Generative Graph Models based on Laplacian Spectra*

Alana Shine
University of Southern California

David Kempe
University of Southern California

ABSTRACT

We present techniques for generating random graphs whose Laplacian spectrum approximately matches that of a given input graph. The motivation for matching the Laplacian spectrum is that it naturally encodes high-level connectivity information about the input graph; most existing models (e.g., variants of the Configuration Model, Stochastic Block Model, or Kronecker Graphs) focus on local structure or limited high-level partitions.

Our techniques succeed in matching the spectrum of the input graph more closely than the benchmark models. We also evaluate our generative model using other global and local properties, including shortest path distances, betweenness centrality, degree distribution, and clustering coefficients. The graphs produced by our model almost always match the input graph better than those produced by the benchmark models with respect to shortest path distance and clustering coefficient distributions. The performance on betweenness centrality is comparable to the benchmarks, while a worse match on the degree distribution is a price our method pays for more global similarity.

Our results suggest that focusing on spectral properties may lead to good performance for other global properties, at a modest loss in local similarity. Since global connectivity patterns are usually more important than local features for processes such as information flow, spread of epidemics, routing, etc., our main goal is to advocate for a shift in focus from graph generative models matching local properties to those matching global connectivity patterns.

CCS CONCEPTS

• **Mathematics of computing → Spectra of graphs**; Probabilistic algorithms; • **Theory of computation → Generating random combinatorial structures**; *Linear programming*.

KEYWORDS

Random graphs; Spectra of graphs; Generative model

ACM Reference Format:
Alana Shine and David Kempe. 2019. Generative Graph Models based on Laplacian Spectra. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313631

*Our implementation is online at https://github.com/alanadakotashine/spectral_generation.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313631

1 INTRODUCTION

Random graph models for generating "realistic-looking" graphs play a number of important roles in network science and network data mining. Their uses include:

- Produce graphs to evaluate the computational performance of algorithms or protocols (see, e.g., [32]).
- Produce graphs to evaluate the correctness or quality of the *output* of algorithms in a controlled setting.
- Formally prove that certain graphs are likely to have certain properties, and derive (descriptive or prescriptive) insights about real-world networks from these proofs.

In all cases, the insights gained from the model can only be useful if the generated graphs resemble the class of real-world networks about which one is aiming to make statements [46]. More specifically, this resemblance must apply to graph parameters that are *relevant* to the kinds of structures or processes that the algorithm depends on, or the theorem reasons about.

To illustrate this issue, consider the following example. A researcher seeks insight into the spread of epidemics in social networks, many of which he has observed to follow particular degree distributions. He proves (or observes from simulations) that when a graph is drawn uniformly at random subject to this degree distribution, with high probability, an epidemic will spread very fast on it. He concludes that social networks are very susceptible to epidemics.

The concern with this inference is that social networks may have additional properties beyond the observed degree distribution; these properties may be exceedingly rare among the sampled graphs. For instance, it is well known (see, e.g., [8, 15]) that even subject to most fixed degree distributions, almost every graph has high expansion; in turn, high expansion facilitates epidemic spread under many natural models [6, 26]. On the other hand, high expansion is also nearly synonymous with a lack of pronounced community structure, whereas social networks typically do have pronounced community structure. As a result, the observation that epidemics spread fast on a random graph subject to a given degree distribution is likely more due to the expansion of random graphs than the given degree distribution.

With some exceptions (e.g., [52, 53, 55]) discussed below, most[1] generative graph models draw a graph (nearly) uniformly from an ensemble, subject to local properties such as (expected) degrees, joint degrees, or triangle counts, and/or a global partition into communities [5, 8, 14, 15, 27, 29, 30, 34, 41, 43, 44, 46, 49]. Going beyond those basic models, exponential random graph models (ERGMs) [24, 47] draw graphs from a distribution that rewards or penalizes user-specified local features; the Kronecker Graph Model [36] allows recursive specification of "community" structure in a generalization of stochastic block models (SBM). Another line of work

[1] An orthogonal class of models prescribe a generative (growing or rewiring) process based upon a plausible real-world dynamic [4, 37, 50].

uses deep neural networks (DNNs) to generate graphs. The hope is that using DNNs, one can avoid explicitly specifying the (local or global) properties one desires to replicate, instead learning them automatically from a training data set of enough "similar" graphs [17, 39, 54, 56]. An exception is the NetGAN approach [7]. NetGAN trains a deep network from a single input graph, with the goal of reproducing "realistic-looking" random walks, i.e., sequences of node visits. From these sequences, NetGAN then extracts empirical edge frequencies, and uses these to generate random graphs resembling the input graph.

While most constraints imposed by such models are typically[2] *local* in nature, most network properties and dynamics of interest to researchers are *global*. These include the spread of epidemics or information, the prevalence or strength of communities in graphs, the distribution of distances between individuals, or the capacity for information flow, among many others. These graph properties are all closely affected by the graph's expansion; hence, if the random graph model overwhelmingly generates expander graphs, the effects due to other parameters may be overshadowed.

The main high-level point of this paper is the following: *If one is interested in global network properties, then a generative model should try to match these global properties with those of the real-world networks the model is trying to mimic.*

1.1 Approximately Matching Graph Spectra

Our approach to "matching global structure" focuses on the *spectrum* (eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$) of the (normalized) graph Laplacian of the input graph. (Definitions of all key concepts are given in Section 2.) The spectrum of the graph Laplacian is known to capture connectivity properties [21, 23]. Best known is Cheeger's Inequality [13], which relates the spectral gap λ_2 with the graph's conductance. More intricate results [13, 35] show that small values of $\lambda_3, \lambda_4, \ldots$ indicate additional node sets that are sparsely connected to each other.

Concretely, given an (undirected) input graph G^* with spectrum $0 = \lambda_1^* \leq \lambda_2^* \leq \cdots \leq \lambda_n^*$, our goal is to generate random graphs G with spectrum $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ such that $\lambda_i \approx \lambda_i^*$ for all i. The intuition is that a similar spectrum should result in similar global connectivity properties.[3]

We propose intricate heuristics for randomly generating such graphs, and evaluate the heuristics extensively. The central viewpoint exploited throughout is that a (symmetric) matrix is naturally characterized by its spectrum and orthonormal eigenbasis. Since the spectrum is given, finding a graph Laplacian reduces to finding a suitable orthonormal basis. The problem is that for most bases, the resulting "graph adjacency matrix" A will have entries far from the set $\{0, 1\}$. Our approach is to produce the actual adjacency matrix gradually, using the following steps (detailed in Section 3):

(1) Sample a random orthonormal basis to construct a candidate Laplacian matrix with the desired spectrum. Use linear programming to find a matrix M whose Laplacian is approximately the candidate matrix.
(2) Keeping the spectrum fixed, perform a walk on possible orthonormal bases, guided by an objective function that pushes the entries of M close to 0 or 1.
(3) Use an LP-based algorithm to minimally perturb the entries of M to end up inside $[0, 1]$, while maintaining the row sums. This may perturb the spectrum, but the LP's objective is designed to keep the perturbation small.
(4) Use a combination of deterministic rounding across eigencuts with small eigenvalues and randomized rounding of other edges to round M to a graph G.

In Section 7, we evaluate this algorithm on multiple real-world networks. We generate samples using our approach, and compare them to the graphs generated according to the configuration model, stochastic block model, and Kronecker graph model. The comparison is based on the following global and local graph parameters: the spectrum itself, betweenness centralities, clustering coefficients, degree distribution, and the distribution of shortest path lengths.

We observe (not surprisingly) that the spectrum is in fact matched significantly better by our approach. For the other metrics, the spectral approach typically performs comparably (sometimes slightly better, sometimes slightly worse) to the best of the other approaches; for shortest path lengths, it outperforms the other models significantly most of the time.

Our work raises many concrete questions and big-picture directions for future work, discussed in more detail in Section 8. Perhaps most immediately, our algorithms currently scale to graphs of a few thousand nodes, but not beyond. In order to apply them to realistic networks, it is important to speed up the generation by several orders of magnitude. We therefore view this work not as a complete or final solution to the problem, but as a first step. Our goal in this work was to show that (1) it is computationally feasible to generate graphs matching a given graph's spectrum fairly well, and (2) doing so also helps in matching other important global graph properties. We expect future work to yield faster heuristics and different approaches which will allow these ideas to be applied to even more practical instances. Beyond the running time of the algorithm, there are also many natural questions for future work regarding provable guarantees for the objectives we pursue. Our algorithms do not come with provable approximation guarantees, and obtaining such guarantees would be desirable — though it also appears very challenging.

1.2 Related Work

Our goal is directly related to *inverse eigenvalue problems*: constructing matrices with a given spectrum, subject to other structural properties [12]. Ours is an *inexact* inverse eigenvalue problem, because the spectrum only needs to be matched approximately. On the other hand, our task is to generate a *diverse* collection of matrices, making a deterministic construction inadequate.

