Community detection in networks using graph embeddings

Aditya Tandon,[1] Aiiad Albeshri [ORCID],[2] Vijey Thayananthan,[2] Wadee Alhalabi [ORCID],[2] Filippo Radicchi,[3,1] and Santo Fortunato [ORCID][3,1]

[1]*Luddy School of Informatics, Computing and Engineering, Indiana University, Bloomington, Indiana 47408, USA*

[2]*Department of Computer Science, Faculty of Computing and Information Technology King Abdulaziz University, Jeddah 21589, Kingdom of Saudi Arabia*

[3]*Indiana University Network Science Institute (IUNI), Bloomington, Indiana 47408, USA*

Graph embedding methods are becoming increasingly popular in the machine learning community, where they are widely used for tasks such as node classification and link prediction. Embedding graphs in geometric spaces should aid the identification of network communities as well because nodes in the same community should be projected close to each other in the geometric space, where they can be detected via standard data clustering algorithms. In this paper, we test the ability of several graph embedding techniques to detect communities on benchmark graphs. We compare their performance against that of traditional community detection algorithms. We find that the performance is comparable, if the parameters of the embedding techniques are suitably chosen. However, the optimal parameter set varies with the specific features of the benchmark graphs, like their size, whereas popular community detection algorithms do not require any parameter. So, it is not possible to indicate beforehand good parameter sets for the analysis of real networks. This finding, along with the high computational cost of embedding a network and grouping the points, suggests that, for community detection, current embedding techniques do not represent an improvement over network clustering algorithms.

I. INTRODUCTION

Community structure is a common feature of many complex systems that can be represented as networks. A community, or cluster, is traditionally conceived as a set of nodes having a substantially higher probability to be linked to each other than to the rest of the graph. Detecting communities is a classic task in network analysis, that helps to uncover important structural and functional information of networks [1–3]. It is an unsupervised classification problem, and as such it is ill defined. Nevertheless, a huge number of algorithms have been developed over the past two decades.

Most techniques rely on a characterization of the structural properties of communities, typically via the density of internal links. Another approach is *spectral clustering* [4], where the nodes of the network are projected in a k-dimensional Euclidean space by using the top (or bottom) k eigenvectors of graph matrices, like the adjacency matrix or, more frequently, the Laplacian. This way the network is *embedded* in a geometric space, and, if it has a pronounced community structure, clusters of nodes appear as groups of points which are close to each other, and well separated from the points in the other groups. Such concentrations of points can be identified via standard data clustering techniques.

In the last decade several approaches have been developed to embed graphs in high-dimensional geometric spaces, while preserving some of their properties [5]. Such embedding techniques have proven to be useful to solve important tasks with many potential applications, e.g., node classification, link prediction, graph visualization [6]. Node classification aims at determining the label of nodes based on other labeled nodes and the topology of the network. Link prediction aims at predicting missing links (e.g., because of incomplete data) or links that are likely to occur in the future.

However, it is unclear whether such embedding strategies make the identification of communities easier than by applying traditional network clustering algorithms. Here, we address this specific issue. We adopt a broad collection of embedding techniques to project in vector spaces artificial networks with planted communities. We use the Lancichinetti-Fortunato-Radicchi (LFR) benchmark graphs [7], which are regularly employed to test clustering algorithms. The resulting distributions of geometric points are divided into clusters by using the k-means algorithm [8].

Our analysis reveals that default values of the parameters for the embedding methods, recommended based on their optimal performance in other tasks, lead to comparable performance as traditional nonparametric community detection methods, though performance tends to degrade for larger networks. A careful exploration of the parameter space often allows superior performance than standard clustering algorithms, but the results are very sensitive to little variations in the parameters' values and we could not find specific sets of values leading to good solutions in most cases.

II. METHODS

A. LFR benchmark

The LFR benchmark is characterized by power-law distributions of degree and community size, reflecting the heterogeneity of these two variables in real graphs [7]. The parameters needed to generate the graphs are the number of nodes N, the exponents of the distributions of degree (τ_1)

and community size (τ_2), the average degree k, the maximum degree $k_{\max}$, the extremes of the range of community sizes $c_{\min}$ (lower) and $c_{\max}$ (upper), and the mixing parameter μ. The latter indicates how pronounced the community structure is. If $\mu = 0$ clusters are disjoint from each other, i.e., all links fall within communities, and are easily detectable. For $\mu = 1$ links fall solely between clusters, which are not communities in the traditional sense of cohesive subsets of nodes, albeit they might be still detectable with particular techniques [9]. Normally, the parameters are fixed at the onset, except μ which is varied to explore different strength of communities. For our tests we used two different sets of parameters, mostly to explore how the clustering performance of embedding techniques is affected by the network size. The two sets are as follows:

(i) $N = 1000$, $\tau_1 = 2$, $\tau_2 = 3$, $\bar{k} = 20$, $k_{\max} = 50$, $c_{\min} = 10$, $c_{\max} = 100$.

(ii) $N = 10\,000$, $\tau_1 = 2$, $\tau_2 = 3$, $\bar{k} = 20$, $k_{\max} = 200$, $c_{\min} = 10$, $c_{\max} = 1,000$.

The similarity between the planted partition of the benchmark and the one found by the clustering algorithm can be measured in various ways. We opted for the normalized mutual information (NMI) [10], a measure borrowed from information theory, which is regularly used in this type of tests. For each value of μ we generated 20 configurations of the benchmark and report the average value of the NMI for such set of configurations as a function of μ.

For a thorough assessment of clustering performance, we also carried out tests on the popular benchmark by Girvan and Newman [11], where communities have the same size and nodes have the same degree. The results are reported in Appendix A and confirm the conclusions of our work.

B. Data clustering

After obtaining the embedding, k-means clustering is adopted to group the points into clusters. k means is very popular in data clustering. To make sure our message is not strongly dependent on the choice of the specific clustering method, we used Gaussian mixture models [12] as well, which leads to similar results. k-means clustering minimizes the squared distance between each data point (corresponding to a network node) and its centroid, which is a virtual point representing its cluster.

In the high-dimensional spaces where the network is embedded, the distance between points becomes less and less useful and the concept of proximity may not be meaningful, as different notions of distances may select different neighbors for the same point. For instance, the ratio of the distances of the nearest and farthest neighbors to a given target in a high-dimensional space is almost one for a wide variety of data distributions and distance functions [13]. For this reason, in our calculations we have used both the standard k means, based on the *Euclidean distance*, as well as spherical k means [14], which uses the *spherical distance*. Given two points i and j identified by the vectors of coordinates $\mathbf{x}_i = (x_i^1, x_i^2, x_i^3, \ldots, x_i^{d-1}, x_i^d)$ and $\mathbf{x}_j = (x_j^1, x_j^2, x_j^3, \ldots, x_j^{d-1}, x_j^d)$, where d is the number of dimensions, the two distance metrics are defined as follows:

(i) *Euclidean distance:*

$$\text{dist}_E(i, j) = \sqrt{\sum_{l=1}^{d} \left(x_i^l - x_j^l\right)^2}. \tag{1}$$

(ii) *Spherical distance:*

$$\text{dist}_S(i, j) = 1 - \frac{\sum_{l=1}^{d} x_i^l x_j^l}{\sqrt{\sum_{l=1}^{d} \left(x_i^l\right)^2 \sum_{l=1}^{d} \left(x_j^l\right)^2}}. \tag{2}$$

Data clustering techniques typically require the knowledge of the number of clusters beforehand. Instead of inferring this number via some criterion, we feed the procedure with the correct number of clusters of the benchmark graphs. This way we will actually assess the optimal performance of the embedding techniques. This is a luxury that we do not have when analyzing real networks, for which the number of clusters is unknown.

