2012

Tied Kronecker Product Graph Models to Capture Variance in Network Populations

Sebastian Moreno
Purdue University, smorenoa@purdue.edu

Jennifer Neville
neville@cs.purdue.edu

Sergey Kirshner
Purdue University, skirshne@purdue.edu

Report Number:
12-012

Tied Kronecker Product Graph Models to Capture Variance in Network Populations

SEBASTIAN MORENO, Computer Science department, Purdue University
JENNIFER NEVILLE, Computer Science department, Purdue University
SERGEY KIRSHNER, Statistic department, Purdue University

Much of the past work on mining and modeling networks has focused on understanding the observed properties of *single* example graphs. However, in many real-life applications it is important to characterize the structure of *populations* of graphs. In this work, we investigate the distributional properties of Kronecker product graph models (KPGMs) [Leskovec et al. 2010]. Specifically, we examine whether these models can represent the natural variability in graph properties observed *across* multiple networks and find surprisingly that they cannot. By considering KPGMs from a new viewpoint, we can show the reason for this lack of variance theoretically—which is primarily due to the generation of each edge independently from the others. Based on this understanding, we propose the *mixed Kronecker Product Graph Model* (mKPGM) a generalization of KPGMs that uses tied parameters to increase the variance of the model, while preserving the expectation. We evaluate the mKPGM model by comparing to several different graph models, through the multi-dimensional Kolgomorov Smirnov statistics, a new statistic that consider the relation among the characteristics of the networks. The results show mKPGMs are able to produce a closer match to real-world graphs, while still providing natural variation in the generated graphs.

Additional Key Words and Phrases: Graph generation models, social network analysis, Kronecker product graph models

1. INTRODUCTION

Graphs and networks are a natural data representation for analysis of a myriad of domains, ranging from systems analysis (e.g., the Internet) to bioinformatics (e.g., protein interactions) to psycho-social domains (e.g., online social networks). Due to the recent interest in small-world networks and scale-free graphs, there has been a great deal of research focused on developing generative models of graphs that can reproduce skewed degree distributions, short average path length and/or local clustering (see e.g., [Frank and Strauss 1986; Watts and Strogatz 1998; Barabsi and Albert 1999; Kumar et al. 2000]). The majority of these works have focused on procedural modeling techniques (see [Newman 2003] for a good overview). As an example, the preferential attachment model of [Barabsi and Albert 1999] is an incremental generative method, which repeatedly adds a node to the graph with a determined number of edges and each of the edges is linked up to existing nodes in the graph with probability proportional to its current degree (i.e., higher degree are more likely to receive additional incoming edges).

There are relatively few statistical models of graph structure that represent probability distributions over graph structures, with parameters that can be *learned* from example networks. One method is the Exponential Random Graph Model (ERGM) (also known as p* models) [Wasserman and Pattison 1996; Robins et al. 2006]. ERGMs represent probability distributions over graphs with an exponential linear model that uses feature counts of local graph properties considered relevant by social scientists (e.g., edges, triangles, paths). Another method is the Chung-Lu model [Chung and Lu 2002], which generates independent edges among nodes through a Bernoulli distribution with the parameter determined by the multiplication of node's weights. A third method is the Kronecker product graph model (KPGM) [Leskovec and Faloutsos 2007]. The KPGM is a fractal model, which starts from a small adjacency matrix with specified probabilities for all pairwise edges, and repeatedly applies the Kronecker product (of the matrix with itself) to grow the model to a larger size.

The aim, for much of the past work on generative graph models, has been to accurately capture the observed properties of a *single* graph – either global properties such as average path length or local graph properties such as transitive triangles. As such, evaluation of the proposed models has generally centered on empirical validation that observed graph properties match those of the generated graphs. Specifically, empirical analysis has consisted of visual comparison of properties of the input and a small number of generated graphs.

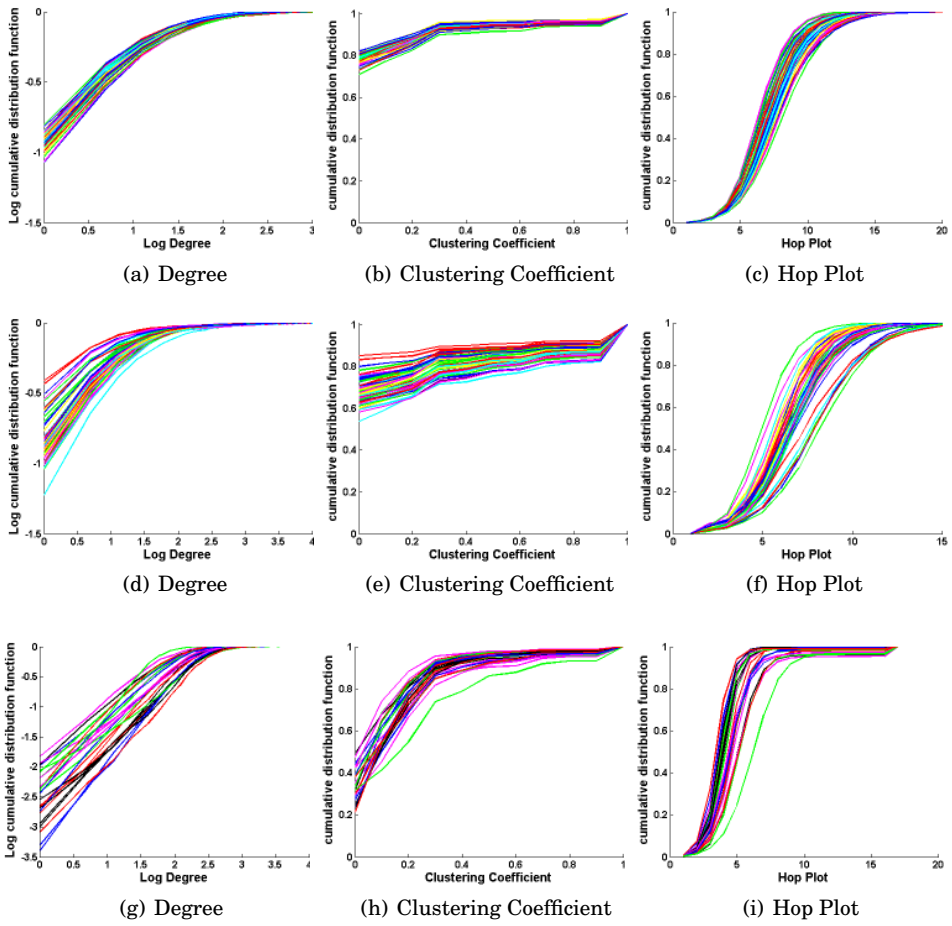
However, in many real-life applications one would like to model *populations* of graphs. That is, rather than capturing the properties of a single observed network, we would like to be able to capture the *range* of properties observed over multiple samples from a *distribution* of graphs. For example, in social network domains, the social processes that govern friendship formation are likely to be consistent across college students in various Facebook networks, so we expect that the networks will have similar structure, but with some random variation. Descriptive modeling of these networks should focus on acquiring an understanding of both their average characteristics and their variability. Similarly, when analyzing the performance of network protocols or network classification methods, we would like to measure performance across a set of network structures that capture the natural variability in these domains.

In recent work [Moreno and Neville 2009], we investigated the distributional properties of state-of-the art generative models for graphs. Specifically, we considered the case when more than one instance of a network is available, and examined whether these models capture the natural variability in graph properties observed *across* multiple networks. In other words, we evaluated whether the models able to match not only the mean graph statistics, but also their spread as observed in real network populations. Our analysis showed that graphs generated by KPGMs [Leskovec and Faloutsos 2007] and ERGMs [Wasserman and Pattison 1996] do not exhibit the variability observed in the network instances in two social network domains. What is particularly surprising is how little variance (compared to the real networks) was produced in the graphs generated from each model class. Each of the models appears to place most of the probability mass in the space of graphs on a relatively small subset of graphs with very similar characteristics (degenerate global models [Handcock 2003]).

Some theoretical insights to explain this phenomenon is available in the context of ERGMs. In recent work it was shown that learning ERGMs with only local features can lead to *degenerate* global models (i.e., the estimated distribution places most of its probability mass on either the empty or complete graphs) [Handcock 2003]. This indicates that the effect may be due to long-range dependencies in social networks which cannot be accurately captured by local features alone.

In this work, we investigate the issue in more detail for KPGM models. We show that a number of relatively simple approaches to increase variance in KPGMs do not work. This includes increasing the number of parameters, estimating and sampling from a posterior distribution of parameters, and learning the model from multiple networks. By considering KPGMs from a new viewpoint, we show the reason for this lack of variance theoretically—which is primarily due to the generation of each edge independently from the others. Based on this understanding we propose a generalization to KPGMs that uses tied parameters to increase the variance of the model, while preserving the expectation. We then show experimentally, that our *mixed-KPGM* can adequately capture the natural variability across a population of networks.

The rest of the paper is organized as follows. First, we describe the network data sets considered in the paper and examine their variability across several graph metrics (Section 2). Next, we provide a background over the three algorithms that we use in this work: KPGM, Chung-Lu and ERGM models (Section 3). We continue analyzing whether KPGM is capable of capturing the variability of the real networks and some



(a) Degree (b) Clustering Coefficient (c) Hop Plot

(d) Degree (e) Clustering Coefficient (f) Hop Plot

(g) Degree (h) Clustering Coefficient (i) Hop Plot

Fig. 1. Natural variability in the characteristics of a population of Purdue Facebook (top), Email (bottom) and AddHealth (bottom) networks.

early approaches to increase the variance of the model (Section 4). We consider the generative process for KPGMs from a slightly different angle which leads us to a variant of KPGM that allows higher variance and more clustering in sampled graphs (Section 5). Next, we explain the training algorithm for mixed-KPGM (Section 6). We then describe our experiment section and the new multidimensional Kolgomorov Smirnov statistics (Section 7), to continue with the application of our new approach to multiple instances of networks (Section 8) and discuss our findings and contributions (Section 9).

2. NATURAL VARIABILITY OF REAL NETWORKS

In this section we analyzed three specific characteristics over three undirected datasets without self loops, to observe the natural variability that can be found in real networks.

We conducted a set of experiments to explore three distributional properties of graphs found in natural social network populations: (1) degree, (2) clustering coefficient, and (3) path lengths. The degree of a node d_i is simply the number of nodes in the graph that are connected to node i. Degree is a local property of nodes, but since many networks have heavy-tailed degree distributions, the overall degree distribution is often considered a global property to match. Clustering coefficient is calculated for

a node i as: $c_i = \frac{2|\Delta_i|}{(d_i-1)d_i}$, where Δ_i is the number of triangles in which the node i participates and d_i is the number of neighbors of node i (degree for undirected networks). Clustering coefficient measures the local clustering in the graph. For path length, we consider the hop plot distribution in the graph, which refers to the number of nodes that can be reached with h "hops" in the graph: $N_h = \sum_v N_h(v)$, where $N_h(v)$ is the number of nodes that are $\leq h$ edges away from node v in G. The hop plot measures the global connectivity of the graph.

Specifically, we investigated three different real-world social network datasets. The first set is drawn from the public Purdue Facebook network. Facebook is a popular online social network site with over 845 million members worldwide. We considered a set of over 50000 Facebook users belonging to the Purdue University network with its over 400000 wall links consisting of a year-long period. To estimate the variance of real-world networks, we sampled 50 networks, each of size 2187 nodes, from the wall graph. To construct each network, we sampled an initial timepoint uniformly at random, then collected edges temporally from that point (along with their incident nodes) until the node set consisted of 2187 nodes. In addition to these nodes and initial edge set, we collected all edges among the set of sampled nodes that occurred within a period of 60 days from the last nodes added to the network (this increased the connectivity of the sampled networks). The characteristics of the set of sampled networks is graphed in Figure 1 (top), with each line corresponding to the cumulative distribution of a single network. The figures show the similarity among the sampled networks as well as the variability that can be found in real domains for networks of the same size.

The second dataset consists of a collection of emails recollected for two weeks from the email logs on the Purdue mailserver(s) [Ahmed et al. 2011]. The original data consists of over 200000 nodes with over 2 millions edges, where an edge represents an email from one user to another. Some of the nodes has an incredibly number of outgoing or incoming mails which are likely to be mailing lists or automatic mailing systems, while other nodes have not receive or send an email during these two weeks. To eliminate these 'anomalies', we remove any node with an outgoing degree greater than 1000 nodes and any node that has an incoming or outgoing degree of 0. The filtered data has over 30000 nodes and over 700000 edges. Similarly to Facebook Data, we sampled 50 networks, each of then with 2187 nodes. Applying the same process than before, most of the networks were mostly disconnected because of the huge number of emails among user. To avoid the huge number of disconnected components, we sampled an initial timepoint uniformly at random and recollected, in the smallest time possible, a determined number of connected nodes. The rest of the nodes and edges were recollected using the same process than Facebook data with a time window of 36 hours. The characteristics of this dataset can be observed in figure 1 (middle).

A third smaller dataset was utilized to test some of our first approaches. This dataset consists of a set of social networks from the National Longitudinal Study of Adolescent Health (AddHealth) [Harris 2008]. The AddHealth dataset consists of survey information from 144 middle and high schools, collected (initially) in 1994-1995. The survey questions queried for the students' social networks along with myriad behavioral and academic attributes, to study how social environment and behavior in adolescence are linked to health and achievement outcomes in young adulthood. In this work, we considered the social networks from 25 schools with sizes varying from 800 to 2000 nodes. The characteristic of these networks are showed in Figure 1 (bottom), where, despite the fact that the networks are of different size, since we compare cumulative distributions, the networks have very similar characteristics.

We consider these sets of networks to be illustrative examples of *populations* of graphs (i.e., drawn from the same distribution). These sets are likely to be af-

fected/generated by similar social processes (college/high school friendships and email communication patterns). In addition, these sets exhibit some similarities in their graph structures, yet with some variation due to random effects.

3. BACKGROUND

For this work, our aim is to study and match the distributional properties of probabilistic models of graph structure using our new method Mixed Kronecker Product Graph Model. We compare, the results against three representative models that can be "learned" from an observed network data.

3.1. Kronecker Model

The Kronecker product graph model (KPGM) [Leskovec and Faloutsos 2007] is a fractal model, which assumes the entries of the adjacency matrix are drawn independently as Bernoulli trials from a probability matrix which is itself an integer Kronecker power of a small matrix. It has been shown empirically that this approach successfully preserves a wide range of global properties of interest, including degree distributions, eigenvalue distributions, and path-length distributions [Leskovec et al. 2010].

