Shortest Path Inference in Incomplete Network

Jiaze Li

June 2025

1 Introduction

The shortest path problem for an observed network can be efficiently solved using Dijkstra's Algorithm. However, when the observed network is incomplete, meaning some edges are missing compared to the original network, the shortest path between any two given nodes can be drastically different or may not exist at all.

In this report, I investigate shortest path inference methods for synthetic incomplete networks, including plain Dijkstra's algorithm, the distance to geodesic method introduced in [1] and Dijkstra's algorithm applied to a fully connected graph where missing edges are assigned weights as inverse edge probabilities. The original networks are generated using the random hyperbolic graph model, and incomplete networks are created by randomly deleting p proportion of edges uniformly.

The Random Hyperbolic Graph model (**RHG**) is a generative graph model that generate network exhibiting some real-world network characteristics, such as scale-free degree distributions, high clustering coefficient, etc. In RHG, node connectivity is determined by geodesic distances based on nodes' embeddings in hyperbolic space, the shorter the distance, the higher the connection probability. Here I follow the definition of RHG in the section III of supplementary material for [1]. The RHG need 4 parameters:

- n: the number of nodes
- $\alpha \in (\frac{1}{2}, 1)$: parameter that determines the exponent of the degree distribution

$$P(k) \sim k^{-\gamma} \ , \tag{1}$$

  where $\gamma = 2\alpha + 1$.

- $T \in [0, 1)$: parameter that controls the clustering coefficient; lower values of T yield higher clustering coefficients
- R: the radius of the hyperbolic disk. The parameter R is related to the average degree $\hat{k}$ by:

$$\hat{k} = \frac{8n\alpha^2}{(2\alpha - 1)^2} \frac{T}{sin(\pi T)} e^{-\frac{R}{2}} \ . \tag{2}$$

To generate a network using RHG, we follow these steps:

1. Sample n nodes' coordinates $\{(r_i, \theta_i)\}$, $i \in \{1, 2, ..., n\}$ according to

$$\begin{aligned} r_i \sim P(r_i) &= \frac{sinh(\alpha r_i)}{cosh(\alpha R) - 1}, r_i \in [0, R] \\ \theta_i \sim P(\theta_i) &= \frac{1}{2\pi}, \theta_i \in [0, 2\pi] \end{aligned} \tag{3}$$

2. Compute the geodesic distance for any two nodes i and j:

$$\begin{aligned} x_{ij} &= arccosh(cosh(r_i)cosh(r_j) - sinh(r_i)sinh(r_j)cos(\Delta\theta_{ij})) \\ \Delta\theta_{ij} &= \pi - |\pi - |\theta_i - \theta_j|| \end{aligned} \tag{4}$$

3. Generate an edge for any nodes pair (i, j) with probability:

$$P(e_{ij} = 1) = \frac{1}{1 + e^{\frac{x_{ij} - R}{2T}}} \tag{5}$$

To evaluate the inferred shortest path in incomplete network, we can use **overlap** or **path edit distance**. Assume the true shortest path in the original network is path $= \{s, i_1, i_2, ..., i_{n-2}, t\}$, the inferred shortest path in incomplete network is path$^* = \{s, j_1, j_2, ..., j_{m-2}, t\}$. The overlap is the Jaccard index of nodes set $\ell = \{i_1, i_2, ..., i_{n-2}\}$ and $\ell^* = \{j_1, j_2, ..., j_{m-2}\}$:

$$\text{overlap}(\text{path}, \text{path}^*) = \frac{|\ell \cap \ell^*|}{|\ell \cup \ell^*|} \ . \tag{6}$$

However, the overlap metric is not sensitive to node ordering. For example, the path $= \{0, 1, 2, 3, 4\}$ and path$^* = \{0, 2, 3, 1, 4\}$ have overlap $= 1$ although they have different order of nodes. To address this limitation, we can use path edit distance. The path edit distance measures the minimum number of single node edits (insertion, deletion and substitution) needed to transform from one path into another. We can use dynamic programming to compute the edit distance efficiently. After computing the edit distance between path and path*, we normalize it by the maximum length of the two path:

$$\text{normEditDistance}(\text{path}, \text{path}^*) = \frac{\text{EditDistance}(\text{path}, \text{path}^*)}{\max(|\text{path}|, |\text{path}^*|)} \ . \tag{7}$$

2 Methods

I employ three methods to find the shortest path between given two nodes s and t in an incomplete network G^*, where the G^* is obtained by perturbing a network generated using the RHG. The first method, which is also the most straightforward, directly applies Dijkstra's algorithm to G^*. Next I will introduce the second method, which is based on the paper [1].

