Learning Mixed Kronecker Product Graph Models with Simulated Method of Moments

Sebastian Moreno
Dept of Computer Science
Purdue University
smorenoa@cs.purdue.edu

Jennifer Neville
Depts of Computer Science and Statistics
Purdue University
neville@cs.purdue.edu

Sergey Kirshner
Dept of Statistics
Purdue University
skirshne@purdue.edu

ABSTRACT

There has recently been a great deal of work focused on developing statistical models of graph structure—with the goal of modeling probability distributions over graphs from which new, *similar* graphs can be generated by sampling from the estimated distributions. Although current graph models can capture several important characteristics of social network graphs (e.g., degree, path lengths), many of them do not generate graphs with sufficient *variation* to reflect the natural variability in real world graph domains. One exception is the *mixed Kronecker Product Graph Model* (mKPGM), a generalization of the Kronecker Product Graph Model, which uses *parameter tying* to capture variance in the underlying distribution [10]. The enhanced representation of mKPGMs enables them to match both the mean graph statistics and their spread as observed in real network populations, but unfortunately to date, the only method to estimate mKPGMs involves an exhaustive search over the parameters.

In this work, we present the first learning algorithm for mKPGMs. The $O(|\mathbf{E}|)$ algorithm searches over the continuous parameter space using constrained line search and is based on *simulated method of moments*, where the objective function minimizes the distance between the observed moments in the training graph and the *empirically estimated* moments of the model. We evaluate the mKPGM learning algorithm by comparing it to several different graph models, including KPGMs. We use multi-dimensional KS distance to compare the generated graphs to the observed graphs and the results show mKPGMs are able to produce a closer match to real-world graphs (10-90% reduction in KS distance), while still providing natural variation in the generated graphs.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning*

Keywords

Link analysis, statistical graph models, Kronecker models, method of moments estimation.

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.

1. INTRODUCTION

Graphs are a natural representation to use for the analysis of complex systems. As such, there has been a great deal of research focusing on the development of models that accurately reflect the characteristics of real-world networks. This work includes stochastic algorithms (e.g., [16, 2, 6]) that are capable of generating graphs with particular properties of interest (e.g., short geodesic distance, high local clustering). It also includes statistical models that are capable of modeling probability distributions over graphs. For example, Exponential Random Graph Models (ERGMs) [15], Chung-Lu models (CL) [3], and Kronecker Product Graph Models (KPGM) [8, 9]. Statistical models such as these offer the advantage that model parameters can be estimated from observed networks and, once learned, new graphs can be generated from the estimated distribution by sampling.

Of the statistical models of graphs, the KPGM method [9] is intuitively appealing for its elegant fractal structure and fast sampling algorithms (i.e., $O(|\mathbf{E}|)$). However, recent work has identified a few limitations of KPGMs. Seshadhri et al. [13] showed via mathematical analysis that graphs generated from the KPGM have 50-75% isolated vertices with no incident edges and Moreno et al. [11] showed empirically that graphs generated from the KPGM do not capture the level of clustering observed in many network datasets.

Moreover, while recent graph models adequately capture several important characteristics of social network graphs (e.g., skewed degree, short path lengths), many of them do not generate graphs with sufficient *variation* to reflect the natural variability of real world graph domains [11]. Specifically, when learned from real-world social network samples, the models appear to place most of the probability mass on a relatively small subset of graphs with very similar characteristics. This clearly limits the applicability of the models for representing and reasoning about graph populations.

Recently, we proposed a *mixed*-KPGM (mKPGM) [10] to address these issues. The mKPGM is a generalization of the KPGM that uses *parameter tying* to model dependencies among edges. These dependencies enable the model to more accurately capture the clustering observed in real-world networks. The dependencies also increase the variance of the estimated distribution while preserving the expectation—thus mKPGMs are able to more accurately capture the natural variation observed in real-world network *populations*. However, the parameter tying makes it more difficult to *learn* the model from observed data (due to edge dependencies) and to date the only method to determine mKPGM parameters from data involves exhaustive search.

In this work, we present the first tractable learning algorithm for mKPGMs. The algorithm is based on *simulated method of moments* where the distance between the observed moments in the training data and the *empirically estimated* moments of the model is minimized using constrained line search for continuous optimization. We evaluate the proposed mKPGM learning algorithm by comparing to several alternative graph models, including KPGMs, on six real-world network domains. The results show that mKPGMs are able to capture the characteristics of the real-world graphs more accurately, while still providing natural variation in the generated graphs.

The contributions of the paper include:

- The first tractable estimation algorithm for mKPGMs based on the simulated method moments, with complexity $O(|\mathbf{E}|)$—which facilitates learning mKPGM models from observed network datasets.
- Development of *multi-dimensional KS distance*—a new method for evaluation to compare joint distributions of multiple graph statistics, rather than calculating the KS distance in each dimension independently.
- Empirical evaluation of mKPGM shows 10-90% reduction in KS distance over KPGMs on several real world population datasets, which shows mKPGMs are significantly more accurate at capturing the characteristics of real world network populations.
- Empirical evaluation of SMM learning in KPGMs shows 10-40% reduction in KS distance over previous KPGM learning algorithms, which shows how mKPGMs generalize KPGMs and our proposed learning algorithm can improve KPGM modeling of single network datasets.

2. BACKGROUND

This section describes the Kronecker Product Graph Model (KPGM) and two algorithms to learn its parameters based on (i) maximum likelihood estimation (MLE) [9] and (ii) method of moments (MoM) [4]. We also describe the mixed-KPGM (mKPGM) [10] and include a brief discussion about the choice of initiator matrix size.