To improve the quality of the clustering, for each network the procedure is run 100 times, and the partition corresponding to the minimum distance between points and their centroids is selected.

C. Community detection

We compare the cluster analysis via embedding algorithms with the performance of widely adopted methods for community detection. We specifically show the comparison with Infomap [15] and the Louvain algorithm [16], that are known to be especially accurate to identify the planted partition of LFR benchmark graphs [17]. Other methods have similar performances and we report them in Appendix B. Infomap is based on diffusion dynamics: if the graph has a pronounced community structure, a random walker will spend a lot of time within a community before finally finding a bridge taking it to another community. This way, the description of an infinitely long random walk can be reduced by using the same labels for nodes of different clusters, much like it is done in geographic maps for the names of towns belonging to different regions and states. The partition leading to the cheapest description of the random walk is the best, by construction. The method does not require any parameter. It has some shortcomings, like the inability to detect clusters smaller than a certain scale [18] and the tendency to split nonclique structures with large diameters, such as strings and lattices [19]. Nevertheless, it is frequently used in applications. The code was taken from the IGRAPH [20] library.

The Louvain algorithm is a fast greedy method to optimize the modularity of Newman and Girvan [21], a function that estimates the goodness of a partition of a graph in communities. It is a very popular technique but, like Infomap, it has important limitations, like the inability to detect clusters smaller than a certain scale [22]. This is why here we consider the partition derived in the first step of the algorithm (the one with the smallest clusters), rather than the one with the largest value of modularity, which gives poor performance [2]. We used the PYTHON-Louvain [23] package for NETWORKX.

[handwritten margin notes: "Use genetic algo's to determine params for embedding." and "From µ = 0.5 ish onwards, performance degrades."]

D. Genetic optimization

Several graph embedding algorithms have multiple free parameters that have to be chosen. In the case of a single free parameter, an exhaustive search can be done over the range of parameter values to pick the one giving the best clustering performance. However, for other algorithms, exhaustive search over all free parameters becomes computationally infeasible. To find the best set of parameters we use a simple genetic algorithm outlined in Ref. [24] and use the PYTHON package DEAP [25] to implement the algorithm. The parameters used for the optimization procedure are as follows:

(1) Number of individuals in the population = 50.

(2) Number of generations = 20.

(3) Mutation function: normal distribution with mean 0 and standard deviation 3.

(4) Mutation probability = 0.2.

(5) Probability of an individual being produced by crossover = 0.5.

III. RESULTS

In this section we present the results of our tests. Each of the following subsections reports the results for a different class of embedding techniques.

A. Matrix-based methods

The approach consists in projecting nodes onto a high-dimensional vector space via eigenvectors of graph matrices, which is the same principle of spectral clustering [4].

1. Laplacian eigenmap (LE)

The Laplacian eigenmap (LE) [26] minimizes, under some constraints, the objective function

$$E_{\mathrm{LE}} = \sum_{ij} |\mathbf{x_i} - \mathbf{x_j}|^2 A_{ij}, \tag{3}$$

where $\mathbf{x}_i$ is the vector indicating the position of the point representing node i in the embedding and $\mathbf{A}$ is the adjacency matrix of the graph, whose elements weigh the square distance between the corresponding points. Nodes that have a higher link weight will consequently be closer together in the embedding. The constraint $\mathbf{x}^T \mathbf{D}\mathbf{x} = 1$, where $\mathbf{D}$ is the diagonal matrix of the degrees of the graph nodes, is added to remove an arbitrary scaling factor in the embedding. If we wish an embedding in d dimensions, it can be shown that the desired one is obtained via the eigenvectors corresponding to the d lowest eigenvalues (except the zero eigenvalue) of the problem $\widetilde{\mathbf{L}}\mathbf{x} = \lambda \mathbf{D}\mathbf{x}$, where $\widetilde{\mathbf{L}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-\frac{1}{2}}$ is the normalized Laplacian matrix of the graph. The projection $\mathbf{x}_i$ is the vector whose components are the ith entries of the d eigenvectors. LE has a single parameter, the embedding dimension d, making it straightforward and easy to use. In Fig. 1 we show how LE performs for different values of d (i.e., 32, 128, 256) and the two k-means clustering procedures we have chosen. The panels correspond to the different sets of parameters of the LFR benchmark listed in Sec. II A. The performance of Infomap is generally superior, despite the fact that the latter has no parameters and ignores the number of
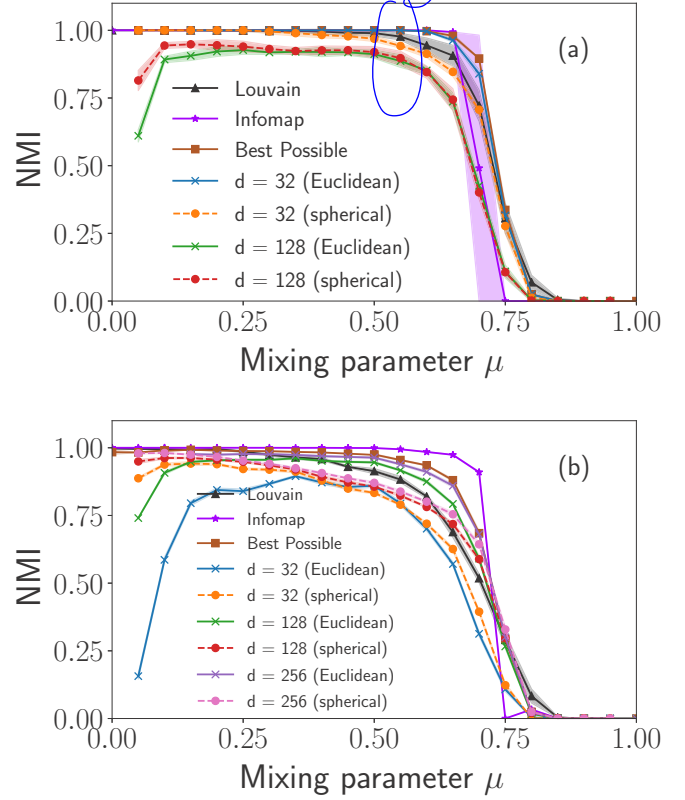

FIG. 1. Performance of Laplacian eigenmap on LFR networks of 1000 nodes (a) and 10 000 nodes (b). The NMI values are averaged over 10 realizations of the same LFR configuration for each μ. The baseline performances are those of Infomap (star) and Louvain (filled triangle). The shaded area of the Infomap curve in (a) reflects the large fluctuations of the NMI when the performance declines: here the partition found by the algorithm oscillates between one correlated with the planted one and the partition into one cluster, so the distribution of the NMI for those values of μ is bimodal (with peaks near 1 and 0), which gives a large variance. The spherical distance metric (bullet) performs better than Euclidean distance (**x**) in the regime of low μ and high embedding dimension d. The best possible performance (filled square) is found by doing an exhaustive search across 50 different values of d ranging from 2 to 500 for each network and selecting the one with the largest NMI.

clusters, which is derived by the algorithm itself. By doing an exhaustive optimization, for each single network (even when μ is the same) we have identified the value of d giving the best performance. The resulting performance curves are a bit better than those of Infomap on the smaller networks, while on the larger ones Infomap has still an edge. However, the optimal d value varies with μ. It is well known that, in spectral clustering, the ideal number of eigenvectors to use to obtain a good clustering typically matches the number of clusters to be found [4]. We find that this is mostly true here, but not always. Louvain has comparable performance as LE on LFR graphs with 1000 nodes, while its curve worsens on the larger graphs.

