Systematic comparison of graph embedding methods in practical tasks

Yi-Jiao Zhang, Kai-Cheng Yang, and Filippo Radicchi

Phys. Rev. E **104**, 044315 — Published 22 October 2021

DOI: 10.1103/PhysRevE.104.044315

Systematic comparison of graph embedding methods in practical tasks

Yi-Jiao Zhang,[1] Kai-Cheng Yang,[2] and Filippo Radicchi[2]

[1] *Institute of Computational Physics and Complex Systems, Lanzhou University, Lanzhou, Gansu 730000, China*

[2] *Center for Complex Networks and Systems Research, Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, Indiana 47408, USA*

(Dated: September 27, 2021)

Network embedding techniques aim at representing structural properties of graphs in geometric space. Those representations are considered useful in downstream tasks such as link prediction and clustering. However, the number of graph embedding methods available on the market is large, and practitioners face the non-trivial choice of selecting the proper approach for a given application. The present work attempts to close this gap of knowledge through a systematic comparison of eleven different methods for graph embedding. We consider methods for embedding networks in the hyperbolic and Euclidean metric spaces, as well as non-metric community-based embedding methods. We apply these methods to embed more than one hundred real-world and synthetic networks. Three common downstream tasks — mapping accuracy, greedy routing, and link prediction — are considered to evaluate the quality of the various embedding methods. Our results show that some Euclidean embedding methods excel in greedy routing. As for link prediction, community-based and hyperbolic embedding methods yield overall performance superior than that of Euclidean-space-based approaches. We compare the running time for different methods and further analyze the impact of different network characteristics such as degree distribution, modularity, and clustering coefficients on the quality of the embedding results. We release our evaluation framework to provide a standardized benchmark for arbitrary embedding methods.

I. INTRODUCTION

Representing complex networks in latent space, or network embedding, has generated a growing interest from multiple disciplines [1–3]. From a theoretical point of view, the geometric representation of a network may provide an intuitive explanation of key properties of real-world systems such as structural features [4], navigability [5, 6], and robustness [7, 8]; when it comes to applications, network embedding can be useful for graph analysis tasks like visualization [9], link prediction [10], and graph clustering [11, 12].

Many embedding methods use Euclidean space as their target space. Euclidean embedding is intuitive and can immediately be used in standard machine learning algorithms [2, 3]. However, network embedding methods are not limited to Euclidean space. For example, many recent approaches represent networks in hyperbolic space, where properties like hierarchy and heterogeneity can be easily captured [13–17]. Community structure can be seen as an alternative approach to network embedding in non-metric spaces [18].

The existence of so many available and diverse embedding techniques presents a challenge for practitioners when they have to choose the proper method for the application at hand. Standardized tests for systematic comparison among methods are lacking. The effectiveness of embedding methods is generally measured on limited types of tasks and small corpora of real-world networks. As a result, gauging the relative performance of a method with respect to another is difficult.

In this work, we address this gap of knowledge by performing a systematic comparison of representative embedding methods. We consider five hyperbolic embedding methods (HyperMap [13, 19], Mercator [14], Poincaré maps [15], Hydra [16], and HyperLink [17]), four Euclidean-space-based approaches (Node2vec [20], Laplacian Eigenmaps (LE) [21], HOPE [22], and Isomap [23]), and the two variants (relying on Louvain [24] and Infomap [25]) of the non-metric community embedding method [18]. We apply these methods to embed more than one hundred real-world and synthetic networks. Three downstream tasks, i.e., mapping accuracy, greedy routing, and link prediction, are considered to evaluate the quality of the various embedding methods. We assess how the performance of the various methods is affected by network characteristics such as degree distribution, modularity, and average clustering coefficient. The various methods are also compared in terms of their computational complexity and their number of tunable parameters.

Our findings indicate that Euclidean embedding methods such as Node2vec and Isomap represent the overall best choice for practitioners as they yield decent performance in all tasks. Hyperbolic embedding methods excel in link prediction; however, their high computational complexity impedes their application to large-scale networks. Community-based methods behave similarly to hyperbolic embedding methods, but they have a lower computational demand. Our systematic analysis includes many different embedding methods. However for obvious reasons, we could not include all methods that are currently available on the market or that will be developed in the future. For example, we did not consider geometric embeddings of networks induced by dynamical processes [26–29], see Ref. [1] for more examples. To ease the analysis of arbitrary embedding methods under our proposed experimental setting, we made it publicly available at `https://github.com/yijiaozhang/hypercompare`.

II. GRAPH VISUALIZATION

To qualitatively illustrate differences between different network embedding methods, we display graphical visualizations produced by the various methods for the same network topology, i.e., the autonomous system (AS) Internet network [30]. The network contains $N = 23,748$ nodes and $E = 58,414$ edges. Visualizations are displayed in Fig. 1.

It is important to stress that all visualizations are displayed in the two-dimensional Euclidean space, thus the original embedding is projected in this space using some ad-hoc recipes. For example, to yield decent embedding results, a high embedding dimension is required for Node2vec, LE, and HOPE. We therefore first learn their 128-dimensional embeddings and then use principal component analysis (PCA) to project the results into the two-dimensional plane of the figure. The visualization by Isomap is obtained directly by setting the embedding dimension to two. For hyperbolic embedding methods, we represent the embedded nodes with their polar coordinates or Poincaré coordinates and plot them in the two-dimensional Euclidean projection of the Poincaré disks. Finally, despite their potential use in graph drawing, we exclude the non-metric community-based embedding methods from the qualitative analysis in order to avoid the use of sophisticated projections in the two-dimensional Euclidean space.

To help the readers making sense of the visualizations, we color the autonomous systems, i.e., the nodes of the network, according to the continents where they are located in. We can see that, although different embedding methods yield drastically different visualizations, all of them can preserve geographic proximity to some extent, i.e., nodes within the same continent are often close one to the other in the visualizations. If we consider polar coordinates for all the embeddings (using the geometric center as the origin for Euclidean embeddings), it becomes clear that the angular coordinates encode the community structure of the graph [18, 31]. The radial coordinates, on the other hand, often convey network centrality information [31].

To quantify such connection, we use over a dozen real-world networks to empirically estimate the Spearman's correlation coefficients between the distance of a node from the geometric center of different embeddings denoted by r_c and different network centrality measures. The results are shown in Fig. 2. Clearly, the radial coordinates r_c of HyperMap, Mercator, and HyperLink represent the degree of the nodes [13, 14, 17]. r_c in the Isomap, Hydra, and Poincaré maps embeddings is highly correlated with closeness centrality [31]. For embeddings obtained by LE and HOPE, r_c is highly correlated with closeness and eigenvector centrality. However, we do not find obvious connection between node centrality and r_c in Node2vec embedding.

III. PERFORMANCE IN DOWNSTREAM TASKS

We now use downstream tasks to quantify the embedding quality of different methods. Specifically, we measure their performance in mapping accuracy, greedy routing, and link prediction. These tasks are conducted on 72 real-world networks representing social, biological, technological, transportation, and communication systems. Details of these networks are included in Ref. [39], Sec. I.

To summarize the results from all the networks for an embedding method on a task, we produce the complementary cumulative distribution function (CCDF) of a performance metric and calculate the area under the CCDF curve (CCDF-AUC) as the overall score. The CCDF-AUC matches the average value of the performance metric over the entire corpus of real-world networks and higher CCDF-AUC values indicate better overall performance.

Some embedding methods have free parameters that could affect the measured value of the performance metric. We tune the parameters for each method to find the optimal value of the overall performance, and use these parameter values for all networks, see Sec. VII A for details.

A. Mapping accuracy

A general principle respected by all the embedding methods is that proximity in the embedding space is representative for similarity or proximity in the original graph. Indeed, some embedding methods work by directly finding the embedding configuration that best preserves pairwise distance or other similarity relationships. For example, Isomap, Poincaré maps, and Hydra aim at preserving the shortest path distance among all pairs of nodes in the embedding space; Node2vec and HOPE try to encode certain similarity information. Other methods follow the principle implicitly by fitting the observed network against proximity-preserving network models (see Sec. VII A for details).

