Network community detection and clustering with random walks

Aditya Ballal ,[1,2] Willow B. Kion-Crosby ,[3] and Alexandre V. Morozov [1,2,*]

[1]*Department of Physics and Astronomy, Rutgers University, Piscataway, New Jersey 08854, USA*

[2]*Center for Quantitative Biology, Rutgers University, Piscataway, New Jersey 08854, USA*

[3]*Helmholtz Institute for RNA-based Infection Research, Würzburg 97080, Germany*

(Received 14 March 2022; accepted 26 August 2022; published 18 November 2022)

We present an approach to partitioning network nodes into nonoverlapping communities, a key step in revealing network modularity and functional organization. Our methodology, applicable to networks with weighted or unweighted symmetric edges, uses random walks to explore neighboring nodes in the same community. The walk-likelihood algorithm (WLA) produces an optimal partition of network nodes into a given number of communities. The walk-likelihood community finder employs WLA to predict both the optimal number of communities and the corresponding network partition. We have extensively benchmarked both algorithms, finding that they outperform or match other methods in terms of the modularity of predicted partitions and the number of links between communities. Making use of the computational efficiency of our approach, we investigated a large-scale map of roads and intersections in the state of Colorado. Our clustering yielded geographically sensible boundaries between neighboring communities.

DOI: 10.1103/PhysRevResearch.4.043117

I. INTRODUCTION

Many complex systems in human society, science, and technology can be represented by networks, a set of N vertices linked by edges [1–4]. Examples include the Internet, the World Wide Web, transportation networks, food webs, social networks, and biochemical and genetic networks in biology. These complex networks often contain distinct groups, with more edges between nodes within the same group than between nodes belonging to different groups. Detecting such distinct groups of nodes, called network communities, has attracted considerable attention in the literature [5–25]; see [26–28] for comprehensive reviews. Parsing complex networks into communities provides useful information about the underlying structure of the network and may provide insights into the common function of each group of nodes. For example, in gene coexpression networks different communities represent different gene modules, with genes in the same module acting together to carry out high-level biological functions such as stress response [29]. Protein-protein interaction networks are also characterized by pronounced modularity which may have been shaped by adaptive evolution [30]. In the context of social networks, communities represent groups of people with similar interests and behavioral patterns.

Despite clear intuition behind the network community concept, precise mathematical definitions of network communities are somewhat elusive, with many empirical metrics proposed in the literature. These metrics include the modularity score, the internal edge density, and the cut ratio [8,31,32]. Interestingly, many of the empirical scores can be understood in terms of random walks on graphs, which reveal the stability of graph communities across timescales [19]. The modularity score plays a central role in several algorithms for network community detection [12–14,33]. Commonly used network community detection methods include Edge Betweenness [5], Fastgreedy [33], Infomap [16,17], Label Propagation [13], Leading Eigenvector [12], Multilevel [14], Spinglass [7], and Walktrap [11]. These methods were benchmarked for computational efficiency and prediction accuracy by Yang *et al.* using an extensive set of artificially generated networks [34]. Other algorithms, benchmarks, community metrics, and current issues in network community detection are discussed in recent reviews [27,28]. Several of these methods rely on random walks for network community detection and scoring [17,19,20,25].

Network community detection is conceptually similar to clustering and data dimensionality reduction, which have a long history of development in machine learning and artificial intelligence communities [35]. Some of the state-of-the-art approaches to data clustering and visualization are rooted in the ideas borrowed from random walks and diffusion theory. These include non-negative matrix factorization (NMF), a clustering method originally developed to provide decompositions into interpretable features in visual recognition and text analysis tasks [36–39], spectral clustering, an algorithm closely related to NMF and based on analyzing the eigenvalues and eigenvectors of the graph Laplacian [40,41], and a dimensionality reduction technique based on diffusion maps which uses random walks to project data points into a lower-dimensional space [42–44].

[*]morozov@physics.rutgers.edu

Here we propose two methods for clustering and network community detection. The first method, which we call the walk-likelihood algorithm (WLA), leverages information provided by random walks to produce a partition of data points or network nodes into m nonoverlapping communities, where the number of communities is known *a priori*. Unlike previous algorithms that employ random walks and diffusion in network community detection [11,12,17,19,20,25], dimensionality reduction [42–44], and spectral clustering [40,41], our approach is based on Bayesian inference of network properties as the network is explored by random walks [45]. Here, the property of interest is the posterior probability for each node to belong to one of the m network communities. Although Bayesian inference has long been used in the prediction of network modules, it was typically employed in the context of specific network-generating processes such as the stochastic block model [15,18,23]. In contrast, here we consider all network nodes and edge weights as given and partition the network into communities on the basis of the number of visits to network nodes by an ensemble of random walks. Intuitively, we utilize the fact that a random walk will, on average, visit the nodes in its own community more frequently than the nodes in all other communities [19].

WLA is used as the main ingredient in our second method, walk-likelihood community finder (WLCF), which predicts the optimal number of clusters (or network communities) m_{opt} using global moves that involve bifurcation and merging of communities, and employs WLA to refine node community assignments at each step. We have subjected both WLA and WLCF to extensive testing on artificial networks against several of the state-of-the-art algorithms mentioned above. After checking their performance and algorithmic complexity, we have applied WLA and WLCF to several real-world networks, including a large-scale network of roads and intersections in the Colorado state.

II. WALK-LIKELIHOOD ALGORITHM

Let us consider a network with N nodes labeled $n = 1 \dots N$. The network edges are defined by $\tilde{A}_{N\times N}$, a symmetric adjacency matrix with $\tilde{A}_{n'n} = 1$ if there is an edge between nodes n and n', and $\tilde{A}_{n'n} = 0$ everywhere else. We focus on undirected networks with weighted edges: $\{w_{ij}\}$, where $w_{ij} = w_{ji}$ is the weight of the edge between nodes i and j. We define $w_i = \sum_{k\in nn(i)} w_{ik}$ as the connectivity or weighted degree of node i, where the sum is over all of its nearest neighbors. Let $A_{N\times N}$ be the transition matrix of the network, where $A_{n'n} = P(n \to n')$ is the probability of jumping from node n to node n' in a single step (see Methods for details). We define an indicator matrix $U_{N\times m}$ to partition the network into m communities labeled by $c = 1 \dots m$, such that each element $U_{nc} = 1$ if and only if $n \in c$, and 0 otherwise. The weighted size of each community c can then be computed as $\mathcal{W}_c = \sum_{n=1}^{N} w_n U_{nc}$, where w_n is the weighted degree of node n. Next, we define a matrix $V_{N\times m}$ such that

$$V_{nc} = \sum_{l=1}^{l_{\max}} \left(\sum_{n'=1}^{N} (A^l)_{nn'} w_{n'} U_{n'c} \right). \tag{1}$$

Note that $V_{nc}/\mathcal{W}_c$ is the expected number of times, *per random walk*, that the node n is visited by random walks with $l_{\max}$ steps that start from nodes n' in community c, where the nodes $n' \in c$ are chosen randomly with the probability $P(n') = w_{n'}/\mathcal{W}_c$. The node n does not have to be in the same community c as the nodes n', although the expected number of visits to the node n should be higher if this is the case. Furthermore, the *expected* number of visits to the node n per random walk given by Eq. (1) corresponds to the *actual* number of visits that would be observed when the total number of random walks with $l_{\max}$ steps that originate from community c, G_c, is very large: $G_c \to \infty$. Then the total number of visits to the node n by G_c random walks originating in the community c is given by $\tilde{V}_{nc} = G_c V_{nc}/\mathcal{W}_c$.