2.1 Kronecker Product Graph Model

Let Θ be a $b \times b$ *initiator matrix* of parameters, where $\forall i,j \;\; \theta_{ij} \in [0,1]$, and let K be a parameter that determines the size of the graph (i.e., $|\mathbf{V}| = b^K$). Then the KPGM algorithm generates a graph $G_K = (\mathbf{V}_K, \mathbf{E}_K)$, where $\mathbf{V}_K$ and $\mathbf{E}_K$ are the set of nodes and edges respectively, as follows. First, the model computes the K^{th} Kronecker power of the initiator matrix Θ via $K-1$ Kronecker products of Θ with itself. This produces a $b^K \times b^K$ matrix $P_K = \Theta^{[K]}$, where $P_K(i,j)$ represents the probability of an edge existing between nodes i and j. P_K is used to generate a graph G_K with b^K nodes, by sampling each edge independently from a Bernoulli($P_K(i,j)$) distribution (i.e., if the trial is successful, the edge e_{ij} is added to $\mathbf{E}_K$).

Given an observed training network $G^\star = (\mathbf{V}^*, \mathbf{E}^*)$, the KPGM likelihood of the graph is:

$$P(G^\star|\Theta,\sigma) = \prod_{(i,j)\in\mathbf{E}^*} P_K(\sigma_i,\sigma_j) \prod_{(i,j)\notin\mathbf{E}^*} (1 - P_K(\sigma_i,\sigma_j)) \quad (1)$$

Here σ_i denotes the position of node i according to a permutation σ. MLE learning algorithm finds the parameters Θ that maximizes the likelihood of the observed graph given a permutation (σ) of the rows and columns of the adjacency matrix [9]. In practice, the true permutation is unknown and the learning algorithm uses a Metropolis-Hasting sampling approach to search over the factorial number of possible permutations of the network. The algorithm then uses a gradient descent approach to update the parameters Θ, where the derivative of the likelihood is approximated given the current σ and Θ.

A second learning algorithm for KPGMs was developed in [4], which is based on *method of moments*. The strength of this approach is that it is permutation invariant—thus it avoids the difficulty of search over permutation space. The MoM learning algorithm searches for parameters Θ that minimize the following objective function:

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

Here F_i is a function over a network $G = (\mathbf{V},\mathbf{E})$ that calculates a statistic of the graph, e.g., for number of edges: $F = |\mathbf{E}|$. Then, $\mathbf{F}^* = \{F_1^*, F_2^*, \cdots, F_{N_m}^*\}$ corresponds to a set of N_m *sample moments* of the training network $G^\star$ and $E[F_i|\Theta]$ is the expected value of those statistics (i.e., *distributional moments*) given particular values of Θ. The MoM learning algorithm [4] considers four moments: the number of (i) edges, (ii) 2-stars, (iii) 3-stars, and (iv) triangles. These moments were selected because their expected values can be analytically calculated for any Θ, provided $b = 2$.

2.2 Mixed Kronecker Product Graph Model

The mKPGM is a generalization of the KPGM, which uses *parameter tying* to capture the clustering and natural variation observed in real-world networks more accurately [10]. The marginal probabilities of edges in P_K are preserved but the edge probabilities are no longer independent.

Specifically, given Θ, K, and a parameter $\ell \in [1,\cdots,K]$ that specifies the level of parameter tying, the mKPGM generation process samples a network of size b^K as follows. First, the model uses the standard KPGM algorithm with initiator matrix Θ to calculate a probability matrix $P_\ell = \Theta^{[\ell]}$ and then a graph G_ℓ is sampled from P_ℓ. Next a subsequent Kronecker product is computed from G_ℓ to produce a new probability matrix $P_{\ell+1} = G_\ell \otimes \Theta$. Then a graph $G_{\ell+1}$ is sampled from $P_{\ell+1}$ for further Kronecker products. The process of sampling a graph before computing subsequent Kronecker products produces dependencies among the sampled edges. This process is repeated $K - \ell - 1$ times to generate the final network G_K. For more details see [10].

The parameter ℓ controls the level of tying and thus impacts the variance and clustering of the model. Lower values of ℓ produce larger dependencies among the edges and greater clustering among the nodes. When $\ell = K$ the model is equivalent to KPGM and this produces lower clustering and lower variance.

The mKPGM likelihood has two parts: the *untied* part (i.e., G_ℓ) is calculated as in the original KPGM, while the *tied* part is based on the $K - \ell$ Kronecker products where edges share parameters. The mKPGM likelihood of an observed graph G^* (given a permutation σ) is based on the matrix G_ℓ^* that was generated to produce the permuted network G^* under σ:

$$P(G^*|\Theta,\sigma) = P(G_\ell^*|\theta,\sigma_\ell) \left(\prod_{(i,j)\in\mathbf{E}^*} G_\ell^* \left(\left\lfloor \frac{\sigma_i - 1}{b^{K-\ell}} \right\rfloor, \left\lfloor \frac{\sigma_j - 1}{b^{K-\ell}} \right\rfloor \right) \right) \cdot$$

$$\prod_{k=1}^{K-\ell} \theta_{i_k j_k} \prod_{(i,j)\notin \mathbf{E}^*} \left(1 - G^*_\ell\left(\left\lfloor \frac{\sigma_i - 1}{b^{K-\ell}} \right\rfloor, \left\lfloor \frac{\sigma_j - 1}{b^{K-\ell}} \right\rfloor\right) \prod_{k=1}^{K-\ell} \theta_{i_k j_k}\right) \quad (3)$$

Here $P(G^*_\ell|\theta, \sigma_\ell)$ is the likelihood of the untied part given by Eq. 1, where $\sigma_\ell = 1, \cdots, |\mathbf{V}^*_\ell|$ and $G^*_\ell(i,j) = 1$ if there is at least one edge in the submatrix of G^* defined by $(i-1) * b^{K-\ell} + 1, \cdots, i * b^{K-\ell} \times (j-1) * b^{K-\ell} + 1, \cdots, j * b^{K-\ell}$ under the *permutation* σ, and $G^*_\ell(i,j) = 0$ otherwise. In addition, $G^*_\ell\left(\left\lfloor \frac{\sigma_i - 1}{b^{K-\ell}} \right\rfloor, \left\lfloor \frac{\sigma_j - 1}{b^{K-\ell}} \right\rfloor\right)$ is the mapping of the pair (i,j) to its corresponding cell in the adjacency matrix G^*_ℓ and results in an edge probability of zero when the cell in G^*_ℓ is 0. $\prod_{k=1}^{K-\ell} \theta_{i_k j_k}$ is the remaining probability for a tied edge, where $\theta_{i_k j_k}$ corresponds to the probability of a successful Bernoulli trial for (i,j) at scale k in the hierarchy—which is determined by the entry of $\mathcal{P}_1$ corresponding to the k^{th} bits of (i,j) (see [10] for a detailed explanation of the hierarchical bit representation).