More specifically, the model generates self-similar graphs, in a recursive way using Kronecker multiplication. The algorithm starts with a initial matrix $\mathcal{P}_1 = \Theta$ with b rows and columns, where each cell value is a probability. Typically $b = 2$ or 3, e.g.:

$$\mathcal{P}_1 = \Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix}$$

To generate graphs of a larger size, the Kronecker product of $\mathcal{P}_1$ is taken $K-1$ times with itself to generate a matrix:

$$\mathcal{P}_K = \mathcal{P}_1^{[K]} = \mathcal{P}_1^{[K-1]} \otimes \mathcal{P}_1 = \underbrace{\mathcal{P}_1 \otimes \ldots \otimes \mathcal{P}_1}_{K-1 \text{ times}}$$

with b^K rows and columns. We will denote the (u, v)-th entry of $\mathcal{P}_K$ by $\pi_{uv} = \mathcal{P}_k[u, v]$, $u, v = 1, \ldots, b^K$. Under KPGM, a graph $G = (\mathbf{V}, \mathbf{E})$, with set of nodes $\mathbf{V} = \{1, \ldots, N\}$ where $N = b^K,\ K \in \mathbb{N}$, is sampled by performing mutually independent Bernoulli trials for each pair (u, v) with probability $\pi_{uv} = \mathcal{P}_K[u, v]$ and placing an edge (u, v) into $\mathbf{E}$ if the trial for (u, v) results in a success.

To estimate a KPGM from an observed graph $G^\star$, the learning algorithm uses maximum likelihood estimation to determine the values of Θ that have the highest likelihood of generating $G^\star$: $l(\Theta) = \log P(G^\star|\Theta) = \log \sum_\sigma P(G^\star|\Theta, \sigma)P(\sigma)$ where σ defines a permutation of rows and columns of the graph $G^\star$. The model assumes that each edge is a Bernoulli random variable, given $\mathcal{P}_1$. Therefore the likelihood of the observed graph $P(G^\star|\Theta, \sigma)$ is calculated as:

$$P(G^\star|\Theta, \sigma) = \prod_{(u,v)\in E} \mathcal{P}_K[\sigma_u, \sigma_v] \prod_{(u,v)\notin E} (1 - \mathcal{P}_K[\sigma_u, \sigma_v]) \tag{1}$$

where σ_u refers to the permuted position of node u in σ.

With this formula, the algorithm uses a gradient descent approach to search for the MLE parameters $\hat{\Theta}$, however the gradient of $l(\Theta)$ involves a summation over an exponential number of permutations over σ. To avoid this calculation, the algorithm simulates draws from the permutation distribution $P(\sigma|G^\star, \Theta)$ until it converges. Then it calculates the expected values of $l(\Theta)$ and the gradient. This sampling is performed with a Metropolis-Hastings algorithm where a new sample σ_s is accepted if the ratio of

the likelihood with σ_{s-1} is greater than a random number uniformly sampled between 0 and 1.

In every iteration of the algorithm, $l(\Theta)$ and its gradient are calculated T times, obtaining their corresponding mean. The parameters $\hat{\Theta}$ are updated with the following until convergence: $\hat{\Theta}_{t+1} = \hat{\Theta}_t + \lambda \frac{\partial l(\hat{\Theta})}{\partial \Theta_t}$.

A second learning algorithm for KPGMs was developed by Gleich and Owen [Gleich and Owen 2012]. This learning algorithm is independent of the permutation of the network (permutation invariant) which avoids the difficulty of search over the permutation space. The training utilizes the method of moments, to search over all possible parameters Θ while minimizing the following function:

$$f(\Theta, \mathbf{F}(G^*)) = \sum_{i=1}^{n_m} \left(\frac{F_i(G^*) - E[F_i(G)|\Theta]}{F_i(G^*)} \right) \tag{2}$$

Here $\mathbf{F}(G^*) = \{F_1, F_2, \cdots, F_{n_m}\}$ corresponds to a set of n_m "moments" of the observed training network and $E[F_i(G)|\Theta]$ is the expected value of the i moment of the network G generated by Θ. The method of moments (MoM) training algorithm for KPGMs considers four moments: the number of (i) edges, (ii) 2-stars, (iii) 3-stars, and (iv) triangles. These moments were selected by the authors because their expected values can be analytically calculated for any Θ, given $b = 2$.

3.2. Chung-Lu Model

The Chung-Lu model is an extension of the Erdos-Renyi model [Erdos and Renyi 1960]. The Chung-Lu model represents the expected degree distribution of a network through a set of weights. To generate a sample graph from the model, each edge is sampled independently with a Bernoulli distribution with parameter $w_i w_j$, where w_i and w_j represent the learned weight for nodes i and j respectively. This generation process allows to generate a new graph where the expected degree distribution matches the degree distribution of real networks.

The training algorithm for Chung-Lu model is determined by the degree of the nodes. Given a graph $G = (\mathbf{V}, \mathbf{E})$, let A be the adjacency matrix for G^* where $A_{ij} = 1$ if the edge $(i, j) \in \mathbf{E}^*$ and 0 otherwise. Defining the diagonal matrix D such that: $D_{ij} = \sum_k A_{ik}$ if $i = j$ and 0 otherwise, this diagonal matrix represents the degree of each node, where D_{ii} is the degree of node i. Based on the diagonal matrix D, the weights are defined by $w_i = D_{ii}/|\mathbf{E}^*|$ where $|\mathbf{E}^*|$ corresponds to the total number of edges in the graph.

Assuming that $D_{ii} < \sqrt{|\mathbf{E}^*|}\ \forall i$, the expected degree distribution of a generated graph is given by:

$$E_G[D_{ii}^{CL}] = \sum_j \frac{D_{ii} D_{jj}}{|\mathbf{E}^*|} = D_{ii} \sum_j \frac{D_{jj}}{|\mathbf{E}^*|} = D_{ii}$$

which corresponds to the degree distribution of the real network. This characteristic permits to generate networks with very similar degree distribution to the real networks using the same number of parameters than nodes $|\mathbf{V}| = N$.

3.3. ERGM Model

The third model is the exponential random graph model (ERGM) [Wasserman and Pattison 1996; Robins et al. 2006] from the mathematical sociology community (also known as the p* model). ERGMs represent probability distributions over graphs with an exponential linear model that uses feature counts of local graph properties considered relevant by social scientists (e.g., edges, triangles, paths).

The approach is focused on modeling the local structure of networks such as triangles or k-stars. For a graph G, we will denote its features with the vector $\mathbf{F}(G)$. ERGM models define a probability distribution of $P(G)$ over the set of possible graphs $\mathcal{G}$:

$$P(G|\eta) = \frac{1}{Z(\eta)} \exp\left(\eta^T \mathbf{F}(G)\right)$$

where $Z(\eta) = \sum_{G' \in \mathcal{G}} \exp\left(\eta^T \mathbf{F}(G')\right)$ is the standard partition function.

As with all exponential models, P can be estimated using the maximum entropy principle: Choose the parameter $\hat{\eta}$ which maximizes $l(\eta)$ the log-likelihood of the input graph $G^\star$. It is easy to show that this corresponds to the model that matches the expected values of features for graphs in $\mathcal{G}$ to the features of the input graph $G^\star$ (i.e., $\mathbf{F}(G^\star) = E_{P(\mathcal{G})}[\mathbf{F}(\mathcal{G})]$). The feature constraints ensure that the local properties of the graph are preserved and the partition function Z ensures global consistency. However, it has recently been discovered that estimation with ERGMs using only local features can result in *near-degenerate* models that place all their probability mass on either the complete or empty graph, and in which the original graph is very unlikely [Handcock 2003]. Sociologists have alleviated extreme degeneracy problems by manually specifying a larger number of features to use in ERGM models, such as alternating k-stars and alternating k-paths [Snijders et al. 2004]. Although, this work has begun to explore a more expansive set of features, it is driven primarily by sociological motivations—the researchers are more focused on identifying subgraph patterns that are semantically meaningful (e.g., alternating k-paths measure the connections among a pair of nodes that are not directly linked) rather than identifying local patterns that will improve preservation of the global graph structure.

4. VARIABILITY OF KPGM MODELS

In this section, we evaluate KPGMs both empirically and analytically to investigate whether graphs generated from learned KPGM models can capture the distributional properties we observe in real-world social networks. We also show our first approaches to increases the variance of the KPGM model, which leads to the conclusion that the lack of variance is due to the independent generation of edges.

4.1. Assessing Variability

To learn KPGM models, we selected a single network from each dataset to use as a training set. To control for variation in the samples, we selected the network that was closest to the median of the degree distributions for Facebook and Email dataset. In the Purdue Facebook data, we selected the eleventh generated network with 5634 edges and in Email data, we selected the twenty-forth network with 6666 edges. For AddHealth, we selected the network from school 72 with 2045 nodes and 15484 edges, which is the closest network to the training parameters that we use ($b = 3, K = 7 \Rightarrow 3^7 = 2187$ nodes).

Using each selected network as a training set, we learned a KPGM model ($b = 3$) using maximum likelihood estimation to estimate $\hat{\Theta}$. For each learned model, we generated 50 sample graphs. The KPGM graphs were generated using the estimated matrix $\mathcal{P}_K = \mathcal{P}_1^{[K]}$. From the 50 samples, we estimated the empirical sampling distributions for degree, clustering coefficient, and hop plots.

The results, for the original datasets and the KPGM generated data, are plotted in figures 15-17 in Section 8.2. The plots, show the median and interquartile range for the set of observed network distributions. Solid lines correspond to the median of the distributions; dashed lines: the 25th and 75th percentiles.

The results in figures 15-17 show two things. First, the KPGM performs as expected, partially capturing the graph properties that have been reported in the past. Specifically, the KPGM model is able to capture some of the degree and hop plot fairly well in the datasets, but it is not able to model the local clustering in none of them. Second, the KPGM model does not reproduce the amount of variance exhibited in the real networks. Moreover, it is surprising that the variance of the generated graphs is so slight that it is almost not apparent in some of the plots. The lack of variance implies that while the KPGM model may be able to reasonably capture the patterns of the input graph, it cannot be used to generate multiple "similar" graphs—since it appears that the generated graphs are nearly isomorphic.

We conjecture that it is KPGM's use of independent edge probabilities and fractal expansion that leads to the small variation in generated graphs. In fact, one obvious possibility is that the low variance is due to the small number of model parameters used in the initiator matrix.

4.2. Increasing Variability through initiator matrix

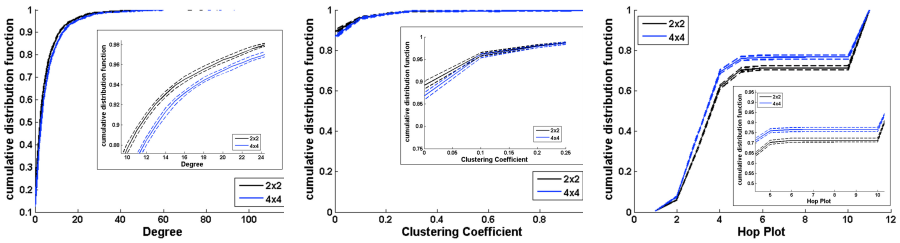
To investigate the change in variance for models with initiator matrices of varying sizes, we conducted the following simulation experiment. We first manually specified a 2×2 model. Then to create a 4×4 matrix that would produce graphs similar to the 2×2 model, we computed the Kronecker product of the 2×2 matrix and then perturbed the parameter in cell by adding a random number $\sim \mathcal{N}(0, 9E{-}4)$. The resulting parameters are listed below

$$\Theta_{2\times 2} = \begin{bmatrix} 0.95 & 0.60 \\ 0.60 & 0.20 \end{bmatrix}$$

$$\Theta_{4\times 4} = \begin{bmatrix} 0.8849 & 0.6355 & 0.5659 & 0.3634 \\ 0.6355 & 0.2220 & 0.3618 & 0.1171 \\ 0.5659 & 0.3618 & 0.1650 & 0.1288 \\ 0.3634 & 0.1171 & 0.1288 & 0.0614 \end{bmatrix}$$

From these specified matrices, we generated 100 networks of size 1024 and measured the variance in the graph distributions. Figure 2 shows the results, which clearly indicate that the variance does not increase. We also tried learning KPGM models of the real datasets (e.g., AddHealth) with initiator matrices up to size 6×6 with no noticeable increase in variance. This indicates that we are unlikely to achieve higher variance by increasing the parametrization of the model. Although much larger initiator matrices (e.g., 30×30) would increase the number of parameters in the model, it would also decrease the number Kronecker products needed to generate a graph of a particular size, which may impact the model's ability to represent fractal structure.

Based on these investigation, we conjecture that the lack of variance is due to model placing most of the probability mass on a few nearly isomorphic graphs for any particular set of parameters $\hat{\Theta}$. We note however, that Figure 2 shows that although the parameters for the two models are similar, the distributions do not match exactly (i.e., the medians are different for the two models). This implies that slight perturbations to the Θ parameters could increase the variance of the generated graphs. Based on this observation, we posit that a Bayesian estimation approach, for the KPGM model, could accurately capture the natural variance in a given domain. Bayesian methods estimate a posterior distribution over parameters values—and if we use the posterior distribution over Θ to sample a set of parameters before each graph generation, we should be able to generate graphs with higher variation that we see with a single MLE set. We outline this approach in detail next.



(a) Degree (b) Clustering Coefficient (c) Hop Plot

Fig. 2. Variance of generated networks, comparing 2×2 to 4×4 KPGM initiator matrices.

4.3. Increasing Variability through Bayesian approaches

Our goal is to generate graphs that can capture the variability observed in real graphs, and in Section 4.1 we used the maximum likelihood (ML) estimate $\hat{\Theta}$ to generate graphs $G \sim P_{KPGM}\left(\cdot|\hat{\Theta}\right)$. Instead of using a point estimate of Θ, we propose to sample the graphs from the predictive posterior distribution $P\left(G|G^{\star}\right)$ where $G^{\star}$ is the observed graph[1]:

$$P\left(G|G^{\star}\right) = \int_{\Theta} P\left(\Theta|G^{\star}\right) P\left(G^{\star}|\Theta\right) d\Theta. \tag{3}$$

Our previous approach approximated sampling from the predictive posterior by generating graphs using the supposed mode of the posterior distribution $P\left(\Theta|G^{\star}\right)$. However, such approximation may not be just in this case as (1) the likelihood surface is not convex and may have multiple modes (not reducible from one to another by a permutation of rows and columns of the initiator matrix); (2) likelihood surface may be flat around the found mode with many different matrices Θ explaining the observed data well; and (3) the ML approach in [Leskovec and Faloutsos 2007] is not even guaranteed to converge (to a mode).