With the stochastic process defined by Eq. (1), a single random walk originating in the community c' visits all nodes in the community c for $\tilde{\ell}_{c'c}$ steps on average:

$$\tilde{\ell}_{c'c} = \sum_{k\in c} \frac{V_{kc'}}{\mathcal{W}_{c'}} = \frac{(V^T U)_{c'c}}{\mathcal{W}_{c'}}. \tag{2}$$

Correspondingly, the total number of steps within the community c (equal to the total number of visits to the nodes in community c) of $G_{c'}$ random walks that originated in the community c' is given by $\ell_{c'c} = G_{c'}\tilde{\ell}_{c'c} = (\tilde{V}^T U)_{c'c}$. The total Poisson rate parameter for visiting the node $n \in c$ by $G_{c'}$ random walks is given by the sum of the Poisson rate parameters for each individual random walk:

$$G_{c'} \frac{w_n \tilde{\ell}_{c'c}}{\mathcal{W}_c} = \frac{w_n \ell_{c'c}}{\mathcal{W}_c}. \tag{3}$$

Therefore, for multiple random walks originating from the same community c', the Poisson probabilities in Eq. (18) are generalized to $\mathcal{P}(\tilde{V}_{nc'}, w_n \ell_{c'c}/\mathcal{W}_c)$ [$\mathcal{P}(\mathcal{K}, \lambda)$ is the Poisson distribution with the number of events $\mathcal{K}$ and the rate parameter λ], where all random walks that start from the nodes in the community c' contribute to a single Poisson process with the total rate given by Eq. (3). Finally, the community identity of the node n can be inferred probabilistically using a straightforward generalization of Eq. (19):

$$\begin{aligned} &P\big(n \in c | \{\tilde{V}_{nc'}\}_{c'=1}^m, \{\ell_{c'c}\}_{c'=1}^m\big) \\ &\quad = \frac{1}{\mathcal{Z}} \Pr(n \in c) \prod_{c'=1}^{m} \mathcal{P}\left(\tilde{V}_{nc'}, \frac{w_n \ell_{c'c}}{\mathcal{W}_c}\right), \end{aligned} \tag{4}$$

where $\Pr(n \in c)$ is the prior probability that $n \in c$ and $\mathcal{Z}$ is the normalization constant. Omitting the conditional dependencies for simplicity, Eq. (4) can be rewritten as:

$$\begin{aligned} \log P(n \in c) = &\sum_{c'=1}^{m} \frac{G_{c'}}{\mathcal{W}_{c'}} (V_{nc'} \log Q_{c'c} - Q_{c'c} w_n) \\ &+ \log \Pr(n \in c) + H(n) - \log \mathcal{Z}, \end{aligned} \tag{5}$$

where

$$Q_{c'c} = \frac{\ell_{c'c} \mathcal{W}_{c'}}{G_{c'} \mathcal{W}_c} = \frac{\tilde{\ell}_{c'c} \mathcal{W}_{c'}}{\mathcal{W}_c} = \frac{(V^T U)_{c'c}}{\mathcal{W}_c} \tag{6}$$

and

$$H(n) = \sum_{c'=1}^{m} \left\{ \frac{G_{c'}}{\mathcal{W}_{c'}} \log \left(\frac{G_{c'}}{\mathcal{W}_{c'}} \right) - \log \left[\left(\frac{G_{c'} V_{nc'}}{\mathcal{W}_{c'}} \right)! \right] \right\} \quad (7)$$

is independent of the community index. Note that according to Eq. (6), $Q_{cc} = \tilde{\ell}_{cc}$, the average number of steps within the community c of a single random walk that originated in the same community.

We find it convenient to parametrize G_c as $G_c = sg_c$, with $s \to \infty$ and finite relative weights g_c (the choice of g_c is discussed below). Then Eq. (4) can be written as

$$P(n \in c) = \lim_{s \to \infty} \frac{e^{sF_{nc} + \log \Pr(n \in c)}}{\sum_{c'=1}^{m} e^{sF_{nc'} + \log \Pr(n \in c')}}, \quad (8)$$

where F_{nc} is given by Eq. (5):

$$F_{nc} = \sum_{c'=1}^{m} \frac{g_{c'}}{\mathcal{W}_{c'}} [V_{nc'} \log Q_{c'c} - Q_{c'c} w_n]. \quad (9)$$

In the $s \to \infty$ limit, the effect of the priors is negligible and the sum in the denominator of Eq. (8) is dominated by a single term with the largest $F_{nc'}$, so that Eq. (8) simplifies to

$$P(n \in c) = \delta_{c\tilde{c}} \text{ for } \tilde{c} = \text{argmax}_{c''} F_{nc''}. \quad (10)$$

Equation (10) allows us to reassign community identities for each node n. These community identities are then used to construct the updated matrix U for the next iteration of the algorithm.

A. Choice of g_c

The relative weights g_c determine the fraction of random walks that start from community c. To remove community-dependent sampling biases, we set g_c so that the mean number of visits to node $n \in c$ from all random walks starting in the community c is independent of its parameters. Note that according to Eq. (17), the mean number of visits to a node $n \in c$ is $\ell_{cc} w_n / \mathcal{W}_c = sg_c Q_{cc} w_n / \mathcal{W}_c$. Thus, setting $g_c = \mathcal{W}_c / Q_{cc}$ ensures that the mean number of visits is sw_n, which is independent of the community index c and depends only on the connectivity of node n. Note that under this scheme, the total Poisson rate parameter for the visits to the node $n \in c$ by all random walks originating in the community c' is given by $sw_n(Q_{c'c}/Q_{c'c'})$.

B. Convergence criterion

To determine how similar the updated assignment of nodes to communities is to the previous one, we use the normalized mutual information (NMI) [46] between the current partition U and the previous partition U' [Eq. (21) in Methods]. We terminate the iterative node reassignment process if the NMI between partitions obtained in subsequent iterations is greater than 0.99.

The iterative node reassignment procedure can be summarized as follows:

WALK-LIKELIHOOD ALGORITHM

```
INPUT:
Network with $N$ nodes
$A_{N\times N}$: Transition matrix of the network
$m$: Number of network communities
$l_{\max}$: Number of random walk steps
$w_n$: Connectivity of each node $n = 1 \ldots N$
$U'_{N\times m}$: Initial guess of the partition of the
  network into $m$ communities
do:
  1. $V_{nc} \leftarrow \sum_{l=1}^{l_{\max}} (\sum_{n'=1}^{N} (A^l)_{nn'} w_{n'} U'_{n'c})$          Eq. (1)
  2. $Q_{cc'} \leftarrow (V^T U')_{cc'} / \sum_{n=1}^{N} w_n U'_{nc'}$          Eq. (6)
  3. $F_{nc} \leftarrow \sum_{c'=1}^{m} Q_{c'c'}^{-1} [V_{nc'} \log Q_{c'c} - Q_{c'c} w_n]$          Eq. (9) with
                                                    $g_c = \mathcal{W}_c / Q_{cc}$
  4. $U_{nc} \leftarrow \delta_{\tilde{c}_n c}$ for $\tilde{c}_n = \text{argmax}_{c''} F_{nc''}$          Eq. (10)
  5. Compute NMI$(U, U')$          Eq. (21)
  6. $U' \leftarrow U$
while not converged [NMI$(U, U') \leqslant 0.99$]
OUTPUT: $U_{N\times m}$: Optimal partition of the network into $m$
        communities
```

III. WALK-LIKELIHOOD COMMUNITY FINDER

Using the walk-likelihood algorithm (WLA) described above, we have developed the walk-likelihood community finder (WLCF), an algorithm for partitioning a network into communities when the number of communities is not known *a priori*. We initialize the WLCF algorithm by assuming that whole network is a single community. The flowchart of the algorithm is shown in Fig. 1, with each major step explained in detail below:

Outer loop:

I. Bifurcation. We divide each network community randomly into two communities. This is illustrated in Fig. 1, panel I, where community C_1' bifurcates into communities C_1 and C_2, and community C_2' bifurcates into communities C_3 and C_4. Note that this step divides the network into two communities at the start of the algorithm.

II. Inner loop. The inner loop consists of three consecutive steps. The loop is terminated if step 2 conditions are not met.

1. Walk-likelihood algorithm: The walk-likelihood algorithm is run to obtain a more accurate partition of the network (Fig. 1, panel II). Note that the number of communities m does not change in this step.

