterations scale in $\mathcal{O}(|V|^2)$ in time because of the likelihood. While caching partial sums of the log-likelihood could lessen the complexity of cluster moves, it would not provide a significant improvement unless the clusters involve a small number of vertices.

In addition to the quadratic scaling of the iterations, the sampling space volume grows exponentially with the number of variables. This is highlighted in Supplementary Figure 1a where using the same thinning as in Fig. 3d of the main text, the Markov chain states still have a significant autocovariance. Supplementary Figure 1a suggests that having little to no autocovariance between states would require skipping 10 iterations, which corresponds to keeping 1 sample point for every $100,000$ iterations. This is a clear indication of the non-linearity of mixing: sampling a graph with 3.33 times the number of vertices requires 10 times as many iterations.

Nonetheless, the chains settle to embeddings compatible with the ground truth and Mercator, as shown in Supplementary Figure 1b-d. This means we can expect BIGUE to work on larger graphs, but it requires a lot of computing time. With the current implementation of the algorithm, good quality samples for graphs much larger than 100 vertices cannot be obtained in a reasonable time.



Supplementary Figure 1: Inference for a synthetic graph of 100 vertices. **(a)** Normalized autocovariance of each chain averaged over all parameters at different lags. Posterior estimation of the **(b)** angular coordinates θ **(c)** parameters κ **(d)** inverse temperature β. In panels (b-d), values obtained from BIGUE, Mercator and the ground truth are displayed in shades of blue, green and gray, respectively. The graph is generated with the same parameters used for the synthetic graph of Fig. 1 of the main text. 400 total embeddings are sampled with four chains with a thinning of $10,000$ iterations. Each chain is initialized without access to ground truth as in Fig.3b and runs for 50 iterations before samples are recorded (warm up or burn-in). The highest potential scale reduction factor for this simulation is $\hat{R}_{\max} = 1.43$, and the effective sample sizes S_{eff} median is 231.80.

Supplementary Note 2: Relationship between $\mathbb{S}^1$ and $\mathbb{H}^2$ models

While we developed our algorithm for the $\mathbb{S}^1$ formulation of hyperbolic random graphs, some measures, like the greedy success rate, and visualization such as Fig. 1 of the main text use the hyperbolic coordinates in $\mathbb{H}^2$ embedding. They are formally related by coordinate transformation

$$r_u = R_{\mathbb{H}} - 2\ln\frac{\kappa_u}{\kappa_{\min}} \tag{S32}$$

with $R_{\mathbb{H}} = 2\ln\frac{|V|}{\mu\pi\kappa_{\min}}$ where $\kappa_{\min} > 0$ is the smallest allowable κ value (we set $\kappa_{\min} = 1$), edge probabilities in the $\mathbb{S}^1$ model can be rewritten as

$$\mathbb{P}[a_{uv} = 1 \mid x, \beta] = \frac{1}{1 + \exp\Big(\beta(r_u + r_v + 2\ln\frac{\Delta(\theta_u,\theta_v)}{2} - R_{\mathbb{H}})\Big)}. \tag{S33}$$

We retrieve the hyperbolic random graph model connection probability [6]

$$\mathbb{P}[a_{uv} = 1 \mid x, \beta] = \frac{1}{1 + \exp(\beta(d_{\mathbb{H}}(x_u, x_v) - R_{\mathbb{H}}))} \tag{S34}$$

using the following approximation of the hyperbolic distance

$$\begin{aligned} d_{\mathbb{H}}(x_u, x_v) &= \operatorname{arccosh}\Big(\cosh(r_u)\cosh(r_v) - \sinh(r_u)\sinh(r_v)\cos(\theta_u - \theta_v)\Big), \\ &\approx r_u + r_v + 2\ln\frac{\Delta(\theta_u,\theta_v)}{2}, \end{aligned} \tag{S35}$$

where $x_u = (r_u, \theta_u)$ is the position of vertex u, and r_u and θ_u are respectively the radial and angular coordinates in the hyperboloid model.