Exact sampling from the predictive posterior is infeasible except for trivial graphs, and neither is sampling directly from the posterior $P\left(\Theta|G^{\star}\right)$ as it requires averaging over permutations of node indices. We can however obtain a Markov Chain Monte Carlo estimation in the augmented space of parameters and permutations as $P\left(\sigma,\Theta|G^{\star}\right) = P\left(\sigma\right)P\left(G^{\star}|\Theta,\sigma\right)/P\left(G^{\star}\right)$ can be computed up to a multiplicative constant. Our approach alternates the draws from the posterior distribution over the parameters Θ and permutations σ. The resulting sampling procedure resembles the maximum likelihood learning algorithm for KPGM [Leskovec and Faloutsos 2007] except that instead of updating Θ by moving along the gradient of the log-likelihood, we resample Θ.

Given a set of parameters Θ and a permutation σ, we use Metropolis-Hastings algorithm (e.g., [Robert and Casella 2004]) to resample a new permutation σ^{new}. As was mentioned in [Leskovec and Faloutsos 2007], the ratio of the posteriors can be computed efficiently,

$$\frac{P\left(\sigma^{new}|G^{\star},\Theta\right)}{P\left(\sigma|G^{\star},\Theta\right)} = \frac{P\left(G^{\star}|\Theta,\sigma^{new}\right)}{P\left(G^{\star}|\Theta,\sigma\right)} = \prod_{(u,v)\in\mathbf{E}} \frac{\mathcal{P}_K\left[\sigma_u,\sigma_v\right]}{\mathcal{P}_K\left[\sigma_u^{new},\sigma_v^{new}\right]} \prod_{(u,v)\notin\mathbf{E}} \frac{\left(1-\mathcal{P}_K\left[\sigma_u,\sigma_v\right]\right)}{\left(1-\mathcal{P}_K\left[\sigma_u^{new},\sigma_v^{new}\right]\right)}$$

Given a permutation σ, $G^{\star}|\sigma$ can be viewed as a graph whose vertices has been relabeled according to σ. For a proposal distribution, we sample uniformly from all possi-

[1] Our approach easily extends to a collection of observed graphs.

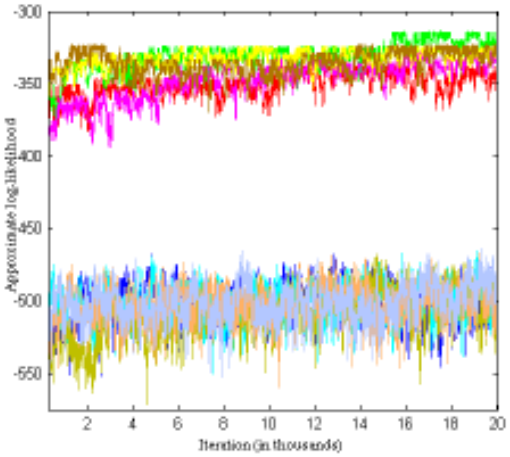


Fig. 3. 10 MCMC chains for BKPGM on a graph with 256 nodes sampled from the initiator matrix with parameters $\theta_{11} = 0.6$, $\theta_{12} = 0.5$, $\theta_{22} = 0.2$, true joint log-likelihood$= -732.2$. MCMC was run with 100 resamples of permutations in each iteration. Chains converge to two different modes with initiator matrix parameters very different from the truth.

ble swaps of two node indices of this graph $G^\star|\sigma$ thus resulting in another permutation (same as in [Leskovec and Faloutsos 2007]). Similarly, we can compute the ratio of the posterior densities over the parameter θ_{ij},

$$\frac{P\left(\theta_{ij}^{new}|G^\star, \Theta_{-ij}, \sigma\right)}{P\left(\theta_{ij}|G^\star, \Theta_{-ij}\right), \sigma} = \frac{P\left(G^\star|\theta_{ij}^{new}, \Theta_{-ij}, \sigma\right)}{P\left(G^\star|\Theta, \sigma\right)},$$

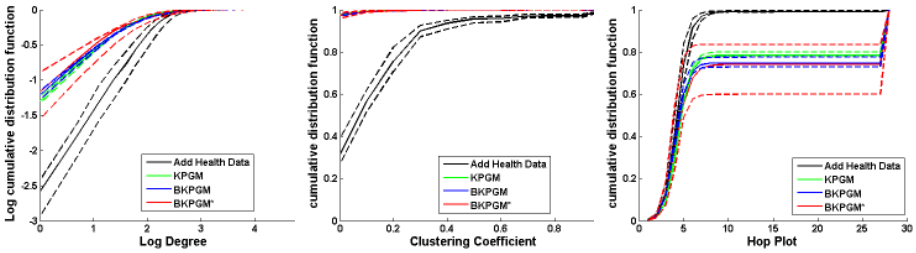
where Θ_{-ij} is the set of parameters excluding θ_{ij}. We considered two proposal distributions $J\left(\theta_{ij}^{new}|\theta_{ij}\right)$ for θ_{ij}^{new}, the uniform $U\left(0,1\right)$ and $Beta\left(\alpha,\beta\right)$. For $Beta$, α and β were chosen so that the mode was equal to θ_{ij} under a constraint $\alpha + \beta = 50$.

We ran multiple chains from randomly selected initial values of Θ and σ until they converged to a mode of the posterior (as evidenced by the convergence of the joint likelihood $P\left(\sigma, \Theta, G^\star\right) \propto P\left(G^\star|\Theta, \sigma\right)$), with samples from the posterior over the parameters drawn from that point on. Unfortunately, due the size of the space of permutations, the algorithm is slow to converge for graphs with large number of nodes.

When investigating the behavior of Bayesian KPGM (BKPGM) on the simulated data, we observed that chains converged to multiple modes, none of which corresponding to the original initiator matrix (e.g., see Figure 3). This suggests that a likelihood optimization procedure could possibly find local maxima.

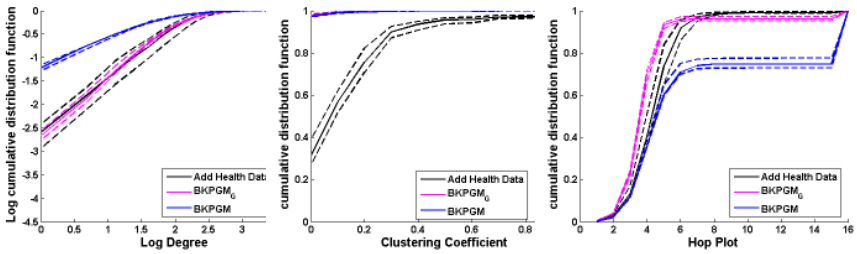
To assess the performance of BKPGM we repeated the experiments reported in subsection 4.1. Given the large converging time of the algorithm, we decreased the initiator matrix ($b = 2$) and the size of the training network (school 83, 1278 nodes and 7982 edges) to see initial results. In this experiment, we compared the following approaches:

— KPGM: In this approach we use the original MLE estimation procedure (see Section 3.1) to learn the model, and networks generated with $\Theta = \hat{\Theta}_{MLE}$.
— BKPGM: In this approach we use the Bayesian estimation procedure recently outlined to estimate the posterior $P(\Theta|G^\star)$. Networks are then generated by sampling Θ from a $Beta(\alpha,\beta)$ distribution, where α and β are set such that $\frac{(\alpha-1)}{(\alpha+\beta-2)} = \hat{\Theta}_{MAP}$ and $\frac{(\alpha\times\beta)}{(\alpha+\beta)^2(\alpha+\beta+1)} = Var(\hat{\Theta})$.



(a) Degree (b) Clustering Coefficient (c) Hop Plot

Fig. 4. Variation of graph properties in generated AddHealth networks.



(a) Degree (b) Clustering Coefficient (c) Hop Plot

Fig. 5. Variation of graph properties in generated AddHealth networks.

— BKPGM*: In this approach we use the BKPGM posterior $P(\Theta|G^{\star})$ and generate networks by sampling Θ from a $Beta(\alpha, \beta)$ distribution, where α and β are set such that $\frac{(\alpha-1)}{(\alpha+\beta-2)} = \hat{\Theta}_{MAP}$ and $\frac{(\alpha\times\beta)}{(\alpha+\beta)^2(\alpha+\beta+1)} = 0.0025$. This method is included to explore the effects of artificially inflating the variance of the posterior for Θ.

Observing the results of Figure 4, the BKPGM* algorithm shows that it is indeed possible to augment artificially the variance of either the BKPGM or the KPGM model to capture the natural variability in a network domain, but this approach is completely subjective and it is not possible to realize a good estimation using the data. We tried to increase the (estimated) variance of the BKPGM model by learning over multiple datasets defining a new model called BKPGM$_{\mathbf{G}}$.

BKPGM$_{\mathbf{G}}$ uses the Bayesian estimation procedure but we learn the model from multiple network examples. Specifically, we estimate the posterior $P(\Theta|\mathbf{G})$, where $\mathbf{G}$ is all the networks for a specific domain. Then we approximate the posterior with a $Beta$ distribution and sample Θ for graph generation in the same manner as for BKPGM. This method is included to explore the effects of learning from multiple samples (rather than a single graph).

In Figure 5 we compare the BKPGM and BKPGM$_{\mathbf{G}}$ models on the AddHealth data. The aim of this comparison is to investigate whether we can increase the (estimated) variance of the BKPGM model by learning over multiple datasets (compared to conventional methods which typically use a single network for estimation). For this reason, we only directly compare to BKPGM and we omit the KPGM and BKPGM* results for clarity. Unfortunately, the results show that the BKPGM$_{\mathbf{G}}$ graphs still do not exhibit sufficient variability, even though they are estimated from the set of networks that exhibit that variability. However, surprisingly the BKPGM$_{\mathbf{G}}$ graphs fit the observed distributions for degree and hop plot much better than BKPGM (and KPGM). This indicates that the models are overfitting when they are learned from a single network.

Based on these investigations, we conjecture that it is KPGMs use of independent edge probabilities and fractal expansion that leads to the small variation in generated graphs. We explore this issue analytically next.

4.4. Theoretical Analysis

We start with a description of how Kronecker product determines these probabilities $\pi_{uv} = \mathcal{P}_K[u,v]$. For convenience, we will label the nodes $\{0,\ldots,N-1\}$ instead of $\{1,\ldots,N\}$ ($N = b^K$). Assume that matrix $\mathcal{P}_1$ is indexed by (i,j) with $i,j = 1,\ldots,b$. Let $(v_K v_{K-1}\ldots v_2 v_1)_b$ be a representation of a number v in base b. We will refer to it as a b-nary representation for v (it is also sometimes called b-adic), and will refer to v_l as the l-th b-it of v.[2] Then

$$v = \sum_{l=1}^{K} (v_l - 1) b^{l-1} \tag{4}$$

with each $v_l \in \{1,\ldots,b\}$. For a given b, such representation is unique for all $v \in \{0,\ldots,N-1\}$. As was pointed out in [Mahdian and Xu 2007] for $b = 2$, and mentioned in [Leskovec et al. 2010], for $u,v \in \{0,\ldots,N-1\}$ with b-nary representations $u = (u_K \ldots u_1)_b$ and $v = (v_K \ldots v_1)_b$,

$$\pi_{uv} = \mathcal{P}_K[u,v] = \prod_{l=1}^{K} \mathcal{P}_1[u_l, v_l] = \prod_{l=1}^{K} \theta_{u_l v_l}. \tag{5}$$

Given a matrix of edge probabilities $\mathcal{P}_K = \mathcal{P}_1^{[K]}$, a graph G with adjacency matrix A is realized (sampled or generated) from KPGM with the initiator matrix $\mathcal{P}_1$ by setting $A_{uv} = 1$ with probability $\pi_{uv} = \mathcal{P}_k[u,v]$ and to 0 with probability $1 - \pi_{uv}$. A_{uv}s can be thought of as Bernoulli trials or binary random variables. With a slight abuse of notation, let $|\mathbf{E}| = \sum_i \sum_j A_{ij}$ be the number of edges in a random graph $G = (\mathbf{V}, \mathbf{E})$ with $N = b^K = |\mathbf{V}|$ nodes sampled from KPGM. Then

$$\begin{aligned} E[|\mathbf{E}|] &= \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} E[A_{uv}] = \sum_{i_1=1}^{b}\sum_{j_1=1}^{b}\cdots\sum_{i_K=1}^{b}\sum_{j_K=1}^{b}\prod_{l=1}^{K}\theta_{u_l v_l} \\ &= \sum_{i_1=1}^{b}\sum_{j_1=1}^{b}\theta_{i_1 j_1}\cdots\sum_{i_K=1}^{b}\sum_{j_K=1}^{b}\theta_{i_K j_K} = \left[\sum_{i=1}^{b}\sum_{j=1}^{b}\theta_{ij}\right]^K = S_\Theta^K \end{aligned}$$

with $S_\Theta = \sum_{i=1}^{b}\sum_{j=1}^{b}\theta_{ij}$ is the sum of entries in the initiator matrix $\mathcal{P}_1$. We can find the variance $Var(|\mathbf{E}|)$ similarly. Since A_{uv}s are independent,

$$\begin{aligned} Var(|\mathbf{E}|) &= \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} Var(A_{uv}) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\pi_{uv}(1-\pi_{uv}) \\ &= \sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\pi_{uv} - \sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\pi_{uv}^2 = \left[\sum_{i=1}^{b}\sum_{j=1}^{b}\theta_{uv}\right]^K - \left[\sum_{i=1}^{b}\sum_{j=1}^{b}\theta_{uv}^2\right]^K = S_\Theta^K - S_{\Theta^2}^K \end{aligned} \tag{6}$$

[2] In this case, the digits are $1,\ldots,b$ rather than the standard $0,\ldots,b-1$.

Data set	Estimated mean	Estimated variance
Purdue Facebook	5753	10587
Purdue Email	6605	58844
AddHealth	8021	9045070

Fig. 6. Observed mean and variance for total number of edges for real data

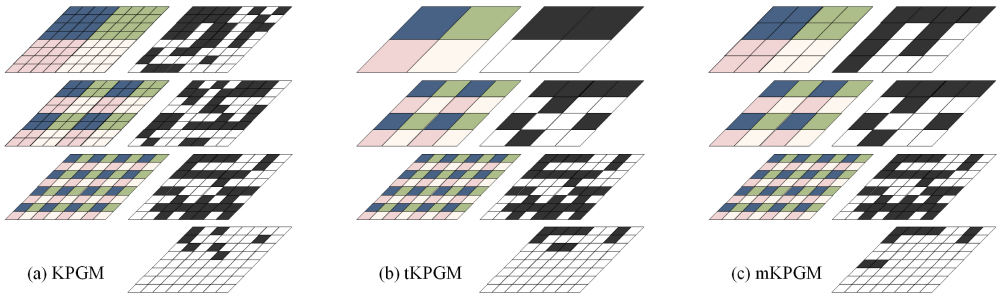


Fig. 7. Generative mechanisms for different Kronecker product graph models: (a) KPGM, (b) tKPGM, (c) mKPGM. Probability matrices are on the left, realizations are on the right. The bottom left matrices are the final sample from each model. For KPGM, $b^k \times b^k$ independent Bernoulli trials are performed at each level. For tKPGM, only $b^l \times b^l$ independent Bernoulli trials are performed at a level l; the result of each trial is then replicated for all $b^{k-l} \times b^{k-l}$ entries in the corresponding submatrix. For mKPGM, the first l levels ($l = 2$ in the figure) are untied.

where $S_{\Theta^2} = \sum_{i=1}^{b} \sum_{j=1}^{b} \theta_{ij}^2$ is the sum of the squares of the entries in the initiator matrix $\mathcal{P}_1$.