We remark that for low values of μ the performance is unexpectedly poor, given that communities are well separated from each other. It turns out that in these cases some nodes of
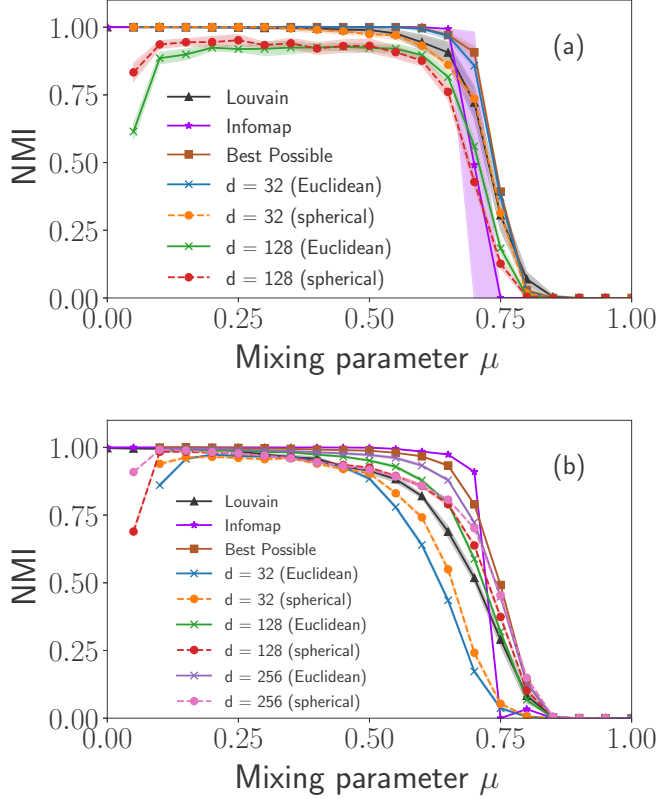
FIG. 2. Performance of locally linear embedding on LFR networks of 1000 nodes (a) and 10 000 nodes (b). The NMI values are averaged over 10 realizations of the same LFR configuration for each μ. The baseline performances are those of Infomap (star) and Louvain (filled triangle). The cosine distance metric (bullet) performs better than the Euclidean distance (x) in the regime of low μ and high embedding dimension (d). The best possible NMI (filled square) is found by doing an exhaustive search across 50 different values of d ranging from 2 to 500, for each network and selecting the one with the largest NMI.

different clusters are projected close to each other, making it hard to correctly classify them (see Appendix C).

2. Locally linear embedding (LLE)

Locally linear embedding (LLE) [27] minimizes the objective function

$$E_{\text{LLE}} = \sum_i \left| \mathbf{x_i} - \sum_j A_{ij} \mathbf{x_j} \right|^2, \qquad (4)$$

where each summand is the square distance between vectors. Each point in the embedded space $\mathbf{x_i}$ is approximated as a linear combination of its neighbors in the original graph. To make the problem well posed, the solutions are required to be centered at the origin, i.e., $\sum_i \mathbf{x_i} = 0$, and have unit variance, i.e., $\frac{1}{N}\mathbf{x}^T\mathbf{x} = I$. With these constraints the solution is approximated by the eigenvectors corresponding to the lowest eigenvalues (disregarding the zero eigenvalue) of the matrix $\mathbf{M} = (\mathbf{I} - \mathbf{A})^T(\mathbf{I} - \mathbf{A})$. Like LE, LLE too has a single parameter, the embedding dimension. The results are in Fig. 2, where we again consider up to three values for the
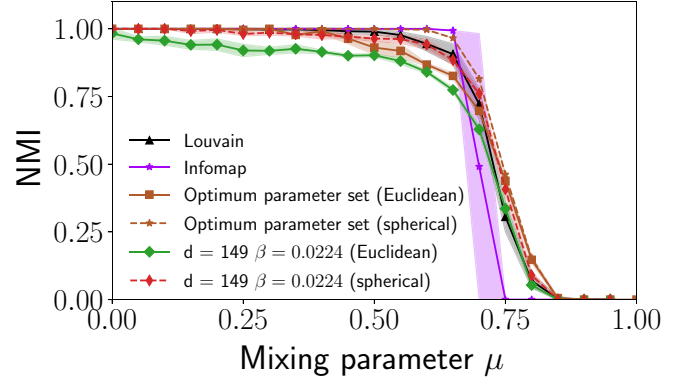

FIG. 3. Performance of higher-order preserving embedding on LFR networks with 1000 nodes. The baseline performances are those of Infomap (star) and Louvain (filled triangle). For each value of the mixing parameter μ, the average NMI over 10 configurations is reported. The best parameters are found by employing a genetic optimization in two different ways: finding the best parameters for each configuration separately (brown) and finding the optimum parameter set across all values of μ (green and red, the optimal parameters are indicated) by optimizing the area under the curve. The spherical distance metric (dashed) performs better than the Euclidean metric (solid lines).

number of dimensions (i.e., 32, 128, 256) and derive the best performance curve by identifying the best d value for each benchmark graph via exhaustive optimization. The conclusion is similar as for LE: LLE generally underperforms Infomap. The optimal performance curve is better but the values of the best number of dimensions vary with the network. Louvain is comparable to the best LLE curve for $N = 1000$, while it is worse for $N = 10\,000$.

3. Higher-order preserving embedding (HOPE)

Higher-order preserving embedding (HOPE) [28] aims to preserve the similarity between nodes. In its general formulation, the function that is optimized is $||\mathbf{S} - \mathbf{xx}^T||$ where $\mathbf{S}$ is the similarity matrix and the matrix distance is the sum of the squares of the differences between the corresponding matrix elements. Although the authors try several different similarity metrics, we found *Katz similarity* [29] to be the best performing one and used it in our experiments here. Katz similarity is defined as

$$\mathbf{S}^{\text{Katz}} = \beta \sum_{l=1}^{\infty} \mathbf{A}^l, \qquad (5)$$

where $\beta < 1$ is a decay parameter. The other parameter d is the embedding dimension.

Figure 3 shows the results of the analysis, where we only considered the smaller LFR graphs, with 1000 nodes, as experiments on the larger networks have a high computational cost. The colored curves correspond to the optimal performance, which is obtained by using the genetic algorithm described in Sec. II D. We followed two different approaches: (1) optimization of the parameters for each individual network; (2) optimization of the parameters for the whole curve, so that their values are fixed for every μ and graph. This is to
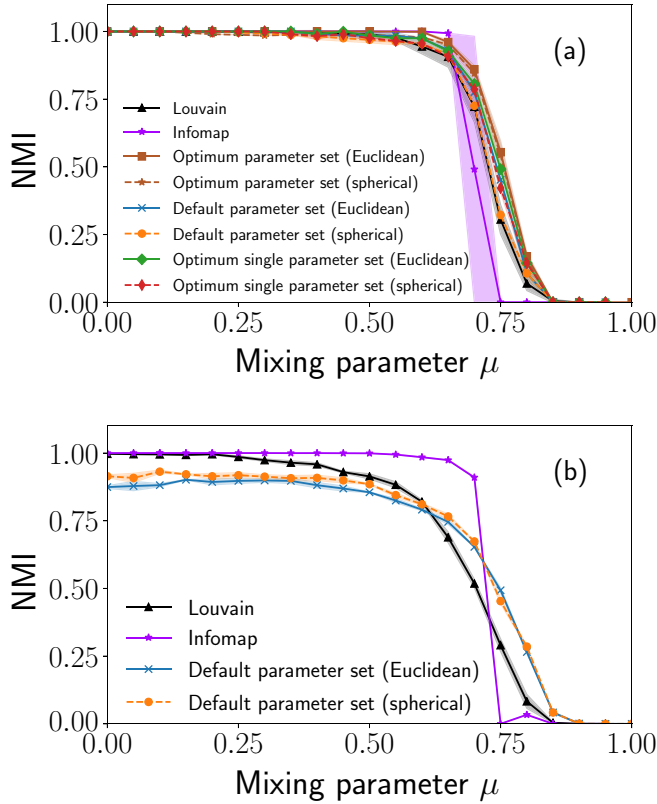
FIG. 4. Performance of modularized non-negative matrix factorization for LFR benchmark graphs of 1000 nodes (a) and 10 000 nodes (b). The baseline performances are those of Infomap (star) and Louvain (filled triangle). The performance obtained by using the default parameters (blue and orange) is almost as good as the best performance obtained by doing an evolutionary optimization over the parameters. The latter can be done optimizing NMI for each graph configuration separately (brown) or optimizing the area under the curve (green and red). For graphs of 10 000 nodes the default parameters lead to a poor performance. There is no significant difference between the two different metrics (Euclidean and spherical).

