Learning Multifractal Structure in Large Networks

Austin R. Benson
Stanford University
Institute for Computational and Mathematical Engineering
arbenson@stanford.edu

Carlos Riquelme
Stanford University
Institute for Computational and Mathematical Engineering
rikel@stanford.edu

Sven Schmit
Stanford University
Institute for Computational and Mathematical Engineering
schmit@stanford.edu

ABSTRACT

Using random graphs to model networks has a rich history. In this paper, we analyze and improve the multifractal network generators (MFNG) introduced by Palla *et al*. We provide a new result on the probability of subgraphs existing in graphs generated with MFNG. This allows us to quickly compute moments of an important set of graph properties, such as the expected number of edges, stars, and cliques for graphs generated using MFNG. Specifically, we show how to compute these moments in time complexity independent of the size of the graph and the number of recursive levels in the generative model. We leverage this theory to propose a new method of moments algorithm for fitting MFNG to large networks. Empirically, this new approach effectively simulates properties of several social and information networks. In terms of matching subgraph counts, our method outperforms similar algorithms used with the Stochastic Kronecker Graph model. Furthermore, we present a fast approximation algorithm to generate graph instances following the multifractal structure. The approximation scheme is an improvement over previous methods, which ran in time complexity quadratic in the number of vertices. Combined, our method of moments and fast sampling scheme provide the first scalable framework for effectively modeling large networks with MFNG.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining

General Terms

Algorithms, Theory

Keywords

graph mining; real-world networks; multifractal; method of moments; graph sampling; stochastic kronecker graph; random graphs

KDD'14, August 24–27, 2014, New York, NY, USA.

ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623718.

1. RECURSIVE GRAPH STRUCTURE

Generative random graph models with recursive or hierarchical structure are successful in simulating large-scale networks [5, 15]. The recursive structure produces graphs with heavy-tailed degree distribution and high clustering coefficient. Random samples from recursive models are used to test algorithms, benchmark computer performance [13], anonymize data, and to understand the structure of networks.

A relatively new model is the multifractal network generators (MFNG, [14]). However, there are two issues that are barriers to making MFNG a practical model for large-scale networks. First, results for fitting MFNG models to graphs have been extremely limited. Current procedures can only match a single graph property, such as the number of nodes with degree d. Second, to our knowledge, all MFNG sampling techniques are $\mathcal{O}(|V|^2)$ algorithms, where V is the vertex set. This makes the generation of large graphs infeasible.

In this paper, we address both barriers and demonstrate that MFNG can be a better alternative to the more popular stochastic Kronecker graphs. In Section 3, we show how to compute several key properties of MFNG (e.g., expected number of edges, triangles, stars, etc.) with computational complexity independent of $|V|$ and the recursion depth. This result lets us develop an extremely efficient method of moments algorithm to fit networks to MFNG. We test our new method of moments algorithm on synthetic data and large social and information networks. In Section 5, we provide a heuristic fast approximate sampling scheme to randomly sample MFNG with complexity $\mathcal{O}(|E| \log |V|)$, where E is the edge set of the network. In Section 6.1, we show that our algorithm can identify model parameters in synthetic graphs sampled from MFNG, and in Section 6.2, we see that our algorithm can match the number of edges, wedges, triangles, 4-cliques, 3-stars, and 4-stars in large networks. Our contributions are summarized as follows:

- We show how to efficiently compute moments of several feature counts in random graphs generated with MFNG and use this to accelerate a method of moments algorithm for fitting large networks to MFNG.
- We provide a fast sampling algorithm for MFNG, so that large networks can be randomly generated in reasonable time.
- We empirically show that MFNG models graph feature counts better than alternatives and that global graph properties are accurately matched.

1.1 Related work

Popular recursive and hierarchical models include Stochastic Kronecker Graphs (SKG, [5]), Block Two-Level Erdős-Rényi (BTER, [15]), and Random Typing Generator (RTG, [1]). An older, popu-

lar model is the recursive matrix (R-MAT, [2]), which is a specific instance of an SKG model with a 2×2 generator matrix.

SKG is popular for several reasons including capturing degree distributions, clustering coefficients, and diameter in large networks. There are several methods for fitting SKG parameters to simulate a target network, including maximum likelihood estimation (the KronFit algorithm, [5, 6]) and the method of moments [3]. Maximum likelihood estimation is also used for the Multiplicative Attribute Graph model [4], and a simulated method of moments is used for mixed Kronecker product graph models [11, 12]. Finally, SKG produces graph samples in time complexity $\mathcal{O}(|E|\log(|V|))$ rather than $\mathcal{O}(|V|^2)$. On the other hand, SKG is constrained by a rather strong assumption on the relationship between the number of recursion levels and the number of nodes in the graph. Specifically, the number of recursive levels is $\lceil \log(|V|) \rceil$.

MFNG decouples the relationship between the recursion depth and the number of nodes and also naturally handles graphs where $|V|$ is not a power of two. While there are ad-hoc methods for SKG when $|V|$ is not a power of two, all analyses in the literature make the assumption. We do not assume that $|V|$ is a power of two in our analysis in Section 3. Furthermore, the variable interval lengths in MFNG allow for more flexibility than is offered by the SKG framework. These reasons make MFNG more flexible as a generator for graphs.

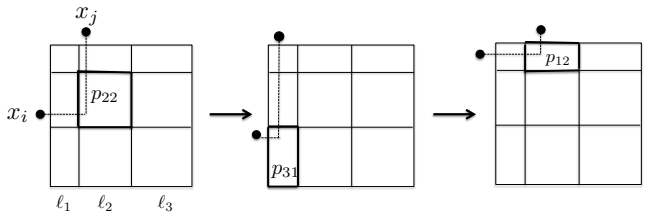
2. OVERVIEW OF MFNG

MFNG is a recursive generative model based on a generating measure, $\mathcal{W}_k$. The measure $\mathcal{W}_k$ consists of an m-vector of lengths ℓ with $\sum_{i=1}^m \ell_i = 1$ and a symmetric $m \times m$ probability matrix $\mathbf{P}$. The subscript k is the number of recursive levels, which we will subsequently explain. In this paper, we refer to the m indices of ℓ as *categories*. Also, since the measure is completely characterized by $\mathbf{P}$, ℓ, and k, we write $\mathcal{W}_k(\mathbf{P}, \ell)$ to explicitly describe the full measure.

An undirected graph $G = (V, E)$ is distributed according to $\mathcal{W}_k(\mathbf{P}, \ell)$ if it is generated by the following procedure:

1. Partition $[0, 1]$ into m subintervals of length $\ell_i, i = 1, \ldots, m$. Recursively partition each subinterval k times into m pieces, using the relative lengths ℓ_i. This creates m^k intervals $\ell_{i_1,\ldots,i_k}$ of length $\prod_{r=1}^k \ell_{i_r}$ such that $\sum_{i_1,\ldots,i_k} \ell_{i_1,\ldots,i_k} = 1$.
2. Sample N points uniformly from $[0, 1]$ and create the nodes $V = \{x_1, \ldots, x_N\}$. Each node x_i is identified by its k-tuple of categories $c(x_i) = (i_1, \ldots, i_k)$, based on its position on $[0, 1]$ and the partitioning in Step 1.
3. For every pair of nodes x_i and x_j identified by the k-tuple of categories $c(x_i) = (i_1, \ldots, i_k)$ and $c(x_j) = (j_1, \ldots, j_k)$, add edge (x_i, x_j) to G with probability $\prod_{r=1}^k p_{i_r j_r}$.