Note that $Var(|\mathbf{E}|) \leq E[|\mathbf{E}|]$ and $SD(|\mathbf{E}|) \leq \sqrt{E[|\mathbf{E}|]}$ independently of the value K. However, in the real-world networks considered in this paper, the estimated variance significantly exceeds the mean—in Facebook the estimated mean number of edges is 5753, while the variance is 10587. Similarly, in Email the mean number of edges is 6605, while the variance is 58844. Finally, in AddHealth the average number of edges and variance is 8021 and 9045070 respectively. This indicates that KPGM models are incapable of reproducing the variance of these real-world network populations. —see Table of Figure 6 where we report the observed mean and variance for the number of edges in the Facebook, Email and AddHealth datasets.

Considering that the variance of the number of edges is lower than the expected number of edges, it is impossible to generate network with enough variance using the KPGM model. For this reason, we analyze another approach to increment the variance of the number of edges modifying the sample process.

5. EXTENDING KPGM TO INCREASE VARIANCE

In this section, we propose a generalization of KPGM that permits larger variance in the properties of the generated graphs by introducing edge dependence in the generation process. This approach increases the variance in the number of edges by introducing positive covariance between the edges. We also realizes some empirical simulations to show the difference between KPGM and our proposed model.

5.1. Another View of Graph Generation with KPGMs

Before introducing our model variant, we present a slightly different view of the graph generation under KPGM. This viewpoint provides an extra dimension to the model that if exploited allows a natural way to couple the edge generation and thus to increase the variance of the graph statistics.

Remembering the previous section, given a matrix of edge probabilities $\mathcal{P}_K = \mathcal{P}_1^{[k]}$, a graph $G = (\mathbf{V}, \mathbf{E})$ with adjacency matrix $A = R(\mathcal{P}_K)$ is *realized* (sampled or generated) by setting $A_{uv} = 1$ with probability $\pi_{uv} = \mathcal{P}_K[u, v]$ and setting $A_{uv} = 0$ with probability $1 - \pi_{uv}$. A_{uv}s are realized through a set of Bernoulli trials or binary random variables (e.g., $\pi_{uv} = \theta_{11}\theta_{12}\theta_{11}$). For example, in Figure 7a, we illustrate the process of a KPGM generation for $K = 3$ to highlight the multi-scale nature of the model. Each level correspond to a set of separate trials, with the colors representing the different parameterized Bernoullis (e.g., θ_{11}). For each cell in the matrix, we sample from three Bernoullis and then based on the set of outcomes the edge is either realized (black cell) or not (white cell).

To formalize this, we start with a description of probabilities in the stochastic Kronecker matrix $\mathcal{P}_K$. Assume $N = b^K$ and index the entries of the initiator matrix $\mathcal{P}_1$ with (i, j), $i, j = 1, \ldots, b$. Let $(v_1 \ldots v_K)_b$ be the representation of a number v in base b (equation 4) and $\pi_{uv} = \prod_{l=1}^{K} \theta_{u_l v_l}$ (equation 5).

This description highlights the multiscale nature of KPGM. The probability of having an edge (u, v) in a graph realization from $\mathcal{P}_K$ is equal to the product of contributions (probabilities $\theta_{u_\ell v_\ell} = \mathcal{P}_1[u_\ell, v_\ell]$) from different scales (ℓ), with higher order b-its (corresponding to small ℓ) responsible for the high-level structure of the graph, and the lower order b-its (corresponding to ℓ close to K) are responsible for the local structure.

Alternatively, each A_{uv} can be thought of as drawn in K stages, one for each b-it of u and v. Let A_{uv}^{ℓ} be a binary random variable with $P\left(A_{uv}^{\ell} = 1\right) = \theta_{u_\ell v_\ell}$ and $P\left(A_{uv}^{\ell} = 0\right) = 1 - \theta_{u_\ell v_\ell}$. Then $I_{uv} = I_{uv}^{K} \wedge I_{uv}^{K-1} \wedge \cdots \wedge I_{uv}^{1} = \prod_{\ell=1}^{K} A_{uv}^{\ell}$ or in other words, an edge (u, v) is included if and only if the trials A_{uv}^{ℓ} resulted in a success for *all* scales $\ell = 1, \ldots, K$. Equivalently, an ℓ-th scale adjacency matrix $A^{\ell} = \left(A_{uv}^{\ell}\right)$ is realized from $(\mathcal{P}_K)_\ell = \underbrace{\mathbf{1}_b \otimes \ldots \otimes \mathbf{1}_b}_{\ell-1} \otimes \mathcal{P}_1 \otimes \underbrace{\mathbf{1}_b \otimes \ldots \otimes \mathbf{1}_b}_{K-\ell}$ where $\mathbf{1}_b$ is a $b \times b$ matrix of ones. An adjacency matrix $A = A^1 \circ \cdots \circ A^K$ is an entriwise (Hadamard) product of the adjacency matrices at K scales. See illustration in Fig 7(a).

Note that each matrix $(\mathcal{P}_K)_\ell$ consists only of the values of the initiator matrix $\mathcal{P}_1$. Each of these values is repeated $b^{K-1} \times b^{K-1}$ times and is contained in the intersection of b^{K-1} rows and columns, with the value θ_{ij} appearing in rows u with $u_\ell = i$ and columns v with $v_\ell = j$. Because of the Kronecker product structure of $(\mathcal{P}_K)_\ell$, pairs (u, v) corresponding to the same probability values appear in blocks of $b^{K-\ell} \times b^{K-\ell}$. It is important to note that even though many of A_{uv}^{ℓ} have the same probability distribution, under KPGM they are all sampled independently of one another. Relaxing this assumption will lead to extension of KPGMs capable of capturing the variability of real-world graphs.

5.2. Tied KPGM

Even though the probability matrix $\mathcal{P}_K$ exhibits hierarchical or multiscale structure, this hierarchy is not explicit in the graphs realized from KPGM because all of the trials at all scales are performed *independently*, or in other words, all of the edges are untied at all scales. We propose a model where the trials have a hierarchical structure as well, leading to a higher grouping of edges and a higher variance in the number of edges. In this model, the edges are tied at *all* common scales. For the KPGMs, the adjacency matrix A is obtained from the edge probability matrix $\mathcal{P}_K$, $A = R(\mathcal{P}_K) = R(\mathcal{P}_1 \otimes \ldots \otimes \mathcal{P}_1)$. Instead, we propose to realize an adjacency matrix *after each Kronecker multiplication*. We denote by $R_t(\mathcal{P}_1, K)$ a realization of this new model with the initiator $\mathcal{P}_1$ and K scales. We define R_t recursively, $R_t(\mathcal{P}_1, 1) = R(\mathcal{P}_1)$, and $R_t(\mathcal{P}_1, K) = R_t(R_t(\mathcal{P}_1, K-1) \otimes \mathcal{P}_1)$. If unrolled,

$$A = R_t(\mathcal{P}_1, K) = \underbrace{R(\dots R(R(\mathcal{P}_1) \otimes \mathcal{P}_1)\dots)}_{K \text{ realizations of } R}.$$

Similar to section 5.1, we define the probability matrix for scale ℓ, $(\mathcal{P}_K)_\ell = \mathcal{P}_1$ for $\ell = 1$, and $(\mathcal{P}_K)_\ell = R_t\left((\mathcal{P}_K)_{\ell-1}\right) \otimes \mathcal{P}_1$ for $\ell \geq 2$. Under this model, at scale ℓ there are $b^\ell \times b^\ell$ independent Bernoulli trials rather than $b^K \times b^K$ as $(\mathcal{P}_K)_\ell$ is a $b^\ell \times b^\ell$ matrix. These $b^\ell \times b^\ell$ trials correspond to different *prefixes* of length ℓ for (u, v), with a prefix of length ℓ covering scales $1, \dots, \ell$. Denote these trials by $T^\ell_{u_1 \dots u_\ell, v_1 \dots v_\ell}$ for the entry (u', v') of $(\mathcal{P}_K)_\ell$, $u' = (u_1 \dots u_\ell)_b$, $v' = (v_1 \dots v_\ell)_b$. The set of all independent trials is then $T^1_{1,1}, T^1_{1,2}, \dots, T^1_{b,b}$, $T^2_{11,11}, \dots, T^2_{bb,bb}, \dots$, $T^K_{\underbrace{1\dots1}_{K}\underbrace{1\dots1}_{K}}, \dots, T^K_{\underbrace{b\dots b}_{K}\underbrace{b\dots b}_{K}}$. The probability of a success for a Bernoulli trial at a scale ℓ is determined by the entry of the $\mathcal{P}_1$ corresponding to the ℓ-th bits of u and v:

$$P\left(T^\ell_{u_1 \dots u_\ell, v_1 \dots v_\ell}\right) = \theta_{u_\ell v_\ell}.$$

One can construct A^ℓ, a realization of a matrix of probabilities at scale ℓ, from a $b^\ell \times b^\ell$ matrix T by setting $A^\ell_{uv} = T^\ell_{u_1 \dots u_\ell, v_1 \dots v_\ell}$ where $u = (u_1 \dots u_K)_b$, $v = (v_1 \dots v_K)_b$. The probability for an edge appearing in the graph is the same as under KPGM as

$$A_{uv} = \prod_{\ell=1}^{K} A^\ell_{uv} = \prod_{\ell=1}^{K} T^\ell_{u_1 \dots u_\ell, v_1 \dots v_\ell} = \prod_{\ell=1}^{K} \theta_{u_\ell v_\ell}.$$

Note that all of the pairs (u, v) that start with the same prefixes $(u_1 \dots u_\ell)$ in b-nary also share the same probabilities for A^ℓ_{uv}, $\ell = 1, \dots, K$. Under the proposed models trials for a given scale t are shared or tied for the same value of a given prefix. We thus refer to our proposed model as *tied KPGM* or *tKPGM* for short. See Figure 7(b) for an illustration.

Just as with KPGM, we can find the expected value and the variance of the number of edges under tKPGM. Since the marginal probabilities for edges (u, v) are the same as under KPGM, the expected value for the number of edges is unchanged, $E[|\mathbf{E}|] = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} E[A_{uv}] = S^K_\Theta$. The variance $Var(|\mathbf{E}|)$ can be derived recursively by conditioning on the trials with prefix of length $\ell = 1$:

$$\begin{aligned}
Var(|\mathbf{E}|) &= E_{P(A_{11},\dots,A_{bb})} Var(X_K | A_{11}, \dots, A_{bb}) + Var_{P(A_{11},\dots,A_{bb})} E[X_K | A_{11}, \dots, A_{bb}] \\
&= \sum_{i=1}^{b} \sum_{j=1}^{b} \left[\pi_{ij} Var(X_{K-1}) + \pi_{ij} E[X_{K-1}]^2 - (\pi_{ij} E[X_{K-1}])^2 \right] \\
&= S_\Theta \times Var(X_{K-1}) + (S_\Theta - S_{\Theta^2}) E[X_{K-1}]^2 \\
&= S_\Theta \times Var(|\mathbf{E}_{K-1}|) + (S_\Theta - S_{\Theta^2}) S^{2(K-1)}_\Theta.
\end{aligned}$$

where $|\mathbf{E}_{K-1}|$ corresponds to the number of edges generated by $K-1$ Kronecker multiplication rather than K ($|\mathbf{E}_K| = |\mathbf{E}|$). Using $Var(|\mathbf{E}_1|) = S_\Theta - S_{\Theta^2}$. The solution to this recursion is

$$Var(|\mathbf{E}|) = S^{K-1}_\Theta \left(S^K_\Theta - 1\right) \frac{S_\Theta - S_{\Theta^2}}{S_\Theta - 1}. \tag{7}$$

5.3. Mixed KPGM

Even though tKPGM provides a natural mechanism for clustering the edges and for increasing the variance in the graph statistics, the resulting graphs exhibit *too much*

variance. One of the possible reasons is that the edge clustering and the corresponding Bernoulli trial tieing should not begin at the highest shared scale (to model real-world networks). To account for this, we introduce a modification to tKPGM that ties the trials starting with prefix of length $\ell + 1$, and leaves the first ℓ scales untied. Since this model will combine or *mix* the KPGM with tKPGM, we refer to it as ***mKPGM***. Note that mKPGM is a generalization of both KPGM ($\ell = K$) and tKPGM ($\ell = 1$). The effect of tieing can be seen in Figure 8—the graph sampled from KPGM exhibits little grouping of the edges, the graph sampled from tKPGM exhibits strong grouping, and the graph sampled from mKPGM falls in between the other two. How close would the properties of a graph from mKPGM resemble one of the other two depends on the proportion of untied scales.

Formally, we can define the generative mechanism in terms of realizations. Denote by $R_m(\mathcal{P}_1, K, \ell)$ a realization of ***mKPGM*** with the initiator $\mathcal{P}_1$, K scales in total, and ℓ untied scales. Then $R_m(\mathcal{P}_1, K, \ell)$ can be defined recursively as $R_m(\mathcal{P}_1, K, \ell) = R(\mathcal{P}_K)$ if $K \leq \ell$, and $R_m(\mathcal{P}_1, K, \ell) = R_t(R_m(\mathcal{P}_1, K-1, \ell) \otimes \mathcal{P}_1)$ if $K > \ell$. Scales $1, \ldots, \ell$ will require $b^\ell \times b^\ell$ Bernoulli trials each, while a scale $s \in \{\ell+1, \ldots, K\}$ will require $b^s \times b^s$ trials. See Figure 7(c) for an illustration.

Intuitively, the graph sampling mechanism under mKPGM can be viewed as generating a binary $b^\ell \times b^\ell$ matrix according to KPGM with $\mathcal{P}_\ell = \mathcal{P}_1^{[\ell]}$, and then for each success (1 in the matrix) generating a $b^{K-\ell} \times b^{K-\ell}$ matrix according to tKPGM with initiator $\mathcal{P}_1$ and $K-\ell$ scales. Failure (0) in the intermediate matrix results in a $b^{K-\ell} \times b^{K-\ell}$ matrix of zeros. These $b^{K-\ell} \times b^{K-\ell}$ then serve as submatrices of the realized $b^K \times b^K$ adjacency matrix.