FIG. 5. Performance of DeepWalk on LFR benchmark graphs of 1000 nodes (a) and 10 000 nodes (b). The baseline performances are those of Infomap (star) and Louvain (filled triangle). The performance of the algorithm using the default parameter set is better if we adopt the Euclidean distance (blue, solid) instead of the spherical distance (orange, dashed). The default parameter performance is not significantly worse than the optimum performance found by doing an evolutionary optimization over each configuration (filled square) or a single parameter set across all configurations (star). For each value of μ, 10 different LFR graphs are generated and the NMI value shown is the average across them.

check whether we can recommend specific pairs of values for clustering purposes. Also, we considered both k-means algorithms. By construction, the graph-based optimization offers a superior performance than the optimization based on the whole curve, though the difference is not big.

The Infomap curve is better than all optimal ones until $\mu \sim 0.7$, then it undershoots them for larger μ, though in that region the overall performance is quite poor because communities are well blended with each other. Louvain is comparable to both best performance curves.

4. Modularized non-negative matrix factorization (M-NMF)

Modularized non-negative matrix factorization (M-NMF) [30] incorporates modularity [31] into the optimization function. Modularity is a function expressing how good a partition in communities is, based on the comparison between the network and randomized versions of it, which are supposed not to have community structure. The minimization of the optimization function guarantees that nodes are projected near each other if they are similar and, at the same time, if they belong to clusters of high-modularity
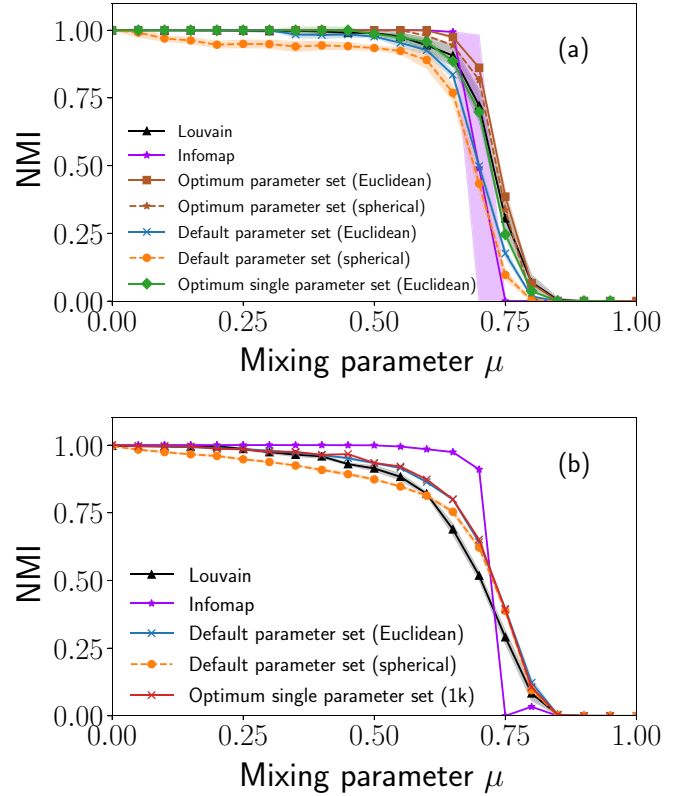
partitions. The similarity between two nodes is the cosine similarity of vectors whose entries are the overlaps between the neighborhoods of the nodes. Although the original paper does not mention a default set of parameters, we use the following ones for the experiments in this paper: dimension $d = 128$, $\lambda = 0.2$, $\alpha = 0.05$, $\beta = 0.05$, $\eta = 5.0$, number of iterations $N_i = 200$.

The results are in Fig. 4. In Fig. 4(a) we show the curve obtained with the default parameters above and the ones obtained by optimizing the NMI for each individual graph and for the whole range of μ values. The optimization is carried out with the procedure described in Sec. II D. All curves are better than those of Infomap. However, when we use the default parameters for the larger LFR graphs [Fig. 4(b)], the performance degrades considerably and Infomap does much better. On such graphs we could not derive the optimal curves. Due to the high number of parameters, the optimization is very costly, computationally, so we only report the curve with the default parameters. Louvain offers closer performances on both network sizes, though it is clearly superior for
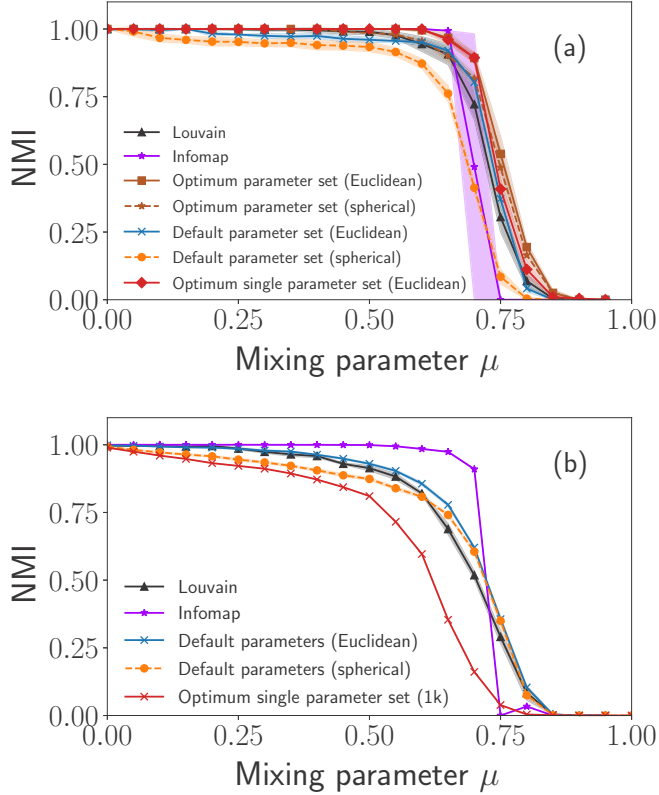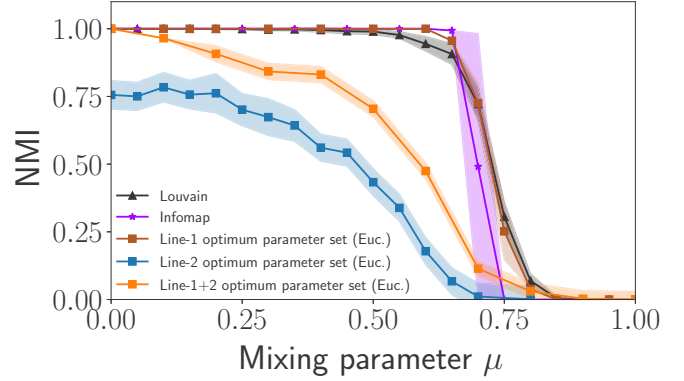
FIG. 7. Performance of large-scale information network embedding on LFR benchmark graphs for 1000 nodes. The baseline performances are those of Infomap (star) and Louvain (filled triangle). The curves show the optimal performance of the three implementations of the method. The first implementation, LINE-1, is clearly the best and outperforms Infomap, albeit by a small margin.