While the generation is intricate, MFNG admit a geometric interpretation. Consider first the partition of the unit square into m^2 rectangles according to the lengths ℓ. The rectangle in position (q, s) has side lengths ℓ_q and ℓ_s, $1 \le q, s \le m$. The point $(x_i, x_j) \in [0, 1] \times [0, 1]$ lands in the unit square, inside some rectangle R with side lengths ℓ_{i_1} and ℓ_{j_1}. The edge 'survives' the first round with probability p_{i_1,j_1}. In the next round, we recursively partition R according to the lengths ℓ. The relative positions of x_i and x_j land the point in a new rectangle with side lengths ℓ_{i_2} and ℓ_{j_2}. The edge survives the second round with probability p_{i_2,j_2}. The process is repeated k times and is illustrated in Figure 1. If an edge survives all k levels, then it is added to the graph.



$$P((x_i, x_j) \in G) = p_{22}p_{31}p_{12}$$

Figure 1: MFNG's recursive edge generation with $m = k = 3$.

3. THEORETICAL RESULTS

The original work on MFNG [14] shows how to compute the expected feature counts for graph properties by examining the entire expanded measure $\mathcal{W}_k(\mathbf{P}, \ell)$. In other words, to count the features, the entire probability matrix of size $m^k \times m^k$ is formed. However, in some cases m^k is of the order of $\mathcal{O}(|V|)$ (see the examples in Section 6.2). Clearly, computing and storing $\mathcal{O}(|V|^2)$ probabilities is infeasible for large networks. Thus, current methods for counting and fitting features are intolerably expensive. Theorem 2 shows that we can count many of the same features by only looking at the probability matrix $\mathbf{P}$ a constant number of times (independent of $|V|$). Hence, we are able to scale these computations to graphs with a large number of nodes.

3.1 Decoupling of recursive levels

We start with a lemma that shows how to decompose a generating measure $\mathcal{W}_k$ with k recursive levels in k measures with depth one. This will make it easier to count subgraphs in Theorem 2.

LEMMA 1. *Consider generating measures $\mathcal{W}_1(\mathbf{P}, \ell)$ and $\mathcal{W}_k(\mathbf{P}, \ell)$, which are parameterized by the same probability matrix $\mathbf{P}$ and lengths ℓ but different recursion depths. Let graphs $H_1, \ldots, H_k \sim \mathcal{W}_1(\mathbf{P}, \ell)$ be independently drawn, and also denote $H_i = (V, E_i)$, with nodes labelled arbitrarily. Then the intersection graph $G = (V, \cap_{i=1}^k E_i) = (V, E_G) \sim \mathcal{W}_k(\mathbf{P}, \ell)$.*

PROOF. We prove the lemma by conditioning on the categories to which the nodes belong (recall that a category is the set of intervals that a node falls into at each level of the recursion). Each node $u \in V$ is identified with some real number in $[0, 1]$. The probability that the k-tuple of categories corresponding to u is $c(u) = (c_1, \ldots, c_k)$ in any graph $H \sim \mathcal{W}_k(\mathbf{P}, \ell)$ is simply $\prod_{r=1}^k \ell_{c_r}$. By independence of the H_i, the probability that the same node u is in the same categories $c_1, \ldots, c_k$ in the graphs $H_1, \ldots, H_k$, respectively, is also $\prod_{r=1}^k \ell_{c_r}$. Note that

$$\begin{aligned}
&\mathbb{P}\left((u, v) \in E_G | c(u) = (c_1^u, \ldots, c_k^u), c(v) = (c_1^v, \ldots, c_k^v)\right) \\
=&\mathbb{P}\left((u, v) \in \cap_{i=1}^k E_i | c(u) = (c_1^u, \ldots, c_k^u), c(v) = (c_1^v, \ldots, c_k^v)\right) \\
=&\prod_{i=1}^k \mathbb{P}\left((u, v) \in E_i | c(u) = (c_1^u, \ldots, c_k^u), c(v) = (c_1^v, \ldots, c_k^v)\right) \\
=&\prod_{i=1}^k \mathbb{P}\left((u, v) \in E_i | [c(u)]_i = c_i^u, [c(v)]_i = c_i^v\right) = \prod_{i=1}^k p_{c_i^u, c_i^v}.
\end{aligned}$$

In the first equality, we use the definition of E_G; in the second and third equalities, we use the independence of the H_i; and in the final equality, we use the definition of $\mathcal{W}_1$. However, for any graph

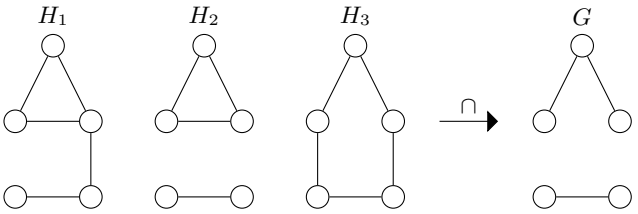


Figure 2: Illustration of Lemma 1. If three graphs H_1, H_2, and H_3 are generated from $\mathcal{W}_1(\mathbf{P}, \ell)$, then their intersection G follows the distribution of $\mathcal{W}_3(\mathbf{P}, \ell)$.

$G' \sim \mathcal{W}_k(\mathbf{P}, \ell)$, we have

$$\begin{aligned}&\mathbb{P}\left((u,v) \in G' | c(u) = (c_1^u, \ldots, c_k^u), c(v) = (c_1^v, \ldots, c_k^v)\right)\\ =&\prod_{i=1}^{k} p_{c_i^u, c_i^v}.\end{aligned}$$

□

Figure 2 illustrates Lemma 1. Our main result is a straightforward consequence of this lemma.

THEOREM 2. *Let $\mathcal{W}_1(\mathbf{P}, \ell)$ and $\mathcal{W}_k(\mathbf{P}, \ell)$ be generating measures defined by the same probabilities $\mathbf{P}$ and lengths ℓ but with different recursion depths. Consider k multifractal graphs $H_i = (V, E_i)$ generated independently from $\mathcal{W}_1(\mathbf{P}, \ell)$ and a multifractal graph $G = (V, E_G)$ generated from $\mathcal{W}_k(\mathbf{P}, \ell)$. For any event A on G that can be written as $A = \{S \subset E_G\}$, where $S \subset \{(i,j) : i, j \in \{1, \ldots, n\}, i < j\}$,*

$$\mathbb{P}_{\mathcal{W}_k}(A) = \mathbb{P}_{\mathcal{W}_1}(A)^k.$$

PROOF.

$$\begin{aligned}\mathbb{P}_{\mathcal{W}_k}(A) &= \mathbb{P}_{\mathcal{W}_k}(s \in E_G, \forall s \in S)\\ &= \mathbb{P}_{(\mathcal{W}_1)^k}(s \in E_i, \forall s \in S, \forall i \in \{1, \ldots, k\})\\ &= \prod_{i=1}^{k} \mathbb{P}_{(\mathcal{W}_1)^k}(s \in E_i, \forall s \in S)\\ &= \mathbb{P}_{(\mathcal{W}_1)^k}(s \in E_1, \forall s \in S)^k\\ &= \mathbb{P}_{\mathcal{W}_1}(A)^k.\end{aligned}$$

□

In other words, the probability that a subset of the edges exists if the graph is drawn from $\mathcal{W}_k$ is the k-th power of the probability that these edges exist if the graph is drawn from $\mathcal{W}_1$. The condition that A can be written as $A = \{S \subset E\}$ is subtle. It states that Theorem 2 holds if we can specify a subset of the edges that must be present in the graph. We can be indifferent about certain edges, but we cannot specify that an edge is *not* present in the graph.

We can now easily compute the moments of subgraph counts, such as the number of edges, triangles, and larger cliques in MFNG. The following corollary shows how to use Theorem 2 for these calculations. for graphs generated by MFNG.