[2] The SBM prescribes a high-level partition, but typically only into a constant number of partitions; the recursive structure prescribed by the Kronecker model is more intricate, but depends only on a small number of parameters.

[3] It may appear natural to aim to match the eigenvalues exactly: $\lambda_i = \lambda_i^*$ for all (or some) i. There are two reasons to prefer an approximation instead. The first is computational — we are not aware of any way to efficiently decide if *any* graph has a given spectrum, let alone find one. Second, while there are non-isomorphic pairs of graphs with the same spectrum, such pairs are exceedingly rare: for example, Butler and Grout [10], based on computational enumeration, estimate that only roughly one in 10^{23} graphs has a non-isomorphic cospectral graph. Thus, most of the time, a generative model would be forced to always return the input graph G^*.

Typical structural properties considered in the context of inverse eigenvalue problems include symmetric non-negative and self-adjoint matrices, both of which have known constructions [20, 22, 33]. Deciding if a spectrum is realized by a real non-negative matrix is NP-hard [9].

Inverse eigenvalue *graph* problems ask whether a graph exists such that a given associated matrix (e.g., adjacency, random walk, Laplacian) has a given spectrum. Godsil and McKay present a method for generating non-isomorphic co-spectral graphs with respect to the adjacency matrix [28]. There has also been work on generating certain families of co-spectral graphs with respect to the Laplacian matrix [40]. We are interested in the (inexact) inverse eigenvalue graph problem with respect to the symmetric normalized Laplacian; with respect to the normalized Laplacian, Butler and Grout [10] have shown how to construct some families of (exactly) cospectral graphs with special structure.

Spectral graph generative approaches have been proposed in some prior work, e.g., [52, 53, 55]. The idea in these approaches is to consider spectral embeddings of the means and covariances of a *set* of graphs, then interpret these embeddings as samples from a distribution, and sample from this distribution, with the goal of interpolating between graphs. The sampling produces a candidate "Laplacian matrix." As discussed above, in general, it is not guaranteed that such a candidate matrix is the Laplacian of any matrix resembling a graph adjacency matrix, and indeed, a lot of our effort here is focused on actually producing a suitable adjacency matrix, and on choosing a "Laplacian matrix" that lends itself to this transformation.

As far as we can tell, [55] does not describe any such procedure. [52] suggests thresholding the values of the matrix, including as edges of the graph those pairs (i, j) for which the entries of the Laplacian are most negative. We compare two different thresholding approaches in this vein to our algorithms in Section 7.3, and find that they perform significantly inferior in matching the spectrum of the input graph.

In very recent work, [3] proposed an approach very similar to ours. Their goal is also to generate graphs whose spectrum approximately matches that of a given input graph; however, [3] use the Modularity matrix [42] instead of the normalized Laplacian. They use a low-rank approximation of the Modularity matrix, which is then transformed back to an adjacency matrix, noised, scaled, and truncated, to define edge probabilities $p_{i,j}$; subsequently, each edge (i, j) is included independently with probability $p_{i,j}$. The rank of the approximation is a user-specified feature, capturing how different the output graphs are likely to be from the input graph.

2 MODEL AND PRELIMINARIES

Throughout this paper, all graphs are undirected, and all matrices (except basis matrices) are real and symmetric. Vectors are denoted by boldface, including $\mathbf{0}$ and $\mathbf{1}$ for the all-0 and all-1 vectors. Graphs have n nodes, and matrices are of dimension $n \times n$.

For any matrix (adjacency or otherwise) $A = (a_{i,j})_{i,j} \in \mathbb{R}^{n \times n}$, we use $\lambda(A)$ to denote the set of A's eigenvalues $\lambda(A) = \{\lambda_1(A) \leq \lambda_2(A) \leq \cdots \leq \lambda_n(A)\}$, also called the *spectrum* of A. We use $\Lambda(A) = \mathrm{diag}(\lambda_1(A), \lambda_2(A), \ldots, \lambda_n(A))$ to denote the diagonal matrix whose diagonal entries are $\lambda_1(A), \lambda_2(A), \ldots, \lambda_n(A)$.

For any matrix A (in particular, an adjacency matrix), we let $d_i(A) = \sum_j a_{i,j}$, and call it the *degree* of node i in A. The degree matrix D for A is $D(A) = \mathrm{diag}(d_1(A), d_2(A), \ldots, d_n(A))$. The identity matrix is denoted by $I = \mathrm{diag}(1, 1, \ldots, 1)$.

Definition 2.1 (Symmetric Normalized Laplacian Matrix). The symmetric normalized Laplacian matrix $L(A)$ for the symmetric $n \times n$ matrix A is $L(A) = I - (D(A))^{-1/2} A (D(A))^{-1/2}$.

We call $\lambda(L(A))$ the *Laplacian spectrum* of A. It is well known that $\lambda_1(L(A)) = 0$, and $\lambda_n(L(A)) \leq 2$ [13]. Throughout this paper, all eigenvalues we consider will be of Laplacians of matrices; when there is no risk of confusion, we then simply refer to the Laplacian eigenvalues as $\lambda_1, \lambda_2, \ldots, \lambda_n$, omitting the dependence on A. In that case, the corresponding (normalized) eigenvectors are denoted by $\boldsymbol{x}_1, \boldsymbol{x}_2 \ldots, \boldsymbol{x}_n$, and the (orthonormal) matrix of eigenvectors is $X(A) = (\boldsymbol{x}_1, \boldsymbol{x}_2 \ldots, \boldsymbol{x}_n)$. The set of all such matrices, i.e., $n \times n$ matrices of orthonormal column vectors, is called the *Stiefel manifold* [18], and denoted by $\mathcal{S}_n$.

The difference $\lambda_2 - \lambda_1 = \lambda_2$ is called the *spectral gap of* $L(A)$. The corresponding eigenvector $\boldsymbol{x}_2$ is called the *Fiedler vector* [21, 23] of A, and plays an important role in identifying low-conductance cuts.

Given a diagonal matrix Λ of eigenvalues, we can obtain a matrix A with spectrum Λ from any basis matrix X simply as $A = W_\Lambda(X) := X \Lambda X^\intercal$; this is nothing but the standard *eigendecomposition* of A. Conversely, given a symmetric real matrix A, there is always a basis (of A's eigenvectors) such that $A = X \Lambda(A) X^\intercal$.

Definition 2.2 (Computational Goal). The given (undirected and connected) graph on n nodes is denoted by G^*, and has (symmetric) adjacency matrix $A^* \in \{0, 1\}^{n \times n}$. The eigenvalues of $L(A^*)$ are $0 = \lambda_1^* \leq \lambda_2^* \leq \cdots \leq \lambda_n^* \leq 2$.

The goal is to randomly sample a graph G with adjacency matrix $A \in \{0, 1\}^{n \times n}$ whose Laplacian spectrum $0 = \lambda_1(L(A)) \leq \lambda_2(L(A)) \leq \cdots \leq \lambda_n(L(A)) \leq 2$ is similar to that of A^*, in the sense that $\lambda_i(L(A)) \approx \lambda_i^*$ for all i.

Definition 2.2 is not formal, in that it does not quantify the meaning of "$\approx$", i.e., it does not specify how close is "close enough" in terms of the Laplacian spectrum of A. It would be desirable to do so, but we are not aware of ways to sample graphs whose spectrum is *provably* close to a given spectrum. Indeed, the following result of Borobia and Canogar [9] suggests that this may be difficult:

Theorem 2.3 ([9]). *The following problem is NP-hard: given a sequence of n real numbers $\lambda_1, \ldots, \lambda_n$, decide if there exists a real nonnegative matrix A of order n with eigenvalues $\lambda_1, \ldots, \lambda_n$.*

Theorem 2.3 does not apply directly to our problem, in that it (1) asks about the spectrum of A, rather than of $L(A)$, and (2) only requires that all entries of A be non-negative, rather than from $\{0, 1\}$. Typically, requiring integrality of a solution makes problems harder (e.g., linear programming), so we expect the hardness result to apply to our problem as well; however, at this time, we have no proof of NP-hardness with the integrality restriction.

3 ALGORITHM OUTLINE

Our graph generative algorithms can be divided into two stages:

(1) **Relaxed Spectrum Fitting:** Find a *template matrix* M whose Laplacian spectrum $\lambda(L(M))$ is very close to the target spectrum $\lambda(L(G^*))$; the entries $m_{i,j}$ may in principle be far from $\{0,1\}$ (including fractional, negative, or larger than 1), but the goal is for them to be close to $\{0,1\}$.
(2) **Template Matrix Perturbation:** Gradually "round" M, first to a matrix $P \in [0,1]^{n\times n}$, then to an actual adjacency matrix. In the process, aim to minimize perturbations to the Laplacian spectrum.

3.1 Relaxed Spectrum Fitting

The Relaxed Spectrum Fitting phase makes heavy use of the correspondence between matrices with a given spectrum and basis matrices. Its goal is to identify a good candidate orthonormal basis X so that the template matrix M defined by $L(M) = W_{\Lambda^*}(X) = X\Lambda^* X^\intercal$ has entries in (or close to) the interval $[0,1]$; or, even better, close to 0 or 1.