FIG. 6. Performance of node2vec on LFR benchmark graphs of 1000 nodes (a) and 10 000 nodes (b). The baseline performances are those of Infomap (star) and Louvain (filled triangle). The performance of the algorithm using the default parameter set is better if we adopt the Euclidean distance (blue, solid) instead of the spherical distance (orange, dashed). The default parameter performance is not significantly worse than the optimum performance found by doing an evolutionary optimization over each configuration (filled square) or a single parameter set across all configurations (star). For each value of μ, 10 different LFR graphs are generated and the NMI value shown is the average across them.

$N = 10\,000$ and low μ values, where it is capable to detect the correct partition, while MNMF fails to do so consistently.

B. Random walk embeddings

In this section we present tests carried out by using embeddings that rely on random walks performed on the graph. We chose the two most popular techniques of this class, described below.

1. DeepWalk

DeepWalk [32] applies language modeling techniques from deep learning on graphs instead of words and sentences. The algorithm uses local information obtained from truncated random walks to learn latent representations by treating walks as the equivalent of sentences in the word2vec [33] language model. The default parameters used for the experiments here are dimension $d = 128$, window $w = 10$, random walk length $t = 40$, number of walks per node $n = 80$.

In Fig. 5 we show the performance curves for the default parameters for both standard (Euclidean) and spherical k means, as well as the optimal curves obtained by maximizing the NMI for each single LFR graph and using a single parameter set for the whole range of μ values. The curves are similar: the default parameter curves are worse than those of Infomap and Louvain, while the optimal performance when the parameters are adjusted for each single network is superior. For the larger LFR graphs (bottom panel) the performance optimization is very expensive, so we only report the curves obtained by using the default parameters and the optimal parameter set found on the smaller graphs. The latter closely follows the curve of the default parameters (Euclidean distance). Infomap is clearly superior here, while Louvain has comparable performance as DeepWalk's default curves.

2. node2vec

node2vec [34] uses the same optimization procedure as DeepWalk, but the process to generate the "sentences" is different. Instead of using simple random walks as in DeepWalk, node2vec uses biased random walks. The walk consists of a mixture of steps following breadth-first and depth-first search, with parameters p and q regulating the relative weights of the two approaches. The default parameters used for the experiments here are dimension $d = 128$, window $w = 10$, random walk length $t = 10$, number of walks per node $n = 80$, biased walk weights $p = 1$ and $q = 1$.

In Fig. 6 we show the performance curves for the default parameters for both standard (Euclidean) and spherical k means, as well as the optimal curves obtained by maximizing the NMI for each single LFR graph and using a single parameter set for the whole range of μ values. For the default parameters the results are very similar as for DeepWalk since for those parameters the two techniques are equivalent (Fig. 5). For the larger LFR graphs we include the curve with the optimal single parameter set found for the smaller networks, which is significantly worse than the default parameter curves in this case. This shows that the optimal parameter set is strongly dependent of the network size, along with other features.

TABLE I. Comparison of the performances of Infomap and the embedding-based clustering algorithms we have adopted in our tests, for LFR benchmark graphs of 1000 nodes. Performance is estimated by computing the area under the curve (AUC). We started computing the AUC from $\mu = 0.1$ because in some cases we do not have results for lower μ values. Each value in the table is the higher score between the one obtained using the Euclidean distance and the one obtained using the spherical distance. The AUC of the best traditional community detection method is indicated in boldface.

	Area under curve		
Algorithm	Default parameter set	Optimum parameter set	Optimum single parameter set
Infomap	0.60		
Louvain	**0.62**		
Laplacian eigenmap		0.64	
LLE		0.64	
HOPE		0.64	0.62
M-NMF	0.63	0.65	0.64
DeepWalk	0.59	0.64	0.61
node2vec	0.62	0.66	0.64
LINE		0.62	

C. Large-scale information network embedding (LINE)

Large-scale information network embedding (LINE) [35] projects nodes the closer to each other the higher their similarity. It considers both first order similarity, based on whether nodes are adjacent or not, and second order similarity, based on the overlap of the neighborhoods of two nodes.

There are two different ways to obtain the embeddings, denoted LINE-1 and LINE-2. The authors also suggest concatenating the two embeddings to obtain a $2d$ dimensional embedding which we denote by LINE-1 + 2.

In Fig. 7 we show the results for the LFR graphs with 1000 nodes. The method is very slow so we could not produce the analogous curve for benchmark graphs with 10 000 nodes. For each of the three different implementations of the method we have derived the optimum parameters via the genetic algorithm of Sec. II D, for each LFR network. For LINE-1 the optimal performance exceeds those of Infomap and Louvain, though the curves are very close, the other two implementations are far worse. We do not show the curves corresponding to the default parameters because they are very poor.

In Tables I and II we summarize our comparative analysis by reporting the performances of the various techniques, measured via the area under the curve.

TABLE II. Same as Table I, but for LFR benchmark graphs of 10 000 nodes. The AUC of the best traditional community detection method is indicated in boldface.

	Area under curve	
Algorithm	Default parameter set	Optimum parameter set
Infomap	**0.62**	
Louvain	0.58	
Laplacian eigenmap		0.66
LLE		0.64
M-NMF	0.57	
DeepWalk	0.60	
node2vec	0.59	

IV. CONCLUSIONS

We have evaluated the performance of graph clustering techniques mediated by embeddings of networks in high-dimensional vector spaces. The identification of the clusters in the vector space is done via data clustering methods, specifically k means. Overall we found that the several embedding strategies we have adopted do not help to resolve the community structure of LFR benchmark graphs better than the best performing community detection algorithms, Infomap and Louvain, which act directly on the network, without any embedding. The parameters of the embedding procedure can be optimized such to get close or even outperform the curve of those algorithms. However, the optimal parameters generally vary with the mixing parameter μ, so we could not come up with a single parameter set that we can recommend for clustering applications. Besides, the optimal parameter values are affected by the network size as well, so there would not be a "one-size-fits-all" parameter set. This means that, for a given real network, we cannot know which parameters are best to reveal its modular structure, which results in noisy partitions. Finally, the combination of embedding techniques plus data clustering is a computationally expensive procedure. While some embedding algorithms can scale up to very large graphs, data clustering techniques (like k means) typically scale superlinearly with the graph size. As a result, the full procedure is much more computationally demanding than fast graph clustering methods, like Infomap and Louvain.