2. Criteria for merging communities: To check if the current division of the network into m communities is optimal, we compute modularity scores [8] for all m communities. Then, for $\binom{m}{2}$ pairs of communities, we check if combining any pair of communities c and c' increases the modularity score of the partition. The change in the modularity score after merging communities c and c' is given by

$$\Delta M_{cc'} = 2(e_{cc'} - a_c a_{c'}), \quad (11)$$

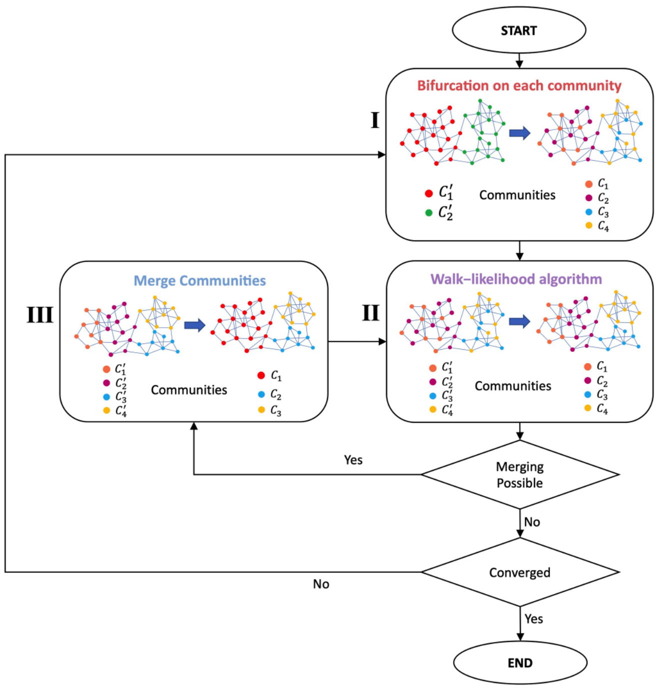

FIG. 1. The flowchart of the WLCF algorithm. The key steps of the algorithm include random community bifurcation in the beginning of the outer loop iteration (panel I); application of the walk-likelihood algorithm (panel II); merging communities on the basis of the changes in the modularity score (panel III).

where $e_{cc'} = (U^T\tilde{A}U)_{cc'}/\sum_{n=1}^{N} w_n$ and $a_c = \sum_{n=1}^{N} U_{nc}w_n/\sum_{n=1}^{N} w_n$ ($\tilde{A}$ is the network adjacency matrix). Note that these definitions generalize the modularity score [Eq. (13) in Methods] to networks with weighted edges. If there exists at least one pair of communities such that $\Delta M_{cc'} > 0$, we proceed to step 3 of the inner loop where one pair of communities is merged, otherwise we exit the inner loop.

3. Merging communities: If step 2 conditions are met, we merge the pair of communities c_1 and c_2 with the largest increase in the modularity score $M_{c_1c_2}$. This is illustrated in Fig. 1, panel III, where communities C_1' and C_2' merge to form C_1.

III. Convergence criteria. The outer loop is terminated if the number of communities in the partitions obtained in subsequent iterations of the outer loop remains constant and the NMI between the communities in the current and the previous partitions is greater than 0.99. The algorithm also stops if the modularity score of the partition decreases by more than 0.01 in subsequent iterations, or if the maximum number of iterations has been reached.

Elimination of spurious bifurcation-merge cycles. The WLCF algorithm can get into a loop where a community c is bifurcated into c_1 and c_2 in step I and then c_1 and c_2 merge again in step 3 of the inner loop (step II of the outer loop) to form the same community c. This indicates that community c cannot be bifurcated any further. In order to avoid such bifurcation-merge cycles, we check if there are any matches between the communities in the current partition and those in the previous partition, by calculating the following score:

$$E_{cc'} = 1 - \frac{2\sum_{i=1}^{N} U_{ic}U'_{ic'}}{\sum_{i=1}^{N}(U_{ic} + U'_{ic'})} \qquad (12)$$

between the communities c and c' of the current partition (U) and the previous partition (U'), respectively. If $E_{cc'} < 0.01$, we assume that the communities c and c' are the same and conclude that further bifurcations of the community c are not possible. Thus, all communities c of the current partition for which there exists a corresponding community c' in the previous partition such that $E_{cc'} < 0.01$, are not bifurcated in the subsequent iteration of the WLCF algorithm (step I of the outer loop).

IV. SYNTHETIC NETWORKS

To test the performance of WLA and WLCF algorithms in a controlled setting using realistic networks with tunable properties, we have generated a comprehensive set of Lancichinetti, Fortunato, and Radicchi (LFR) benchmark graphs [47]. The LFR benchmark was specifically created to provide a challenging test for community detection algorithms. It was recently used to test many state-of-the-art algorithms in a rigorous comparative analysis [34]. Similar to real-world networks, LFR networks are characterized by power-law distributions of the node degree and community size. Each node in a given LFR network has a fixed mixing parameter $\mu = \sum_{i=1}^{N} k_i^{\text{ext}}/\sum_{i=1}^{N} k_i$, where k_i^{ext} is the number of links between node i and nodes in all other communities and k_i is the total number of links of node i. Thus, every node shares a fraction $1-\mu$ of its links with the other nodes in its community and a fraction μ with the rest of the network [47]. Note that $\mu = 0$ corresponds to the communities that are completely isolated from one another, while $\mu < \frac{1}{2}$ results in well-defined communities in which each node has more connections with the nodes in its own community than with the rest of the graph. Generally speaking, network communities become more difficult to detect as μ increases.

The parameters of the networks in our LFR benchmark set are summarized in the Supplemental Material [48] (Table S1). These parameters were chosen to enable direct comparisons with the large-scale evaluation of community detection algorithms carried out by Yang *et al.* [34]. In order to investigate algorithm performance on larger networks, we have also added graphs with $N = 5 \times 10^4$ and $N = 10^5$ to our implementation of the LFR benchmark. For each value of N, we have created networks with 25 different mixing parameters μ ranging from 0.03 to 0.75. For each value of N and μ, 20 independent network realizations were created for networks with $N = 5 \times 10^4$ and $N = 10^5$; for all smaller networks, 10^2 independent network realizations were created. We used the Github package LFR-BENCHMARK_UNDIRWEIGHTOVP by eXascale Infolab [49].

First, we have used a single realization of the LFR network with $\mu = 0.15$ and $N = 10^3$ to study the effects of $l_{\max}$ on the network exploration (see Supplemental Material [48], Fig. S1). Conceptually similar to diffusion maps [42–44], the value of $l_{\max}$ is qualitatively related to the scale of the network structures explored by random walks: lower $l_{\max}$ values create a bias towards local exploration, while higher $l_{\max}$ values enable global exploration of the entire network and transitions between communities. The natural upper cutoff for $l_{\max}$ is the network diameter, which is often $\sim \log N$ in scale-free, real-world networks [2–4]. Indeed, we observe that small $l_{\max}$ values lead to more visits to nodes in the same community as the starting node (compared to nodes in all other communities) as local network neighborhoods are explored (see Supplemental Material [48], Fig. S1). However, the exploration is noisy since many nodes cannot be reached by short random walks, even if they belong to the same community. As $l_{\max}$ increases, the difference between visiting nodes in the two categories decreases, but the uncertainties in the number of visits decrease at the same time. For very large $l_{\max}$ values, the whole network is explored.

Next, we have tested how $l_{\max}$ affects the performance of the WLA algorithm on the entire LFR benchmark (see Supplemental Material [48], Fig. S2). WLA was provided with the exact number of communities $m^\star$ as input, and NMF-based clustering was used to initialize it. We observe that while $l_{\max} = 1$ is suboptimal, all other values produce very similar results. Therefore, WLA does not appear to be overly sensitive with respect to this parameter. Nonetheless, it would be advisable to start with a value of $l_{\max}$ guided by the network diameter ($\sim \log N$ in scale-free networks) and explore a few values around it. Very sparse networks with long chains of nodes and few crosslinks are likely to require larger values of $l_{\max}$. Overall, we conclude that either using intermediate values of $l_{\max}$ or alternating between intermediate and low values should lead to reasonable performance. As with hyperparameter settings in many other algorithms, finding an acceptable range of $l_{\max}$ values may require some numerical experimentation.

Finally, we discuss the choice of WLA prior probabilities for partitioning networks into communities. The choice of the prior does not explicitly affect node assignments to communities in each WLA iteration, due to the $s \to \infty$ limit in Eq. (10). Nonetheless, cluster initialization affects WLA's numerical performance because self-consistent iterations generally depend on initial conditions. The least informative choice is the uniform prior, implemented as random assignment of all network nodes into m communities with equal probabilities $1/m$. However, in many cases nonrandom initialization with the help of existing algorithms such as NMF or NNDSVD (see Methods) may be preferable. If the scaling exponent β for the power-law distribution of community sizes is available, this information can also be used to initialize WLA with more realistic communities. As expected, using the knowledge of the distribution of cluster sizes generally leads to a boost in the WLA performance on the LFR benchmark (see Supplemental Material [48], Fig. S3).

We have carried out an extensive comparison of the WLCF and WLA algorithms with four other state-of-the-art community network detection and clustering methods (Fig. 2). Two methods, Multilevel [14] and Label Propagation [13], were chosen because they were recommended by the previous large-scale investigation of algorithm performance on the LFR benchmark [34]. We also included Leading Eigenvector [12] because its cluster bifurcation approach is similar to that employed by WLCF (Fig. 1). We used the network analysis package IGRAPH [50] to implement Multilevel, Label Propagation, and Leading Eigenvector; all parameters were set to their default values.