2.3 Initiator matrix

To investigate the impact of the size of the 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 9,239 different values of Θ with $b = 2$. For each Θ, we generated 50 networks with $K = 11$ from KPGM ($\ell = K$) and mKPGM ($\ell = 6$). This produced networks with 3,500-15,200 edges. We also generated the same quantity of networks, with similar number of nodes, edges and equivalent levels of parameter tying for Θ of size $b = 3$. For $b = 3$, we considered 18,816 different values of Θ and generated networks with $K = 7$ from KPGMs ($\ell = K$) and mKPGM ($\ell = 4$). This produced networks with 3,400-16,300 edges. For the generated networks, we calculated two moments: (i) average clustering coefficient, and (ii) average geodesic distance (see Figure 1).

Figure 1 shows that mKPGM and KPGM are able to produce networks with higher clustering coefficient when using a larger initiator matrix ($b = 3$). Moreover, the increased size of the initiator matrix does not change considerably the average geodesic distance of the resulting networks. Consequently, since a slightly larger initiator matrix increases the coverage of the graph space without significantly increasing the number of parameters that need to be estimated (4 vs. 9), in this paper we use $b = 3$ for the initiator matrix size.

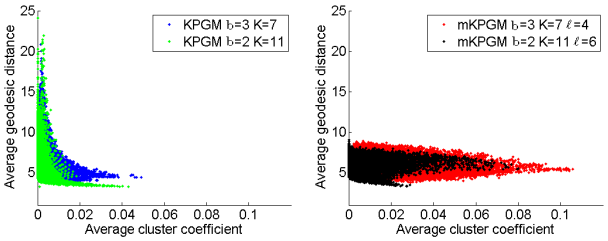

Figure 1: Variation of graph properties for synthetic networks using generator matrix of size b=2 and b=3 for KPGM (left) and mKPGM (right)

3. MKPGM ESTIMATION

The simulation experiments in [10] demonstrate the advantages of the mKPGM representation for capturing important characteristics of real world networks (i.e., clustering and variation). However, previous experiments used exhaustive search to identify model parameters—to date there is no training algorithm to *learn* mKPGM parameters automatically from an observed network. Here we describe the first tractable learning algorithm for mKPGMs. In Section 5 we will apply the algorithm to learn mKPGM parameters from real-world network populations and show that the learned models are more accurate than competing models.

First we note that even though the mKPGM likelihood (Eq. 3) is similar to that of KPGMs (Eq. 1), it can not easily be used as an objective function to estimate the parameters of the model. In principle, if the level of tying (ℓ) is known apriori, a MLE algorithm could alternate between sampling permutations and estimating the parameters of the untied and tied parts of the model. However, we found that MLE is not likely to be successful in this case, since local search (i.e., swapping a single pair of nodes) in permutation space is extremely unlikely to discover the block structure that results from parameter tying. Since it is necessary to recover the block structure permutation to accurately estimate the parameters with MLE, in practice MLE only works well when the search starts from very close to the true permutation—which is not the case for real datasets.

Next, we note that the MoM approach to learn KPGMs [4] is difficult to apply to mKPGMs since analytical expressions for even simple moments are difficult to derive for mKPGMs due to the complex dependencies between the edges (the key characteristic of the model). For example, let $A_{ij} = 1$ when there is an edge between nodes i and j, and 0 otherwise. Then for nodes that have common (tied) parameters in the network generation of $G_{K-\ell}$: $E[A_{ij}A_{kl}] \neq E[A_{ij}]E[A_{kl}]$. This makes it difficult to calculate distributional moments analytically and directly minimize $f(\Theta, \mathbf{F}^*)$ in Eq. 2.

However, a strength of MoM estimation is that it is permutation invariant—i.e., it successfully avoids the search over the factorial permutation space which is even more difficult in the space of mKPGMs. At the same time, a strength of both mKPGMs and KPGMs is their ability to generate sample networks in sub-quadratic time. To exploit these two strengths we developed a *simulated method of moments* (SMM) learning algorithm that approximates the objective function in Eq. 2 with empirically estimated moments. Simulated method of moments (see e.g., [12]) 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 $E[\mathbf{F}|\Theta]$ with an empirical estimation $\widehat{E}[\mathbf{F}|\Theta]$ based on simulated networks from Θ. Algorithm 1 outlines the function *estObjFunc* where we estimate $f(\Theta, \mathbf{F}^*)$ empirically. Specifically it estimates $\widehat{E}[\mathbf{F}|\Theta]$ from a set of sampled networks $\mathcal{S}$. Each network G_i ($i \in \{1, \cdots, N_s\}$), where $N_s = |\mathcal{S}|$, is generated with the mKPGM algorithm using the input parameters Θ, K, and ℓ and the specified moments are calculated in each G_i. Each expected moment is then estimated from the median of the moments observed in the set $\mathcal{S}$: $\widehat{E}[F_i|\Theta] = median(F_{i1}, \cdots, F_{iN_s})$. Finally, the value of the objective function is calculated using the estimated distributional moments. The complexity of this function depends on both the complexity of sampling N_s networks from the model and the complexity of calculating the moments in