A natural way to assess the quality of a method is to measure how accurately the embedding method maps nodes in the space so that pairwise graph proximity is preserved in the embedding. We quantify the mapping accuracy of an embedding method in terms of the Spearman's correlation coefficient ρ between the pairwise shortest path distance in the network and the pairwise distance in the embedding space. Note that it is infea-

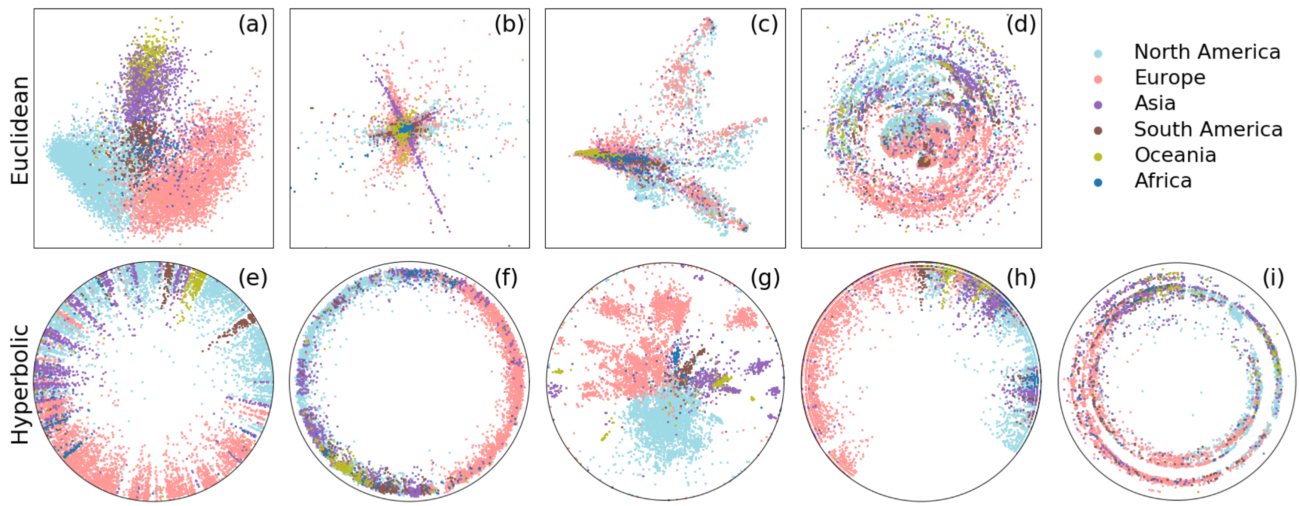

FIG. 1. **Geometric embedding of the Internet.** We display the visualization of the autonomous system (AS) Internet network in Euclidean space inferred by (a) Node2vec, (b) HOPE, (c) LE, (d) Isomap, and in the Euclidean projection of the hyperbolic embedding as inferred by (e) HyperMap, (f) Mercator, (g) HyperLink, (h) Poincaré maps, (i) Hydra. The color of a point is representative for the continent where the corresponding AS is located in. For clarity of the visualization, only nodes with degree larger than one are shown. For the visualization of Node2vec, HOPE, and LE, we first get the coordinates with dimension $d = 128$, and then use PCA to obtain a two-dimensional projection. For the other methods, we directly plot their two-dimensional embeddings.

TABLE I. **Key features and results of different network embedding methods.** From left to right, we report: name of the method, the target embedding space (space), programming language of the publicly available implementation (lang.), network structural information preserved by the method (struct. preserv.), computational complexity (complexity), CCDF-AUC for mapping accuracy (mapp. acc.), CCDF-AUC for greedy routing (greedy rout.), and CCDF-AUC for link prediction (link pred.). For each task, we highlight in bold face the CCDF-AUC values of the top three embedding methods. In the expressions of computational complexity, N is the number of the nodes, E is the number of the edges, d is the embedding dimension, C is the cost to compute each entry of the shortest path length matrix, e is the number of epochs (we set $e = 1,000$), $b = \min\{512, N/10\}$ is the batch size, m is the number of node layers, and $\langle k \rangle$ is the average degree of the network. More details about the methods can be found in Sec. VII A. The CCDF-AUC values are generated by aggregating the performance on 72 real-world networks for mapping accuracy and greedy routing. For link prediction, the CCDF-AUC values are computed on a subset of 46 real-world networks with size larger than 300. The CCDF-AUC values for HyperLink are marked with * because the method is unable to embed several networks. Restricting the analysis on the subset of real-world networks that HyperLink can process yields qualitatively similar results in all three tasks (see Ref. [39], Sec. II).

Method	Space	Lang.	Struct. preserv.	Complexity	Mapp. acc.	Greedy rout.	Link pred.
Node2vec [20]	Euclidean	Python	Tunable	$O(dN)$	0.561	**0.818**	0.787
HOPE [22]	Euclidean	Python	Global	$O(d^2E)$	0.575	**0.703**	0.769
Laplacian Eigenmaps (LE) [21]	Euclidean	Python	Local	$O(d^2E)$	0.464	0.566	0.749
Isomap [23]	Euclidean	Python	Global	$O(CN^2 + dN^2)$	**0.858**	**0.861**	0.848
HyperMap [19]	hyperbolic	C++	Local	$O(N^2)$	0.388	0.584	0.840
Mercator [14]	hyperbolic	Python	Local	$O(N^2)$	0.557	0.530	**0.902**
HyperLink [17]	hyperbolic	C++	Local	$O(m\langle k\rangle N^2)$	0.516*	0.510*	0.897*
Poincaré maps [15]	hyperbolic	Python	Global	$O(N^2 + ebN)$	**0.628**	0.494	0.822
Hydra [16]	hyperbolic	R	Global	$O(N^\alpha), \alpha > 2$	**0.799**	0.683	0.846
Community embedding (Infomap) [18]	non-metric	Python	Local	$O(N\log N)$	0.618	0.619	**0.902**
Community embedding (Louvain) [18]	non-metric	Python	Local	$O(N\log N)$	0.561	0.454	**0.914**

sible to consider every possible pair of nodes for large networks. We therefore use a maximum of 10^5 random pairs of nodes to approximate the Spearman's ρ in case the total number of node pairs $N(N-1)/2 > 10^5$.

As mentioned above, we calculate the mapping accuracy of different embedding methods on 72 real-world networks. For sake of clarity, in Fig. 3(a), we plot the CCDF for some selected methods only. The CCDF-AUC values of all embedding methods are listed in Table I. Overall, we find that all methods do a good job in preserving graph proximity in the embedding space.

Isomap and Hydra top the ranking on this task. The finding is not surprising given that both methods aim at optimizing the congruence between pairwise proxim-

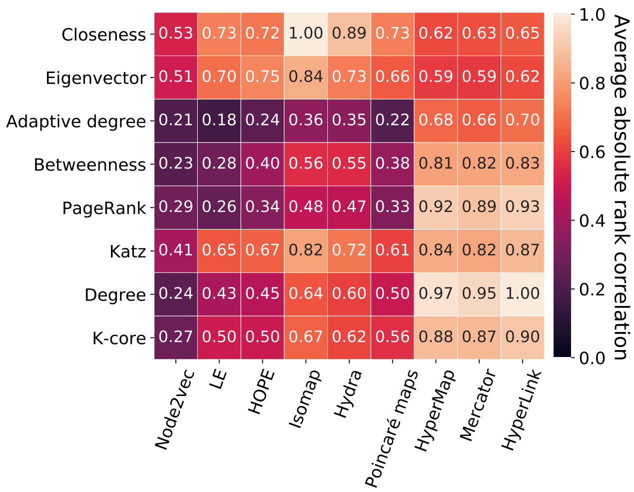

FIG. 2. **Interpretation of the radial coordinates in embedding space.** We show the pairwise Spearman's correlation coefficients between the distance of a node from the geometric center of different embeddings and different centrality metrics such as closeness[32], eigenvector[33], adaptive degree[34], betweenness[35], PageRank[36], Katz[37], degree, and K-core[38] centralities. The values are obtained by averaging the results from 13 real-world networks with size $N \in [1000, 5000]$ in our dataset.

ity of nodes in the graph and in the embedding space. The mapping accuracy of Poincaré maps is not as high even though it also aims at preserving the shortest distance among pairs of nodes. An advantage of Isomap and Hydra is that they can perform embedding in arbitrarily high-dimensional spaces, while Poincaré maps can only work in two-dimensional hyperbolic space. Our experiments show that the mapping accuracy of Isomap and Hydra increases as the embedding dimension increases. The results of Fig. 3(a) and Table I are obtained with $d = 128$. By setting $d = 2$, Poincaré maps achieves the best performance; the performance of Hydra is also better than that of Isomap. The main reason is that the two-dimensional Euclidean space may not be large enough to properly embed large networks (see Ref. [39], Sec. II).

B. Greedy routing

Network embeddings may be used in greedy routing protocols devised for efficient network navigation [5, 74]. The task regards the delivery of a packet from a source node s to a target node t. The packet performs hops on the network edges, moving from one node to one of its neighbors at each stage of the navigation process. In particular, according to the greedy protocol, at every stage of the process the packet moves to the neighbor that is closest to target t according to a metric of distance. Such a metric of distance is computed using knowledge about the embedding space and the nodes' coordinates. If the packet reaches the target node t, the delivery is considered successful. However, if the packet visits the same node twice, the delivery fails. A good embedding for this task should be able to allow a high rate of successful deliveries along delivery paths that are not much longer than the true shortest paths.