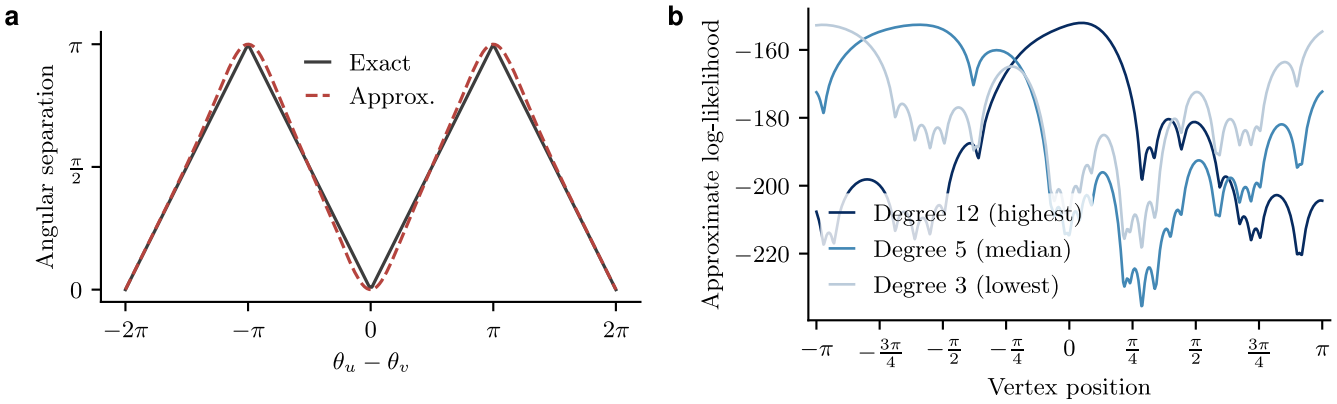
Supplementary Note 3: Differentiable $\mathbb{S}^1$ model

As suggested in Fig. 2 of the main text for unconnected vertices, the gradient of the likelihood is undefined when two vertices, u and v, have the same angular coordinate $\theta_u = \theta_v$. The culprit is the derivative of $\Delta(\theta_u, \theta_v)$ with respect to θ_u, since the function contains two absolute values. For a given value of θ_v, we have that $-\pi + \theta_v \leq \theta_u < \pi + \theta_v$, which contains at most three discontinuities: $\theta_u = \theta_v$ and $\theta_u - \theta_v \in \{-2\pi, -\pi, \pi, 2\pi\}$. One way to smoothen these sharp transitions is to replace each absolute value with

$$|x| \approx x\left(\frac{2}{1 + e^{-bx}} - 1\right) := \tilde{a}(x). \tag{S36}$$

This approximation is exact in the limit of $b \to \infty$, which allows us to control the sharpness of the gradient. Beyond the obvious distortions around the discontinuities, another artifact of the approximation is that the rotation symmetry is lost: the gradient doesn't decrease to 0 at $\pm 2\pi$, while it does at $\theta_u = \theta_v$ and $\theta_u - \theta_v = \pm\pi$, where the approximation is used. Further, while $\tilde{a}(0) = 0$, the approximate separation is not zero for $\theta_u = \theta_v$ because $\pi - \tilde{a}(\pi) > 0$.

We found in practice that using $b > 3$ caused gradient divergences in Stan. Supplementary Figure 2 compares the exact angular separation to the approximate one with $b = 3$. While the error seems small, densely connected groups of vertices are usually tightly grouped angularly, which can result in incorrect embeddings.



Supplementary Figure 2: **(a)** Differentiable approximation of the angular separation with $b = 3$. **(b)** Figure 2 of the main text using the Differentiable $\mathbb{S}^1$ model with $b = 3$. The approximate angular separation is not invariant to rotations because the gradient doesn't decrease to 0 at $\theta_u = \theta_v$ and at $\theta_u - \theta_v = \pm\pi$. The gradient discontinuities are gone at the cost of distortions of the likelihood.

Supplementary Note 4: Numerical values of Figure 7

Supplementary Table 1 contains the number of vertices, the effective sample size, the density, the clustering, the shortest path length average and the global hierarchy level with their uncertainty for each dataset of Fig. 7 of the main text.

