A Scalable Method for Exact Sampling from Kronecker Family Models

Sebastian Moreno
Purdue University
West Lafayette, Indiana
smorenoa@cs.purdue.edu

Joseph J. Pfeiffer III
Purdue University
West Lafayette, Indiana
jpfeiffer@cs.purdue.edu

Jennifer Neville
Purdue University
West Lafayette, Indiana
neville@cs.purdue.edu

Sergey Kirshner*
Unaffiliated
sergey.kirshner@gmail.com

***Abstract*—The recent interest in modeling complex networks has fueled the development of generative graph models, such as Kronecker Product Graph Model (KPGM) and mixed KPGM (mKPGM). The Kronecker family of models are appealing because of their elegant fractal structure, as well as their ability to capture important network characteristics such as degree, diameter, and (in the case of mKPGM) clustering and population variance. In addition, scalable sampling algorithms for KPGMs made the analysis of large-scale, sparse networks feasible for the first time. In this work, we show that the scalable sampling methods, in contrast to prior belief, do not in fact sample from the underlying KPGM distribution and often result in sampling graphs that are very unlikely. To address this issue, we develop a new representation that exploits the structure of Kronecker models and facilitates the development of novel *grouped* sampling methods that are provably *correct*. In this paper, we outline efficient algorithms to sample from mKPGMs and KPGMs based on these ideas. Notably, our mKPGM algorithm is the first available scalable sampling method for this model and our KPGM algorithm is both faster and more accurate than previous scalable methods. We conduct both theoretical analysis and empirical evaluation to demonstrate the strengths of our algorithms and show that we can sample a network with 75 million edges in 87 seconds on a single processor.**

I. Introduction

Due to the recent interest in modeling complex systems as networks, considerable research has focused on developing generative graph models to understand the underlying properties of these systems (see e.g., [1], [2], [3], [4]). Here the aim has been to develop graph models that can match common characteristics of real world networks (e.g., skewed degree distributions, local clustering). Since networks generated from the same process exhibit similar (but variable) characteristics, recent work has focused on modeling statistical *distributions* of network structure. Researchers have developed statistical methods to model the network distribution, along with algorithms to learn model parameters given an observed network. These models are used for a wide variety of tasks—for example, by learning a model of "normal" network behavior, we can use the model to test for network anomalies [5].

Most of the statistical models of graphs define a $N_v \times N_v$ probability matrix $\mathcal{P}$, where N_v is the number of nodes and $\mathcal{P}_{ij} = \pi_{ij}$ represents the probability of an edge between nodes i and j. By sampling from $\mathcal{P}$, models can generate networks from the underlying distribution. One of the first proposed models is the Erdös-Rényi graph model [6]. This model defines $\mathcal{P}_{ij} = p \ \ \forall \ i, j \in \{1, \cdots, N_v\}$, meaning every edge in the network has equal probability. Recent models (e.g., [7], [8], [9], [10]) consider different probabilities among edges. For example, the Chung-Lu model defines the probability of an edge to be proportional to the degree of the incident nodes [7]. The Kronecker Product Graph Model (KPGM) [8] and mixed-KPGM [9] are fractal models parameterized by a small "seed" matrix, which use repeated Kronecker multiplications to produce the probabilities in $\mathcal{P}$.

Many of the recent statistical models successfully reproduce important network characteristics in their samples. However, straightforward algorithms to generate networks, which sample edges independently using a Bernoulli distribution for every pair of nodes, have a time complexity that is quadratic in the number of nodes (i.e., $O(N_v^2)$). We refer to these pairwise sampling algorithms, which clearly cannot scale to large networks with millions of nodes and edges, as *Independent Probability* methods. To more efficiently model sparse real-world networks (e.g., where $N_e << N_v^2$), researchers have developed scalable *Edge-by-edge* sampling algorithms, which only consider where to place a set of N_e sampled edges through the network. For example, KPGMs [11] and MAGMs [12] have associated *Edge-by-edge* sampling algorithms that can generate a network in time $O(\log(N_v) \cdot N_e)$.

Notably, the availability of scalable sampling algorithms for KPGMs made analysis of large-scale, sparse networks feasible for the first time [11]. However, in contrast to prior expectations—scalable sampling algorithms (using edge-based sampling) do not correctly draw from the underlying model distribution and this often produces network samples that are very unlikely. Specifically, we prove that *Edge-by-edge* KPGM sampling algorithms model a different space of graphs, and they do not sample from the correct distribution even when we constrain the space of graphs to be equivalent.

To address these issues, we develop provably correct sampling methods for both KPGMs and mKPGMs, based on a key insight which uses *Grouped Probabilities*. The methods center on a novel representation for the Kronecker family of models, which allows us to exploit their fractal structure to group together pairs of nodes that have the same edge probability (e.g., i, j, u, v s.t. $\mathcal{P}_{ij} = \mathcal{P}_{uv}$). The edges are sampled for each

*Sergey Kirshner contributed while employed at Purdue University.

group independently, by first sampling a Binomial to determine the number of edges for the group, and subsequently placing them over the pairs in the group. This process generates networks from the true underlying distribution, and we develop efficient implementations using well-known Normal/Poisson approximations for sampling from Binomial distributions.

We conduct both theoretical analysis and empirical evaluation to demonstrate the strengths of our algorithms. In particular, we demonstrate the inaccuracies of Edge-by-edge algorithms as well as the advantages of our proposed algorithms. Notably, our *Grouped Probability* algorithms sample networks with over 8 million vertices and 75 million edges on a single processor in 87 seconds.

II. Background

We review the details of Kronecker family models (KPGM [8] and mKPGM [9]) below. In general, Kronecker models define a probability matrix $\mathcal{P}$ for graphs through a fractal structure. A constant sized "seed" matrix (e.g., 2x2) parameterizes the model, while repeated Kronecker multiplications of the parameters define the full $\mathcal{P}$.

A. Kronecker Product Graph Model

Model: KPGM models a distribution of networks via a set of N_v^2 independent edge probabilities. Although the edge probabilities are independent, the parameter values are recursively constructed based on Kronecker multiplications, and thus exhibit self-similarity. Specifically, KPGM utilizes a small, constant size $b \times b$ seed matrix $\mathcal{P}_1 = \Theta$, where each cell value represents a probability parameter. Typically $b = 2$ or 3, e.g., $\mathcal{P}_1 = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix}$.

To define a probability matrix $\mathcal{P}$ to model distributions of large scale networks, KPGM takes the Kronecker product of $\mathcal{P}_1$ with itself $K-1$ times. Given the small number of seed parameters and the fractal structure of the model, multiple probabilities are repeated in $\mathcal{P}$. Specifically, a network with b^K nodes is modeled with $\mathcal{P}_K = \mathcal{P}_1^{[K]} = \mathcal{P}_1^{[K-1]} \otimes \mathcal{P}_1 = \underbrace{\mathcal{P}_1 \otimes \ldots \otimes \mathcal{P}_1}_{K-1 \text{ times}}$.

Sampling: Let $G = (\mathbf{V}, \mathbf{E})$ be a graph with $\mathbf{V} = \{1, \ldots, N_v\}$, where $N_v = b^K$ and $K \in \mathbb{N}$, and $\mathbf{E} = \{E_{11}, E_{12}, \ldots, E_{N_v N_v}\}$, where E_{ij} is a Bernoulli random variable and $N_e = \sum_{i=1}^{N_v} \sum_{j=1}^{N_v} E_{ij}$. KPGM samples a graph G by performing independent Bernoulli trials for each pair (u, v) with probability $\mathcal{P}_K(u, v) = \pi_{uv}$. If the trial for (u, v) results in a success, KPGM places an edge (u, v) into $\mathbf{E}$ ($E_{uv} = 1 \Rightarrow E_{uv} \in \mathbf{E}$).

Complexity: Since $\mathcal{P}_K$ comprises a set of independent edge probabilities, to sample a network from the model it is sufficient to sample an edge for every pair of nodes: $E_{uv} \sim Bernoulli(\pi_{uv})$. Thus a straightforward sampling implementation will have time complexity $O(N_v^2)$, which is not scalable for large networks.

Space of graphs: We will refer to the space of graphs (i.e., all possible graphs) modeled by a specific KPGM as $\mathbf{G}_o^{\mathcal{K}}$, where o stands for the *original* model and $\mathcal{K}$ stands for KPGM. Formally, $\mathbf{G}_o^{\mathcal{K}} = \{G = (\mathbf{V}, \mathbf{E})$ such that $(|\mathbf{V}| = N_v = b^K)$ and $(0 \leq |\mathbf{E}| = N_e \leq N_v^2)$ and $(\nexists\ E_{uv}, E_{ij} \in \mathbf{E} : u = i$ and $v = j)$ and $(\forall\ E_{uv} \in \mathbf{E}, 1 \leq u, v \leq N_v)\}$. KPGM defines a probability distribution over this space of graphs, denoted $P_o^{\mathcal{K}}$. Using these definitions, we can see that the KPGM generation process samples a network G from $P_o^{\mathcal{K}}$, where $P_o^{\mathcal{K}}(G) : \mathbf{G}_o^{\mathcal{K}} \rightarrow [0, 1]$.

B. mixed Kronecker Product Graph Model

Model: mKPGM models a distribution of networks via a set of dependent edge probabilities [9]. Similar to KPGMs, the mKPGM parameters values are recursively constructed based on Kronecker multiplications of a small $b \times b$ matrix $\mathcal{P}_1$, where each cell value represents a probability parameter. However, in addition to the Θ and K parameters from KPGM, mKPGM incorporates an additional parameter $\ell \in [1, 2, \cdots, K]$, which ties parameters to create edge dependencies.

In particular, mKPGM models a network with b^K nodes with $\underbrace{R(\cdots R(R(R(\mathcal{P}_\ell) \otimes \Theta) \otimes \Theta) \otimes \cdots \otimes \Theta)}_{K-\ell-1 \text{ times}} \otimes \Theta$, where $\mathcal{P}_\ell$ is a KPGM probability matrix and the function $R(\cdot)$ refers to a realization (sampling) of the probability matrix prior to the Kronecker multiplication. These realizations tie the edges and produces dependencies among them. The value of ℓ specifies the level of tying and affects the variability of the generated networks (low ℓ values imply higher variability).

We note that KPGM is a special case of mKPGM (where $\ell = K$) and mKPGM preserves the same marginal probabilities of edges as a KPGM with equal Θ and K parameters, but the mKPGM edge probabilities are no longer independent.

Sampling: To generate networks with $N_v = b^K$ nodes, mKPGM initially uses a KPGM (parameterized by Θ and ℓ) to sample an intermediate graph G_ℓ. Next, mKPGM computes a Kronecker product between G_ℓ and Θ to produce a new probability matrix $\mathcal{P}_{\ell+1} = G_\ell \otimes \Theta$. Then, mKPGM samples $G_{\ell+1}$ from $\mathcal{P}_{\ell+1}$ and uses this for the next Kronecker product. The process of sampling a graph before computing subsequent Kronecker products produces dependencies among the sampled edges. The algorithm repeats this process $K - \ell - 1$ times to generate the final network G_K.