In addition, we used SCIKIT-LEARN to implement the NMF clustering method (see Methods for details). Since NMF requires the number of clusters as input, we provided $m^\star$, the exact number of communities in each LFR network. Both WLCF and WLA used $l_{\max} = 8$. In WLCF, random assignment of nodes to communities upon bifurcation was employed. Similar to NMF, WLA had to be provided with the exact number of communities $m^\star$ as input. As in Fig. S2 (see Supplemental Material [48]), NMF-based clustering was used to initialize the stand-alone WLA since random partition of the network into m communities in the beginning results in somewhat inferior performance, as shown below.

We observe that, qualitatively speaking, WLCF matches or outperforms all other algorithms in terms of NMI, with NMF and Multilevel being the most competitive alternatives (Fig. 2). However, the performance of NMF and Multilevel tends to deteriorate faster for larger networks. We also note that WLA provides a significant advantage over NMF (both algorithms require the exact number of clusters as input). As expected, the performance of all the algorithms degrades with μ since network communities become less well separated as μ increases. Another measure of performance is the relative error in predicting the number of clusters, $\Delta_m = |m - m^\star|/m^\star$, where m is the predicted and $m^\star$ is the exact number of communities in each LFR benchmark network. WLCF also outperforms Multilevel, Label Propagation, and Leading Eigenvector using this measure (see Supplemental Material [48], Fig. S4), especially with $\mu > 0.5$. The next best-performing algorithm is Multilevel, except for $N = 10^5$ where Label Propagation performs much better than Multilevel but still worse than WCLF. In summary, WLCF outperforms the other algorithms in terms of both NMI and Δ_m measures of prediction accuracy.

We have also explored how the performance of WLCF is affected by various hyperparameter, initialization, and algorithmic choices within its main pipeline (Fig. 1). In addition to the random assignment of nodes to two new communities at the bifurcation step which was used in the standard WLCF algorithm (Fig. 2), we have investigated the effects of more sophisticated community initialization protocols that employ either NMF or NNDSVD-based node assignment to provide better initial conditions for WLA within the WLCF pipeline (see Supplemental Material [48], Fig. S5). However, the effect was found to be minor on the LFR benchmark, leading us to conclude that nonrandom community initialization is not necessary as part of the WLCF protocol. Interestingly, there was a noticeable gain when stand-alone WLA was initialized with NMF-predicted rather than random communities (see Supplemental Material [48], Fig. S5). Apparently, gains related to NMF or NNSVD-based WLA initialization largely disappear when the number of new communities is always two, as is the case in the WLCF bifurcation step. Another potential reason is the WLCF community merge step, which may help rectify errors incurred by the randomly initialized WLA.

Since WLA depends on the maximum number of random walk steps $l_{\max}$, we have also investigated a version of WLCF

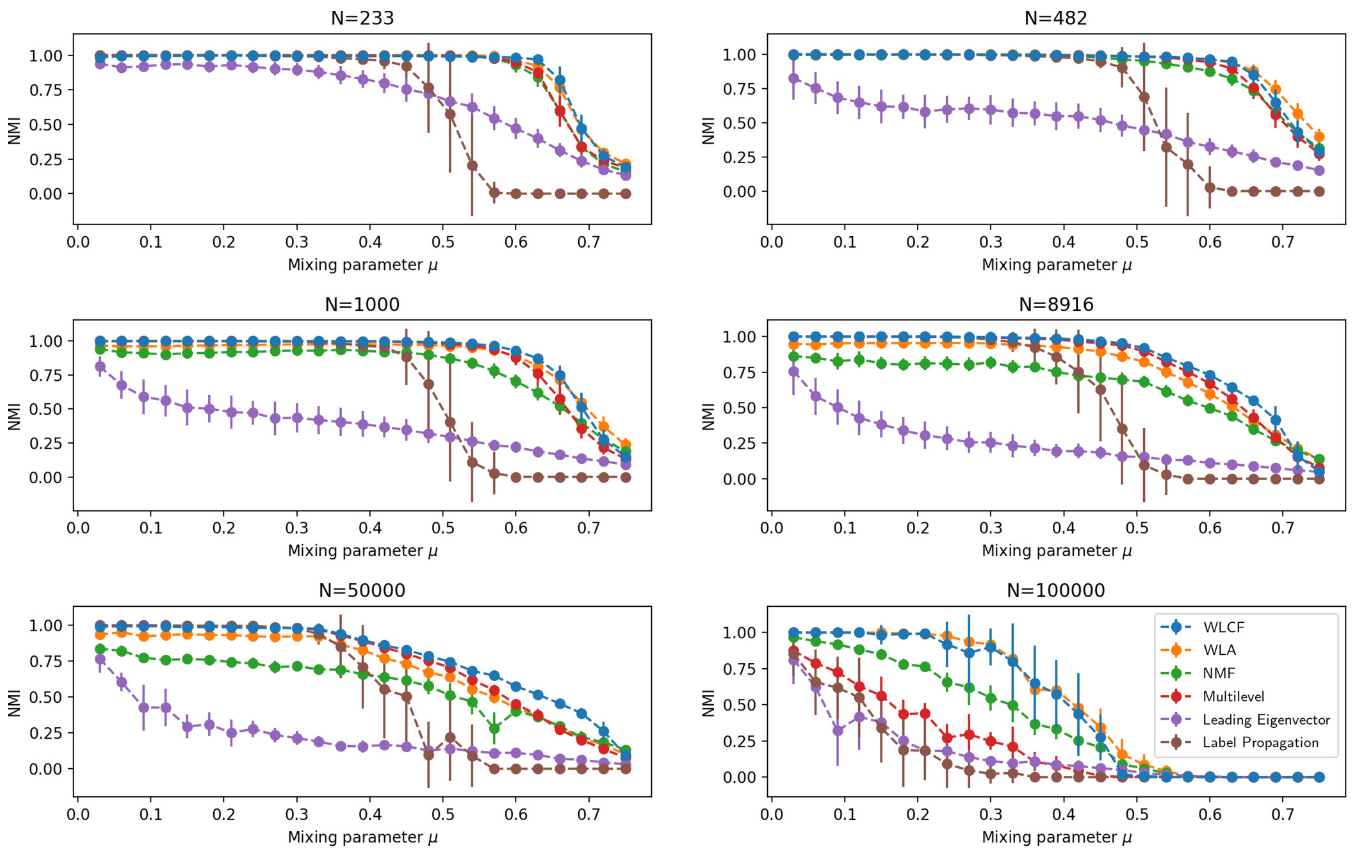

FIG. 2. Performance of WLCF and WLA on the LFR benchmark (NMI). In each panel, normalized mutual information (NMI) [Eq. (21)] is plotted as a function of the mixing parameter μ for a given LFR network size N (LFR network parameters are listed in Supplemental Material [48], Table S1). WLCF and WLA are compared with four state-of-the-art network community detection and clustering algorithms: Multilevel [14], Leading Eigenvector [12], Label Propagation [13], and non-negative matrix factorization (NMF) [36,37]. For each value of N and μ, we show $\langle \text{NMI} \rangle \pm \sigma_{\text{NMI}}$, where all averages and standard deviations are computed over independent network realizations.

in which WLA was run with $l_{\max} = 8$ followed by $l_{\max} = 1$ at every subsequent iteration of the main loop within WLA, starting with $l_{\max} = 8$. The alternation between high and low values of $l_{\max}$ was designed to explore both large- and small-scale network structures; however, no substantial gain was observed compared to WLA with $l_{\max} = 8$ (see Supplemental Material [48], Fig. S5). Finally, we have explored the overall role of WLA in the WLCF pipeline by replacing it completely with NMF-based node assignment (cf. purple curves in Supplemental Material [48], Fig. S5). Excluding WLA from the pipeline leads to significant degradation of the WLCF performance, leading us to conclude that the performance boost provided by WLA is indispensable for the overall success of the WLCF algorithm.

We have also studied how the time complexity of WLCF and WLA scales with the network size N. We empirically observe power-law behavior of the runtime on the LFR networks from our data set $T \sim N^{\alpha}$, with the scaling exponents ranging from 1.19 to 1.74 for WLCF and from 1.39 to 1.91 for WLA (see Supplemental Material [48], Fig. S6). This relatively weak dependence on the network size leads us to conclude that both of our algorithms are capable of treating large-scale networks.