Our method starts by generating a random graph $\hat{G}$ from the configuration model [5] with the degree distribution of G^*. Let $\hat{A}$ be the adjacency matrix of $\hat{G}$, and $\hat{X}$ its eigenbasis. In the basic version of the Relaxed Spectrum Fitting phase, we use a linear programming (LP) based approach to find a template matrix M with $L(M) \approx W_\Lambda(\hat{X})$ (see Section 4).

The matrix M could have many entries far from $[0,1]$. Thus, rather than simply using $\hat{X}$, the extended version of our algorithm uses continuous optimization techniques to search the Stiefel manifold for a basis X such that the corresponding template matrix M has entries closer to 0 and 1. This optimization does not scale to large graphs, but more often than not makes small improvements.

3.2 Template Matrix Rounding

As a first step to rounding the template, M is converted to a matrix $P \in [0,1]^{n\times n}$, whose entries can be viewed as edge strengths or probabilities for randomized rounding. We use the following proxy problem: find an "error" matrix E such that $p_{i,j} = m_{i,j} + e_{i,j} \in [0,1]$ for all i,j and the spectrum of M is close to that of P. We show (in Section 6) how to cast the problem of finding E as a linear program, and prove that the objective function gives an upper bound on the sum of eigenvalue perturbations.

Finally, P needs to be converted to an adjacency matrix A. Small eigenvalues λ_k are particularly sensitive to changes going from M to P or from P to A. We therefore treat them separately. Our goal is to make initial edge modifications judiciously to counteract such effects, and to prevent large changes in λ_k for small k. To do so, we focus on the first approximately balanced[4] *eigencut* $(S_k, \overline{S_k})$, defined by $S_k = \{i \mid x_{k,i} < 0\}$ and $\overline{S}_k = \{i \mid x_{k,i} \geq 0\}$, where $\boldsymbol{x}_k$ are eigenvectors of $L(P)$. For each edge crossing $(S_k, \overline{S_k})$, we estimate how much its addition or removal would affect λ_k. We then select edges (i,j) to round $p_{i,j}$ up or down to correct λ_k as well as possible. We observe experimentally that when $k \neq 2$, if rounding edges across $(S_k, \overline{S}_k)$ shrinks λ_k, the spectral gap is typically lowered as well. There could be other candidate cuts that provide valuable connectivity information, but we restrict our search to eigencuts to keep the computational cost manageable.

In a final step, we round all remaining edges, independently including edge (i,j) with probability $p_{i,j}$.

3.3 Randomness

There are two sources of randomness in our generative process: the random initial generation of an orthonormal matrix X via a configuration model graph $\hat{G}$, and the independent random edge inclusions in the final part of the Template Matrix Rounding phase.

Naturally, the process does not guarantee sampling a graph *uniformly* from an ensemble of graphs with approximately correct spectra; however, experimentally, it leads to significant variation in the graphs that are generated, obviously an important criterion for graph generative processes [48].

4 RELAXED SPECTRUM FITTING

We are given a basis X and associated matrix $A = W_{\Lambda^*}(X)$, and would like to identify a matrix M whose normalized Laplacian is $L(M) = A$; ideally M, should be the adjacency matrix of a graph or close to one. It is obvious how to compute $L(M)$ from M. However, we need to compute M given a candidate $L(M)$. The transformation requires knowing the degrees d_i of nodes in M. The following lemma motivates our LP-based approach for finding d_i:

Lemma 4.1. *The matrix A is the Laplacian of some symmetric matrix M with positive row sums if and only if there exists a vector $\boldsymbol{y} > \mathbf{0}$ (all entries are strictly positive) such that $A\boldsymbol{y} = \mathbf{0}$.*

Proof. For the first direction, assume that $A = L(M) = I - D^{-1/2}MD^{-1/2}$ for some symmetric matrix M with positive row sums. Let $d_i = \sum_j m_{i,j} > 0$ be the degree of i in M, and let $\boldsymbol{y}$ be the vector with $y_i = d_i^{1/2}$. Clearly, $\boldsymbol{y} > \mathbf{0}$, and

$$A\boldsymbol{y} = \boldsymbol{y} - D^{-1/2}MD^{-1/2}\boldsymbol{y} = \boldsymbol{y} - D^{-1/2}M\cdot\mathbf{1} = \boldsymbol{y} - D^{-1/2}\boldsymbol{d} = \mathbf{0}.$$

For the converse direction, let $\boldsymbol{y} > \mathbf{0}$ be a solution to $A\boldsymbol{y} = \mathbf{0}$. Let $Y = \mathrm{diag}(y_1, y_2, \ldots, y_n)$, and define $M = Y^2 - YAY$. The vector of row sums of M is

$$M\cdot\mathbf{1} = (y_i^2)_i - YAY\cdot\mathbf{1} = (y_i^2)_i - YA\boldsymbol{y} = (y_i^2)_i > \mathbf{0}.$$

And because

$$L(M) = I - Y^{-1}MY^{-1} = I - Y^{-1}(Y^2 - YAY)Y^{-1} = I - I + A = A,$$

A is indeed the symmetric normalized Laplacian of M. □

The vector entries y_i in Lemma 4.1 are the square roots of the degrees of node i. During the computation, the basis X may be such that the associated matrix $W_{\Lambda^*}(X)$ is not the Laplacian of any matrix M with positive row sums.[5] In order to compute an approximate and usable matrix M, we relax the constraint that $M\boldsymbol{y} = \mathbf{0}$, and instead aim to minimize $||M\boldsymbol{y}||_\infty$. We approximate the positivity constraint by requiring that $y_i \geq \epsilon$ for all i, for a very small positive constant $\epsilon > 0$; we then obtain the following linear program:

[4] For most inputs, $k = 2$ and we round edges across the Fiedler cut. Occasionally, the Fiedler cut is very unbalanced. In those cases, rounding edges across the Fiedler cut very judiciously is largely misspent effort, since later steps are likely to disconnect the smaller side (plus some nodes) from the rest of the graph.

[5] Because $\lambda_1^* = 0$, the matrix $W_{\Lambda^*}(X)$ is always singular. However, the corresponding eigenvector $\boldsymbol{x_1}$ will typically have negative entries, so we are not guaranteed a vector $\boldsymbol{y} > \mathbf{0}$.

$$\begin{array}{lll} \text{Minimize} & b & \\ \text{subject to} & |\sum_{j=1}^{n} a_{i,j} y_j| \le b & i = 1, \ldots, n \\ & y_j \ge \epsilon & j = 1, \ldots, n. \end{array} \quad (1)$$

The procedure for fitting a graph M is now as follows:

(1) Given an orthonormal basis X, compute $A = W_{\Lambda^*}(X)$.
(2) Solve the LP (1), obtaining a solution $(b, \boldsymbol{y})$.
(3) Output the *template matrix* $M = Y^2 - YW_{\Lambda^*}(X)Y$, where $Y = \text{diag}(\boldsymbol{y})$.
(4) To match the edge density of M to that of the target, set $M := \frac{\sum_i d_i^*}{\sum_i d_i} \cdot M$, where d_i^* are the node degrees in the target graph, and $d_i = \sum_j m_{i,j}$.

If $b = 0$, then $L(M) = W_{\Lambda^*}(X)$, and we have found a template matrix M such that $L(M)$ has exactly the desired spectrum Λ^*. We note that the spectrum is invariant to the scaling in Step (4). We perform the scaling to match the target edge density before perturbing the edges, in order to avoid having to perturb the matrix twice to yield entries in the interval $[0, 1]$. Even in this case, individual entries $m_{i,j}$ of the template matrix may be negative or larger than 1 (and of course fractional): the solution $\boldsymbol{y}$ only guarantees that the *total* degree of each node is positive.

When $b > 0$, we have that $L(M) \neq W_{\Lambda^*}(X)$, meaning that typically $\Lambda(M) \neq \Lambda^*$ as well, i.e., the spectrum can be perturbed. We show that the deviation in spectrum can be bounded using the LP objective b; the lemma also motivates our choice of LP objective.

Lemma 4.2. *Let δ be such that $b \le \delta y_i$ for all i. Then, the perturbed eigenvalues satisfy $|\lambda_i(L(M)) - \lambda_i(A)| \le \frac{\delta}{1-\delta} \cdot |\lambda_i(A)|$.*

Proof. The degrees under M are $d_i = \sum_k m_{i,k} = y_i^2 - y_i \sum_k a_{i,k} y_k$, so the perturbed Laplacian matrix $L = L(M)$ has entries

$$\ell_{i,j} = \frac{a_{i,j} y_i y_j}{(y_i^2 - y_i \sum_k a_{i,k} y_k)^{1/2} (y_j^2 - y_j \sum_k a_{j,k} y_k)^{1/2}}.$$

Writing $z_i = \frac{y_i}{(y_i^2 - y_i \sum_k a_{i,k} y_k)^{1/2}}$ and $Z = \text{diag}(z_1, z_2, \ldots, z_n)$, the perturbed Laplacian therefore satisfies $L = ZAZ^\intercal$.