COROLLARY 3. *The expected number of edges $|E|$ in a graph sampled from MFNG is*

$$\mathbb{E}[|E|] = \binom{n}{2} s^k, \tag{1}$$

where $s = \sum_{i,j \in [m]} p_{ij} \ell_i \ell_j$.

PROOF. Let u and v, $u \neq v$, be two random nodes of G. Let A denote the event $A = \{(u,v) \in E\}$, and we define A_i to denote the analogous event restricted to H_i in the multifractal generator. By Theorem 2, we have that

$$\mathbb{P}(A) = \prod_{i=1}^{k} \mathbb{P}(A_i) = \mathbb{P}(A_1)^k.$$

Now we can restrict ourselves to A_1,

$$\begin{aligned}\mathbb{P}(A_1) &= \sum_{i,j \in [m]} \mathbb{P}(A^{(1)} | c_1^u = i, c_1^v = j)\, \mathbb{P}(c_1^u = i, c_1^v = j) &(2)\\ &= \sum_{i,j \in [m]} p_{ij}\, \mathbb{P}(c_1^u = i) \mathbb{P}(c_1^v = j) &(3)\\ &= \sum_{i,j \in [m]} p_{ij}\, \ell_i \ell_j = s. &(4)\end{aligned}$$

We conclude that

$$\mathbb{P}(A) = \mathbb{P}((u,v) \in E) = s^k. \tag{5}$$

The expected number of edges is then given by

$$\mathbb{E}[|E|] = \binom{n}{2} s^k. \tag{6}$$

□

COROLLARY 4. *Graphs sampled from MFNG also have the following moments. The expected number of d-stars [1] S_d is:*

$$\mathbb{E}[S_d] = n \binom{n-1}{d} \left(\sum_{i_1, \ldots, i_{d+1} \in [m]} \prod_{j=2}^{d+1} p_{i_1 i_j} \prod_{j=1}^{d+1} \ell_{i_j} \right)^k.$$

In particular, the expected number of wedges (2-stars) is

$$\mathbb{E}[S_2] = n \binom{n-1}{2} \left(\sum_{i_1, i_2, i_3 \in [m]} p_{i_1 i_2} p_{i_1 i_3}\, \ell_{i_1} \ell_{i_2} \ell_{i_3} \right)^k.$$

The variance $\sigma_E = Var(|E|)$ of the number of edges is

$$\sigma_E = \binom{n}{2} s^k \left(1 - \binom{n}{2} s^k\right) + 2\, \mathbb{E}[S_2] + \binom{n}{2} \binom{n-2}{2} s^{2k},$$

where s is the same as in Corollary 3.

The expected number of t-cliques [2] C_t is

$$\mathbb{E}[C_t] = \binom{n}{t} s_t^k, \tag{7}$$

where

$$s_t := \sum_{i_1, \ldots, i_t \in [m]} \left(\prod_{\substack{j,q \in [t]\\ j<q}} p_{i_j i_q} \right) \ell_{i_1} \ell_{i_2} \cdots \ell_{i_t}. \tag{8}$$

In particular, the expected number of triangles (3-cliques) is:

$$\mathbb{E}[C_3] = \binom{n}{3} \left(\sum_{i,j,t \in [m]} p_{ij} p_{it} p_{jt}\, \ell_i \ell_j \ell_t \right)^k. \tag{9}$$

[1] A d-star is a graph with $d+1$ vertices and d edges that connect the first node to all other vertices.

[2] A t-clique is a graph with t vertices where every possible edge between the vertices exists.

Finally, the expected number of nodes with degree d, E_d, satisfies $\mathbb{E}[E_{|V|-1}] = \mathbb{E}[S_{|V|-1}]$ *and*

$$\mathbb{E}[E_d] = \mathbb{E}[S_d] - \sum_{i=d+1}^{|V|-1} \binom{i}{d} \mathbb{E}[E_i]. \tag{10}$$

PROOF. The proofs follow the same patterns as of the proof of Corollary 3. We include the proofs in supplementary material online [3]. □

These are some examples of properties for which we can compute the *exact* expectation. However, we can also compute useful approximations. For a given measure $\mathcal{W}_k$, we could empirically compute the value of $\mathbb{E}[C_t]$ for each t until we find $\mathbb{E}[C_{t^*}] \geq 1 > \mathbb{E}[C_{t^*+1}]$, which is a good estimator of the expected maximum clique size.

Finally, we note that there are graph properties which will certainly not translate to this theoretical framework. Let $\mu(G)$ be the chromatic number of G, i.e., the smallest number of colors needed to color the vertices such that vertices connected by an edge are not the same color. Suppose we want to compute $\mathbb{P}(\mu(G) < 10)$. If the theorem is used directly, then the result is $\mathbb{P}(\mu(G) < 10) = \mathbb{P}(\mu(H_1) < 10)^k$. But $\mathbb{P}(\mu(G) < 10) \geq \mathbb{P}(\mu(H_1) < 10)$ since taking the intersection of graphs can only reduce the chromatic number. In this case, $\mathbb{P}(\mu(G) < 10)$ cannot be written as an event on the subset of the edges of the graph. Hence, the assumptions of the theorem are violated.

4. METHOD OF MOMENTS LEARNING ALGORITHM

Now we change gears and look at how we can use the theory laid out above to fit multifractal measures to real networks. Given a graph G, we are interested in finding a probability matrix $\mathbf{P}$, a set of lengths ℓ, and a recursion depth k, such that graphs generated from the measure $\mathcal{W}_k(\mathbf{P}, \ell)$ are similar to G. The theoretical results in Section 3 make it simple to compute moments for MFNG, so a method of moments is natural. In particular, given a set of desired features counts f_i (such as number of edges, 2-stars, and triangles), we seek to solve the following optimization problem:

$$\begin{aligned} \underset{\mathbf{P},\ell,k}{\text{minimize}} \quad & \sum_i \frac{|f_i - \mathbb{E}_{\mathcal{W}_k}[F_i]|}{F_i} \\ \text{subject to} \quad & 0 \leq p_{ij} = p_{ji} \leq 1, \qquad 1 \leq i \leq j \leq m \\ & 0 \leq \ell_i \leq 1, \qquad\qquad\qquad 1 \leq i \leq m \\ & \sum_{i=1}^{m} \ell_i = 1 \end{aligned} \tag{11}$$

Here, F_i denotes the actual count of feature f_i in the MFNG.

If certain features are more important to fit, then we can weight the terms in the objective function, but for simplicity of our numerical experiments, we only use an unweighted objective in this paper. Similar objective functions were proposed for SKG [3] and for mixed Kronecker product graph models [12]. In Section 6, we see that the simple objective function works well on synthetic and real data sets.

4.1 Desired features

We want to model real world networks accurately and efficiently. Theorem 2 shows that, given a generating measure $\mathcal{W}_k(\mathbf{P}, \ell)$, we can quickly compute moments of several (local) feature counts. However, (global) graph properties such as degree distribution and clustering coefficient are not covered by our theoretical results. Therefore, we use local feature counts, such as number of d-stars and t-cliques[4], as a proxy. If the number of d-stars and t-cliques are similar, then we expect the degree distribution and clustering to be similar as well. For example, the global clustering coefficient is three times the ratio of the number of triangles (3-cliques) to the number of wedges (2-stars) in the graph. In Section 6.2, we show that matching star and clique subgraph counts in social and information networks leads to a generating measure that produces graphs with a similar degree distribution.

4.2 Solving the optimization problem

Optimization problem (11) is not trivial to solve, as there are many local minima and some of them turn out to be very poor. On the other hand, if we are given the feature counts of a graph and fix k, running a standard optimization solver such as `fmincon` in Matlab, we find a critical point quickly: we only have to fit $m^2 + m + 1$ variables, where, typically, m is two or three. Thus, we solve the optimization problem with many random restarts and use the best result. For each random restart, we first choose a random k and then solve the optimization problem with k fixed. We demonstrate that this crude method works on several practical examples (see Section 6). More sophisticated methods are beyond the scope of this work. We also point out that the bottleneck of the estimation is performing the feature counts, not solving the above optimization problem (despite the many restarts).

