Random Spatial Network Models with Core-Periphery Structure

Junteng Jia
Cornell University
jj585@cornell.edu

Austin R. Benson
Cornell University
arb@cs.cornell.edu

ABSTRACT

Core-periphery structure is a common property of complex networks, which is a composition of tightly connected groups of core vertices and sparsely connected periphery vertices. This structure frequently emerges in traffic systems, biology, and social networks via underlying spatial positioning of the vertices. While core-periphery structure is ubiquitous, there have been limited attempts at modeling network data with this structure. Here, we develop a generative, random network model with core-periphery structure that jointly accounts for topological and spatial information by "core scores" of vertices. Our model achieves substantially higher likelihood than existing generative models of core-periphery structure, and we demonstrate how the core scores can be used in downstream data mining tasks, such as predicting airline traffic and classifying fungal networks. We also develop nearly linear time algorithms for learning model parameters and network sampling by using a method akin to the fast multipole method, a technique traditional to computational physics, which allow us to scale to networks with millions of vertices with minor tradeoffs in accuracy.

ACM Reference Format:
Junteng Jia and Austin R. Benson. 2019. Random Spatial Network Models with Core-Periphery Structure. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3289600.3290976

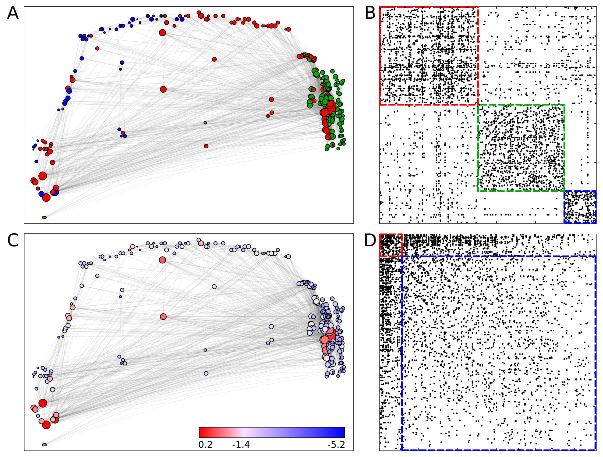

Figure 1: Community (A–B) and core-periphery (C–D) structure in the *C. elegans* network, where vertices are neurons, and edges are neural connections. Vertex coordinates are neuron locations in the lateral plane [16] and vertex sizes are proportional to the square root of degrees. (A) The Louvain algorithm [19] finds three communities (identified here by the three colors). (B) The adjacency matrix ordered by the three communities. (C) Our proposed model learns vertex "core scores" based on spatial location and connectivity, where larger core scores are more indicative of a vertex being in the core; here, the maximum and minimum core scores are 0.2 and -5.2. (D) The adjacency matrix ordered by decreasing vertex core score. Vertices with high core scores are both spatially distributed and densely connected.

1 NETWORK CORE-PERIPHERY STRUCTURE

Networks are widely used to model the interacting components of complex systems emerging from biology, ecosystems, economics, and sociology [1–3]. A typical network consists of a set of vertices V and a set of edges E, where the vertices represent discrete objects (e.g., people or cities) and the edges represent pairwise connections (e.g., friendships or highways). Networks are often described in terms of local properties such as vertex degree or local clustering coefficients and global properties such as diameter or the number of connected components. At the same time, a number of mesoscale proprieties are consistently observed in real-world networks, which often reveal important structural information of the underlying complex systems; arguably the most well-known is community structure, and a tremendous amount of effort has been devoted to its explanation and algorithmic identification [4–7].

Another important mesoscale structure is *core-periphery* structure, although such structure has received relatively little attention. In contrast to community detection, which separates vertices into several well-connected modules, core-periphery identification involves finding sets of cohesive core vertices and periphery vertices loosely connected to both each other and the cores. This type of structure is common in traffic [8, 9], economic [9, 10], social [11–13], and biological [14] networks. Oftentimes, spatial information is a driving factor in the core-periphery structure [15–17]. For example, in the *C. elegans* neural network shown in Fig. 1, most long-distance neural connections are between tightly connected early-born neurons, which serve as hubs and constitute the core [16]. Identifying core-periphery structure not only provides us with a new perspective to study the mesoscale structure in networked systems but can also leads to insights on their functionalities [18].

At the same time, random network models are useful for analyzing and understanding networks [20, 21]. For example, they are routinely used as the null model to verify that features of real world networks are not due randomness [22–24] and also serve to identify community structure, as is the case with the stochastic block model and its variants [25–27] as well as methods such as BigCLAM [28] and CESNA [29]. These models successfully incorporate community structure, but have yet to effectively model core-periphery structure. While block modeling can also be used to identify core-periphery structure [12], we later show that such an approach is limited. Moreover, in general, spatial information is not often incorporated into random network models, even though it can play a large role in their structure.

Here, we present a random network model that generates networks with core-periphery structure. In our model, each vertex has a real-valued *core score* to reflect its role in the core. Our model assumes the edges are generated by a random process and the probability that two vertices connect is an increasing function of their core scores. Given an arbitrary network, we infer the model parameters (core scores) by maximum likelihood estimation. We prove that at any local optimum of the likelihood function, the expected degree of any node under the model is the same as the given network. Therefore, our model can be an alternative to the Chung-Lu model for generating random networks [30].

Our model also accounts for spatial locations of vertices (if such data is available). Spatial networks emerge appear in application such as trade, transportation, the power grid, and the Internet, where there is a cost associated with the length of the edges [31] and are known to carry core-periphery structure [9, 32, 33]. In such cases, topology alone does not explain many proprieties in spatial networks, such as "small world" navigability. Our model accounts for spatial information by decreasing the probability of an edge between a pair of vertices with their distance. We show that at local optima of our likelihood function, the log of the expected geometric mean edge length of networks generated by our model is the same as in the given network from which the parameters are learned.

Spatial information enables us to design efficient algorithms for maximizing likelihood and generating random networks. The main idea is that if a set S of vertices are far away in space from some given vertex u, then the effect of S on u can be efficiently approximated. We perform this approximation in a hierarchical manner similar to the fast multipole method, a "top 10 algorithm of the twentieth century" [34]. Although the algorithm is traditionally used to accelerate N-body simulations in physics, we adapt it here to develop nearly linear time algorithms for likelihood optimization and network sampling. This lets us scale to networks with millions of vertices using a commodity server.

Our model has substantially higher likelihood compared to the two other random graph models that explicitly incorporate core-periphery structure. We also show that the learned core scores are useful for downstream data mining and machine learning tasks; specifically, learned core scores out-perform existing baselines for predicting airport traffic and classifying fungal networks.

2 MODEL AND BASIC INFERENCE

In this section, we develop our model for generating networks with core-periphery structure and a straightforward maximum likelihood procedure to learn model parameters for a given input network. We analyze the basic properties of the model without worrying about computation (efficient algorithms are the focus of Section 3). We provide two technical results about local optima of the likelihood function for our model: (i) the expected degree of each vertex matches the input network and (ii) the expected aggregated log-distance matches the input network.