Complexity: A straightforward implementation of mKPGM sampling has three steps which affect its time complexity: the generation of the network $G_\ell = (\mathbf{V}_\ell, \mathbf{E}_\ell)$ using KPGM ($O(N_\ell^2)$ where $N_\ell = |\mathbf{V}_\ell|$); the number of Bernoulli trials used to generate the network $G_{\ell+k}$ ($O(b^2 \cdot |\mathbf{E}_{\ell+k-1}|)$); and the complexity of the Kronecker multiplications ($O(N_v^2)$ for the final Kronecker multiplication). This produces a total complexity of $O(N_v^2)$, the same as above for the KPGM.

Space of graphs: As with KPGM, the network generation process samples from a specific distribution defined by the model. We use $\mathcal{M}$ to refer to the mKPGM, and the generation process samples $G \sim P_o^{\mathcal{M}}(G)$, where $P_o^{\mathcal{M}}(G)$ defines a distribution over the space of graphs $\mathbf{G}_o^{\mathcal{M}}$. (Note that $\mathbf{G}_o^{\mathcal{M}} \equiv \mathbf{G}_o^{\mathcal{K}}$.)

III. Group Probability Sampling for KPGM

This section and the next outline new Kronecker model representations and develop our new *Group Probability* (GP) sampling algorithms for KPGM and mKPGM, respectively. The new algorithms correctly sample networks from the underlying probability distribution defined by the original Kronecker models and our implementations generate networks in time proportional to the number of edges.

A. Representation

KPGM defines the final probability matrix $\mathcal{P}_K$ through $K-1$ Kronecker multiplications of the parameter matrix Θ with itself. Importantly, due to the commutative property of multiplication, a single probability q can appear in many places in $\mathcal{P}_K$ (e.g., $q = \pi_{ij} = \pi_{kl}$ where $i, j \neq k, l$). For example, $\pi_{i_1 j_1} = \theta_{11}\theta_{12}\theta_{11}$ has the same probability as $\pi_{i_2 j_2} = \theta_{11}\theta_{11}\theta_{12}$ and $\pi_{i_3 j_3} = \theta_{12}\theta_{11}\theta_{11}$, but their positions in $\mathcal{P}_K$ are different. Thus, rather than sampling a Bernoulli for each $\pi_{i.j.}$, we can sample the total number of edges for each unique probability value using a binomial distribution. Then, we determine the positions to place the sampled number of edges from among the set of ij pairs with the associated probability value.

Before describing the implementation of our GP sampling algorithm, we create a new representation for the probability of the edges. Given b, K, and Θ, the probability of an edge in the original model ($p_o^{\mathcal{K}}(E_{uv}) = \pi_{uv}$) is equal to the multiplication of K different θ_{ij}. Let Γ_{uv} be a matrix of size $b \times b$ where each element γ_{ij} represents the number of times θ_{ij} occurs in the calculation of π_{uv}. Then $p_o^{\mathcal{K}}(E_{uv}) = \pi_{uv} = \theta_{11}^{\gamma_{11}}\theta_{12}^{\gamma_{12}} \cdots \theta_{bb}^{\gamma_{bb}}$ where each γ_{ij} is an integer in the range $[0, 1, \cdots, K]$ and $\sum_{i=1}^{b}\sum_{j=1}^{b}\gamma_{ij} = K$.

This new representation makes it easy to group together cells in the matrix $\mathcal{P}_K$ that have the same probability value and reduce the N_v^2 probabilities in $\mathcal{P}_K$ to a smaller set of unique probabilities. Let $\mathbf{U}$ be the set of unique probability values in $\mathcal{P}_K$, then $|\mathbf{U}|$ is the number of possible combinations of integer values of γ_{ij} subject to the constraint $\sum_{i=1}^{b}\sum_{j=1}^{b}\gamma_{ij} = K$. This corresponds to a k-combination with repetitions problem [13], where we have to pick K elements with replacement from the set $\{\theta_{11}, \theta_{12}, \cdots, \theta_{bb}\}$, obtaining

$$|\mathbf{U}| = \binom{b^2 + K - 1}{K} \tag{1}$$

Let $\pi'_k = \theta_{11}^{\gamma_{11_k}}\theta_{12}^{\gamma_{12_k}} \cdots \theta_{bb}^{\gamma_{bb_k}}$ be the k^{th} unique probability in $\mathbf{U}$, where γ_{ij_k} is γ_{ij} in Γ_k. Then, the number of times T_k of a particular probability π'_k repeating in $\mathcal{P}_K$ (i.e. the ij pairs such that $\pi_{ij} = \pi'_k$), is equal to all the possible permutations of the different elements in Γ_k :

$$T_k = \frac{K!}{\gamma_{11_k}!\gamma_{12_k}! \cdots \gamma_{bb_k}!} \tag{2}$$

For example, given Θ, $b = 2$ and $K = 3$, then $\mathcal{P}_3$ has total of 64 cells. From these cells, $|\mathbf{U}| = \binom{2^2+3-1}{3} = 20$ of them correspond to unique probability values. Consider an example unique probability $\pi_k = \theta_{11}\theta_{11}\theta_{12}$ from $\mathcal{P}_3$, then $\pi'_k = \theta_{11}^2\theta_{12}^1\theta_{21}^0\theta_{22}^0$, $\Gamma_k = \begin{bmatrix} 2 & 1 \\ 0 & 0 \end{bmatrix}$, and $T_k = \frac{3!}{2!1!0!0!} = 3$.

Algorithm 1 Group Probability Sampling for KPGM

Require: Θ, K, b
1: $\mathbf{V} = \{1, \cdots, b^K\}, \mathbf{E} = \{\}$
2: Construct $\mathbf{U}$, the set of unique probability values π'_k
3: **for** $k = 1; k++; k \leq |\mathbf{U}|$ **do**
4: Obtain π'_k the k-th unique probability of the set $\mathbf{U}$
5: Let $\Gamma_k = [\gamma_{11_k}, \gamma_{12_k}, \cdots, \gamma_{bb_k}]$ s.t. $\pi'_k = \theta_{11}^{\gamma_{11_k}}\theta_{12}^{\gamma_{12_k}} \cdots \theta_{bb}^{\gamma_{bb_k}}$
6: Calculate $T_k = \frac{K!}{\gamma_{11_k}!\gamma_{12_k}! \cdots \gamma_{bb_k}!}$
7: Draw $x_k \sim Bin(T_k, \pi'_k)$
8: countEdge=0
9: Let $\Lambda_i = [i_1, i_2, \cdots, i_K]$ and $\Lambda_j = [j_1, j_2, \cdots, j_K]$ be θ indexes s.t. $\prod_{l=1}^{K}\theta_{\Lambda_i(l)\Lambda_j(l)} = \pi'_k$.
10: **while** $countEdge < x_k$ **do**
11: Let σ be a random permutation of the vector $[1, 2, \cdots, K]$
12: $u, v = \sum_{l=1}^{K}(\Lambda_{i,j}(\sigma(l)) - 1) \cdot b^{l-1} + 1$
13: **if** $E_{uv} \notin \mathbf{E}$ **then**
14: countEdge++
15: $\mathbf{E} = \mathbf{E} + \{E_{uv}\}$
16: Return $G = (\mathbf{V}, \mathbf{E})$

B. Algorithm

We can use this new representation to sample a network in three steps. First, calculate $\mathbf{U}$. Second, for each unique probability $\pi'_k \in \mathbf{U}$, sample the number of edges x_k using a Binomial distribution ($P(X_k = x_k) \sim Bin(n, p)$ where $n = T_k$ and $p = \pi'_k$). Third, place the x_k sampled edges uniformly at random among the cells with probability π'_k.

Algorithm 1 describes the pseudocode of the GP sampling process. Line 2 constructs $\mathbf{U}$ (first step). Line 7 samples x_k (second step). Before sampling x_k, lines 4 to 6 determine π'_k, Γ_k, and T_k respectively. Lines 9 to 15 place the edges (third step). Line 9 determines the indexes of the θs utilized to calculate π'_k. Line 11 determines a random permutation of $\sigma = [1, 2, \cdots, K]$, which is used to calculate the indexes u and v in line 12. Lines 13 to 15 add E_{uv} to $\mathbf{E}$ if it has not already been sampled. Line 10 to 15 repeat the loop until x_k edges are placed in the group.

C. Complexity

The time complexity of the algorithm is as follows. Construction of the set of unique probabilities costs $O(K \cdot |\mathbf{U}|)$. Line 4 costs $O(1)$ and lines 5 and 6 have a complexity of $O(K)$. For line 7, naive sampling from a Binomial distribution would be $O(T_k)$ and result in an overall time complexity of $O(N_v^2)$. However, applying well-known Normal/Poisson approximations for sampling from Binomial distributions, we reduce the time complexity of line 7 to $O(1)$. In lines 11 and 12, obtaining the vectors $\Lambda_.$ is $O(K)$. The random permutation and the calculation of the indexes are also $O(K)$. Finally, the edge rejection process has a constant complexity $O(1)$.

Using the above analysis, we proceed to calculate the total complexity of the algorithm. An iteration of lines 10 to 15 has a complexity of $O(3K \cdot (x_k + x'_k))$ where x'_k corresponds to the number of rejected edges (when E_{uv} is generated but is

already in $\mathbf{E}$). In the worst case, when $\pi'_k = 0.5$, $E[x'_k] = x_k$ and the complexity is $O(K \cdot x_k)$. The outside loop has a complexity of $O(1{+}2K{+}1{+}K{\cdot}x_k){=}O(K{\cdot}x_k)$. Incorporating the summation over $|\mathbf{U}|$, and adding the complexity of line 2, we obtain a total complexity of $O\left(K{\cdot}|\mathbf{U}| + K\sum_{i=1}^{|\mathbf{U}|} x_k\right)$. It is easy to prove by induction that $|\mathbf{U}| \leq N_v$ for large K (i.e., $K \geq 7, 10$ for $b = 2, 3$, respectively). To understand the behavior of $|\mathbf{U}|$ and N_v with respect to large K, we empirically demonstrate that $|\mathbf{U}| \leq N_v$ for $b = 2, 3$ in the left plot of Figure 1. Finally, considering that $\sum_{i=1}^{|\mathbf{U}|} x_k = N_e$, then the final complexity is $O\left(K{\cdot}N_v + K{\cdot}N_e\right) = \tilde{O}(N_e)$.

D. Analysis

The GP sampling algorithm is also a sampling mechanism from a specific probability distribution. Let $\mathbf{G}^{\mathcal{K}}_{gp}$ be the space of graphs for the GP sampling process (*gp* refers to *group probability* algorithm), defined by $\{G = (\mathbf{V}, \mathbf{E})$ such that $(N_v = b^K)$ and $(0 \leq N_e \leq N_v^2)$ and $(\nexists\ E_{uv}, E_{ij} \in \mathbf{E} : u = i \wedge v = j)$ and $(\forall\ E_{uv} \in \mathbf{E}, 1 \leq u, v \leq N_v)\}$. Note that $\mathbf{G}^{\mathcal{K}}_{gp} \equiv \mathbf{G}^{\mathcal{K}}_{o}$. Then, the generation of a network is sampling G from $P^{\mathcal{K}}_{gp}(G)$, where $P^{\mathcal{K}}_{gp}(G) : \mathbf{G}^{\mathcal{K}}_{gp} \rightarrow [0, 1]$.