Algorithm 1 Function *estObjFunc*

Require: Θ, K, ℓ, N_s, $\mathbf{F}^*$
1: **for** $i = 1; i++; i \leq N_s$ **do**
2: Generate network G_i using mKPGM with (Θ, K, ℓ)
3: Calculate moments $\mathbf{F}_i$ for G_i.
4: **for** $i = 1; i++; i \leq |\mathbf{F}|$ **do**
5: $\widehat{E}[F_i|\Theta] = median(F_{i1}, \cdots, F_{iN_s})$
6: **return** $\sum_{|\mathbf{F}|} \left(\frac{\mathbf{F}^* - \widehat{E}[\mathbf{F}|\Theta]}{\mathbf{F}^*} \right)$

each network. While generating a network from mKPGM is $O(|\mathbf{E}|)$, the time needed for moment estimation depends on the chosen moments. For example, computing average node degree is $O(|\mathbf{E}|)$ but computing average geodesic distance could be $O(|\mathbf{V}||\mathbf{E}|\ log|\mathbf{V}|)$. In this work, we approximate the calculation of moments to make them at most linear in $\mathbf{E}$ (e.g., by sampling shortest paths). Thus, the complexity of estimating the objective function is $O(N_s \cdot |\mathbf{E}|)$.

The use of SMM avoids the difficulties of determining an analytical expression for each moment and facilitates the incorporation of a wider range of moments in the learning algorithm. However, since the objective function in Algorithm 1 is not convex, we need a way to search over the parameter space to minimize $f(\Theta, \mathbf{F}^*)$. Moreover, without a closed form expression for the moments, we also can not estimate their gradients, thus a gradient descent-type optimization method is not applicable. To offset these issues, we developed a line search optimization method, using the empirically estimated moments from SMM.

To consider whether linear search would be reasonable to pursue, we explored whether the objective function is locally convex. We varied each parameter θ_{ij}, while keeping the rest of parameters constant, and evaluated $f(\Theta, \mathbf{F}^*)$ for different θ_{ij} (see Figure 2, left). In all cases, we observed a locally convex error function with a minimum around the value of θ_{ij} such that $S_\Theta^K = (\sum_i \sum_j \theta_{ij})^K \approx |\mathbf{E}^*|$ (the number of edges of the training network). This implies that a one dimensional linear search (i.e., changing a single parameter) will not be effective—once it reaches a local minima that matches $|\mathbf{E}^*|$ it will not explore any further because changing a single parameter in isolation will always change the expected number of edges. However, a two dimensional (2D) search, where we keep S_Θ^K constant through a simultaneous increase in one parameter and equal decrease in another, could enable a successful search to improve $f(\Theta, \mathbf{F}^*)$. To test this, we varied combinations of two parameters in Θ and evaluated $f(\Theta, \mathbf{F}^*)$ for different values of the parameters, while keeping the rest of parameters constant (figure 2, right). Again, in all cases, the 2D curve was locally convex.

Thus, under the assumption that the 2D space is likely to be locally convex, our algorithm approximates a full search over parameter space by performing a linear search in 2D, while constraining the parameters to match $S_\Theta^K = |\mathbf{E}^*|$.

3.1 mKPGM Learning Algorithm

Algorithm 2 outlines our learning algorithm which combines SMM with a constrained line search in two dimensions (2D) to identify the best set of parameters. The algorithm begins calculating $K = \left\lceil \frac{\log(|\mathbf{V}^*|)}{\log(b)} \right\rceil$, where $|\mathbf{V}^*|$ is the number of nodes of the training network $G^\star = (\mathbf{V}^*, \mathbf{E}^*)$ and $b \times b$ is the size of the parameter matrix Θ. It continues with the initialization $\forall i, j\ \theta_{ij} = \sqrt[K]{|\mathbf{E}^*|}/b^2$, which ensures the constraint $S_\Theta^K = |\mathbf{E}^*|$ is met. With the initial set of parameters, the initial error EF^* is calculated with *estObjFunc*. The algorithm continues with the generation of the set of $N_c = \binom{b^2}{2}$ possible pairs of parameters Θ_{pairs} to consider in the 2D search. In case of undirected networks the set Θ_{pairs} is reduced to $N_c = \binom{b(b+1)/2}{2}$ elements, due to the symmetric relationship in Θ.

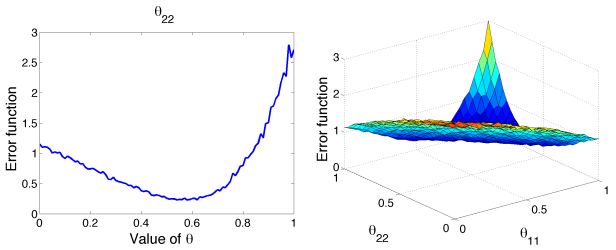

Figure 2: Error function with respect to the variation of one (left) and two (right) parameters.

Algorithm 2 mKPGM training algorithm

Require: $G^\star = (\mathbf{V}^*, \mathbf{E}^*)$, b, δ, ℓ, $iter$, N_s
Calculate the moments $\mathbf{F}^*$ for $G^\star$
$K = \left\lceil \frac{\log(|\mathbf{V}^*|)}{\log(b)} \right\rceil$
Initialize $\forall\ i, j\ \theta_{ij} = \sqrt[K]{|\mathbf{E}^*|}/b^2$
$EF^* = estObjFunc(\Theta, K, \ell, N_s, \mathbf{F}^*)$
$N_c = \binom{b^2}{2}$ {combinations of 2 parameters}
Let $\Theta_{pairs} = \{(11, 12), \cdots, (b(b-1), bb)\}$
for $i = 1; i++; i \leq iter$ **do**
  $j_1 = 0$
  $idx = N_c$
  while $j_1 < idx$ **do**
    $index = \Theta_{pairs}(mod(j_1, N_c) + 1)$
    for $j_2 = -3; j_2++; j_2 \leq 3$ **do**
      $\Phi = \Theta$
      $\phi_{index(1)} = \phi_{index(1)} + j_2 * \delta$
      $\phi_{index(2)} = \phi_{index(2)} - j_2 * \delta$
      if $\forall\ i_1, i_2 \in \{1, \cdots, b\},\ \ \phi_{i_1,i_2} \in [0, 1]$ **then**
        $EF' = estObjFunc(\Phi, K, \ell, N_s, \mathbf{F}^*)$
        if $EF' < EF^*$ **then**
          $EF^* = EF'$
          $\Theta = \Phi$
          $idx = idx + mod(j_1, N_c) + 1$
          {Search is extended for the next N_c iterations}
    j_1++
  $\delta = \delta/2$
return Θ