2.1 A generative core-periphery model

Basic model. In our basic model, we start with n vertices, where each vertex u has a real-valued core score θ_u. For every pair of vertices u and v, we add an edge between them with probability

$$\rho_{uv} = e^{\theta_u + \theta_v} \big/ (e^{\theta_u + \theta_v} + 1). \tag{1}$$

As a sanity check, the edge probability $\rho_{uv} \in [0, 1]$ and increases monotonically as a function of the combined core score. Thus, vertices that are both "in the core" (i.e., have large core scores) are more likely to connect. Two special cases have significant meaning. First, if all vertices have the same core score θ_0, then the model is the Erdős-Rényi model with edge probability $p = e^{2\theta_0}/(e^{2\theta_0} + 1)$. Second, if the vertices are partitioned into a core V_c with core score θ_0 and a periphery V_p with core score $-\theta_0$, then as θ_0 increases, the model converges to a traditional block model for social networks: the core vertices V_c form a clique while the periphery vertices V_p form an independent set, and every pair of vertices $u \in V_c$ and $v \in V_p$ is connected with probability 0.5.

Full model. Now we incorporate spatial information into the model. All of our subsequent algorithms and analysis are then presented for this model, which includes the basic model in Eq. (1) as a special case. Our model incorporates spatial information by adding a kernel function K_{uv} to the denominator of edge probability

$$\rho_{uv} = e^{\theta_u + \theta_v} \big/ (e^{\theta_u + \theta_v} + K_{uv}^{\epsilon}). \tag{2}$$

The kernel function K_{uv} can be any arbitrary non-negative function provided by the user, but we mainly focus on metric kernels to study spatial networks where core-periphery structure is prevalent. We will introduce kernel functions as we go through examples, but a simple kernel is Euclidean distance: $K_{uv} = \|\boldsymbol{x}_u - \boldsymbol{x}_v\|_2$, where $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ are the spatial positions of vertices u and v. We also include a tuning parameter ϵ to control the "nearsightedness" of vertices. When $\epsilon = 0$, the edge probability ρ_{uv} is independent of the edge length, and we recover the basic model. As ϵ increases, the fraction of long distance edges in the generated networks steeply decrease. While the parameter ϵ could be baked into the kernel, we find it useful to optimize ϵ in conjunction with the core scores $\boldsymbol{\theta}$ when fitting the model to network data (see Section 2.2).

Relationship to small worlds. Our model is inspired in part by the Kleinberg navigable small-world model, where random edges are added to a lattice network with probability inversely proportional to powers of distance [35, 36]. In real-world networks that are sparse, we expect the edge probability ρ_{uv} in the corresponding generalized model to often be small. In those cases, the edge probability in our model can be approximated by

$$\rho_{uv} \approx \rho_{uv} \big/ (1 - \rho_{uv}) = e^{\theta_u + \theta_v} \big/ K_{uv}^{\epsilon}. \tag{3}$$

Our model thus has an interpretation for social networks: actors live somewhere in space and have different social statuses; people have a higher chance to connect if they are closer, and individuals with higher social status are likely to have more acquaintances.

2.2 Inference via Likelihood Maximization

For inference, we take as input an undirected network and a kernel function and output a set of real-valued vertex core scores and a real-valued tuning parameter ϵ for the kernel function. To do so, we maximize the log-likelihood Ω:

$$\Omega = \textstyle\sum_{u<v} \left[A_{uv} \log \rho_{uv} + (1 - A_{uv}) \log(1 - \rho_{uv})\right]. \quad (4)$$

The gradient of the edge probability with respect to the model parameters is given by simple closed-form expressions:

$$\frac{\partial \rho_{uv}}{\partial \theta_u} = \rho_{uv}(1 - \rho_{uv}), \quad \frac{\partial \rho_{uv}}{\partial \epsilon} = -\rho_{uv}(1 - \rho_{uv}) \cdot \log K_{uv}. \quad (5)$$

First, we focus on the derivatives of the objective function with respect to the core scores:

$$\begin{aligned} \frac{\partial \Omega}{\partial \theta_w} &= \textstyle\sum_{u<v} \frac{\partial \rho_{uv}}{\partial \theta_w} \left(\frac{A_{uv}}{\rho_{uv}} - \frac{1-A_{uv}}{1-\rho_{uv}} \right) = \sum_{u<v} (\delta_{wu} + \delta_{wv})(A_{uv} - \rho_{uv}) \\ &= \textstyle\sum_{u \neq w} A_{wu} - \sum_{u \neq w} \rho_{wu}. \quad (6) \end{aligned}$$

Here, δ_{ab} is the Kronecker delta function. Note that $\sum_{u \neq w} A_{wu}$ is the degree of vertex w in the given network, while $\sum_{u \neq w} \rho_{wu}$ is the expected degree of vertex w in the model. This observation implies our first theorem about the learned model.

Theorem 1. *At any local maximizer of the objective function Ω with respect to the core scores $\boldsymbol{\theta}$, the expected degree of every vertex in the random model equals the degree in the input network.*

Next, we look at the derivative of the objective function with respect to the tuning parameter ϵ,

$$\begin{aligned} \frac{\partial \Omega}{\partial \epsilon} &= \textstyle\sum_{u<v} \frac{\partial \rho_{uv}}{\partial \epsilon} \left(\frac{A_{uv}}{\rho_{uv}} - \frac{1-A_{uv}}{1-\rho_{uv}} \right) \\ &= -\textstyle\sum_{u<v} \log K_{uv} \cdot \rho_{uv}(1 - \rho_{uv}) \cdot \left(\frac{A_{uv}}{\rho_{uv}} - \frac{1-A_{uv}}{1-\rho_{uv}} \right) \\ &= -\textstyle\sum_{u<v} A_{uv} \log K_{uv} + \sum_{u<v} \rho_{uv} \log K_{uv}. \quad (7) \end{aligned}$$

Let $\sum_{u<v} A_{uv} \log K_{uv}$ be the *aggregated log-distance*, which measures the overall edge length. Then $\sum_{u<v} \rho_{uv} \log K_{uv}$ is the expected aggregated log-distance in networks generated by the learned model. This observation implies our next result.

Theorem 2. *At any local maximizer of the objective function Ω with respect to ϵ, the expected aggregated log-distance in the random model equals the true aggregated log-distance in the input network.*

In other words, optimizing the tuning parameter ϵ forces the overall distances in the model to match the original network.

Next, define the log geometric mean edge length (log-GMEL) of a network as

$$\text{log-GMEL} = \log \left[\textstyle\prod_{u<v \in E} K_{uv}\right]^{\frac{1}{|E|}} = \frac{1}{|E|} \textstyle\sum_{u<v \in E} \log K_{uv} \quad (8)$$

We argue that the log-GMEL of a random network is close to the log-GMEL of the input graph as well. By the law of large numbers, in the limit of large networks, the number of edges $|E|$ model concentrates around its expectation. When the number of edges is sharply concentrated about its expectation, we can approximate the expected log-GMEL by $\mathbf{E}\,[\text{log-GMEL}] \approx \mathbf{E}\left[\sum_{u<v \in E} \log K_{uv}\right] / \mathbf{E}\,[|E|]$, which is the expected aggregated log-distance divided by the expected number of edges. According to Theorems 1 and 2, the expected number of edges and the expected aggregated log-distance equal to those in the input network, respectively. Thus, the expected log-GMEL should roughly be the log-GMEL of the input network. In Section 5, we validate this numerically.

Since the derivatives of the objective function can be evaluated analytically, we use a gradient-based approach to optimize the likelihood. However, the log-likelihood objective Ω is not necessarily convex. Therefore the computed optimal set of parameters $\{\boldsymbol{\theta}^*, \epsilon^*\}$ is not guaranteed to be the global maximizer of Ω. Finally, even though spatial networks are our primary focus in this paper, Theorem 1 still holds for the basic model in Eq. (1). Thus, the basic model can be used for both core-periphery structure detection and as an alternative to the Chung-Lu model for generating random networks with arbitrary sequences of expected degrees.

