



# CLOVE, a Travelling Salesman's approach to hyperbolic embeddings of complex networks with communities



Sámuel G. Balogh<sup>1,2</sup>, Bendegúz Sulyok<sup>2,3</sup>, Tamás Vicsek <sup>2</sup> & Gergely Palla <sup>2,3</sup>

The embedding of complex networks into metric spaces has emerged as a prominent area of research, accompanied by a diverse array of proposed methodologies. Low-dimensional hyperbolic spaces provide a natural target for such embeddings, facilitating an approximately uniform spatial distribution of nodes – even in scale-free networks – while enabling efficient navigability and accurate estimation of linking probabilities. Despite ongoing state-of-the-art advancements, hyperbolic embedding techniques increasingly exhibit diminishing marginal returns. Recent findings indicate that, following optimization, the communities within a complex network can be effectively represented as distinct angular sectors in the hyperbolic space. In this work, we present CLOVE, a scalable embedding approach that leverages this property through an iterative hierarchical arrangement of communities down to the level of individual nodes. A key step of our method involves determining the optimal angular ordering of communities at each hierarchical level, a challenge addressed by formulating and solving an instance of the Travelling Salesman Problem. Given that CLOVE surpasses many alternative techniques across various embedding quality metrics while maintaining high computational efficiency, it holds significant promise for downstream machine learning applications, including AI-driven pattern recognition.

The network approach for describing and analysing complex systems has become ubiquitous in the last two decades<sup>1–5</sup>, building on the fundamental concept of representing the interactions between the constituents of the studied system by a graph. A general approach for augmenting the network reflecting the structure of the web of connections (that serve as a sort of a skeleton for a complex system) is to apply network embedding techniques<sup>6–9</sup>. These methods are aimed at finding an optimal arrangement of the network in a metric space, thereby associating coordinates to the nodes of the network based on the network topology. These coordinates can be useful from several different aspects, e.g., they enable the prediction of missing links, can help navigation over the network, or may serve as input for further machine learning tasks such as node classification, community finding, etc.

Although the majority of network embedding techniques operate in Euclidean spaces, hyperbolic methods offer an alternative approach with unique advantages<sup>9,10</sup>. Probably most important is that while Euclidean algorithms usually embed in high dimensions, hyperbolic approaches can yield good-quality embeddings already in 2 dimensions. The intuitive reason behind this is that the exponential growth of the volume as a function of

the radius for spheres in hyperbolic spaces allows more “freedom” in node placement compared to the case of Euclidean spaces, where the volume is increasing only like a power-law<sup>11</sup>. Most of the hyperbolic embedding methods work in the native representation of the hyperbolic space, which in 2 dimensions is often referred to as the native disk. In this representation, the radial coordinates are usually strongly coupled with the node degree, where the high-degree nodes tend to be placed closer to the centre of the native disk, while the low-degree nodes occupy the disk periphery. (A brief description of the native disk and hyperbolic geometry is given in Methods).

Several different hyperbolic embedding algorithms have been proposed in the literature, ranging from likelihood optimization with respect to hyperbolic network models<sup>12,13</sup>, to non-linear dimensionality reduction techniques based on Laplacian matrices<sup>14,15</sup>, and dimensionality reduction of Lorentz matrices using the hyperboloid model<sup>16,17</sup>. The family of coalescent embeddings applies dimensionality reduction to various pre-weighted matrices that capture the network structure<sup>18</sup>. In addition, hybrid methods that combine dimensionality reduction with local optimization<sup>19–21</sup> have also been developed. A related approach, the Minimum Curvilinear

<sup>1</sup>National Laboratory for Health Security, HUN-REN Alfréd Rényi Institute of Mathematics, Budapest, Hungary. <sup>2</sup>Department of Biological Physics, Eötvös Loránd University, Budapest, Hungary. <sup>3</sup>Health Services Management Training Centre, Semmelweis University, Budapest, Hungary.

e-mail: [palla.gergely@emk.semmelweis.hu](mailto:palla.gergely@emk.semmelweis.hu)

Automaton (MCA), employs minimum spanning trees to generate hyperbolic embeddings of complex networks<sup>22</sup>.

Hyperbolic embeddings have also been explored in machine learning and neural networks through various methods, such as replacing Euclidean vector spaces in the Skipgram model with hyperbolic counterparts<sup>23</sup>, introducing hyperbolic graph convolutional networks operating on the hyperboloid model of hyperbolic space<sup>24</sup>, and applying implicit hierarchical learning within hyperbolic space<sup>25</sup>.

It is important to note that hyperbolic embeddings are also closely coupled with the modular structure of networks<sup>26–29</sup>. On the one hand, graphs generated by geometric network models operating explicitly in hyperbolic spaces have been shown to exhibit a highly pronounced modular nature, wherein communities (corresponding to densely connected modules in the networks) occupy tightly localized domains within the geometric space sharing an asymptotically negligible fraction of inter-connections between one another<sup>27–29</sup>. On the other hand, this separability of network modules in the metric space can also be considered a fundamental prerequisite for high-quality hyperbolic embeddings, suggesting a deep connection between the embeddings and the community structure of complex networks<sup>26</sup>. Indeed, when embedding a given network, we essentially mean to provide an  $f_E$  mapping function of the form  $f_E : V \rightarrow \mathbb{R}^d$  equipped with a metric, where  $V$  denotes the set of nodes in the network and  $d$  is the dimension of the embedding space. In parallel, partitioning the same network is equal to the construction of a  $f_P : V \rightarrow \mathbb{N}$  mapping, which can be regarded to some extent as a coarsened version of its embedding<sup>26</sup>. Furthermore, it has been shown that the embedding technique based on the Laplacian Eigenmap is simply a specific instance of a more general trace maximization problem involving the generalized modularity matrix<sup>30</sup>.

Notably, the emergence of this formal analogy between embedding and partitioning gives rise to a variety of intriguing implications; e.g., one can reasonably assess the quality of hyperbolic embeddings by quantifying the extent to which nodes within the same community have similar angular coordinates in the embedding space (angular coherence of the communities). As expected, state-of-the-art hyperbolic embedding methods such as the coalescent embedding<sup>18</sup>, or the *D*-Mercator<sup>20</sup> perform excellently in this respect, as shown through specific quality measures capturing the communities' angular coherence<sup>20,31</sup>. Additionally, an efficient Markov chain Monte Carlo algorithm – BIGUE (Bayesian Inference of a Graph's Unknown Embedding) – has been introduced, which uses a set of cluster (community) based transformations to improve the exploration of the posterior distribution<sup>32</sup>.

Perhaps, an even more explicit manifestation of the previous analogy emerges when the hyperbolic embeddings of a given network are constructed based on the information encoded in its community structure<sup>33,34</sup>. Herein, the authors introduce a family of embedding methods that rely on the iterative assignment of the network communities and their respective sub-communities to distinct angular sectors on the native disk. It is important to note, however, that the crux of the aforementioned procedure lies in the reasonable arrangement of communities, a task that unfortunately lacks a well-principled systematic solution scheme. Although a computationally very fast greedy-like methodology has been proposed under the name of Hyperbolic Mapping based on the hierarchical Community Structure (HMCS) method<sup>34</sup>, our empirical findings show its diminished efficiency under specific circumstances. Driven by this incompleteness, in the present paper we propose a hyperbolic embedding method built upon the modular structure of networks, where the arrangement problem of the found communities is solved according to the renowned Travelling Salesman Problem<sup>35–38</sup> (TSP). Originally, the TSP focuses on finding the minimum weight Hamiltonian path, which, in this context, can directly be used to determine the angular order of (sub-)communities on the native disk. Since the angular arrangement is optimised according to a well-known route finding problem borrowed from the domain of computer science, we abbreviate our method as CLOVE, standing for Cluster Level Optimised Vertex Embedding. The core idea of this method is to identify communities within the network, build a weighted super-graph where each node

represents a community, and then use approximate solutions to the Travelling Salesman Problem to find a minimum-weight cycle that determines the placement order of the communities in the hyperbolic disk. This process is applied hierarchically at progressively finer scales, iteratively refining the positions of smaller subgraphs, until no additional community structure can be uncovered.

On the one hand, since the TSP has to be solved only on relatively small networks, the method is surprisingly fast, capable of embedding networks having millions of nodes in just a few hours. On the other hand, due to the multi-scale optimisation process, the quality of the obtained embedding is high according to various different quality measures. In the upcoming sections, we introduce CLOVE in detail, and compare its performance with various state-of-the-art embedding algorithms in terms of both the computation time and the quality of the end result.

## Results

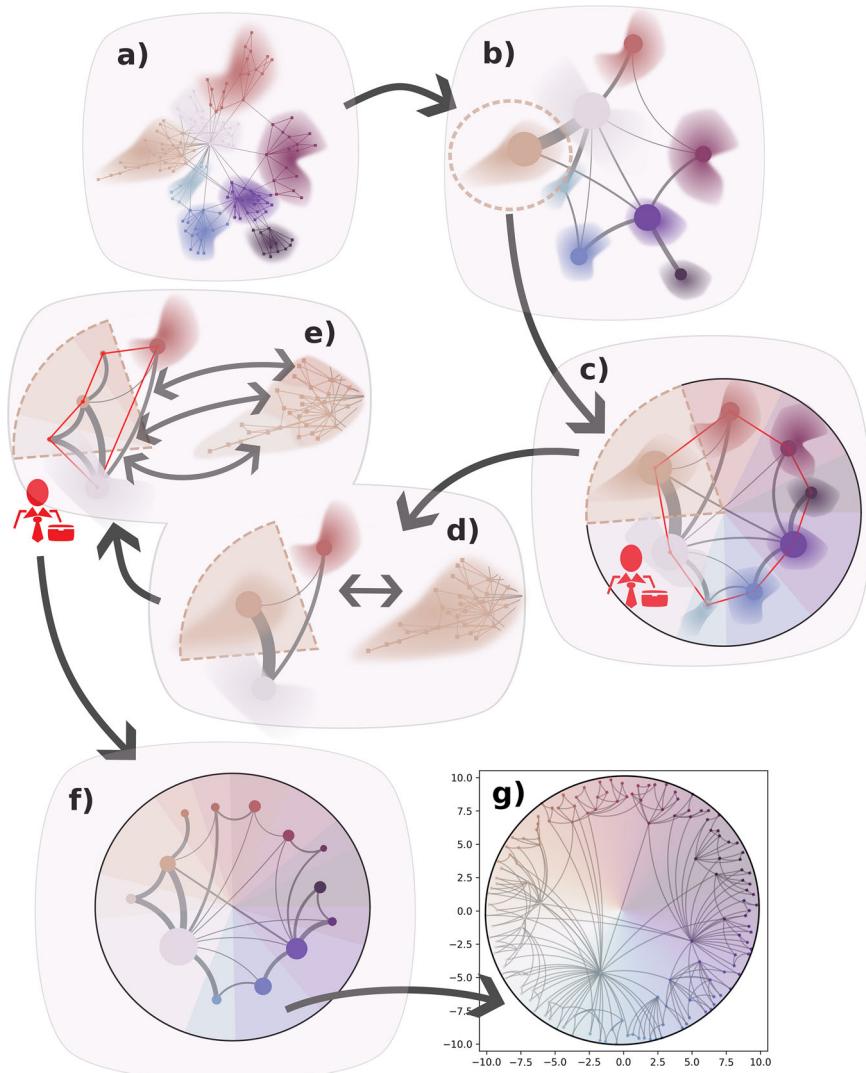
### Embedding networks into hyperbolic space via the Travelling Salesman problem

The Travelling Salesman Problem (TSP) is one of the most well-known and extensively studied optimization problem in computer science and mathematics<sup>35–38</sup>. It deals with the issue of finding the shortest possible route that a salesman can take to visit a given set of cities and return to the starting point, visiting each city only once (tour). The problem can effectively be modelled as a graph, wherein the nodes represent the cities to be visited by the salesman, whereas the edges of the graph correspond to the paths along which the salesman may travel. Each edge connecting two cities in the graph is assigned a weight being equivalent to the distance or cost of travelling between the two cities. In addition, provided that the resulting graph is fully connected, i.e., all pairwise distances are known in advance, the TSP can eventually be reformulated as the task of finding the shortest Hamiltonian cycle in the graph.

In our approach, the first step is the identification of the communities in the network and the definition of weighted links between them based on their level of connectivity. Notably, the pre-weighting scheme we apply satisfies the triangle inequality, endowing the assigned weights with the role of virtual distance measures encapsulating the hyperbolic proximity between the detected communities (see Supplementary Note 1.2 in the Supplementary Information for more details). Consequently, this metric property ensures the seamless adaptation of the TSP to unveil the optimal angular arrangement of the modules in the native disk. As a next step of the algorithm, sub-modules are identified separately within each community and are arranged locally, again with the help of the TSP. This iteration is continued hierarchically, always dividing the communities at a given level into smaller parts, defining weighted links between the found sub-modules and optimising the angular arrangement of the sub-modules within the original community via the TSP. After settling the angular coordinates in the above manner, the radial coordinate  $r$  of the nodes are determined based on the node degree  $k$ , following a simple relation between  $r$  and  $k$  established in multiple hyperbolic network models<sup>11,19,39</sup> and used in various other embedding methods<sup>13,21,33,34</sup> (the details are described in Methods).

An illustrating flow-chart of our algorithm is presented in Fig. 1, where the communities detected in the original network are marked by the different colours in Fig. 1a. This is followed by the definition of a complete, weighted graph between the modules found (shown in Fig. 1b), where the strength of a given connection roughly quantifies the extent of surprise associated with observing that link relative to a configuration-model-like baseline, while the resulting weights also satisfy the triangle-inequality (see Methods and Supplementary Note 1.2 in the Supplementary Information for details). By solving the TSP on this weighted graph and taking the found shortest Hamiltonian path, we can arrange the communities on the native disk representation of the 2 dimensional hyperbolic space (Fig. 1c), where each community occupies an angular range proportional to its size, measured in the number of member nodes.