5. FAST SAMPLING FOR SPARSE GRAPHS

In this section, we discuss a heuristic method for generating sample graphs following the multifractal measure that is effective when the graph to be generated is sparse, i.e. has relatively few edges. This is important because the naive sampling method takes $\mathcal{O}(|V|^2)$ time—it considers the edge for every pair of nodes in the graph. The fast heuristic algorithm is inspired by the "ball-dropping" scheme for SKG (see Section 3.6 of [5]). Unlike the SKG case, however, our algorithm is merely a heuristic due to the stochastic nature of the location of the nodes. The speed-up is obtained by fixing the number of edges in advance and only considering $\mathcal{O}(|E|)$ pairs of vertices. We will demonstrate that our sampling algorithm runs in time $\mathcal{O}(|E| \log(|V|))$. The pseudo-code is given in Algorithm 1. In the next sections, we give the details of the algorithm and briefly discuss the performance.

5.1 The algorithm

In order to avoid looping over all pairs of nodes, we fix the number of edges. The number of edges is sampled from a normal random variable with mean $\mathbb{E}[|E|]$ and variance σ_E, as provided by Corollaries 3 and 4. Since the number of edges is a sum of Bernoulli trials, this is well approximated using a Gaussian random variable.

Once the number of edges in the graph is fixed, we start adding edges. Because node locations are random (i.e., every node has a random category), it is nontrivial to select a candidate edge. This contrasts with SKG, where the edge probabilities for a given node is deterministic. The algorithm selects node categories level by level, for each edge. To select categories, we sample an index (c, c') of a matrix $\mathbf{Q}$:

$$\mathbf{Q}_{ij} = p_{ij} \ell_i \ell_j.$$

[3] `http://stanford.edu/~arbenson/mfng.html`

[4] From now on we implicitly mean counting subgraphs if we say counting d-stars or t-cliques.

The sampling is done proportional to the entries in $\mathbf{Q}$. The matrix $\mathbf{Q}$ reflects the relative probability mass corresponding to an edge falling into those categories. In other words, it is the probability of selecting the categories c and c' at a given level *and* the edge surviving the level. The category sampling is performed k times, one for each level of recursion. This gives two k-tuples of categories: $c = (c_1, \ldots, c_k)$ and $c' = (c'_1, \ldots, c'_k)$.

Now we want to add an edge between nodes u and u' that have the categories c and c'. However, we have to be careful about the number of nodes that have the same category. We can think of the category pair (c, c') as a box B on the generating measure. Consider two boxes B_1 and B_2 and suppose that both have the same area in the unit square, and the probability between potential boxes in B_1 and B_2 is the same.

A simple example is the following case:

- $k = m = 2$
- $p_{11} = p_{22} = p_{12} = 0.5$
- $\ell_1 = \ell_2 = 0.5$

The edge probabilities in any two boxes B_1 and B_2 in the measure are the same, and the probabilities of selecting either box (from sampling the $\mathbf{Q}$ matrix) are the same. However, because of the randomness categories for nodes, there may be 10 node pairs in B_1 and only one node pair in B_2. If we simply pick a node pair at random from a box, the probability of connecting the node pair in B_2 is much higher than in B_1.

To overcome this discrepancy, we take into account the difference between the *expected* number of nodes pairs in a box and the *actual* number of node pairs in a box. Note that the joint distribution of nodes is Multinomial$(n; l_1, l_2, \ldots, l_{m^k})$ where l_i denotes the length of interval i (after recursive expansion). Let $p_{c,c'}$ be the edge probability in the box corresponding to the category pair (c, c'). Let the box's sides have lengths l and l'. Using standard properties of the Multinomial distribution, the expected number of nodes in a box, $n_{c,c'}$, is:

$$n_{c,c'} = \begin{cases} |V|(|V|l^2 - l^2 + l) & \text{if } c = c' \\ |V|(|V| - 1)ll' & \text{if } c \neq c' \end{cases}$$

Finally, we sample

$$e_{\text{to add}} \sim \text{Poisson}\left(\frac{n_{c,c'}}{\lambda |V_c||V_{c'}|}\right),$$

where $V_c = \{v \in V | \text{category of v is } c\}$. We then add $e_{\text{to add}}$ edges to the box (c, c'). Thus, if there are more node pairs in a box than expected, we add more edges to the box.

There are a couple of details we have swept under the rug. First, we haven't discussed what to do if the box (c, c') is empty. In this case, we simply re-sample c and c'. In practice, this does not occur too frequently. Second, we have introduced some dependence between edges, and MFNG samples edges independently. For this reason, we use the accuracy factor λ. By increasing λ, the sampling takes longer, but there is less dependency between edges.

5.2 Performance

The speedup achieved by this fast approximation algorithm really depends on the type of graph. We trade an $\mathcal{O}(|V|^2)$ algorithm for an algorithm that takes $\mathcal{O}(|E| \log |V|)$ time if there are no rejected tries due to empty boxes, edges that are already present, etc. In the case that the graph is sparse and $k <\approx \log_m n$, this is fine. However, for denser graphs, this fast method will actually turn out to be slower. To arrive at a complexity of $\mathcal{O}(|E| \log |V|)$ we note that it takes $\mathcal{O}(|V| \log |V|)$ time to compute the categories for each

Algorithm 1 Fast approximate sampling algorithm

Input: Generating measure $\mathcal{W}_k(\mathbf{P}, \ell)$, accuracy factor λ
Output: Graph G with distribution approximately $\mathcal{W}_k(\mathbf{P}, \ell)$.
Add $|V|$ nodes by uniformly sampling on $[0, 1]$ and assigning the proper categories to each node.
Set $V_c = \{v \in V | \text{category of v is } c\}$ for each category c.
Fix number of candidate edges $|E| = \lfloor \mathcal{E} \rfloor$, where $\mathcal{E} \sim N(\mu_{|E|}, \sigma_{|E|})$.
Compute $\mathbf{Q}$, where $Q_{ij} = p_{ij}\ell_i\ell_j$ for $1 \leq i, j, \leq m$
Set $e_{\text{global}} = 0$
while $e_{\text{global}} < |E|$ **do**
  for $h = 1$ **to** k **do**
    Pick category c_h, c'_h independently and with probability proportional to $\mathbf{Q}_{c_h, c'_h}$
  end for
  $c = (c_1, \ldots, c_k)$, $c' = (c'_1, \ldots, c'_k)$.
  Set l, l' to lengths of interval corresponding to c, c'
  if $|V_c||V_{c'}| \neq 0$ **then**
    if $c = c'$ **then**
      $n_{c,c'} = |V|(|V|l^2 - l^2 + l)$
    else
      $n_{c,c'} = |V|(|V| - 1)ll'$
    end if
    Draw $e_{\text{to add}} \sim$Poisson$(n_{c,c'}/(\lambda |V_c||V_{c'}|))$
    Set k = 0
    Set $e_{\text{local}} = 0$
    while $e_{\text{local}} < e_{\text{to add}}$ and $k < \max_k$ **do**
      Pick $u \in V_c$ and $v \in V_{c'}$ uniform at random.
      if $(u, v) \notin E$ and $u \neq v$ **then**
        Add (u, v) to E
        Set $e_{\text{local}} = e_{\text{local}} + 1$
      end if
      Set $k = k + 1$
    end while
    $e_{\text{global}} = e_{\text{global}} + e_{\text{local}}$
  end if
end while
Return $G = (V, E)$

$u \in V$. Then, assuming that the number of retries is small, the while loop of Algorithm 1 is executed $\mathcal{O}(|E|)$ times, each taking $\mathcal{O}(k) = \mathcal{O}(\log |V|)$ steps. Therefore, in total, the algorithm has complexity $\mathcal{O}(|E| \log |V|)$.