With a naive implementation of model inference, evaluating the objective function (Eq. (4)) or gradient (Eqs. (6) and (7)) takes $O(|V|^2)$ time, regardless of the choice for the kernel function. Similarly, if we sample a random network by determining the connectivity of every pair of vertices sequentially, the overall cost is also $O(|V|^2)$, even if the generated network is sparse. In the next section, we design nearly linear-time approximation algorithms for both model inference and network generation when the kernel function is a metric that satisfies the triangle inequality.

3 FAST ALGORITHMS

The quadratic scaling of the naive algorithm limits the model's applicability to large-scale networks. In order to resolve this problem, we use a method akin to the fast multipole method (FMM) to exploit structural sparsity in the computations when the kernel is a metric, which results in nearly linear time algorithms by sacrificing a controlled amount of accuracy. The main idea of the approach is that the joint influence of a group of vertices S on a far away group of vertices T can be well approximated for metric kernels.

3.1 Efficient Model Inference

We will use a gradient-based method to optimize the objective function, but the computational bottleneck of optimizing model parameters is the evaluation of the objective function and its gradient. Here, we take advantage of the fact that the expressions for the objective function and its gradient are in close analogy to the gravitational potential and forces in N-body simulations. Similar to the gravitational potential, the objective function Ω as well as its derivative $\partial\Omega/\partial\epsilon$ consists of $O(|V|^2)$ pairwise interactions which decay as a function of distance (assuming a metric kernel). Furthermore, our objective function and derivative also accumulate contributions with respect to the core scores over all pairs of vertices. These similarities motivate us to use ideas from the FMM to exploit the "structural sparsity" in our computations.

Background on the FMM. The FMM is a numerical algorithm for accelerating computation of N-body simulations, which require the (approximate) accumulation of $O(N^2)$ pairwise interactions [37, 38]. In physics, these calculations look like $P(u) = \sum_{v \in S} k(u, v) f(v)$, where P is the potential (e.g., gravitational potential), k is the kernel (e.g., $k(u, v) = 1/\|\boldsymbol{x}_u - \boldsymbol{x}_v\|_2$ in gravitational potential), and f is a weight function, and we want $P(u)$ for all $u \in S$. The key conceptual idea is that the interaction between two groups of well-separated particles can be well-approximated by a *single* interaction between the total mass of the two groups. This gives rise to the "structural sparsity" of the problem. The main mathematical idea is to use a (multipole) expansion of the kernel function to approximate these well-separated interactions [39, 40].

We use use a hierarchical metric-tree decomposition of the spatially distributed vertices. The root of the tree (level $l = 0$) is a metric-ball containing all vertices, and we recursively bisect the network to fit into smaller metric-balls at lower levels (see Figs. 2 and 3). For example, each node at the $l = 1$ level of metric-tree represents a metric-ball that encaptulates half of the vertices in the network, which is further divided into two child nodes at the $l = 2$ level. Leaf nodes in the metric-tree represent a metric-ball that contains only one vertex from the network. We are using the term *node* to refer to metric-balls in this data structure—not to actual vertices in a network; we use *vertex* when referring to graphs.

Evaluating the objective function. In order to exploit the "separation" of far-away interactions, we re-write the objective in Eq. (4) to first separate the vertex pairs that are connected:

$$\begin{aligned}\Omega &= \textstyle\sum_{u<v} [A_{uv} \log \rho_{uv} + (1 - A_{uv}) \log(1 - \rho_{uv})] \\ &= \textstyle\sum_{u<v \in E} \log \rho_{uv} + \sum_{u<v \notin E} \log(1 - \rho_{uv}) \\ &= \textstyle\sum_{u<v \in E} \log \frac{\rho_{uv}}{(1-\rho_{uv})} - \sum_{u<v} \log(1 + e^{\theta_u+\theta_v} / K^{\epsilon}_{uv}).\end{aligned}$$

Recall from Theorem 1 that at any local maximizer of the objective function, the expected degree of a vertex in the model equals its degree in the given network (i.e., $\sum_{u\neq w} \rho_{wu} = \sum_{u\neq w} A_{wu}$). If the given network is sparse, then $\sum_{u\neq w} A_{wu} \ll |V|$ holds for most vertices, $\rho_{uv} \ll 1$ holds for most vertex pairs, and $z_{uv} \equiv e^{\theta_u+\theta_v} / K^{\epsilon}_{uv} = \rho_{uv} / (1 - \rho_{uv}) \ll 1$. In other words, the learned core scores in the given network are small enough that $e^{\theta_u+\theta_v} \ll K_{uv}$ for most pairs of vertices. Thus, we can use the Maclaurin expansion for $\log(1 + z_{uv})$ to approximate the objective function,

$$\Omega \approx \textstyle\sum_{u<v\in E} \log \frac{\rho_{uv}}{1-\rho_{uv}} - \sum_{u<v} \sum_{t=1}^{T} \frac{(-1)^{t-1}}{t} \left(\frac{e^{\theta_u+\theta_v}}{K^{\epsilon}_{uv}}\right)^t.$$

Here, T is a small constant ($T = 4$ in our implementation; larger expansions provided little benefit in accuracy). The first term in the expansion only sums over connected vertex pairs and can be calculated in $O(|E|)$ time. The summation over the $O(|V|^2)$ pairwise interactions in the second term of the expansion is accelerated by grouping vetices in the same metric-ball and computing interactions between two groups of vertices at once. While we have made several approximations in our arguments, our numerical experiments in Section 4.1 show that they are valid on real-world networks.

Our FMM-like implementation is similar to the algorithm presented in the original literature known as the "tree-code" [41]. At each level l of the metric-tree, for every pair of sibling metric-balls I, J, we sum over the interaction between every pair of vertices with one end in I and the other in J,

$$\textstyle\sum_{u<v} \left(\frac{e^{\theta_u+\theta_v}}{K^{\epsilon}_{uv}}\right)^t = \sum_l \sum_{\substack{I<J\in B_l \\ p(I)=p(J)}} \left[\sum_{u\in I, v\in J} \left(\frac{e^{\theta_u+\theta_v}}{K^{\epsilon}_{uv}}\right)^t\right], \quad (9)$$

where B_l represents the set of metric-balls at level l of the metric-tree, and $p(I), p(J)$ denote the parent metric-balls of I, J respectively.

In order to guarantee the accuracy of the FMM algorithm, the pairwise interactions between vertices in I and J (in the square bracket of Eq. (9)) can only be computed at once if the separation between the metric-balls is large relative to their radii, i.e.,

$$K_{IJ} / (r_I + r_J) > \delta_1, \quad (10)$$

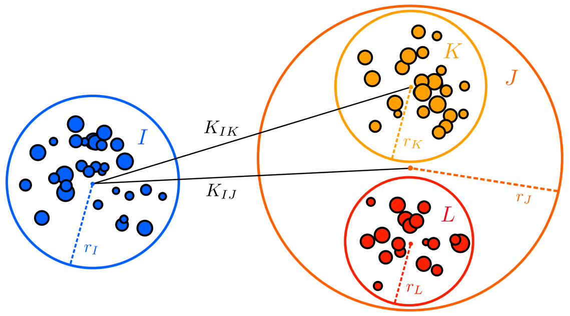

Figure 2: Interactions between vertices in I and J with our algorithm. If the accuracy criteria in Eqs. (10) and (11) are satisfied, then we approximate the interaction between all vertices in I and all vertices in J with a single point in each ball (Eq. (12)). Otherwise, we recurse and consider the interactions between I and K as well as I and L.

where r_I, r_J are the radii of the metric-balls, K_{IJ} measures the metric distance between their centers of mass, and δ_1 is a user-input accuracy. For our problem, we also have to make sure the assumption $z_{uv} \ll 1$ holds for every pair of vertices with one end in I and the other in J, i.e.,

$$e^{\max\{\theta_u \mid u\in I\}} \cdot e^{\max\{\theta_v \mid v\in J\}} / K_{IJ} < \delta_2, \quad (11)$$

where δ_2 is another user-input accuracy parameter. We use $\delta_1 = 2.0$ and $\delta_2 = 0.2$ as default in our implementation. However, the user can tune these parameters to trade off accuracy and computation time (we conduct trade-off experiments in Section 4.1).