With these definitions, the next theorems prove that our GP algorithm samples networks from the original KPGM probability distribution. All proofs are provided in the appendix.

Theorem 1. *Given a valid KPGM with probability $p^{\mathcal{K}}_{o}(\cdot)$ and the GP sampling algorithm from Alg. 1, with probability $p^{\mathcal{K}}_{gp}(\cdot)$, then $\forall\, u,v\ \ p^{\mathcal{K}}_{o}(E_{uv}) = \pi_{uv} = \pi'_{uv} = p^{\mathcal{K}}_{gp}(E_{uv})$.*

Theorem 2. *Given a valid KPGM with probability $p^{\mathcal{K}}_{o}(\cdot)$ and the GP sampling algorithm from Alg. 1, with probability $p^{\mathcal{K}}_{gp}(\cdot)$, then $\forall\, G\ \ p^{\mathcal{K}}_{o}(G){=}p^{\mathcal{K}}_{gp}(G)$, thus $P^{\mathcal{K}}_{o}(G){=}P^{\mathcal{K}}_{gp}(G)$.*

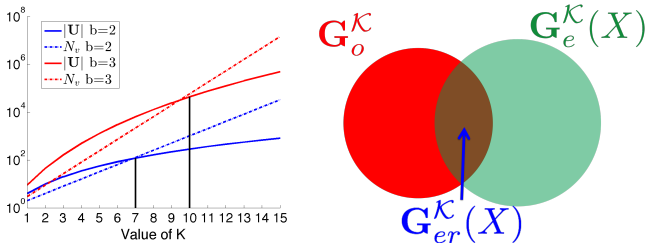


Fig. 1. Left: $|\mathbf{U}|$ and $N_v = b^K$ for $b = 2, 3$ with respect to K. Right: Venn diagram of the space of graphs for $X > 1$.

IV. Group Probability Sampling for mKPGM

Our new edge representation can also be used for mKPGMs. As mentioned previously, to date there is no scalable generation algorithm for mKPGMs due to dependencies among the Kronecker multiplications. But by generating G_ℓ using Algorithm 1, and then sampling each layer based on the new representation, we can generate a mKPGM network in $\tilde{O}(N_e)$.

A. Representation

Given Θ, K, and ℓ, the *original* mKPGM generation algorithm samples a network as follows: first, it generates $G_\ell = (\mathbf{V}_\ell, \mathbf{E}_\ell)$ with $|E_\ell| = N_{e_\ell}$ using KPGM. Then, it generates each layer from $G_{\ell+k}$ to G_K by calculating $\mathcal{P}_{\ell+k} = G_{\ell+k-1} \otimes \Theta$ and sampling from it. $\mathcal{P}_{\ell+k}$ is the Kronecker multiplication between a binary matrix (the adjacency matrix for $G_{\ell+k-1}$) and Θ. Consequently, each $\mathcal{P}_{\ell+k}$ has $b^2 = |\Theta|$ unique positive probability values ($\mathbf{U} = \{\theta_{11}, \theta_{12}, \cdots, \theta_{bb}\}$). Moreover, each unique positive probability value occurs exactly $N_{e_{\ell+k-1}}$ times in the matrix.

Algorithm 2 Group Probability Sampling for mKPGM

Require: Θ, K, b, ℓ
1: $\mathbf{V}_K = \{1, \cdots, b^K\}$
2: Generate G_ℓ using algorithm 1 with parameters Θ, ℓ, b
3: Construct $\mathbf{U} = \{\theta_{11}, \theta_{12}, \cdots, \theta_{bb}\}$
4: **for** $k = \ell + 1; k++; k \leq K$ **do**
5: $\quad \mathbf{E}_k = \{\}$
6: $\quad$ Let $\Lambda_q = [q_1, q_2, \cdots, q_{N_{e_{k-1}}}]$ and
$\quad\quad \Lambda_r = [r_1, r_2, \cdots, r_{N_{e_{k-1}}}]$ be indexes of edges in G_{k-1}
7: $\quad$ **for** $i = 1; i++; i \leq b$ **do**
8: $\quad\quad$ **for** $j = 1; j++; j \leq b$ **do**
9: $\quad\quad\quad$ Obtain θ_{ij} {a unique probability value in $\mathbf{U}$}
10: $\quad\quad\quad$ Calculate $T_{ij} = N_{e_{k-1}} = |\mathbf{E}_{k-1}|$
11: $\quad\quad\quad$ Draw $x_{ij_k} \sim Bin(T_{ij}, \theta_{ij})$
12: $\quad\quad\quad$ Let σ be a random permutation of vector $[1, 2, \cdots, N_{e_{k-1}}]$
13: $\quad\quad\quad$ **for** $m = 1; m++; m \leq x_{ij_k}$ **do**
14: $\quad\quad\quad\quad$ $u, v = (\Lambda_{q,r}(\sigma(m)) - 1) \cdot b + i$
15: $\quad\quad\quad\quad$ $\mathbf{E}_k = \mathbf{E}_k + \{E_{uv}\}$
16: Return $G_K = (\mathbf{V}_K, \mathbf{E}_K)$

B. Algorithm

Now, we can sample a network as follows: sample G_ℓ using GP sampling (Alg. 1). Then, sample each layer $G_{\ell+k}$ in three steps: first, determine $\mathbf{U}{=}\{\theta_{11}, \theta_{12}, \cdots, \theta_{bb}\}$, the set of unique probabilities in $\mathcal{P}_{\ell+k}$. Second, for each $\theta_{ij} \in \mathbf{U}$ sample the number of edges $x_{ij_k} \sim Bin(N_{e_{\ell+k-1}}, \theta_{ij})$. Third, place the x_{ij_k} sampled edges among the cells with probability θ_{ij}. Last, return the final layer G_K as the generated network.

In Algorithm 2, we show the pseudocode of our GP sampling algorithm. The algorithm generates the initial G_ℓ using Algorithm 1 in line 2, and constructs $\mathbf{U}$ in line 3. Lines 4 to 15 iteratively generates the remaining layers $\ell + 1$ to K. Line 6 obtains the indexes of edges from $G_{\ell+k-1}$. Lines 7 and 8 iterate over the set $\mathbf{U}$. Before sampling x_{ij_k} using the Binomial distribution in line 11, lines 9 and 10 determine θ_{ij} and T_{ij}. To avoid collisions, the x_{ij_k} edges are located based on the edge positions of $G_{\ell+k-1}$. To do this, in line 12 we randomly permute the vector $\sigma = [1, 2, \cdots, N_{e_{k-1}}]$. Finally, lines 14 to 15 calculate the new indexes using the first x_{ij_k} edges of $G_{\ell+k-1}$ based on the random permutation of σ.

C. Complexity

The overall complexity for the GP sampling for mKPGM is $\tilde{O}(N_e)$. The complexity of line 2 is $O(\ell(N_{v_\ell}{+}N_{e_\ell}))$. In line 6, the index vectors can be constructed in $O(N_{e_{k-1}})$. While lines 9, 10, 14 and 15 are O(1) (Θ_{ij}, T_{ij}, calculation of indexes, and adding the edge to the network), lines 11 are 12 have complexity $O(N_{e_{k-1}})$ (sampling from a Binomial distribution, and the random permutation).

Using the above analysis, we can calculate the complexity of the entire algorithm. The sampling and placement of x_{ij_k} edges (lines 7 to 15) is $O(2b^2 \cdot N_{e_{k-1}} + \sum_{i=1}^{b}\sum_{j=1}^{b} x_{ij_k}) = O(b^2 \cdot N_{e_{k-1}} + N_{e_k})$. In expectation $N_{e_k} = S^k$, where $S = \sum_{ij}\theta_{ij}$. Thus, the generation of all layers (lines 4 to 15) have a expected complexity of $O(\sum_{k=\ell+1}^{K}(b^2 \cdot N_{e_{k-1}} + N_{e_k} + 2N_{e_{k-1}})) = O(\sum_{k=\ell+1}^{K} b^2 \cdot S^{k-1}) = O(b^2 \frac{S^K - S^\ell}{1-S}) = O(b^2 \cdot N_e)$. Finally, the total time of the algorithm is $O(\ell(N_{v_\ell} + N_{e_\ell}) + b^2 + b^2 \cdot N_e) = \tilde{O}(N_e)$. This complexity reduction is a result of avoiding the Kronecker multiplication in the generation process.

D. Analysis

Again, the generation of a network can be considered as a sampling process from a specific distribution: $G \sim P_{gp}^{\mathcal{M}}(G)$, where $\mathcal{M}$ stands for mKPGM, and $P_{gp}^{\mathcal{M}}(G)$ is defined over the space of graphs $\mathbf{G}_{gp}^{\mathcal{M}}$.

With these definitions, we next state the theorems and corollary that prove that our GP sampling algorithm samples from the same probability distribution as the original mKPGM.

Theorem 3. *Given a valid mKPGM with probability $p_o^{\mathcal{M}}(\cdot)$, the GP sampling algorithm from Alg. 2 (with probability $p_{gp}^{\mathcal{M}}(\cdot)$), and a graph G_{k-1}, if $\ell < k \leq K$ then $\forall\ u, v$ $p_o^{\mathcal{M}}(E_{uv} \in G_k | G_{k-1}) = p_{gp}^{\mathcal{M}}(E_{uv} \in G_k | G_{k-1})$.*

Theorem 4. *Given a valid mKPGM with probability $p_o^{\mathcal{M}}(\cdot)$, the GP sampling algorithm from Alg. 2 (with probability $p_{gp}^{\mathcal{M}}(\cdot)$), and a graph G_{k-1}, if $\ell < k \leq K$ then $p_o^{\mathcal{M}}(G_k | G_{k-1}) = p_{gp}^{\mathcal{M}}(G_k | G_{k-1})$.*

Corollary 1. *Given a valid mKPGM with probability $p_o^{\mathcal{M}}(\cdot)$, the GP sampling algorithm from Alg. 2 (with probability $p_{gp}^{\mathcal{M}}(\cdot)$), then $\forall\, G\ \ p_o^{\mathcal{M}}(G) = p_{gp}^{\mathcal{M}}(G)$, thus $P_o^{\mathcal{M}}(G) = P_{gp}^{\mathcal{M}}(G)$.*

V. Deficiencies of edge by edge sampling algorithms

This section proves that edge by edge generation algorithms for KPGM do not sample networks from the original KPGM probability distribution. First, we demonstrate that the algorithms generate a different space of graphs, leading to a different probability distribution compared to the original algorithm. However, even if we reduce the space of graphs to that of the original KPGM, the distributions are still different. Again all proofs are in the appendix.