Since the marginal probabilities for edges are unchanged, $P(A_{uv}) = \pi_{uv} = \prod_{l=1}^{K} \theta_{u_l v_l}$, the expected value for the number of edges is unchanged as well, $E[|\mathbf{E}|] = S_\Theta^K$. However, the variance expression is different from that in (7), and it can be obtained conditioning on the Bernoulli trials of the ℓ highest order scales:

$$Var(|\mathbf{E}|) = S_\Theta^{K-1}\left(S_\Theta^{K-\ell} - 1\right)\frac{S_\Theta - S_{\Theta^2}}{S_\Theta - 1} + \left(S_\Theta^\ell - S_{\Theta^2}^\ell\right) S_\Theta^{2(K-\ell)}. \tag{8}$$

Note that for $\ell = 1$, the variance is equal to $S_\Theta^{K-1}\left(S_\Theta^K - 1\right)\frac{S_\Theta - S_{\Theta^2}}{S_\Theta - 1}$, the same as for tKPGM, and the variance of mKPGM is smaller than that of tKPGM for $\ell > 1$. When $\ell = K$, the variance is equal to $S_\Theta^K - S_{\Theta^2}^K$, the same as for KPGM, and the variance of mKPGM is greater than that of KPGM for $\ell < K$.

From another point of view, given Θ, K, and $\ell \in [1, \cdots, K]$ the number of untied levels, the mKPGM generation process samples a network of size b^K as follows. First, the model uses a KPGM model with parameters Θ and ℓ to calculate a probability matrix P_ℓ and sample a graph G_ℓ. Then, a subsequent Kronecker product, $G_\ell \otimes \Theta$, is realized to obtain a new probability matrix $P_{\ell+1}$. To tie the parameters, a graph $G_{\ell+1}$ is sampled from $P_{\ell+1}$ before any further Kronecker products. This process is then repeated $K - \ell - 1$ times to generate the final network $G = (\mathbf{V}, \mathbf{E})$.

Considering this generation process, the running time of a mKPGM network is separated in the generation of the KPGM network $G_\ell = (\mathbf{V}_\ell, \mathbf{E}_\ell)$ (with b^ℓ nodes and S_Θ^ℓ edges) and the generation of the $K - \ell$ tied levels for each edge of the set $\mathbf{E}_\ell$. The running time for KPGM is $O(|\mathbf{E}_\ell|)$ [Leskovec and Faloutsos 2007], while the running time of one subnetwork with $K - \ell$ tied levels corresponds to the total number of bernoulli trial for the generation. Considering that the expected number of edges for the level $i - 1$ is given by S_Θ^{i-1}, and that for each edge b^2 bernoulli trials are realized (one for

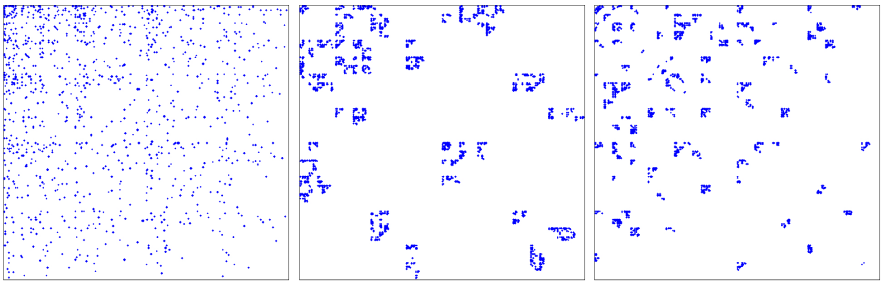
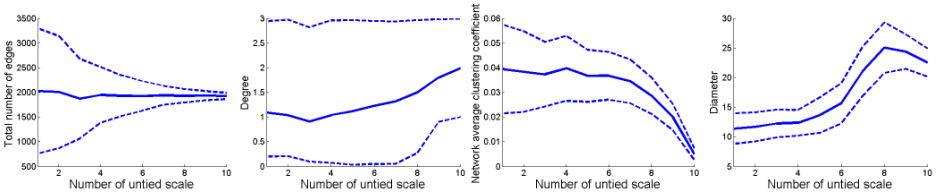


Fig. 8. Generated networks of 2^8 nodes for different Kronecker product graph models: KPGM (left), tKPGM (center), mKPGM (right). For mKPGM the number of untied scales was 5.



(a) Total Edges (b) Degree (c) Cluster Coefficient (d) Diameter

Fig. 9. Mean value (solid) $\pm$ one sd (dashed) of characteristics of graphs sampled from mKPGM as a function of l, number of untied scales.

each each parameter of Θ), the total number of trials is given by:

$$\sum_{i=1}^{K-\ell} b^2 S_\Theta^{i-1} = b^2 \sum_{i=0}^{K-\ell-1} S_\Theta^i = b^2 \frac{1 - S_\Theta^{K-\ell}}{1 - S_\Theta}$$

considering each trial as $O(1)$, the running time for the generation of $K - \ell$ tied levels is $O(b^2 S_\Theta^{K-\ell})$. Multiplying this value for the number of edges of G_ℓ, the final running time for the entire network is $O(S_\Theta^\ell + S_\Theta^\ell b^2 S_\Theta^{K-\ell}) \approx O(b^2 S_\Theta^K) \approx O(|\mathbf{E}|)$.

5.4. Empirical Simulations

We perform two shorts empirical analysis of mKPGMs using simulations. Figure 9 shows the variability over four different graph characteristics, calculated over 300 sampled networks of size 2^{10} ($b = 2$ and $K = 10$) with $\Theta = \begin{bmatrix} 0.99 & 0.20 \\ 0.20 & 0.77 \end{bmatrix}$, for $\ell = \{1, .., 10\}$. In each plot, the solid line represents the mean of the analysis (median for plot (b)) while the dashed line correspond to the mean plus/minus one standard deviation (first and third quartile for plot(b)).

In Figure 9(a), we can see that the total number of edges does not change significantly with the value of ℓ, however the variance of this characteristic decreases for higher values of ℓ, confirming that the KPGM ($\ell = 10$) has the lowest variance of all.

Figure 9(b) shows how the median degree of a node increases proportionally to the value of ℓ. Also, considering that the number of nodes in the network remains constant for all values of ℓ, it is clear that as ℓ increases the edges are assigned more uniformly throughout the nodes compared to the tKPGM ($\ell = 1$)—where some nodes get the majority of the edges.

The Network average cluster coefficient presented in plot (c) of the same figure, demonstrates how the clustering coefficient of the network and its respective variance decrease as ℓ increase. Considering that KPGM has a uniformly repartition of the edges through the nodes, there is a small probability that three nodes are connected

among them which reduced the probability to get a higher clustering coefficient. On the other hand, the sampling process of tKPGM allows, with a higher probability, that some groups of nodes will be connected among them increasing the clustering coefficient of the network.

In the final plot (d) can be observed that the diameter increases with the value of ℓ. The uniform assignment over different nodes in the KPGM model produces large chains of nodes that are not connected between them, generating a huge diameter for the network. On the other hand, how tKPGM produces different groups of nodes connected among them the diameter is just incremented by the connection between these groups.

The clustering coefficient and diameter phenomena explained before can be observed in figure 8 where in the left plot (KPGM) the edges are distributed over all the nodes, producing a higher diameter but a small clustering coefficient. In the middle plot tKPGM has group of nodes with several edges among them increasing the clustering coefficient value however there is few edges connecting these groups. Finally the right plot shows how mKPGM produces some groups of nodes and at the same time some connection between these groups.

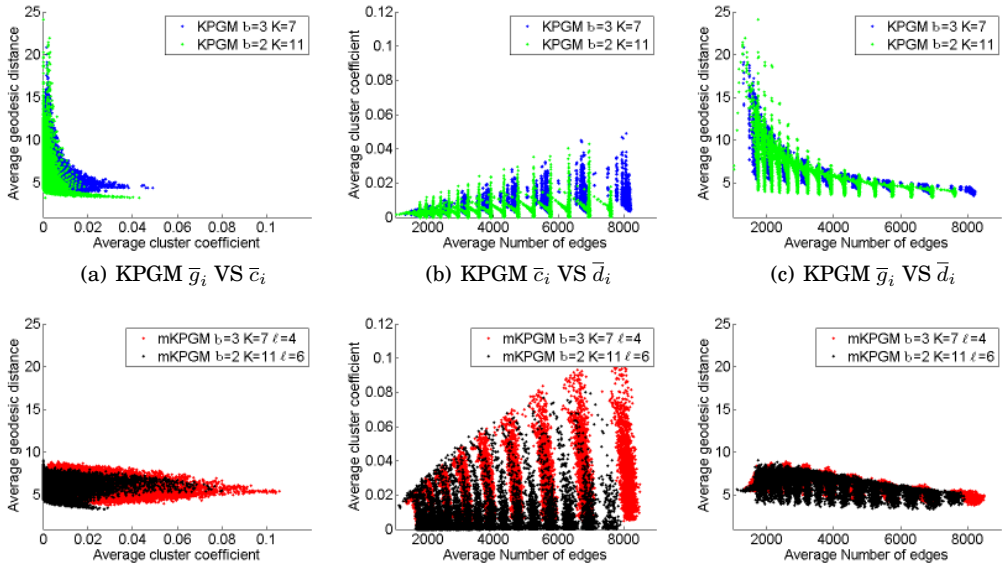
The second empirical simulation explores the advantages of the mKPGM representation over KPGM, as well, as the utility of a larger initiator matrix. We generated networks over a wide range of parameter values in Θ and measured the characteristics of the resulting graphs. Specifically, we considered 9239 different values of Θ for initial matrices of size $b = 2$. For each Θ setting, we generated 50 networks with $K = 11$ from both the KPGM ($\ell = K$) and the mKPGM ($\ell = 6$). This resulted in networks with 3500-15200 edges. We also generated the same quantity of networks, with similar number of nodes, edges and equivalent levels of parameter tying for initial matrices of size $b = 3$. For $b = 3$, we considered 18816 different values of Θ and generated networks with $K = 7$ from KPGMs ($\ell = K$) and from mKPGM ($\ell = 4$). This resulted in networks with 3400-16300 edges. From the set of networks generated by each parameter setting, we calculated three moments for each graph i: (i) number of edges $(\overline{d}_i)$, (ii) average clustering coefficient $(\overline{c}_i)$, and (iii) average geodesic distance $(\overline{g}_i)$. We plot the observed moments in Figure 10.

Figure 10 first column shows that mKPGM (d) is capable of generating higher clustering coefficient than the KPGM (a), obtaining the highest clustering coefficient with $b = 3$. However, the KPGM is capable of producing larger geodesic distances. Figures (b) and (e) show that the average geodesic distance is inversely proportional to the number of edges for KPGM, which implies that the high geodesic distance is obtained via small numbers of edges connected in chain-like structures. Finally, figures (c) and (f) confirms that mKPGM can generate higher clustering coefficient than KPGM, and that mKPGMs can generate a wider range of networks with higher clustering coefficient using $b = 3$.

6. MKPGM ESTIMATION

This section explains the training algorithm of the mKPGM model. Lamentably, the training algorithm for KPGM can not be applied for mKPGM algorithm. However, considering the advantage of the permutation independent of the MoM training algorithm (subsection 3.1), we develop a new training algorithm which combines the simulated method of moments with a constrained line search in two dimensions (2D) to obtain the best set of parameters.

The empirical simulations indicate that the structure of tKPGMs (high clustering, small diameter) will be unlikely to capture the characteristics of real world networks, thus we are more interested in estimating the parameters for mKPGM models. We



(a) KPGM $\overline{g}_i$ VS $\overline{c}_i$ (b) KPGM $\overline{c}_i$ VS $\overline{d}_i$ (c) KPGM $\overline{g}_i$ VS $\overline{d}_i$

(d) mKPGM $\overline{g}_i$ VS $\overline{c}_i$ (e) mKPGM $\overline{c}_i$ VS $\overline{d}_i$ (f) mKPGM $\overline{g}_i$ VS $\overline{d}_i$

Fig. 10. Variation of graph properties for synthetic networks using generator matrix of size b=2 and b=3 for KPGM model.

conjecture that mKPGMs will be able to capture the variance and clustering of real world social networks more accurately than conventional KPGMs.

Even though the likelihood of mKPGMs is similar to the KPGM likelihood (Eq. 1), it can not be used to calculate the parameters of the model. The likelihood of the mKPGM model has two parts, which vary as a function of ℓ, the number of untied scales in the model. One part of the likelihood (ℓ levels) is based on the original KPGM. The other part of the likelihood ($K - \ell$ levels) is based on tKPGM. The likelihood of the observed graph $P\left(G^{\star}|\Theta,\sigma\right)$ for a mKPGM model is calculated as:

$$P(G^{\star}|\Theta,\sigma) = \prod_{(u,v)\in\mathbf{E}} \pi_{uv} \prod_{(u,v)\notin\mathbf{E}} (1-\pi_{uv}) = \prod_{(u,v)\in\mathbf{E}} \prod_{i=1}^{K} \theta_{u_i v_i} \prod_{(u,v)\notin\mathbf{E}} \left(1 - \prod_{i=1}^{K} \theta_{u_i v_i}\right)$$

$$= \prod_{(u,v)\in\mathbf{E}} \mathcal{P}_{\ell}[\sigma_u,\sigma_v] \prod_{i=1}^{K-\ell} \theta_{u_i v_i} \prod_{(u,v)\notin\mathbf{E}} \left(1 - \mathcal{P}_{\ell}[\sigma_u,\sigma_v] \prod_{i=1}^{K-\ell} \theta_{u_i v_i}\right)$$

One might think that the KPGM MLE estimation algorithm could easily be extended to estimate the parameters of mKPGMs. If the level of untied scales is known, then the algorithm can alternate sampling a permutations and estimating the parameters of the KPGM and tKPGM models. However, we found out that this estimation is not straightforward in this context, since local search (i.e., swap of single pair of nodes) in permutation space is extremely unlikely to discover the block structure that is necessary to accurately estimate mKPGMs and tKPGMs. In practice, we found that straightforward MLE estimation only works well when starting from very close to the true permutation—which will not work for real datasets.

Discarding MLE estimation, we searched for a training method which is independent of the position of the nodes in the network. Based on the work [Gleich and Owen

2012], we tried to implement the method of moments (MoM) for mKPGM. However, because of the complex relationship between the edges in mKPGMs, it is difficult to derive analytical expressions for even simple moments. In particular, consider the case where $A_{ij} = 1$ if there is an edge between nodes i and j, and 0 otherwise. Then $E[A_{ij}A_{kl}] \neq E[A_{ij}]E[A_{kl}]$ for nodes that have common parameters in the network generation of $G_{K-\ell}$. This makes it difficult to directly minimize $f(\Theta, \mathbf{F}(G^*))$ in Eq. 2.