We acknowledge that the embedding is just a component of the overall clustering algorithm, and that the performances we observe might be due to the data clustering approach used to group the points into clusters. We have used k means and Gaussian mixture models (results not shown), which are regularly adopted for data clustering, without finding significant differences. Still, the number of clusters, which is usually unknown in practice, needs to be specified as input, and we set it equal to the correct value for the planted partition. By doing that we have significantly helped the performance, whereas many network clustering algorithms (including Infomap and Louvain) are able to infer the number of clusters. This means that the curves we have shown in our plots are better, in
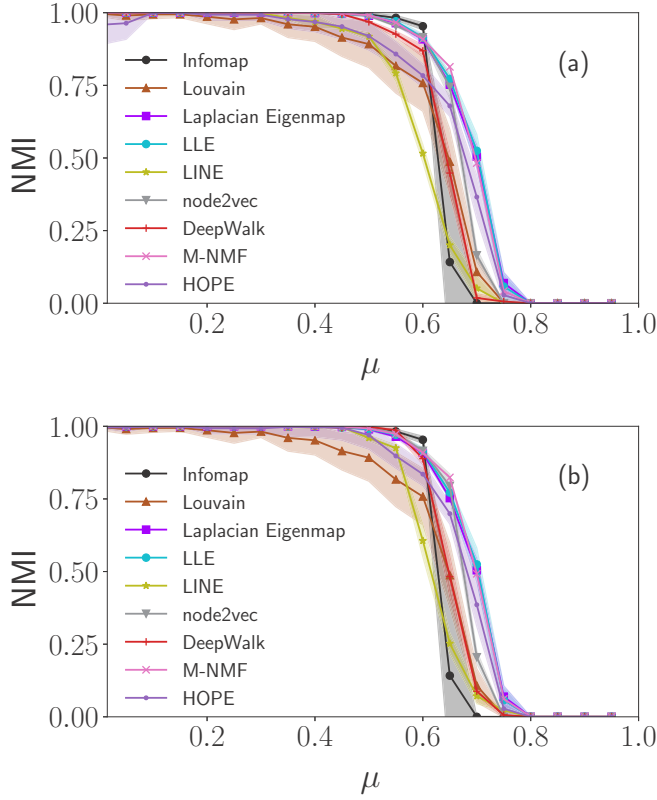
FIG. 8. Performance of all the considered embedding methods on a stochastic block model (SBM) of 20 groups of 50 nodes each. The average degree is set to 20. The mixing parameter μ is reported on the x axis, for each μ value 10 graphs are generated and the average NMI over them is shown on the y axis. (a) Curves corresponding to the default parameters. (b) Best curves obtained via the genetic optimization over the parameters (Sec. II D).

general, than the ones we would obtain if the number of clusters were inferred via some criterion. We also recognize that the concept of distance becomes problematic in high-dimensional spaces and work is in progress to alleviate the drawbacks deriving from that.

Embedding techniques have not been designed to tackle specifically the clustering task, so it is not surprising that they do not excel in this task. To improve their clustering performance embedding strategies that focus on preserving the modular structure of networks should be developed. Spectral-based embeddings seem particularly promising in that regard because of the provable optimal performance of spectral clustering in synthetic graphs with communities built with stochastic block models [9] and because they provide ways to estimate the number of clusters [36–40].
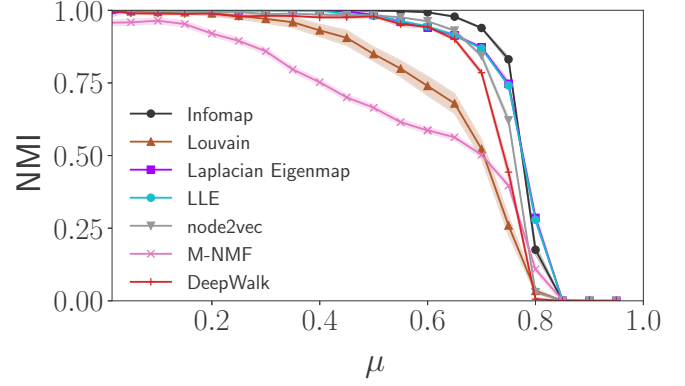
FIG. 9. Performance of all the considered embedding methods on a stochastic block model (SBM) of 200 groups of 50 nodes each. The average degree is set to 20. The mixing parameter μ is reported on the x axis, for each μ value 10 graphs are generated and the average NMI over them is shown on the y axis for the default parameters.

APPENDIX A: COMPARATIVE ANALYSIS OF EMBEDDING METHODS ON SBM's

We also performed tests on networks generated by the classic stochastic block model (SBM) [9] (Fig. 8). The SBM is a model of graphs with built-in community structure. The probability to form a link between two nodes only depends on the groups the nodes belong to. The LFR benchmark is a special case of SBM. We considered graphs with 20 groups with 50 nodes each, the average degree is 20. The mixing parameter μ is again the ratio between the external degree of a node and its total degree. The best parameters of the embedding methods are chosen after genetic optimization in the manner outlined in Sec. II D: the optimum NMI curves are plotted in Fig. 8(b). Like for the LFR benchmark tests, the performance curves of Infomap and Louvain are shown for comparison. Embedding techniques do a bit better there (except LINE). In Fig. 9 we show the performance curves of the embedding clustering methods for graphs with 10 000 nodes, with 200 groups of 50 nodes each. Average degree is still 20. For LE and LLE the number of dimensions, their
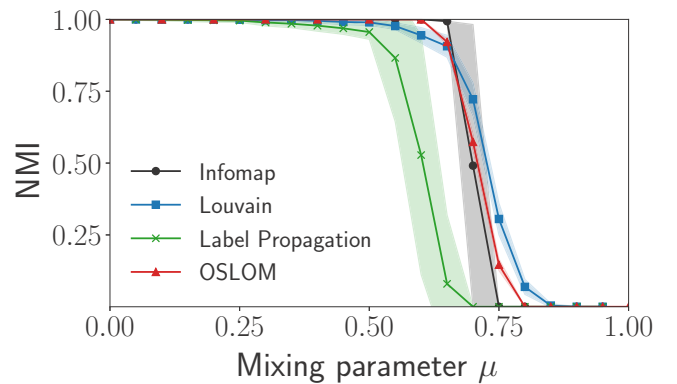

FIG. 10. A comparison of traditional community detection methods: Infomap, Louvain, Label propagation, and OSLOM on the LFR benchmark. The average degree, $k = 20$, and number of nodes $N = 1000$. The additional parameters for the LFR benchmark are $\tau_1 = 2$, $\tau_2 = 3$, $k_{max} = 50$, $c_{min} = 10$, and $c_{max} = 100$.
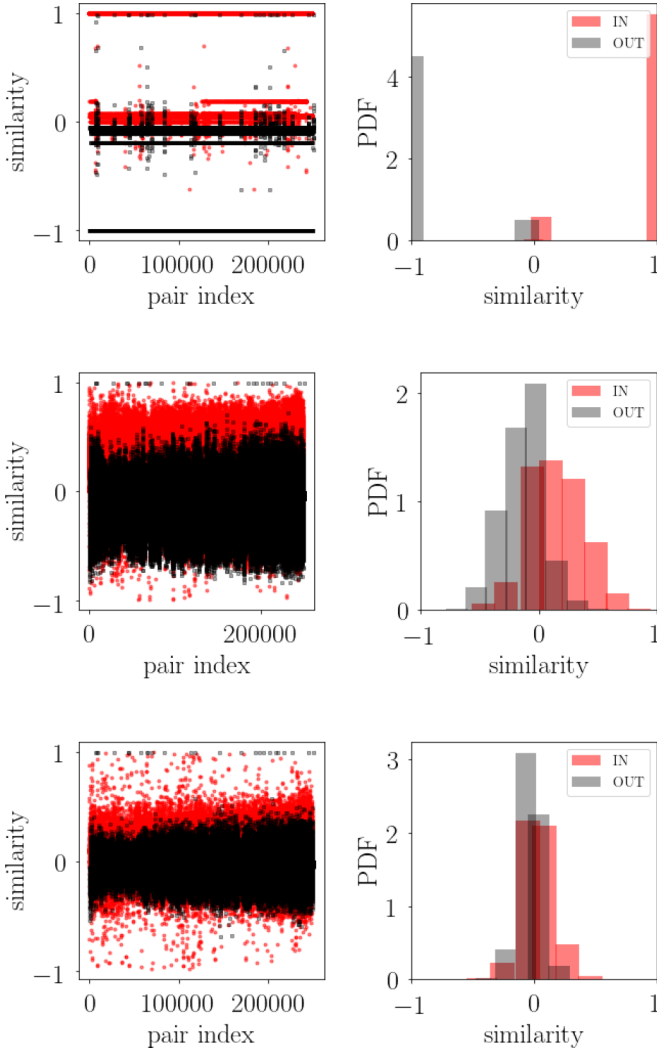
FIG. 11. Problem of spectral embeddings for well-separated communities. The network is generated by an SBM with two blocks, $p_{in} = 1$ and $p_{out} = 0.0001$, and $N = 500$ in each block. Laplacian eigenmap is applied for dimensions $d = 20, 50, 100$ (top to bottom). The cosine similarity for within cluster node pairs overlaps with that of the node pairs across clusters for higher embedding dimension, making the identification of the clusters hard even for this straightforward example.