In this work, we follow the literature and use the greedy routing score (GR score) to measure the performance of different embeddings in greedy routing [75]. The GR score is defined as

$$\text{GR score} = \frac{2}{N(N-1)} \sum_{i>j} \frac{D_{ij}}{R_{ij}} , \tag{1}$$

where D_{ij} is the shortest path length between nodes i and j in the original network, and R_{ij} is the length of the actual delivery path followed by the packet according to the greedy routing protocol. All pairs of nodes are considered in the sum of Eq. (1), including those leading to successful and unsuccessful deliveries. For an unsuccessful delivery, R_{ij} is infinite and $D_{ij}/R_{ij} = 0$. For a successful delivery along one of the shortest paths connecting i to j, we have $D_{ij}/R_{ij} = 1$. The GR score is 0 when all the deliveries are unsuccessful. The GR score equals 1 when all packets are successfully delivered along the shortest path in the original network. Note that it is impossible to test every pair of source-target nodes for large networks. In our experiments, we randomly select 10^4 source-target pairs to approximate the GR score in case the total number of node pairs $N(N-1)/2 > 10^4$.

We show the CCDF of the GR scores for selected embedding methods in Fig. 3(b) and the CCDF-AUC values for all methods in Table I. We note that all methods can facilitate network navigation to some extent. In general, there is a non-trivial relationship between the performance in mapping accuracy and the one in greedy routing. It is already known that Isomap performs well in this task [31]. The relatively good performance of Node2vec is instead a new result. In part, the result can be explained by considering that embeddings obtained by Node2vec are based on the exploration of graph paths, a process that well informs a greedy navigation protocol. On the other hand, it seems that Euclidean-space-based embeddings better suit for this task than embedding methods relying on hyperbolic geometry and non-metric spaces. A possible explanation of our finding is that many of the non-Euclidean embedding methods focus on preserving local network properties rather than global ones. The only exception to this rule is Hydra, which in fact displays relatively higher performance than that of the other hyperbolic embedding methods.

C. Link prediction

Link prediction is a standard task to evaluate the performance of graph embedding methods [3, 10]. The goal

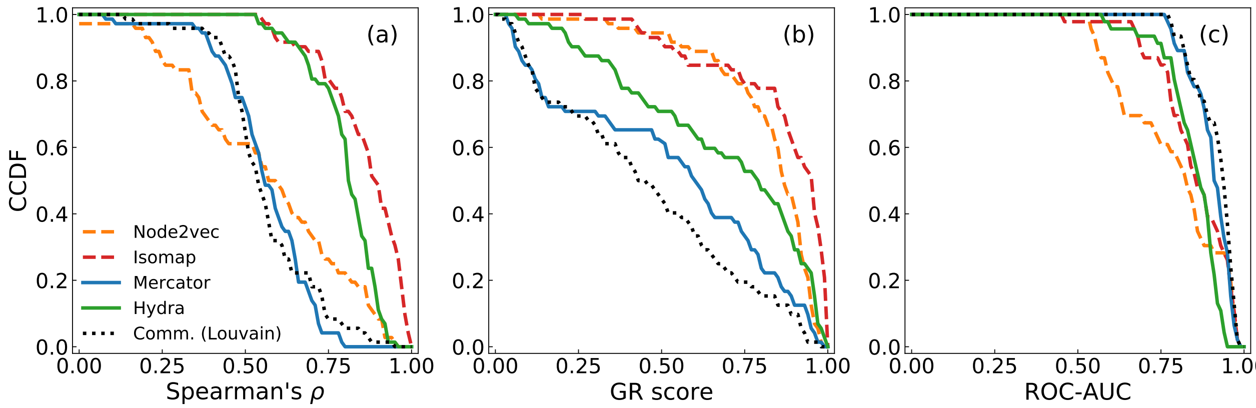

FIG. 3. **Aggregate performance in downstream tasks.** We show the complementary cumulative distribution function (CCDF) of (a) the Spearman's correlation coefficients of the mapping accuracy, (b) the GR scores of greedy routing, and (c) the ROC-AUC scores of link prediction for different embedding methods on real-world networks. The average performance over all networks of an embedding on a task is equal to the area under the curve of the corresponding CCDF. Since most of the embedding methods are stochastic, the data points in the figure are obtained by averaging the results from five independent repetitions.

is predicting the existence or the non existence of edges between non-observed pairs of nodes. There are potentially many different ways to implement the task. In our case, we first remove 30% randomly chosen edges from the original network while ensuring that the remaining graph is still formed by a single connected component. The removed edges are used as the positive test set. Then, we randomly sample a negative test set of non-existent edges with size identical to that of the positive test set. The remaining network is fed to the embedding methods. For each pair of nodes, the closer they are in the embedding space, the more likely they are connected. We stress that the information about removed edges is not provided to any embedding methods except for Hyperlink, for which the percentage of the removed edges is an input parameter.

The ability of an embedding to distinguish the edges from the positive and negative sets is measured by the area under the receiver-operating characteristic curve (ROC-AUC). The ROC-AUC score ranges from 0.5 to 1. For perfect prediction, the ROC-AUC score equals to 1. The score is 0.5 for random guesses. For small networks, removing 30% of the edges may substantially distort the network structure and the link prediction results. Therefore, we only consider real-world networks with more than 300 nodes for the link prediction task in this paper. We show the CCDF of ROC-AUC scores for selected embedding methods in Fig. 3(c) and report the CCDF-AUC values for all methods in Table I as before. All embedding methods yield comparable performance in this task. Mercator and the community-based methods yield slightly better performance than the other methods. The result can be a reflection of the fact that the embeddings are obtained by fitting graphs against probability laws for network connections, which immediately provide predictions for missing links. We also measure the area under the precision-recall curve (AUPR) for each method in the link prediction task, the results are qualitatively similar to those obtained for ROC-AUC (see Ref. [39], Sec. II).

D. Embedding performance on synthetic networks

In order to systematically analyze the performance of the different embedding methods, we also use 34 instances of synthetic networks generated by five types of network models: the popularity-similarity-optimization (PSO) model [4, 19], the Lancichinetti-Fortunato-Radicchi (LFR) model [76], the configuration model with power-law degree distribution and Poisson degree distribution (power-law networks and Poisson networks), and the model for spatial networks by Daqing *et al.* [77] (see Sec. VII B for details of network models and parameters used).

We apply the embedding methods to the synthetic networks, repeat the evaluation on three downstream tasks and report the performance in Table II. We can see that the results on the synthetic network models are consistent with the results obtained on the real-world networks. Isomap and Hydra are the top two methods for mapping accuracy. Euclidean embeddings such as Node2vec and Isomap perform better than hyperbolic and community-based embeddings on greedy routing, while hyperbolic and community-based embeddings outperform Euclidean-based embedding methods on link prediction.

By tuning the parameters of the network models, we can further study the effect of network characteristics on the performance of different embedding methods. The network models and the corresponding network characteristics analyzed in this paper are listed in Table III.

TABLE II. **Embedding performance on synthetic networks.** We summarize all the results obtained by the different embedding methods on the synthetic network models considered in this paper (i.e., PSO models, LFR networks, power-law networks, spatial networks, and Poisson networks). From left to right, we report: name of the method, the CCDF-AUC of mapping accuracy on the various network models, the CCDF-AUC of greedy routing scores on the same set of network models, and the CCDF-AUC of link prediction ROC-AUC scores on the same set of network models. Link prediction results for Poisson networks are excluded since no meaningful prediction can be made for the edges of random and homogeneous networks. See details about synthetic networks in Sec. VII B. We highlight in bold face the top three methods for each network model and task combination. Some values for Mercator and HyperLink are marked with * because the methods are not able to embed several networks. The results are qualitatively similar if we restrict the analysis on the subset of networks that all methods can process.

	Mapping accuracy					Greedy routing					Link prediction			
Method	PSO	LFR	power-law	spatial	Poisson	PSO	LFR	power-law	spatial	Poisson	PSO	LFR	power-law	spatial
Node2vec	0.710	**0.626**	**0.692**	**0.692**	**0.578**	**0.892**	**0.886**	**0.925**	**0.903**	**0.876**	0.825	0.674	0.491	**0.770**
HOPE	**0.740**	0.444	0.662	0.547	0.442	0.742	**0.740**	**0.873**	**0.775**	**0.768**	0.750	0.678	0.523	0.697
LE	0.540	0.462	0.523	0.485	0.452	0.785	0.641	0.673	0.662	0.692	0.762	0.662	0.607	0.618
Isomap	**0.943**	**0.789**	**0.853**	**0.863**	**0.652**	**0.872**	**0.846**	**0.887**	**0.885**	**0.794**	0.818	**0.729**	**0.647**	**0.733**
HyperMap	0.379	0.314	0.365	0.283	0.266	**0.797**	0.265	0.528	0.371	0.294	**0.848**	0.695	**0.653**	0.660
Mercator	0.459*	0.384	0.375	0.450	0.339	0.607*	0.198	0.253	0.298	0.192	0.847*	0.698	0.623	0.687
Poincaré maps	0.618	0.379	0.412	0.489	0.315	0.577	0.256	0.228	0.418	0.218	0.808	0.672	0.590	0.680
HyperLink	0.303	0.375	0.370	0.345	0.317*	0.593	0.295	0.313	0.355	0.233*	0.742	0.719	0.642	0.662
Hydra	**0.898**	**0.666**	**0.773**	**0.685**	**0.528**	0.765	0.371	0.574	0.422	0.480	0.783	0.671	**0.663**	0.632
Comm. (Infomap)	0.586	0.434	0.402	0.437	0.329	0.743	0.318	0.473	0.442	0.360	**0.883**	**0.735**	0.633	**0.738**
Comm. (Louvain)	0.543	0.384	0.353	0.388	0.309	0.592	0.178	0.185	0.203	0.149	**0.883**	**0.740**	0.638	0.732

TABLE III. Synthetic network models considered in our analysis together with the corresponding network characteristics varied in our tests.