A. Edge by Edge Algorithm

Due to the excessive time to generate a network using the original KPGM algorithm, an edge-by-edge generation algorithm [8], which was later referred to as R-MAT [14], was developed. This generation algorithm draws from the normalized parameter matrix $\left(\frac{\Theta}{\sum_{ij}\Theta}\right)$, K times, to place an edge (u, v) into $\mathbf{E}$, where $u, v = \left(\sum_{l=1}^{K}(u_l, v_l - 1)b^{l-1})\right) + 1$, $u_l, v_l \in \{1, \cdots, b\}$, and u_l, v_l are the resulting positions of the l^{th} draw with respect to the matrix Θ. To generate a network, this process is repeated N_e times (where N_e is defined by the user), resulting in a complexity of $O(K \cdot N_e) = \tilde{O}(N_e)$. However, this new generation process can sample two or more edges in the same position, producing multigraphs.

Under this algorithm, the probability of an edge and a graph change with respect to the original sampling process. Let $p_e^{\mathcal{K}}(E_{uv})$ be the probability of an edge (e refers to edge-by-edge), then $p_e^{\mathcal{K}}(E_{uv}) = \frac{\theta_{u_1 v_1}}{S}\frac{\theta_{u_2 v_2}}{S}\cdots\frac{\theta_{u_K v_K}}{S} = \pi_{uv}/S^K$ where $S = \sum_{ij}\Theta$, and $\theta_{u_k v_k}$ is the sampled parameter at iteration k. This corresponds to sample an edge from a multinomial distribution over $\mathcal{P}_K$. Thus, the algorithm draws a specific sequence of edges from a multinomial distribution to generate a network. Let X be a random variable corresponding to the number of edges in a network, then $p_e^{\mathcal{K}}(G|X = N_e)$ is the probability of a graph G with $X = N_e$ edges, and is equal to the summation over all possible sequences of edges that can lead to G. Let $\mathcal{E} = E_{u_1 v_1}, E_{u_2 v_2}, \cdots, E_{u_{N_e} v_{N_e}}$ be a sequence of $X = N_e$ edges. The probability of this sequence is:

$$p_e^{\mathcal{K}}(\mathcal{E}) = p_e^{\mathcal{K}}(E_{u_1 v_1}) p_e^{\mathcal{K}}(E_{u_2 v_2}) \cdots p_e^{\mathcal{K}}(E_{u_{N_e} v_{N_e}}) = \frac{1}{S^{KN_e}} \prod_{E_{uv} \in \mathcal{E}} \pi_{uv}$$

Now, a network G with $X = N_e$ edges is the result of any of the $N_e!$ sequence of edges:

$$\begin{aligned} p_e^{\mathcal{K}}(G|X = N_e) &= p_e^{\mathcal{K}}(\mathcal{E}_1) + p_e^{\mathcal{K}}(\mathcal{E}_2) + \cdots + p_e^{\mathcal{K}}(\mathcal{E}_{N_e!}) \\ &= \frac{1}{S^{KN_e}} \prod_{E_{uv} \in \mathbf{E}} \pi_{uv} + \cdots + \frac{1}{S^{KN_e}} \prod_{E_{uv} \in \mathbf{E}} \pi_{uv} = \frac{N_e!}{S^{KN_e}} \prod_{E_{uv} \in \mathbf{E}} \pi_{uv} \end{aligned} \quad (3)$$

B. Edge by Edge with Rejection Algorithm

A simple solution to avoid multigraphs is to include a rejection process for edges that were previously sampled (edge-by-edge with rejection). By incorporating this step, the probability of a new edge E_{uv} ($p_{er}^{\mathcal{K}}(E_{uv})$, where er refers to edge-by-edge with rejection) depends on the previously sampled edges. Given a set of sampled edges $(E_{u_1 v_1}, E_{u_2 v_2}, \cdots, E_{u_i v_i})$, $p_{er}^{\mathcal{K}}(E_{uv})$ is equivalent to sampling from a multinomial distribution over all remaining ij pairs:

$$p_{er}^{\mathcal{K}}(E_{uv} | E_{u_1 v_1}, E_{u_2 v_2}, \cdots, E_{u_i v_i}) = \frac{\pi_{uv}}{S^K - \sum_{j=1}^{i} \pi_{u_j v_j}}$$

$p_{er}^{\mathcal{K}}(G|X)$ is not as simple as $p_e^{\mathcal{K}}(G|X)$ due to the dependencies among the edges. This makes the probability of a sequence of edges different based on the sampling order. Given $X = N_e$, $b^K = N_v$, and a specific sequence of edges $\mathcal{E} = E_{u_1 v_1}, E_{u_2 v_2}, \cdots, E_{u_{N_e} v_{N_e}}$, the probability of this sequence using edge-by-edge with rejection algorithm is:

$$\begin{aligned} p_{er}^{\mathcal{K}}(\mathcal{E}) &= p_{er}^{\mathcal{K}}(E_{u_1 v_1}) p_{er}^{\mathcal{K}}(E_{u_2 v_2} | E_{u_1 v_1}) \cdots p_{er}^{\mathcal{K}}(E_{u_{N_e} v_{N_e}} | \mathbf{E} - E_{u_{N_e} v_{N_e}}) \\ &= \frac{\pi_{u_1 v_1}}{S^K} \frac{\pi_{u_2 v_2}}{S^K - \pi_{u_1 v_1}} \cdots \frac{\pi_{u_{N_e} v_{N_e}}}{S^K - \sum_{i=1}^{N_e - 1} \pi_{u_i v_i}} = \prod_{i=1}^{N_e} \frac{\pi_{u_i v_i}}{S^K - \sum_{j=1}^{i-1} \pi_{u_j v_j}} \end{aligned}$$

Under this sampling algorithm, a network G with $X = N_e$ edges can be the result of any $N_e!$ permutation of the edges. Thus, the probability of a graph G is:

$$\begin{aligned} p_{er}^{\mathcal{K}}(G|X = N_e) &= p_{er}^{\mathcal{K}}(\mathcal{E}_1) + p_{er}^{\mathcal{K}}(\mathcal{E}_2) + \cdots + p_{er}^{\mathcal{K}}(\mathcal{E}_{N_e!}) \\ &= \sum_{k=1}^{N_e!} \prod_{i_k=1}^{N_e} \frac{\pi_{u_{i_k} v_{i_k}}}{S^K - \sum_{j_k=1}^{i_k - 1} \pi_{u_{j_k} v_{j_k}}} \end{aligned} \quad (4)$$

C. Analysis

Edge-by-edge algorithms are also a sampling mechanism from a probability distribution. The space of graphs and probability distribution for the *edge-by-edge* generation algorithm is $\mathbf{G}_e^{\mathcal{K}}(X) = \{G = (\mathbf{V}, \mathbf{E})$ such that $(|\mathbf{V}| = b^K)$ and $(|\mathbf{E}| = X)$ and $(\forall\ E_{uv} \in \mathbf{E}, 1 \leq u, v \leq |\mathbf{V}|)\}$, and $P_e^{\mathcal{K}}(G|X) : \mathbf{G}_e^{\mathcal{K}}(X) \rightarrow [0, 1]$. In the case of the *edge-by-edge with rejection algorithm*, $\mathbf{G}_{er}^{\mathcal{K}}(X) = \{G = (\mathbf{V}, \mathbf{E})$ such that $(G \in \mathbf{G}_e^{\mathcal{K}}(X))$ and $(\nexists\ E_{uv}, E_{ij} \in \mathbf{E} : u = i$ and $v = j)\}$, and $P_{er}^{\mathcal{K}}(G|X) : \mathbf{G}_{er}^{\mathcal{K}}(X) \rightarrow [0, 1]$.

By definition networks generated by edge-by-edge algorithms have exactly X edges. Thus, these generation algorithms can not generate all networks from $\mathbf{G}_o^{\mathcal{K}}$ (the space of graphs of the original KPGM sampling process). As a consequence, networks sampled using edge-by-edge algorithms are not sampled from $P_o^{\mathcal{K}}(G)$.

Theorem 5. *Given a valid KPGM with probability $p_o^{\mathcal{M}}(\cdot)$ and the edge-by-edge sampling algorithms with probability $p_e^{\mathcal{K}}(\cdot)$ and $p_{er}^{\mathcal{K}}(\cdot)$, then $P_{er}^{\mathcal{K}}(G|X) \neq P_o^{\mathcal{K}}(G) \neq P_e^{\mathcal{K}}(G|X)$.*

This theorem proves that edge-by-edge algorithms differ from the original KPGM sampling process, because of the differences in the space of graphs. We illustrate this idea graphically in Figure 1 (right plot), where $\mathbf{G}_o^{\mathcal{K}}$ is different than $\mathbf{G}_e^{\mathcal{K}}(X)$, and $\mathbf{G}_{er}^{\mathcal{K}}(X)$ is a subset of $\mathbf{G}_o^{\mathcal{K}}$ and $\mathbf{G}_e^{\mathcal{K}}(X)$. $\mathbf{G}_e^{\mathcal{K}}(X)$ is the result of the edge-by-edge generation algorithm sampling multigraphs (which do not exist in $\mathbf{G}_o^{\mathcal{K}}$ and $\mathbf{G}_{er}^{\mathcal{K}}(X)$). Additionally, the original algorithm can generate networks with different X (even the empty graph belongs to $\mathbf{G}_o^{\mathcal{K}}$), unlike edge-by-edge algorithms.

Even if we reduce the space of graph for KPGM to the subset $\mathbf{G}_o^{\mathcal{K}}(X)$ (i.e., networks with X edges), $P_o^{\mathcal{K}}(G|X)$ (the probability distribution over $\mathbf{G}_o^{\mathcal{K}}(X)$) can be different than $P_e^{\mathcal{K}}(G|X)$ and $P_{er}^{\mathcal{K}}(G|X)$. First, $P_o^{\mathcal{K}}(G|X) \neq P_e^{\mathcal{K}}(G|X)$ because the possibility of multigraphs makes the space of graphs different (similar to theorem 5). Second, unless very stringent conditions are met we also have $P_o^{\mathcal{K}}(G|X) \neq P_{er}^{\mathcal{K}}(G|X)$, even though the spaces of graphs are the same: $\mathbf{G}_o^{\mathcal{K}}(X) \equiv \mathbf{G}_{er}^{\mathcal{K}}(X)$.

Let $p_o^{\mathcal{K}}(G|X = N_e)$ be the probability of a graph under the original KPGM with $X = N_e$, then $p_o^{\mathcal{K}}(G|X = N_e) = \frac{p_o^{\mathcal{K}}(G)}{Z_{N_e}}$ iff $G \in \mathbf{G}_o^{\mathcal{K}}(N_e)$, where $Z_{N_e} = \sum_{G \in \mathbf{G}_o^{\mathcal{K}}(N_e)} p_o^{\mathcal{K}}(G)$. Now, we can establish the conditions under which $p_o^{\mathcal{K}}(G_1|X) = p_{er}^{\mathcal{K}}(G_1|X)$ based on another graph G_2 in the next theorem.