V. REAL-WORLD NETWORKS

A. Eight networks

After exploring the performance of our algorithms on the LFR benchmark, we have applied WLCF to eight small- and medium-size real-world networks widely studied in the network literature: Bottlenose dolphins network [51], Les Misérables network [52], American college football teams network [5], Jazz musicians network [53], *C. elegans* neural network [54], Erdos coauthorship network [55,56], Edinburgh associative thesaurus network [57], and High-energy theory (HET) citation network [58] [see Supplemental Material [48], Methods for the details of each network].

We find that WLCF and Multilevel produce comparable modularity scores (Table I), while the performance of the Leading Eigenvector and the Label Propagation algorithms is worse overall (see Supplemental Material [48], Table S2). Interestingly, WLCF tends to predict fewer clusters than Multilevel, furnishing more interpretable partitions without a substantial loss in the modularity score. To investigate the nature of the network partitions found by the four algorithms, we have also computed the distributions of internal edge density and cut ratio scores [31,32] (see Methods). Despite being

TABLE I. Performance of community detection algorithms on real-world networks. Shown are the average and the standard deviation of the modularity score M [Eq. (13)] and the number of clusters $N_{\rm cl}$ predicted by WLCF and Multilevel algorithms on eight real-world networks (see Supplemental Material [48] for detailed network descriptions). All statistics are computed using 10^2 independent runs of each algorithm per network. The networks are unweighted (i.e., all edge weights are set to 1.0). N is the number of nodes in the network and $\langle k\rangle$ is the average number of links per node, a measure of network sparseness.

			WLCF		Multilevel	
Network	N	$\langle k\rangle$	$\langle M\rangle \pm \sigma_M$	$\langle N_{\rm cl}\rangle \pm \sigma_{N_{\rm cl}}$	$\langle M\rangle \pm \sigma_M$	$\langle N_{\rm cl}\rangle \pm \sigma_{N_{\rm cl}}$
Dolphin groups	62	5.13	0.5181 ± 0.0123	4.21 ± 0.45	0.5204 ± 0.0029	5.15 ± 0.55
Les Misérables characters	77	6.60	0.5467 ± 0.0109	5.45 ± 0.65	0.5563 ± 0.0028	6.34 ± 0.55
Football teams	115	10.66	0.6023 ± 0.0050	9.75 ± 0.54	0.6039 ± 0.0018	9.69 ± 0.52
Jazz musicians	198	27.70	0.4404 ± 0.0034	3.35 ± 0.48	0.4430 ± 0.0025	3.84 ± 0.37
C. elegans neurons	297	15.80	0.3957 ± 0.0086	4.79 ± 0.82	0.4093 ± 0.0054	5.75 ± 0.50
Erdos coauthors	6927	3.42	0.6650 ± 0.0097	25.41 ± 1.93	0.6957 ± 0.0018	31.77 ± 1.77
Thesaurus words	23219	67.95	0.3201 ± 0.0027	7.62 ± 0.81	0.3149 ± 0.0028	12.20 ± 1.12
HET citations	27770	25.41	0.6529 ± 0.0030	16.37 ± 1.11	0.6554 ± 0.0028	171.56 ± 1.83

normalized by the total number of possible links, both scores tend to correlate with the number of clusters into which the network is partitioned since the internal edge density is high in small, densely connected clusters, whereas the cut ratio is low in large clusters with relatively few outside links.

We observe that WLCF clusters do not have the highest internal edge density scores: the scores tend to be consistently smaller than those of Multilevel clusters (see Supplemental Material [48], Table S3) and the results are mixed vs Leading Eigenvector and Label Propagation clusters (see Supplemental Material [48], Table S4). The biggest discrepancies can be traced to the differences in the number of clusters predicted by the four algorithms. For example, WLCF produces many fewer clusters in the HET citations network, resulting in much lower internal edge density scores. However, WLCF tends to produce lower cut ratio scores compared with the other three algorithms, a sign of more self-contained clusters with fewer external links. Overall, we conclude that WLCF optimizes modularity and cut ratio scores to a larger extent than internal edge density, partly because it partitions the network into fewer clusters.

We have also investigated how WLCF cluster predictions are affected by including edge weights. We have focused on two of the networks where edge weights are available in the primary data: Les Misérables characters and Thesaurus words (see SM Methods for edge weight definitions). With the Les Misérables characters network, we obtain $\langle M\rangle \pm \sigma_M = 0.5621 \pm 0.0064$ and $\langle N_{\rm cl}\rangle \pm \sigma_{N_{\rm cl}} = 5.82 \pm 0.41$ over 10^2 independent runs of the WLCF algorithm when the weights are included. These results are similar to those on the unweighted network, and indeed $\langle {\rm NMI}\rangle \pm \sigma_{\rm NMI} = 0.78 \pm 0.05$ between weighted and unweighted network partitions, showing that they are fairly consistent. In contrast, for Thesaurus words we observe $\langle M\rangle \pm \sigma_M = 0.4759 \pm 0.0069$ and $\langle N_{\rm cl}\rangle \pm \sigma_{N_{\rm cl}} = 15.30 \pm 1.62$, a much more modular network with twice as many clusters compared to the unweighted version (Table I). The low overlap between weighted and unweighted network clusters ($\langle {\rm NMI}\rangle \pm \sigma_{\rm NMI} = 0.30 \pm 0.02$) shows that the decision to include or disregard edge weights plays a major role in this case. These findings underscore the necessity of the careful design of the experiments that generate primary data.

B. Colorado road map

To investigate whether our approach can be applied to large-scale networks, we have chosen a graph defined by geographical coordinates of road intersections and other landmarks in the state of Colorado [59]. The network is very sparse, with $N = 435\,666$ nodes and $\mathcal{E} = 528\,533$ edges. We have made the network unweighted by assigning unit weights to each edge and run the WLA algorithm on it multiple times (Fig. 3). We observe that with $m \leqslant 16$, independent runs result in somewhat different cluster assignments, as can be seen from the lower NMI values and the error bars in Fig. 3(a). However, as the number of clusters increases, the assignment of nodes to clusters becomes more reproducible, with the NMI values around 0.87 and high consistency between the runs. Similarly, the modularity score improves with the number of clusters, with the values around 0.97 for $m > 40$ [Fig. 3(b)]. These high values of modularity scores are not surprising since, given the sparseness of the network, it is relatively easy to partition the graph into smaller clusters that are only weakly connected to one another.

Figure 3(c) shows a single randomly chosen realization of partitioning the network into $m = 16$ clusters (see Supplemental Material [48], Fig. S7 for three additional examples with $m = 2, 4, 8$). In all of these examples, the results are intuitively compelling: each cluster occupies a geographically contiguous region and the boundaries between neighboring communities often coincide with mountain ranges, major rivers, and other geographical landmarks. We conclude that our approach can be used to detect community structure in large-scale complex networks.

VI. DISCUSSION

In this work, we have developed an approach to partitioning complex networks into nonoverlapping communities. Networks that occur in nature and society often exhibit community structure, with nodes within communities connected by more links than nodes in different communities (see, e.g., Refs. [5,6,26–28]). However, this structure is often challenging to detect and there may be many alternative solutions of similar quality, confronting community detection algorithms