FIG. 12. Problem of spectral embeddings for well-separated communities. The network is generated by an SBM with two blocks, $p_{in} = 1$ and $p_{out} = 0.001$, and $N = 500$ in each block. Laplacian eigenmap is applied for dimensions $d = 20, 50, 100$ (top to bottom). The cosine similarity for within cluster node pairs has limited overlap with that of the node pairs across clusters for higher embedding dimension, making the identification of the clusters easier.

only parameter, is set to the number of clusters, 200. We see that the embedding methods' performance worsens a bit here, while Infomap improves and outperforms all of them. Louvain's performance degrades considerably. We remind that the correct number of clusters is fed into the embedding-based methods, in contrast to the standard clustering techniques, which are capable to guess it. This confers a major advantage to embedding-based clustering methods over the traditional ones.

APPENDIX B: COMPARISON OF TRADITIONAL COMMUNITY DETECTION METHODS

Here, we compare the performance of four clustering algorithms on the LFR benchmark. Aside from Infomap and Louvain the other algorithms are:
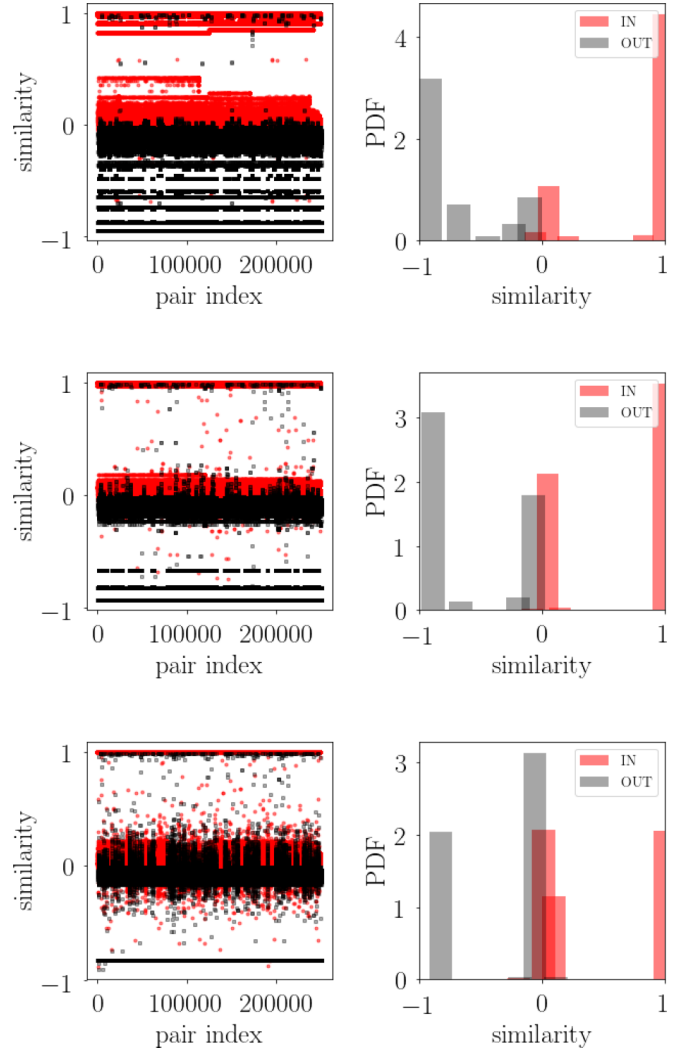
(i) The order statistics local optimization method (OSLOM) [41] seeks statistically significant clusters in networks. We use the OSLOM code [42], with default parameters.

(ii) The label propagation algorithm (LPA) [43]. It is a technique that assigns nodes to the community to which the majority of its neighbors belong to. The code was taken from the IGRAPH library.

Fig. 10 shows that Infomap, Louvain, and OSLOM have comparable performance, whereas label propagation is worse. We used Infomap and Louvain in the tests of the main text as representative of well-performing algorithms on the LFR benchmarks, and for no other reason.
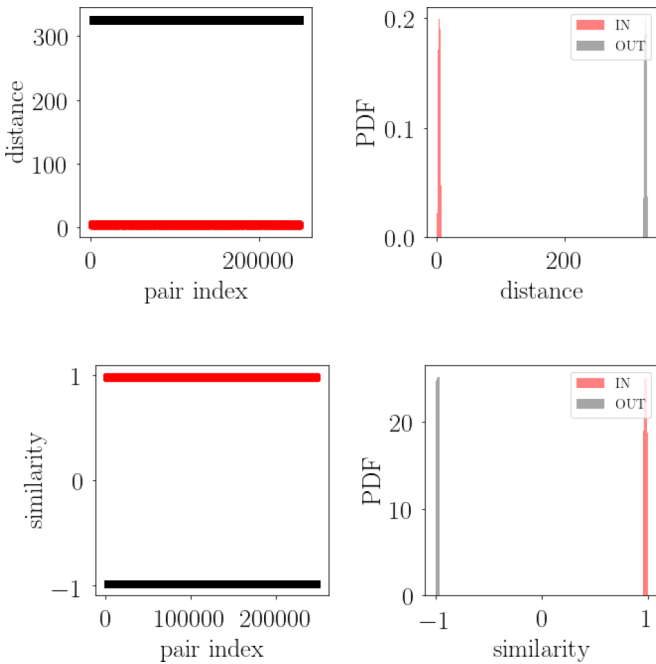
FIG. 13. Improving Laplacian eigenmap. The network is generated by the same SBM as in Fig. 11. A modified version of Laplacian eigenmap for $d = 100$ is applied, in that the eigenvector components are multiplied by the (inverse) eigenvalue. In the first row we show analogous diagrams as in Figs. 11 and 12, for the Euclidean distance (top row) and the cosine similarity (bottom row). The clusters are now well separated.