If the two accuracy criteria in Eq. (10) and Eq. (11) are satisfied, then the long-range pairwise interactions between I and J can be approximated as follows:

$$\textstyle\sum_{u\in I, v\in J} \left(\frac{e^{\theta_u+\theta_v}}{K^{\epsilon}_{uv}}\right)^t \approx \left(\sum_{u\in I} e^{\theta_u}\right)^t \left(\sum_{v\in J} e^{\theta_v}\right)^t / K^{t\epsilon}_{IJ}. \quad (12)$$

On the other hand, if one or more accuracy criteria are not satisfied, then without loss of generality, I is the metric-ball with smaller radius ($r_I < r_J$), and we compute the interactions between metric-ball I and each of the two child metric-balls of J separately (Fig. 2). The exact computational complexity of the FMM algorithm depends on the parameters δ_1 and δ_2. In practice, the objective function is approximated to within 1% error using the default parameters (see Section 4.1), and the overall time complexity is $O(|E| + |V| \log|V|)$.

Evaluating the gradient. We can also use the FMM to approximate the derivative of Ω with respect to the core score θ_w. At each level l of the metric tree, we sum over the interactions between vertex w and all the vertices in the metric-ball J_l which is the sibling to the metric-ball I_l that contains w,

$$\begin{aligned}\frac{\partial\Omega}{\partial\theta_w} &= \textstyle\sum_{u\neq w} A_{wu} - \sum_{u\neq w} e^{\theta_u+\theta_w} \Big/ \left[e^{\theta_u+\theta_w} + K^{\epsilon}_{uw}\right] \\ &= \textstyle\sum_{u\neq w} A_{wu} - \sum_l \sum_{u\in J_l} e^{\theta_u+\theta_w} \Big/ \left[e^{\theta_u+\theta_w} + K^{\epsilon}_{uw}\right] \\ &\approx \textstyle\sum_{u\neq w} A_{wu} - e^{\theta_w} \sum_{l,u\in J_l} e^{\theta_u} \Big/ \left[\frac{e^{\theta_w}}{|J_l|} \sum_{u\in J_l} e^{\theta_u} + K^{\epsilon}_{I_l J_l}\right].\end{aligned} \quad (13)$$

The first term of Eq. (13) is the degree of vertex w, which can be evaluated upfront in $O(|E|)$ time. Moreover, if we precompute and store $\sum_{u\in J_l} e^{\theta_u}$ for all metric-balls, then the second term can be computed in $O(|V|\log|V|)$ time. Therefore, the overall cost of evaluating the core score derivatives for all the vertices is $O(|V|\log|V| + |E|)$. Furthermore, we can also approximate the derivative of the objective function with respect to ϵ:

$$\frac{\partial\Omega}{\partial\epsilon} = \sum\nolimits_{u<v\in E} -\log K_{uv} + \sum\nolimits_{u<v} \frac{e^{\theta_u+\theta_v}\log K_{uv}}{e^{\theta_u+\theta_v}+K^{\epsilon}_{uv}}$$

$$\sum\nolimits_{u<v\in E} -\log K_{uv} + \sum\nolimits_{l} \sum_{\substack{I<J\in B_l\\ p(I)=p(J)}} \sum\nolimits_{u\in I, v\in J} \frac{e^{\theta_u+\theta_v}\log K_{uv}}{e^{\theta_u+\theta_v}+K^{\epsilon}_{uv}}.$$

The first term in the last equation can be calculated in $O(|E|)$ time, while the second term can be evaluated in $O(|V|\log|V|)$ following the same scheme as the objective function,

$$\sum_{\substack{u\in I\\ v\in J}} \frac{e^{\theta_u+\theta_v}\log K_{uv}}{e^{\theta_u+\theta_v}+K^{\epsilon}_{uv}} \approx \frac{\sum_{u\in I} e^{\theta_u}\cdot\sum_{v\in J} e^{\theta_v}\cdot\log K_{IJ}}{\frac{1}{|I|}\sum_{u\in I} e^{\theta_u}\cdot\frac{1}{|J|}\sum_{v\in J} e^{\theta_v} + K^{\epsilon}_{IJ}}. \quad (14)$$

Unlike the FMM-style approximation for the objective function (Eq. (12)), the denominators in the expression of the objective function gradient (Eqs. (13) and (14)) are not metric kernels due to the extra terms that involve the vertex core scores. However, when the given network is sparse and the fitted core scores are small, the metric kernels $K^{\epsilon}_{I_l J_l}$ and K^{ϵ}_{IJ} dominate the denominators of Eq. (13) and Eq. (14), making them good approximations for metric kernels.

The FMM-style algorithm enables efficient model inference, and the learned core scores are accurate enough for data mining purposes, which we show in Section 5. Furthermore, we also want to generate random spatial networks, which naively takes $O(|V|^2)$ time. We show how to accelerate this process in the next section.

3.2 Efficient Random Network Generation

Given the core scores, we can naively sample a random networks by flipping a biased coin for each pair of vertices. However, this process is inefficient when the output network is sparse. To address this problem, we develop an efficient algorithm to hierarchically sample the edges. The key idea is to sample edges between pairs of metric-balls instead of pairs of vertices.

Our algorithm proceeds as follows. First, we recursively partition the vertices to fit into a metric-tree. Second, at each level of the tree, for every pair of sibling metric-balls I, J, we compute the expected number of edges n_{IJ} that has one end in each of I and J:

$$n_{IJ} \approx \frac{\sum_{u\in I} e^{\theta_u}\cdot\sum_{v\in J} e^{\theta_v}}{\frac{1}{|I|}\sum_{u\in I} e^{\theta_u}\cdot\frac{1}{|J|}\sum_{v\in J} e^{\theta_v} + K^{\epsilon}_{IJ}}. \quad (15)$$

Third, we determine the edges between every pair of sibling metric-balls I, J by sampling n_{IJ} vertices independently from I with probability $\rho_u \propto e^{\theta_u} \Big/ \left[\frac{e^{\theta_u}}{|J|}\sum_{v\in J} e^{\theta_v} + K^{\epsilon}_{IJ}\right]$, and n_{IJ} vertices independently from J with probability $\rho_v \propto e^{\theta_v} \Big/ \left[\frac{e^{\theta_v}}{|I|}\sum_{u\in I} e^{\theta_u} + K^{\epsilon}_{IJ}\right]$. Finally, we pair samples from I and J sequentially. The cost for sampling vertices with non-uniform probability in the third step is $O(|I| + |J| + n_{IJ}\log n_{IJ})$. Assuming the generated network is sparse and the number of edges grows linearly with the number of vertices ($n_{IJ} \propto |I| + |J|$), the overall cost is $O\left(|V|(\log|V|)^2\right)$.