Applying Theorem 4.3 (below) with $D = D^\intercal = Z$, we get that the relative perturbation in eigenvalues is at most

$$|\lambda_i(L(M)) - \lambda_i(A)| \le |\lambda_i(A)| \cdot ||Z^\intercal Z - I||_2.$$

The norm of a diagonal matrix is its maximum entry, so

$$||Z^\intercal Z - I||_2 = \max_i \frac{\sum_k a_{i,k} y_k}{y_i - \sum_k a_{i,k} y_k} \le \max_i \frac{b}{y_i - b},$$

by the LP's first constraint. Because $b \le \delta y_i$ for all i by assumption, we obtain the claimed bound. □

Theorem 4.3 (Theorem 2.1 of [19]). *Let $\tilde{A} = D^\intercal A D$, where D is a nonsingular matrix. Let λ_i and $\tilde{\lambda}_i$ be the eigenvalues of A and $\tilde{A}$, respectively. Then, $|\tilde{\lambda}_i - \lambda_i| \le |\lambda_i| \cdot ||D^\intercal D - I||_2$, for all i.*

5 STIEFEL MANIFOLD OPTIMIZATION

While the techniques in Section 4 find a good M so $L(M) \approx X\Lambda X^\intercal$, if X was a bad basis, no M will be satisfactory, and the entries will lie far outside $[0, 1]$, or the Laplacian eigenvalues will be perturbed. To improve the basis before committing to it, we can perform a walk on the Stiefel manifold, using known techniques to locally optimize an objective function that rewards template matrices M with entries close to 0 or 1. Specifically, we define the objective function $F(M) = \sum_{i<j} (1 - m_{i,j})^2 m_{i,j}^2$, which has minima when all $m_{i,j}$ are in $\{0, 1\}$ and steeply penalizes entries far from 0 and 1.

For any basis $X \in \mathcal{S}_n$, let $\boldsymbol{y}_X$ be the solution to the linear program (1) (applied to $A = X\Lambda^* X^\intercal$), and $Y_X = \text{diag}(\boldsymbol{y})$. Then, $M(X) = Y_X^2 - Y_X X \Lambda^* X^\intercal Y_X$ is the candidate template matrix for X. The objective is then to find a basis $X \in \mathcal{S}_n$ (approximately) minimizing $F(M(X))$.

While the objective function F itself is differentiable (which would allow for a straightforward application of existing manifold optimization techniques), the transformation $X \mapsto M(X)$ is computed via the solution to a linear program. It is not clear how optimal solutions to the linear program (1) change with X, and in particular whether $X \mapsto F(M(X))$ is continuous or differentiable. Therefore, few guarantees for known optimization techniques apply in our setting. To compensate, we use a small maximum step size of .0001 in the optimization.

To perform optimization over the Stiefel manifold, we use known iterative optimization techniques. Specifically, we use the *Polar Decomposition retraction scheme* [2][6]. Retraction schemes in each step t identify a search direction $\boldsymbol{\eta}_t$ in the tangent bundle of the current basis X_t, such that moving in the direction $\boldsymbol{\eta}_t$ minimizes F. (See [1] for definitions of the notions of *tangent bundle* and *retraction*.) The scheme moves X_t by a step size τ_t in the direction $\boldsymbol{\eta}_t$, then projects it back onto the manifold using a retraction $R_X(\tau)$. A retraction scheme specifies both the retraction $R_X(\tau)$ and search direction $\boldsymbol{\eta}$ such that $R_X(\tau)$ is a descent direction: the derivative $R'_X(\tau)$ must be equal to the projection of $-\text{grad}(F(M(X_t)))$ onto the tangent bundle at X. The step size τ_t is typically chosen according to the *Armajiro-Wolfe Conditions*, which ensure that at the point $X_t + \tau_t$, the decrease $F(X_t) - F(X_t + \tau_t)$ is proportional to τ (*sufficient-decrease*), but also that τ_t is sufficiently large such that one cannot decrease $F(X_t + \tau_t)$ by taking a larger step (*curvature condition*) [45]. One iteration of the Stiefel manifold optimization can be summarized as follows:

(1) Using X_t, solve the template fit LP (1) to compute $M(X_t)$.
(2) Treat the y_{X_t} computed from LP (1) as constant, and using $\text{grad}(F(M(X_t)))$, compute a search direction $\boldsymbol{\eta}_t$ using a retraction method.
(3) Perform a line search technique to find a step size τ_t that obeys the Armajiro-Wolfe conditions [45].
(4) Set $X_{t+1} = X_t + \tau_t \boldsymbol{\eta}_t$.
(5) Repeat until a local optimum is reached.

We implemented and experimented with two different retraction methods: the Cayley Transform Retraction [51] and the Polar-Decomposition Retraction [2]. Our experiments showed that the Polar-Decomposition Retraction performed better most of the time.

6 ROUNDING THE TEMPLATE MATRIX

6.1 Perturbing the Template Matrix

The next step is to compute an additive perturbation matrix E that leaves the row sums (i.e., degrees) of M intact (first constraint of

[6] Manifold optimization techniques can generally be divided into two categories: retraction schemes and geodesic schemes. We use retractions as geodesics are often difficult to compute [1].

LP (2)), and ensures that all entries of $P = M + E$ are in $[0, 1]$ (third constraint). Subject to this, we aim to minimize the weighted sum of absolute perturbations. The variables are $e_{i,j}$ and $q_{i,j} = |e_{i,j}|$ (second constraint) for $i \neq j$ (with $e_{i,i} = 0$ implicitly). $d_i = \sum_j m_{i,j}$ denotes the (fractional) degree of i.

$$\begin{array}{lll} \text{Minimize} & \sum_{i \neq j} \frac{q_{ij}}{d_i^{1/2} d_j^{1/2}} & \\ \text{subject to} & \sum_j e_{i,j} = 0, & i = 1, \ldots, n \\ & q_{i,j} \geq |e_{i,j}| & \text{for all } i \neq j \\ & 0 \leq m_{i,j} + e_{i,j} \leq 1 & \text{for all } i \neq j \\ & e_{i,j} = e_{j,i} & \text{for all } i \neq j \end{array} \tag{2}$$

Keeping the degrees constant ensures that the normalization factors for the Laplacian are the same for P and M. The choice of objective function is justified by the following lemma.

Lemma 6.1. *Let $(e_{i,j})$ be an optimal solution of the LP (2), with objective value b. Then, the eigenvalues of the Laplacians of M and $P = M + E$ are close: $\sum_i |\lambda_i(L(M)) - \lambda_i(L(P))| \leq b$.*

Proof. Consider the perturbation $\Delta = L(M) - L(P)$ of the Laplacian matrix. Applying Theorem 6.2 (below) with $\Phi(x) = |x|$, we obtain that $\sum_i |\lambda_i(L(M)) - \lambda_i(L(P))| \leq \sum_i |\lambda_i(\Delta)|$. In the remainder of the proof, we will show that $\sum_i |\lambda_i(\Delta)| \leq b$.

Because Δ is symmetric and real-valued, $\Delta = U^\intercal \Lambda U$ for a diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ of eigenvalues λ_i of Δ, and orthonormal U. Let $\hat{\Lambda} = \mathrm{diag}(|\lambda_1|, \ldots, |\lambda_n|)$ be the diagonal matrix of absolute values of Δ's eigenvalues, and $\hat{\Delta} = U^\intercal \hat{\Lambda} U$. Define $\sigma_i = \mathrm{sgn}(\lambda_i)$ (with $\mathrm{sgn}(0) := 1$), and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$ to be the diagonal matrix of signs of λ_i, so that $\hat{\Lambda} = \Sigma\Lambda$. Then, $\hat{\Delta} = U^\intercal \Sigma U \Delta = U' \Delta$, where $U' = U^\intercal \Sigma U$ is a unitary matrix. Because U' is unitary, its entries are bounded by 1 in absolute value. We can therefore bound

$$\sum_i |\lambda_i(\Delta)| = \mathrm{tr}(\hat{\Delta}) = \mathrm{tr}(U'\Delta) = \sum_i \sum_j u'_{i,j} \delta_{j,i} \leq \sum_{i,j} |\delta_{j,i}|.$$

We have shown that the sum of absolute eigenvalues of a (real symmetric) matrix is bounded by the sum of absolute values of its entries[7]. It remains to bound the sum of absolute entries of Δ. Because the degrees (row sums) are the same, i.e., d_i, for M as for P, we obtain that $\delta_{i,j} = \frac{m_{i,j} - p_{i,j}}{d_i^{1/2} d_j^{1/2}}$. Summing their absolute values over all i, j, this is exactly the objective function of the LP (2), which we assumed to be bounded by b. □

Theorem 6.2 (Theorem 1 of [31]). *If A, B are $n \times n$ Hermitian matrices, their eigenvalues can be enumerated in such a way that for every real-valued convex function Φ on $\mathbb{R}$, we have $\sum_i \Phi(\lambda_i(A) - \lambda_i(B)) \leq \sum_i \Phi(\lambda_i(B - A))$.*

6.2 Cut Rounding

As outlined in Section 3, the cuts corresponding to small λ_k are particularly important for global connectivity properties. Our rounding procedure therefore first focuses on matching those eigenvalues. The main point of comparison is the Fiedler cut $(S^*, \bar{S^*})$ of the input graph G^*; without loss of generality, we assume that $|S^*| \leq n/2$. Ideally, we would like to focus on rounding edges across the Fiedler cut of the fractional graph P, so as to match $\lambda_2(G^*)$. However, it is possible[8] that the Fiedler cut of P is extremely unbalanced, with one side only having a handful of nodes. Efforts to round such unbalanced cuts are largely misplaced.