APPENDIX C: FAILURE OF SPECTRAL METHODS AT HIGH DIMENSIONS FOR $\mu \to 0$

In Figs. 1 and 2 we have seen that the performance for spectral embedding methods like Laplacian eigenmap and locally linear embedding is not very good for low values of μ. This is against intuition, as clusters there are well

separated from each other and fairly easy to resolve. To check what happens we have considered a relatively simple graph with two communities, built via the stochastic block model (SBM) [9]. Here, we consider only two communities, with 500 nodes each. The probability of having links within the groups is $p_{in} = 1$, the probability of having links between the groups is $p_{\text{out}} = 0.0001$, so that the communities are well separated from each other. In this example we expect that the embedding generates two sets of points well separated from each other. In Fig. 11 we see what happens if we do the embedding of this network using Laplacian eigenmap for different numbers of dimensions: $d = 20, 50, 100$. Each row of plots refers to a value of d (growing from top to bottom). For each d, we compute the cosine similarity of all pairs of nodes, computed via their respective vectors. Red and black (gray) indicate pairs in the same versus different groups. The expectation is that node pairs in the same cluster have higher similarity than pairs in different clusters. The left plot shows the actual similarity values for each pair and indeed we see that pairs of nodes in the same community are very similar, while pairs of nodes in different communities are very dissimilar. This is further indicated by the right diagram, showing the probability distribution of the similarities. We see that, as d increases, the difference between within-community and between-community pairs reduces. For $d = 100$ pairs of nodes have very low similarity, regardless of their group memberships, which makes it difficult to separate the nodes. This is due to the peculiar behavior of the eigenvectors of the Laplacian when the clusters are almost disjoint. On the other hand, in Fig. 12, we consider an SBM like that in Fig. 11, but with $p_{in} = 1$ and $p_{\text{out}} = 0.001$. Now we see that the two groups can be identified even at high dimensions. In the special case of Laplacian eigenmap, we can modify the embedding strategy by multiplying the eigenvector components by the inverse of the corresponding eigenvalue. By doing that we see that the two groups are clearly separated (Fig. 13). This trick, however, cannot be easily extended to other embedding strategies relying on graph spectra.

[1] S. Fortunato, Community detection in graphs, Phys. Rep. **486**, 75 (2010).

[2] S. Fortunato and D. Hric, Community detection in networks: A user guide, Phys. Rep. **659**, 1 (2016), community detection in networks: A user guide.

[3] M. A. Porter, J.-P. Onnela, and P. J. Mucha, Communities in networks, Not. Amer. Math. Soc. **56**, 1082 (2009).

[4] U. von Luxburg, A tutorial on spectral clustering, Technical Report No. 149, Max Planck Institute for Biological Cybernetics, 2006 (unpublished).

[5] P. Goyal and E. Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowledge-Based Syst. **151**, 78 (2018).

[6] M. Pereda and E. Estrada, Visualization and machine learning analysis of complex networks in hyperspherical space, Pattern Recognition **86**, 320 (2019).

[7] A. Lancichinetti, S. Fortunato, and F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E **78**, 046110 (2008).

[8] J. B. MacQueen, Some methods for classification and analysis of multivariate observations, in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, edited by L. M. Le Cam and J. Neyman (University of California Press, Berkeley, 1967), pp. 281–297.

[9] T. P. Peixoto, Bayesian stochastic blockmodeling, in *Advances in Network Clustering and Blockmodeling*, edited by Patrick Doreian, Vladimir Batagelj, and Anuska Ferligoj (Wiley, Oxford, UK, 2020), pp. 289–332.

[10] L. N. F. Ana and A. K. Jain, Robust data clustering, in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003 Proceedings*, Vol. 2 (IEEE Madison, WI, 2003), pp. II–II.

[11] M. Girvan and M. E. Newman, Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA **99**, 7821 (2002).

[12] D. A. Reynolds, Gaussian mixture models, Encycl. Biometrics **741**, 659 (2009).

[13] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, When is "nearest neighbor" meaningful? in *International Conference on Database Theory* (Springer, Berlin, 1999), pp. 217–235.

[14] I. S. Dhillon and D. S. Modha, Concept decompositions for large sparse text data using clustering, Mach. Learn. **42**, 143 (2001).

[15] M. Rosvall and C. T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proc. Natl. Acad. Sci. USA **105**, 1118 (2008).

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

[17] A. Lancichinetti and S. Fortunato, Community detection algorithms: A comparative analysis, Phys. Rev. E **80**, 056117 (2009).

[18] T. Kawamoto and M. Rosvall, Estimating the resolution limit of the map equation in community detection, Phys. Rev. E **91**, 012809 (2015).

[19] M. T. Schaub, R. Lambiotte, and M. Barahona, Encoding dynamics for multiscale community detection: Markov time sweeping for the map equation, Phys. Rev. E **86**, 026112 (2012).

[20] http://igraph.org/.

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

[22] S. Fortunato and M. Barthélemy, Resolution limit in community detection, Proc. Natl. Acad. Sci. USA **104**, 36 (2007).

[23] https://github.com/taynaud/python-louvain.

[24] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary Computation 1: Basic Algorithms and Operators* (CRC Press, Boca Raton, FL, 2018), Chap. 7.

[25] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner Gardner, M. Parizeau, and C. Gagné, DEAP: Evolutionary algorithms made easy, J. Mach. Learn. Res. **13**, 2171 (2012).

[26] M. Belkin and P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. **15**, 1373 (2003).

[27] S. T. Roweis and L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science **290**, 2323 (2000).

[28] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, Asymmetric transitivity preserving graph embedding, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, New York, 2016), pp. 1105–1114.

[29] L. Katz, A new status index derived from sociometric analysis, Psychometrika **18**, 39 (1953).

[30] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, Community preserving network embedding, in *Thirty-first AAAI Conference on Artificial Intelligence* (AAAI, Menlo Park, CA, 2017).

[31] M. E. J. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. USA **103**, 8577 (2006).

[32] B. Perozzi, R. Al-Rfou, and S. Skiena, Deepwalk: Online learning of social representations, in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, New York, 2014) pp. 701–710.

[33] T. Mikolov, K. Chen, G. Corrado, and J. Dean, Efficient estimation of word representations in vector space, preprint arXiv:1301.3781.

[34] A. Grover and J. Leskovec, node2vec: Scalable feature learning for networks, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, New York, 2016), pp. 855–864.

[35] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, Line: Large-scale information network embedding, in *Proceedings of the 24th International Conference on World Wide Web* (ACM, New York, 2015), pp. 1067–1077.

[36] T. Qin and K. Rohe, Regularized spectral clustering under the degree-corrected stochastic blockmodel, in *Advances in Neural Information Processing Systems* (Curran Associates, Red Hook, NY, 2013), pp. 3120–3128.

[37] Y. Zhao, E. Levina, and Ji Zhu, Consistency of community detection in networks under degree-corrected stochastic block models, Ann. Stat. **40**, 2266 (2012).

[38] A. Saade, F. Krzakala, and L. Zdeborová, Spectral clustering of graphs with the bethe hessian, in *Advances in Neural Information Processing Systems* (Curran Associates, Red Hook, NY, 2014), pp. 406–414.

[39] L. Dall'Amico, R. Couillet, and N. Tremblay, Revisiting the bethe-hessian: improved community detection in sparse heterogeneous graphs, in *Advances in Neural Information Processing Systems* (Curran Associates, Red Hook, NY, 2019), pp. 4037–4047.

[40] F. Krzakala, C. Moore, E. Mossel, J. Neeman, A. Sly, L. Zdeborová, and P. Zhang, Spectral redemption in clustering sparse networks, Proc. Natl. Acad. Sci. USA **110**, 20935 (2013).

[41] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, Finding statistically significant communities in networks, PLoS One **6**, e18961 (2011).

[42] Available at http://oslom.org/.

[43] U. N. Raghavan, R. Albert, and S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E **76**, 036106 (2007).