Network model	Characteristic
PSO model [4, 19]	Clustering coefficient
LFR model [76]	Modularity
Poisson network	Average degree
power-law network	Power-law exponent
spatial networks [77]	Power-law exponent

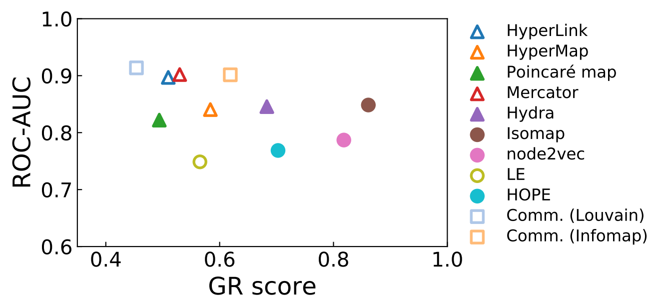

FIG. 4. **Average performance in link prediction and greedy routing over a large corpus of real-world networks.** We summarize here the same results as of Table I. We plot the CCDF-AUC values of ROC-AUC scores and GR scores for different embedding methods. Circles, triangles and squares represent Euclidean-, hyperbolic- and community-based embedding methods, respectively. The hollow and solid symbols represent methods that preserve local and global network structural information, respectively.

We find that certain network characteristics have strong effects on downstream tasks as follows: (1) the ability of embedding methods to preserve graph distance deteriorates as the density of the network grows; (2) the ability of embedding methods to inform the greedy routing protocol improves as the network clustering coefficient increases but its modularity decreases; (3) the ability of embedding methods in inferring links between non-observed pairs of nodes improves as the network clustering coefficient increases, the network modularity grows, and the heterogeneity of the degree distribution increases. Detailed results can be found in Ref. [39], Sec. IV. These effects are universal across different methods with a few exceptions. For example, Isomap and Node2vec perform well in greedy routing regardless of the network characteristics.

E. Summary of the results

To provide an overview of the performance of different embedding methods, we focus on link prediction and greedy routing, and summarize the results in Fig. 4. The same analysis for synthetic networks can be found in Ref. [39], Sec. IV. We can see that Isomap and Node2vec outperform the other methods in greedy routing while community embedding, Mercator, and HyperLink yield better performance in link prediction. However, no single method outperforms all the other methods in both tasks according to Fig. 4.

We remark that the two tasks are fundamentally different, as link prediction is a local prediction task while greedy routing is a global discovery task. Also, the position of an embedding method in the performance diagram shown in Fig. 4 seems partially predictable based on the type of space targeted by the embedding method and/or the type of network structural information that the method is able to preserve (see Table I). As a general rule of thumb, methods that preserve local information excel in link prediction, and algorithms that preserve global structure achieve optimal performance in greedy routing.

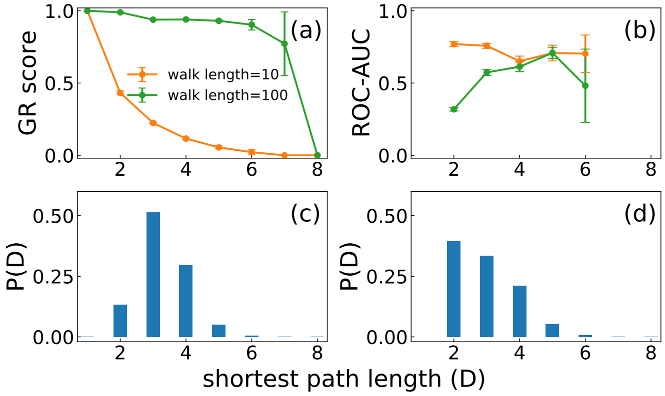

FIG. 5. **Greedy routing and link prediction results obtained by Node2vec with different walk length on the IPv6 Internet network.** We display (a) the relation between GR score and the shortest path length between node pairs involved when using Node2vec with different walk length ($l = 10$ and $l = 100$) to guide greedy routing, (b) same as (a), but for ROC-AUC scores in link prediction, (c) the distribution of distance between node pairs involved in greedy routing, and (d) same as (c), but for link prediction. The data points in the figure are obtained by averaging the results of 10 experiments, the error bars indicate one standard deviation from the mean.

To further validate our rule of thumb, we take advantage of Node2vec. The algorithm acquires structural information by means of random walks with restart. The length of the random walks serves as a proxy for the typical scale of structural information that is preserved by the embedding. We apply Node2vec with walk length $l = 10$ and $l = 100$ on the Ipv6 Internet network [78] and use the resulting embeddings to perform greedy routing and link prediction. Instead of reporting the overall performance, we group the node pairs involved in the tasks by their shortest path distance in the network and then calculate the scores within each group. For $l = 10$, the GR score decreases quickly as the distance between source and target nodes increases. The performance for $l = 100$ in greedy routing is instead almost unaffected by the source-to-target distance. Performance in link prediction obtained for $l = 10$ is far better than the one obtained for $l = 100$. We note that the vast majority of links tested have distance $D = 2$, which corresponds to the maximum gap in performance between the embeddings obtained for $l = 10$ and $l = 100$.

IV. COMPUTATIONAL COMPLEXITY AND RUNNING TIME

Scalability is another important factor when choosing the proper embedding method. We summarize the computational complexity in Table I. Hyperbolic embedding methods have $O(N^2)$ computational complexity at least, while Euclidean and non-metric methods often scale linearly with the system size.

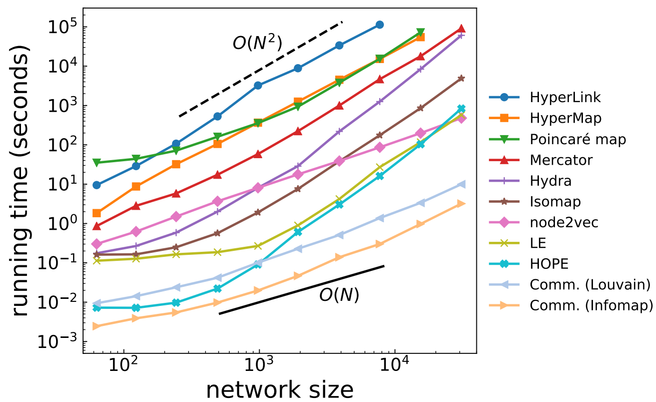

FIG. 6. **Running time vs. network size.** We show the running time of different embedding methods in relation to the size of PSO models. The network size ranges from $N = 2^6$ to $N = 2^{15}$. Other parameters of the PSO models are: average degree $\langle k \rangle = 5$, power-law exponent $\gamma = 2.1$, temperature $T = 0.5$. Each data point is the average of five simulations. For HyperMap, we use the hybrid algorithm without correction steps and enable the speedup mode by setting $k_{\mathrm{speedup}} = 10$ (see Sec. VII A for details). The black full line indicates linear scaling; the black dashed line denotes quadratic scaling.

TABLE IV. **Node2vec and community embedding on large networks.** We report the performance on mapping accuracy (Spearman's ρ), greedy routing (GR score), and link prediction (ROC-AUC score) as well as the running time (seconds) of Node2vec and community embeddings with Infomap and Louvain algorithms on the YouTube friend network (N = 1,134,890) and the AS Skitter network (N = 1,694,616).