The algorithm then begins the search over the parameter space, which consists of three loops. The first loop iterates $iter$ times over the step sizes δ, which determines the parameters value changes. The second loop iterates over the set Θ_{pairs}, selecting two parameters in each iteration, which determine the part of parameter space that is searched. The two indexes of the selected parameters are given by the pair $index = \Theta_{pairs}(mod(j_1, N_c)+1)$, where $index(1)$ and $index(2)$ correspond to the parameters indexed by the first and second element of $index$ respectively. The third loop (over j_2), 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 ($\Phi = \Theta$), then two parameters are modified: $\phi_{index(1)} = \phi_{index(1)} + j_2$ and $\phi_{index(2)} = \phi_{index(2)} - j_2$. This modification searches over the two dimensional parameter space while constraining $S_\Theta^K = |\mathbf{E}^*|$. If all the parameters are in the range $[0, 1]$, then the value EF' is calculated by *estObjFunc* using Φ. If $EF' < EF^*$, then Φ is accepted, the error is updated, and the search extended for the next N_c iterations.

The complexity of the mKPGM learning algorithm is $O(c \cdot estObjFunc)$, where $c = iter \cdot N_c$ depends on the number of iterations needed for learning and the size of the initiator matrix (i.e., number of parameters). Since the objective function can be estimated in $O(N_s \cdot |\mathbf{E}|)$, the overall complexity of learning is $O(c \cdot N_s \cdot |\mathbf{E}|)$.

Our SMM learning algorithm has three important advantages over the previous MoM learning method for KPGMs. First, our algorithm is not limited to moments that can be calculated analytically. The only consideration for including additional moments is the additional time needed to calculate them empirically in network samples. Second, the SMM approach facilitates learning for different sizes of initiator matrices (i.e., $b \geq 2$). Third, our algorithm is not limited to undirected networks, since the SMM approach can handle the complexity of directed networks.

In this work, we use $b = 3$ for our initiator matrix and consider 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), (iv) size of the largest connected component, and (v) number of nodes with degree greater than zero (to solve the KPGM problem of isolated nodes [13]).

4. EXPERIMENTS

We compare mKPGMs, learned with our SMM algorithm, to four alternative statistical models of networks and evaluated their ability to model both synthetic and real-world network populations. To assess whether the generated networks capture the properties we observe in real network populations, we use four evaluation measures and visual comparison of the properties of networks generated from the learned models. We also apply our learning algorithm to three single network datasets, where due to the lack of information about population variance, we train the mKPGM algorithm with $\ell = K$ (which is equivalent to a KPGM). The results indicate that our new SMM approach can learn more accurate parameters than current learning methods for KPGMs.

4.1 Datasets

We considered one synthetic dataset and six real-world network datasets in this paper. All datasets were transformed to undirected graphs without self loops. For the synthetic data, we generated 50 networks utilizing the mKPGM algorithm with $K = 7$ and $\ell = 4$ and the Θ_{orig} parameters specified in Figure 2.

The first real dataset (Facebook) 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 400,000 Facebook wall links in a year-long period among over 50,000 Facebook users belonging to the Purdue University network. From this data, we sampled 50 networks based on the same process describe by Moreno [10]. Each network has 2,187 nodes $(b=3)^{(K=7)}$ and the edges were collected over time windows of 60 days.

The second dataset (Email) is drawn from a Purdue email network which was constructed from anonymized email logs on the Purdue mail-servers. The email traffic was recorded over 189 days from August 22, 2011 to February 28, 2012 and is comprised of all email transactions from one Purdue user to another. In order to remove the effects of mailing lists and automated emails, we drop any node that has an incoming or outgoing degree of 0. In our analysis we focus on the 137 daily snapshots of networks that occur during during class periods (i.e., weekdays). The resulting networks have an average of 10,946 nodes and 26,562 edges per day.

The third dataset (AddHealth) consists of a set of social networks from the National Longitudinal Study of Adolescent Health [5]. 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. In this work, we considered a set of 25 school networks with sizes ranging from 800 to 2,000 nodes.

These three real datasets are illustrative examples of graph *populations*—sets of networks that exhibit similar graph structures with natural variation. Each set of graphs is likely to be drawn from the same underlying distribution (e.g., email communication patterns are generated by similar social processes). While the networks population are small to medium-sized networks, the empirical performance of mKPGMs on larger-sized single networks indicates that results will generalize to larger graphs as well.

Three single large networks data were obtained from the Stanford Network Analysis Project[1] and [14]. We learned mKPGM models with $\ell = K$ from these single networks and compared 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 third dataset based on Facebook friendship links in New Orleans (FBOR) as described by [14], consists of 46,952 nodes and 183,412 edges.

4.2 Models

We compare mKPGMs to four alternative models:

Kronecker Product Graph Model MLE (KPGM MLE). The KPGM algorithm using the maximum likelihood training method described in section 2.2.

Kronecker Product Graph Model MoM (KPGM MoM). The KPGM algorithm using the method of moments training method described in section 2.2.

Chung Lu Model (CL) [3]. The CL algorithm models the expected degree distribution via a set of weights w_i proportional to degree of node i. To generate a sample graph from the model, each edge is sampled independently with a Bernoulli distribution with $P(i, j) = w_i w_j$.