Instead, we focus on the first approximately balanced cut defined by an eigenvector of P. Let $k \geq 2$ be smallest such that the smaller side of the cut defined by $S_k = \{i \mid x_{k,i} < 0\}$ (again, w.l.o.g. S_k rather than its complement) satisfies $|S_k| \geq \frac{1}{2}|S^*|$. We let $\boldsymbol{x} = \boldsymbol{x}_2, \boldsymbol{x'} = \boldsymbol{x}_k$ and refer to $(S_k, \bar{S}_k)$ as the *critical cut*. We write $d_i = \sum_j p_{i,j}$. Our heuristic is guided by the standard characterization $\lambda_2 = \boldsymbol{x} L(P) \boldsymbol{x}^\intercal = \sum x_i^2 - \sum_{i \neq j} x_i x_j \frac{p_{i,j}}{\sqrt{d_i d_j}}$.

When i and j are on opposite sides of the Fiedler cut,[9] their signs in $\boldsymbol{x}$ are opposite, making the term $-x_i x_j \frac{p_{i,j}}{\sqrt{d_i d_j}}$ positive in $\boldsymbol{x} L(P) \boldsymbol{x}^\intercal$. This motivates scoring each edge (i, j) based on $x_i x_j$ as candidates for removal (rounding $p_{i,j}$ down to 0) or addition (rounding $p_{i,j}$ up to 1). However, the heuristic of rounding edges in such a way suffers from two drawbacks:

(1) The degrees d_i and d_j are affected when the edge (i, j) is removed or added; the changes in the terms for edges (i', j) or (i, j') could offset the rounding progress.
(2) The Fiedler vector (and other eigenvectors) may change after removing or adding an edge (i, j), making it difficult to compute a set of edges to remove or add one by one.

Ideally, after each small change to some $p_{i,j}$, one should recompute the eigenbasis before continuing. This is computationally expensive, so instead we adapt an idea of the *NetMelt* algorithm, which was designed to identify good edge removals/additions to decrease/increase the spectral gap of an *adjacency* matrix [11]. We assign scores for each edge removal and edge addition. When working with the adjacency matrix, each edge removal/addition affects only one entry of the perturbation matrix E. For the Laplacian matrix, an edge removal or addition can change the spectrum also through the changes in the nodes' degrees (and thus the normalization).

To help safeguard against the possibility of perturbing many edges adjacent to the same node, thereby changing the degrees drastically and misestimating additional rounding effects, we designate a *budget* b for how many edges can be removed; once the budget has been exceeded, the Fiedler vector $\boldsymbol{x}$, eigenvector $\boldsymbol{x'}$, and scores are recomputed. In addition, if the (fractional) number of edges crossing the Fiedler cut is below 1, we stop the procedure. To keep the overall edge density reasonably constant, we alternate between edge removals (across the cut) and additions (on some side of the cut).

The central part of the rounding algorithm is the following *Critical Cut Rounding* procedure. It is called repeatedly from the overall rounding procedure (described further below), which handles special cases such as "disconnected" probability matrices P and very unbalanced Fiedler Cuts.

[7] We suspect that this must be a well-known fact, but could not find a reference.

[8] We found that this only happened on input graphs with spectral gap less than .01 and with sparse edge density (for example, the Euro Road graph).

[9] The intuition behind the reasoning applies to $\boldsymbol{x'}$ as well, although the variational characterization of λ_k is more complex.

(1) Compute $L(P)$, the Fiedler vector $\boldsymbol{x}_2$, the spectral gap $\lambda_2(L(P))$, $\boldsymbol{x'}$ and $\lambda_k(L(P))$, and let $\delta = \lambda_k(L(P)) - \lambda_k^*(L(P))$ be the amount by which we would like to decrease λ_k. If $\delta < c$ (where c is a small constant) or the number of (fractional) edges crossing the critical cut or the Fiedler cut is less than 1, exit the Critical Cut Rounding procedure.
(2) For all (i,j) with $x'_i x'_j < 0$ (on opposite sides of the critical cut), let the score be $s_{i,j} = -\frac{x'_i x'_j p_{i,j}}{\sqrt{d_i d_j}} \geq 0$. All such edges (i,j) are removal candidates; we consider them in order from highest to lowest score.
(3) For all (i,j) with $x'_i x'_j > 0$ (on the same side of the critical cut), let the score be $s_{i,j} = \frac{x'_i x'_j (1-p_{i,j})}{\sqrt{d_i d_j}} \geq 0$. All such edges (i,j) are addition candidates; we consider them in order from highest to lowest score.
(4) Keep track of the following quantities: q is the sum of $p_{i,j}$ for all edges (i,j) that have been rounded, s the sum of all scores of alterations, and y the total change in entries in P (where positive and negative changes cancel out). Initially, all of q, s, y are 0.
(5) Until $q \geq b$ or $s \geq \delta/2$, or all addition and removal candidates have been processed:
   - If $y \geq 0$, consider the removal candidate (i,j) with largest score $s_{i,j}$; otherwise, consider the addition candidate (i,j) with largest score $s_{i,j}$. (In either case, remove (i,j) from the candidates for the future.)
   - If $s + s_{i,j} \leq \delta$ and (i,j) is not the last candidate edge keeping the critical cut connected, update the values as follows (otherwise, skip (i,j)):
     - If (i,j) was a candidate for removal, then set $p_{i,j} = 0$ and $y = y - p_{i,j}$; otherwise, set $p_{i,j} = 1$ and $y = y + (1 - p_{i,j})$.
     - In both cases, update $s = s + s_{i,j}$ and $q = q + p_{i,j}$.
(6) If any changes were made to P, repeat from Step (1).

$L(P)$ and the eigenvectors are computed only once for each overall iteration of the critical cut rounding procedure. An iteration terminates once the approximate score s reaches $\delta/2$, or the algorithm has reached its budget b. At this point, the critical cut and Fiedler cut are recomputed to check whether the algorithm has indeed reached its goal of ensuring $\lambda_k^* \leq \lambda_k \leq \lambda_k^* + c$, and to adapt should the cuts have changed. In our implementation, we use a constant of $c = .0001$ and a budget b equal to $\frac{1}{4}$ of the number of edges crossing the critical cut. When the Critical Cut Rounding procedure terminates, either all edges across the critical cut have been considered, or λ_k is a good approximation to the target: $\lambda_k^* \leq \lambda_k \leq \lambda_k^* + c$.

While Critical Cut Rounding is the key component to our rounding approach, we need to take care of special cases: disconnected template matrices P and extremely unbalanced Fiedler cuts. This is accomplished by the following *Cut Rounding* procedure.

(1) Consider the graph with edges (i,j) iff $p_{i,j} > 0$. If this graph is disconnected, then permanently discard everything except its largest connected component. For the remainder, consider only the remaining nodes.
(2) Compute $L(P)$, the Fiedler vector $\boldsymbol{x}_2$, and the spectral gap $\lambda_2(L(P))$. While the Fiedler cut is severely unbalanced (having fewer than five vertices on the smaller side of the cut), permanently discard the vertices on the smaller side, and recompute $L(P)$, $\boldsymbol{x}_2$, and $\lambda_2(L(P))$. [•]
(3) If $\lambda_2(L(P)) - \lambda_2^*(L(P)) \geq c$ and the number of (fractional) edges crossing the Fiedler cut is at least 1, then perform Critical Cut Rounding, as described above. If Critical Cut Rounding made any changes to P, then repeat from Step (1). [•]
(4) While the (fractional) number of edges crossing the Fiedler Cut is less than 1, round the largest fractional edge $p_{i,j}$ crossing the Fiedler cut up to 1.

David deleted here

David deleted here

In our experiments, the largest fraction of nodes ever disconnected was .5% (including nodes removed due to unbalanced Fiedler cuts).

When the Critical Cut Rounding procedure made the sum of fractional edges crossing the Fiedler cut less than one, the graph would be likely to become disconnected in the later independent rounding step. To prevent this, the final step rounds the largest fractional entry up to 1.

When there are remaining edges to round, we include them in the independent rounding stage described next; this does not change the expected density of edges across the critical cut.