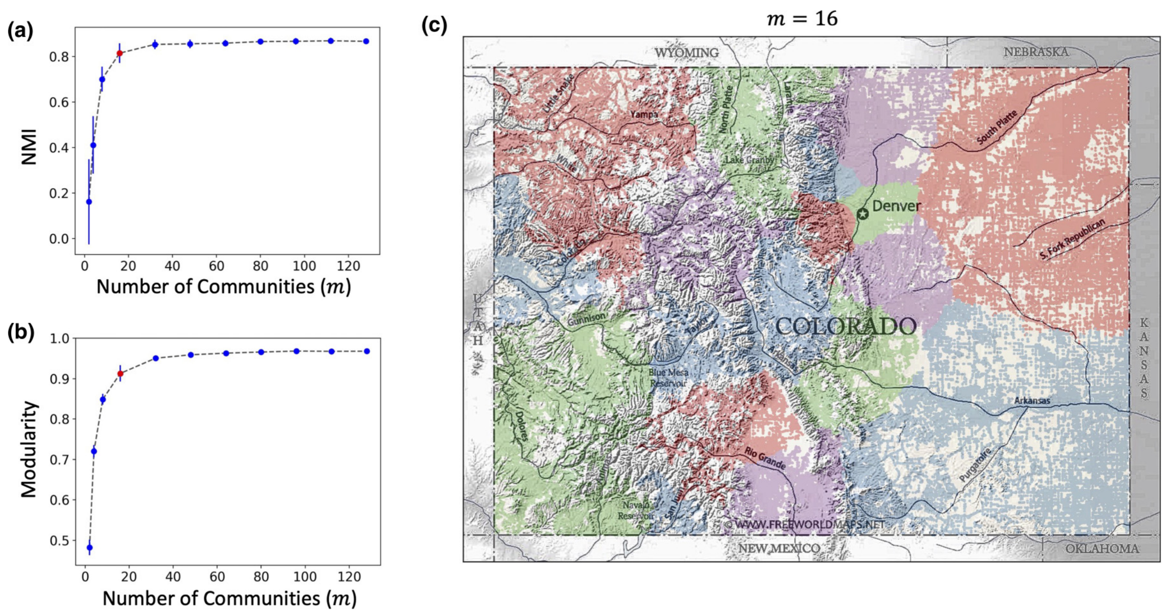

FIG. 3. WLA clustering of the Colorado road network. WLA was run 20 times for each value of m, with $m = \{2, 4, 8, 16, 32, 48, 64, 80, 96, 112, 128\}$ (220 independent runs in total). Each run started from a random initial assignment of nodes to communities and used $l_{\max} = 10^5$. (a) Mean and standard deviation of the normalized mutual information (NMI) for the ensemble of all $\binom{20}{2}$ unique pairs of network partitions for each value of m. (b) Mean and standard deviation of the modularity score for 20 runs for each value of m. (c) Visualization of one randomly chosen network partition with $m = 16$ communities [shown as a red dot in (a) and (b)]. Each node was assigned the color of its community and superimposed on a Colorado map using its longitude and latitude coordinates. The geographical map of the Colorado state was obtained from [60] and rendered black and white. Colors were assigned to each community using the greedy coloring algorithm NETWORKX.ALGORITHMS.COLORING.GREEDY_COLOR from the NetworkX Python network analysis package [61]. The coloring algorithm assigned 4 colors (red, green, blue, and purple) to 16 communities such that no pair of adjacent communities have the same color.

with a hard optimization problem. The task of finding communities in networks is similar to a clustering problem in machine learning, in which, in the case of hard clustering, the data set is divided into disjoint subsets on the basis of pairwise distances between data points [35].

Our approach is based on the observation that short random walks that start in a given community will preferentially explore that community. To avoid potential issues related to finite sampling, we formally consider the limit of an infinite number of random walks that start from all nodes in the network. For each random walk, the expected number of visits to each node in the network is computed exactly using the transition matrix of the network. Since the total number of random walks is infinite, there is no sampling noise and the expected number of visits to each node provides an exact statistic, which is then used to assign nodes to communities in a Bayesian sense. The number of steps in each random walk, $l_{\max}$, is a key hyperparameter of the algorithm: choosing a very small value will mean that walks may not be able to reach some of the nodes within their own community, while choosing a very large value will make it more difficult to differentiate between communities (see Supplemental Material [48], Fig. S1). Nonetheless, our approach does not appear to be overly sensitive to the exact value of $l_{\max}$ (see Supplemental Material [48], Fig. S2).

In practice, our algorithm, which we call the walk-likelihood algorithm, or WLA for short, is run iteratively starting from the initial condition that may be random, drawn from an informative prior, or provided by another algorithm such as non-negative matrix factorization (NMF) [36,37]. The algorithm is terminated once the partition of the network into m communities stops changing substantially from iteration to iteration. Since WLA requires the total number of communities m as input, we have created another algorithm, the walk-likelihood community finder, or WLCF, which uses WLA as a basic building block to produce the optimal number of network communities m_{opt} through global moves such as community bifurcation and merging (Fig. 1). Alternatively, WLA can be simply run multiple times within a prespecified range of m values, with m_{opt} equal to m of the most modular partition.

Our main score for judging the success of the clustering procedure is the network modularity score [8], although we have also considered two additional measures: the internal edge density and the cut ratio [31,32]. To benchmark WLA and WLCF against other algorithms in a controlled setting, we have employed the LFR benchmark which was created to provide a challenging test for community detection algorithms [47]. On this benchmark, WLA and WLCF compare very favorably with several state-of-the-art community detection and clustering algorithms (Fig. 2 and Fig. S4 in the Supplemental Material [48]). Moreover, in accordance with Fig. S2 (Supplemental Material [48]), the dependence on the exact values of $l_{\max}$ is also found to be weak in this comparison (see Supplemental Material [48], Figs. S3 and S5).

Another data set we have considered consists of eight small- and medium-size real-world networks that are often investigated in the network science literature (Table I and Table S2 in the Supplemental Material [48]). On this group of networks, WLCF produces modularity scores comparable to those predicted by another algorithm, Multilevel [14], while partitioning the network into fewer clusters. WLCF also tends to produce low cut ratio scores, a sign that it identifies self-contained clusters with few external links. However, WLCF clusters are not characterized by the highest internal edge density scores compared to the other algorithms (see Supplemental Material [48], Tables S3 and S4), probably because these scores increase trivially with the number of communities and WLCF tends to produce fewer clusters.

Using a set of networks from the LFR benchmark, we find a power-law relation between the WLCF and WLA running times T and the total number of nodes in the network: $T \sim N^{\alpha}$, with the scaling exponent $1.0 < \alpha < 2.0$ that depends on the network type (see Supplemental Material [48], Fig. S6). Therefore, our approach can be used to analyze large-scale networks which may present difficulties to other algorithms. To demonstrate this ability, we have applied WLA to a network of roads in the state of Colorado with almost half a million nodes (Fig. 3 and Fig. S7 in the Supplemental Material [48]). The results are geographically sensible, with neighboring clusters separated by major rivers, mountain ranges, or corresponding to urban agglomerations such as Denver metropolitan area.

To summarize, our computational framework for clustering and network community detection is efficient and robust with respect to the choice of initial conditions and hyperparameter values. It compares favorably with several state-of-the-art algorithms. Although ideas centered on random walks and diffusion processes were previously explored in network science and machine learning in the context of network community detection [11,12,17,19,20,25], diffusion maps [42–44], and spectral clustering [40,41], our approach is unique in its use of random walks to assign nodes to communities probabilistically in a Bayesian sense. This is a significant extension of our previous work, which used conceptually similar ideas to infer properties of the entire network, such as its size, on the basis of sparse exploration by random walks, but without partitioning the network into distinct communities [45]. In the future, we will investigate both applications and algorithmic extensions of our approach, including its adaptation to the soft clustering problem.

VII. METHODS

A. Network community metrics

Consider a network (undirected graph) with N nodes, or vertices. The network is divided into m nonoverlapping communities, or clusters, with N_c nodes in community $c = 1 \dots m$: $N = \sum_{c=1}^{m} N_c$. The network contains $\mathcal{E}$ edges in total; we also define $\mathcal{I}_c$, the total number of internal edges that connect nodes within community c, and $\mathcal{E}_c$, the total number of external edges that connect nodes in community c to nodes in all other communities. Finally, a node i $(i = 1 \dots N)$ has k_i edges attached to it, such that $\mathcal{E} = (1/2)\sum_{i=1}^{N} k_i$ and $T_c = \sum_{i \in c} k_i$ is the total number of edge ends attached to the nodes in community c.

With these definitions, the modularity score is given by [8]

$$M = \sum_{c=1}^{m} \left(e_{cc} - a_c^2\right), \tag{13}$$

where $e_{cc} = \mathcal{I}_c/\mathcal{E}$ is the fraction of all network edges that are internal to community c and $a_c = T_c/2\mathcal{E}$ is the fraction of all edge ends that are attached to the vertices in community c, such that a_c^2 is the expected value of the fraction of edges internal to the community c if the edges were placed at random. Thus, the modularity score is a sum over differences between the observed and the expected fraction of internal edges in each community. By construction, the positive modularity score indicates nontrivial groupings of nodes within the network with, on average, more connections between nodes within each community than could be expected by chance.

We also introduce two additional metrics used to estimate the quality of network partitions into communities [31,32]: (i) the internal edge density $\mathcal{D}_c = 2\mathcal{I}_c/N_c(N_c - 1)$, which measures the fraction of all possible internal edges observed in cluster c, averaged over all clusters: $\mathcal{D} = (1/m)\sum_{c=1}^{m} \mathcal{D}_c$; (ii) the average cut ratio $\mathcal{R} = (1/m)\sum_{c=1}^{m} \mathcal{R}_c$, where $\mathcal{R}_c = \mathcal{E}_c/N_c(N - N_c)$ is the fraction of all possible external edges leaving the cluster.