Exponential Random Graph Model (ERGM) [15]. ERGMs represent probability distributions over graphs with an exponential linear model that uses feature counts of local graph properties (e.g., edges, triangles).

4.3 Evaluation

Our evaluation investigates whether the models capture three important graph characteristics of real network datasets: degree, clustering coefficient, and hop plot (see e.g., [11] for a description of the characteristics). To evaluate the ability of the models to capture these characteristics, we compare the cumulative distribution functions (CDFs) of networks gen-

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

erated from the learned models to the CDFs of the original data. We plot each CDF independently and use those for visual comparison, but we would also like to evaluate the *relationships* among the distributions of characteristics to determine if the models are able to *jointly* capture the characteristics through the network. To measure this quantitatively, we extend the Kolmogorov-Smirnov distance (KS), which measures distributional distance in a single variable, to multiple distributions. In addition to these results, we consider three more network characteristics that were not used as moments during mKPGM learning to assess how the models perform on non-optimized measures.

Distributions of Network Characteristics

The CDF provides a more complete description of the network structure compared to a single aggregate statistic (e.g., average degree). To compare the algorithms, we generated 50 sample graphs from the learned models. From these samples, we estimated the empirical sampling distributions for degree, clustering coefficient, and hop plots. We plot the median and error bars corresponding to the interquartile range for the set of observed network distributions. Solid lines correspond to the median of the distributions and error bars to the 25th and 75th percentiles.

Kolmogorov-Smirnov Distance Between Distributions

Since network characteristics are correlated, independent evaluation of multiple CDFs can make it difficult to accurately evaluate the models. For example, when two or more variables are correlated (e.g., degree and clustering), an algorithm may generate networks that match each (marginal) distribution but do not capture the joint distribution accurately. At the same time, another algorithm may generate networks that do not accurately match the marginal distributions (e.g., one is overestimated and another is underestimated), but the joint distribution may be a closer match.

To quantitatively measure the differences between multidimensional discrete CDFs, we outline a new distance measure—the KS_{3D} which is based on the Kolmogorov-Smirnov (KS) distance. The KS distance between two one dimensional CDFs is the maximum absolute difference between the CDFs: $KS(CDF_1, CDF_2) = max_x|CDF_1(x) - CDF_2(x)|$. This distance varies between 0 and 1, where zero indicates a perfect match of the two distributions.

The KS_{3D} distance captures the correlation among graph measures in multiple dimensions, calculating the maximum difference between two three-dimensional CDFs. To calculate the KS_{3D}, we represent every node as a three-dimensional point $Po_i =< d_i, c_i, g_i >$ where d_i is the degree, c_i is the clustering coefficient, and g_i is the average geodesic distance, for node i respectively. We then calculate the maximum percentage difference between two distributions of Po. Specifically, given two graphs $G_1 = (\mathbf{V}_1, \mathbf{E}_1)$ and $G_2 = (\mathbf{V}_2, \mathbf{E}_2)$, the KS_{3D} distance is defined as:

$$KS_{3D}(G_1, G_2) = max_{Po_x}|cp_1(Po_x) - cp_2(Po_x)| \quad (4)$$

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

The code for the KS_{3D} distance 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 current maximum distance, we save the new difference and continue with the next point. Once that all points are considered, the maximum distance is return. An approximation of the KS_{3D} can be calculated using random selected nodes from each network and using a 3D-grid for the comparison ($KS_{3D}(G_1, G_2) = max_x|cp_1(x) - cp_2(x)|$, where x is a point of the 3D-grid)

Algorithm 3 KS_{3D} distance 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++
    $cp_j = cp_j/|\mathbf{V}_j|$
  if $maxDist < |cp_1 - cp_2|$ **then**
    $maxDist = |cp_1 - cp_2|$
return $maxDist$

Non-Optimized Network Measures

We also evaluate three different characteristics not utilized as moments in the mKPGM learning algorithm, to asses whether the learned mKPGMs can capture other network characteristics that were not explicitly optimized.

K-core: The K-core of a network is the largest induced subgraph where each node has minimum degree k in the subgraph. The distribution of k-core sizes (for varying k) demonstrates the connectivity and community structure of the graph [1]. Considering that two k-core distributions could differ in size, we use skew divergence to measure the divergence of the two distributions PDF_1 and PDF_2. The skew divergence distance is defined as $KL[\alpha PDF_1 + (1-\alpha)PDF_2, \alpha PDF_2 + (1-\alpha)PDF_1]$ with $\alpha = 0.99$.

eigVal: The eigenvalues of the adjacency matrix of the network G measure the centrality of each node in the network. We compare the largest 25 eigenvalues of each network utilizing the standardize absolute distance between them, a distance of zero implies a similarity among the importance of the nodes in the network.

netVal: The first eigenvector of an adjacency matrix contains important information for data analysis [7]. We calculate the euclidean distance between the largest 100 network values of each first eigenvector, where a value of zero implies that important component characteristics are modeled by the networks.

Average: To summarize the results of these three measures, we normalized each measure to the range $[0, 1]$ and averaged the three for each model on each dataset.

5. RESULTS

We evaluated our training algorithm on one synthetic dataset and six real datasets. For each dataset, we selected a single network to use as the training set. To control for variation in the samples, we selected the network that was closest to the

median of the degree distribution on synthetic (network 27, 13,460 edges) and Facebook data (network 11, 5,634 edges). For Email and AddHealth we selected the network which is the closest to $3^8 = 6,561$ and $3^7 = 2,187$ nodes (network 36 with 14,756 edges and network 72 with 15,484 edges respectively). The other real datasets consist of a single network.

Dataset	Real data	$\ell 2$	$\ell 3$	$\ell 4$	$\ell 5$
Synthetic	1,277	-	2,529	**1,107**	-
Facebook	323	-	-	536	**291**
Email	1,037	-	-	1,905	**947**
AddHealth	2,491	**2,826**	1,789	-	-

Table 1: Standard deviation of the number of edges, for real data and mKPGM algorithm.