We also experimented with using a dependent rounding scheme [25] that includes each edge with probability $p_{i,j}$, but correlates the random choices so that the the number of edges crossing the Fiedler cut exactly matches the expectation (up to fractional parts). Experimentally, we found the performance of both approaches comparable, so we did not include dependent rounding in our final algorithm.

6.3 Independent Rounding

The final step is to include each remaining edge (i,j) independently with probability $p_{i,j}$, akin to the generation of $G(n,p)$, SBM, or Kronecker graphs. The following theorem by Chung and Radcliffe bounds the resulting spectral perturbations.

Theorem 6.3 (Theorem 2 of [16]). *Let G be a random graph, generated by including each edge (i,j) independently with probability $p_{i,j}$. Let A be the adjacency matrix of the random graph, and $\delta = \min_i \sum_j p_{i,j}$ the minimum expected degree of any node. For every $\epsilon > 0$, there exists a constant $k = k(\epsilon)$ such that if $\delta > k \ln n$, then with probability at least $1 - \epsilon$, all eigenvalues of $L(A)$ and $L(P)$ satisfy $|\lambda_i(L(A)) - \lambda_i(L(P))| \leq 3\sqrt{\frac{3\ln(4n/\epsilon)}{\delta}}$.*

7 EXPERIMENTAL EVALUATION

We evaluated our generative model (*SpectralGen*) against the Configuration Model (*Config.*), Stochastic Block Model (*SBM*), and Kronecker graph model (*Kronecker*), based on many real-world data sets, and using a number of metrics.

7.1 Data, Models, and Metrics

Data Sets. We report on results[10] from running on several standard network data sets, available for download from https://icon.

[10] We also ran experiments on several other (smaller) network data sets from the Colorado repository: Political books, Football, High School dynamic contacts, Computer Science Faculty Hiring, Physicians social network, Irvine student forum, Jazz. The results were similar to those reported here.

colorado.edu/#!/networks, and https://tuvalu.santafe.edu/~simon/styled-9/styled-10/. They are a network of flights between 500 commercial airports in the US, weights representing the number of seats (*Airport*), an email exchange network between members of a university (*Email*), self-reported friendships between high school students (*Health*), a road network made up of European cities and roads (*Euro Road*), and links between Wikipedia pages on editorial norms (*Wiki*). Even when the input network was directed (e.g., Wikipedia), we treated the edges as undirected. In addition to these real-world networks, we include a grid graph in our evaluation to test if our method can prevent generating an expander when the target is far from one.

For all networks (input graphs and generated networks), we focused on the the largest connected component rather than the entire graph. This avoids difficulties in comparing spectra or assigning arbitrary distances to disconnected pairs of nodes. We also removed all self loops. The input graphs used and their basic parameters (of the largest connected component) are summarized in Table 1.

	Num Vertices	Num Edges	Spectral Gap	Fiedler Cut Conductance
Airport	500	2980	0.0274	0.0588
Email	1133	5452	0.1211	0.1617
Health	2535	10455	0.0379	0.0654
Euro Road	1039	1305	0.0005	0.0093
Grid 900	900	1740	0.0029	0.0299
Wiki	1872	15367	0.1295	0.5135

Table 1: Basic properties of the input graphs.

Generative Models. We generated networks using the following four models. For the first three, we implemented the models ourselves in MATLAB.

SpectralGen. Our own model, as described in Sections 3–6.[11]

Config. The configuration model [5, 41], matching the input degree sequence.

SBM. For the SBM [29], we used the Fiedler cut to find a high-level bipartition, then fit the edge densities across this partition and within each set to the input graph. Without knowledge of the number of clusters that best fit the SBM to the ground truth graphs, we use two blocks, and choose the edge densities so as to match the conductance across the Fiedler cut in expectation.

Kronecker. The Kronecker model [36], fit and generated using the Stanford Network Analysis Platform (SNAP) [38], to generate a network of size 2^k which is closest to n (the input graph's size). We used the default initial gradient descent matrix and initiator matrix size and 100 gradient descent iterations.

Evaluation Metrics. We compared the four models along multiple network properties.

Spectrum. The difference between the spectra of the generated graphs and the spectrum of the input graph. Doing well for this metric was the main goal of our work.

Path Length Distribution. Distribution of lengths of shortest paths from s to t for all vertex pairs (s, t).

Clustering Coefficient Distribution. Writing $T(v)$ for the number of triangles that v participates in, the clustering coefficient of v is $\frac{T(v)}{d_v(d_v-1)}$, the fraction of pairs of neighbors of v that are neighbors of each other.

Betweenness Centrality Distribution. Letting $\sigma_{s,t}$ denote the number of shortest paths between s and t, and $\sigma_{s,t}(v)$ the number of shortest paths from s to t that pass through v, the betweenness centrality of v is $\sum_{s,t\neq v} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$.

Degree Distribution. Frequency for each degree.

All graph properties were computed using the Python Networkx library. For betweenness centrality, clustering coefficients, degree distributions, and path length distributions, we evaluated how close the distribution of the generated graph was to the distribution of the input graph using the (one-dimensional) Earth Mover's Distance (EMD). EMD is a well-suited distance measure between distributions for our purpose because it increases when more probability mass needs to be shifted, or it needs to be shifted larger distances. For example, we want to consider graphs more similar if path distances are mostly off by 1 than when they are mostly off by 5.

To compare EMD results across different graph properties, we normalized each distribution to the unit interval: more specifically, when computing the EMD between two empirical distributions, we normalized the elements in the supports of both distributions by dividing them by the largest element in the union of their supports. We compute the loss reductions (or increases) of our method as

$$\Delta = (\text{mean EMD BB} - \text{mean EMD SpecGen})/(\text{mean EMD BB}),$$

where "BB" is the *best* benchmark (having the smallest average EMD).

7.2 Results

Figure 1 shows plots of the first 50 eigenvalues of the graphs generated according to the different models; we plot the average and standard deviation for each eigenvalue across 20 runs. The eigenvalues tend to be tracked closer by our method than by the other generative models.

Next, we report the mean and standard deviation EMD[12] of each graph property discussed above for each generative model. These data are reported in Table 2. The spectral generation approach generally outperforms the benchmarks for shortest path lengths and clustering coefficients; however, there are two exceptions.

Our method does not match shortest path lengths on the Wikipedia graph well. This is likely due to the large discrepancy (.384) between the conductance across the Fiedler cut and the spectral gap of the Wikipedia graph, since we observed similar behavior on other graphs with this discrepancy. For graphs with such a large discrepancy, using the density of the Fiedler cut as a stand-in for the graph conductance is a poor approximation.

We included the grid graph in our evaluation, because it is explicitly *not an expander*, while most graph models tend to generate expander graphs. Indeed, our method matches path lengths and

[11] The code is online at https://github.com/alanadakotashine/spectral_generation.

[12] Statistics are taken over $k = 20$ runs for all graphs.

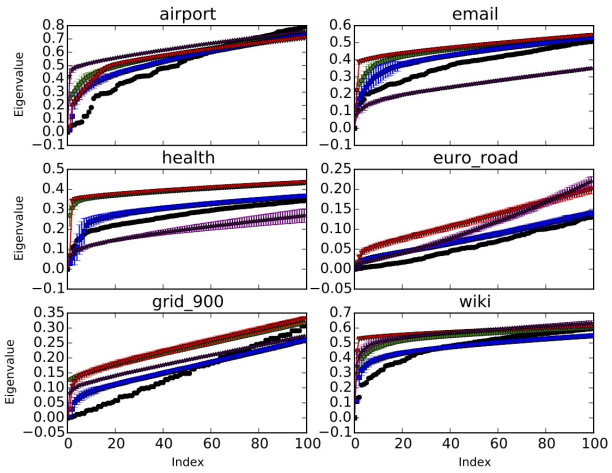

Figure 1: Comparison of spectra. Input graph (black circles), SpectralGen (blue squares), Config (green hexagons), SBM (red triangles), Kronecker (purple stars).

betweenness centrality significantly better. It does not do well on clustering coefficients, because it generates too many triangles.

The results are mixed on betweenness centrality. The EMDs for betweenness centrality are smaller than the EMD for other graph properties, which indicates that all methods, including ours, track betweenness centrality reasonably well relative to other graph properties.

The spectral generation approach does not match the degree distribution as well as the configuration model. This is unsurprising considering that the configuration model is designed explicitly to match the degree distribution, making mistakes only when it generates self-loops or multi-edges.

7.3 Evaluating Algorithm Components

Our approach utilizes multiple heuristics. To study the contribution of each heuristic to the final result, in Figure 2, we plot (for two graphs) the spectra obtained by leaving out various steps. We also compare the spectra against those of simple thresholding approaches as they seem to be utilized by [52]. In addition to the spectra of the input graphs, we show the spectra of six matrices.