Network	Metric	Node2vec	Infomap	Louvain
YouTube friend	Mapping accuracy	0.620	0.499	0.352
	Greedy routing	0.478	0.071	0.588
	Link prediction	0.959	0.962	0.976
	Running time	33,045 s	4,938 s	732 s
AS Skitter	Mapping accuracy	0.582	0.403	0.033
	Greedy routing	0.348	0.117	0.363
	Link prediction	0.998	0.991	0.983
	Running time	85,356 s	3,149 s	1,895 s

To directly compare the running time of the various embedding techniques, we apply all the methods to a series of networks with different sizes generated by the popularity-similarity-optimization (PSO) model [4, 19]. All the experiments are performed on a server equipped with Intel Xeon Platinum 8268 CPUs (2.90GHz) and 1.5TB RAM. Although the server have multiple processors, all the methods are allowed to use one processor only. Figure 6 shows the relation between the running time and the network size for all the embedding methods. The results confirm that the Euclidean and the non-metric embedding methods tend to be much faster than the hyperbolic embedding methods. When we apply the embedding algorithms to different network models and measure their computational time, results are qualitatively similar.

Among the methods tested, only Node2vec and community embedding methods (both variants with Louvain [24] and Infomap [25]) can easily scale up to large networks. As a demonstration, we apply them to two real-world networks with more than one million nodes. They complete the embedding in about 24 hours and 1.4 hours, respectively, without compromising the performance on downstream tasks (see details in Table IV). In order to avoid unnecessary memory and time usage while applying Node2vec on networks with millions of nodes, we use a program optimized for unweighted networks and specific algorithm parameter values ($p = 1$ and $q = 1$).

In our experiments, we try to use the implementation shared by the creators whenever possible. For classic methods such as LE and Isomap, we use the implementation provided by the Python package scikit-learn [79]. We implement Node2vec and community embedding in Python with the help of some open source packages. Note that this is not the ideal setup for comparing the running time of different methods since the programming language (see Table I) used can heavily affect the results and the implementation used in our experiments can sometimes be further optimized. Instead, our experiments mimic a more practical scenario where practitioners hope to quickly apply the embedding methods without spending too much time improving or even implementing the methods themselves. The results here provide a rough estimation of the expected running time when using the most accessible implementation.

V. DISCUSSION

In this work, we consider a large corpus of real-world and synthetic networks, and measure the performance of several embedding methods in solving specific network tasks. We find that Isomap and Node2vec outperform the other methods in greedy routing. As for link prediction, community embedding, Mercator, and HyperLink all yield excellent performance. Our results on synthetic network models indicate that type and feature of the target networks are not important when choosing the embedding method. Instead, one possible principle is that the methods aim at preserving global network structure excel in greedy routing, and methods only capturing local information achieve optimal performance in link prediction. Also, our analyses of the algorithm running time show that hyperbolic methods are much slower than other methods, suggesting that they are not yet well suited for embedding large-scale networks.

We stress that not all factors that are important in the decision of using an embedding method instead of another are measurable and quantifiable. Some methods may provide valuable insights into the characteristics of networks although their performance may not be comparable with that of others in certain tasks. For practical tasks, many other features may also be crucial. A method can be chosen because its implementation is easy to access and configure, and the method can process different input networks. For instance, we had to exclude some embedding methods from our experiments because we were unable to find adequate implementations. Also, some of the methods considered in this paper require proper calibration of input parameters to be successful in downstream tasks [12]. For example, choosing a large value for the embedding dimension for Node2vec, LE, and HOPE does not always lead to good results. These methods can suffer from overfitting on certain tasks when the embedding dimension is too high. Calibration is generally a computationally expensive operation, and there may be practical situations where calibration can not even be performed.

All things considered, we believe that the Euclidean embedding methods like Node2vec and Isomap should be the first options for practitioners since they have stable and widely available implementations, and they yield decent performance in all tasks. The non-Euclidean embedding methods still present some challenges. Their non-Euclidean nature makes it non-trivial to incorporate their results to common downstream tasks in general, which may limit their applicability. Nevertheless, the fact that the non-Euclidean methods stand out in certain tasks suggests that they have a great potential, calling for further investigation and improvement.

VI. ACKNOWLEDGEMENTS

Y.-J.Z. acknowledges support from China Scholarship Council (No.201906180029). Y.-J.Z. and F.R. acknowledge partial support from the National Science Foundation (Grant No. CMMI-1552487). F.R. acknowledges partial support from the US Army Research Office (Grant No. W911NF-21-1-0194).

VII. METHODS

A. Network Embedding Methods

Network embedding methods are sets of procedures that map the nodes of the input network into points in the target space. The coordinates of the nodes serve as the vector representation of the networks and the pairwise distance of different nodes correspond to their proximity or similarity in the input networks. Depending on the target spaces, the representation of the embedded network and the definition of the distance between nodes in the embedding space vary. Here we group different embedding methods by their target spaces, i.e., Euclidean, hyperbolic, and non-metric spaces.

1. Euclidean embedding methods

For Euclidean embedding methods, each node i can be described by a d-dimensional vector $\mathbf{x}_i = (x_i^{(1)}, ..., x_i^{(d)})$ where d is the space dimension and serves as a free parameter for all Euclidean embedding methods. There are several ways to calculate the distance between two nodes in Euclidean embedding space. The most common two, Euclidean distance and dot product, are used in this work. The Euclidean distance between node i and j is defined as

$$\mathrm{dist}_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{\sum_{v=1}^{d} (x_i^{(v)} - x_j^{(v)})^2} \ . \tag{2}$$

The dot product between node i and j is given by

$$\mathbf{x}_i \cdot \mathbf{x}_j = \sum_{v=1}^{d} x_i^{(v)} x_j^{(v)} \ . \tag{3}$$

Note that the similarity between two vectors is proportional to their dot product. So we use

$$\mathrm{dist}_{ij} = -\mathbf{x}_i \cdot \mathbf{x}_j \ , \tag{4}$$

as effective distance in the dot product approach.

Node2vec, LE, HOPE, and Isomap are the four Euclidean embedding methods we consider in this paper. We use either the distance of Eq. (2) or Eq. (4) depending on the objective function that a method minimizes and the actual downstream task. Eq. (2) is used for LE and Isomap in this paper. For Node2vec and HOPE, we use Eq. (4) for link prediction according to their objective functions, and Eq. (2) for mapping accuracy and greedy routing because it yields much better performance than when distance is calculated according to Eq. (4) (see Ref. [39], Sec. III).

Next, we briefly introduce each method and the parameters used in our experiments.

(1) *Node2vec* [20]: Node2vec first generates multiple node sequences using random walks with fixed length, then finds the vector representations that maximize the probability of co-occurrence of the nodes in the sequences. There are some tunable parameters for Node2vec, such as walk length l, window size, the bias parameters of the random walk dynamics p and q, and the embedding dimension d. In this work, we use the default setting: window size = 10, $p = 1$ and $q = 1$.

We find that the walk length can greatly affect different downstream tasks. The main reason is that walk length directly control the type of information that the resulting embedding preserves. Short walk lengths preserve local structural information; long walk lengths preserve global structure. As expected, according to our tests on several real-world networks, increasing the walk length improves the performance of mapping accuracy and greedy routing, but worsens link prediction (see Ref. [39], Sec. III). So we set $l = 10$ for link prediction and $l = 100$ for the other two tasks in this paper.

In general, the larger the dimension d, the better the embedding. But for Node2vec, the performance in downstream tasks may decrease slightly as d increases (see Ref. [39], Sec. III). In this work, we set $d = \min\{N, 128\}$ for all embedding methods that can work with high ($d > 2$) dimensional embedding space, which is considered a sufficiently high value to achieve nearly-optimal embeddings of networks [80]. We make this choice to maintain the simplicity of the experiments without introducing strong biases towards certain methods.

(2) *Laplacian Eigenmaps (LE)* [21]: LE aims to place the nodes that are connected with each other closely in the embedding space by minimizing the objective function

$$E_{\mathrm{LE}} = \sum_{ij} \|\mathbf{x}_i - \mathbf{x}_j\|^2 A_{ij} = tr(\mathbf{X}^T \mathbf{L} \mathbf{X}) \ , \tag{5}$$

where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n)^T$ is the low-dimensional representation matrix of the network, $\mathbf{A}$ is the adjacency matrix of the network ($A_{ij} = A_{ji} = 1$ if nodes i and j are connected, otherwise $A_{ij} = A_{ji} = 0$), $\mathbf{L} = \mathbf{K} - \mathbf{A}$ is the Laplacian matrix and $\mathbf{K}$ is the diagonal matrix with $K_{ii} = \sum_j A_{ji}$. LE further requires $\mathbf{X}^T \mathbf{K} \mathbf{X} = \mathbf{I}$ to eliminate trivial solutions. To obtain a d-dimensional embedding, one can simply extract the eigenvectors that correspond to the d smallest non-zero eigenvalues of the solution to $\mathbf{L}\mathbf{x} = \lambda \mathbf{K} \mathbf{x}$.

LE only has one tunable parameter: dimension d. We set it to $d = \min\{N, 128\}$.