B. Non-negative matrix factorization (NMF)

NMF is based on decomposing a non-negative matrix $X_{N_1 \times N_2}$ into two non-negative matrices $L_{N_1 \times m}$ and $R_{m \times N_2}$: $X = LR$ [36–39]. To cluster a graph into m communities using NMF, the adjacency matrix of the graph, $\tilde{A}$, is factorized into L and R, and each node is assigned to the community with the largest matrix element in the corresponding row of L. Note that $N_1 = N_2 = N$ in this case. Specifically, the indicator matrix U is defined as $U_{ic} = \text{argmax}_i(L_{ic})$, resulting in the partition of the graph into m communities. We used SCIKIT-LEARN to implement the NMF clustering method [62], with the coordinate descent solver (*solver='cd'*), Non-negative Double Singular Value Decomposition (NNDSVD) initialization (*init='‘ndsvd'*) [38], and all other parameters left at their default values.

C. Random walks on networks with communities

Consider a discrete-time random walk on an undirected network with weighted edges: $\{w_{ij}\}$, where $w_{ij} = w_{ji}$ is the edge weight or rate of transmission from node i to j (note that $w_{ij} = 1$, $\forall\, i, j$ for unweighted networks). At each step the random walker jumps to its nearest neighbor with the probability $P(i \to j) = w_{ij}/w_i$, where $w_i = \sum_{k \in nn(i)} w_{ik}$ is the connectivity (weighted degree) of node i and the sum is over all nearest neighbors of node i. For unweighted networks $w_i = k_i$, the node degree defined as the total number of edges attached to node i. We assume that the network has a community c with N_c nodes. Then the average return time (i.e., the average number of random walk steps) to a node $n \in c$, provided that there are no transitions outside of the community, is given by $\mathcal{W}_c/w_n$ [45,63–65], where $\mathcal{W}_c = \sum_{i=1}^{N_c} w_i$ is the weighted size of all nodes in community c. Note that for a set $S = \{n_1, n_2, \ldots, n_{N_p}\}$ of nodes in community c, the average return time to any of the nodes in the set S is given by $\mathcal{W}_c/\mathcal{W}_p$ in the absence of intercommunity transitions, where $\mathcal{W}_p = \sum_{i=1}^{N_p} w_i$ is the weighted size of all nodes in set S.

Assuming that the distribution of return times is exponential, or memoryless [45,66], the probability density to return to a node $n \in c$ after exactly $\Delta\ell_c$ steps, with no transitions outside of the community c, is given by

$$P(\Delta\ell_c) = \frac{w_n}{\mathcal{W}_c} e^{-(w_n/\mathcal{W}_c)\Delta\ell_c} \tag{14}$$

in the continuous approximation [note that $\int_0^\infty \Delta\ell_c P(\Delta\ell_c) = 1$]. Correspondingly, the probability of not making a return for $\Delta\ell_c$ steps (i.e., the survival probability) is

$$S(\Delta\ell_c) = 1 - \int_0^{\Delta\ell_c} \Delta\ell_c' P(\Delta\ell_c') = e^{-(w_n/\mathcal{W}_c)\Delta\ell_c}. \tag{15}$$

Under the memoryless assumption, the probability of making $\mathcal{K}$ returns to the node n if ℓ_c steps are taken within the community c is given by the Poisson distribution $\mathcal{P}$ [45] with the rate parameter $w_n\ell_c/\mathcal{W}_c$:

$$P(\mathcal{K}|\ell_c) = \mathcal{P}\left(\mathcal{K}, \frac{w_n\ell_c}{\mathcal{W}_c}\right) = \frac{1}{\mathcal{K}!}\left(\frac{w_n}{\mathcal{W}_c}\ell_c\right)^{\mathcal{K}} e^{-(w_n/\mathcal{W}_c)\ell_c}. \tag{16}$$

The mean number of visits to the node $n \in c$ is equal to the Poisson rate parameter:

$$E(\mathcal{K}|n) = \frac{w_n\ell_c}{\mathcal{W}_c}. \tag{17}$$

Next, consider a stochastic process in which node n is visited $\{\mathcal{K}_{nc'}\}_{c'=1}^m$ times by m random walks, on a network with m communities. Each random walk starts from an arbitrary node in a distinct community $c' = 1 \ldots m$ and therefore can be unambiguously labeled by the community index. During a random walk originating in community c', the random walker takes $\ell_{c'c}$ steps on each community c of weighted size $\mathcal{W}_c$: $\{\ell_{c'c}\}_{c'=1}^m$ (we adopt the convention that a jump from community c' to c is counted as a step in community c). Now, if the node $n \in c$, the probability of visiting this node $\{\mathcal{K}_{nc'}\}_{c'=1}^m$ times by m random walks defined above is given by

$$P(\{\mathcal{K}_{nc'}\}_{c'=1}^m | n \in c, \{\ell_{c'c}\}_{c'=1}^m) = \prod_{c'=1}^m \mathcal{P}\left(\mathcal{K}_{nc'}, \frac{w_n\ell_{c'c}}{\mathcal{W}_c}\right). \tag{18}$$

If the community assignment of the node n is not known, we can use Bayes' theorem to find the posterior probability that the node $n \in c$:

$$P(n \in c | \{\mathcal{K}_{nc'}\}_{c'=1}^m, \{\ell_{c'c}\}_{c'=1}^m) = \frac{1}{\mathcal{Z}} \Pr(n \in c) \prod_{c'=1}^m \mathcal{P}\left(\mathcal{K}_{nc'}, \frac{w_n\ell_{c'c}}{\mathcal{W}_c}\right), \tag{19}$$

where the normalization constant $\mathcal{Z}$ is given by

$$\mathcal{Z} = \sum_{c=1}^m \left[\Pr(n \in c) \prod_{c'=1}^m \mathcal{P}\left(\mathcal{K}_{nc'}, \frac{w_n\ell_{c'c}}{\mathcal{W}_c}\right)\right] \tag{20}$$

and $\Pr(n \in c) = N_{c,0}/N$ is the prior probability that node n is found in the community c of the initial size $N_{c,0}$. If all communities are *a priori* assumed to be of equal size, $N_{c,0} = N/m$ and the prior is uniform: $\Pr(n \in c) = m^{-1}$. More informative priors may also be used when something is known about the distribution of community sizes, for example, if $P(N_{c,0}) \sim N_{c,0}{}^{-\beta}$ with known β. In this case, the initial community sizes can be generated by sampling from $P(N_{c,0})$.

Normalized mutual information

We use NMI [46] to quantify the similarity between network partitions U and U':

$$\mathrm{NMI}(U, U') = \frac{2\sum_{c=1}^m \sum_{c'=1}^{m'} P_{UU'}(c,c') \log\left[P_{UU'}(c,c')/(P_U(c)P_{U'}(c'))\right]}{\sum_{c=1}^m P_U(c)\log P_U(c) + \sum_{c=1}^{m'} P_{U'}(c)\log P_{U'}(c)}, \tag{21}$$

where $P_U(c) = N^{-1}\sum_{n=1}^N U_{nc}$, $P_{UU'}(c,c') = N^{-1}\sum_{n=1}^N U_{nc}U'_{nc'}$, and m and m' refer to the number of communities in the partitions U and U', respectively. Note that NMI is always between 0 and 1, with $\mathrm{NMI}(U, U') = 1$ if and only if the partitions U and U' are exactly the same. Although Eq. (21) is valid for general values of m and m', we focus on $m = m'$ because WLA node reassignment procedure does not change the number of communities.

A Python implementation of WLA and WLCF is available in [67].

ACKNOWLEDGMENTS

A.B. and A.V.M. were supported by a grant from the National Science Foundation (Grant No. NSF MCB1920914). A.B. would like to thank A. Bhrushundi for suggesting the benchmarks to test our algorithms against and for many other discussions related to this project.

[1] D. J. Watts and S. H. Strogatz, Collective dynamics of 'small-world' networks, Nature (London) **393**, 440 (1998).

[2] R. Albert and A. L. Barabási, Statistical mechanics of complex networks, Rev. Mod. Phys. **74**, 47 (2002).

[3] A. Barrat, M. Barthelemy, and A. Vespignani, *Dynamical Processes on Complex Networks* (Cambridge University Press, Cambridge, UK, 2008).

[4] M. E. J. Newman, *Networks: An Introduction* (Oxford University Press, Oxford, UK, 2010).

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

[6] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, Defining and identifying communities in networks, Proc. Natl. Acad. Sci. USA **101**, 2658 (2004).