**Fig. 1 | Illustration of the Cluster Level Optimised Vertex Embedding (CLOVE) algorithm.** **a** A network with the detected communities indicated by the different colours. **b** The weighted network between the communities. **c** Optimal arrangement of the communities on the native disk according to the solution of the Travelling Salesman Problem (TSP) on the weighted network in **(b)**. **d** Zooming into one of the modules with the two neighbouring communities also shown. **e** Sub-modules in the previous community and their optimal arrangement based on the TSP, taking into account also the neighbours from the top-level. **f** Optimal arrangement of the sub-modules at the second level based on the local TSPs. **g** Shows the network embedded into the native disk.



In the next stage, we iterate over the communities, locating and arranging sub-modules within each of them. These sub-modules are found by simply applying the same community finding method as in the case of the original network, but now only on the sub-graph of the given community (detached from the original network). Similarly to the top-level communities, we arrange the sub-modules based on the TSP; however, this time the weighted graph between the sub-modules also includes two neighbouring communities from the top-level as indicated by Fig. 1d, e. The reason behind this is that these provide “anchors” for the sub-modules, allowing an arrangement that uses information coming from the surroundings of the original module. The angular range of the sub-modules is again proportional to their size.

The above procedure is repeated hierarchically over each sub-module (and the even smaller sub-modules found within). When reaching to the point where the community finding method does not break the sub-module to further smaller communities, one can either use a simple heuristic for the arrangement of the nodes within the sub-module (detailed in Methods) or treat the individual nodes as if they were the communities to be arranged on the next level below (and use again the TSP as in the case of the higher levels in the community hierarchy).

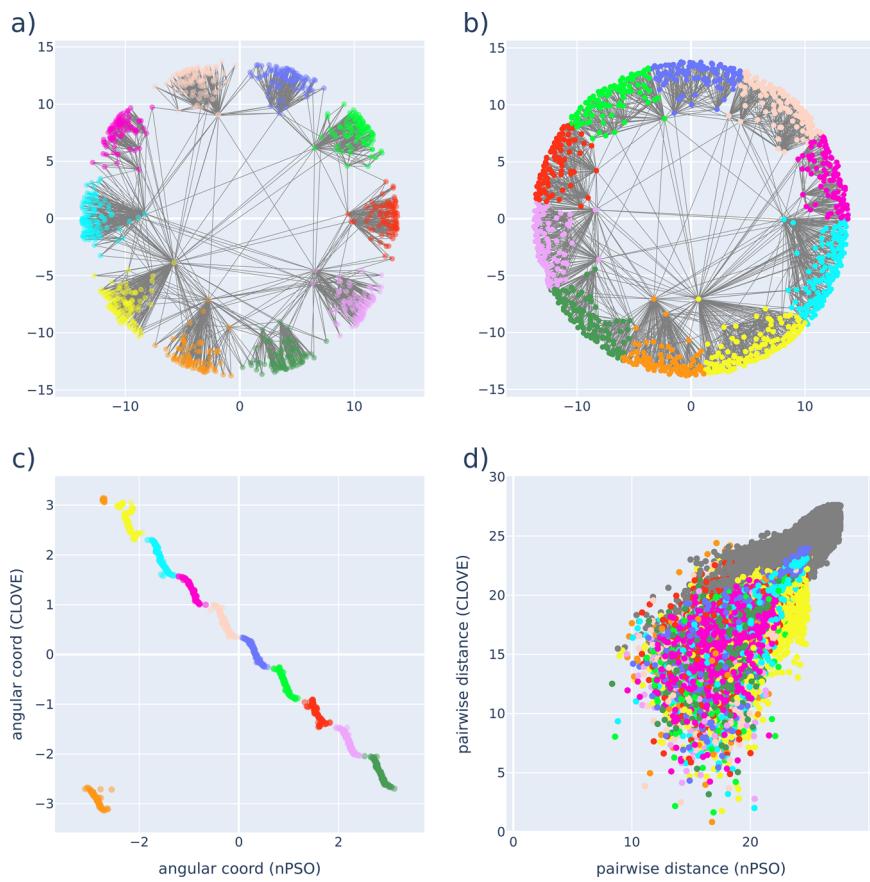
In addition to the method for arranging the nodes at the lowest level of the hierarchy, our framework also allows a large flexibility at multiple strategic points of the embedding algorithm. First, the applied community finding method can be freely chosen (we use Leiden<sup>40</sup>, corresponding

to a fast method that guarantees well-connected communities). Second, there are several different possibilities for solving the TSP (we apply the Christofides algorithm, sometimes referred to as Christofides-Serdyukov<sup>41,42</sup> algorithm with a threshold accepting boosting scheme<sup>43</sup>). Third, to make the approach even more general, our framework also allows for replacing the TSP-based angular arrangement by other arrangement procedures (in the present work, we included a solution that relies on the minimum curvilinear attachment<sup>22</sup> (MCA) process). A fully detailed description of our embedding algorithm is given in Methods.

#### Tests using synthetic networks

The efficacy of our embedding framework is best demonstrated using networks with known geometric properties and modular structure. The non-uniform Popularity Similarity Optimisation (nPSO) model<sup>44</sup> provides a natural method for generating such hyperbolic networks with controllable communities. The original Popularity Similarity Optimisation (PSO) model<sup>39</sup>, a well-known hyperbolic network model, places nodes sequentially into the native disk representation of the 2d hyperbolic space with logarithmically increasing radial coordinates and uniformly random angular coordinates, with a connection probability decreasing as a function of the hyperbolic distance. This model generates highly clustered, small-world and scale-free random graphs, reproducing the most important key characteristics of real-world networks. The nPSO model extends the PSO model by sampling the angular coordinates from either a Gaussian-mixture

**Fig. 2 | Test for a hyperbolic network with known ground truth communities.** **a** A non-uniform Popularity Similarity Optimisation (nPSO) network of size  $N = 1000$  with 10 planted communities, indicated by the different node colours. The further parameters of the nPSO network were  $m = 4$ ,  $\beta = 0.5$  and  $T = 0.1$ . **b** The embedding of this network with Cluster Level Optimised Vertex Embedding (CLOVE), where nodes are coloured according to the ground truth communities in **(a)**. **c** The angular coordinate of the nodes in the embedding, as a function of the angular coordinate in the original nPSO network. **d** The hyperbolic distance between node pairs in the embedding, as a function of the hyperbolic distance in the original nPSO layout. Colours indicate node pairs in the same ground truth community, whereas gray colour indicates nodes in different communities.



distribution or a Gamma-mixture distribution instead of a simple uniform random distribution. This creates denser angular regions, leading to the formation of communities. (A more detailed description of the PSO and nPSO models is provided in Supplementary Note 3.1 in the Supplementary Information.)

In Fig. 2a we show an nPSO network of size  $N = 1000$  where the angular coordinates were drawn from a mixture of 10 equally spaced Gaussian distributions (having also equal standard deviations and also uniform relative weights). The layout generated by embedding this network with CLOVE, displayed in Fig. 2b, demonstrates that our algorithm correctly captured the angular ordering of the ground truth communities. Although a rotation and mirroring of the original angular ordering can be observed, this is natural, since the link generation process in the nPSO model depends solely on the hyperbolic distance between the nodes rather than their absolute coordinate in the native disk. In Fig. 2c we plot the angular coordinate in the embedding,  $\theta^{\text{CLOVE}}$ , as a function of the original angular coordinate  $\theta^{\text{nPSO}}$ , illustrating that CLOVE could also correctly determine the ordering of the nodes within the communities in most of the cases. Finally, Fig. 2d shows a scatter-plot of the hyperbolic distance between node pairs in the embedding as a function of the original hyperbolic distance during the nPSO network generation. The Pearson correlation coefficient,  $R_{\text{Pearson}} = 0.930$ , and the Spearman's rank correlation coefficient  $R_{\text{Spearman}} = 0.938$  indicate a very strong agreement. These high correlation values confirm that CLOVE not only accurately captured the community structure but also faithfully reproduced the distance relationships between the nodes.

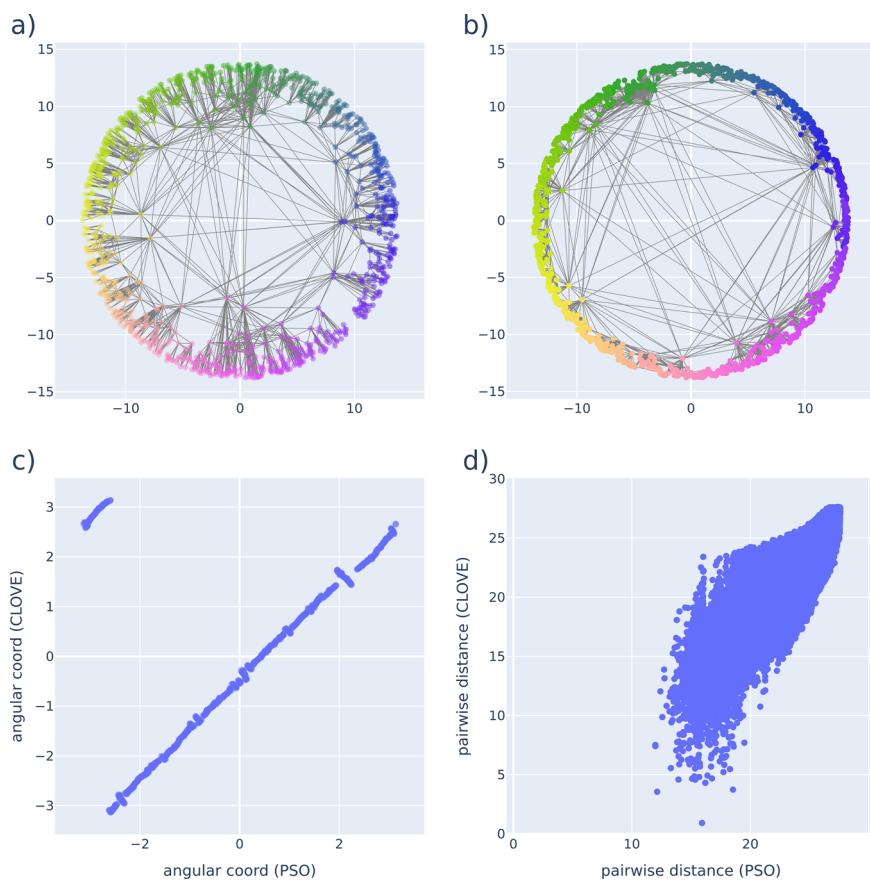
In addition to the above discussed correlation coefficients, we also evaluated the Angular Separation Index<sup>31</sup> (ASI), quantifying how well CLOVE separated the ground truth communities, and the C-score, providing an alternative similarity measure between the compared angular orderings of the nodes in the entire network. (A detailed description of the ASI and the C-score is given

in Methods.) For the embedding shown in Fig. 2, we obtained ASI = 0.989 and C-score = 0.948. By repeating this experiment on 20 nPSO networks (with the same parameters as in the example shown in Fig. 2), we also calculated the average value for these indicators, resulting in  $\overline{R_{\text{Pearson}}} = 0.859$ ,  $\overline{R_{\text{Spearman}}} = 0.823$ ,  $\overline{\text{ASI}} = 0.967$  and  $\overline{\text{C-score}} = 0.819$ . These high values show that CLOVE performs very well on hyperbolic networks with known ground truth communities.

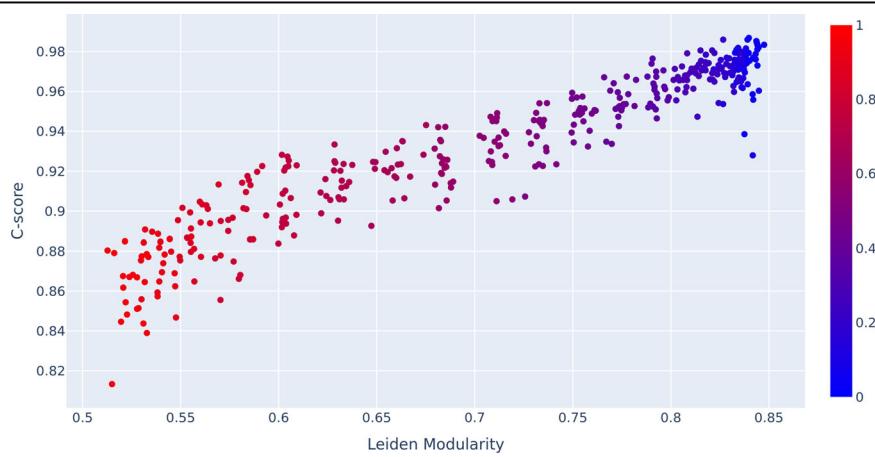
Apart from nPSO networks, we also tested CLOVE on synthetic networks generated by the original PSO model. Although these lack adjustable planted communities, according to recent works, they still possess a strong modular structure where communities arise in a somewhat spontaneous manner<sup>28,29,45</sup>. This makes PSO networks well suited to be embedded by CLOVE, which is demonstrated in Fig. 3, showing the results for a PSO network of  $N = 1000$  nodes in a similar fashion to Fig. 2. Apparently, CLOVE produced an embedding (Fig. 3b) that shows a remarkably high similarity with the original layout of the PSO network (Fig. 3a). According to Fig. 3c, the angular ordering of the smaller and larger subgraphs in the embedding can achieve a perfect match with the original down to the level of individual nodes in some parts of the system. However, occasional small regions with reversed ordering can also be observed. Nevertheless, the Pearson correlation coefficient,  $R_{\text{Pearson}} = 0.897$ , the Spearman's rank correlation coefficient,  $R_{\text{Spearman}} = 0.882$ , and the C-score, = 0.977 indicate a very high overall similarity between the embedding and the original PSO network.

Similarly to the nPSO networks, we repeated the embedding experiment for the PSO network as well, and calculated the average value of these indicators over 20 instances, resulting in  $\overline{R_{\text{Pearson}}} = 0.829$ ,  $\overline{R_{\text{Spearman}}} = 0.865$  and  $\overline{\text{C-score}} = 0.882$ . These indicate that CLOVE is also very well suited for embedding PSO networks, representing an emblematic example for hyperbolic networks with homogeneous angular node coordinates in the hyperbolic plane.