Creating edges by pairing up independently sampled vertices is called "ball-dropping" in Erdős-Rényi sampling [42]. In our model, the number of edges between metric-balls I and J does not have a

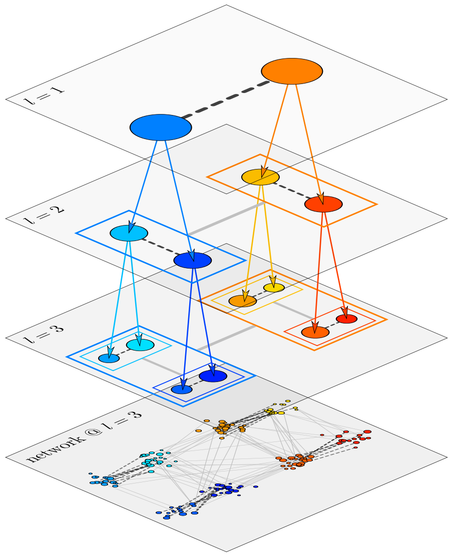

Figure 3: The top three levels in the metric-tree data structure. Each circle in the diagram represents a metric-ball that encapsulates spatially adjacent vertices. This data structure is used for efficient evaluation of the objective function and its gradient as well as for efficient network generation. For objective function evaluation, the dashed lines at each layer of the diagram represent the pairwise interactions accumulated at the corresponding level of the metric-tree. For random network generation, the dashed lines represent the edges sampled at each level of the metric-tree between sibling metric-balls, while the solid lines represent edges that are already generated in previous levels. The bottom layer of the diagram shows the random network generated at the $l = 3$ level of the metric tree; edges between vertices within the same metric ball at this level are not yet considered.

closed-formed probability distribution, and our algorithm is only taking the mean of that distribution by sampling n_{IJ} edges. Therefore, our fast sampling scheme is only an approximation without any theoretical guarantees. However, as we show in the next section, this approximation empirically preserves network properties.

4 METHODOLOGICAL EXPERIMENTS

In this section, we first numerically validate our approximations and then show that our model achieves much higher likelihood than competing generative models for core-periphery structure. Section 5 then explores data mining tasks aided by our model.

4.1 Approximation validation

In the previous section, we made a number of approximations in our methods. We now show that these approximations indeed have small error. To optimize the objective function, we use the limited-memory BFGS (L-BFGS) algorithm from the `Optim.jl` package [43]. We also use a diagonal preconditioner in case the approximated Hessian used within L-BFGS becomes ill-conditioned. We construct the metric-tree with the `NearestNeighbors.jl` package. Finally, we set the tolerance parameters to $\delta_1 = 2.0$ and $\delta_2 = 0.2$.[1]

[1]Our software is available at https://github.com/000Justin000/spatial_core_periphery.

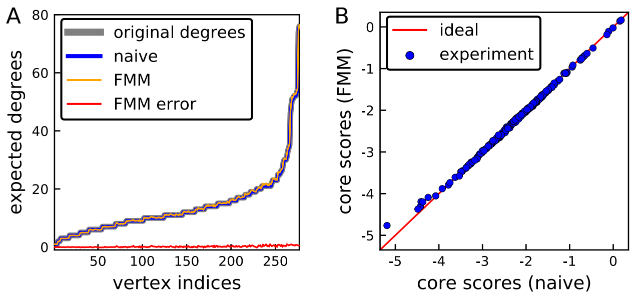

Figure 4: (A) Agreement between the naive (blue) and fast (orange) algorithms on the expected vertex degrees. The error of the FMM-style approximation is in red. We also include the vertex degrees in the original network (gray) as numerical validation for Theorem 1. (B) Correlation between the learned core scores from the naive algorithm and the FMM algorithm. The Pearson correlation coefficient is 0.999.

Accuracy in evaluating the objective function and derivative. To test our fast algorithm, we first learn parameters for the *C. elegans* network from Fig. 1 using the naive algorithm and the Euclidean distance kernel. Next, we evaluate Ω and the gradient using the same parameters, but this time using the fast algorithm to sum over the pairwise interactions between vertices. The relative difference in Ω between the naive and the fast algorithms was less than 1%. To evalualuate agreement in the derivative, we compare expected degrees obtained by the fast algorithm and the naive algorithm and find them to be nearly identical (Fig. 4A). As a sanity check, the expected vertex degrees computed by both algorithms are very close to the vertex degrees in the original network, which is a necessary condition for Ω to be at a local maximum (Theorem 1).

To understand how the quality of the approximation depends on the input accuracy parameters, we repeat the above experiments using different combinations of δ_1 and δ_2. We characterize the approximation error with the root mean square error in expected degrees between the fast algorithm and the naive algorithm (Fig. 6A) and report dependence of the running time on the accuracy parameters (Fig. 6B). Both accuracy and running time have a stronger dependency on δ_2 as compared to δ_1. This reason is that when our algorithm determines the radius of the parent metric-ball, it assumes the worst-case condition and chooses a large value. Therefore in most cases, $\delta_1 = 1.0$ could already guarantee that all the vertices between two metric-balls are well-separated.

Next, we use our fast algorithm to learn the model parameters. The learned vertex core scores with the naive and the fast algorithms had Pearson correlation of 0.999 and empirically look similar (Fig. 4B). Furthermore, the learned ϵ by the naive algorithm and the FMM algorithm were 0.499 and 0.506, respectively.

Quality of generated random networks. Using the model parameters learned by the naive algorithm, we now confirm that random networks generated by the naive sampling algorithm and the fast sampling algorithm are similar. We first consider the degree distribution. For both the naive algorithm and the fast algorithms, we compare the mean degrees over three sampled networks to the degree in the original network (Fig. 6A). Indeed, the degrees in the random networks generated by both algorithms are highly correlated with that in the original network.

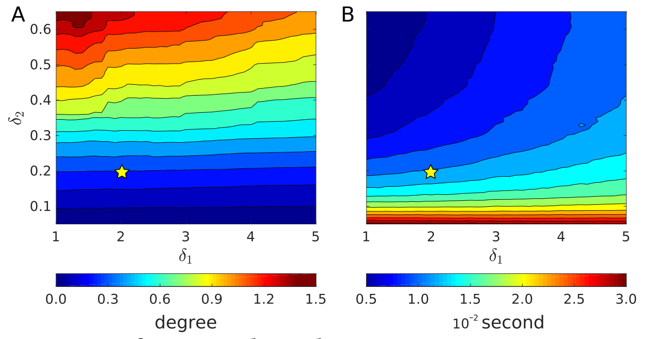

Figure 5: Performance dependence on accuracy parameters on the *C. elegans* network. The yellow stars represent the location of default parameters. (A) Root mean square error in expected degrees. (B) Running time per gradient evaluation.

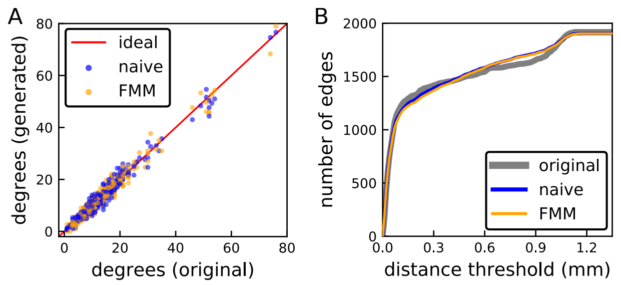

Figure 6: (A) Correlation of vertex degrees in *C. elegans* and the random networks generated by the naive and fast algorithm; the Pearson correlation coefficients are 0.983 and 0.981, respectively. (B) Number of edges whose kernel distance is below a distance threshold. The agreement numerically validates our fast sampling algorithm as well as our arguments in Section 3 on preserving distances.