[7] J. Reichardt and S. Bornholdt, Detecting Fuzzy Community Structures in Complex Networks with a Potts Model, Phys. Rev. Lett. **93**, 218701 (2004).

[8] M. E. J. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E **69**, 066133 (2004).

[9] M. E. J. Newman, Detecting community structure in networks, Eur. Phys. J. B **38**, 321 (2004).

[10] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature (London) **435**, 814 (2005).

[11] P. Pons and M. Latapy, Computing communities in large networks using random walks, in *Computer and Information Sciences - ISCIS 2005*, edited by P. Yolum, T. Güngör, F. Gürgen, and C. Özturan (Springer, Berlin, 2005), pp. 284–293.

[12] M. E. J. Newman, Finding community structure in networks using the eigenvectors of matrices, Phys. Rev. E **74**, 036104 (2006).

[13] 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).

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

[15] J. M. Hofman and C. H. Wiggins, Bayesian Approach to Network Modularity, Phys. Rev. Lett. **100**, 258701 (2008).

[16] M. Rosvall and C. T. Bergstrom, An information-theoretic framework for resolving community structure in complex networks, Proc. Natl. Acad. Sci. USA **104**, 7327 (2007).

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

[18] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, Inference and Phase Transitions in the Detection of Modules in Sparse Networks, Phys. Rev. Lett. **107**, 065701 (2011).

[19] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, Stability of graph communities across time scales, Proc. Natl. Acad. Sci. USA **107**, 12755 (2010).

[20] R. Lambiotte, J.-C. Delvenne, and M. Barahona, Random walks, Markov processes and the multiscale modular organization of complex networks, IEEE Trans. Netw. Sci. Eng. **1**, 76 (2014).

[21] M. Gerlach, T. P. Peixoto, and E. G. Altmann, A network approach to topic models, Sci. Adv. **4**, eaaq1360 (2018).

[22] T. P. Peixoto, Parsimonious Module Inference in Large Networks, Phys. Rev. Lett. **110**, 148701 (2013).

[23] T. P. Peixoto, Network Reconstruction and Community Detection from Dynamics, Phys. Rev. Lett. **123**, 128301 (2019).

[24] L. Zhang and T. P. Peixoto, Statistical inference of assortative community structures, Phys. Rev. Res. **2**, 043271 (2020).

[25] A. Gosztolai and A. Arnaudon, Unfolding the multiscale structure of networks with dynamical Ollivier-Ricci curvature, Nat. Commun. **12**, 4561 (2021).

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

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

[28] T. P. Peixoto, Descriptive vs. inferential community detection: Pitfalls, myths and half-truths, arXiv:2112.00183.

[29] A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown, Genomic expression programs in the response of yeast cells to environmental changes, Mol. Biol. Cell **11**, 4241 (2000).

[30] A. Mihalik and P. Csermely, Heat shock partially dissociates the overlapping modules of the yeast protein-protein interaction network: A systems level model of adaptation, PLoS Comput. Biol. **7**, e1002187 (2011).

[31] J. Leskovec, K. Lang, and M. Mahoney, Empirical comparison of algorithms for network community detection, in *Proceedings of the 19th International Conference on World Wide Web, WWW 2010* (Association for Computing Machinery, New York, 2010), pp. 1–10.

[32] Y. Yang, Y. Sun, S. Pandit, N. V. Chawla, and J. Han, Perspective on measurement metrics for community detection algorithms, *Mining Social Networks and Security Informatics* (Springer, Dordrecht, The Netherlands, 2013), pp. 227–242.

[33] A. Clauset, M. E. J. Newman, and C. Moore, Finding community structure in very large networks, Phys. Rev. E **70**, 066111 (2004).

[34] Z. Yang, R. Algesheimer, and C. J. Tessone, A comparative analysis of community detection algorithms on artificial networks, Sci. Rep. **6**, 30750 (2016).

[35] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, New York, 2006).

[36] D. D. Lee and H. S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature (London) **401**, 788 (1999).

[37] D. Lee and H. S. Seung, Algorithms for non-negative matrix factorization, in *Advances in Neural Information Processing Systems*, Vol. 13, edited by T. Leen, T. Dietterich, and V. Tresp (MIT Press, Cambridge, MA, 2001), pp. 1–7.

[38] C. Boutsidis and E. Gallopoulos, SVD based initialization: A head start for nonnegative matrix factorization, Pattern Recognition **41**, 1350 (2008).

[39] D. Kuang, C. Ding, and H. Park, Symmetric nonnegative matrix factorization for graph clustering, in *Proceedings of the 2012 SIAM International Conference on Data Mining (SDM)* (SIAM, Philadelphia, 2012), pp. 106–117.

[40] C. Ding, X. He, and H. D. Simon, On the equivalence of nonnegative matrix factorization and spectral clustering, in *Proceedings of the 2005 SIAM International Conference on Data Mining (SDM)* (SIAM, Philadelphia, 2005), pp. 606–610.

[41] U. von Luxburg, A tutorial on spectral clustering, Stat. Comput. **17**, 395 (2007).

[42] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker, Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps, Proc. Natl. Acad. Sci. USA **102**, 7426 (2005).

[43] R. R. Coifman and S. Lafon, Diffusion maps, Appl. Comput. Harmonic Anal. **21**, 5 (2006).
[44] J. de la Porte, B. M. Herbst, W. A. Hereman, and S. J. van der Walt, An introduction to diffusion maps (unpublished).
[45] W. B. Kion-Crosby and A. V. Morozov, Rapid Bayesian Inference of Global Network Statistics Using Random Walks, Phys. Rev. Lett. **121**, 038301 (2018).
[46] A. Strehl and J. Ghosh, Cluster ensembles–a knowledge reuse framework for combining multiple partitions, J. Mach. Learn. Res. **3**, 583 (2002).
[47] A. Lancichinetti, S. Fortunato, and F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E **78**, 046110 (2008).
[48] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevResearch.4.043117 for details on eight real-world networks, Figs. S1–S7, and Tables S1–S4.
[49] `https://github.com/eXascaleInfolab`.
[50] `https://igraph.org`.
[51] D. Lusseau, The emergent properties of a dolphin social network, Proc. R. Soc. London B **270**, S186 (2003).
[52] D. E. Knuth, *The Stanford GraphBase: A Platform for Combinatorial Computing* (ACM Press, New York, 1993).
[53] P. M. Gleiser and L. Danon, Community structure in jazz, Adv. Complex Syst. **06**, 565 (2003).
[54] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner, The structure of the nervous system of the nematode Caenorhabditis elegans, Philos. Trans. R Soc. London, Ser. B **314**, 1 (1986).
[55] J. Kunegis, KONECT: the Koblenz network collection, in *Proceedings of the International Conference on World Wide Web Companion* (International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 2013), pp. 1343–1350.
[56] R. A. Rossi and N. K. Ahmed, The network data repository with interactive graph analytics and visualization, in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (Association for the Advancement of Artificial Intelligence, Menlo Park, CA, 2015), pp. 4292–4293.
[57] G. R. Kiss, C. Armstrong, R. Milroy, and J. Piper, An associative thesaurus of English and its computer analysis, in *The Computer and Literary Studies*, edited by A. J. Aitkin, R. W. Bailey, and N. Hamilton-Smith (University Press, Edinburgh, UK, 1973).
[58] J. Gehrke, P. Ginsparg, and J. Kleinberg, Overview of the 2003 KDD cup, ACM SIGKDD Explorations Newsletter **5**, 149 (2003).
[59] `https://www.diag.uniroma1.it/challenge9/download.shtml`.
[60] `https://www.freeworldmaps.net`.
[61] `https://networkx.org`.
[62] `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html`.
[63] J. D. Noh and H. Rieger, Random Walks on Complex Networks, Phys. Rev. Lett. **92**, 118701 (2004).
[64] S. Condamin, O. Bénichou, V. Tejedor, R. Voituriez, and J. Klafter, First-passage times in complex scale-invariant media, Nature (London) **450**, 77 (2007).
[65] S. Condamin, O. Bénichou, and M. Moreau, Random walks and Brownian motion: A method of computation for first-passage times and related quantities in confined geometries, Phys. Rev. E **75**, 021111 (2007).
[66] M. Manhart, W. Kion-Crosby, and A. V. Morozov, Path statistics, memory, and coarse-graining of continuous-time random walks on networks, J. Chem. Phys. **143**, 214106 (2015).
[67] https://github.com/lordareicgnon/Walk_likelihood/.