Theorem 6. *Assume there is a graph $G_2 = (\mathbf{V}_2, \mathbf{E}_2)$ with $|\mathbf{E}_2| = X$ such that $p_{er}^{\mathcal{K}}(G_2|X) = p_o^{\mathcal{K}}(G_2|X) > 0$. Then let $G_1 = (\mathbf{V}_1, \mathbf{E}_1)$ be another graph such that $|\mathbf{V}_1| = |\mathbf{V}_2|$ and $|\mathbf{E}_1| = |\mathbf{E}_2|$, with $p_o^{\mathcal{K}}(G_1|X) > 0$ and $p_o^{\mathcal{K}}(G_1|X) \neq p_o^{\mathcal{K}}(G_2|X)$. In this case, $p_{er}^{\mathcal{K}}(G_1|X) = p_o^{\mathcal{K}}(G_1|X)$ only if:* $\frac{p_{er}^{\mathcal{K}}(G_1|X) \prod_{E_{uv} \in \mathbf{E}_1 \wedge E_{uv} \notin \mathbf{E}_2} \frac{1-\pi_{uv}}{\pi_{uv}}}{p_{er}^{\mathcal{K}}(G_2|X) \prod_{E_{ij} \in \mathbf{E}_2 \wedge E_{ij} \notin \mathbf{E}_1} \frac{1-\pi_{ij}}{\pi_{ij}}} = 1$

If this condition does not hold, there will be a graph G_1 such as $p_o^{\mathcal{K}}(G_1|X) \neq p_{er}^{\mathcal{K}}(G_1|X)$, thus $P_{er}^{\mathcal{K}}(G|X) \neq P_o^{\mathcal{K}}(G|X)$. Next, we define a corollary, where even for graphs with a single edge (i.e. $X = 1$) this condition does not hold.

Corollary 2. *Let $G_1 = (\mathbf{V}_1, \mathbf{E}_1)$ be a graph such that $E_1 = \{E_{11}\}$ (i.e., $|E_1| = 1$) where $p_o^{\mathcal{K}}(G_1) = \frac{p_o^{\mathcal{K}}(G_\emptyset)}{Z_X} \frac{\pi_{11}}{1-\pi_{11}} > 0$; and let $G_2 = (\mathbf{V}_2, \mathbf{E}_2)$ be a graph where $E_2 = \{E_2\}$ and $p_o^{\mathcal{K}}(G_2) = \frac{p_o^{\mathcal{K}}(G_\emptyset)}{Z_X} \frac{\pi_{22}}{1-\pi_{22}} > 0$ such that $|\mathbf{V}_1| = |\mathbf{V}_2|$, $X = |\mathbf{E}_1| = |\mathbf{E}_2|$, and $p_o^{\mathcal{K}}(G_1|X) \neq p_o^{\mathcal{K}}(G_2|X)$. Then $p_{er}^{\mathcal{K}}(G_1|X) \neq p_o^{\mathcal{K}}(G_1|X)$.*

A particular case where $P_{er}^{\mathcal{K}}(G|X) = P_o^{\mathcal{K}}(G|X)$ is when all the edges have the same probabilities ($\theta_{ij} = \theta$, $\forall i, j \in \{1, 2, \cdots, b\}$, i.e. Erdos Renyi model).

Finally, to compare $P_e^{\mathcal{K}}(G)$ and $P_{er}^{\mathcal{K}}(G)$ against $P_o^{\mathcal{K}}(G)$, we marginalize over the number of edges: $P_\cdot^{\mathcal{K}}(G) \sim \sum_X P_\cdot^{\mathcal{K}}(G|X) P_\cdot^{\mathcal{K}}(X)$. Unfortunately, the real distributions for $P_e^{\mathcal{K}}(X)$ and $P_{er}^{\mathcal{K}}(X)$ are unknown and difficult to estimate. However, we will show empirically that even using distributions for $P_e^{\mathcal{K}}(X)$ and $P_{er}^{\mathcal{K}}(X)$, as suggested by [11] and [12], the final marginalized distributions are different than that of the original model.

VI. Experimental results

We use three experiments to empirically validate our theoretical characterization of the GP sampling algorithms.[1] In particular, we show that our implemented algorithms: (1) sample from the original KPGM probability distribution, (2) generate networks that have the same characteristics as those sampled from the original Kronecker model, and (3) have a generation time $\tilde{O}(N_e)$.

A. Distribution over space of graphs

We compared the analytical and empirical cumulative probability distributions (CDFs) among all described methods for a small network of size $N_v = 4$. Even though N_v is small, the space of graphs $\mathbf{G}_o^{\mathcal{K}}$ has $|\mathbf{G}_o^{\mathcal{K}}| = 2^{N_v^2} = 65,536$ networks. The analytical distributions were previously defined in sections II-V. Here, we calculate empirical distributions based on 5,000,000 networks, using the following parameters $\Theta = [0.9\ 0.7;\ 0.5\ 0.1]$, $b = 2$, $K = 2$, and $\ell = 1$ for mKPGM. For the empirical distributions $P_\cdot^{\mathcal{K}}(G|X)$, we fixed $X = 5$ because around 25% of the 5,000,000 generated networks had this number of edges under the original KPGM algorithm.

To compare CDFs, we calculated the maximum absolute value between two CDFs (the Kolmogorov Smirnov distance, KS). We report the results in table I—values close to 0% imply similar CDFs. We also apply the KS hypothesis test [15] to determine if the differences between two CDFs are statistically significant. Hypothesis tests that are not rejected are in bold font in table I.

Figure 2(a) shows that the original KPGM and GP sampling algorithms for KPGM match the analytical KPGM distribution. This is also confirmed by the low KS distances (0.03% in both cases), and the fact that the null hypothesis is not rejected (bold font in table I). These results confirm that the empirical distribution of our GP algorithm matches the analytical distribution for KPGM ($P_{gp}^{\mathcal{K}}(G) = P_o^{\mathcal{K}}(G)$).

[1]The code to replicate these experiments is available at http://nld.cs.purdue.edu

To compare against edge-by-edge algorithms, we marginalize over X, sampling $P_e^{\mathcal{K}}(X)$ and $P_{er}^{\mathcal{K}}(X)$, as suggested by [11] and [12], from $N(S^K, S^K - S_2^K)$ where $S = \sum_{ij} \theta_{ij}$ and $S_2^K = \sum_{ij} \theta_{ij}^2$. As shown in Figure 2(a), $P_e^{\mathcal{K}}(G)$ and $P_{er}^{\mathcal{K}}(G)$ do not match the analytical distribution for $P_o^{\mathcal{K}}(G)$. The KS evaluation produces large distances of 16.10% and 62.27%, which results in rejection of the null hypothesis. Moreover, the CDF of $P_e^{\mathcal{K}}(G)$ does not sum up to 1 because $|\mathbf{G}_e^{\mathcal{K}}| >> |\mathbf{G}_o^{\mathcal{K}}|$ due to multigraphs. These results support our claim that edge-by-edge algorithms do not sample the networks from the original KPGM probability distribution.

Figure 2(b) shows the results when we compared the distributions using a specific number of edges ($P_{\cdot}^{\mathcal{K}}(G|X)$). The analytical and empirical distributions for edge-by-edge algorithms match, obtaining a low KS distance (0.06% and 0.13% respectively), but the null hypothesis is only rejected for $P_e^{\mathcal{K}}(G|X)$. Again the CDF of $P_e^{\mathcal{K}}(G|X)$ does not sum up to 1. These issues related to $P_e^{\mathcal{K}}(G|X)$ are the result of the multigraphs that were not considered. Aside from these results, the empirical CDFs for $P_o^{\mathcal{K}}(G|X)$ and $P_{gp}^{\mathcal{K}}(G|X)$ overlap, with a KS distance of 0.09%, which does not result in rejection of the null hypothesis (i.e., the differences are not statistically significant; note this KS result is not included in the table, because both are empirical distributions). Moreover, these distributions are different than the analytical CDFs of $P_e^{\mathcal{K}}(G|X)$ and $P_{er}^{\mathcal{K}}(G|X)$, resulting in large KS distances and the rejection of the null hypothesis (table I).

Figure 2(c) shows the empirical CDFs for the original and GP sampling algorithms for mKPGM. Again the CDFs overlap, with a low KS distance of 0.08%, which does not result in rejection of the null hypothesis (KS results again are not listed in the table because both are empirical distributions). These findings confirm that the empirical distributions are similar ($P_o^{\mathcal{M}}(G) = P_{gp}^{\mathcal{M}}(G)$).

Figure 2(d) shows the average log ratio per edge under the KPGM likelihood for the different sampling methods. In this experiment, we used the Θ for the GRQC dataset (Table II), $K = 11 \Rightarrow N_v = 177,147$ and $E[N_e] = (\sum_{\Theta} \theta_{ij})^K = 1,398,967$. We did not compare against the original KPGM algorithm, because of the amount of time required to generate large networks with that approach. To calculate and compare the log ratios we rewrite the KPGM likelihood equation:

$$p_o^{\mathcal{K}}(G) = p_o^{\mathcal{K}}(G_\emptyset) \frac{\prod_{E_{uv} \in \mathbf{E}} (1 - \pi_{uv})}{\prod_{E_{uv} \notin \mathbf{E}} (1 - \pi_{uv})}$$

where $p_o^{\mathcal{K}}(G_\emptyset)$ is the probability of the empty graph. We drop this component because is the same for all algorithms. To avoid multigraphs and make a fair comparison, we eliminated duplicate edges and considered the average log ratio over the generated edges. The plot confirms that networks generated by the GP sampling algorithm have a higher average log ratio per edge than edge-by-edge generation methods. This is evidence that, even in large graphs, edge-by-edge algorithms generate networks that are less likely under the KPGM.

KS dist		Empirical			
		$P_o^{\mathcal{K}}(G)$	$P_{gp}^{\mathcal{K}}(G)$	$P_e^{\mathcal{K}}(G)$	$P_{er}^{\mathcal{K}}(G)$
Analytical	$P_o^{\mathcal{K}}(G)$	**0.03%**	**0.03%**	62.27%	16.10%
		$P_o^{\mathcal{K}}(G\|X)$	$P_{gp}^{\mathcal{K}}(G\|X)$	$P_e^{\mathcal{K}}(G\|X)$	$P_{er}^{\mathcal{K}}(G\|X)$
Analytical	$P_e^{\mathcal{K}}(G\|X)$	69.56%	69.56%	**0.06%**	69.47%
	$P_{er}^{\mathcal{K}}(G\|X)$	15.55%	15.54%	69.52%	0.13%

TABLE I
KS DISTANCES TO ANALYTICAL $P_o^{\mathcal{K}}(G)$, $P_e^{\mathcal{K}}(G|X)$ AND $P_{er}^{\mathcal{K}}(G|X)$.

Dataset	N_v	N_e	Θ
Email	6,503	14,756	[.09 .17 .48 .17 .97 .07 .48 .07 .81]
GRQC	5,242	28,980	[.99 .80 .02 .80 .03 .01 .02 .01 .95]

TABLE II
LEARNED PARAMETERS FOR DATASETS

B. Network characteristics

The second experiment compared the characteristics of the generated network for different Θ learned over two datasets [16]. We give the Θs and characteristics of the networks in Table II. The GRQC dataset is a single networks that we can use to see the differences among the generation algorithms for KPGM. The Email dataset is an illustrative example of a graph *population* that we can use to compare the generation algorithms for mKPGM ($\ell = 5$).