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 Table 1.

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 $N_s = 10$ (which is sufficient considering that we use the median of the characteristics for learning). From each learned model, we generated 50 sample graphs and calculate the minimum of the measures mentioned in Section 4.3, to control for the effect of network variation.

5.1 Synthetic Data

The synthetic data experiments are intended to evaluate whether our proposed mKPGM estimation algorithm is able to learn parameters successfully in networks generated from mKPGM distributions. Table 2 reports the parameters learned by each of the KPGM-type models. KPGM-MLE does not learn the original parameters, which is also reflected in the CDFs and the evaluation metrics. Even though our mKPGM training algorithm does not recover the *exact* parameters, the learned parameters can emulate the properties of the original synthetic dataset—as is confirmed by the results in Figures 3-6. When MoM is used, we note that models are not identifiable if the number of moments is less 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.

Figures 3-5 (first column) show the CDFs for all methods on the synthetic data. Noticeably, mKPGM is the only method able to capture the observed variance. The large error for ERGMs may be due to degeneracy problems, even though we used some prescribed solutions to avoid it. We expected that KPGMs would generate graphs with less variance than mKPGM, and this is confirmed by the results. Finally, the lack of variance in the CL graphs can also be explained by the models assumption of edge independence, as is demonstrated in Appendix B. The conclusions from visual CDF comparisons are further confirmed by the quantitative KS_{3d} results (see Figure 6) and the non-optimized measures (see Table 3). The low error for mKPGMs confirms the ability of the SMM learning algorithm to identify a good set of parameters.

Θ_{orig}	Θ_{mKPGM}	$\Theta_{KPGM_{MLE}}$	$\Theta_{KPGM_{MoM}}$
$\begin{bmatrix} 0.90 & 0.50 & 0.10 \\ - & 0.90 & 0.10 \\ - & - & 0.70 \end{bmatrix}$	$\begin{bmatrix} 0.81 & 0.07 & 0.42 \\ - & 0.81 & 0.17 \\ - & - & 0.96 \end{bmatrix}$	$\begin{bmatrix} 0.83 & 0.18 & 0.82 \\ - & 0.76 & 0.12 \\ - & - & 0.08 \end{bmatrix}$	$\begin{bmatrix} 1.00 & 0.43 \\ - & 0.38 \end{bmatrix}$

Table 2: Original and learned parameters for Kronecker algorithms in synthetic data.

Synthetic				
Model	K-core	eigVal	netVal	Average
KPGM MoM	1.42	0.37	0.12	0.43
KPGM MLE	0.59	0.18	0.13	0.27
CL	0.26	0.19	0.22	0.33
ERGM	3.18	0.71	0.36	1.00
mKPGM	**0.21**	**0.03**	**0.04**	**0.08**

Table 3: K-core, eigVal and netVal distances over synthetic data for all models.

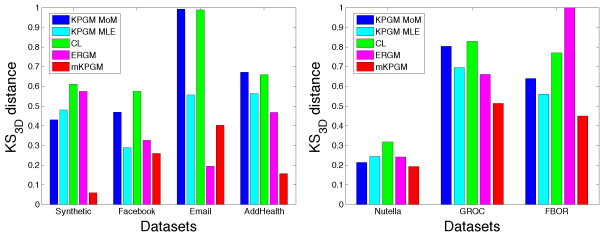

Figure 6: KS_{3d} distance over all dataset.

5.2 Real Data

Similar to results on synthetic data, the only model that can capture the variance observed in the CDFs of real networks is the mKPGM (see Figures 3-5). The CL model is the closest method to the median of the degree distribution in most datasets (figure 3), however, it fails considerably in the other characteristics. Moreover, the low variance of the method makes it difficult for it to match entire distributions. The ERGM can only model the Email data, but while it matches the median of the CDFs fairly well, it still does not capture the variance. Moreover, it produces large errors in the other two datasets. The KPGM cannot model any of the CDFs well and critically lacks the ability to capture any of the clustering in the data (see Figure 4). Finally the mKPGM can model all datasets fairly well. In particular, the mKPGM is the best model for the Facebook and AddHealth data, matching not only the median of the CDFs but also their variance.

These results are further supported by the KS_{3d} distance (Figure 6, left plot), where mKPGM obtains the lowest error in two of the three datasets and the second lowest on the third dataset. Notably, the mKPGM model results in up to a 77% reduction of KS_{3d} distance compared to the KPGM models. These results confirm that the SMM training algorithm not only learns the average of the selected moments but also can learn parameters which capture the correlations among the characteristics. It is important to observe the high error of other models on some of the real datasets. For example CL and KPGM MoM obtain the highest possible error on Email data.

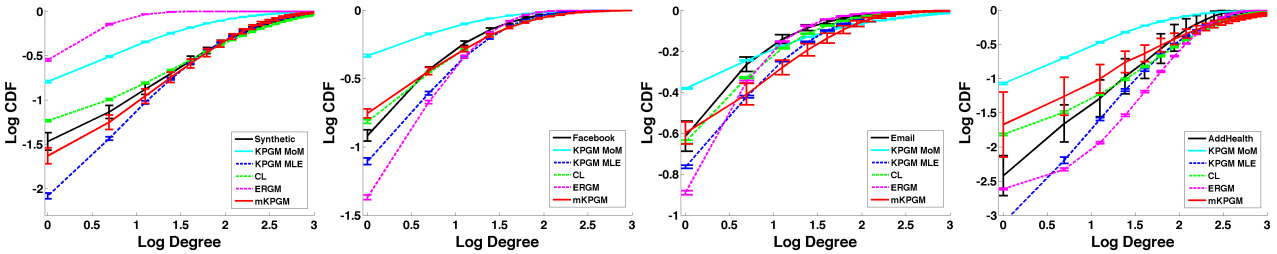

Figure 3: Variation of degree distribution: Synthetic (1st), Purdue (2nd), Email (3rd) and AddHealth (4th).