6. EXPERIMENTAL RESULTS

In the next sections, we demonstrate the effectiveness of our approach to modeling networks. First, we show that our method is able to recover the multifractal structure if we generate synthetic graphs following the MFNG paradigm. Thereafter, we consider several real-world networks and compare the performance of our method to alternative methods that use the SKG framework.

6.1 Identifiability and learning synthetic networks

Before turning to real networks, it is important to see if our method of moments algorithm recovers the structure of graphs that are actually generated by MFNG with some measure $\mathcal{W}_k$. In other words, can our method of moments identify graphs generated from our model? There are two success metrics for recovery of the generating measure. First, we want the method of moments to recover a measure similar to $\mathcal{W}_k$. Second, even if we cannot recover the

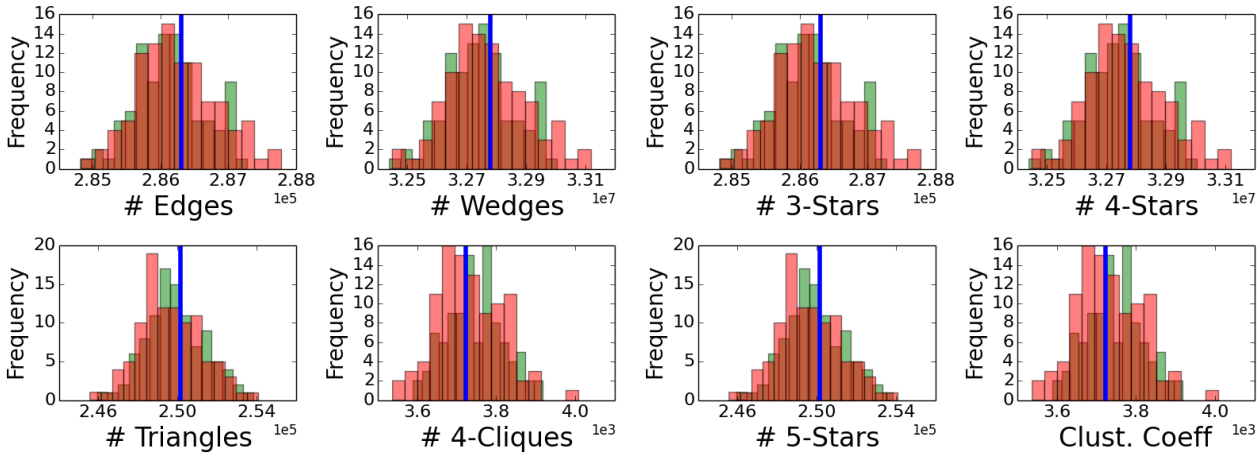


Figure 4: Empirical distributions of feature counts and clustering coefficient for the original MFNG (green) and the retrieved MFNG (red) found with the method of moments algorithm. The blue line is the feature count from the single sample of the original MFNG used in the method of moments. In this case, the original MFNG followed an Erdős-Rényi model. The original and retrieved measures produce similar distributions.

Generating Measure	$\lvert V\rvert$	m	k	ℓ_1	ℓ_2	p_{11}	p_{12}	p_{22}
Original $\mathcal{W}_k$	5,000	2	12	0.5	0.5	0.73	0.73	0.73
Retrieved $\bar{\mathcal{W}}_{\bar{k}}$	5,000	2	10	0.0574	0.9425	0.0074	0.7273	0.6829

Table 1: Comparison of original measure to the measure retrieved by using the method of moments algorithm from Section 4. The graph features used for the method of moments were: number of edges, number of d-stars for $d = 2, 3, 4, 5$, and number of t-cliques for $t = 3, 4$. The original generative measure is an Erdős-Rényi random graph model. While the recursion depth, probabilities, and lengths vector are quite different, the retrieved measure is similar to the same Erdős-Rényi model (see the discussion in Section 6.1).

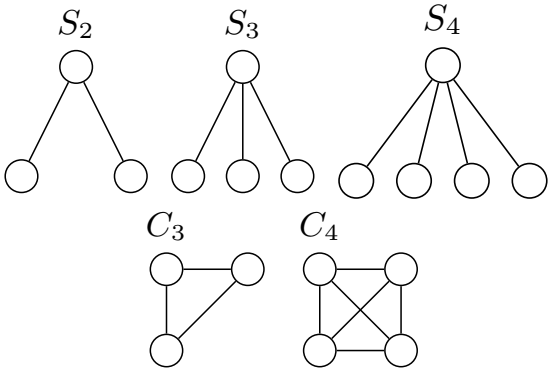


Figure 3: d-stars and t-cliques features that are counted in the experiments in Section 6.

measure, we want a measure that has similar feature counts. Our experiments in this section show that we can be successful in both metrics. If we can recover a measure with similar moments, then the new measure will be a useful model for the old one.

Our basic setup is as follows:

1. Construct a measure $\mathcal{W}_k(\mathbf{P}, \ell)$ and generate a *single* graph G from the measure.
2. Run the method of moment algorithm from Section 4 with G using 10,000 random restarts. Fit the moments for the following graph features: number of edges, number of d-stars for $d = 2, 3, 4, 5$, and number of t-cliques for $t = 3, 4$. The measure given by the method of moments is denoted $\bar{W}_k(\bar{\mathbf{P}}, \bar{\ell})$.
3. To compare $\mathcal{W}_k(\mathbf{P}, \ell)$ and $\bar{W}_k(\bar{\mathbf{P}}, \bar{\ell})$, sample 100 graphs from each measure and look at the histogram of the features that were considered by the method of moments algorithm.

We use two different measures $\mathcal{W}_k$ for testing. The first is equivalent to an Erdős-Rényi random graph. This is modeled by a generating measure $\mathcal{W}_k(\mathbf{P}, \ell)$ where every entry of $\mathbf{P}$ is identical. In this case, MFNG is an Erdős-Rényi generative model with edge probability $\mathbf{P}_{11}^k$, independent of ℓ. Table 1 shows the retrieved measure $\bar{\mathcal{W}}_{\bar{k}}(\bar{\mathbf{P}}, \bar{\ell})$ and the original Erdős-Rényi measure $\mathcal{W}_k(\mathbf{P}, \ell)$. While $\bar{\mathbf{P}}$ and $\bar{\ell}$ are quite different than $\mathbf{P}$ and ℓ, $\bar{\mathcal{W}}_{\bar{k}}(\bar{\mathbf{P}}, \bar{\ell})$ still represents a measure close to an Erdős-Rényi random graph model. The reason is that the length vector ℓ is heavily skewed to the second component ($\ell_2 \approx 0.94$). In expectation, $0.94^{\bar{k}} \approx 0.53$ of the nodes correspond to the same category. These nodes are all connected with probability $0.6829^{\bar{k}} \approx 0.022$, which is nearly the same as the edge probability in the original Erdős-Rényi measure. Figure 4 shows the histograms of the features that were used in the method of moments algorithms (as well as the clustering coefficient). The green histogram is the data for graphs sampled from $\mathcal{W}_k(\mathbf{P}, \ell)$, the red histogram is the same data for graphs sampled from $\bar{\mathcal{W}}_{\bar{k}}(\bar{\mathbf{P}}, \bar{\ell})$, and the blue line is the feature count in the original graph G used as input to the method of moments. There is remarkable overlap between the empirical distribution of the features for $\bar{\mathcal{W}}_{\bar{k}}(\bar{\mathbf{P}}, \bar{\ell})$ and the distribution of the features for the original measure.