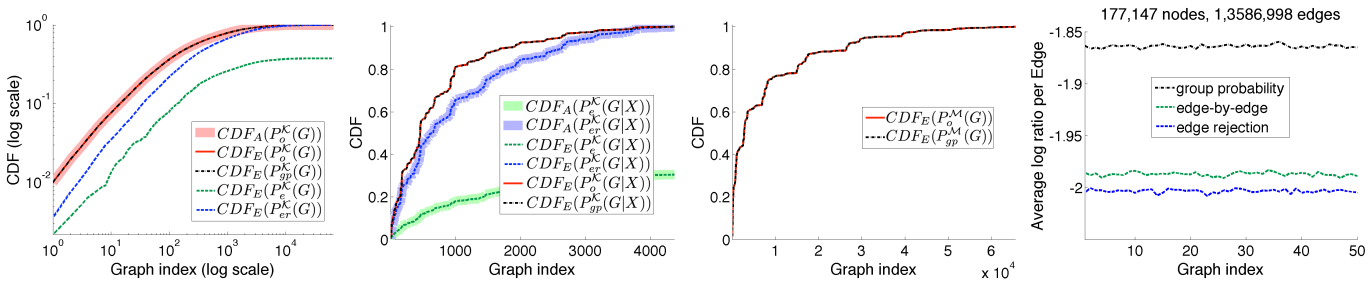
We generated 200 networks for each dataset, and compared their degree and clustering coefficient CDFs. The degree of a node d_i is the number of nodes in the graph connected to node i. The clustering coefficient of a node i is: $c_i = \frac{2|\delta_i|}{(d_i - 1)d_i}$, where δ_i is the number of triangles in which the node i participates.

Figure 3 confirms that GP sampling can replicate the characteristics of the networks generated by the original KPGM and mKPGM algorithms, replicating not only the median of the distribution but also their variability in the case of the mKPGM (given by the error bars of the right plots). In contrast, for KPGM, networks generated by edge-by-edge algorithms (with and without rejection) only match the degree distribution but not the clustering coefficient. Even though the goal of this work is to replicate the networks generated by the original sampling algorithm, we can observe that original and GP sampling methods are closer to the distributions of the real data, which confirms the importance of sampling accurately from the original distribution.

C. Running time

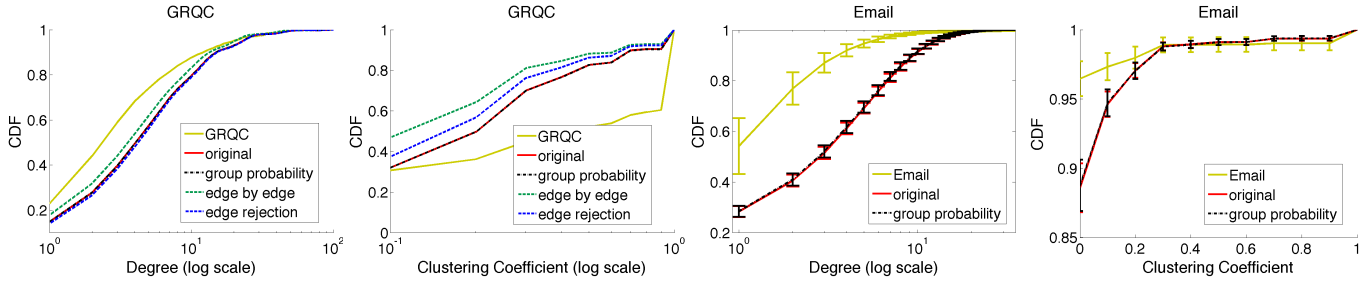
The last experiment compared the generation time in seconds among all generative algorithms. We ran this experiment on a Mac with processor 2.9 GHz Intel Core i7 and 8 GB 1600 MHz DDR3 of RAM memory, under OS X Version 10.9.2. The software utilized for the experiment was Matlab version 7.14.0.739 (R2012a). We implemented all the methods to reduce the runtime of the generation algorithm, even if we had to increase memory.

The results report the average time in seconds over 100 networks using $\Theta = [0.9\ 0.7; 0.5\ 0.1]$, $b = 2$, $K = \{5, \cdots, 20\}$, and $\ell = \lceil K/2 \rceil$ (for mKPGM). Unfortunately, we could not



(a) CDFs $P_{\cdot}^{\mathcal{K}}(G)$ (b) CDFs $P_{\cdot}^{\mathcal{K}}(G|X)$ (c) CDFs $P_{\cdot}^{\mathcal{M}}(G)$ (d) average log ratio per edge

Fig. 2. Cumulative distributions function over the space of graph $N_v = 4$ for different sampling methods. (a): Space of graph given by $N_v = 4$. (b): Space of graph restricted to $X = 5$. (c): Space of graphs for mKPGM. (d): Average ratio per edge for large networks.



(a) GRQC degree (b) GRQC clustering (c) Email degree (d) Email clustering

Fig. 3. Degree and Clustering Coefficient for KPGM and mKPGM generative algorithms on GRQC (left) and Email (right) datasets.

generate networks for some of the larger values of K for the original and edge-by-edge (with and without rejection) generation algorithms because of infeasible run times.

Left plot of Figure 4 corroborates the inefficiency of the original KPGM algorithm, which on average takes approximately 11 seconds to generate a network with $2^{14} = 16,384$ nodes. Edge-by-edge generation algorithms are a little faster than GP sampling when $K \leq 15$ and $K \leq 18$ (with and without rejection respectively). However, their times increase considerably for large K, because of memory issues. The empirical results confirm the linear time complexity of GP sampling with respect to the number of edges $(\tilde{O}(N_e))$, and it is the fastest algorithm for larger K. We were able to generate, in memory, a network with $2^{23} = 8,388,608$ nodes and approximately $75,114,133$ edges in 265 seconds.

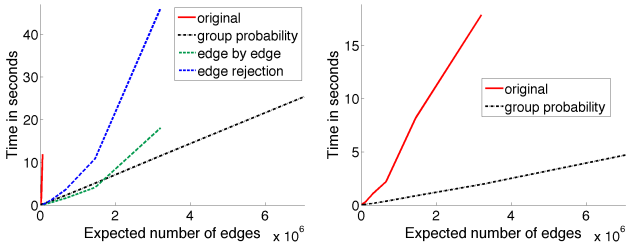


Fig. 4. Generation time in seconds against the expected number of edges, for KPGM (left) and mKPGM (right) generative algorithms

We give similar results for mKPGM (right plot, Figure 4). We can observe the $O(N_v^2)$ time complexity for the original mKPGM algorithm. In contrast, the GP sampling algorithm is faster than the original mKPGM and KPGM algorithms (for this particular ℓ), and the results confirm its linear time with respect to the number of edges $(\tilde{O}(N_e))$. We were able to generate, in memory, a network with $2^{23} = 8,388,608$ nodes and approximately $75,114,133$ edges in 87 seconds. Note that mKPGM generation is even faster than KPGM, because its algorithmic complexity depends on the addition of processes, rather than the multiplication used in KPGM.

VII. Conclusions

The main contribution of this paper is a new representation of Kronecker models, which facilitates the creation of algorithms for KPGM and mKPGM that correctly sample from the original probability distribution. Our implemented algorithms: (1) reproduce the probability distribution over the space of graphs intended by the original Kronecker family of models (KS distances less than 0.1%), (2) replicate the characteristics of the networks generated by the original algorithms, and (3) efficiently generate networks with time complexity $\tilde{O}(N_e)$. Notably, we can generate a network with $\sim$8 million nodes and $\sim$75 million edges in 87 seconds.

We also prove that previous edge-by-edge generation algorithms do not generate networks from the same space of graphs, nor do they replicate the probability distributions of the original KPGM (KS distances greater than 15.45%).

In the future, we will apply the GP sampling ideas to develop scalable sampling methods for other statistical network models that sample edges from a probability matrix. We will also parallelize the algorithm by generating the set of edges for each unique probability value separately.

APPENDIX

For all the following demonstrations, it is assumed that $K > 0$ and $0 \leq \theta_{ij} \leq 1,\ \forall\ i,j \in \{1,\cdots,b\}$

Proof Theorem 1: Let $\pi'_{uv} = \theta_{11}^{\gamma_{11}}\theta_{12}^{\gamma_{12}}\cdots\theta_{bb}^{\gamma_{bb}}$ and $T = \frac{K!}{\gamma_{11}!\gamma_{12}!\cdots\gamma_{bb}!}$ be the new representation of the probability of an edge E_{uv}, and the number of edges with unique probability π'_{uv} respectively, then

$$p_{gp}^{\mathcal{K}}(E_{uv}) = \sum_{k=0}^{T} P(E_{uv}|X=k)P(X=k) \tag{5}$$

where $X \sim Bin(T, \pi'_{uv})$ and $P(E_{uv}|X=k)$ is defined by

$$P(E_{uv}|X=k) = 1 - P(\overline{E_{uv}}|X=k) = 1 - \prod_{i=1}^{k}\left(1 - \frac{1}{T - \sum_{j=1}^{i-1} 1}\right)$$

$$= 1 - \prod_{i=1}^{k}\left(1 - \frac{1}{T-(i-1)}\right) = 1 - \prod_{i=1}^{k}\left(\frac{T-i}{T-i+1}\right) = 1 - \frac{T-k}{T} = \frac{k}{T}$$

Reemploying $P(E_{uv}|X=k)$ in equation 5

$$p_{gp}^{\mathcal{K}}(E_{uv}) = \sum_{k=0}^{T}\frac{k}{T}Bin(X=k;T,\pi'_{uv}) = \frac{1}{T}E[X] = \frac{1}{T}\pi'_{uv}T = p_o^{\mathcal{K}}(E_{uv})$$

■

Proof Theorem 2: While the sampling of edges for an unique probability value are dependent, the edges between different unique probability values are independent of each other. Let $\mathbf{E}_k$ be the set of edges with unique probability π_k and $|\mathbf{E}_k| = x_k$, then

$$p_{gp}^{\mathcal{K}}(G) = \prod_{k=1}^{|\mathbf{U}|} p_{gp}^{\mathcal{K}}(\mathbf{E}_k) = \prod_{k=1}^{|\mathbf{U}|}\sum_{i=1}^{T_k} p_{gp}^{\mathcal{K}}(\mathbf{E}_k|Y=i)P(Y=i)$$
$$= \prod_{k=1}^{|\mathbf{U}|} p_{gp}^{\mathcal{K}}(\mathbf{E}_k|Y=x_k)P(Y=x_k)$$

where $Y \sim Bin(\pi'_k, T_k)$, $p_{gp}^{\mathcal{K}}(\mathbf{E}_k|Y=i) = 0$ if $i \neq x_k$ and $p_{gp}^{\mathcal{K}}(\mathbf{E}_k|Y=x_k)$ is given by the probability over the $x_k!$ possible sequences of edges. Let $\mathcal{E}_{k_i}$ the k_i be the possible sequence of edges of the set $\mathbf{E}_k$. Edges previously sampled are rejected and resampled, and all edges have the same probability to be sampled, then $p(\mathcal{E}_{k_i}) = \frac{1}{T_k}\frac{1}{T_k-1}\cdots\frac{1}{T_k-x_k+1}$.