**Fig. 3 | Test on a homogeneous hyperbolic network.** **a** A PSO (Popularity Similarity Optimisation) network of size  $N = 1000$ , where node colours are distributed according to the angular coordinate. The further parameters of the PSO network were  $m = 4$ ,  $\beta = 0.5$ , and  $T = 0.1$ . **b** The embedding of this network with CLOVE (Cluster Level Optimised Vertex Embedding), where nodes are coloured according to the ground truth coordinate in (a). **c** The angular coordinate of the nodes in the embedding, as a function of the original angular coordinate. **d** The hyperbolic distance between node pairs in the embedding, as a function of the original hyperbolic distance.



**Fig. 4 | The C-score between the Cluster Level Optimised Vertex Embedding (CLOVE) and the ground truth coordinates as a function of the modularity for Popularity Similarity Optimisation (PSO) networks.** We embedded PSO networks with the temperature parameter ranging between  $T = 0$  and  $T = 1$  while keeping the other parameters fixed at  $N = 1000$ ,  $m = 4$  and  $\beta = 0.5$ , and located the communities using the Leiden algorithm. Each symbol in the scatter plot corresponds to a single PSO network where the  $T$  parameter is indicated by the colour of the symbol. The increasing tendency of the scatter plot shows that we can expect a higher similarity between the embedding and the original coordinates when the modular structure of the input network is strong.



PSO networks also provide an ideal testing ground for the sensitivity of CLOVE concerning the strength of the modular structure. The fact that CLOVE embeds the input networks based on the communities found in their structure implies that this approach is expected to work best for systems with a strong modular organisation. The detailed examination of the parameter space of PSO networks for the generated community structure revealed that the modularity (corresponding to the most widely used quantity for measuring the strength of communities<sup>46–48</sup>) is maximal when the temperature parameter is set to  $T = 0$  and shows a decreasing tendency if  $T$  is increased<sup>28,29,45</sup>.

According to the above, in order to study the connection between the embedding quality and the strength of the community structure, we also embedded PSO networks generated with varying  $T$  and all the other

parameters kept fixed. In Fig. 4, we show the C-score calculated between the embedding and the original PSO coordinates as a function of the modularity of the communities found by the Leiden algorithm. Although the C-score is consistently high across all modularity values, the distinct increasing trend of the point cloud indicates that (in agreement with the expectations) CLOVE provides the best embeddings when the input network possesses the strongest modular structure.

In Supplementary Note 4.2 in the Supplementary Information, we also extend the studies of nPSO and PSO networks by embedding nPSO networks with a hierarchical ground truth community structure. The results confirm that thanks to the efficacy of the Leiden algorithm, CLOVE can recover and arrange the hierarchically nested communities with high fidelity.

**Table 1 | Average quality scores for small and medium sized real networks**

	MA	EPP	AUC	GR	GS	GE	Running Time (min.)	Peak Mem. (GB)
CLOVE(default)	0.365	0.485	0.962	0.505	0.426	0.180	0.496	0.426
CLOVE (with SA)	0.364	<b>0.487</b>	0.962	<b>0.507</b>	<b>0.428</b>	<b>0.181</b>	0.500	0.453
CLOVE (Louvain)	0.362	0.485	0.962	0.500	0.421	0.178	0.546	0.438
ncMCE (hyperbolic)	0.328	0.173	0.946	0.168	0.147	0.066	5.918	9.475
Mercator	<b>0.506</b>	0.449	<b>0.976</b>	0.329	0.299	0.119	123.921	4.176
HMCS	0.331	0.237	0.957	0.300	0.262	0.117	<b>0.220</b>	<b>0.356</b>

We show the results for the mapping accuracy, MA, the edge prediction precision, EPP, the area under the receiver operating characteristic curve, AUC, the greedy routing score, GR, the greedy success rate, GS and the greedy routing efficiency, GE, averaged over 10 real networks with size ranging between  $N = 1000$  and  $N = 20,000$  nodes. For each indicator, the best performing value is shown in bold. Besides the quality scores, we also display the running time in seconds and the peak memory usage in GB. In the top part of the table, we list the scores obtained for CLOVE with default settings, for CLOVE with simulated annealing optimisation during the solution of the TSP problem and for CLOVE with Louvain communities. For comparison, in the bottom part of the table, we give the results for hyperbolic ncMCE, Mercator and HMCS.

**Table 2 | Average quality scores for large real networks**

	MA	EPP	AUC	GR	GS	GE	Running Time (min.)	Peak Mem. (GB)
CLOVE(default)	<b>0.278</b>	0.405	<b>0.964</b>	0.304	0.223	<b>0.079</b>	663.259	<b>2.708</b>
CLOVE (with SA)	0.276	<b>0.409</b>	<b>0.964</b>	<b>0.306</b>	<b>0.225</b>	<b>0.079</b>	693.016	3.163
CLOVE (Louvain)	0.277	0.406	0.962	0.291	0.214	0.076	649.248	3.181
HMCS	0.271	0.108	0.955	0.073	0.059	0.021	<b>89.061</b>	3.113

We display the measured average scores for the mapping accuracy, MA, the edge prediction precision, EPP, the area under the receiver operating characteristic curve, AUC, the greedy routing score, GR, the greedy success rate, GS and the greedy routing efficiency, GE, averaged over 17 real networks with size ranging between  $N = 20000$  and  $N = 2.7 \cdot 10^6$  nodes. For each indicator, the best performing value is shown in bold. Besides the quality scores, we also display the running time and the peak memory usage in GB. In the top part of the table, we list the scores obtained for CLOVE with default settings, for CLOVE with simulated annealing optimisation during the solution of the TSP problem and for CLOVE with Louvain communities. For comparison, in the bottom row we give the results for HMCS.

### Comparison with current state-of-the-art methods on real networks

By moving from synthetic networks to real systems, we tested CLOVE on several real networks that represent the network of connections in various complex systems. The size of these networks spanned from  $N = 10^3$  nodes to  $N = 2.7 \cdot 10^6$  nodes and the studied systems belonged to various domains, including social, biological and technological networks alike. We compared the performance of our approach with different state-of-the-art hyperbolic embedding methods according to multiple quality scores. These include the mapping accuracy<sup>7</sup>, MA, measuring the correlation between the shortest path distance and the geometric distance in the embedding space, the edge prediction precision, EPP, and the area under the receiver operating characteristic curve, AUR, in graph reconstruction<sup>49,50</sup>, the greedy routing success rate<sup>51</sup>, GR, corresponding to the fraction of successful paths when navigating according to the node coordinates in the network, the greedy routing score<sup>18</sup>, GS, taking into account also the length of the paths during greedy routing and the greedy routing efficiency<sup>52</sup>, GE, comparing the geometric distances and the projected greedy routing paths (the precise definition for all of these measures is provided in Methods). Before actually analysing the performance of CLOVE for real networks, we also tested the behaviour of these quality scores using CLOVE embeddings of the synthetic PSO networks where the strength of the community structure was varying. We detail the results in Supplementary Note 3.3.3 in the Supplementary Information, where we observed high correlation coefficients with modularity. This is in agreement with the tendency shown in Fig. 4, which indicates that for input networks with a strong modular structure, CLOVE is expected to produce better results compared to systems where the communities are blurred or absent.

The alternative embedding methods - serving as a baseline for comparison - were the following ones: i) The hyperbolic non-centered minimum curvilinear embedding (ncMCE)<sup>18</sup>, relying on the dimension reduction of a weighted matrix encoding the distance relations; ii) Mercator<sup>19</sup>, combining the dimension reduction of the Laplacian matrix with a local optimisation concerning the random hyperbolic graph; iii) The HMCS method<sup>34</sup>, taking advantage of the hierarchical community structure of networks in a similar

fashion to our approach, however, arranging the modules and sub-modules in a simple greedy fashion.

In Table 1 we show the quality scores averaged over 10 real networks falling into the size range between  $N = 1000$  and  $N = 20000$ . (In Tables S2–S11 in the Supplementary Information we also display the results for the individual networks one by one.) In addition to quality scores, Table 1 also provides the running time and the peak memory usage during the different processes. According to the results for the different algorithms, Mercator achieved far the best mapping accuracy score and the best AUC value, whereas CLOVE with an additional simulated annealing during the solution of the TSP turned out to be the best according to the edge prediction precision, the greedy routing score, the greedy success rate and the greedy routing efficiency. We note that all CLOVE versions outperformed both HMCS and hyperbolic ncMCE according to all quality scores, and also Mercator regarding the greedy routing-based scores (GR, GS and GE).

In terms of time consumption, HMCS was, as expected, the fastest, followed by the various CLOVE variants. Moreover, in our experiments, all CLOVE variants were approximately 10 times faster than hyperbolic ncMCE and over 200 times faster than Mercator. For a more detailed discussion of CLOVE's time complexity, see the Methods section. Finally, HMCS has the lowest peak memory usage, where the results for the different CLOVE versions are not far behind and are considerably smaller compared to the memory usage of hyperbolic ncMCE and Mercator.

In Table 2 we provide the average values for the studied embedding quality scores in large networks, corresponding to systems where the number of nodes varies between  $N = 2 \cdot 10^3$  and  $N = 1.3 \cdot 10^6$ . The same quality indicators for the individual networks are listed similarly in tables S12–S25 in the Supplementary Information. An important difference compared to the case of smaller networks is that, since the scores are defined as various sums over node pairs, their exact evaluation becomes unfeasible, and therefore, we relied on sampling from all possible node pairs when calculating the quality measures. In Supplementary Note 4 in the Supplementary Information, we examine the relation between the exact quality score values and their estimates based on sampling in smaller systems, arriving at the conclusion that sampling offers a reasonably precise estimate

**Fig. 5 | Resource usage of the studied algorithms as a function of the size of the embedded networks.** We plot the average running time in (a) and the peak memory usage in (b), with the colour code of the different methods given in the legends. The resource usage for the Cluster Level Optimised Vertex Embedding (CLOVE) with default settings is shown in dark blue, for CLOVE with simulated annealing optimisation at the solution of the Travelling Salesman Problem in red, for CLOVE based on communities found by the Louvain algorithm instead of Leiden in green, for Hyperbolic Mapping based on the hierarchical Community Structure (HMCS) in purple, for Mercator in orange and for the hyperbolic non-centered minimum curvilinear embedding (ncMCE) in light blue.



of the exact values already at relatively low frequency values. A further difference compared to Table 1 is that due to the larger resource requirements in terms of computation time or memory, Mercator and the hyperbolic ncMCE method were not applied to the larger networks.

According to the results shown in Table 2, CLOVE significantly outperforms HMCS according to all quality scores at the cost of having a roughly 8 times as large computation time. While CLOVE with extra simulated annealing seems to be the best among the different CLOVE versions in Table 2, when examining the detailed list of results for the individual networks in tables S12–S25 in the Supplementary Information, it becomes clear that in certain systems it is the default version or the one relying on Louvain communities that achieves the best result. However, in addition to MA and AUC, a clear gap is always present between CLOVE scores and HMCS scores.

In Fig. 5 we show the computational resource usage of the studied embedding methods as a function of the network size (measured in the number of nodes). Naturally, all of the curves show an overall increasing tendency; however, they are not strictly monotonic, indicating that besides the size, also the structure of the network can have a strong effect on the amount of computational resources needed for the embedding. The comparison between the different curves leads to a conclusion that is consistent with the previous results shown in Tables 1, 2: As expected, among the studied methods HMCS is the fastest followed by our different CLOVE implementations. The time curves for hyperbolic ncMCE and Mercator seem steeper than the previous approaches, and these methods run slower by at least one order of magnitude at the upper size limit of smaller networks ( $N = 20,000$  in our study). In parallel, the peak memory usage (Fig. 5b) displays two bundles of curves, where the CLOVE implementations and HMCS show very similar memory needs, which are considerably more moderate compared to those of Mercator and hyperbolic ncMCE.

#### Hyperbolic maps of real networks with ground-truth modules

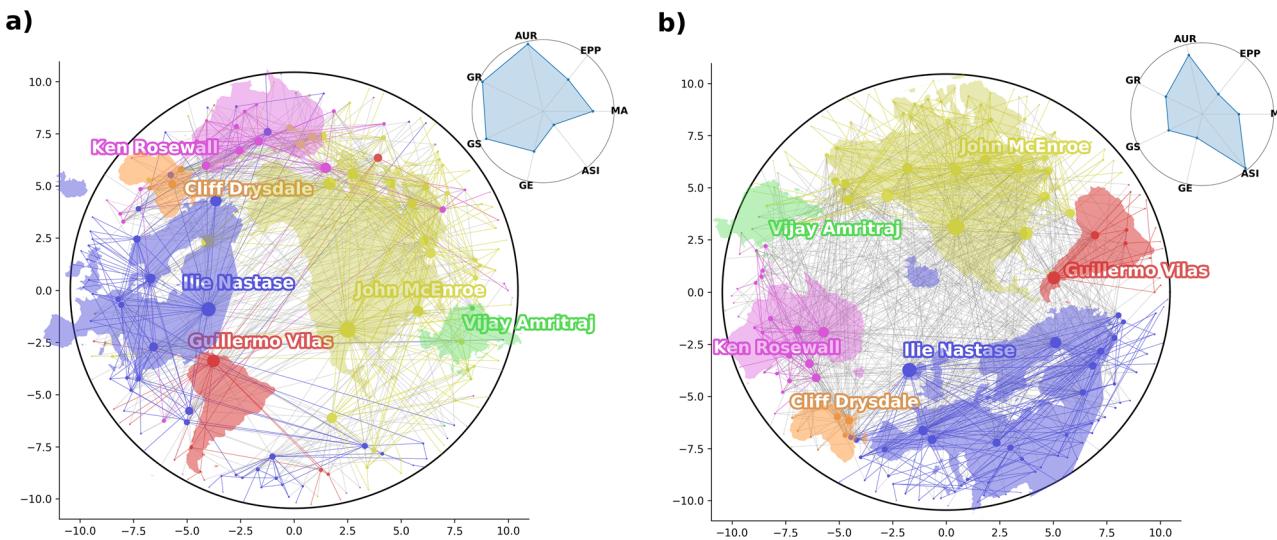
In this section, we demonstrate that the embeddings generated by our approach can provide intuitive node arrangements in the native disk for different complex systems. For a small fraction of the networks we analysed, “ground truth” modules and/or additional node labels were also available besides the network topology. Although our method is agnostic concerning any extra node labels and calculates the coordinates solely based on the network structure, still, the organisation of the obtained layouts is meaningful also in the light of these additional features.