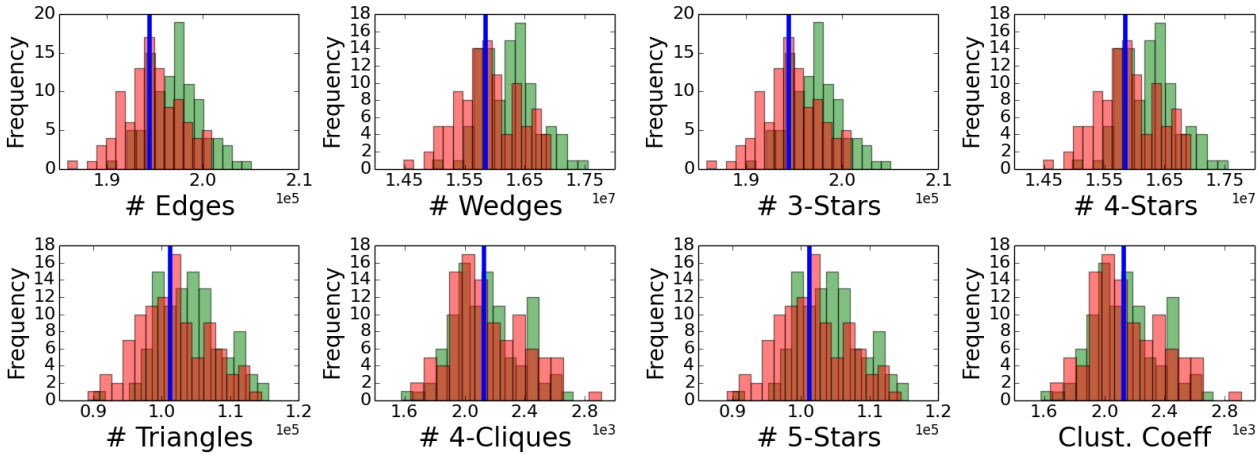


Figure 5: Empirical distributions of feature counts and clustering coefficient for the original MFNG (green) and the retrieved MFNG (red) found with the method of moments algorithm. The blue line is the feature count from the single sample of the original MFNG used in the method of moments. The original and retrieved measures produce similar distributions.

Generating Measure	$\lvert V\rvert$	m	k	ℓ_1	ℓ_2	p_{11}	p_{12}	p_{22}
Original $\mathcal{W}_k$	6,000	2	10	0.25	0.75	0.59	0.43	0.78
Retrieved $\bar{\mathcal{W}}_{\bar{k}}$	6,000	2	9	0.2728	0.7272	0.5431	0.4101	0.7593

Table 2: Comparison of original measure to the measure retrieved by using the method of moments algorithm from Section 4. The graph features used for the method of moments were: number of edges, number of d-stars for $d = 2, 3, 4, 5$, and number of t-cliques for $t = 3, 4$. All parameters in the retrieved measure are remarkably similar to the parameters in the original measure.

For a second experiment, we used an original measure $\mathcal{W}_k(\mathbf{P}, \ell)$ that did not possess the uniform generative structure of Erdős-Rényi random graphs. Table 2 shows the retrieved measure and the original measure. In this case, the method of moments identified a similar generative measure. The parameters $\bar{k}$, $\bar{\mathbf{P}}$, and $\bar{\ell}$ are remarkably similar to k, $\mathbf{P}$, and ℓ. Figure 5 shows the distribution of the features in graphs sampled from the two measures. Again, there is rather significant overlap in the empirical distributions.

Finally, we compare the degree distributions of the original and retrieved measures in Figure 6. The degree distributions are nearly identical.

These results show that the method of moments algorithm described in Section 4 can successfully identify MFNG instances using a single sample.

6.2 Learning real networks

We now show how the method of moments from Section 4 performs when fitting to the following four real-world networks to MFNG:

1. The Gnutella graph is a network of host computers sharing files on August 31, 2012 [8].
2. The Citation network is from a set of high energy physics papers from arXiv [7].
3. The Twitter network is a combination of several ego networks from the Twitter follower graph [9].
4. The Facebook network is a combination of several ego networks from the Facebook friend graph [9].

All data sets are from the SNAP collection. We use the optimization procedure described in Section 4 with 2,000 random restarts. The features we use (the f_i in Section 4) are number of edges, wedges (S_2), 3-stars (S_3), 4-stars (S_4), triangles (C_3), and 4-cliques (C_4). For each network, we use $m = 2, 3$ and $k = \lceil \log_m(|V|) \rceil$. With these values of m we are able to effectively fit the networks to MFNG. We do not believe that larger values of m are useful: we would need to estimate too many parameters and we lose a lot in interpretability of results. While k can be arbitrary, a smaller value of k leads to many nodes belonging to the same categories and hence having the same statistical properties. In large graphs, this causes a "clumping" of properties such as degree distribution near a small set of discrete values. While smaller k may be satisfactory for testing algorithms, keeping k near $\log_m(|V|)$ produces more realistic graphs. In an additional set of experiments, we only fit the number of edges, wedges, and triangles. We also compare against KronFit and the SKG method of moments [3].

The results are summarized in Table 3. In addition, the online material lists all recovered parameters. Overall, for both $m = 2$ and $m = 3$, the method of moments can effectively match most feature counts. The number of 4-stars (S_4) was the most difficult parameter to fit. We see that when only fitting the number of edges, wedges, and triangles, the other feature moments can be significantly different from the original graph. In particular, the number of 4-cliques tends to be severely under- or over-estimated. Although KronFit does not explicitly try to fit moments, the results show that it severely underestimate several feature counts. The method of moments approach to SKG can fit three of the features, which is consistent with results on other networks [3].

As mentioned in Section 4.1, the clustering coefficient is three times the ratio of the number of triangles (3-cliques) to the number of wedges (2-stars) in the graph. The results of Table 3 show that

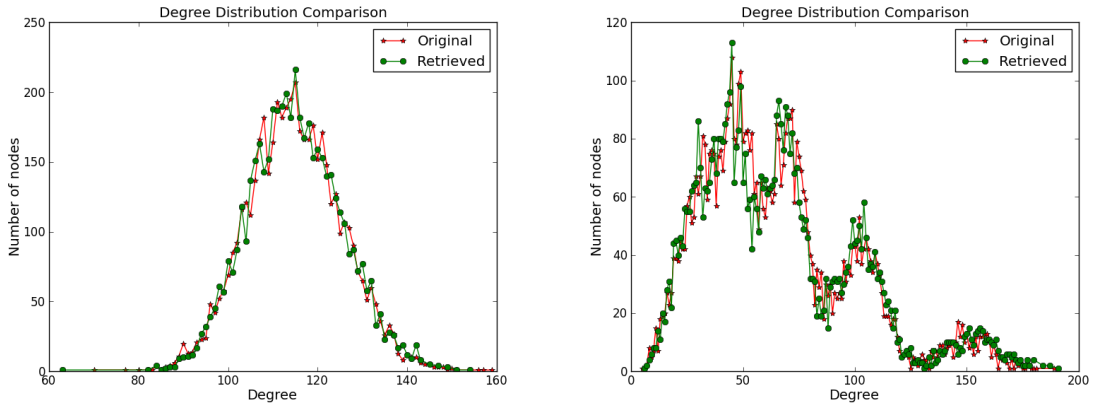


Figure 6: On the left, degree distribution of graphs generated according to the original Erdős-Rényi measure (red) given in Table 1 and the retrieved measure (green). On the right, degree distribution of graphs generated according to the original measure (red) described in Table 2 and the retrieved measure (green). The retrieved measure was found by the method of moments algorithm from Section 4. The original and retrieved measures produce almost identical distributions.