(3) *HOPE* [22]: Given a node similarity definition, HOPE seeks to preserve the similarity matrix $\mathbf{S}$ in the embedding space by minimizing

$$E_{\mathrm{HOPE}} = \|\mathbf{S} - \mathbf{x}\mathbf{x}^T\|, \tag{6}$$

through singular value decomposition (SVD). HOPE can work with different node similarity definitions; here we use Katz index, which is calculated by

$$\mathbf{S}^{\mathrm{Katz}} = \beta \sum_{l=1}^{\infty} \mathbf{A}^l \ , \tag{7}$$

where $\mathbf{A}$ is the adjacency matrix of the network and β is the decay parameter. HOPE requires $\beta < 1/\lambda_{max}$, with λ_{max} principal eigenvalue of the matrix $\mathbf{A}$. We set $\beta = 1/\lambda_{max} - 0.001$ for all experiments. The embedding dimension d is set to $d = \min\{N, 128\}$.

(4) *Isomap* [23]: Isomap tries to preserve the shortest path distance between each pair of nodes. It first calculates the shortest path distance matrix $\mathbf{D}$ of a network. Then multidimensional scaling (MDS) [81] is applied to $\mathbf{D}$ to obtain a d-dimensional representation of the network that minimize the stress function

$$E_{\mathrm{ISO}} = \sum_{ij} \left[D_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|\right]^2 \ . \tag{8}$$

We set the embedding dimension $d = \min\{N, 128\}$ for Isomap in all experiments.

2. *Hyperbolic embedding methods*

For hyperbolic embedding, nodes are usually considered as points on the Poincaré disk. Two coordinate systems are often used in the literature, i.e., the polar coordinates (r, θ) and the Poincaré coordinates $\mathbf{y} = (y^{(1)}, y^{(2)})$. The Poincaré coordinates are similar to the Euclidean coordinates but represent points in hyperbolic space. They can also be extended to arbitrary dimension, i.e., $\mathbf{y} = (y^{(1)}, ..., y^{(d)})$, to represent points in the Poincaré ball.

When using the polar coordinates, the distance between node i and j can be calculated by

$$\mathrm{dist}_{ij} = \mathrm{arcosh}(\mathrm{cosh}r_i\mathrm{cosh}r_j - \mathrm{sinh}r_i\mathrm{sinh}r_j\cos(\Delta\theta)) \ , \tag{9}$$

where $\Delta\theta = \pi - |\pi - |\theta_i - \theta_j||$ is the angle between the two nodes.

When using the Poincaré coordinates, the distance between node i and j can be calculated by

$$\mathrm{dist}_{ij} = \mathrm{arcosh}\left(1 + 2\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{(1 - \|\mathbf{y}_i\|^2)(1 - \|\mathbf{y}_j\|^2)}\right) . \tag{10}$$

The two-dimensional Poincaré coordinates $(y^{(1)}, y^{(2)})$ and the polar coordinates (r, θ) of hyperbolic space can be converted to each other by

$$\begin{aligned} r &= 2\mathrm{artanh}(\sqrt{(y^{(1)})^2 + (y^{(2)})^2}) \ , \\ \theta &= \mathrm{atan2}(y^{(2)}, y^{(1)}) \ . \end{aligned} \tag{11}$$

Among the hyperbolic embedding methods considered in this work, HyperMap, Mercator, and HyperLink use polar coordinates; Poincaré maps and Hydra use Poincaré coordinates. Poincaré maps focus on the two-dimensional disk while Hydra can embed networks in higher-dimensional hyperbolic spaces.

We briefly introduce each method and the parameters used in our experiments in the following.

(1) *HyperMap* [13, 19]: Popularity-similarity-optimization (PSO) model [4, 19] is a growing network model that can simultaneously capture the heterogeneity degree distribution and the strong clustering structure of real-world networks. Nodes of PSO model are embedded in hyperbolic space and their coordinates have clear interpretations: the radial coordinate represent the node popularity, and the difference between angular coordinates of a node pair represents the similarity between them. The PSO model consists of a probability law for the existence of edges between pairs of nodes in the network depending on their distance in the hyperbolic space, i.e., Eq. (9).

As an embedding method, HyperMap embeds an input network to the hyperbolic space by fitting the network against the PSO model. The fit is performed by maximizing the likelihood of observed edges according to the PSO connection probability law. As the maximum likelihood problem cannot be solved exactly, different variants of the HyperMap algorithm exploit different strategies to find approximate solutions. These variants include the link-based method [19], the common-neighbors based method (also called HyperMap-CN) [13], and the hybrid method [13] that uses the common-neighbors based method for high degree nodes and the link-based method for the rest of the nodes. The computational complexity of the above-mentioned algorithms is at least $O(N^3)$. There is also a speed-up version of the hybrid method, which can reduce the computational complexity of the method down to $O(N^2)$ without compromising the embedding quality too much.

In this paper, we use the speed-up version of HyperMap. This method has extra correction steps that can marginally improve the results but have a very high computational complexity so we disable them. It has a parameter k_{speedup} to control the level of acceleration. We set $k_{\mathrm{speedup}} = 10$ for networks with size $N < 10,000$ and $k_{\mathrm{speedup}} = 40$ for networks with size $N > 10,000$.

The input parameters of HyperMap include the temperature $T \in [0, 1)$, which reflects the average clustering level of a network. A higher temperature means that the network is less clustered. Identifying the ideal temperature value for each network requires scanning the parameter space, which is infeasible in our experiments. Instead, we test the overall performance of HyperMap for different values of the temperature parameter on several real-world networks and find that temperatures that are not too large nor too small generally yield decent performance (see Ref. [39], Sec. III). So we set temperature $T = 0.5$ in all experiments. Another input parameter of HyperMap is the exponent $\gamma \geq 2$ of the power-law degree distribution of the network. Note that not all real-world networks display a power-law degree distribution. To apply HyperMap to all the networks considered, we use the code shared by Broido *et al.* [82] to estimate a suitable γ value for every network. If the estimated γ value is smaller than 2.1, we set $\gamma = 2.1$.

(2) *Mercator* [14]: Mercator learns the hyperbolic representations of networks by matching them with the $\mathbb{S}^1/\mathbb{H}^2$ model [83, 84]. The $\mathbb{S}^1/\mathbb{H}^2$ model is the static version of the PSO model. While PSO model can only generate networks with pure power-law degree distribution, the $\mathbb{S}^1/\mathbb{H}^2$ model can generate networks with arbitrary degree distributions. Besides the input network itself, Mercator does not require any input parameters.

(3) *Poincaré maps* [15]: Poincaré maps aims to preserve the pairwise shortest path length just like Isomap. There are several free parameters of Poincaré maps. For example, the Gaussian kernel width σ_P is related to the calculation of the global proximity of the original network, the scaling parameter γ_P is used to tune the scattering of the embedding. We find that these parameters have little effect on the results. In this paper, we use the default setting $\sigma_P = 1$ and $\gamma_p = 2$ in all experiments. The maximum number of epochs for the embedding optimization is set to $e = 1000$.

(4) *Hydra* [16]: Like Poincaré maps and Isomap, Hydra (HYperbolic Distance Recovery and Approximation) also seeks to preserve pairwise shortest path length. The difference between Poincaré maps and Hydra is that Hydra can work in hyperbolic spaces of arbitrary dimension, while Poincaré maps is designed for the two-dimensional space only. The dimension d is the only one free parameter of Hydra. We set $d = \min\{N, 128\}$ in all experiments.

(5) *HyperLink* [17]: HyperLink is a model-based hyperbolic embedding method designed for link prediction. It tries to fit the networks to the random hyperbolic graphs (RHGs) model, which is equivalent to the $\mathbb{S}^1/\mathbb{H}^2$ model used in Mercator. HyperLink assumes that a fraction p of links are missing when embedding a network. In addition to p, other input parameters of HyperLink include the exponent $2 < \gamma < 3$ of the degree distribution, the temperature T, the number of layers m, and the coefficient g that controls the size of the mesh in the angular space. In our experiments, we use the default settings $m = 20$ and $g = 1$. We aid the method by setting $p = 0.3$ in link prediction, and $p = 0$ in other tasks. The estimation of γ is the same as in HyperMap. We set $\gamma = 2.1$ if the estimated $\gamma < 2.1$ and $\gamma = 2.9$ if the estimated $\gamma > 2.9$ in order to satisfy the requirement. Like HyperMap, the temperature T is a free parameter for HyperLink. We test the overall performance of HyperLink for different T values on some real-world networks, and find that $T = 0.3$ yields the best performance overall (see Ref. [39], Sec. III). Therefore, we set $T = 0.3$ in our experiments.

3. Non-metric embedding method

Community embedding [18] is a non-metric embedding method inspired by the analogy between hyperbolic embeddings and network community structure. It embeds networks using information about their community structures: node i is represented by the coordinates (k_i, σ_i) where k_i is node's degree and σ_i is the index of the community that the node belongs to. There are many community detection algorithms available on the market. Here, we use two popular ones: Infomap [25] and Louvain [24]. After the community partition of a network is obtained, nodes in the same communities are merged together to generate supernodes, which then form a supernetwork. The edge weight between community a and b in the supernetwork is defined as

$$w_{ab} = 1 - \ln \rho_{ab} \text{ , if } \rho_{ab} > 0 \text{ ,} \tag{12}$$

and $w_{ab} = 0$, otherwise. ρ_{ab} is the ratio between the total number of edges between communities a and b and the sum of the node degrees in community a.