Even though, a direct implementation of MoM is not possible for mKPGMs, since the MoM training is permutation independent, it successfully avoids the difficult search over the factorial permutation space. To exploit this, we developed a *simulated method of moments* (SMM) approach that approximates the objective function in Eq. 2 with empirically estimated moments. Simulated method of moments (see e.g., [Pakes and Pollard 1989]) is often used to estimate models where the moments are complicated functions that cannot easily be evaluated analytically (e.g., in econometric models). In SMM methods, simulation experiments are used to empirically estimate the moments and/or their derivatives.

In our SMM method, we replace the analytical expression of the moment i given by $E[F_i(G)|\Theta]$ with an empirical estimation $\widehat{E}[F_i(G)|\Theta]$ based on simulation of networks from Θ. In the function *estObjFunc* below, we show how to estimate $f(\Theta, \mathbf{F}(G^*))$ empirically. The function estimates the error between $\mathbf{F}(G^*)$ and the estimate $\widehat{E}[\mathbf{F}(G)|\Theta]$ calculated empirically via SMM, by averaging the moments observed in S sampled networks. Each network $G(i)$ ($i \in \{1, \cdots, S\}$) is generated with the mKPGM algorithm using Θ, K, and ℓ. For each $G(i)$, the vector of moments $\mathbf{F}(G(i))$ is obtained and the estimation of the expected moments is calculated by $\widehat{E}[\mathbf{F}(G)|\Theta] = \frac{1}{S}\sum_{i=1}^{S}\mathbf{F}(G(i))$. Finally, the value of the objective function is calculated by Eq. 2. The pseudocode for this function is provided in Algorithm 1.

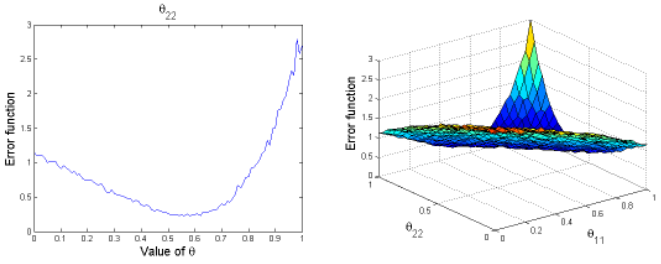
ALGORITHM 1: Function *estObjFunc*

Require: Θ, K, ℓ, S, $\mathbf{F}^*$

1: **for** $i = 1; i++; i \leq S$ **do**
2: Generate $G(i)$ mKPGM network using (Θ, K, ℓ)
3: Calculate moments $\mathbf{F}(G(i))$.
4: **end for**
5: $\widehat{E}[\mathbf{F}(G)|\Theta] = \dfrac{1}{S}\displaystyle\sum_{i=1}^{S}\mathbf{F}(G(i))$
6: **return** $\sum_{i=1}^{|\mathbf{F}(G)|}\left(\frac{F_i(G^*)-\widehat{E}[F_i(G)|\Theta]}{F_i(G^*)}\right)$

The calculation of the objective function through SMM avoids the complexities of determining an analytical expression for the moment, and facilitates the inclusion of a wider range of moments in the learning algorithm. However, since the objective function is not convex, we still need to determine a way to search over the parameter space to minimize $f(\Theta, \mathbf{F}(G^*))$. Moreover, without a closed form expression for the moments, we can not estimates their gradients to implement a gradient descent-type optimization. To address this, we develop a line search optimization method, where the moments are empirically estimated by SMM.

To investigate whether linear search is a promising direction to pursue, we explored whether the objective function is locally convex. We considered each parameter θ_{ij}, while keeping the rest of parameters constant, and evaluated $f(\Theta, \mathbf{F}(G^*))$ for different values of θ_{ij} (see Figure 11(a)). In all cases, we observed a locally convex error function with a minimum around the value of θ_{ij} such that expected number of edges is



(a) 2D error (b) 3D error

Fig. 11. Error function with respect to the variation of one (left) and two (right) parameters.

approximate to the number of edges of the training network ($S_\Theta^K \approx |\mathbf{E}^*|$). Then, the minimum is founded when θ_{ij} makes S_Θ^K around the number of edges of the training network. This implies that one dimensional linear search (i.e., changing a single parameter) will not be able to easily explore the parameter space once it reaches a local minima that matches $|\mathbf{E}^*|$ (since changing a single parameter in isolation will always affect the expected number of edges). However, a two dimensional (2D) search, where we keep S_Θ^K constant, could improve $f(\Theta, \mathbf{F}(G^*))$ for other moments.

To explore this, we considered each combination of two parameters in Θ and evaluated $f(\Theta, \mathbf{F}(G^*))$ for different values of the parameters, while keeping the rest of parameters constant (see Figure 11(b)). Again, in all cases, the 2D curve was locally convex. Thus, knowing that the 2D parameter space is likely to be locally convex we will approximate a full search over the parameter space by initializing the parameters such that $S_\Theta^K = |\mathbf{E}^*|$ and performing a linear search in two dimensions, while constraining the parameters to match $S_\Theta^K = |\mathbf{E}^*|$. This will help the algorithm from getting trapped in local minima.

6.1. mKPGM Learning Algorithm

The training algorithm for mKPGM realizes a constraint linear search over possible combinations of two parameters to find the best set of parameters that can reproduce the desires moments. The moments are approximated by $\widehat{E}[\mathbf{F}(G)|\Theta]$ using the SMM. The pseudocode for this function is provided in Algorithm 2.

The algorithm first calculates the required number of Kronecker multiplications through $K = \left\lceil \frac{\log(|V^*|)}{\log(b)} \right\rceil$, where $|V^*|$ is the number of nodes of the training network $G^\star = (V^*, E^*)$ and $b \times b$ is the size of the parameter matrix Θ. It continues with the initialization of each parameter Θ by $\forall i, j\ \theta_{ij} = \sqrt[K]{|\mathbf{E}^*|}/b^2$, which ensures the constraint $S_\Theta^K = |\mathbf{E}^*|$ explained before. With the initial set of parameters Θ, the initial error of the moments EF^* is calculated by *estObjFunc*. The algorithm continues with the generation of the set of $N_c = \binom{b^2}{2}$ possible pairs of parameters to consider in the 2D search ($\Theta_{pairs} = \{(11, 12), \cdots, (b(b-1), bb)\}$, where $\Theta_{pairs}(i) = (ij, kl)$ corresponds to the two indexes of the i-th element of the set). In case of an undirected networks the set Θ_{pairs} is reduced to $N_c = \binom{b(b+1)}{2}$ elements, due to the symmetric relationship in Θ.

Once that the EF^* and Θ_{pairs} are calculated, the algorithm initializes the search over the parameter space. The algorithm consists in three loops: The first loop iterates over step sizes δ, which determines the changes of parameters values θ_{ij} ($\theta_{ij} \pm \delta$). The second loop iterates over the set Θ_{pairs}, where in each iteration two indexes are selected determining the part of space that is searched. The two indexes of the selected parameters are given by the pair $index = \Theta_{pairs}(mod(j, N_c) + 1)$, where $index(1)$ and $index(2)$ correspond to the parameters indexed by the first and second element of $index$

ALGORITHM 2: mKPGM training algorithm

Require: $G^\star = (\mathbf{V}^*, \mathbf{E}^*)$, b, δ, ℓ, $iter$, S
Calculate the moments $\mathbf{F}(G^*)$ for $G^\star$
$K = \left\lceil \frac{\log(|V^*|)}{\log(b)} \right\rceil$
Initialize $\forall\ i, j\ \theta_{ij} = \sqrt[K]{|\mathbf{E}^*|}/b^2$
$EF^* = esObjFunc(\Theta, K, \ell, S, \mathbf{F}(G^*))$
$N_c = \binom{b^2}{2}$ {combinations of 2 parameters}
Let $\Theta_{pairs} = \{(11, 12), \cdots, (b(b-1), bb)\}$
for $i = 1; i++; i \leq iter$ **do**
  $j = 0$
  $idx = N_c$
  while $j < idx$ **do**
    $index = \Theta_{pairs}(mod(j, N_c) + 1)$
    for $k = -3; k++; k \leq 3$ **do**
      $\Omega = \Theta$
      $\Omega_{index(1)} = \Omega_{index(1)} + k * \delta$
      $\Omega_{index(2)} = \Omega_{index(2)} - k * \delta$
      if $(0 \leq \Omega_{index(1)}, \Omega_{index(2)} \leq 1)$ **then**
        $EF' = estError(\Omega, K, \ell, S, \mathbf{F}(G^*))$
        if $EF' < EF^*$ **then**
          $EF^* = EF'$
          $\Theta = \Omega$
          $idx = idx + mod(j, N_c) + 1$
          {Search is extended for the next N_c iterations}
        end if
      end if
    end for
    $j++$
  end while
  $\delta = \delta/2$
end for
return Θ

respectively (for example if $\Theta_{pairs}(1) = (11, 12)$, then $index(1) = 11$ modifying θ_{11} and $index(2) = 12$ modifying θ_{12}). The third loop (over k), implements the restricted linear search, by iterating from -3δ to 3δ with a step size of δ. The loop begins with a copy of the original set of parameters ($\Omega = \Theta$), then two parameters of Ω are modified according to $\Omega_{index(1)} = \Omega_{index(1)} + k$ and $\Omega_{index(2)} = \Omega_{index(2)} - k$. This modification searches over the two dimensional parameter space while constraining $S_\Theta^K = |\mathbf{E}^*|$. If $\forall\ i, j\ 0 \leq \Omega_{ij} \leq 1$, then the moment error EF' for the new set of parameter Ω is calculated by *estObjFunc*. If the new moment error is lower than the minimum error ($EF' < EF^*$), then Ω is accepted, the error is updated, and the search extended for the next N_c iterations.

The running time of the mKPGM training algorithm is $O(C|\mathbf{E}|)$, where C is at least $iter * N_c * 7 * S$. This value is calculated from the inner loop (*estObjFunc* function) to the outer loop (the number of iteration for δ loop). The running time of the function *estObjFunc* is given by $O(S(|\mathbf{E}| + |\mathbf{V}| + |\mathbf{E}|)) \approx O(S|\mathbf{E}|)$. This value corresponds to the generation and analysis of S networks, where the generation time of mKPGM algorithm is dominated by $O(|\mathbf{E}|)$ (subsection 5.3), and the moment estimation is dominated by $O(|\mathbf{E}|+|\mathbf{V}|)$ (the running time of a breadth-first search algorithm to calculate the geodesic distance for a sample of nodes). The range of three main loops contribute the other values of C.

Our proposed training method has three important advantages in comparison to the previous MoM learning method for KPGMs. First, our algorithm is not limited to the specific moments considered in the original MoM method. The only consideration for including additional moments involve the time complexity for calculating them empirically in network samples. In particular, we considers five moments in our training algorithm: (i) average number of edges, (ii) average cluster coefficient, (iii) average geodesic distance (approximated by a sample of nodes rather than all pairs), (iv) size of the largest connected component, and (v) number of nodes with degree greater than zero (to solve the problem that Kronecker models often generate isolated nodes [Seshadhri et al. 2011]). The second advantage is that the SMM method facilitates application of the training algorithm to any size of the initial generator matrix (i.e., $b \geq 2$). Third, our training algorithm is not limited to undirected networks, since the SMM approach can handle the complexity of directed networks.

7. EXPERIMENTS

This section describes new single networks datasets utilized in the paper and a new method of evaluation of the results based on the Kolgomorov Smirnov test.

7.1. Datasets

Besides the three datasets describes in section 2, we used three others single networks to evaluate the training performance in single networks. The three single network data were obtained from the Stanford Network Analysis Project.[3] These single networks were model by mKPGM with $\ell = K$ to compare our learning algorithm against KPGM training methods. The first network is the Gnutella peer-to-peer network (Nutella), which is a sequence of snapshots of the Gnutella peer-to-peer file sharing network from August 2002 with 6,301 nodes and 20,777 edges. The second dataset is the Arxiv General Relativity and Quantum Cosmology (GRQC) collaboration network, where each of the 5,242 nodes represent authors, and the 28,980 edges indicates a publication between two authors. The last dataset is the CAIDA AS Relationships Datasets (CAIDA) from December 2004 with 18,501 nodes and 76,530 edges.

7.2. Methodology

We aim to capture three specific characteristics of real network datasets: degree, clustering coefficient, and hop plot (described in section 2). To measure these characteristics, we compare the cumulative distribution functions (CDFs) of the three characteristics. The CDF provides a more complete description of the network structure compared to a single aggregate statistic (e.g., average degree). Specifically, while some generated networks may easily match the average of a particular characteristics, it is not as simple to match the entire distribution in the original network.

To quantitatively measure the differences between the CDFs observed in the training data and those of the generated networks, we outline a new test measure– the 3-dimensional Kolmogorov-Smirnov distance (KS_{3D}). The Kolmogorov-Smirnov (KS) test measure defines the difference between two cumulative distribution as the maximum value of the absolute difference between the two distributions $KS(CDF_1, CDF_2) = max_x|CDF_1(x) - CDF_2(x)|$. This distance, which varies between 0 and 1, is utilized to define how similar are the two distributions, with zero indicating a perfect match. KS distance is commonly used to evaluate the characteristics of graph (see e.g.,[Ahmed et al. 2011]). However, to assess whether the model matches *multiple* characteristics simultaneously, researchers considered each distribution independently, even though there are clear dependencies among graph measures. This

[3]http://snap.stanford.edu/data/

could lead to a perfect match of each distribution by separate but a completely different distribution when two or more variables are considered at the same time.

To address this issue, we outline the KS_{3D} test measure, which captures the correlation among graph measures. The KS_{3D} test measure corresponds to the maximum difference between two discrete cumulative distributions in 3 dimensional space (3D). To construct the 3D-CDF for each network, we represent every node i as a 3D point $Po_i =< d_i, c_i, g_i >$ where d_i stands for degree, c_i for clustering coefficient, and g_i is the average geodesic distance of paths from node i. We then calculate the maximum percentage difference between two distributions. Specifically, given two graphs $G(1) = (\mathbf{V}(1), \mathbf{E}(1))$ and $G(2) = (\mathbf{V}(2), \mathbf{E}(2))$, where $\mathbf{V}(i)$ and $\mathbf{E}(i)$ represent the set of nodes and edges respectively, the KS_{3D} test measure is defined by:

$$KS_{3D}(G(1), G(2)) = max_{Po_x} |cp_1(Po_x) - cp_2(Po_x)| \tag{9}$$

where $cp_i(Po_x)$ represents the percentage of points from network $G(i)$ that are lower or equal than the point Po_x in all dimensions. The KS_{3D} test measure varies between 0 and 1, with 0 indicating a perfect match between the two distributions. Although we describe the distance based on three dimensions, generalization to higher dimensions is straightforward.