Next, we consider the edge length distribution given by the kernel function. We argued that the expected log-GMEL in the generated random networks equals that in the original network at any local maximum of the likelihood function. Indeed, the GMEL in *C. elegans* is 0.079mm, while the GMEL in the random networks generated by the naive algorithm and the fast algorithm (averaged over three samples) are 0.078mm and 0.079mm. For a more detailed analysis, we picked equispaced distance thresholds from 0.0mm to 1.4mm and counted the number of edges in the network whose length is smaller than each threshold. The counts of the original network, the mean of three naive random samples, and the mean of three samples with our fast algorithm are nearly identical (Fig. 6B).

Scalability of the FMM-style algorithm. Finally, we validate the computational complexity of our proposed methods. We run our algorithms on synthetic 2-block core-periphery networks where vertex coordinates are randomly generated in the 2-dimensional unit square. We choose 5% of the vertices in the networks as the core, and their core score (denoted as θ_c) is set to be 1.0 unit larger than the core score of the periphery vertices (denoted as θ_p). The core scores θ_c, θ_p in this family of networks are chosen so that the number of edges grows linearly with the number of vertices (in the five networks, $\theta_c = -1.25, -2.77, -4.13, -5.43, -6.69$). Fig. 7 shows that the empirical running time follows the theoretical analysis.

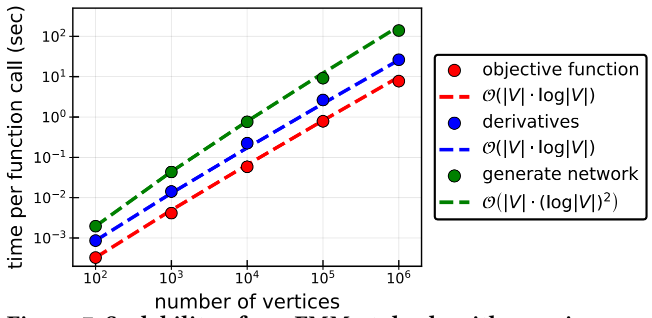

Figure 7: Scalability of our FMM-style algorithms using synthetic networks, where the average degree in each network is 10. Observed timings are scattered with circles, and ideal efficiencies are plotted in dashed lines.

Table 1: Number of nodes (n), number of edges (m), and optimized log-likelihood of our model, SBM-CP and logistic-CP. In all datasets, our model has a larger likelihood. (Results for logistic-CP are only available for connected graphs.)

			Log likelihood		
Dataset	n	m	our model	SBM-CP	logistic-CP
C. elegans	279	1.9K	$-6.3 \cdot 10^3$	$-7.1 \cdot 10^3$	$-7.0 \cdot 10^3$
London Under.	315	370	$-6.0 \cdot 10^2$	$-2.2 \cdot 10^3$	$-2.1 \cdot 10^3$
Pv_M_I_U_N_42d_1	2.4k	2.6K	$-6.4 \cdot 10^3$	$-2.1 \cdot 10^4$	—
OpenFlights	7.2K	18.6K	$-4.7 \cdot 10^4$	$-1.1 \cdot 10^5$	—
Brightkite	50.7K	194K	$-1.3 \cdot 10^6$	$-1.9 \cdot 10^6$	—
LiveJournal	1.16M	7.19M	$-7.5 \cdot 10^7$	$-8.9 \cdot 10^7$	—

4.2 Likelihood comparison

We now demonstrate the quality of our model by comparing its optimized log-likelihood with the only other two generative models for core-periphery structure. The first is a 2-block stochastic block model with a belief propagation learning algorithm that explicitly incorporates core-periphery structure (henceforth, SBM-CP) [12]. Since the belief propagation algorithm is sensitive to the initial conditions, we perform 5 independent runs of the algorithm from different random initial conditions and record the best result. The second model defines the edge probability as a logistic function of the "centrality rank" of the incident vertices (henceforth, logistic-CP) [44]. For this model, we tune the hyperparameters s, t in the logistic function with grid search and record the best result.

We learn the parameters of our model, SBM-CP and logistic-CP on six datasets (basic summary statistics are listed in Table 1):
(i) The neural network of the nematode worm *C. elegans* from Fig. 1.
(ii) The network of the London underground transportation system, where vertices are tube stations and edges connect stations.[2]
(iii) The fungal network Pv_M_I_U_N_42d_1 constructed from a laboratory experiment, where each hypha (tubular cell) is a vertex, and each cord transporting nutrients between two hyphae is an edge [46]. The fungus grows on a 2-dimensional plane.
(iv) The airline transportation network from OpenFlights.[3] Vertices are airports with latitude and longitude spatial locations, and edges correspond to available direct flights.
(v) The Brightkite social network [47]. Vertices are users and spatial locations come from the user's most recent check-in.
(vi) The LiveJournal social network [48]. Vertices are bloggers and the spatial locations are given by the city listed in the user's profile. Edges are friendships listed in a user's profile.

A small amount of noise was added to locations in the Brightkite and LiveJournal datasets to avoid having vertices in the same location. For the OpenFlights, Brightkite, and LiveJournal datasets, spatial positions are latitude and longitude, so we use the great circle distance kernel. For *C. elegans* and the fungal network, we use Euclidean distance. In the London Underground dataset, tube stations are almost always connected to their closest spatial neighbors due to construction cost of railways. Therefore, we choose a symmetric version of rank distance [48] as the kernel. Table 1 lists the optimized log-likelihoods of our model, SBM-CP and logistic-CP; our model always has substantially higher likelihood.

[2]The network was derived based on a similar one of Rombach et al. [45]. Vertex coordinates were collected from Wikipedia.
[3]Data collected from https://openflights.org/data.html#route.

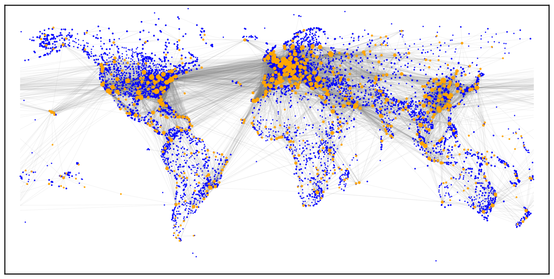

Figure 8: Vertex core scores in the airline network. The top 10% of the vertices with highest core scores are colored in orange, while the rest are colored in blue. The radius of a vertex is proportional to the square root of its degree.

5 DATA MINING EXPERIMENTS

Now that we have addressed the accuracy, scalability, and efficacy of our algorithms and model, we use the output of our model to study real-world spatial networks. We find that the learned core scores from our model are good predictors for traffic in an airport network and for the classification of fungal networks.

5.1 Case study I: Predicting enplanement

We first analyze the OpenFlights network introduced in Section 4.2. Recall that this network has vertices representing airports and edges representing direct flights between airports. Given the network and vertex coordinates, our goal is to predict the total number of passengers boarding (enplanement) at each airport in the year 2017 using vertex core scores. To this end, we first compute the vertex core scores using the FMM-style algorithm with the great-circle distance kernel (Fig. 8) and then we use a decision tree to correlate the vertex core scores with enplanement (Fig. 9)

We obtained the enplanement metadata for 447 airports in the United States.[4] We first split the airports randomly into a training set of 357 airports (80%) and test set of 90 airports (20%); after, we build a decision tree using the training set and test its prediction accuracy on the testing set. Figure 9 shows an instance of the decision tree we build using the training set.

[4]Data collected from https://www.faa.gov/airports/planning_capacity.