(1) The result of using the initial template matrix C from the configuration model (with entries $c_{i,j} = d_i d_j / m$), and applying our rounding procedure from Section 6, without first applying LP (1). C is a reasonable candidate for rounding because it matches the degrees of G^*, and all its entries are already in $[0, 1]$.
(2) The template matrix output by the LP (1). This matrix may have entries outside $[0, 1]$.
(3) The matrix output by the LP (2), which forced entries from the template inside $[0, 1]$.
(4) The final rounded output using the Stiefel manifold optimization along with the LP (1) for relaxed spectrum fitting.
(5) The final rounded output without using the Stiefel manifold optimization, but only using the LP (1) for relaxed spectrum fitting.

Shortest Paths

	SpecGen	Config	Kronecker	SBM	Δ
Air.	.02 ± .008	.03 ± .001	.04 ± .003	.02 ± .003	.12
Eml.	.02 ± 4e-3	.03 ± 2e-3	.08 ± .01	.03 ± 2e-3	.4
Hlth.	.03 ± 8e-3	.07 ± 3e-4	.04± 6e-3	.05 ± 5e-3	.29
Euro	.14± .02	.15 ± 2e-3	.15 ± 7e-3	.17± .01	.07
Grid	.2 ± 9e-3	.25± 2e-4	.25 ± 8e-4	.25 ± 2e-3	.17
Wiki	.04 ± 3e-3	.01 ± 5e-4	.01 ± 3e-3	.03 ± 5e-4	-2.4

Clustering Coefficients

	SpecGen	Config	Kronecker	SBM	Δ
Air.	.31 ± .02	.39 ± .02	.48 ± .02	.59 ± .003	.25
Eml.	.13 ± .03	.2 ± .002	.21 ± .002	.21 ± 7e-4	.53
Hlth.	.09 ± .006	.14 ± 6e-4	.14 ± .002	.14± 5e-4	.55
Euro	.01± 6e-3	.02± 2e-3	.02 ± .003	.02± 2e-3	.02
Grid	.04 ± .01	.002± .001	.0001 ± 1e-4	.01 ± .001	-39
Wiki	.25 ± .01	.27 ± .003	.3 ± .01	.37 ± 3e-4	.07

Betweenness Centralities

	SpecGen	Config	Kronecker	SBM	Δ
Air.	.01 ± 1e-3	8e-3 ± 4e-4	.01 ± 4e-4	.02 ± 5e-4	-.4
Eml.	.01 ± 2e-3	7e-3 ± 3e-4	.02 ± 5e-3	.04 ± 6e-4	-.6
Hlth.	5e-3 ± 3e-3	.02 ± 9e-5	.04 ± .02	.03 ± 5e-4	.76
Euro	.03± 4e-3	.04 ± 3e-4	.02 ± 2e-3	.03± 2e-3	-.6
Grid	.09 ± .04	.34± 4e-4	.34 ± .02	.32 ± .04	.71
Wiki	2e-3 ± 7e-4	5e-3 ± 9e-4	4e-3 ± 2e-4	6e-3 ± 3e-3	.5

Degree Distribution

	SpecGen	Config	Kronecker	SBM	Δ
Air.	.02 ± 2e-3	.01 ± .08	.03 ± 1e-3	.07 ± 1e-3	-.2
Eml.	8e-3 ± 3e-3	2e-3± 2e-4	.07 ± 5e-4	.06 ± 8e-4	-2.8
Hlth.	.01 ± 7e-3	8e-3 ± 2e-4	.14 ± 7e-3	.04 ± 9e-4	-.25
Euro	.01± 3e-3	.002± 7e-4	.03 ± 3e-3	.05± 5e-3	-5.9
Grid	.06 ± 8e-3	2e-3± 9e-4	.12 ± .01	.12 ± .01	-30.7
Wiki	4e-3 ± 5e-4	3e-3 ± 6e-5	7e-3 ± 2e-4	.03 ± 2e-4	-.4

Table 2: Models' Performance for matching graph property distributions and loss reduction.

(6) We derive the following random graph using methods similar to [52]. We combine the spectrum of the target graph and a uniformly random orthonormal matrix X to define the Laplacian matrix $L = X\Lambda X^\intercal$. A graph is produced by simple thresholding: the edge (i, j) is included iff $l_{i,j} < \theta$ (recall that a Laplacian matrix has negative off-diagonal entries for (i, j) that correspond to edges), with the threshold θ chosen so that the number of edges matches the target.
(7) Again in the spirit of [52], we consider a thresholding of our template matrix M. In this case, we include all edges (i, j) with $m_{i,j} > \theta$, again choosing θ such that the number of edges matches the target.

Our experimental results show that the template matrix spectrum closely matches the desired spectrum; thus, the deviation in the final spectrum is mostly a result of pushing the template matrix entries into $[0, 1]$, then rounding them. Using C as a template and using our rounding scheme normally suffices to produce strong performance on the spectral *gap*, but fails to match the other eigenvalues. This demonstrates that a careful choice in template helps preserve more of the spectrum. Thresholding the Laplacian and adjacency matrices

performs poorly not only on the overall spectrum, but even on the spectral gap. This confirms that a more careful rounding procedure and generation of a suitable template are necessary to generate graphs matching a desired spectrum.

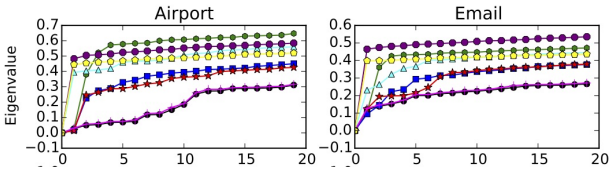

Figure 2: Spectra of matrices produced by employing different sets of heuristics. Input graph (black circles), rounding an unfitted matrix (green hexagons), fitted template matrix (magenta plus), fractional graph (cyan triangles), output graph without Stiefel (blue squares), output graph with Stiefel (red stars), thresholding the Laplacian (yellow pentagons), thresholding the adjacency matrix (purple octagons).

We observe that the Stiefel manifold optimization did not always improve performance of the final result, but more often than not provided small improvements. However, for large graphs, the computation becomes expensive.

8 CONCLUSION

We developed heuristics for randomly generating graphs whose Laplacian spectra approximately match that of a given input graph. Our experiments on multiple real-world network data sets show that for other graph parameters of interest, too, the graphs generated in this way match their real-world counterpart as well as or better compared to several widely used generative models.

Our work is nowhere near the final word on the subject, but should instead be considered as an invitation to the community to focus on spectral properties as an important criterion in random graph generative models.

Among the most obvious directions for future work is to devise faster heuristics. While our approach scales to graphs of several thousand nodes, real-world networks of interest often have millions of nodes; we would like to generate graphs of this size that match global spectral properties of given networks.

A second natural direction is to derive sampling algorithms with *provable* guarantees. The most obvious desirable guarantee would be that the spectrum of the generated graphs be "close" to Λ^* *provably*, or alternatively, that the distribution of deviations can be characterized. An even stronger type of guarantee would be to show that the generated graphs be drawn (nearly) *uniformly* from a suitable class of graphs with spectra similar to Λ^*.

A (perhaps more likely) alternative would be to prove hardness results. Paralleling the two tasks described in the previous paragraph, hardness results could come in two varieties: (1) show that deciding whether graphs exist whose Laplacian spectrum is close to a given Λ^* is hard, or (2) show that sampling nearly uniformly from graphs with the desired Laplacian spectra is hard.

The motivation for our work was that the spectrum naturally characterizes global connectivity properties. However, it is not the only graph metric to do so, and one could consider several others. For example, an alternative approach would be to sample graphs G such that for every partition $(S, \bar{S})$ of the vertices, the conductance (or expansion, or number of edges, or some other parameter) is approximately the same as in G^*. An obvious downside of this approach is that even to verify whether G is similar to G^* in this sense requires inspecting exponentially many cuts instead of linearly many eigenvalues, unless more efficient algorithms are designed.

Finally, the initial motivation for our work raises a very intriguing general direction. We implied (in Section 1) that even when local properties (such as degree distributions, triangle counts, other motifs) of a graph are fixed, generating graphs under such constraints still produces expander graphs with high probability[13]. Without any qualifications, this claim is definitely false — for example, a 3-regular graph in which each node participates in 3 triangles must be the union of disjoint 4-cliques, not an expander. However, under a careful formalization, we believe that such a claim should hold true, and it would be desirable to corroborate the intuition that local motifs do not constrain global structure enough to prevent most graphs from being expanders.

ACKNOWLEDGMENTS

We would like to thank Aaron Clauset, Aram Galstyan, Kristina Lerman, Elchanan Mossel, and Shanghua Teng for useful discussions and pointers, and anonymous reviewers for useful feedback. Work supported in part by NSF grant IIS-1619458 and ARO MURI 72924-NS-MUR.