The fitness between nodes j and i is defined as

$$f_{ij} = \beta D_{\sigma_i \sigma_j} - (1-\beta) \ln k_i \text{ ,} \tag{13}$$

where $D_{\sigma_i \sigma_j}$ is the shortest path length between communities σ_i and σ_j in the supernetwork, k_i is the degree of node i, and $0 \le \beta \le 1$ is a free parameter. In order to maximize the overall performance of community embedding on different tasks, we test the effect of β for the tasks on some real-world networks, and set $\beta = 0.3$ for all experiments (see Ref. [39], Sec. III). Note that the fitness of Eq. (13) is an asymmetric function, i.e., $f_{ij} \neq f_{ji}$. In this paper we symmetrize it as

$$\bar{f}_{ij} = \frac{f_{ij} + f_{ji}}{2}, \tag{14}$$

and we treat it at the same footing as of a distance between nodes i and j, i.e.,

$$\text{dist}_{ij} = \bar{f}_{ij} \text{ ,} \tag{15}$$

even though $\bar{f}_{ij}$ is not a proper metric of distance.

B. Networks

In this paper, we use both real-world networks and synthetic networks. All networks are unweighted and undirected. We consider 72 real-world networks from different domains, including social, biological, technological, transportation, and Internet networks. Sizes of these networks ranges from 32 to 37,542 nodes. Figure 7 shows the average degree versus network size for all the 72 real-world networks used. Two networks with more than one million nodes are also considered for Node2vec and community embedding particularly to demonstrate their scalability. The full list of the real-world networks and some

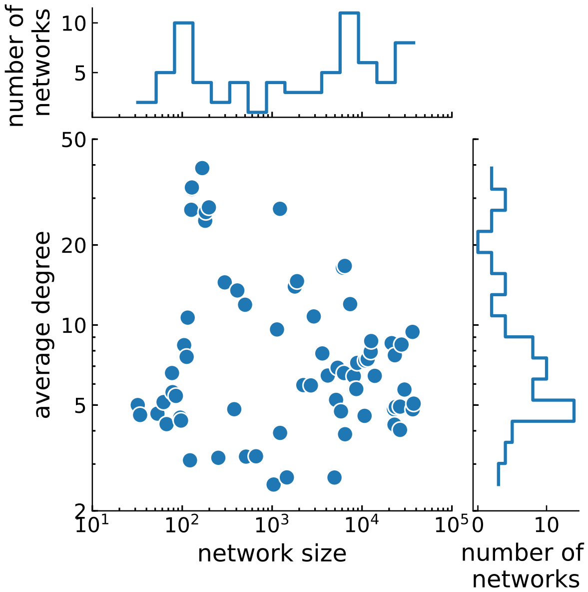

FIG. 7. **Summary statistics of the real-world networks considered in this study.** In the main panel, we show the scatter plot of the average degree $\langle k\rangle$ versus network size N. Each point is a real network in our dataset. Side panels are used to display non-normalized distributions of $\langle k\rangle$ and N.

of their basic information can be found in Ref. [39], Sec. I. Only the largest connected component of the various network is considered in our analysis.

We use 34 synthetic networks generated according to different models. We ensure that each network instance consists of one connected component only. The network models considered are reported below.

(1) *Popularity-similarity-optimization (PSO) model* [4, 19]: PSO model grows networks by adding nodes to a hidden hyperbolic space. Nodes close with each others in the hidden space are then connected to form the edges. There are several parameters that could affect the properties of the generated networks: network size N, temperature T, average degree $\langle k\rangle$, and exponent γ of the power-law degree distribution $P(k)\sim k^{-\gamma}$. Temperature $T\in[0,1)$ controls the average clustering in the network, which is maximized at $T=0$. We generate six instances of the PSO model with the following parameters: $N=\{1000;10,000\}$, $T=\{0.1;0.5;0.9\}$, $\gamma=2.1$, $\langle k\rangle=5$.

(2) *Lancichinetti-Fortunato-Radicchi (LFR) model* [76]: The LFR model generates networks with community structure, and both the degree distribution $P(k)$ and community size distribution $P(S)$ follow power-law distribution, i.e., $P(k)\sim k^{-\gamma}$ and $P(S)\sim S^{-\tau}$. Input parameters that are required to generate instances of the model are the network size N, the exponents γ and τ, the average degree $\langle k\rangle$, the maximum degree $k_{\max}$, the minimum and maximum community size $c_{\min}$ and $c_{\max}$, and the mixing parameter μ that determines how strong the community structure is. A small value of μ corresponds to a strong community structure. We generate eight instances of LFR models, the parameters are $N=\{1000;10,000\}$, $\mu=\{0.1;0.3;0.5;0.7\}$, $\gamma=2.1$, $\tau=2$, $\langle k\rangle=5$, $k_{\max}=50$, $c_{\min}=10$, $c_{\max}=0.1N$.

(3) *Poisson networks*: They are generated by feeding Poisson degree distributions to the configuration model. Two tunable parameters are the size of network N and average degree $\langle k\rangle$. We use eight instances of Poisson networks with the following parameters: $N=\{1000;10,000\}$, $\langle k\rangle=\{4;6;8;10\}$.

(4) *Power-law networks*: They are generated by feeding power-law degree distributions to the configuration model. The tunable parameters are the network size N and the power-law exponent γ. The average degree of a network can be controlled by setting the minimum value of the node degrees, namely $k_{\min}$. We use six instances of power-law networks and the parameters are $N=\{1000;10,000\}$, $\gamma=\{2.1;2.5;2.9\}$, $k_{\max}=100$. We use either $k_{\min}=2$ or $k_{\min}=3$ for nodes in the network to ensure an average degree $\langle k\rangle\simeq 5$.

(5) *Spatial networks* [77]: The model generate spatial networks that are embedded in two-dimensional regular lattice. Both the degree distribution $P(k)$ and the Euclidean distance distribution of edges $P(r)$ follow power-law distributions, i.e., $P(k)\sim k^{-\gamma}$ and $P(r)\sim r^{-\alpha}$. The input parameters of the model are the network size N, the exponents γ and α, the minimum and maximum degree $k_{\min}$ and $k_{\max}$. We use six instances of the model, with parameters chosen as $N=\{1000;10,000\}$, $\gamma=\{2.1;2.5;2.9\}$, $\alpha=2$, $k_{\max}=100$. We use either $k_{\min}=2$ or $k_{\min}=3$ for nodes in the network to ensure an average degree $\langle k\rangle\simeq 5$.

[1] M. Boguñá, I. Bonamassa, M. De Domenico, S. Havlin, D. Krioukov, and M. Á. Serrano, Nat. Rev. Phys , 1 (2021).
[2] W. L. Hamilton, R. Ying, and J. Leskovec, arXiv:1709.05584 (2017).
[3] P. Goyal and E. Ferrara, Knowl.-Based Syst. **151**, 78 (2018).
[4] F. Papadopoulos, M. Kitsak, M. Á. Serrano, M. Boguná, and D. Krioukov, Nature **489**, 537 (2012).
[5] M. Boguñá, D. Krioukov, and K. C. Claffy, Nat. Phys.

5, 74 (2009).

[6] A. Gulyás, J. J. Bíró, A. Kőrösi, G. Rétvári, and D. Krioukov, Nat. Commun. **6**, 7651 (2015).

[7] K.-K. Kleineberg, L. Buzna, F. Papadopoulos, M. Boguñá, and M. A. Serrano, Phys. Rev. Lett. **118**, 218301 (2017).

[8] S. Osat, F. Radicchi, and F. Papadopoulos, Phys. Rev. Research **2**, 023176 (2020).

[9] L. van der Maaten and G. Hinton, J. Mach. Learn. Res. **9**, 2579 (2008).

[10] D. Liben-Nowell and J. Kleinberg, J. Am. Soc. Inform. Sci. Tech. **58**, 1019 (2007).

[11] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon, in *Proceedings 2001 IEEE International Conference on Data Mining*, edited by N. Cercone, T. Y. Lin, and X. Wu (IEEE Computer Society, San Jose, California, USA, 2001) pp. 107–114.

[12] A. Tandon, A. Albeshri, V. Thayananthan, W. Alhalabi, F. Radicchi, and S. Fortunato, Phys. Rev. E **103**, 022316 (2021).

[13] F. Papadopoulos, R. Aldecoa, and D. Krioukov, Phys. Rev. E **92**, 022807 (2015).