**The network of tennis matches between ATP players.** ATP stands for the Association of Tennis Professionals, which serves as the governing body for men’s professional tennis. It is responsible for overseeing and managing various aspects of this sport, including the organization of tournaments and the establishment of player rankings. Related to that, here we examine a tennis dataset accessible at<sup>53</sup>, with a central question in mind: Does the network representing matches between tennis players fit well to the two-dimensional hyperbolic space? Can the two-dimensional hyperbolic space efficiently host the network representing the matches between ATP tennis players?

In order to investigate this question in detail, we first build the network by considering the matches between the top-ranked ATP players who competed against each other during the period from 1969 to 1989 and participated in at least 7 official matches. Subsequently, we apply the CLOVE algorithm with various parameter settings to map this network to the native disk representation of the two-dimensional hyperbolic space. Our approach consists of two rather different embedding strategies. In the first case, we run our algorithm with its default settings, where communities are identified and arranged in a nested fashion using a fast community detection method (e.g., Louvain or Leiden) applied across increasingly finer scales. The resulting hyperbolic layout is displayed in Fig. 6a, along with the angular sectors where players from distinct continents are predominantly clustered. Moreover, in Fig. 6a we also indicate the position of a prominent tennis player for each continent.

In our second embedding approach, the identification of network modules to be positioned on the native disk is not dictated by the output of a pre-defined community detection method. Instead, we rely on a two-level dendrogram that incorporates ground-truth information regarding the ethnicities of the players. The first level pertains to the nationalities of the players, while the second level maps nations to continents, thus forming a complete dendrogram of communities. This regional dendrogram is passed to the embedding algorithm, which then arranges the communities accordingly, again based on the TSP. We show the obtained hyperbolic layout in Fig. 6b, where, similarly to Fig. 6a, both the angular sectors corresponding to the continents and the same top-tier players for each continent are highlighted.

Overall, by observing the quality scores displayed at the top-right corner of the panels in Fig. 6, we can deduce that the embedding quality is superior in the first scenario, i.e., when the modules to be arranged on the disk are derived from a community detection method, rather than being constructed based on the regional dendrogram. This phenomenon can roughly be explained by the presence of intercontinental links in the ATP



**Fig. 6 | The ATP tennis network embedded into the two-dimensional hyperbolic space using two different strategies. a** The hyperbolic layout obtained by running the Cluster Level Optimised Vertex Embedding (CLOVE) method in its default settings alongside the associated metric scores displayed in a radar chart at the top-right corner. We show the results for the mapping accuracy, MA, the edge prediction precision, EPP, the area under the receiver operating characteristic curve, AUC, the greedy routing score, GR, the greedy success rate, GS, and the greedy routing efficiency, GE. **b** Embedding the tennis network by relying on a regional dendrogram

comprising ground-truth information about the ethnicities of the players. In a similar fashion to (a), the same metric scores are presented again in a radar chart at the top-right corner. In both panels, the network nodes are coloured based on the continent to which the corresponding players belong, with the continents outlined and positioned according to the angular coordinates of their respective players. In (a), the higher metric scores and fewer edge crossings suggest that using CLOVE with the default settings, as shown, generally yields better embedding quality.

network. More specifically, when modules are defined based on regional information, these intercontinental links can become excessively long, as different continents may be positioned far apart on the native disk, eventually leading to a sub-optimal embedding. Contrarily, when modules to be arranged by the algorithm are derived from a community detection method, the majority of links tend to fall within the same angular sector. This spatial concentration of the links results in shorter average link lengths, which in turn enhances the overall quality of the embedding. This explanation is perfectly corroborated by the observation of fewer link crossings in the embedding shown in Fig. 6a.

**The air transportation network.** The OpenFlights database<sup>54</sup> provides detailed information on regular commercial flights between major airports worldwide, containing more than 3000 airports and roughly 67,000 flights, defining a transportation network of crucial importance. Similarly to the ATP tennis network, in our study of this system we applied CLOVE both with default settings (results shown in Fig. 7a) and with a pre-defined dendrogram of geographical regions (results shown in Fig. 7b). The seemingly large similarity between the two layouts in Fig. 7 indicates that our algorithm was able to find a natural arrangement for the nodes, even when it was completely unaware of the ground truth geographical categorisation of the airports and calculated the embedding coordinates solely based on the network structure.

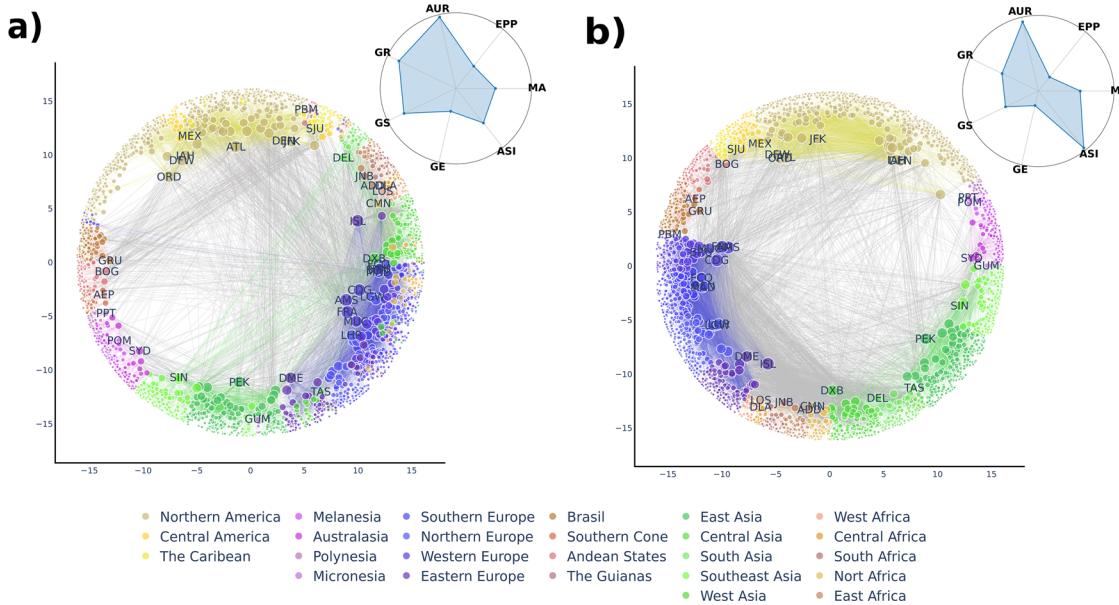
Additionally, in Fig. 8a we plot the embedding distance (measured on the native disk) as a function of the real-world geodesic distance a given flight covers between two airports. The intercontinental flights (Fig. 8b) tend to travel the largest distance in both the real world and in the embedded space. In turn, the flights within a given continent (Fig. 8c-h) are usually shorter, again according to both distance measures. This shows that despite the difference in the curvature of the underlying geometry and the fact that the embedding is completely unaware of the true flight distances (i.e., it is inputted an unweighted network), still our algorithm is finding an arrangement of the airports on the native disk which is coherent with the real world geographical positioning of the airports. This is also supported by a Pearson correlation coefficient of 0.40 between the embedding distance and the geodesic distance.

In summary, as demonstrated by the examples of ATP and Openflights networks, CLOVE performs notably well, whether using its default settings (see Figs. 6a, 7a) or a pre-defined dendrogram of geographical regions (see Figs. 6b, 7b). However, the former strategy is generally better than the latter, as evidenced by the reduced number of long-range interconnections in Figs. 6a, 7a corresponding to the default versions of CLOVE. This superiority is further reflected by the fact that running the method with its default settings almost always yields higher metric scores (shown in the upper right corner of panels Figs. 6a, b and 7a, b). Nonetheless, it is important to note an exception, specifically the ASI score, which measures the angular coherence of communities. In general, a high ASI value indicates well-separated communities in terms of angular coordinates, thus reaching its maximal value when the arrangement is explicitly constructed based on the ground-truth dendrogram of communities. This observation is supported by the radar charts illustrated in Figs. 6b, 7b. For a more detailed description of ASI and the other metric scores employed in our analysis, please refer to the Methods section.

### Customizing the embedding framework

Our framework, built on recursive hierarchical partitioning of the network, opens up multiple possibilities for customization. As an illustration, we explore one such modification involving the method used for arranging the extracted modules. Although the TSP approach, corresponding to the default method in CLOVE, has proven to be fast and very efficient, further options may also be considered, especially when aiming for a further increase in the speed of the algorithm.

An alternative arranging scheme we have tested and built into our framework is provided by the minimum curvilinear attachment (MCA) process<sup>22</sup>. In this approach, the ordering of the modules is obtained according to a growing minimum spanning tree based on Prim's algorithm, where we use the same weighting scheme as in the case of the TSP for defining the weighted graph between the extracted communities. (The details of the MCA approach are given in Supplementary Note 1.3.2 in the Supplementary Information). Due to its simpler nature (lower computational complexity), this is expected to give even lower running times compared to the TSP-based embeddings.



**Fig. 7 | Embedding of the air transportation network.** The major geographical regions, such as continents and sub-continents, are colour coded and the size of the nodes indicates the degree. The most important airports are marked by their International Air Transport Association (IATA) code and the radar plots in the insets show the different quality scores of the embedding. **a** The embedding obtained with Cluster Level Optimised Vertex Embedding (CLOVE) at default settings. **b** The embedding with CLOVE using a dendrogram corresponding to the hierarchy of geographic locations. In both panels, the radar charts positioned in

the top-right corners show the qualities of the embeddings using the same metric scores as depicted in Fig. 6. The large similarity between the panels indicates that CLOVE with default settings in (a) found an arrangement very close to the ground truth categorisation of geographical regions solely based on the network structure. This is accompanied by a clear separation of continents in terms of angular coordinates, despite the embedding being completely agnostic to geographical information.

In Table 3 we compare the embedding quality scores and the resource usage of CLOVE with MCA-based arrangement to the default settings relying on TSP. According to the results, the MCA-based angular arrangement is indeed faster, with running times roughly equal to one half of that of the CLOVE default version. Meanwhile, the CLOVE default version is superior according to all the quality scores except for EPP, where CLOVE-MCA is slightly better, and for AUC, where they perform equally well. Nevertheless, the overall performance of CLOVE-MCA falls not far behind that of CLOVE with default settings.

## Discussion

A prevalent and very essential feature of numerous complex systems – observed in either nature or society – lies in the presence of an inherent hierarchical structure that governs the relationships among their constituent components<sup>55–57</sup>. Gaining access to these nested hierarchical structures can be beneficial from various aspects; for instance, it can streamline the design of efficient search protocols among the constituents<sup>58</sup>, facilitate optimal decision-making<sup>55</sup>, and even economize the costs associated with reliable information transfer<sup>59</sup>.

In this study, we utilised these distinctive architectures to effectively address the hyperbolic embedding of complex networks. Specifically, we introduced a method called CLOVE, which accomplishes the mapping of networks into the two-dimensional hyperbolic space through a series of optimization tasks performed hierarchically. When dealing with a given network, the CLOVE method involves two fundamental steps; initially, it begins by reasonably partitioning the network into smaller interconnected entities, followed by determining their optimal arrangement within the hyperbolic disk. While advanced community finding methods, such as the Leiden method, can effectively achieve a sensible partitioning of the network into smaller units, finding the optimal arrangement of these sub-modules on the hyperbolic disk remains a highly challenging task. The CLOVE method brings significant progress in addressing this challenge by leveraging the Travelling Salesman Problem<sup>35–38</sup> – an extensively studied problem in computer science – to optimize the arrangement of communities and their

respective sub-communities in the hyperbolic disk. Although the MCA method employs a somewhat related minimum spanning tree-based approach<sup>22</sup>, to our knowledge, this study is the first to explicitly use the TSP for solving the embedding of complex networks. CLOVE introduces a whole family of embedding techniques, providing a highly efficient alternative framework to well-established methods such as likelihood optimization and spectral-based embeddings.