The code for the KS_{3D} test measure is presented in Algorithm 3. Given $G(1) = (\mathbf{V}(1), \mathbf{E}(1))$ and $G(2) = (\mathbf{V}(2), \mathbf{E}(2))$, the set $\mathbf{V} = \mathbf{V}(1) \cup \mathbf{V}(2)$ is defined. For every node i in $\mathbf{V}$, we use Po_i to calculate cp_1 and cp_2 for $G(1)$ and $G(2)$, if the absolute difference between them is greater than the maximum distance known, we save the new difference and continue with the next point. Once that all points are considered, the maximum distance is return. If $|\mathbf{V}|$ is too large to calculate the KS_{3D} test measure, it can be approximated through the use of a 3D-grid. In this particular case, Eq. 9 is rewritten by $KS_{3D}(G(1), G(2)) = max_x |cp_1(x) - cp_2(x)|$, where x is a point of the 3D-grid.

ALGORITHM 3: KS_{3D} test measure algorithm

Require: $G(1) = (\mathbf{V}(1), \mathbf{E}(1))$, $G(2) = (\mathbf{V}(2), \mathbf{E}(2))$
Let $\mathbf{V} = \mathbf{V}(1) \cup \mathbf{V}(2)$
$maxDist = 0$
for node i in $\mathbf{V}$ **do**
  Let $Po_i =< d_i, c_i, g_i >$ {degree, clustering coefficient and geodesic distance of node i}
  for $j = 1; j++; j <= 2$ **do**
    $cp_j = 0$ {Proportion of points lower than Po_i in $G(j)$}
    for node k in $\mathbf{V}(j)$ **do**
      Let $Po_k =< d_k, c_k, g_k >$
      if $d_k \leq d_i$ **and** $c_k \leq c_i$ **and** $g_k \leq g_i$ **then**
        cp_j++
      end if
    end for
    $cp_j = cp_j / |\mathbf{V}(j)|$
  end for
  if $maxDist < |cp_1 - cp_2|$ **then**
    $maxDist = |cp_1 - cp_2|$
  end if
end for
return $maxDist$

Dataset	Real data	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$
Synthetic	1277	-	2529	**1107**	-
Facebook	323	-	-	536	**291**
Email	765	-	1056	**483**	-
AddHealth	2491	**2826**	1789	-	-

Fig. 12. Standard deviation of the number of edges, for real data and mKPGM algorithm.

Θ_{orig}

$$\begin{bmatrix} 0.90 & 0.50 & 0.10 \\ - & 0.90 & 0.10 \\ - & - & 0.70 \end{bmatrix}$$

Θ_{mKPGM}

$$\begin{bmatrix} 0.77 & 0.02 & 0.17 \\ - & 0.54 & 0.64 \\ - & - & 0.92 \end{bmatrix}$$

$\Theta_{KPGM-MLE}$

$$\begin{bmatrix} 0.83 & 0.18 & 0.82 \\ - & 0.76 & 0.12 \\ - & - & 0.08 \end{bmatrix}$$

$\Theta_{KPGM-MoM}$

$$\begin{bmatrix} 1.00 & 0.43 \\ & 0.38 \end{bmatrix}$$

Fig. 13. Original and learned parameters for Kronecker algorithms in synthetic data.

8. RESULTS

We evaluated our training algorithm over one synthetic dataset and six real datasets. For each dataset, we selected a single network to use as a training set. To control for variation in the samples, we selected the network that was closest to the median of the degree distribution. For the synthetic data, we selected the network 27 (13460 edges), for Facebook data we selected the network 11 (5634 edges), and for Email data the network 24 (6666 edges). For AddHealth, we selected the network from school 72 with 2045 nodes and 15484 edges, which is the closest network to $3^7 = 2187$ nodes. The other real datasets have only a single network.

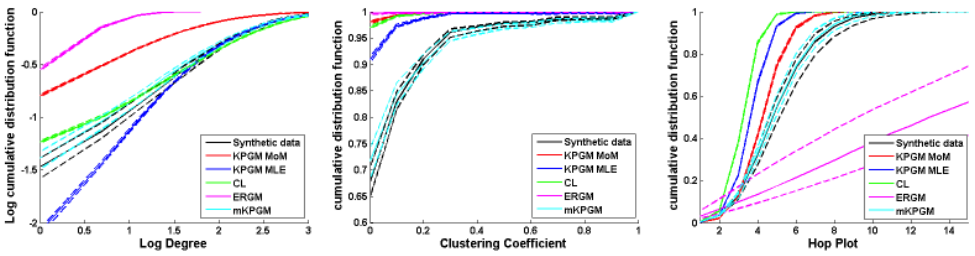
To determine the best value of ℓ, we compared the standard deviation of the number of edges observed in the real data against the standard deviation generated by the learned parameters. We chose the value of ℓ to be the value with closest match to the real data. The standard deviation for the number of edges and the selected values of ℓ (in **bold**) are included in figure 12.

Using each selected network as a training set, we learned each of the described models. For mKPGM and KPGM-MLE we used $b = 3$; for KPGM-MoM we used $b = 2$ since the algorithm is specific to that size of initial matrix. For mKPGMs, we used $\delta = 0.15, iter = 9,$ and $S = 50$. From each learned model, we generated 50 sample graphs. From these samples, we estimated the empirical sampling distributions for degree, clustering coefficient, and hop plots. The results are plotted in figures 14-17 for datasets with multiple network and in figures 20-22 for single networks. The plots show the median and interquartile range for the set of observed network distributions. Solid lines correspond to the median of the distributions; dashed lines: the 25th and 75th percentiles.

8.1. Synthetic Data

The synthetic data experiment is intended to evaluate whether our proposed mKPGM estimation algorithm is able to learn parameters successfully in networks generated from mKPGM distributions. The results for all models can be observed in Figure 14, while the learned parameters, for the methods based on Kronecker multiplication, can be observed in Figure 13.

According to Figure 13, KPGM-MLE does not learn the original parameters, this is also reflected in the generated networks that do not match the original networks (Figure 14). Even though our mKPGM training algorithm does not recover the exact parameters, it can emulate the properties of the original synthetic dataset—which can be seen in both the visual comparison in the figures and the small KS_{3D} error (figure 19(a), first column). We note that models are not identifiable when MoM is used



(a) Degree (b) Clustering Coefficient (c) Hop Plot

Fig. 14. Variation of graph properties in generated synthetic networks.

with fewer moments than the number of model parameters. In our experiments, we used five moments to estimate six parameters, and combined this with constrained search—with quite accurate results. In future work, we will explore whether additional moments can be used to improve estimation without incurring additional computational costs.

The only method able to model the observed variance in the synthetic networks is mKPGM. The lack of variance and huge error for ERGM may be due to degeneracy problems, even though we used some prescribed solutions to avoid it. We expected that KPGMs would generated graphs with less variance than mKPGM, and this is confirmed by the results. Finally, the lack of variance in the Chung-Lu graphs can also be explained by the models assumption of edge independence. The variance of the number of edges for a graph $G = (\mathbf{V}, \mathbf{E})$, with adjacency matrix A, generated by the Chung-Lu model is:

$$Var\left(|\mathbf{E}|\right) = \sum_{i=1}^{N}\sum_{j=1}^{N} Var\left(A_{ij}\right) = \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j (1 - w_i w_j)$$

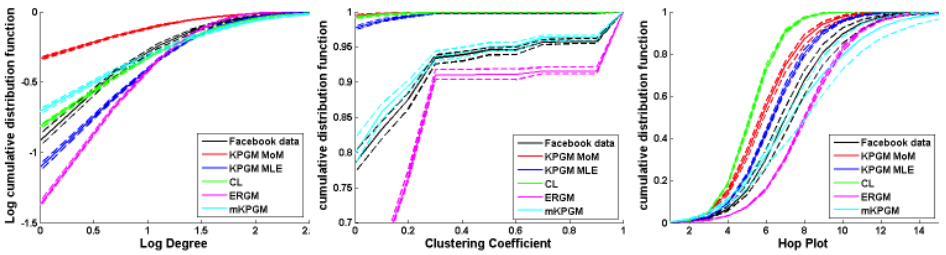
$$= \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j - \sum_{i=1}^{N}\sum_{j=1}^{N} w_i^2 w_j^2 = E[|\mathbf{E}|] - \sum_{i=1}^{N}\sum_{j=1}^{N} w_i^2 w_j^2$$

where w_i is the weight of node i (subsection 3.2). Given that $\sum_{i=1}^{N}\sum_{j=1}^{N} w_i^2 w_j^2 \geq 0$ then it is demonstrated that $\Rightarrow Var\left(|\mathbf{E}|\right) \leq E[|\mathbf{E}|]$.

The only model that can match all characteristics is mKPGM, this is not surprising since the data were generated with a mKPGM. The first set of columns in Figure 19(a) shows the KS_{3D} distance for the synthetic data. To avoid the effect of the variation in the networks, we selected the minimum KS_{3D} distance between the set of generated networks and $G^{\star}$. The results show that mKPGM is almost a perfect match with respect to the original data (represented by the low value), while the other models can not capture the joint characteristics of the synthetic data.

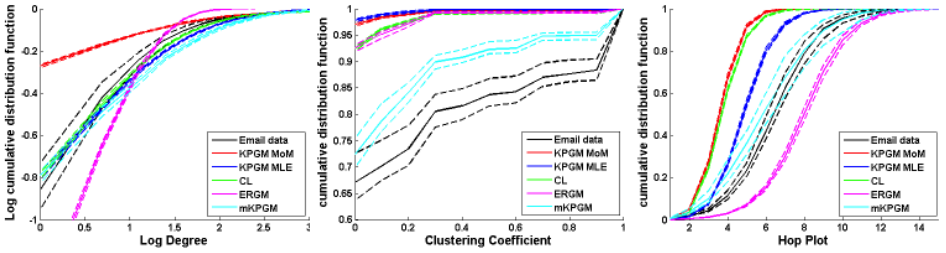
8.2. Real Data

Similar to the results of the synthetic data, the only model that can capture the variance observed in real networks is the mKPGM. The Chung-Lu model is the closest method to the median of the degree distribution in all datasets (Figures 15(a)-17(a)). However, the low variance of the method makes it difficult for it to match the entire degree distribution. The mKPGM is able of match not only the median of the distributions but also capture the variance of the degree distribution. On the other hand, KPGM-MLE is able to match part of the degree distribution but not the variance. KPGM-MoM and ERGM are not able to match any of the degree distributions.



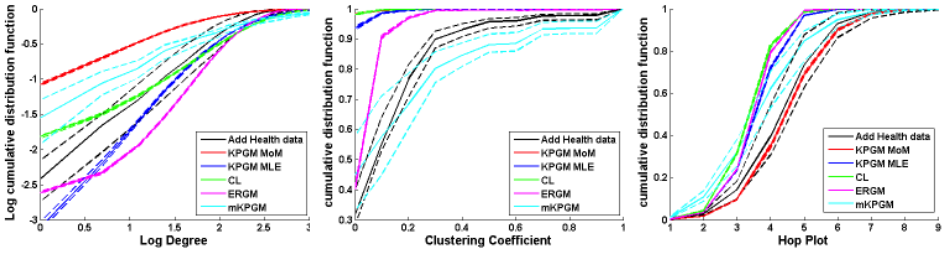
(a) Degree (b) Clustering Coefficient (c) Hop Plot

Fig. 15. Variation of graph properties in generated Facebook networks.



(a) Degree (b) Clustering Coefficient (c) Hop Plot

Fig. 16. Variation of graph properties in generated Email networks.



(a) Degree (b) Clustering Coefficient (c) Hop Plot

Fig. 17. Variation of graph properties in generated AddHealth networks.

KPGMs and Chung-Lu model generate networks with almost no clustering coefficient (Figures 15(b)-17(b)). In all the cases, over 90% percent of the nodes in these models have zero clustering coefficient. In contrast, our mKPGM training algorithm finds a good set of parameters for mKPGM, which are able to reproduce the clustering observed in social networks, resulting in an almost perfect match on the Facebook and AddHealth datasets. Lastly, ERGMs are not able to match any distribution over the datasets.

Besides being the only method which can model the variance in real networks, the mKPGM is the only method that can match all the Hop Plot distributions in the real data (Figures 15(c)-17(c)). In contrast, KPGMs and Chung-Lu underestimate the hop plot (except the KPGM-MoM on the Email data). These models are generating networks with a small diameter and at the same time without clustering coefficient—this does not reflect the properties of real social networks.

Figure 19(a) shows the KS_{3D} distance for the three real datasets. In all cases, the most accurate model is the mKPGM. With the exception of the Facebook data, where

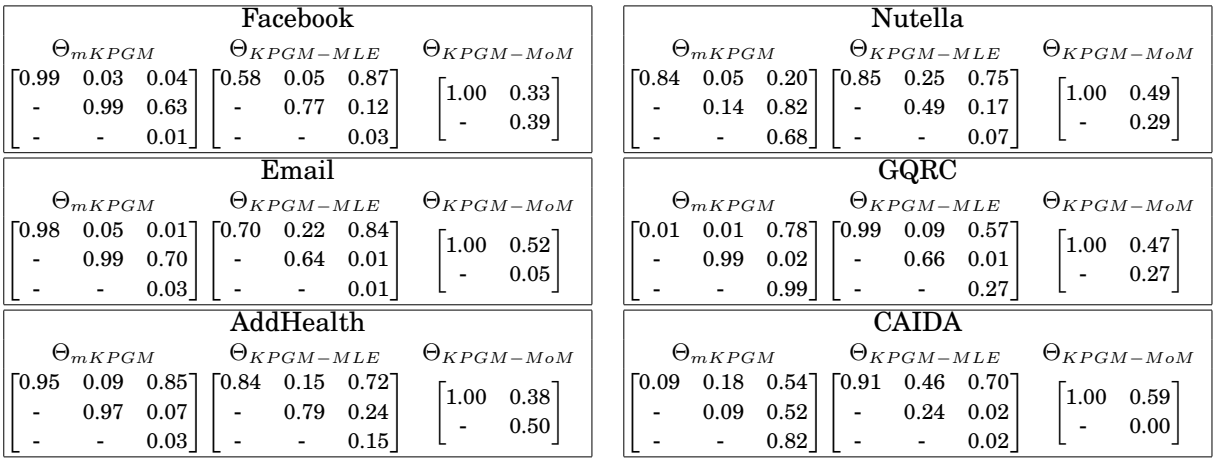


Fig. 18. Learned parameters for Kronecker algorithms. Left: Multiple networks datasets: Facebook, Email, and Addhealth. Right: Single networks datasets: Nutella, GRQC, and CAIDA.