[14] G. García-Pérez, A. Allard, M. Á. Serrano, and M. Boguñá, New J. Phys. **21**, 123033 (2019).

[15] A. Klimovskaia, D. Lopez-Paz, L. Bottou, and M. Nickel, Nat. Commun. **11**, 2966 (2020).

[16] M. Keller-Ressel and S. Nargang, J. Complex Netw. **8**, cnaa002 (2020).

[17] M. Kitsak, I. Voitalov, and D. Krioukov, Phys. Rev. Research **2**, 043113 (2020).

[18] A. Faqeeh, S. Osat, and F. Radicchi, Phys. Rev. Lett. **121**, 098301 (2018).

[19] F. Papadopoulos, C. Psomas, and D. Krioukov, IEEE/ACM Trans. Netw. **23**, 198211 (2015).

[20] A. Grover and J. Leskovec, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16 (ACM, New York, NY, USA, 2016) p. 855864.

[21] M. Belkin and P. Niyogi, in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01 (MIT Press, Cambridge, MA, USA, 2001) p. 585591.

[22] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '16 (ACM, New York, NY, USA, 2016) p. 11051114.

[23] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, Science **290**, 2319 (2000).

[24] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, J. Stat. Mech.: Theory Exp. **2008**, P10008 (2008).

[25] M. Rosvall and C. T. Bergstrom, Proc. Natl. Acad. Sci. USA **105**, 1118 (2008).

[26] E. Estrada, Phys. Rev. E **85**, 066122 (2012).

[27] E. Estrada, M. Snchez-Lirola, and J. A. de la Pea, Discret. Appl. Math. **176**, 53 (2014).

[28] D. Brockmann and D. Helbing, Science **342**, 1337 (2013).

[29] C. Hens, U. Harush, S. Haber, R. Cohen, and B. Barzel, Nat. Phys. **15**, 403 (2019).

[30] M. Boguñá, F. Papadopoulos, and D. Krioukov, Nat. Commun. **1**, 62 (2010).

[31] Y.-J. Zhang, K.-C. Yang, and F. Radicchi, Phys. Rev. E **103**, 012305 (2021).

[32] G. Sabidussi, Psychometrika **31**, 581 (1966).

[33] P. Bonacich, J. Math. Sociol. **2**, 113 (1972).

[34] W. Chen, Y. Wang, and S. Yang, in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09 (ACM, New York, NY, USA, 2009) p. 199208.

[35] L. C. Freeman, Sociometry **40**, 35 (1977).

[36] S. Brin and L. Page, Comput. Netw. ISDN Syst. **30**, 107 (1998).

[37] L. Katz, Psychometrika **18**, 39 (1953).

[38] M. Kitsak, L. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. Stanley, and H. Makse, Nat. Phys. **6**, 888 (2010).

[39] See Supplemental Material at [URL will be inserted by publisher] for a full list of the real-world networks we considered in this paper, the detailed results of all the embedding methods on the downstream tasks, and a fully explanation of the parameters selection for different embedding methods. The Supplemental Material includes Refs. [40–73].

[40] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon, Science **303**, 1538 (2004).

[41] W. W. Zachary, J. Anthropol. Res. **33**, 452 (1977).

[42] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, Behav. Ecol. Sociobiol. **54**, 396 (2003).

[43] D. E. Knuth, *The Stanford GraphBase: a platform for combinatorial computing*, Vol. 1 (ACM, New York, NY, USA, 1993).

[44] S. Mangan and U. Alon, Proc. Natl. Acad. Sci. USA **100**, 11980 (2003).

[45] L. A. Adamic and N. Glance, in *Proceedings of the 3rd International Workshop on Link Discovery*, LinkKDD '05 (ACM, New York, NY, USA, 2005) p. 3643.

[46] M. E. J. Newman, Phys. Rev. E **74**, 036104 (2006).

[47] M. Girvan and M. E. J. Newman, Proc. Natl. Acad. Sci. USA **99**, 7821 (2002).

[48] J. Fournet and A. Barrat, PLOS ONE **9**, 1 (2014).

[49] J. Kunegis, in *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13 Companion (ACM, New York, NY, USA, 2013) p. 13431350.

[50] R. Michalski, S. Palus, and P. Kazienko, in *Business Information Systems*, edited by W. Abramowicz (Springer Berlin Heidelberg, Berlin, Heidelberg, 2011) pp. 197–206.

[51] N. D. Martinez, Ecol. Monogr. **61**, 367 (1991).

[52] P. M. GLEISER and L. DANON, Adv. Complex Syst. **06**, 565 (2003).

[53] D. J. Watts and S. H. Strogatz, nature **393**, 440 (1998).

[54] L. Isella, J. Stehl, A. Barrat, C. Cattuto, J.-F. Pinton, and W. Van den Broeck, J. Theor. Biol. **271**, 166 (2011).

[55] V. Colizza, R. Pastor-Satorras, and A. Vespignani, Nat. Phys. **3**, 276 (2007).

[56] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, Science **298**, 824 (2002).

[57] L. Šubelj and M. Bajec, Eur. Phys. J. B **81**, 353 (2011).

[58] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas, Phys. Rev. E **68**, 065103 (2003).

[59] H. Jeong, S. P. Mason, A.-L. Barabási, and Z. N. Oltvai, Nature **411**, 41 (2001).

[60] T. Opsahl and P. Panzarasa, Soc. Netw. **31**, 155 (2009).

[61] D. Bu, Y. Zhao, L. Cai, H. Xue, X. Zhu, H. Lu, J. Zhang, S. Sun, L. Ling, N. Zhang, G. Li, and R. Chen, Nucleic Acids Res. **31**, 2443 (2003).

[62] T. Opsahl, F. Agneessens, and J. Skvoretz, Soc. Netw.

32, 245 (2010).

[63] R. Guimerà, S. Mossa, A. Turtschi, and L. A. N. Amaral, Proc. Natl. Acad. Sci. USA **102**, 7794 (2005).

[64] J. Leskovec, J. Kleinberg, and C. Faloutsos, ACM Trans. Knowl. Discov. Data **1**, 2es (2007).

[65] M. Newman, "The structure of scientific collaboration networks," in *The Structure and Dynamics of Networks* (Princeton University Press, 2011) pp. 221–226.

[66] L. Šubelj and M. Bajec, in *Proceedings of the First International Workshop on Software Mining*, SoftwareMining '12 (ACM, New York, NY, USA, 2012) p. 916.

[67] M. Ripeanu and I. T. Foster, in *Peer-to-Peer Systems*, edited by P. Druschel, M. F. Kaashoek, and A. I. T. Rowstron (Springer, Berlin, Heidelberg, 2002) pp. 85–93.

[68] J. Leskovec, J. M. Kleinberg, and C. Faloutsos, in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, edited by R. Grossman, R. J. Bayardo, and K. P. Bennett (ACM, New York, NY, 2005) pp. 177–187.

[69] M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas, Phys. Rev. E **70**, 056122 (2004).

[70] M. Ley, in *String Processing and Information Retrieval*, edited by A. H. F. Laender and A. L. Oliveira (Springer Berlin Heidelberg, Berlin, Heidelberg, 2002) pp. 1–10.

[71] L. Šubelj and M. Bajec, in *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13 Companion (Association for Computing Machinery, New York, NY, USA, 2013) p. 527530.

[72] M. De Choudhury, H. Sundaram, A. John, and D. D. Seligmann, in *Proceedings of the 2009 International Conference on Computational Science and Engineering*, CSE '09, Vol. 4 (IEEE Computer Society, San Jose, California, USA, 2009) pp. 151–158.

[73] J. Yang and J. Leskovec, Knowl. Inf. Syst. **42**, 181 (2015).

[74] J. M. Kleinberg, Nature **406**, 845 (2000).

[75] A. Muscoloni, J. M. Thomas, S. Ciucci, G. Bianconi, and C. V. Cannistraci, Nat. Commun. **8**, 1615 (2017).

[76] A. Lancichinetti, S. Fortunato, and F. Radicchi, Phys. Rev. E **78**, 046110 (2008).

[77] L. Daqing, K. Kosmidis, A. Bunde, and S. Havlin, Nat. Phys. **7**, 481 (2011).

[78] K.-K. Kleineberg, M. Boguná, M. Á. Serrano, and F. Papadopoulos, Nat. Phys. **12**, 1076 (2016).

[79] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, J. Mach. Learn. Res. **12**, 2825 (2011).

[80] W. Gu, A. Tandon, Y.-Y. Ahn, and F. Radicchi, Nat. Commun. **12**, 3772 (2021).

[81] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications*, Springer Series in Statistics (Springer, New York, 2005).

[82] A. D. Broido and A. Clauset, Nat. Commun. **10**, 1017 (2019).

[83] M. A. Serrano, D. Krioukov, and M. Boguñá, Phys. Rev. Lett. **100**, 078701 (2008).

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