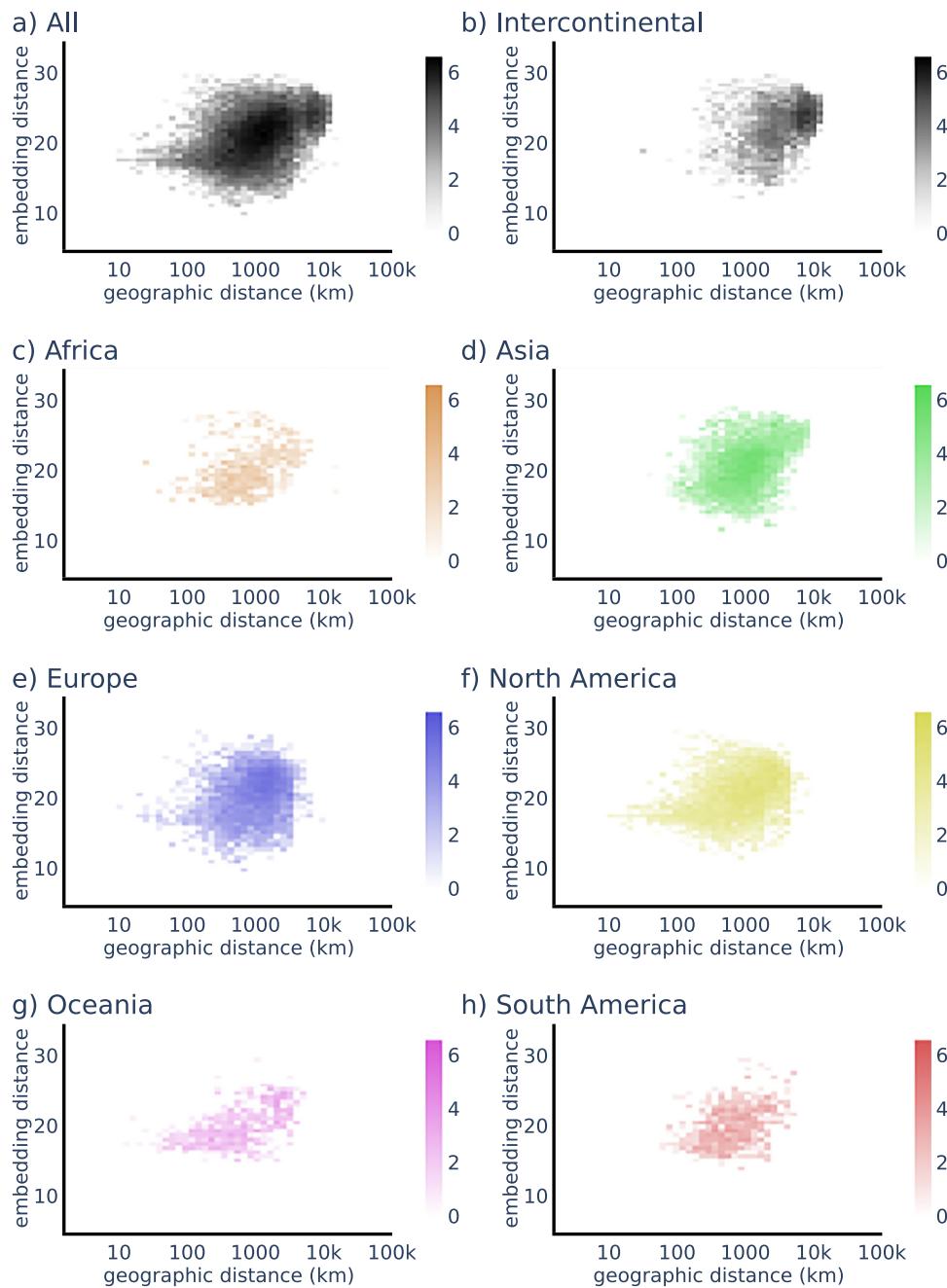
The TSP is undoubtedly one of the best-known combinatorial optimisation problems, with applications ranging from DNA sequencing<sup>60</sup>, aerospace engineering<sup>61</sup>, the analysis of crystals' structures<sup>62</sup>, to the planning of telescope movement in astronomy<sup>63,64</sup>. Additionally, it has proven to be highly effective in defining and measuring the geometric separability (both linear and non-linear) of mesoscale patterns in multidimensional datasets<sup>65</sup>. In this paper, we introduced a further application in complex network theory, facilitating the rapid optimization of node arrangements in the two-dimensional hyperbolic space.

While the chosen heuristic approximation method for solving the TSP has a complexity of  $\mathcal{O}(C^3)$  concerning the number of “cities”  $C$ , it is applied only to modules co-occurring at the same level within individual branches of the module hierarchy, rather than to all nodes simultaneously. Consequently, assuming that the community dendrogram is given by a  $q$ -ary tree, where each level  $l = 1, \dots, \log_q(N) - 1$  comprises  $q^l$  number of communities with sizes  $N/q^l$  and  $q(N) \sim N^c$  for some  $0 < c < 1$ , the overall complexity becomes bounded above by  $\mathcal{O}(N^{2c+1})$  (for further details see the Methods section). This favourable scaling enables the embedding of networks with millions of nodes in under 50 hours. Although this falls behind the running time of very fast methods like HMCS<sup>34</sup>, in our opinion, CLOVE provides a favourable balance between speed and accuracy. On average, CLOVE outperformed HMCS according to all studied quality indicators and its embedding quality is comparable, and in many cases, even superior to state-of-the-art methods, such as Mercator<sup>19</sup>.

To make our embedding framework more general, besides the TSP, we have also built in the possibility for using the MCA algorithm<sup>22</sup> when

**Fig. 8 | Embedding distance and geodesic distance in the air transportation network.**

We plot the distance measured on the hyperbolic disk (the embedding distance) for connected airport pairs as a function of the geodesic distance on the globe, measured in kilometres. The panels depict heat maps corresponding to different large geographical regions. Panel (a) shows the results for all pairs, while the remaining panels (b–h) show results for specific continental pairs: (b) intercontinental, (c) Africa, (d) Asia, (e) Europe, (f) North America, (g) Oceania, and (h) South America. The fact that intercontinental connections tend to be longer than continental ones also in the embedding space reinforces that the embedding obtained solely based on the network structure captures essential features of the original system.

**Table 3 | Average quality scores when CLOVE is arranging the communities according to the MCA algorithm**

	MA	EPP	AUC	GR	GS	GE	Running Time (min.)	Peak Mem. (GB)
CLOVE-MCA (Symmetric)	0.361	<b>0.488</b>	<b>0.962</b>	0.479	0.410	0.177	<b>0.212</b>	0.476
CLOVE-MCA (Asymmetric)	0.360	0.478	<b>0.962</b>	0.453	0.390	0.170	<b>0.212</b>	0.478
CLOVE(default)	<b>0.365</b>	0.485	<b>0.962</b>	<b>0.505</b>	<b>0.426</b>	<b>0.180</b>	0.496	0.426

We show the results for the same quality scores as in Tables 1, 2 for the same networks as in Table 1. For each indicator, the best performing value is shown in bold. First row corresponds to the symmetric version of MCA, whereas the second row corresponds to the asymmetric version (the details of MCA are given in Supplementary Note 1.3.2 in the Supplementary Information). For comparison, in the third row we show the results for CLOVE with default settings.

arranging the hierarchical communities. This CLOVE-MCA version is preferred in applications where extremely low running time is crucial, whereas otherwise the usage of the TSP-based default CLOVE embedding seems more advantageous. While CLOVE provides a highly flexible framework with multiple points of customization throughout the algorithm,

one minor limitation is its ability to embed networks exclusively into the two-dimensional hyperbolic space. Recent progress in complex network theory has both generalised the fundamental hyperbolic network models to higher dimensions<sup>66–68</sup> and has also introduced higher-dimensional hyperbolic embeddings<sup>17,20</sup>. Nonetheless, extending CLOVE to higher

dimensions poses a non-trivial task, offering an intriguing challenge for future research, although it falls beyond the scope of the present paper.

In conclusion, owing to the scalable nature of CLOVE, it becomes feasible to map even very large networks into hyperbolic space within a reasonable amount of time, moreover, with a high level of reliability. This remarkable efficiency of the CLOVE method undoubtedly represents a significant step towards the creation of hyperbolic maps for a wide range of real-world complex networks.

## Methods

### Networks in the native disk representation of the hyperbolic space

A common approach to the study of hyperbolic network geometry is the use of the native representation of the two-dimensional hyperbolic space<sup>11</sup>, where the hyperbolic plane of constant curvature  $K < 0$  is represented by a disk of infinite radius in the Euclidean plane. The advantage of this representation compared to the famous Poincaré disk model is that the radial coordinate  $r$  of a point (defined as its Euclidean distance from the disk centre) is equal to its actual hyperbolic distance from the disk centre. In addition, the Euclidean angles between hyperbolic lines are also equal to their hyperbolic counterparts.

The hyperbolic distance between two points can be measured along the connecting geodesic, which is either a hyperbola, or – if the disk centre falls on the Euclidean line connecting the two points – the corresponding diameter of the disk. The hyperbolic distance  $x$  between two points at polar coordinates  $(r, \theta)$  and  $(r', \theta')$  can be calculated from the hyperbolic law of cosines written as

$$\cosh(\zeta x) = \cosh(\zeta r) \cosh(\zeta r') - \sinh(\zeta r) \sinh(\zeta r') \cos(\Delta\theta), \quad (1)$$

where  $\zeta = \sqrt{-K}$  and  $\Delta\theta = \pi - |\pi - |\theta - \theta'||$  is the angle between the examined points. For sufficiently large  $\zeta r$  and  $\zeta r'$ , and when  $2 \cdot \sqrt{e^{-2\zeta r} + e^{-2\zeta r'}} < \Delta\theta$ , the hyperbolic distance can be approximated as<sup>11</sup>

$$x \approx r + r' + \frac{2}{\zeta} \cdot \ln\left(\frac{\Delta\theta}{2}\right). \quad (2)$$

When generating random graphs via geometric network models operating in the native disk, or embedding networks into the native disk, there seems to be an intimate relation between the node degree and the radial position. Hyperbolic network models are centred around the idea of placing nodes on the native disk (in a uniform or close to uniform fashion) and drawing links with a probability depending on the metric distance. In general, such models can be regarded as a particular case of a broader hidden variable framework<sup>69–73</sup>, where the hidden variables of the nodes are associated with the coordinates of the nodes in the hyperbolic space, whereas the connection probability between pair of nodes depends specifically on their respective distances.

One of the best-known hyperbolic network models is given by the Popularity-Similarity Optimisation (PSO) model<sup>39</sup>. In case of the PSO model (where new nodes are added to the network one by one with logarithmically increasing radial coordinate and random angular coordinate), a rather intuitive analogy was drawn between the coordinates and plausibly important features of the nodes, such as the popularity and similarity that govern the network growth. In this picture, a small angular distance indicates a high similarity between a node pair, whereas the popularity (the degree) of the nodes is controlled by their radial coordinate, with hubs appearing closer to the disk centre and low degree nodes occupying the disk periphery.

More specifically, in the PSO-model the expected degree of node  $i$  at time point  $t$  in the network generation is  $k_i(t) \sim \exp(r_{it} - r_{tt})$  where  $r_{tt} = \frac{2}{\zeta} \ln t$  is the radial coordinate of the newly appearing node at  $t$  (with  $\zeta = \sqrt{-K}$  originating from the hyperbolic curvature  $K$ , usually assumed to be  $\zeta = 1$ ) and  $r_{it}$  is the actual radial coordinate of node  $i$  that was shifted from its original  $r_{ii}$  value as  $r_{it} = \beta r_{ii} + (1 - \beta)r_{tt}$ , where  $\beta \in (0, 1]$  corresponds to the

popularity fading parameter<sup>39</sup>. Related to this, when assuming that a network was generated by the PSO-model, the maximum likelihood estimate for the radial coordinate be given as<sup>13</sup>

$$r_{ii}^* = \frac{2}{\zeta} \ln i^*, \quad (3a)$$

$$r_{iN}^* = \beta r_{ii}^* + (1 - \beta)r_{NN}^*, \quad (3b)$$

where the optimal ordering of the nodes given by  $i^*$  is following the node degrees, with the largest degree node in the network obtaining  $i^* = 1$ , second largest degree node receiving  $i^* = 2$ , etc., and equation (3a) corresponds to the initial radial coordinate of node  $i^*$ , whereas equation (3b) takes into account also the outward drift due to the popularity fading. (A more detailed description of network generation in the PSO model is provided in Supplementary Note 4.1.1 in the Supplementary Information).

A similarly close relation occurs between the node degree and the radial coordinate in the random hyperbolic graph (RHG) model<sup>11</sup>, also known as the  $\mathbb{S}^1/\mathbb{H}^2$  model<sup>19</sup>. In this static approach, nodes are given uniform random angular coordinates and a hidden degree variable  $\kappa$  sampled from a power-law distribution. Node pairs are connected according to a probability that is decreasing as a function of the angular distance but also takes into account the product of the hidden variables, resulting in a scale-free network where the degree decay exponent is the same as for the hidden variable distribution and the expected degree of node  $i$  is given by  $\kappa_i$ . When mapping the network onto the native disk, the radial coordinate is defined as  $r_i = R_0 - 2 \ln(\kappa_i)$  where  $R_0$  is a constant depending on the model parameters. Hence, similarly to the PSO-model, the hubs are placed close to the disk centre, the low-degree nodes are located towards the periphery and there is an overall logarithmic dependence between the degree and the radial coordinate.

Numerous hyperbolic embedding methods take advantage of the above intrinsic connection between the radial coordinate and the node degree. For example, Hypermap<sup>13</sup>, one of the first hyperbolic embedding methods, is based on likelihood maximisation concerning a generalised version of the PSO model, where the optimisation shuffles only the angular coordinates with the radial coordinates being assigned according to the degree. Another well-known hyperbolic embedding approach is provided by the family of coalescent embeddings<sup>18</sup>, where the angular coordinates are inferred using dimension reduction techniques on weighted matrices representing the distance relations between the nodes, however, the radial coordinates are again distributed according to the PSO model, based on the degree. This choice for setting the radial coordinates was left unchanged when the coalescent embedding approach was combined with local angular optimisation of the node positions<sup>21</sup>. The radial arrangement of the nodes is according to the PSO model, also in the case of Laplacian Eigenmaps<sup>14</sup>, where the angular coordinates are obtained from the non-linear dimension reduction of Laplacian matrices. The RHG model can also be used for inferring the radial coordinates based on the node degree, as was shown in the case of the Mercator embedding method<sup>19,20</sup>. Nevertheless, the radial coordinates assigned based on the PSO model or based on the RHG model are very similar, since both depend logarithmically on the node degree. The only major difference between these two options is that all nodes obtain a unique radial coordinate according to the PSO model, whereas it is allowed for multiple nodes to have the same radial coordinate in the RHG model.

### Detailed description of the CLOVE method

Let us consider the task of embedding an arbitrary undirected (and not necessarily connected) network consisting of  $N$  number of nodes and  $E$  number of edges into the two-dimensional native disk representation of the hyperbolic space. We employ a hierarchical multi-level arrangement of the communities within the native disk by leveraging information about the connectedness of these communities and their respective sub-communities across different scales of the network. We denote the communities at a given

hierarchy level  $l$  by  $t_m^{(l)}$ , where the lower index  $m$  runs from 0 to the total number of communities at the given level.