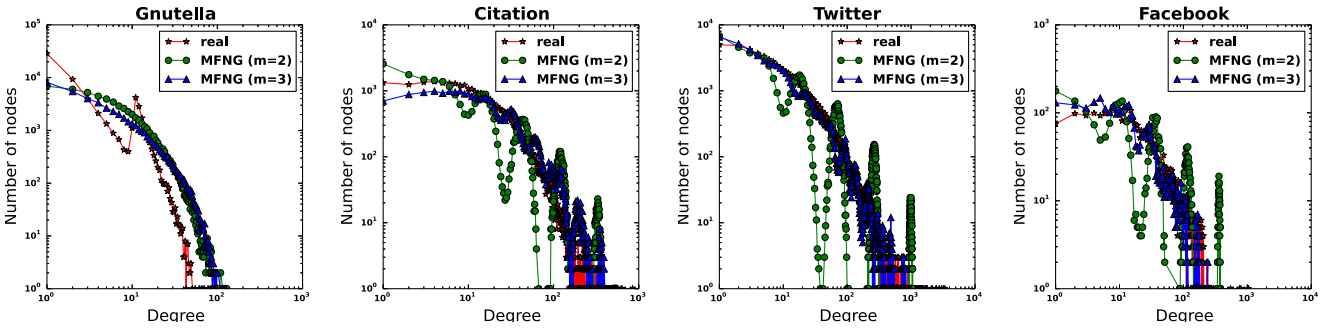


Figure 7: Degree distributions for the original graphs and MFNG graphs with $m = 2, 3$ for several networks. The degree distributions of the MFNG graphs are similar to those of the original network, even though we only fit d-star and t-clique moments. In the Twitter, Citation, and Facebook graphs, the MFNG fit with $m = 2$ results in oscillating degree distributions. In Section 6.3, we show how to add noise to dampen the oscillations. The graph samples were generated with the fast sampling algorithm in Section 5.

the method of moments can match both the number of triangles and the number of wedges in expectation. This does not make any guarantees about the *ratio* of these random variables, but the synthetic experiments (Section 6.1) demonstrated that their variances are not too large. Therefore, the expectation of the ratio is near the ratio of the expectations, and we approximately match the global clustering coefficient.

Figure 7 shows the degree distributions for the original networks and a sample from the corresponding MFNG, using the fast sampling algorithm. We see that, even though we only fit feature moments, the global degree distribution is similar to the real network. However, the MFNG degree distributions experience oscillations, especially in the case when $m = 2$. This is a well-known issue in SKG [16], and we address this issue in Section 6.3. Finally, note that we only plot the degree distribution for a single MFNG sample. The reason is that the samples tend to have quite similar degree distributions. This lack of variance has been observed for SKG [10], and addressing this issue for MFNG is an area of future work.

Table 4 shows the diameter, effective diameter, and average node eccentricity for the original networks and sampled MFNG networks. Effective diameter is the 90-th percentile of the linearly interpolated distribution of shortest path lengths [5]. The values between the original network and the MFNG samples are similar.

Network	Diameter	Eff. Diameter	Avg. Eccentricity
	original / MFNG ($m = 2$) / MFNG ($m = 3$)		
Gnutella	11 / 13 / 12	6.73 / 5.66 / 5.09	8.94 / 6.05 / 4.27
Citation	13 / 15 / 21	4.99 / 5.62 / 6.56	9.15 / 7.02 / 12.17
Twitter	7 / 11 / 21	4.52 / 4.47 / 6.47	5.92 / 5.57 / 10.94
Facebook	8 / 10 / 14	4.76 / 3.98 / 4.90	6.35 / 5.91 / 8.40

Table 4: Diameter, effective diameter and average node eccentricity for the original networks and sampled MFNG networks. For MFNG, each value is the median of five samples from the method described in Section 5. For diameter and effective diameter, 20% of the nodes were used to approximate the property.

6.3 Noisy MFNG

Figure 7 shows that the graphs generated with MFNG experience oscillations in the degree distribution. The oscillations for the degree distribution are a well-known issue in SKG [16]. Seshadhri *et*

Network method	m	features for fitting	$\|V\|$	$\|E\|$	S_2	S_3	S_4	C_3	C_4
Gnutella	–	–	62,586	147,892	1.57e+06	8.17e+06	4.38e+07	2.02e+03	1.6e+01
MFNG MoM	2	all	–	1.13	1.00	0.97	1.00	1.00	1.00
MFNG MoM	3	all	–	1.00	0.97	1.00	1.00	1.00	1.00
MFNG MoM	2	$\|E\|, S_2, C_3$	–	1.00	1.00	1.10	1.15	1.00	0.05
MFNG MoM	3	$\|E\|, S_2, C_3$	–	1.00	1.00	1.21	1.54	1.00	0.26
SKG MoM	–	$\|E\|, S_2, S_3, C_3$	–	1.14	1.00	1.00	18.34	0.30	< 0.69
KronFit	–	–	–	0.54	0.30	0.23	3.67	0.06	< 0.01
Citation	–	–	34,546	420,921	2.63e+07	1.34e+09	1.04e+10	1.28e+06	2.57e+06
MFNG MoM	2	all	–	0.79	1.02	1.00	0.61	1.00	1.00
MFNG MoM	3	all	–	1.00	1.03	1.00	1.00	1.00	1.00
MFNG MoM	2	$\|E\|, S_2, C_3$	–	0.99	1.00	0.77	0.42	1.00	4.65
MFNG MoM	3	$\|E\|, S_2, C_3$	–	1.00	1.00	0.85	0.60	1.00	1.08
SKG MoM	–	$\|E\|, S_2, S_3, C_3$	–	1.00	0.89	1.00	11.60	0.02	< 0.01
KronFit	–	–	–	0.53	0.21	0.09	0.57	< 0.01	< 0.01
Twitter	–	–	81,306	1,342,310	2.30e+08	6.35e+10	2.99e+13	1.31e+07	1.05e+08
MFNG MoM	2	all	–	1.00	1.59	1.00	0.33	1.00	1.00
MFNG MoM	3	all	–	1.00	1.16	1.00	1.00	0.89	1.00
MFNG MoM	2	$\|E\|, S_2, C_3$	–	1.00	1.00	0.44	0.12	1.00	2.83
MFNG MoM	3	$\|E\|, S_2, C_3$	–	1.00	1.00	0.44	0.11	1.00	2.71
SKG MoM	–	$\|E\|, S_2, S_3, C_3$	–	1.00	1.05	1.00	0.01	0.03	< 0.01
KronFit	–	–	–	0.69	0.30	0.10	< 0.01	< 0.01	< 0.01
Facebook	–	–	4,039	88,234	9.31e+06	7.27e+08	9.71e+10	1.61e+06	3.00e+07
MFNG MoM	2	all	–	0.96	1.19	1.00	0.42	1.00	1.00
MFNG MoM	3	all	–	1.00	1.06	1.00	0.69	0.90	1.00
MFNG MoM	2	$\|E\|, S_2, C_3$	–	0.90	1.00	0.80	0.34	1.00	1.88
MFNG MoM	3	$\|E\|, S_2, C_3$	–	1.00	1.00	0.75	0.33	1.00	1.13
SKG MoM	–	$\|E\|, S_2, S_3, C_3$	–	1.00	1.03	1.00	0.19	0.08	0.03
KronFit	–	–	–	0.49	0.20	0.07	0.04	0.01	< 0.01

Table 3: Results of method of moments (MoM) fit to MFNG for several graphs. Each column gives the ratio of the expected feature count to the true feature count. S_d is the number of d-stars in the graph, and C_t is the number of t-cliques in the graph. A value of 1.00 means that the moment is an exact fit to two decimal places. In all cases, MFNG is able to fit many of the feature counts exactly in expectation. For MFNG, we fit all feature moments listed and fitting just the number of edges, wedges, and triangles. The SKG MoM and KronFit are included for comparison. For these methods, S_4 and C_4 were estimated by taking the mean from 10 sample graphs (closed-form moment formulas are not available for these feature counts). Our MFNG MoM outperforms both KronFit and SKG MoM in fitting feature moments.