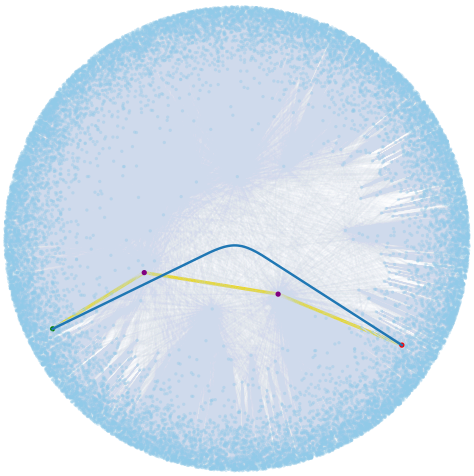

Figure 1: Visualization of the shortest path and geodesic in network 1. The yellow path is the shortest path between node 12681 and node 16010, while the blue curve is the geodesic between these two nodes (See section II.C of supplementary material for [1]). Nodes positions are determined by their true hyperbolic coordinates.

2.1 Distance to Geodesic

An important phenomenon identified in [1] is that the nodes lying on shortest path between endpoints s and t are located close to the geodesic connecting s and t. I also verify this observation through a simple visualization of the shortest path and geodesic, as shown in Figure 1.

For the network G^*, where hyperbolic coordinates are not directly available, we first need to infer them by embedding the nodes into hyperbolic space. Here, I use the embedder HyperLink [2]. The HyperLink use MLE approach to infer the nodes' hyperbolic coordinates with given the parameter T , α (γ should be provided in command) and the proportion of missing edges p.

Once the perturbed network G^* is embedded, we can estimate the shortest path between s and t through following procedure. First, we compute the distance of each other nodes to the geodesic $\widehat{s,t}$. This distance can be approximately calculated as [1]:

$$x_{i,\widehat{s,t}} = \frac{1}{2}(x_{is} + x_{it} - x_{st}) + \ln 2 \tag{8}$$

Next, we select the top m nodes with the smallest distances to the geodesic as candidate nodes that potentially lie along the shortest path. The parameter m, which is related to the length of shortest path, needs to be specified in advance.

Finally, we sort these m candidate nodes by their distance to source node s and reconstruct the inferred shortest path accordingly.

2.2 Dijkstra on fully connected weighted network

Inspired by RHG, I propose an alternative method to infer the shortest path in G^*. After obtaining the hyperbolic coordinates of nodes using HyperLink, I compute the connection probability between every non-adjacent node pair based on Equation (5), given parameter T and R.

Next, I connect these node pairs with an edge weighted by inverse connection probability: $\frac{1}{P(e_{ij})}$ and constructing a fully connected weighted network. This weighting reflects the likelihood of connectivity in hyperbolic space—lower probability corresponds to higher traversal cost. Finally, I apply Dijkstra's algorithm to this weighted network to infer the shortest path between the source and target nodes.

3 Experiment

For large network, such as the Network 1 and Network 2, which each contain approximately 20,000 nodes, the HyperLink algorithm requires several days to complete the embedding, making it impractical for experimental purposes. Therefore, I generate a smaller synthetic network G using RHG with parameter $n = 500$, $\alpha = 0.75$, $T = 0.5$ and $R = 13.604$(corresponding to an expected average degree $\hat{k} = 5$).