- **Arranging the communities at the topmost  $l = 0$  level**

(a) *Detecting communities:* We can identify the top-level communities  $t_m^{(0)}$  by using any arbitrary non-overlapping community finding algorithm. Here, the Leiden method<sup>40</sup> is adopted as the default approach for community detection, which is an advanced technique based on modularity maximisation. Nevertheless, other built-in options, such as the Louvain method, are also available in the provided code. (A brief description of both the Leiden and the Louvain approaches, as well as the concept of modularity is provided in Supplementary Note 1.1 in the Supplementary Information). If an entire hierarchical dendrogram of the communities is accessible, e.g., as might be the case for the Louvain algorithm<sup>74</sup>, in this step we use the partition at the topmost level ( $l = 0$ ) of the dendrogram.

(b) *Defining a weighted network between the communities:* We construct the proximity graph of the communities, i.e., build up a complete weighted super-graph, whose nodes correspond to the communities  $t_m^{(0)}$  found earlier in step 1a). The edge weight between any pair of super-nodes  $i$  and  $j$  is defined as

$$W_{ij} = f\left(\frac{2E_l C_{ij}}{K_i K_j}\right) + 1, \quad (4)$$

where  $E_l = E_0$  is the number of edges,  $K_i$  and  $K_j$  denotes the number of intra-community links within the communities  $t_i^{(0)}$  and  $t_j^{(0)}$ , respectively, and  $C_{ij}$  stands for the number of inter-connections between  $t_i^{(0)}$  and  $t_j^{(0)}$ . Note that although the function  $f$  defined in Eq. (4) can be any arbitrary decreasing function of its argument, taking values on the unit interval, we use an exponentially decaying form  $f(x) = e^{-x}$  by default. In Supplementary Note 1.2 in the Supplementary Information, we demonstrate that adopting this choice for the weights between modules guarantees compliance with the triangle inequality, thereby justifying the utilization of TSP in the later steps.

(c) *Approximate solution for the TSP:* We look for the minimal-weight Hamiltonian cycle of the super-nodes (communities) in the proximity graph defined in 1b). This corresponds to solving the TSP on the proximity graph, and the obtained solution represents the inferred angular order of communities. We use the Christofides method supplemented with a threshold accepting boost<sup>43</sup> for solving the TSP by default, however, further possible choices are also available in the provided code, including e.g., the greedy method, simulated annealing, or the threshold accepting method solely. Note that the latter two metaheuristic algorithms can also be applied in combination with the greedy or Christofides method, providing therefore, an option of boosting that may enhance the quality of the final embedding in particular cases. (We give a summary of the implemented TSP solvers in Supplementary Note 3.1 in the Supplementary Information).

(d) *Alternatively, applying MCA for angular ordering:* If preferring a low running time over high quality, one may opt for using the MCA instead of the TSP in the calculation of the angular order between the modules. Here, the communities (nodes in the proximity graph) are inserted one by one into a growing minimum spanning tree following Prim's algorithm, where we use the weights given by (4). The MCA has both a symmetric and an asymmetric version, for which the details are described in Supplementary Note 3.2 in the Supplementary Information.

(e) *Circular arrangement of the communities:* We arrange the communities on the native disk such that subsequent communities become adjacent on the disk. Each community is allocated a circular sector, the size of which is proportional to the number of nodes within that

community. More precisely, the community  $t_i^{(0)}$  in the minimal-weight order is assigned to the angular interval

$$\left[\Phi_{i,\text{start}}^{(0)}, \Phi_{i,\text{end}}^{(0)}\right) = \left[\frac{2\pi}{N} \sum_{j=1}^{i-1} n_j^{(0)}, \frac{2\pi}{N} \sum_{j=1}^i n_j^{(0)}\right) \quad (5)$$

where  $n_m^{(0)}$  denotes the number of nodes in community  $t_m^{(0)}$ .

- **Arranging the communities at level  $l + 1 > 0$**  For convenience, the current level is considered to be level  $l + 1$ , whereas the previous level (immediately above the hierarchy) is assumed to be level  $l$ .

a. *Detecting sub-communities:* For each community at the previous level,  $l$ , we run the same community finding algorithm as in 1a) on the sub-graph spanning between the community members (detached from the rest of the network). Let us focus on the sub-modules found this way within community  $t_i^{(l)}$  from the previous level, and let us denote these sub-modules as  $t_{i1}^{(l+1)}, t_{i2}^{(l+1)}, \dots, t_{ik}^{(l+1)}$  for convenience.

b. *Defining a weighted network between the sub-communities:* For each group of sub-modules found within a specific larger community from the previous level, we define a separate weighted network, similarly to step 1b). However, an important difference is that this time we also include two extra nodes in this complete graph, corresponding to the neighbouring communities from the previous level. These serve as “anchors” for a more optimal arrangement of the sub-modules. Specifically, for the sub-modules  $t_{i1}^{(l+1)}, t_{i2}^{(l+1)}, \dots, t_{ik}^{(l+1)}$  listed in 2a), we include the left and right neighbouring communities of  $t_i^{(l)}$  according to the angular arrangement of the communities in level  $l$ . The link weights are defined again by using (4).

c. *Approximate solution for the TSP:* For each separate weighted complete graph defined in 2b), we solve the TSP using the same heuristic as in 1c), receiving a Hamiltonian cycle over the sub-modules and the two extra neighbouring communities from the previous level.

d. *Alternatively, applying MCA for angular ordering:* If have chosen to use MCA instead of TSP in the angular arrangement, then for each separate weighted complete graph defined in 2b) we apply the MCA similarly to as in 1d), receiving a Hamiltonian cycle over the sub-modules and the two extra neighbouring communities from the previous level.

e. *Arrangement of the sub-communities:* Naturally, the sub-modules located within  $t_i^{(l)}$  must be placed inside the angular range  $\left[\Phi_{i,\text{start}}^{(l)}, \Phi_{i,\text{end}}^{(l)}\right)$  associated with  $t_i^{(l)}$ . Any sub-module  $t_{ik}^{(l+1)}$  receives a circular sector having a central angle of  $\frac{2\pi}{N} n_{ik}^{(l+1)}$  (with  $n_{ik}^{(l+1)}$  denoting the number of nodes in  $t_{ik}^{(l+1)}$ ), and the order of the sub-modules is determined by the Hamiltonian cycle received in 2c). Under optimal circumstances, the “anchoring” super-nodes (communities from level  $l$ ) are neighbours in the Hamiltonian cycle, and we can apply a cyclic permutation bringing the “anchor” placed aside  $t_i^{(l)}$  at  $\Phi_{i,\text{start}}^{(l)}$  to the beginning of the cycle and the “anchor” placed aside  $t_i^{(l)}$  at  $\Phi_{i,\text{end}}^{(l)}$  to the end of the cycle. Based on the cycle obtained, now aligned with the “anchor” positions, the angular range of  $t_{ik}^{(l+1)}$  can be given as

$$\left[\Phi_{ik,\text{start}}^{(l+1)}, \Phi_{ik,\text{end}}^{(l+1)}\right) = \left[\Phi_{i,\text{start}}^{(l)} + \frac{2\pi}{N} \sum_{j=1}^{k-1} n_j^{(l+1)}, \Phi_{i,\text{start}}^{(l)} + \frac{2\pi}{N} \sum_{j=1}^k n_j^{(l+1)}\right) \quad (6)$$

In cases where the “anchors” are not adjacent in the Hamiltonian cycle obtained in step 2c), we first seek a cyclic permutation where the left “anchor” is positioned correctly, specifically as the starting point (leftmost element) of the Hamiltonian cycle. Beginning from this left anchor node, we proceed to the right, preserving the sequence obtained in step 2c) until we encounter the right “anchor”. To ensure

that this right “anchor” becomes the rightmost element in the final order, we perform a reflection transformation (chirality change) on the remaining segment of the cycle, starting from the right “anchor” node. This modified segment is then concatenated with the preceding unchanged segment. By constructing the final order in this way, we ensure that both anchors are correctly positioned and the longest directionally consistent sub-sequences of the Hamiltonian cycle are maintained, preserving the structural integrity of the original sequence as much as possible.

- **Iteration and stopping criterion for the angular arrangement of the communities** After the completion of the angular arrangement of the communities at any level  $l$ , we proceed to the next level as described in 2. However, if for any communities  $t_i^{(l)}$  the community finding algorithm returns no sub-modules in 2a), meaning that  $t_i^{(l)}$  is already so small and compact that it is not worth dividing into sub-communities, we do not carry out steps 2b-d, and leave  $t_i^{(l)}$  as it is. Although  $t_i^{(l)}$  can still act as an anchor for the sub-modules of neighbouring communities, the angular arrangement procedure is locally stopped at  $t_i^{(l)}$ . Naturally, for other communities at the same level, the algorithm will carry on and may discover contained sub-modules, where we position these according to steps 2.

When the recursive discovery of contained sub-communities is stopped locally everywhere, we have reached the stage where it is not worth dividing further any of the modules at the lowest level in any branch of the community hierarchy. (Naturally, the maximal depth of the branches can vary.) In order to fully specify the angular coordinates of the individual nodes, we can now move on to the next phase in the algorithm, described in step 4.

- **Angular arrangement of individual nodes within communities** There are several options for arranging the members of a given community (assumed to be on the possible lowest level in the corresponding branch of the community hierarchy). In all cases, the node positions are distributed in a uniform regular fashion inside the considered sub-module, where the angular distance between neighbouring nodes is always  $\frac{2\pi}{N}$ .

- Probably the most natural choice is to apply the same principles as in the case of the sub-modules, outlined in step 2. Here we basically replace the sub-modules  $t_{i1}^{(l+1)}, t_{i2}^{(l+1)}, \dots, t_{ik}^{(l+1)}$  by the individual community members, but otherwise carry out exactly the same steps from 2b to 2e. Although this is likely to provide the best quality local arrangement among the other options, it is also computationally the most demanding.
- Another very simple choice is to distribute the members randomly among the available angular positions. This is the fastest option, albeit also with the lowest quality.
- A further heuristic solution we propose is based on the node degrees. If the number of community members is odd, the member with the largest degree will occupy the central position and the node with second largest degree will be its left or right neighbour (chosen at random). If the number of members is even, the first two nodes according to the degrees will occupy the two central positions (again, in random order). The further nodes are added in the order of their degree, always occupying a position next to the already occupied positions either from the left or from the right. We decide about inserting to the left or to the right based on the number of connections between the given node and the already inserted nodes on the right or on the left. (In the case we observe an equal number of connections to the right and to the left, we choose randomly). This method yields usually better quality arrangements compared to random positions and it is faster compared to option a).

By default we use option c), however, the code we provide allows both a) and b) as well.

- **Radial arrangement of the nodes** The radial coordinates are defined solely based on the node degree, independently of the angular

coordinates. For simplicity, we use the radial coordinates predicted based on the PSO model and apply Eqs. ((3b)-(3b)) for assigning  $r_i$ , where the node indices are distributed according to the order dictated by the node degrees, as explained in Section Networks in the native disk representation of the hyperbolic space. The parameter  $\beta$  necessary for calculating the coordinates is obtained by fitting the tail of the degree distribution of the embedded network with a power-law decaying function and applying the well-known relation  $\beta = \frac{1}{\gamma-1}$  between the degree decay exponent  $\gamma$  and the popularity fading parameter.

#### Additional parameters of the CLOVE method.

- **Number of “anchor” nodes** Originally, CLOVE uses  $z = 2$  number of “anchor” nodes in steps 2b-d) by default. However, the implementation we provide allows to handle neighbors of higher orders as well. In such cases, for each sub-community, we include  $z = 2l$ ,  $l \in \mathbb{N}^+$ ,  $l > 1$  number of neighbouring communities from the preceding level, hence exploiting a more global information about the connectedness of the communities in the arrangement step.

#### • **Decomposition of isolated nodes and components**

**Embedding networks with multiple components** Despite the difficulty that most embedding algorithms have in dealing with networks comprising multiple connected components, the CLOVE algorithm can handle this type of networks in a natural manner. If the network we need to embed is not fully interconnected, the default approach for CLOVE is to start by optimizing the position of the different components on the hyperbolic disk instead of the top-level communities. Subsequently, the algorithm proceed conventionally by detecting sub-communities inside these distinct components using a predefined community finding method. Notably, the default application of the Leiden algorithm ensures the preserved connectivity of these identified sub-communities<sup>40</sup>. This embedding option of the algorithm is referred to as the decomposition of connected components, which is governed by a Boolean variable in the provided code. Conversely, if the decomposition of connected components is disabled, the algorithm can still effectively manage multiple components. In such cases, instead of seeking the optimal arrangement of the separate components at the highest level, the algorithm directly arranges the communities themselves consistently across all scales.

**Decomposition of nodes with degree  $k = 0$**  If the network contains isolated nodes, CLOVE can embed these isolated nodes separately by detaching them from the rest of the network. When this feature is enabled, random angular coordinates are allocated to the isolated nodes, while the remaining portion of the network is embedded using the standard procedure outlined in steps 1-5 above. The assignment of the radial coordinates are not affected. It is important to note that this option is primarily designed to improve runtime efficiency; nevertheless, it may also result in enhanced accuracy in particular cases. We refer to this feature of the algorithm as the “decomposition of  $k$  nodes” controlled by a boolean variable in the provided code.

**Decomposition of nodes with degree  $k = 1$**  Similarly to the decomposition of isolated nodes, CLOVE allows the decomposition of nodes with degree  $k = 1$  as well. Upon enabling this feature, controlled again by a boolean variable, the algorithm starts by detaching nodes with only one degree from the rest of the network. First, the remaining part of the network is embedded, then detached nodes with only one degree receive the same angular coordinates as their single neighbor. In case two nodes are only connected to each other, hence having been detached during the decomposition procedure, they both receive the same uniformly sampled random angular coordinate. The assignment of the radial coordinates are not affected here either.

- **The sizes of angular sectors corresponding to the communities** During the arrangement of the communities in steps 1d) and 2d), the CLOVE method allocates an angular sector to each community with the central angle being proportional to the number of nodes it contains,

as demonstrated by Eq.(5) and Eq.(6). However, in the code we provide, there is also an option to allocate circular sectors to each community in such a way that their central angle is proportional to the sum of node degrees within the community. This method, also utilized by the HMCS method<sup>34</sup>, enhances the flexibility of the algorithm.