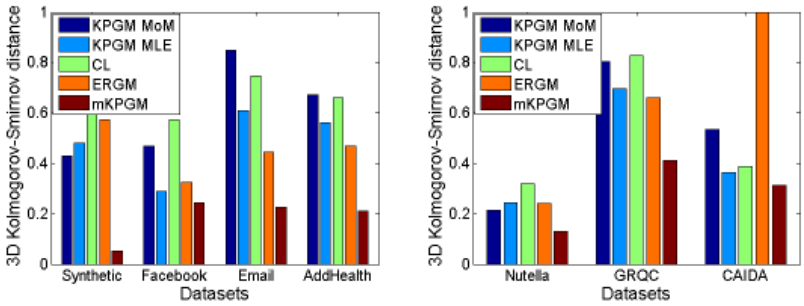
the mKPGM only exhibits a small improvement over KPGM-MLE and ERGM, the mKPGM achieves a large improvement over all the comparison statistical models. Surprisingly, one of the worst models is Chung-Lu, even though it explicitly replicates the degree distribution. KPGM-MoM performs worse than KPGM-MLE in all the cases, but this could be explained by its use of a smaller number of parameters (i.e., $b = 2$). Except for Facebook datasets, the KPGM-MLE does not adequately models any other datasets, confirming the limitations of the current representation and learning algorithms.

In Figure 18(left column), we report the learned parameters for KPGM and mKPGM models. The similarity, among the learned parameter for mKPGM, suggests the generation of two important groups of inter-connected nodes (high values in the main diagonal), which increase the clustering coefficient of the network, and a third group of nodes with a sparse connectivity (low value in the main diagonal). The connection among these groups, given by the upper triangles parameters, allow the connectivity to capture the geodesic distance. Most of the parameters learned by KPGM-MLE are middle-high or middle-low, generating a large connectivity between the nodes (which explains the underestimation of the hop plot distribution) and a low clustering coefficient. Similar to mKPGMs, the learned parameters for KPGM-MoM has high values in the main diagonal and median values in the second diagonal, however the limitation of the training algorithm and the number of parameters ($b = 2$) do not enable the KPGM-MoM to model the complexities of the real datasets.

In summary, our mKPGM training method is able to learn parameters from real data—in order to accurately capture not only the structural characteristics of the observed networks but also the variation in the network population.

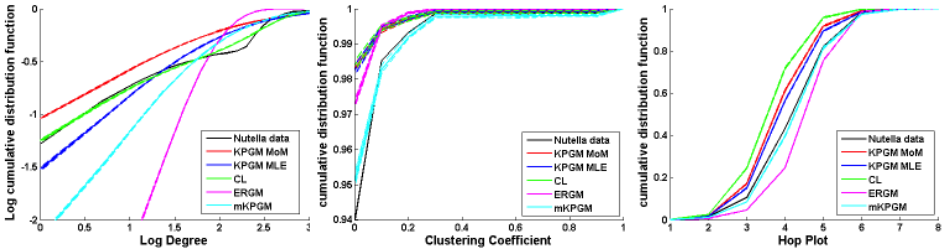
Given that KPGMs are a special case of mKPGMs (where $\ell = K$), we apply our training method to three individual network datasets and compare our learning algorithm with $\ell = K$, against the current KPGM training algorithms. Similar to the previous results, once we learned the models we generated 50 networks to compare against the real data. Lamentably, ERGM can not be tested in the largest dataset for memory problem, so these results were omitted.

Similar to the previous results, mKPGM is the best algorithm to model the data, which is confirmed by the KS_{3D} test measure. Figures 20(a)-22(a) show the degree



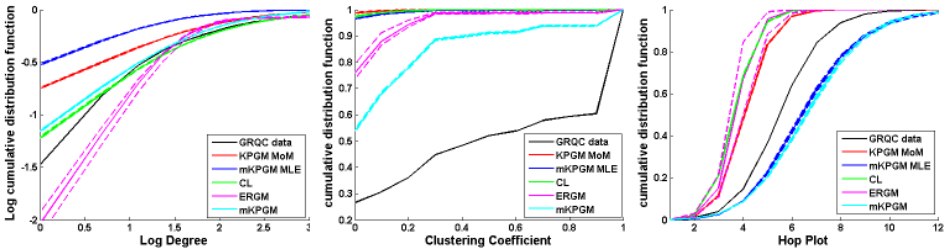
(a) Multiple networks data (b) Single networks data

Fig. 19. 3 dimensional Kolgomorov-Smirnov distance for all datasets.



(a) Degree (b) Clustering Coefficient (c) Hop Plot

Fig. 20. Variation of graph properties in generated Nutella networks.



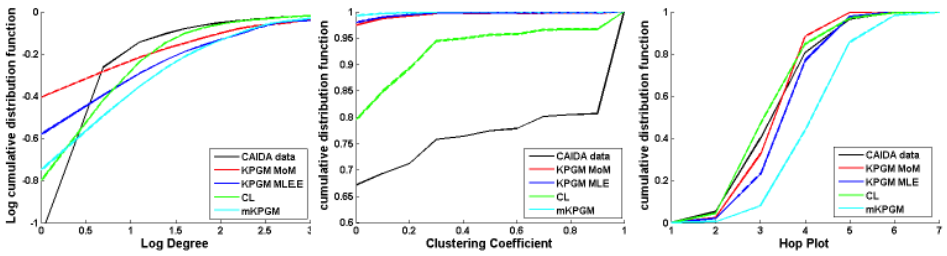
(a) Degree (b) Clustering Coefficient (c) Hop Plot

Fig. 21. Variation of graph properties in generated GRQC networks.

distribution over the different data, beside Nutella network, mKPGM is one of the best model with Chung-Lu. While KPGM-MOM underestimate the degree in all the cases, ERGM overestimate. Finally KPGM-MLE can model two of the datasets, doing a complete underestimation over the GRQC data.

The low clustering coefficient for Nutella data Figures 20(b) is almost perfectly matched by mKPGM, being the best model in comparison to the other data. Given the very high clustering coefficient of GRQC dataset (Figure 21(b)), it is almost impossible to the model match this characteristic, being mKPGM the closest model to match it. However, the clustering coefficient for CAIDA networks was not model by mKPGM, which could be related by the nature of business relation (Figure 22(b)).

Figures 20(c)-22(c) show the hop plot distribution for all the networks. mKPGM is the best model in two of the dataset, having an almost perfect match with Nutella data. Similarly, than previous results, the hop plot distribution in CAIDA network is not as well as expected.



(a) Degree (b) Clustering Coefficient (c) Hop Plot

Fig. 22. Variation of graph properties in generated CAIDA networks..

In Figure 18(right column), we report the learned parameters for KPGM and mKPGM models. Given the different behavior of the network, the differences over the learned parameter for each model is expected. While GRQC behaves similarly to the previous dataset to model the extremely high clustering coefficient of the network (high values in the main diagonal), the NUTELLA learned parameters does not have a extremely high value to avoid a high clustering coefficient. On the other hand, CAIDA learned parameters, has very low values in the main diagonal, and two middle values in the upper left triangle, explaining the low clustering coefficient observed in the network.

Figure 19(b) shows the KS_{3D} distance for the three single datasets. The results show that mKPGM algorithm has the smallest error in all datasets, outperforming all the other models (remember that we could not learn an ERGM on CAIDA due to memory issues). In two of the datasets, mKPGM achieves almost a 50% reduction in KS distance compared to the next best competitor. This confirms that our learning algorithm can also be applied to model datasets where we have no information about variability and outperform existing KPGM learning methods. This implies that other models could have a better independent match of the characteristics by separate, however they are not resembling the real structure of the networks.

9. DISCUSSION AND CONCLUSIONS

In this paper, we investigated whether the state-of-the-art generative models for large-scale networks are able to reproduce the properties of multiple instances of real-world networks generated by the same source. Surprisingly Chung-Lu, ERGM and KPGMs methods, some of the most commonly used models, produces very little variance in the simulated graphs, significantly less than the observed in real data. To explain this effect, we showed analytically that KPGMs and Chung-Lu cannot capture the variance in the number of edges that we observe in real network populations, and attempts to inflate the variance of characteristics of KPGM graphs by increasing the number of parameters or by employing Bayesian approach were not sufficient. Moreover, KPGMs and Chung-Lu can not reproduce the clustering coefficient observed in real network being incapable to model most of social networks.

We demonstrate, that the lack of variance in the number of edges for KPGM and Chung-Lu can be explained by the independent drawn of edges, in the generation process. To solve this problem for KPGM, we proposed a tied sampled of the edges. The Tied KPGM, that preserves the marginal probabilities of edges in a given location but does not treat them as independent, results in considerably more variance than the original KPGM model and the variance reflect in real-world domains. Due to this, we introduce the *mixed-KPGM*, that introduces dependence in the edge generation process by performing Bernoulli trials to determine whether new edges are added in a

hierarchy. By choosing the level where the hierarchy begins, one can tune the amount of edges that are grouped in a sampled graph.

The mKPGM is a generalization of KPGM ($\ell = K$) and tKPGM ($\ell = 1$), with special properties when the value of ℓ increase. An experimental analysis empirically demonstrates that as ℓ increment from 1 to K the mean and variance of the clustering coefficient for the sampled network decrease, and the variance of the total number of edges decrease too. As an opposite behavior, the diameter and the degree increment proportionally to the value of ℓ. Another experimental analysis indicates that when the initial size of Θ is increased to $b = 3$, mKPGM can generate a wider range of networks with high clustering coefficient. In contrast, KPGMs do not exhibit as much change with respect to b.

One synthetic and three real Multiple networks datasets were analyzed using Chung-Lu, ERGM, KPGM and mKPGM. mKPGM is the best model among them, mKPGM improves the fit in clustering coefficient, and specially in hop plot distribution where is the only model to match, almost perfectly, this characteristic. Moreover, mKPGM is the only model able to reproduce the observed variance in network characteristics. The improved fit is due primarily to the ability of mKPGMs to jointly capture the clustering coefficient and the hop plot distribution.

We also demonstrated, over three single networks real datasets, that mKPGM can be applied to learn from a single network by setting $\ell = K$. In this case, mKPGMs offer a significant improvement over current KPGM learning algorithms—by avoiding the difficulties of search over permutations and the complexities of analytical moment calculations through the use of simulated method of moments.

These results agree with the new statical measures utilized in the paper, the 3D Kolmogorov-Smirnov test measure. This measure considers the correlation among graph distributions utilized in this paper and enables a more accurate assessment of the characteristics of generated networks by considering the empirical joint distribution. In the seven datasets utilized in this paper, the mKPGM algorithm obtained the best performance among all datasets. Confirming, the capability of mKPGM to model real data.

In the future, we will extend the mKPGM learning algorithm to consider the use of additional moments, including higher order moments (i.e., variance) that can be used to learn the most appropriate way to tie parameters (e.g., by varying levels throughout the graph).

REFERENCES

AHMED, N., NEVILLE, J., AND KOMPELLA, R. 2011. Network sampling via edge-based node selection with graph induction. Tech. Rep. CSD TR #11-016, Dept of Computer Science, Purdue University.

BARABSI, A.-L. AND ALBERT, R. 1999. Emergence of scaling in random networks. *Science 286,* 5439, 509–512.

CHUNG, F. AND LU, L. 2002. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences of the United States of America 99,* 25, 15879–15882.

ERDOS, P. AND RENYI, A. 1960. On the evolution of random graphs. In *PUBLICATION OF THE MATHEMATICAL INSTITUTE OF THE HUNGARIAN ACADEMY OF SCIENCES*. 17–61.

FRANK, O. AND STRAUSS, D. 1986. Markov graphs. *Journal of the American Statistical Association 81,* 395, pp. 832–842.

GLEICH, D. F. AND OWEN, A. B. 2012. Moment based estimation of stochastic Kronecker graph parameters. *Internet Mathematics X,* X, XX–XX.

HANDCOCK, M. S. 2003. Assessing degeneracy in statistical models of social networks. Working Paper 39, Center for Statistics and the Social Sciences, University of Washington.

HARRIS, K. 2008. The National Longitudinal Study of Adolescent health (Add Health), Waves I & II, 1994-1996; Wave III, 2001-2002 [machine-readable data file and documentation]. *Chapel Hill, NC: Carolina Population Center, University of North Carolina at Chapel Hill.*.

KUMAR, R., RAGHAVAN, P., RAJAGOPALAN, S., SIVAKUMAR, D., TOMKINS, A., AND UPFAL, E. 2000. Stochastic models for the web graph. In *Proceedings of the 42st Annual IEEE Symposium on the Foundations of Computer Science*.

LESKOVEC, J., CHAKRABARTI, D., KLEINBERG, J., FALOUTSOS, C., AND GHAHRAMANI, Z. 2010. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research 11,* Feb, 985–1042.

LESKOVEC, J. AND FALOUTSOS, C. 2007. Scalable modeling of real graphs using kronecker multiplication. In *Proceedings of the 24th international conference on Machine learning*. ICML '07. ACM, New York, NY, USA, 497–504.

MAHDIAN, M. AND XU, Y. 2007. Stochastic Kronecker graphs. In *5th International WAW Workshop*. 179–186.

MORENO, S. AND NEVILLE, J. 2009. An investigation of the distributional characteristics of generative graph models. In *Proceedings of the The 1st Workshop on Information in Networks*.

NEWMAN, M. E. J. 2003. The structure and function of complex networks. *SIAM Review 45,* 2, 167–256.

PAKES, A. AND POLLARD, D. 1989. Simulation and the asymptotics of optimization estimators. *Econometrica 57,* 5 Sep, 1027–1057.

ROBERT, C. AND CASELLA, G. 2004. *Monte Carlo statistical methods*. Springer Verlag.

ROBINS, G., SNIJDERS, T., WANG, P., HANDCOCK, M., AND PATTISON, P. 2006. Recent developments in exponential random graph (p*) models for social networks. *Social Networks 29*, 192–215.

SESHADHRI, C., PINAR, A., AND KOLDA, T. G. 2011. An in-depth study of stochastic kronecker graphs. *Data Mining, IEEE International Conference on 0*, 587–596.

SNIJDERS, T., PATTISON, P., ROBINS, G., AND HANDCOCK, M. 2004. New specifications for exponential random graph models. *Sociological Methodology 36*, 99–153.

WASSERMAN, S. AND PATTISON, P. E. 1996. Logit models and logistic regression for social networks: I. An introduction to Markov graphs and p*. *Psychometrika 61*, 401–425.

WATTS, D. AND STROGATZ, S. 1998. Collective dynamics of 'small-world' networks. *Nature 393*, 440–42.