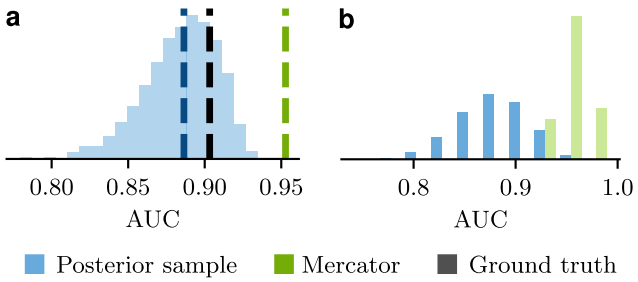
Dataset	$\|V\|$	S_{eff}	Density			Clustering			Shortest path length average			Global hierarchy level	
			Orig.	BIGUE	Mercator	Orig.	BIGUE	Mercator	Orig.	BIGUE	Mercator	BIGUE	Mercator
Macaque	47	431 [361, 611]	0.29,	0.28 [0.27, 0.29],	0.29 [0.28, 0.29]	0.55,	0.51 [0.50, 0.53],	0.54 [0.53, 0.55]	1.85,	1.83 [1.80, 1.86],	1.84 [1.81, 1.85]	0.61 [0.60, 0.64],	0.63 [—]
Zachary	33	933 [705, 1159]	0.15,	0.14 [0.13, 0.15],	0.15 [0.14, 0.15]	0.26,	0.30 [0.27, 0.33],	0.31 [0.30, 0.34]	2.39,	2.10 [2.00, 2.20],	2.30 [2.22, 2.35]	0.55 [0.51, 0.64],	0.80 [—]
Critics	29	959 [683, 1154]	0.18,	0.17 [0.16, 0.18],	0.13 [0.12, 0.14]	0.17,	0.31 [0.28, 0.35],	0.25 [0.21, 0.28]	2.28,	2.11 [1.99, 2.21],	2.60 [2.44, 2.75]	0.35 [0.31, 0.42],	0.62 [—]
Gangs	23	1059 [553, 1152]	0.27,	0.26 [0.24, 0.27],	0.27 [0.26, 0.28]	0.36,	0.41 [0.39, 0.44],	0.42 [0.40, 0.44]	1.77,	1.80 [1.74, 1.86],	1.81 [1.77, 1.85]	0.35 [0.32, 0.43],	0.58 [—]
Zebras	23	645 [385, 806]	0.42,	0.41 [0.40, 0.43],	0.42 [0.41, 0.42]	0.84,	0.76 [0.74, 0.80],	0.85 [0.84, 0.86]	1.86,	1.78 [1.70, 1.84],	1.90 [1.85, 1.91]	0.32 [0.30, 0.35],	0.56 [—]
Terrorism	18	1048 [759, 1135]	0.41,	0.40 [0.37, 0.41],	0.41 [0.39, 0.41]	0.56,	0.55 [0.52, 0.58],	0.58 [0.57, 0.59]	1.65,	1.65 [1.60, 1.70],	1.65 [1.62, 1.67]	0.48 [0.45, 0.52],	0.56 [—]
Kangaroo	16	509 [411, 600]	0.75,	0.74 [0.72, 0.75],	0.75 [0.74, 0.75]	0.85,	0.84 [0.83, 0.86],	0.85 [0.85, 0.86]	1.25,	1.26 [1.24, 1.27],	1.25 [1.24, 1.25]	0.23 [0.20, 0.27],	0.22 [—]
Tribes	16	962 [521, 1125]	0.48,	0.46 [0.41, 0.47],	0.49 [0.47, 0.51]	0.53,	0.55 [0.51, 0.59],	0.56 [0.52, 0.58]	1.54,	1.57 [1.49, 1.60],	1.53 [1.48, 1.55]	0.28 [0.22, 0.32],	0.42 [—]

Supplementary Table 1: Numerical values of the properties shown in Fig. 7 of the main text. For each property, the reported values are in order: original, BIGUE, and Mercator. For the effective sample size S_{eff}, the interquartile range is given instead of the highest density interval. Each sample is a combination of 4 chains of length 300. The effective sample size can be greater than the sample size when the autocovariance is negative on odd lags.

Supplementary Note 5: AUC ROC of link prediction

A common metric used in binary classification is the receiver operating characteristic curve (ROC). It quantifies the sensitivity of the estimator as a function of the false positive rate. The area under the curve of the AUC (AUC ROC, here shortened as AUC), is a scalar that summarizes this curve: a perfect estimator has an AUC of 1 and a random classification estimator has an AUC of 0.5. The AUC for link prediction is computed using each pair of vertices in the graph.

Supplementary Figure 3a illustrates that Mercator is a better predictor of the original graph than the Bayesian model—this is simply because Mercator finds a embedding with a higher likelihood. After removing 5% of the edges, Mercator's AUC drops but remains higher than that of the Bayesian model. This is not surprising because most edges still exist and removed edges are a small portion of the set of pairs of vertices.



Supplementary Figure 3: Area under the receiver operating characteristic curve (AUC) for **(a)** the synthetic graph of Fig. 1 **(b)** the synthetic graphs where 5% of the edges were removed. The blue dotted line in panel (a) is the median of the posterior sample.

Supplementary References

[1] A. Gelman and D. B. Rubin, Statist. Sci. **7**, 457 (1992).

[2] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian Data Analysis*, 3rd ed. (CRC Press, 2013).

[3] A. Sokal, in *Functional Integration*, edited by C. DeWitt-Morette, P. Cartier, and A. Folacci (Springer US, 1997) pp. 131–192.

[4] A. Vehtari, A. Gelman, D. Simpson, B. Carpenter, and P.-C. Bürkner, Bayesian Anal. **16**, 667 (2021).

[5] S. R. Jammalamadaka and A. Sengupta, *Topics In Circular Statistics* (World Scientific Publishing, 2001).

[6] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá, Phys. Rev. E **82**, 036106 (2010).