So $p_{gp}^{\mathcal{K}}(\mathbf{E}_k|Y=x_k) = p_{gp}^{\mathcal{K}}(\mathbf{E}_k|x_k)$ is given by

$$p_{gp}^{\mathcal{K}}(\mathbf{E}_k|x_k) = \sum_{i=1}^{x_k!} p(\mathcal{E}_{k_i}) = x_k!\frac{1}{T_k}\frac{1}{T_k-1}\cdots\frac{1}{T_k-x_k+1} = \frac{x_k!(T_k-x_k)!}{T_k!}$$

Joining $p_{gp}^{\mathcal{K}}(\mathbf{E}_k|Y=x_k) = p_{gp}^{\mathcal{K}}(\mathbf{E}_k|x_k)$ with $P(Y=x_k)$

$$p_{gp}^{\mathcal{K}}(\mathbf{E}_k|Y=x_k)P(Y=x_k) = \frac{x_k!(T_k-x_k)!}{T_k!}\binom{T_k}{x_k}\pi_k'^{x_k}(1-\pi'_k)^{T_k-x_k}$$
$$= \frac{1}{\binom{T_k}{x_k}}\binom{T_k}{x_k}\pi_k'^{x_k}(1-\pi'_k)^{T_k-x_k} = \pi_k'^{x_k}(1-\pi'_k)^{T_k-x_k}$$

Reemploying in $p_{gp}^{\mathcal{K}}(G)$

$$p_{gp}^{\mathcal{K}}(G) = \prod_{k=1}^{|\mathbf{U}|} p_{gp}^{\mathcal{K}}(\mathbf{E}_k) = \prod_{k=1}^{|\mathbf{U}|}\pi_k'^{x_k}(1-\pi'_k)^{T_k-x_k}$$
$$= \prod_{E_{uv}\in E} p_o^{\mathcal{K}}(E_{uv}) \prod_{E_{uv}\notin E}(1 - p_o^{\mathcal{K}}(E_{uv})) = p_o^{\mathcal{K}}(G)$$

Given that the space of graphs are the same ($\mathbf{G}_o^{\mathcal{K}} = \mathbf{G}_{gp}^{\mathcal{K}}$), therefore $P_o^{\mathcal{K}}(G) = P_{gp}^{\mathcal{K}}(G)$. ■

Proof Theorem 3: In mKPGM, let E_{uv} be an edge of the layer G_k, if $\ell < k \leq K$, then

$$p_o^{\mathcal{M}}(E_{uv} \in G_k|G_{k-1})$$
$$= p_o^{\mathcal{M}}(E_{uv} \in G_k|E_{F_{uv}^{[k]}} \in G_{i-1}) + p_o^{\mathcal{M}}(E_{uv} \in G_i|E_{F_{uv}^{[k]}} \notin G_{i-1})$$
$$= \theta_{F_{uv}^{[k]}} + 0 = \theta_{ij}$$

where $F_{uv}^{[k]} = i,j$ corresponds to the father/parent indexes of E_{uv} in layer k. Similarly, in the GP sampling process, let E_{uv} be an edge of the layer G_k, if $\ell < k \leq K$, then

$$p_{gp}^{\mathcal{M}}(E_{uv} \in G_k|G_{k-1})$$
$$= p_{gp}^{\mathcal{M}}(E_{uv} \in G_k|E_{F_{uv}^{[k]}} \in G_{k-1}) + p_{gp}^{\mathcal{M}}(E_{uv} \in G_k|E_{F_{uv}^{[k]}} \notin G_{k-1})$$
$$= p_{gp}^{\mathcal{M}}(E_{uv} \in G_k|E_{F_{uv}^{[k]}} \in G_{k-1}) = \sum_{x=0}^{N_{e_{k-1}}} \frac{x}{N_{e_{k-1}}} Bin(x; N_{e_{k-1}}, \theta_{ij}) \tag{6}$$
$$= \frac{1}{N_{e_{k-1}}}E[X] = \frac{\theta_{ij}N_{e_{k-1}}}{N_{e_{k-1}}} = \theta_{ij} = p_o^{\mathcal{M}}(E_{uv} \in G_k|G_{k-1})$$

where equality 6 is developed in the proof of theorem 1. ■

Proof Theorem 4: In the GP sampling process, let G_k be a layer, if $\ell < k \leq K$, then $p_{gp}^{\mathcal{M}}(G_k|G_{k-1}) = \prod_{i=1}^{b}\prod_{j=1}^{b} p_{gp}^{\mathcal{M}}(\mathbf{E}_{ij_k}|G_{k-1})$, where $p_{gp}^{\mathcal{M}}(\mathbf{E}_{ij_k}|G_{k-1})$ is the set of edges with unique probability θ_{ij} and $|\mathbf{E}_{ij_k}| = x_{ij_k}$. Rewriting $p_{gp}^{\mathcal{M}}(\mathbf{E}_{ij_k}|G_{k-1})$ we obtain

$$p_{gp}^{\mathcal{M}}(\mathbf{E}_{ij_k}|G_{k-1}) = p_{gp}^{\mathcal{M}}(\mathbf{E}_{ij_k}|Y = x_{ij_k}, G_{k-1})P(Y = x_{ij_k}|G_{k-1})$$

as explained in theorem 2, $Y \sim Bin(N_{e_{k-1}}, \theta_{ij})$, and $p_{gp}^{\mathcal{M}}(\mathbf{E}_{ij_k}|Y = x_{ij_k}, G_{k-1})$ is given by the probability over the $x_{ij_k}!$ possible sequences of edges. Similar to theorem 2, let $\mathcal{E}_{ij_{k_m}}$ be the k_m possible sequence of edges of the set E_{ij_k}, then $p(\mathcal{E}_{ij_{k_m}}) = \frac{1}{N_{e_{k-1}}}\frac{1}{N_{e_{k-1}}}\cdots\frac{1}{N_{e_{k-1}}-x_{ij_k}+1}$.
So $p_{gp}^{\mathcal{M}}(\mathbf{E}_{ij_{k_m}}|Y = x_{ij_k}, G_{k-1})$ is given by

$$p_{gp}^{\mathcal{M}}(\mathbf{E}_{ij_{k_m}}|Y = x_{ij_k}, G_{k-1}) = \sum_{i=1}^{x_{ij_k}!} p(\mathcal{E}_{ij_{k_m}})$$
$$= x_{ij_k}!\frac{1}{N_{e_{k-1}}}\frac{1}{N_{e_{k-1}}-1}\cdots\frac{1}{N_{e_{k-1}}-x_{ij_k}+1} = \frac{1}{\binom{N_{e_{k-1}}}{x_{ij_k}}}$$

Joining $p_{gp}^{\mathcal{M}}(\mathbf{E}_{ij_k}|Y = x_{ij_k})$ with $P(Y = x_{ij_k})$

$$p_{gp}^{\mathcal{M}}(\mathbf{E}_{ij_k}|Y=x_{ij_k})P(Y=x_{ij_k}) = \frac{\binom{N_{e_{k-1}}}{x_{ij_k}}}{\binom{N_{e_{k-1}}}{x_{ij_k}}}\theta_{ij}^{x_{ij_k}}(1-\theta_{ij})^{N_{e_{k-1}}-x_{ij_k}}$$
$$= \theta_{ij}^{x_{ij_k}}(1-\theta_{ij})^{N_{e_{k-1}}-x_{ij_k}}$$

Reemploying in $p_{gp}^{\mathcal{M}}(G_k|G_{k-1})$

$$p_{gp}^{\mathcal{M}}(G_k|G_{k-1}) = \prod_{i,j=1}^{b} p_{gp}^{\mathcal{M}}(\mathbf{E}_{ij_k}|G_{k-1}) = \prod_{i,j=1}^{b}\theta_{ij}^{x_{ij_k}}(1-\theta_{ij})^{N_{e_{k-1}}-x_{ij_k}}$$
$$= \prod_{E_{uv}\in E_k} p_o^{\mathcal{M}}(E_{uv}\in G_k|G_{k-1})\prod_{E_{uv}\notin E_k}(1-p_o^{\mathcal{M}}(E_{uv}\in G_k|G_{k-1})) = p_o^{\mathcal{M}}(G_k|G_{k-1})$$

■

Proof Corollary 1: In the original mKPGM sampling process, let G be any network belonging to $\mathbf{G}_o^{\mathcal{M}}$, and $\mathbf{G}_{K-1}$ be the set of all possible graph that can be generated of size b^{K-1}, then

$$p_o^{\mathcal{M}}(G_K)=\sum_{G_{i_1}\in\mathbf{G}_{K-1}} p_o^{\mathcal{M}}(G_K|G_{i_1})p_o^{\mathcal{M}}(G_{i_1})=\sum_{G_{i_1}\in\mathbf{G}_{K-1}} p_{gp}^{\mathcal{M}}(G_K|G_{i_1})p_o^{\mathcal{M}}(G_{i_1})$$

So, $p_o^{\mathcal{M}}(G_K)$ is rewritten as the summation of $p_o^{\mathcal{M}}(G_K|G_{i_1})p_o^{\mathcal{M}}(G_{i_1})$. Given that $p_o^{\mathcal{M}}(G_K|G_{i_1}) = p_{gp}^{\mathcal{M}}(G_K|G_{i_1})$ (theorem 4), then we have to demonstrate that $p_o^{\mathcal{M}}(G_{i_1}) = p_{gp}^{\mathcal{M}}(G_{i_1})$. Applying the same process multiple times

$$p_o^{\mathcal{M}}(G_K)=\sum_{G_{i_1}\in\mathbf{G}_{K-1}} p_{gp}^{\mathcal{M}}(G_K|G_{i_1})p_o^{\mathcal{M}}(G_{i_1})$$

$$=\sum_{G_{i_1}\in\mathbf{G}_{K-1}} p_{gp}^{\mathcal{M}}(G_K|G_{i_1})\left(\sum_{G_{i_2}\in\mathbf{G}_{K-2}} p_o^{\mathcal{M}}(G_{i_1}|G_{i_2})p_o^{\mathcal{M}}(G_{i_2})\right)$$

$$=\sum_{G_{i_1}\in\mathbf{G}_{K-1}} p_{gp}^{\mathcal{M}}(G_K|G_{i_1})\left(\cdots\left(\sum_{G_\ell\in\mathbf{G}_\ell} p_{gp}^{\mathcal{M}}(G_{\ell+1}|G_\ell)p_o^{\mathcal{M}}(G_\ell)\right)\right)$$

Considering that $p_o^{\mathcal{M}}(G_\ell)$ and $p_{gp}^{\mathcal{M}}(G_\ell)$ are generated by KPGM, then by Theorem 2 $p_o^{\mathcal{M}}(G_\ell) = p_{gp}^{\mathcal{M}}(G_\ell)$, we obtain

$$p_o^{\mathcal{M}}(G_K)=\sum_{G_{i_1}\in\mathbf{G}_{K-1}} p_{gp}^{\mathcal{M}}(G_K|G_{i_1})\left(\cdots\left(\sum_{G_\ell\in\mathbf{G}_\ell} p_{gp}^{\mathcal{M}}(G_{\ell+1}|G_\ell)p_o^{\mathcal{M}}(G_\ell)\right)\right)$$

$$=\sum_{G_{i_1}\in\mathbf{G}_{K-1}} p_{gp}^{\mathcal{M}}(G_K|G_{i_1})\left(\cdots\left(\sum_{G_\ell\in\mathbf{G}_\ell} p_{gp}^{\mathcal{M}}(G_{\ell+1}|G_\ell)p_{gp}^{\mathcal{M}}(G_\ell)\right)\right)=p_{gp}^{\mathcal{M}}(G_K)$$

■

Proof Theorem 5: $\forall\ G=(\mathbf{V},\mathbf{E})$ such that $G\in\mathbf{G}_o^{\mathcal{K}}$, thus $p_o^{\mathcal{K}}(G)>0$, because every edge has positive probability ($0<p_o^{\mathcal{K}}(E_{uv})<1\ \forall\ E_{uv}$).