### Time complexity analysis of CLOVE

The overall running time of CLOVE is primarily determined by two key factors: first, the complexity of the community detection algorithm used, and second, the complexity of the method that arranges the obtained communities on the hyperbolic disk. Since both processes are applied iteratively at increasingly finer scales and more frequently, the computational cost depends heavily on the structure of the resulting dendrogram, making it difficult to precisely estimate CLOVE's time complexity.

In order to provide a somewhat simplified, yet reasonable estimation, let us assume that the community detection method operates in  $\mathcal{O}(n^a)$  time, where  $n$  represents the size of the (sub-)network, and the arrangement of communities has a complexity of  $\mathcal{O}(C^b)$ , where  $C$  is the number of communities to be arranged on the hyperbolic disk. Since no community detection algorithm is faster than purely linear, and most TSP solvers have at least quadratic complexity in  $C$ <sup>41–43,75,76</sup>, the reasonable range for the parameters  $a$  and  $b$  is  $a \geq 1$  and  $b \geq 2$ . Additionally, for simplicity, we neglect the logarithmic factors that may appear in either the complexity of community detection or community arrangement. Taking all of these factors into account, we begin by presenting the complexity in the worst-case scenarios in the following subsection, and then proceed to discuss more realistic estimations.

**Complexity in the worst-case scenarios.** Let us suppose that the width of the dendrogram produced by the algorithm is constant, meaning that each level  $l = 1, \dots, N - 1$  consists of exactly two communities with sizes  $N - l$  and 1, where  $N$  denotes the size of the input network. In this case, community detection is performed on a network of size  $N - l$  at each level  $l$ , while community arrangement is carried out for 2 different communities at each level. As a consequence, the complexity of CLOVE  $\mathcal{C}_t$  can be estimated by

$$\begin{aligned} \mathcal{C}_t(N) &= \mathcal{O}\left(\sum_{l=0}^{N-1} ((N-l)^a + 2^b)\right) = \mathcal{O}\left(\sum_{l=1}^N (l^a + 2^b)\right) \\ &\approx \mathcal{O}\left(\frac{N^{a+1}}{a+1} + 2^b N\right) \approx \mathcal{O}(N^{a+1}), \end{aligned} \quad (7)$$

which indicates that the overall runtime is primarily determined by the runtime of the community detection algorithm. This happens when the dendrogram is extremely deep, but has minimal width. On the other hand, the opposite scenario—where the width is maximized and the depth is minimized—also leads to a highly suboptimal case. In this situation, there is only a single level containing  $N$  distinct communities, each of which has size 1, resulting in a complexity of  $\mathcal{C}_t = \mathcal{O}(N^b)$ . Here, the dominant factor in the overall complexity comes from the method used to arrange the communities on the hyperbolic disk. Combining these observations, the estimated worst-case complexity of CLOVE can concisely be expressed as

$$\mathcal{C}_t(N) = \mathcal{O}(N^{\max(a+1,b)}). \quad (8)$$

Note, however, that the estimation in Eq. (8) is highly unrealistic, as CLOVE typically produces dendograms whose width and depth scale with the size of the network. To provide a more reasonable estimation, in the following subsection, we assume that the dendrogram generally takes the form of a tree with a branching factor of  $q$ . In other words, each community at level  $l$  decomposes into  $q$  smaller sub-communities at level  $l + 1$ , where  $q$  may vary with the network size, i.e.,  $q = q(N)$ .

**Complexity in case of  $q$ -ary tree dendrogram.** If the dendrogram is a  $q$ -ary tree each level  $l = 1, \dots, \log_q(N) - 1$  comprises  $q^l$  number of communities with sizes  $N/q^l$ . To resolve the sub-communities at the subsequent level  $l + 1$ , the community detection method used in CLOVE must be applied to the graphs induced by these communities. Each of these operations has a time complexity of  $\mathcal{O}((N/q^l)^a)$  alone, and there are altogether  $q^l$  number of such operations, which gives an overall  $\mathcal{O}(q^l(N/q^l)^a)$  complexity for extracting the communities at level  $l + 1$ . Since each of the  $q^l$  communities at level  $l$  breaks down into  $q$  smaller communities at level  $l + 1$ , and each of these smaller groups requires local sorting with a complexity of  $\mathcal{O}(q^b)$ , the overall complexity for arranging all the communities at level  $l + 1$  is  $\mathcal{O}(q^{l+b})$ . Taking all these into account and summing up the contributions at each level  $l = 1, \dots, \log_q(N) - 1$ , the complexity of CLOVE in case of a  $q$ -ary tree dendrogram can roughly be estimated by

$$\begin{aligned} \mathcal{C}_t(N) &= \mathcal{O}\left(\sum_{l=0}^{L-1} q^l (N/q^l)^a + q^{l+b}\right) \\ &= \mathcal{O}\left(N^a \sum_{l=0}^{\log_q(N)-1} (q^{1-a})^l + q^b \sum_{l=0}^{\log_q(N)-1} q^l\right) \end{aligned} \quad (9)$$

$$= \mathcal{O}\left(N^a \frac{(q^{1-a})^{\log_q(N)} - 1}{q^{1-a} - 1} + q^b \frac{q^{\log_q(N)} - 1}{q - 1}\right) \quad (10)$$

$$= \mathcal{O}\left(N^a \frac{N^{1-a} - 1}{q^{1-a} - 1} + q^b \frac{N - 1}{q - 1}\right). \quad (11)$$

Since  $q > 1$ , in the limit of large networks, i.e., as  $N \rightarrow \infty$ , Eq. (11) can be safely approximated by

$$\mathcal{C}_t(N) \approx \begin{cases} \mathcal{O}\left(\frac{N^a}{1-q^{1-a}} + \frac{Nq^b}{q-1}\right), & \text{if } a \neq 1, \\ \mathcal{O}\left(N \frac{\ln N}{\ln q} + \frac{Nq^b}{q-1}\right), & \text{if } a = 1. \end{cases} \quad (12)$$

Under the additional assumption that  $q(N) \sim N^c$  for some  $0 < c < 1$ , this expression simplifies further to

$$\begin{aligned} \mathcal{C}_t(N) &\approx \begin{cases} \mathcal{O}(N^a + N^{cb-c+1}), & \text{if } a \neq 1, \\ \mathcal{O}(N + N^{cb-c+1}), & \text{if } a = 1, \end{cases} \\ &= \mathcal{O}(N^{\max(a,cb-c+1)}). \end{aligned} \quad (13)$$

The default implementation of CLOVE employs Leiden<sup>40</sup> for community detection, which corresponds to  $a = 1$  in the case of sparse networks (disregarding logarithmic factors), and utilizes the Christofides algorithm with an additional boost as a TSP solver<sup>41</sup>, implying  $b \approx 3$ . By substituting these hyperparameter values back into Eq. (13), the computational complexity of the default CLOVE method further simplifies to

$$\mathcal{C}_t(N) \approx \mathcal{O}(N^{2c+1}), \quad (14)$$

which matches reasonably well with the results of numerical simulations in Fig. 5. Although Eq. (13) offers a more realistic estimate than Eq. (8), it still slightly overestimates the overall complexity, as the dendograms used in CLOVE are typically not perfect  $q$ -ary trees expanding uniformly with each level, but rather irregular, narrowing graph structures that decrease complexity compared to ideal trees.

### Embedding quality metrics

Broadly speaking, embedding quality metrics are scalar values ranging from 0 to 1, used to quantify how well an embedding of a given network fits into

the two-dimensional hyperbolic space. In order to reasonably assess the quality of our resulting embeddings and to make meaningful comparisons with other state-of-the-art methods, we systematically tracked various such metrics for each and every studied embedding algorithm. The results of these assessments are presented in the main text of the manuscript, while the subsequent section provides a comprehensive list and detailed explanations for each metric score employed in our analysis.

**The Angular Separation Index (ASI) and the C-score.** The introduction of the ASI was motivated by the fact that in hyperbolic networks, communities tend to occupy well-defined angular regions, having little or no overlap with each other, as observed in several studies in the literature<sup>31,33,34,45</sup>. In order to quantify this tendency, the ASI counts the number of “mistakes” in the angular arrangement, given by the number  $o_i$  of nodes belonging to other communities falling between the boundaries of the given module  $i$ , aggregated over all communities. This is then compared to the maximum of the observed number of mistakes when the angular coordinates are shuffled at random over a number of randomisations, formally expressed as<sup>31</sup>

$$\text{ASI} = 1 - \frac{\sum_{i=1}^{N_{\text{com}}} o_i}{\max_r \left( \sum_{i=1}^{N_{\text{com}}} o_i^{(r)} \right)}, \quad (15)$$

where we used 1000 independent shuffles labeled as  $r = 1, 2, \dots, 1000$  by following<sup>31</sup>. Based on the above, an ASI value near 1 indicates well-separated clusters with minimal angular intermixing, while a value near 0 suggests random angular arrangement of cluster members.

The C-score was introduced for hyperbolic networks with known ground truth coordinates used during network generation as a quantitative comparison between the angular ordering of the nodes in a hyperbolic embedding and in the original layout<sup>18</sup>. Formally it is defined as

$$\text{C-score} = \frac{\sum_{i,j, i \neq j} \delta(i, j)}{N(N-1)/2}, \quad (16)$$

where  $\delta(i, j)$  is 1 if the shortest angular distance between  $i$  and  $j$  has the same direction (i.e., clockwise or anticlockwise) in both layouts, and is 0 otherwise. In order to allow the mirroring of the layout, the C-score is computed also with the inferred angular ordering in the opposite clock direction, and the maximum value between the C-score of the embedding and its mirrored version is chosen.

**Mapping accuracy.** Mapping accuracy (MA) assesses the relationship between geodesic distances and topological shortest paths in an embedded network by determining the Spearman’s rank correlation between the two:

$$\text{MA} = \frac{\text{cov}[R(\text{GD}), R(\text{TP})]}{\theta_{R(\text{GD})} \theta_{R(\text{TD})}}. \quad (17)$$

Here, GD and TD represent the lists of geodesic distances and topological distances for vertex pairs, respectively, while  $R(\text{GD})$  and  $R(\text{TD})$  denote the corresponding ranks of these lists.

**Edge prediction AUROC.** The Edge Prediction AUROC is a measure that evaluates how well an embedding reflects the anticipated pattern of positioning connected vertices closer together than unconnected ones. It involves computing the area under the ROC curve, where predicted scores are determined by the inverse of vertex distances. Positive ground truth classes are represented by existing edges, and negative ground truth classes are represented by non-existing edges. The ROC curve depicts the true positive rate (TPR) against the false positive rate (FPR), showcasing the performance of a binary classifier across different acceptance thresholds. An AUROC score of 0.5 is expected for a random predictor.

**Edge prediction AUPRC.** Edge Prediction AUPRC is an alternative metric that assesses the same behavior as AUROC, but employs a different approach. In this case, the area under the Precision-Recall curve is computed for the same predictions and ground truth occurrences.

**Greedy routing success rate.** The Greedy Routing Success Rate (GR) is an embedding metric that evaluates the efficiency of Greedy Routing paths in reaching their target vertex. This is determined by simply counting the number of successful greedy paths and dividing this sum by the total number of directed vertex pairs. More precisely, the GR score is defined as

$$\text{GR} = \frac{1}{|V|(|V| - 1)/2 - 2|E|} \sum_{\substack{\forall u \in V \\ \forall v \in \bar{N}(u)}} \rho(u, v), \quad (18)$$

where  $\rho(u, v)$  counts the number of successful greedy paths between vertices  $u, v$ ,  $\bar{N}(u)$  is the complement of the neighbourhood of the vertex  $u$ , i.e. the set of vertices  $v \in V$  that are not adjacent to  $u$  (including  $u$  itself). By excluding adjacent vertex pairs in Eq. (18), we eliminate a constant offset from the definition of the GR metric. This is due to the fact that such pairs consistently represent successful greedy paths in embeddings that forbid the assignment of identical coordinates to more than one vertex.

**Greedy routing score.** The Greedy Routing Score (GS) is an extension of the previously discussed Greedy Routing Success Rate, introducing weights to provide a more comprehensive metric for evaluating the embedding quality. In this refined approach, successful paths are assigned weights determined by the ratio of the topological shortest path length between the source and target vertices to the number of visited vertices along the greedy path. This weighting scheme ensures that the contribution to the score is diminished for successful greedy paths that are significantly longer than the topological shortest path. Mathematically, the GS can be expressed as

$$\text{GS} = \frac{1}{|V|(|V| - 1)/2 - 2|E|} \sum_{\substack{\forall u \in V \\ \forall v \in \bar{N}(u)}} \frac{\text{TSPL}(u, v)}{\text{GPL}(u, v)}, \quad (19)$$

where  $\text{TSPL}(u, v)$  is the length of the topological shortest path between vertices  $u$  and  $v$ , and  $\text{GPL}(u, v)$  is the length of the greedy path starting from vertex  $u$  and ending in  $v$ . If a path is unsuccessful,  $\text{GPL}(u, v)$  is set to infinity, thus having zero contribution to the GS in Eq.(19).

**Greedy routing efficiency.** The Greedy Routing Efficiency metric evaluates the relationship between geodesic distances and projected greedy paths, as given by the formula:

$$\text{GE} = \frac{1}{|V|(|V| - 1)/2 - 2|E|} \sum_{\substack{\forall u \in V \\ \forall v \in \bar{N}(u)}} \frac{\text{GD}(u, v)}{\text{PGPL}(u, v)}, \quad (20)$$

where  $\text{GD}(u, v)$  is the geodesic distance between vertices  $u$  and  $v$ , and  $\text{PGPL}(u, v)$  is the projected greedy path length between  $u$  and  $v$ , i.e. the sum of the lengths traveled along the greedy path.

## Data availability

All data used in this work is publicly available from the KONECT project at <http://www.orgnet.com>. The access links the individual network data sets are provided in the Supplementary Information.

## Code availability

The Python implementation of CLOVE will be available at <http://github.com/samu32ELTE/hypCLOVE> upon publication.