The experiment proceeds as follows. For various perturbation probabilities p, I randomly delete p proportion of edges from the giant connected component of G, resulting in a perturbed network G^*. I then extract the giant connected component G^{gcc*} of G^* and embed it using HyperLink to get the hyperbolic coordinates of nodes in G^{gcc*}.

Next, I randomly select a source node s and target node t within G^{gcc*}, and identify all the actual shortest paths between s and t in the original unperturbed network G. Then I apply the three methods to infer the shortest path path*. For the second method, the true length of shortest path is given; for the third method, the true RHG parameters T and R are provided.

To evaluate the path*, I use two metrics: overlap and normalized edit distance. For each metric, the best match between path* and all actual shortest paths is selected. The experimental results are presented in Figure 2.

As shown in Figure 2, the plain Dijkstra's algorithm outperforms the other two methods in most cases when the perturbation probability p is small. However, as p increases, its performance will decrease. This is expected, since higher perturbation levels result in more missing edges, which directly affect the accuracy of shortest path inference. In contrast, the distance to geodesic method maintains relatively stable performance even under high perturbation. Notably, when $p \simeq 0.8$, it surpasses the Dijkstra-based approaches in accuracy, demonstrating its robustness to network incompleteness.

My proposed method introduced in Section 2.2 achieves performance comparable to that of the plain Dijkstra's algorithm, though it is slightly worse in most cases. This performance gap may be attributed to the suboptimal design of the weighting strategy used in constructing the fully connected weighted network.

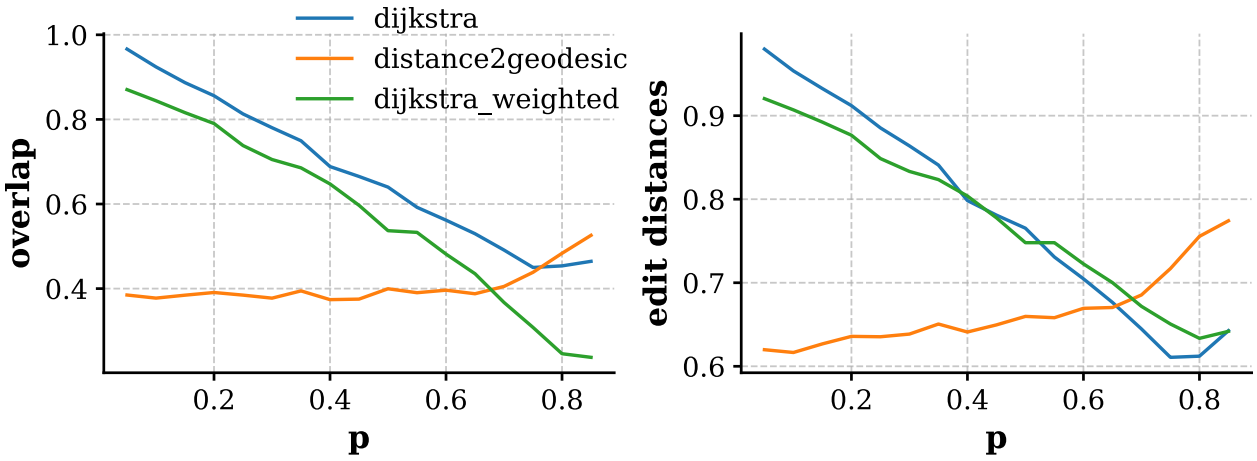

Figure 2: The experiment results for inferring the shortest path in incomplete network. For each p, I perturb network G for 20 times, and for each perturbed network G^{gcc*}, I random select 120 times shortest path endpoints pair to infer.

References

[1] Maksim Kitsak, Alexander Ganin, Ahmed Elmokashfi, Hongzhu Cui, Daniel A Eisenberg, David L Alderson, Dmitry Korkin, and Igor Linkov. Finding shortest and nearly shortest path nodes in large substantially incomplete networks by hyperbolic mapping. *Nature Communications*, 14(1):186, 2023.

[2] Maksim Kitsak, Ivan Voitalov, and Dmitri Krioukov. Link prediction with hyperbolic geometry. *Physical Review Research*, 2(4):043113, 2020.