REFERENCES

[1] P.-A. Absil, Robert Mahony, and Rodolphe Sepulchre. 2009. *Optimization algorithms on matrix manifolds.* Princeton University Press.
[2] P.-A. Absil and Jérôme Malick. 2012. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization* 22, 1 (2012), 135–158.
[3] Luca Baldesi, Carter T Butts, and Athina Markopoulou. 2018. Spectral Graph Forge: Graph Generation Targeting Modularity. In *Proc. 37th IEEE INFOCOM Conference.* IEEE, 1727–1735.
[4] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439 (1999), 509–512.
[5] Edward A. Bender and E. Rodney Canfield. 1978. The asymptotic number of labelled graphs with given degree sequences. *Journal of Combinatorial Theory (A)* 24 (1978), 296–307.
[6] Noam Berger, Christian Borgs, Jennifer Chayes, and Amin Saberi. 2005. On the Spread of Viruses on the Internet. In *Proc. 16th ACM-SIAM Symp. on Discrete Algorithms.* 301–310.
[7] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. 2018. NetGAN: Generating Graphs via Random Walks. In *Proc. 35th Intl. Conf. on Machine Learning.* 610–619.
[8] Béla Bollobás. 1998. Random Graphs. In *Modern Graph Theory.* Springer, 215–252.
[9] Alberto Borobia and Roberto Canogar. 2017. The real nonnegative inverse eigenvalue problem is NP-hard. *Linear Algebra Appl.* 522 (2017), 127 – 139.
[10] Steve Butler and Jason Grout. 2011. A construction of cospectral graphs for the normalized Laplacian. *The Electronic Journal of Combinatorics* 18, 1 (2011), 231.
[11] Chen Chen, Hanghang Tong, B. Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. 2016. Eigen-Optimization on large graphs by edge manipulation. *ACM Transactions on Knowledge Discovery from Data* 10, 4 (2016), 49.
[12] Moody T. Chu and Gene H. Golub. 2005. *Inverse Eigenvalue Problems: Theory, Algorithms, and Applications.* Oxford University Press.
[13] Fan R. K. Chung. 1997. *Spectral Graph Theory.* American Mathematical Society.
[14] Fan R. K. Chung and Linyuan Lu. 2002. The average distance in random graphs with given expected degrees. *Proc. Natl. Acad. Sci. USA* 99 (2002), 15879–15882.
[15] Fan R. K. Chung, Linyuan Lu, and Van Vu. 2003. The spectra of random graphs with given expected degrees. *Proc. Natl. Acad. Sci. USA* 100, 11 (2003), 6313–6318.

[13] Newman [43] alludes to a similar observation that random graphs tend to be "locally tree-like;" contrary to us, he considers this property to be desirable in a model to facilitate analysis.

[16] Fan R. K. Chung and Mary Radcliffe. 2011. On the spectra of general random graphs. *The Electronic Journal of Combinatorics* 18, 1 (2011), P215.

[17] Nicola De Cao and Thomas Kipf. 2018. MolGAN: An implicit generative model for small molecular graphs. In *ICML 2018 Workshop on Theoretical Foundations and Applications of Deep Generative Models*.

[18] Alan Edelman, Tomás A. Arias, and Steven T. Smith. 1998. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.* 20, 2 (1998), 303–353.

[19] Stanley C. Eisenstat and Ilse C. F. Ipsen. 1995. Relative perturbation techniques for singular value problems. *SIAM J. Numer. Anal.* 32, 6 (1995), 1972–1988.

[20] Matthew Fickus, Dustin G. Mixon, Miriam J. Poteet, and Nate Strawn. 2013. Constructing all self-adjoint matrices with prescribed spectrum and diagonal. *Advances in Computational Mathematics* 39, 3-4 (2013), 585–609.

[21] Miroslav Fiedler. 1973. Algebraic connectivity of graphs. *Czechoslovak mathematical journal* 23, 2 (1973), 298–305.

[22] Miroslav Fiedler. 1974. Eigenvalues of nonnegative symmetric matrices. *Linear Algebra Appl.* 9 (1974), 119–142.

[23] Miroslav Fiedler. 1975. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal* 25, 4 (1975), 619–633.

[24] Ove Frank and David Strauss. 1986. Markov Graphs. *J. Amer. Statist. Assoc.* 81, 395 (1986), 832–842.

[25] Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. 2006. Dependent rounding and its applications to approximation algorithms. *J. ACM* 53, 3 (2006), 324–360.

[26] Ayalvadi Ganesh, Laurent Massoulié, and Donald Towlsey. 2005. The effect of network topology on the spread of viruses. In *Proc. 24th IEEE INFOCOM Conference*. 1455–1466.

[27] Minas Gjoka, Maciej Kurant, and Athina Markopoulou. 2013. 2.5 k-graphs: from sampling to generation. In *Proc. 32nd IEEE INFOCOM Conference*. IEEE, 1968–1976.

[28] Chris D. Godsil and Brendan D. McKay. 1982. Constructing cospectral graphs. *Aequationes Mathematicae* 25, 1 (1982), 257–268.

[29] Paul W. Holland, Kathryn B. Laskey, and Samuel Leinhardt. 1983. Stochastic Blockmodels: Some First Steps. *Social Networks* 5 (1983), 109–137.

[30] Brian Karrer and Mark E. J. Newman. 2011. Stochastic Blockmodels and Community Structure in Networks. *Physical Review E* 83 (2011), 016107.

[31] Tosio Kato. 1987. Variation of Discrete Spectra. *Communications in Mathematical Physics* 111 (1987), 501–504.

[32] Vaishnavy Krishnamurthy, Michalis Faloutsos, Marek Chrobak, Jun-Hong Cui, Li Lao, and Allon G. Percus. 2007. Reducing Large Internet Topologies for Faster Simulations. *Computer Networks* 51, 15 (2007), 4284–4302.

[33] Thomas J. Laffey and Helena Šmigoc. 2007. Construction of nonnegative symmetric matrices with given spectrum. *Linear algebra and its applications* 421, 1 (2007), 97–109.

[34] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Physical Review E* 78 (2008), 046110.

[35] James R. Lee, Shayan Oveis Gharan, and Luca Trevisan. 2014. Multiway spectral partitioning and higher-order Cheeger inequalities. *J. ACM* 61, 6 (2014), 37.

[36] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. 2010. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research* 11 (2010), 985–1042.

[37] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph Evolution: Densification and Shrinking Diameters. *ACM Transactions on Knowledge Discovery from Data* 1, 1 (2007), 2.

[38] Jure Leskovec and Rok Sosič. 2016. SNAP: A General-Purpose Network Analysis and Graph-Mining Library. *ACM Transactions on Intelligent Systems and Technology* 8, 1 (2016), 1.

[39] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. 2018. Learning deep generative models of graphs. (2018). arXiv preprint arXiv:1803.03324.

[40] Russell Merris. 1997. Large families of Laplacian isospectral graphs. *Linear and Multilinear Algebra* 43, 1–3 (1997), 201–205.

[41] Michael Molloy and Bruce Reed. 1995. A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms* 6, 2-3 (1995), 161–180.

[42] Mark E. J. Newman. 2006. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E* 74 (2006), 036104.

[43] Mark E. J. Newman. 2009. Random graphs with clustering. *Physical Review Letters* 103, 5 (2009), 058701.

[44] Mark E. J. Newman, Steven H. Strogatz, and Duncan J. Watts. 2001. Random graphs with arbitrary degree distributions and their applications. *Physical Review E* 64, 2 (2001), 026118.

[45] Jorge Nocedal and Stephen Wright. 2006. *Numerical Optimization*. Springer.

[46] Chiara Orsini, Marija M. Dankulov, Pol Colomer-de Simón, Almerima Jamakovic, Priya Mahadevan, Amin Vahdat, Kevin E. Bassler, Zoltán Toroczkai, Marián Boguñá, Guido Caldarelli, Santo Fortunato, and Dmitri Kiroukov. 2015. Quantifying randomness in real networks. *Nature Communications* 6 (2015), 8627.

[47] Garry Robins, Philippa Pattison, Yuval Kalish, and Dean Lusher. 2007. An introduction to exponential random graph (p^*) models for social networks. *Social Networks* 29 (2007), 173–191.

[48] Christian L Staudt, Michael Hamann, Alexander Gutfraind, Ilya Safro, and Henning Meyerhenke. 2017. Generating realistic scaled complex networks. *Applied Network Science* 2, 1 (2017), 36.

[49] Yuchung J. Wang and George Y. Wong. 1987. Stochastic Blockmodels for Directed Graphs. *J. Amer. Statist. Assoc.* 82, 397 (1987), 8–19.

[50] Duncan J. Watts and Steven Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature* 393 (1998), 440–442.

[51] Zaiwen Wen and Wotao Yin. 2013. A feasible method for optimization with orthogonality constraints. *Mathematical Programming* 142, 1–2 (2013), 397–434.

[52] David White and Richard C. Wilson. 2007. Spectral generative models for graphs. In *Proc. Intl. Conf. on Image Analysis and Processing*. 35–42.

[53] David H. White. 2009. *Generative Models for Graphs*. Ph.D. Dissertation.

[54] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2019. A comprehensive survey on graph neural networks. (2019). arXiv preprint arXiv:1901.00596.

[55] Bai Xiao and Edwin R. Hancock. 2006. A spectral generative model for graph structure. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. 173–181.

[56] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. In *Proc. 35th Intl. Conf. on Machine Learning*. 5708–5717.