Received: 10 October 2024; Accepted: 4 September 2025;

Published online: 10 October 2025

## References

- Albert, R. & Barabási, A.-L. Statistical mechanics of complex networks. *Rev. Mod. Phys.* **74**, 47–97 (2002).
- Mendes, J. F. F. & Dorogovtsev, S. N. *Evolution of Networks: From Biological Nets to the Internet and WWW* (Oxford Univ. Press, Oxford, 2003).
- Newman, M. E. J., Barabási, A.-L. & Watts, D. J. (eds.) *The Structure and Dynamics of Networks* (Princeton University Press, Princeton and Oxford, 2006).
- Holme, P. & Saramäki, J. Temporal networks. *Phys. Rep.* **519**, 97–125 (2012).
- Barrat, A., Barthelemy, M. & Vespignani, A. *Dynamical processes on complex networks* (Cambridge University Press, Cambridge, 2008).
- Goyal, P. & Ferrara, E. Graph embedding techniques, applications, and performance: A survey. *Knowl.-Based Syst.* **151**, 78–94 (2018).
- Zhang, Y.-J., Yang, K.-C. & Radicchi, F. Systematic comparison of graph embedding methods in practical tasks. *Phys. Rev. E* **104**, 044315 (2021).
- Yang, C., Shi, C., Liu, Z., Tu, C. & Sun, M. *Network Embedding*. Synthesis Lectures on Artificial Intelligence and Machine Learning (Springer Cham, 2022).
- Baptista, A., Sánchez-García, R. J., Baudot, A. & Bianconi, G. Zoo guide to network embedding. *J. Phys.: Complex.* **4**, 042001 (2023).
- Boguñá, M. et al. Network geometry. *Nat. Rev. Phys.* **3**, 114–135 (2021).
- Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A. & Boguñá, M. Hyperbolic geometry of complex networks. *Phys. Rev. E* **82**, 036106 (2010).
- Boguñá, M., Papadopoulos, F. & Krioukov, D. Sustaining the internet with hyperbolic mapping. *Nat. Commun.* **1**, 62 (2010).
- Papadopoulos, F., Psomas, C. & Krioukov, D. Network mapping by replaying hyperbolic growth. *IEEE/ACM Trans. Netw.* **23**, 198–211 (2015).
- Alanis-Lobato, G., Mier, P. & Andrade-Navarro, M. Efficient embedding of complex networks to hyperbolic space via their Laplacian. *Sci. Rep.* **6**, 301082 (2016).
- 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).
- Keller-Ressel, M. & Nargang, S. Hydra: a method for strain-minimizing hyperbolic embedding of network- and distance-based data. *J. Complex Netw.* **8**, <https://doi.org/10.1093/comnet/cnaa002> (2020).
- Kovács, B. & Palla, G. Model-independent embedding of directed networks into Euclidean and hyperbolic spaces. *Commun. Phys.* **6**, 28 (2023).
- 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).
- García-Pérez, G., Allard, A., Serrano, M. Á. & Boguñá, M. Mercator: uncovering faithful hyperbolic embeddings of complex networks. *N. J. Phys.* **21**, 123033 (2019).
- 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).
- Kovács, B. & Palla, G. Optimisation of the coalescent hyperbolic embedding of complex networks. *Sci. Rep.* **11**, 8350 (2021).
- Muscoloni, A. & Cannistraci, C. V. Minimum curvilinear automata with similarity attachment for network embedding and link prediction in the hyperbolic space (2018). Preprint at arXiv:1802.01183 [physics.soc-ph].
- Chamberlain, B. P., Clough, J. & Deisenroth, M. P. Neural embeddings of graphs in hyperbolic space (2017). Preprint at arXiv:1705.10359 [stat.ML].
- Chami, I., Ying, Z., Ré, C. & Leskovec, J. Hyperbolic graph convolutional neural networks. *Adv. Neural Info. Processing Syst.* **32**, 4868–4879 (2019).
- Yang, M., Zhou, M., Xiong, H. & King, I. Hyperbolic temporal network embedding. *IEEE Trans. Knowl. Data Eng.* **35**, 11489–11502 (2022).
- Faqeeh, A., Osat, S. & Radicchi, F. Characterizing the analogy between hyperbolic embedding and community structure of complex networks. *Phys. Rev. Lett.* **121**, 098301 (2018).
- Chellig, J., Fountoulakis, N. & Skerman, F. The modularity of random graphs on the hyperbolic plane. *J. Complex Netw.* **10**, (2021). <https://doi.org/10.1093/comnet/cnab051>. Cnab051, <https://academic.oup.com/comnet/article-pdf/10/1/cnab051/41987187/cnab051.pdf>.
- Balogh, S. G., Kovács, B. & Palla, G. Maximally modular structure of growing hyperbolic networks. *Commun. Phys.* **6**, 76 (2023).
- Balogh, S. G. & Palla, G. Intra-community link formation and modularity in ultracold growing hyperbolic networks. *Phys. A: Stat. Mech. its Appl.* **642**, 129784 (2024).
- Chang, C.-S., Huang, C.-C., Chang, C.-T., Lee, D.-S. & Lu, P.-E. Generalized modularity embedding: a general framework for network embedding (2019). ArXiv preprint arXiv:1904.11027.
- Muscoloni, A. & Cannistraci, C. V. Angular separability of data clusters or network communities in geometrical space and its relevance to hyperbolic embedding (2019). Preprint at arXiv:1907.00025 [cs.LG].
- Lizotte, S., Young, J. G. & Allard, A. Symmetry-driven embedding of networks in hyperbolic space. *Commun. Phys.* **8**, 199 (2025).
- Wang, Z., Li, Q., Jin, F., Xiong, W. & Wu, Y. Hyperbolic mapping of complex networks based on community information. *Phys. A: Stat. Mech. its Appl.* **455**, 104 – 119 (2016).
- Wang, Z., Sun, L., Cai, M. & Xie, P. Fast hyperbolic mapping based on the hierarchical community structure in complex networks. *J. Stat. Mech.: Theory Exp.* **2019**, 123401 (2019).
- Robinson, J. J. *On the Hamiltonian Game (A Traveling Salesman Problem)* <https://api.semanticscholar.org/CorpusID:122464330> (1949).
- Verblunsky, S. On the shortest path through a number of points. *Proc. Am. Math. Soc.* **2**, 904–913 (1951).
- Dantzig, G., Fulkerson, R. & Johnson, S. Solution of a large-scale traveling-salesman problem. *J. Oper. Res. Soc. Am.* **2**, 393–410 (1954).
- Held, M. & Karp, R. M. A dynamic programming approach to sequencing problems. *J. Soc. Ind. Appl. Math.* **10**, 196–210 (1962).
- Papadopoulos, F., Kitsak, M., Serrano, M. Á., Boguñá, M. & Krioukov, D. Popularity versus similarity in growing networks. *Nature* **489**, 537 EP – (2012).
- Traag, V. A., Waltman, L. & van Eck, N. J. From Louvain to Leiden: guaranteeing well-connected communities. *Sci. Rep.* **9**, <https://doi.org/10.1038/s41598-019-41695-z> (2019).
- Christofides, N. Worst-case analysis of a new heuristic for the travelling salesman problem. Research Report ADA025602, Carnegie-Mellon University, Management Sciences Research Group (1976).
- Serd'yukov, A. On some extremal walks in graphs. *Upravlyayemye Sist.* **17**, 76–79 (1978).

43. Dueck, G. & Scheuer, T. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *J. Computational Phys.* **90**, 161–175 (1990).

44. Muscoloni, A. & Cannistraci, C. V. A nonuniform popularity-similarity optimization (npso) model to efficiently generate realistic complex networks with communities. *N. J. Phys.* **20**, 052002 (2018).

45. Kovács, B. & Palla, G. The inherent community structure of hyperbolic networks. *Sci. Rep.* **11**, 16050 (2021).

46. Newman, M. E. J. & Girvan, M. Finding and evaluating community structure in networks. *Phys. Rev. E* **69**, 026113 (2004).

47. Fortunato, S. Community detection in graphs. *Phys. Rep.* **486**, 75–174 (2010).

48. Fortunato, S. & Hric, D. Community detection in networks: A user guide. *Phys. Rep.* **659**, 1–44 (2016).

49. Clauset, A., Moore, C. & Newman, M. E. J. Hierarchical structure and the prediction of missing links in networks. *Nature* **453**, 98–101 (2008).

50. Kitsak, M., Voitalov, I. & Krioukov, D. Link prediction with hyperbolic geometry. *Phys. Rev. Res.* **2**, 043113 (2020).

51. Boguñá, M., Krioukov, D. & Claffy, K. Navigability of complex networks. *Nat. Phys.* **5**, 74–80 (2009).

52. Cannistraci, C. V. & Muscoloni, A. Geometrical congruence, greedy navigability and myopic transfer in complex networks and brain connectomes. *Nat. Commun.* **13**, 7308 (2022).

53. Sackmann, J. Tennis atp dat.a. GitHub repository. [https://github.com/JeffSackmann/tennis\\_atp](https://github.com/JeffSackmann/tennis_atp) (2024).

54. Openflights network. <https://networks.skewed.de/net/openflights> Accessed: [insert date accessed].

55. Zafeiris, A. & Vicsek, T. *Why We Live in Hierarchies? A Quantitative Treatise* (Springer, Berlin, 2018).

56. Corominas-Murtra, B., Goñi, J., Solé, R. V. & Rodríguez-Caso, C. On the origins of hierarchy in complex networks. *Proc. Natl Acad. Sci. USA* **110**, 13316–13321 (2013).

57. Balogh, S. G., Zagyva, D., Pollner, P. & Palla, G. Time evolution of the hierarchical networks between pubmed mesh terms. *PLOS ONE* **14**, 1–15 (2019).

58. Uras, T. & Koenig, S. Identifying hierarchies for fast optimal search. *Seventh Annual Symposium on Combinatorial Search - Extended Abstracts* **5** (2014).

59. Guimerá, R., Arenas, A. & Díaz-Guilera, A. Communication and optimal hierarchical networks. *Phys. A* **299**, 247–252 (2001).

60. Nikolakopoulos, A. & Sarimveis, H. A metaheuristic approach for the sequencing by hybridization problem with positive and negative errors. *Eng. Appl. Artif. Intell.* **21**, 247–258 (2008).

61. Plante, R. D., Lowe, T. J. & Chandrasekaran, R. The product matrix traveling salesman problem: An application and solution heuristics. *Oper. Res.* **35**, 772–783 (1987).

62. Bland, R. G. & Shallcross, D. F. Large travelling salesman problems arising from experiments in x-ray crystallography: A preliminary report on computation. *Oper. Res. Lett.* **8**, 125–128 (1989).

63. Cook, W. J. *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation* (Princeton University Press, 2012).

64. Rana, J., Anand, S. & Bose, S. Optimal search strategy for finding transients in large-sky error regions under realistic constraints. *Astrophysical J.* **876**, 104 (2019).

65. Acevedo, A., Wu, Y., Traversa, F. L. & Cannistraci, C. V. Geometric separability of mesoscale patterns in embedding representation and visualization of multidimensional data and complex networks. *PLOS Complex Systems* **1**, e0000012 (2024).

66. Yang, W. & Rideout, D. High dimensional hyperbolic geometry of complex networks. *Mathematics* **8**, 1861 (2020).

67. Budel, G., Kitsak, M., Aldecoa, R., Zuev, K. & Krioukov, D. Random hyperbolic graphs in  $d+1$  dimensions. *Phys. Rev. E* **109**, 054131 (2024).

68. Kovács, B., Balogh, S. G. & Palla, G. Generalised popularity-similarity optimisation model for growing hyperbolic networks beyond two dimensions. *Sci. Rep.* **12**, 968 (2022).

69. Caldarelli, G., Capocci, A., De Los Rios, P. & Muñoz, M. A. Scale-free networks from varying vertex intrinsic fitness. *Phys. Rev. Lett.* **89**, 258702 (2002).

70. Servedio, V. D. P., Caldarelli, G. & Buttà, P. Vertex intrinsic fitness: How to produce arbitrary scale-free networks. *Phys. Rev. E* **70**, 056126 (2004).

71. Boguñá, M. & Pastor-Satorras, R. Class of correlated random networks with hidden variables. *Phys. Rev. E* **68**, 036112 (2003).

72. Garlaschelli, D., Ahnert, S. E., Fink, T. M. A. & Caldarelli, G. Low-temperature behaviour of social and economic networks. *Entropy* **15**, 3148–3169 (2013).

73. Balogh, S. G., Palla, G. & Pollner, P. Generalised thresholding of hidden variable network models with scale-free property. *Sci. Rep.* **9**, 11273 (2019).

74. Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech.: Theory Exp.* **2008**, P10008 (2008).

75. Reinelt, G. *The Traveling Salesman: Computational Solutions for TSP Applications* (Springer-Verlag, 1994).

76. Dorigo, M. & Gambardella, L. M. Ant colonies for the travelling salesman problem. *Biosystems* **43**, 73–81 (1997).

## Acknowledgements

S. G. B. and G. P. acknowledge support from the National Laboratory for Health Security, RRF-2.3.1-21-2022-00006. This research was also partially supported by the National Research, Development and Innovation Office - NKFIH, grants No: OTKA-SNN 139598 and K128780.

## Author contributions

S.G.B. and G.P. developed the concept of the study, S.G.B. proposed and developed the application of the Travelling Salesman Problem (TSP) in the embedding method, S.G.B. and B.S. implemented the embedding methods, B.S. pre-processed the network data, tested the embedding methods and performed the analyses, B.S. and S.G.B. prepared the figures, B.S. prepared the tables, G.P. and S.G.B. wrote the paper, G.P., S.G.B. and T.V. contributed to the interpretation of the results. All authors reviewed the manuscript.

## 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/s42005-025-02306-8>.

**Correspondence** and requests for materials should be addressed to Gergely Palla.

**Peer review information** *Communications Physics* thanks Carlo Vittorio Cannistraci and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

**Reprints and permissions information** is available at <http://www.nature.com/reprints>

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025