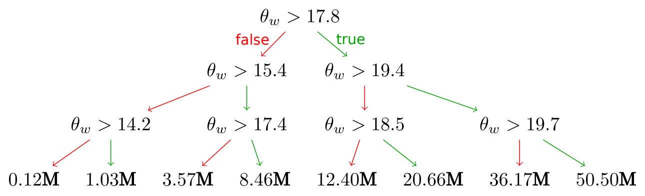

Figure 9: Top three levels of the decision tree for predicting airport enplanement using vertex core scores. The leaf nodes in the decision tree is the average enplanement (in millions) among airports with core scores in the given range.

Table 2: The R^2 values between ground truth and predicted enplanement with different features. Our learned core scores have the largest value.

	degree	BC	CC	EC	PR	core score
R^2	0.762	0.293	0.663	0.542	0.637	**0.846**

We average the prediction accuracy of decision trees over 10 random split of the training and testing set, and the mean coefficient of determination (R^2) between ground truth and predicted enplanement is 0.846. Analogous experiments are repeated using vertex degrees, betweenness centrality (BC) [49], closeness centrality (CC) [50], eigenvector centrality (EC) [51] or PageRank (PR) [52] instead of the core score as the independent variable (Table 2). The vertex core scores outperforms other centrality measures in characterizing airport enplanement. This high accuracy indicates that our model effectively utilizes spatial information.

5.2 Case study II: Classifying fungal networks

In this case study, we use core scores to predict types of fungal networks (the network examined in Section 4.2 is one such network). In these networks, vertices are hyphae in 2-dimensional Euclidean space and edges are cords transporting nutrients between two hyphae. Based on the species and growing conditions, the fungal networks were categorized into 15 different classes [46]. Here, we use core scores as features to predict the class label of each network.

First, we learned the vertex core scores for every fungal network using our model with the Euclidean distance kernel and the FMM-style algorithm. Then, for each network, we created a 4-element feature vector using the maximum, mean, and standard deviation of the core scores, along with the number of vertices. We trained a logistic regression model with these features (and an intercept term) and evaluated prediction accuracy with 5-fold cross validation. We repeated the experiment using the centrality measures from Section 5.1. Table 3 shows that core scores from our model are the best predictors. As evidence for why this might be true, Fig. 10 plots the mean and maximal core score for each network, along with the class labels. We see class separation with these two features.

Furthermore, we use a logistic regression model by concatenating information from vertex degrees and core scores as features. In other words, the predictors are the maximum, mean and standard deviation of vertex degrees; the maximum, mean, and standard deviation of vertex core scores; and the number of vertices. We observe a large increase in prediction accuracy from 43.5% to 65.6%. This result shows that vertex core scores captures different information than degrees in the fungal networks.

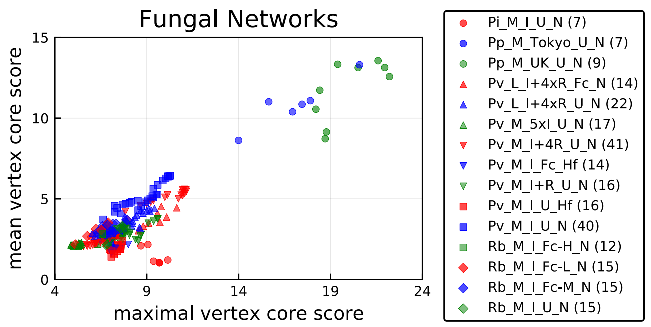

Figure 10: Maximal and mean vertex core scores for 260 fungal networks. The integer in the parentheses after each class label is the number of networks in that class. The plot shows some clustering of class type based on these features. We use these features along with a couple others derived from the learned core scores to train a logistic regression classifier with good performance (Table 3).

Table 3: Cross validation accuracy of fungal networks classification using different features. Our learned core scores give the best prediction. "Random" is random guessing.

random	degree	BC	CC	EC	PR	core score
6.7%	38.0%	23.7%	19.0%	20.2%	18.6%	**43.5%**

6 ADDITIONAL RELATED WORK

The idea of core-periphery structure has a long history in the theory of social networks [53–55], where the core arises due to differential status. Borgatti and Everett developed the first significant computational approach to identify core-periphery structure [55]. Since then, several methods have been deisgned to find core-periphery structure, based on paths [32, 56], vertex covers [57], spectral information [13, 58], *k*-cores and generalizations [59, 60], hand-crafted "core quality" objective functions [8, 45, 61], and user studies [62]. Such methods are typically analyzed via synthetic benchmarks, density measures, or network visualizations. The key difference with our work is that we propose a *generative model*, whereas prior work performs *post hoc identification* of core-periphery structure using the network topology. The only other generative models are due to Zhang et al. [12] and Tudisco and Higham [44] against which we compared in Section 4.2 (unlike our approach, these methods do not use spatial information).

Network centrality captures similar properties to core-periphery structure. Indeed, closeness centrality is used to detect core vertices in networks [11, 14], and centrality measures are baselines for core-periphery identification measures [8] (see also our experiments in Section 5). A subtlety is that network centrality can be a consequence of core-periphery dynamics, but centrality in and of itself does not lead to a *reason* for core-periphery structure. This is one reason why we developed a generative model in this paper.

7 DISCUSSION

We have developed a random network model for core-periphery structure, where each vertex has a real-valued core score. We focused our analysis on spatial data, which connects to the small-world model where edge probabilities are inversely correlated with distance. The spatial structure enabled us to develop fast algorithms for both gradient-based inference and network sampling. We showed both theoretically and numerically that our model preserves the expected degree of each vertex as well as the aggregated log-distance. Our model can be an alternative to the Chung-Lu model, even if there is no spatial metadata. Furthermore, our model does not need to learn from a network—given a prescribed set of core scores and a kernel function, we can generate networks.

In terms of likelihood, our model out-performs the only other generative models for core-periphery structure (SBM-CP and logistic-CP). The basic version of our model can be thought of as a continuous relaxation of SBM-CP, since the core scores permit a continuum of edge probabilities. Finally, we also demonstrated that the learned core scores are useful vertex features for downstream network analysis and machine learning tasks in two very different complex systems (airport traffic and fungal growth), which provides evidence that our model could be incorporated into a wide area of application domains.

ACKNOWLEDGMENTS

We thank David Liben-Nowell for providing the LiveJournal data. This research was supported in part by NSF Award DMS-1830274.