Algorithm 2 Noisy MFNG ($m = 2$)

Input: 2×2 probability matrix $\mathbf{P}$, lengths vector ℓ, number of recursive levels k, noise level b
Output: noisy MFNG matrix G
for $i = 1$ **to** k **do**
  Sample $\mu_i \sim \text{Uniform}[-b, b]$.
  $\mathbf{P}^{(i)} = \begin{pmatrix} \mathbf{p}_{11} - \frac{2\mu_i \mathbf{p}_{11}}{\mathbf{p}_{11}+p_{22}} & \mathbf{p}_{12} + \mu_i \\ \mathbf{p}_{21} + \mu_i & \mathbf{p}_{22} - \frac{2\mu_i \mathbf{p}_{22}}{\mathbf{p}_{11}+p_{22}} \end{pmatrix}$
  $\mathbf{P}^{(i)} = \min(\max(\mathbf{P}^{(i)}, 0), 1)$ entry-wise
  Sample $H_i \sim \mathcal{W}_1(\mathbf{P}^{(i)}, \mathbf{l})$
end for
$G := \cap_{i=1}^{k} H_i$

al. present a "Noisy SKG" model that perturbs the initiator matrix at each recursive level, which dampens the oscillations. Inspired by their work, we present a similar "Noisy MFNG" in this section.

We first note that Figure 7 shows that using $m = 3$ results in less severe oscillations in the degree distribution. The intuition behind this is that more categories get mixed at each recursive level, producing a larger variety of edge probabilities. For $m = 2$, we propose a Noisy MFNG model, which is described in Algorithm 2. The idea is to perturb the probability matrix slightly at each level. In the context of Lemma 1, this means that the noisy MFNG graph is the intersection of several graphs generated from slightly different probability matrices. The fast generation method still works in this case—a different matrix at each level determines the categories instead of one single matrix. The probability perturbations are analogous to those performed by Seshadhri *et al*. We test Noisy MFNG on the citation and Twitter networks, and the results are in Figure 8. The graphs are sampled using the fast sampling algorithm. We see that increasing the noise significantly dampens the degree distribution. However, the far end of the tail still experiences some oscillations.

7. DISCUSSION

We have shown that the multifractal graph paradigm is well suited to model and capture the properties of real-world networks by building on the work of Palla *et al*. [14] and incorporating several ideas from SKG. The foundation of our theoretical work is Theorem 2, which has opened the door to quick evaluation of the expected value of a number of important counts of subgraphs, such as d-stars and t-cliques. Combined with standard optimization routines, we are

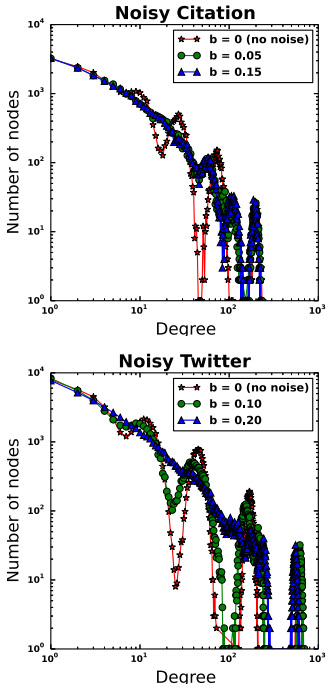


Figure 8: Degree distributions for fitting the citation and Twitter networks to Noisy MFNG with varying degrees of noise. We see that adding noise dampens the oscillations in the degree distributions. At the far end of the tail, it is still difficult to control the degree distribution. The graph samples were generated with the fast sampling algorithm in Section 5.

able to fit large graphs fast and accurately. Our method of moments algorithm identifies synthetically generated MFNG and also produces close fits for real-world networks. It is quite amazing how fitting a few 'local' properties leads to a generator that fits the overall structure of graphs well.

This would not be too useful if we were not able to also generate multifractal graphs of the same scale. For this, we presented a fast heuristic approximation algorithm that generates such graphs in $\mathcal{O}(|E| \log |V|)$ complexity, rather than the naive $\mathcal{O}(|V|^2)$ algorithm. Since many real-world networks are sparse, this is a significant improvement.

7.1 Future work

Future work includes the development of approximation formulas for the moments of global properties like graph diameter and a more tailored approach in the optimization routines for the fitting. A pressing issue is the theory behind the fast generation method. While the generation tends to produce similar graphs to the naive generation in practice, we want to prove that the approximation is good. Furthermore, it is possible to improve the generation further by considering a parallel implementation. Lastly, it would be interesting to do a theoretical analysis of the oscillatory degree distribution, similar in spirit to [16].

8. ACKNOWLEDGEMENTS

We thank David Gleich and Victor Minden for helpful discussions. Austin R. Benson is supported by an Office of Technology Licensing Stanford Graduate Fellowship. Carlos Riquelme is supported by a DARPA grant research assistantship under the supervision of Prof. Ramesh Johari. Sven Schmit is supported by a Prins Bernhard Cultuurfonds Fellowship.

9. REFERENCES

[1] L. Akoglu and C. Faloutsos. RTG: a recursive realistic graph generator using random typing. *Data Mining and Knowledge Discovery*, 19(2):194–209, 2009.

[2] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A recursive model for graph mining. In *SDM*, volume 4, pages 442–446. SIAM, 2004.

[3] D. F. Gleich and A. B. Owen. Moment-based estimation of stochastic Kronecker graph parameters. *Internet Mathematics*, 8(3):232–256, 2012.

[4] M. Kim and J. Leskovec. Multiplicative attribute graph model of real-world networks. In *Algorithms and Models for the Web-Graph*, pages 62–73. Springer, 2010.

[5] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *The Journal of Machine Learning Research*, 11:985–1042, 2010.

[6] J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using Kronecker multiplication. In *Proceedings of the 24th international conference on Machine learning*, pages 497–504. ACM, 2007.

[7] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM, 2005.

[8] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007.

[9] J. McAuley and J. Leskovec. Learning to discover social circles in ego networks. In *Advances in Neural Information Processing Systems 25*, pages 548–556, 2012.

[10] S. Moreno, S. Kirshner, J. Neville, and S. Vishwanathan. Tied Kronecker product graph models to capture variance in network populations. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pages 1137–1144, Sept 2010.

[11] S. Moreno and J. Neville. Network hypothesis testing using mixed Kronecker product graph models. In *ICDM*, pages 1163–1168, 2013.

[12] S. I. Moreno, J. Neville, and S. Kirshner. Learning mixed Kronecker product graph models with simulated method of moments. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1052–1060. ACM, 2013.

[13] R. C. Murphy, K. B. Wheeler, B. W. Barrett, and J. A. Ang. Introducing the graph 500. *Cray User's Group (CUG)*, 2010.

[14] G. Palla, L. Lovász, and T. Vicsek. Multifractal network generator. *Proceedings of the National Academy of Sciences*, 107(17):7640–7645, 2010.

[15] C. Seshadhri, T. G. Kolda, and A. Pinar. Community structure and scale-free collections of Erdős-Rényi graphs. *Physical Review E*, 85(5):056109, 2012.

[16] C. Seshadhri, A. Pinar, and T. G. Kolda. An in-depth analysis of stochastic Kronecker graphs. *Journal of the ACM (JACM)*, 60(2):13, 2013.