Considering that $\exists\ G$ such that $\Big((G\in\mathbf{G}_o^{\mathcal{K}})$ and $(G\notin\mathbf{G}_e^{\mathcal{K}}(X))$ and $(G\notin\mathbf{G}_{er}^{\mathcal{K}}(X))\Big)$ then $\Big((p_o^{\mathcal{K}}(G)>0)$ and $(p_e^{\mathcal{K}}(G|X)=0)$ and $(p_e^{\mathcal{K}}(G|X)=0)\Big)$ therefore $P_{er}^{\mathcal{K}}(G|X)\neq P_o^{\mathcal{K}}(G)\neq P_e^{\mathcal{K}}(G|X)$. ■

Proof Theorem 6: Assume that $p_{er}^{\mathcal{K}}(G_1|X) = p_o^{\mathcal{K}}(G_1|X)$, rewriting $p_o^{\mathcal{K}}(G_1|X)$ based on $p_o^{\mathcal{K}}(G_2|X)$ using the empty graph $G_\emptyset=(\mathbf{V}_\emptyset,\mathbf{E}_\emptyset)$ ($|\mathbf{V}_\emptyset|=|\mathbf{V}_1|$ and $|\mathbf{E}_\emptyset|=0$), then

$$p_{er}^{\mathcal{K}}(G_1|X)=p_o^{\mathcal{K}}(G_1|X)\Rightarrow 1=\frac{p_{er}^{\mathcal{K}}(G_1|X)}{\frac{p_o^{\mathcal{K}}(G_\emptyset)}{Z_X}\prod_{E_{uv}\in\mathbf{E}_1}\frac{\pi_{uv}}{1-\pi_{uv}}}$$

$$\Rightarrow 1=\frac{p_{er}^{\mathcal{K}}(G_1|X)}{\frac{p_o^{\mathcal{K}}(G_\emptyset)}{Z_X}\prod_{\substack{E_{uv}\in\mathbf{E}_1\\ \wedge E_{uv}\notin\mathbf{E}_2}}\frac{\pi_{uv}}{1-\pi_{uv}}}\cdot\frac{1}{\prod_{\substack{E_{uv}\in\mathbf{E}_1\\ \wedge E_{uv}\notin\mathbf{E}_2}}\frac{\pi_{uv}}{1-\pi_{uv}}\prod_{\substack{E_{uv}\in\mathbf{E}_2\\ \wedge E_{uv}\notin\mathbf{E}_1}}\frac{\pi_{uv}(1-\pi_{uv})}{\pi_{uv}(1-\pi_{uv})}}$$

$$\Rightarrow 1=\frac{p_{er}^{\mathcal{K}}(G_1|X)}{\frac{p_o^{\mathcal{K}}(G_\emptyset)}{Z_X}\prod_{E_{uv}\in\mathbf{E}_2}\frac{\pi_{uv}}{1-\pi_{uv}}}\cdot\frac{1}{\left(\prod_{\substack{E_{uv}\in\mathbf{E}_1\\ \wedge E_{uv}\notin\mathbf{E}_2}}\frac{\pi_{uv}}{1-\pi_{uv}}\prod_{\substack{E_{uv}\in\mathbf{E}_2\\ \wedge E_{uv}\notin\mathbf{E}_1}}\frac{1-\pi_{uv}}{\pi_{uv}}\right)}$$

$$\Rightarrow 1=\frac{p_{er}^{\mathcal{K}}(G_1|X)}{p_o^{\mathcal{K}}(G_2|X)\prod_{E_{uv}\in\mathbf{E}_1\wedge E_{uv}\notin\mathbf{E}_2}\frac{\pi_{uv}}{1-\pi_{uv}}\prod_{E_{ij}\in\mathbf{E}_2\wedge E_{ij}\notin\mathbf{E}_1}\frac{1-\pi_{ij}}{\pi_{ij}}}$$

$$\Rightarrow 1=\frac{p_{er}^{\mathcal{K}}(G_1|X)\prod_{E_{uv}\in\mathbf{E}_1\wedge E_{uv}\notin\mathbf{E}_2}\frac{1-\pi_{uv}}{\pi_{uv}}}{p_{er}^{\mathcal{K}}(G_2|X)\prod_{E_{ij}\in\mathbf{E}_2\wedge E_{ij}\notin\mathbf{E}_1}\frac{1-\pi_{ij}}{\pi_{ij}}} \tag{7}$$

Thus, if equality 7 does not hold, then $p_{er}^{\mathcal{K}}(G_1|X)\neq p_o^{\mathcal{K}}(G_1|X)$. ■

Proof Corollary 2: Assume $p_{er}^{\mathcal{K}}(G_2|X)=p_o^{\mathcal{K}}(G_2|X)$, otherwise G_2 shows that $p_{er}^{\mathcal{K}}\neq p_o^{\mathcal{K}}$, and assume equality 7 holds. Then

$$\frac{p_{er}^{\mathcal{K}}(G_1|X)\prod_{E_{uv}\in\mathbf{E}_1\wedge E_{uv}\notin\mathbf{E}_2}\frac{1-\pi_{uv}}{\pi_{uv}}}{p_{er}^{\mathcal{K}}(G_2|X)\prod_{E_{ij}\in\mathbf{E}_2\wedge E_{ij}\notin\mathbf{E}_1}\frac{1-\pi_{ij}}{\pi_{ij}}}=1\Rightarrow\frac{p_{er}^{\mathcal{K}}(G_1|1)\frac{1-\pi_{11}}{\pi_{11}}}{p_{er}^{\mathcal{K}}(G_2|1)\frac{1-\pi_{22}}{\pi_{22}}}=1$$

$$\Rightarrow\frac{\frac{\pi_{11}}{S^K}\frac{1-\pi_{11}}{\pi_{11}}}{\frac{\pi_{22}}{S^K}\frac{1-\pi_{22}}{\pi_{22}}}=1\Rightarrow\frac{1-\pi_{11}}{1-\pi_{22}}=1\Rightarrow\pi_{11}=\pi_{22}$$

However, by definition $p_o^{\mathcal{K}}(G_1|X)\neq p_o^{\mathcal{K}}(G_2|X)$, which determines

$$p_o^{\mathcal{K}}(G_1|X)\neq p_o^{\mathcal{K}}(G_2|X)\Rightarrow\frac{p_o^{\mathcal{K}}(G_\emptyset)}{Z_X}\frac{\pi_{11}}{1-\pi_{11}}\neq\frac{p_o^{\mathcal{K}}(G_\emptyset)}{Z_X}\frac{\pi_{22}}{1-\pi_{22}}$$

$$\Rightarrow\frac{\pi_{11}}{1-\pi_{11}}\neq\frac{\pi_{22}}{1-\pi_{22}}\Rightarrow\pi_{11}\neq\pi_{22}$$

However $\pi_{11}\neq\pi_{22}$ and the equality does not hold. Thus by contradiction $p_{er}^{\mathcal{K}}(G_1|X)\neq p_o^{\mathcal{K}}(G_1|X)$, which shows that $P_{er}^{\mathcal{K}}(G|X)\neq P_o^{\mathcal{K}}(G|X)$. ■

ACKNOWLEDGMENT

This research is supported by NSF and DARPA under contract number(s) IIS-0916686, IIS-1219015, IIS-1017898, and N660001-1-2-4014. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of NSF, DARPA or the U.S. Government.

REFERENCES

[1] O. Frank and D. Strauss, "Markov graphs," *Journal of the American Statistical Association*, vol. 81:395, pp. 832–842, 1986.

[2] D. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–42, 1998.

[3] A. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, 1999.

[4] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal, "Stochastic models for the web graph," in *Proceedings of FOCS*, 2000.

[5] S. Moreno and J. Neville, "Network hypothesis testing using mixed kronecker product graph models," in *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, Dec 2013, pp. 1163–1168.

[6] P. Erdos and A. Renyi, "On the evolution of random graphs," in *Publication of the mathematical institute of the Hungarian Academy of Sciences*, 1960, pp. 17–61.

[7] F. Chung and L. Lu, "The average distances in random graphs with given expected degrees," *PNAS*, vol. 99, no. 25, pp. 15 879–15 882, 2002.

[8] J. Leskovec and C. Faloutsos, "Scalable modeling of real graphs using kronecker multiplication," in *Proceedings of the 24th international conference on Machine learning*, ser. ICML '07, 2007, pp. 497–504.

[9] S. Moreno, S. Kirshner, J. Neville, and S. Vishwanathan, "Tied kronecker product graph models to capture variance in network populations," in *Allerton'10*, 2010, pp. 17–61.

[10] M. Kim and J. Leskovec, "Multiplicative attribute graph model of real-world networks," in *Algorithms and Models for the Web-Graph*, ser. LNCS, vol. 6516. Springer Berlin Heidelberg, 2010, pp. 62–73.

[11] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *JMLR*, vol. 11, no. Feb, pp. 985–1042, 2010.

[12] H. Yun and S. V. N. Vishwanathan, "Quilting stochastic kronecker product graphs to generate multiplicative attribute graphs," in *AISTATS*, 2012, pp. 1389–1397.

[13] R. Sheldon, *A First Course in Probability*. Pearson Education, 2002.

[14] C. Groër, B. D. Sullivan, and S. Poole, "A mathematical analysis of the r-mat random graph generator," *Netw.*, vol. 58, no. 3, pp. 159–170, Oct. 2011. [Online]. Available: http://dx.doi.org/10.1002/net.20417

[15] F. J. Massey, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American Statistical Association*, vol. 46, pp. 68–78, 1951.

[16] S. 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*, ser. KDD '13, 2013, pp. 1052–1060.