REFERENCES

[1] M. Newman, *Networks: An Introduction.* Oxford University Press, 1 ed., 2010.
[2] D. Easley and J. Kleinberg, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World.* New York, NY, USA: Cambridge University Press, 2010.
[3] A. Barabási and M. Pósfai, *Network Science.* Cambridge University Press, 2016.
[4] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, 2007.
[5] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis," *PRE*, vol. 80, p. 056117, 2009.
[6] S. Fortunato, "Community detection in graphs," *Phys. Reports*, 2010.
[7] C. Moore, "The computer science and physics of community detection: Landscapes, phase transitions, and hardness," *Bulletin of the EATCS*, vol. 121, 2017.
[8] P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha, "Core-periphery structure in networks (revisited)," *SIAM Review*, vol. 59, no. 3, pp. 619–646, 2017.
[9] F. Rossa, F. Dercole, and C. Piccardi, "Profiling core-periphery network structure by random walkers," *Scientific Reports*, vol. 3, p. 1467, 2013.
[10] J. P. Boyd, W. J. Fitzgerald, M. C. Mahutga, and D. A. Smith, "Computing continuous core/periphery structures for social relations data with MINRES/SVD," *Social Networks*, vol. 32, no. 2, pp. 125 – 137, 2010.
[11] P. Holme, "Core-periphery organization of complex networks," *PRE*, 2005.
[12] X. Zhang, T. Martin, and M. E. J. Newman, "Identification of core-periphery structure in networks," *PRE*, vol. 91, p. 032803, 2015.
[13] M. Cucuringu, P. Rombach, S. H. Lee, and M. A. Porter, "Detection of core-periphery structure in networks using spectral methods and geodesic paths," *European Journal of Applied Mathematics*, vol. 27, no. 6, pp. 846–887, 2016.
[14] M. R. da Silva, H. Ma, and A.-P. Zeng, "Centrality, network capacity, and modularity as parameters to analyze the core-periphery structure in metabolic networks," *Proceedings of the IEEE*, vol. 96, pp. 1411–1420, 2008.
[15] E. S. Wellhofer, "Core and periphery: Territorial dimensions in politics," *Urban Studies*, vol. 26, no. 3, pp. 340–355, 1989.
[16] S. Varier and M. Kaiser, "Neural development features: Spatio-temporal development of the caenorhabditis elegans neuronal network," *PLOS Computational Biology*, vol. 7, no. 1, pp. 1–9, 2011.
[17] T. Verma, F. Russmann, N. A. M. Araújo, J. Nagler, and H. J. Herrmann, "Emergence of core-peripheries in networks," *Nature Communications*, vol. 7, p. 10441, 2016.
[18] T. Verma, N. A. M. Araújo, and H. J. Herrmann, "Revealing the structure of the world airline network," *Scientific Reports*, vol. 4, p. 5638, 2014.
[19] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech.: Theory and Experiment*, 2008.
[20] M. E. J. Newman, "Random graphs as models of networks," *arXiv:cond-mat/0202208*, 2002.
[21] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM TKDD*, 2007.
[22] M. Middendorf, E. Ziv, and C. H. Wiggins, "Inferring network mechanisms: The drosophila melanogaster protein interaction network," *PNAS*, 2005.
[23] A. Goldenberg, A. X. Zheng, S. E. Fienberg, and E. M. Airoldi, "A survey of statistical network models," *Found. Trends Mach. Learn.*, 2010.
[24] H. Yin, A. R. Benson, and J. Leskovec, "Higher-order clustering in networks," *PRE*, vol. 97, p. 052306, 2018.
[25] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social Networks*, vol. 5, no. 2, pp. 109–137, 1983.
[26] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," *JMLR*, 2008.
[27] T. P. Peixoto, "Bayesian stochastic blockmodeling," *arXiv:1705.10225*, 2017.
[28] J. Yang and J. Leskovec, "Overlapping community detection at scale: a nonnegative matrix factorization approach," in *Proceedings of WSDM*, ACM, 2013.
[29] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *International Conference on Data Mining*, IEEE, 2013.
[30] W. Aiello, F. Chung, and L. Lu, "A random graph model for power law graphs," *Experiment. Math.*, vol. 10, no. 1, pp. 53–66, 2001.
[31] M. Barthélemy, "Spatial networks," *Phys. Rep*, vol. 499, pp. 1–101, 2011.
[32] S. H. Lee, M. Cucuringu, and M. A. Porter, "Density-based and transport-based core-periphery structures in networks," *PRE*, vol. 89, p. 032810, 2014.
[33] F. Dong, D. Liu, J. Wu, L. Ke, C. Song, H. Wang, and Z. Zhu, "Constructing core backbone network based on survivability of power grid," *JEPE*, 2015.
[34] B. A. Cipra, "The best of the 20th century: Editors name top 10 algorithms," *SIAM News*, vol. 33, no. 4, pp. 1–2, 2000.
[35] J. M. Kleinberg, "Navigation in a small world," *Nature*, 2000.
[36] J. M. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proceedings of STOC*, 2000.
[37] V. Rokhlin, "Rapid solution of integral equations of classical potential theory," *Journal of computational physics*, vol. 60, no. 2, pp. 187–207, 1985.
[38] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *Journal of computational physics*, vol. 73, no. 2, pp. 325–348, 1987.
[39] L. Ying, "A pedestrian introduction to fast multipole methods," *Science China Mathematics*, vol. 55, no. 5, pp. 1043–1051, 2012.
[40] R. Beatson and L. Greengard, "A short course on fast multipole methods," *Wavelets, multilevel methods and elliptic PDEs*, vol. 1, pp. 1–37, 1997.
[41] A. W. Appel, "An efficient program for many-body simulation," *SIAM Journal on Scientific and Statistical Computing*, vol. 6, no. 1, pp. 85–103, 1985.
[42] A. S. Ramani, N. Eikmeier, and D. F. Gleich, "Coin-flipping, ball-dropping, and grass-hopping for generating random graphs from matrices of edge probabilities," *CoRR*, vol. abs/1709.03438, 2017.
[43] P. K. Mogensen and A. N. Riseth, "Optim: A mathematical optimization package for Julia," *Journal of Open Source Software*, vol. 3, no. 24, p. 615, 2018.
[44] F. Tudisco and D. J. Higham, "A nonlinear spectral method for core–periphery detection in networks," *arXiv:1804.09820*, 2018.
[45] M. P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha, "Core-periphery structure in networks," *SIAM J. on Applied Mathematics*, 2014.
[46] S. H. Lee, M. D. Fricker, and M. A. Porter, "Mesoscale analyses of fungal networks as an approach for quantifying phenotypic traits," *J. Complex Networks*, 2017.
[47] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *KDD*, ACM, 2011.
[48] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins, "Geographic routing in social networks," *PNAS*, 2005.
[49] L. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, 1977.
[50] A. Bavelas, "Communication patterns in task-oriented groups," *JASA*, 1950.
[51] M. Newman, "The structure and function of complex networks," *SIREV*, 2003.
[52] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," in *WWW*, 1998.
[53] R. S. Burt, "Positions in networks," *Social Forces*, vol. 55, no. 1, p. 93, 1976.
[54] R. Breiger, "Structures of economic interdependence among nations," *Continuities in structural inquiry*, pp. 353–380, 1981.
[55] S. P. Borgatti and M. G. Everett, "Models of core/periphery structures," *Social Networks*, vol. 21, no. 4, pp. 375 – 395, 2000.
[56] J. Gamble, H. Chintakunta, A. Wilkerson, and H. Krim, "Node dominance: Revealing community and core-periphery structure in social networks," *IEEE Transactions on Signal and Information Processing over Networks*, 2016.
[57] A. R. Benson and J. Kleinberg, "Found graph data and planted vertex covers," in *Advances in Neural Information Processing Systems*, 2018.
[58] C. Ma, B.-B. Xiang, H.-S. Chen, M. Small, and H.-F. Zhang, "Detection of core-periphery structure in networks based on 3-tuple motifs," *Chaos*, 2018.
[59] B. Dumba and Z.-L. Zhang, "Uncovering the nucleus of social networks," in *Proceedings of the 10th ACM Conference on Web Science*, pp. 37–46, ACM, 2018.
[60] A. E. Sariyuce, C. Seshadhri, A. Pinar, and U. V. Catalyurek, "Finding the hierarchy of dense subgraphs using nucleus decompositions," in *WWW*, 2015.
[61] S. Kojaku and N. Masuda, "Core-periphery structure requires something else in the network," *New Journal of Physics*, vol. 20, no. 4, p. 043012, 2018.
[62] M. Joblin, S. Apel, C. Hunsen, and W. Mauerer, "Classifying developers into core and peripheral: An empirical study on count and network metrics," in *ICSE*, 2017.