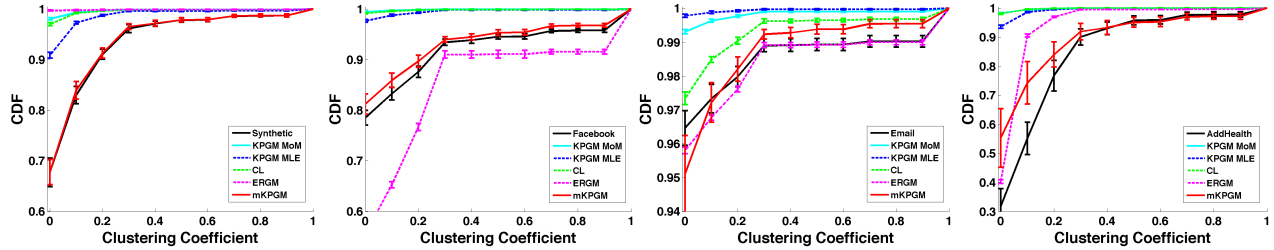

Figure 4: Variation of cluster coefficient: Synthetic (1st), Purdue (2nd), Email (3rd) and AddHealth (4th).

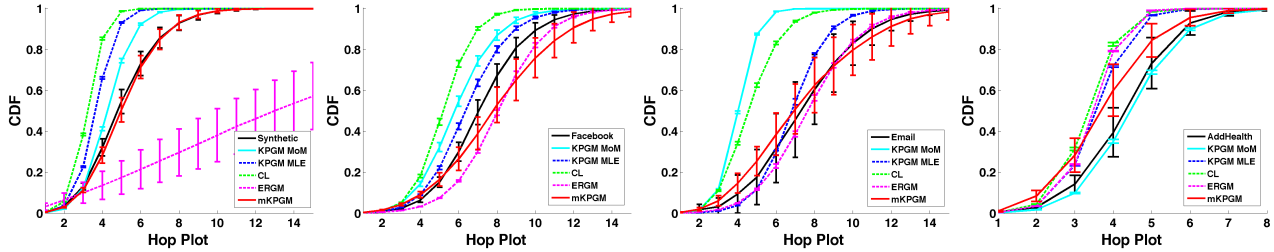

Figure 5: Variation of hop plot: Synthetic (1st), Purdue (2nd), Email (3rd) and AddHealth (4th).

The results for the non-optimized measures of K-core, eigVal and netVal (see Table 4) confirm that the training method for mKPGM is not overfitting to the moments used in the objective function and the learned parameters produce graphs that also capture other characteristics. Specifically, even though the training used different characteristics, mKPGM produces the best match for the three characteristics on the Facebook and AddHealth data, where it obtains the lowest average error. The only exception is the Email data, where the mKPGM model produces networks with a high value in the K-core distance. This is due to the skewness of the K-core distribution in the email data (the highest K-core is three). Beside the ERGM, which is the best model for Email data, the other models can only capture one or two characteristics at the same time, obtaining most of the time an average distance over 0.4.

Given that KPGMs are a special case of mKPGMs ($\ell = K$), we apply our training method to three single network datasets and compare our learning algorithm with $\ell = K$, against the current KPGM training algorithms. From the learned models we generated 50 networks to compare against the real data (except by FBOR where we generate a smaller number of samples because of the network size). For these experiment, we only present the KS_{3D} distances (Figure 6, right plot), and omit other results due to space constraints.

In spite of the fact that mKPGM generates networks equivalent to KPGMs ($\ell = K$), the results show that mKPGMs obtain the lowest KS_{3D} error among all the models—up to 36% of reduction in KS error. This demonstrates that our SMM training algorithm also improves on the previous KPGM training algorithms—even in large networks such as FBOR (46,952 nodes). While other KPGM models can capture some aspects of the networks, CL is the worst model in single datasets (except for FBOR where ERGM cannot be learned due to its size).

Facebook				
Model	K-core	eigVal	netVal	Average
KPGM MoM	**0.07**	0.21	0.10	0.55
KPGM MLE	0.17	0.11	0.12	0.50
CL	0.13	**0.04**	0.21	0.50
ERGM	0.45	0.18	0.11	0.80
mKPGM	0.11	0.06	**0.06**	**0.26**
Email				
Model	K-core	eigVal	netVal	Average
KPGM MoM	0.43	0.26	0.28	0.82
KPGM MLE	0.26	0.27	0.29	0.74
CL	0.21	0.02	0.38	0.47
ERGM	**0.09**	**0.08**	**0.15**	**0.29**
mKPGM	0.53	0.11	0.39	0.80
AddHealth				
Model	K-core	eigVal	netVal	Average
KPGM MoM	2.36	0.33	0.11	0.74
KPGM MLE	0.64	0.16	0.20	0.39
CL	**0.16**	0.18	0.37	0.47
ERGM	0.80	0.22	0.47	0.67
mKPGM	0.21	**0.08**	**0.05**	**0.14**

Table 4: K-core, eigVal and netVal distances over network populations for all models.

6. DISCUSSION AND CONCLUSIONS

In this paper, we presented the first tractable learning algorithm for mixed Kronecker Product Graph Models, making it feasible to *learn* these models from data for the first time. Our empirical evaluation demonstrates that mKPGMs, learned with our simulated method of moments training algorithm (SMM), outperform several competing statistical models of graphs—not only by matching the characteristics of the networks, but also by capturing the variability observed in network populations.

Specifically, we compared against CL, ERGM, KPGM (MLE and MoM) and showed that mKPGMs are significantly more accurate, resulting in 10-90% reduction in KS distance. The improved fit is due to the ability of mKPGMs to jointly capture the clustering coefficient and the hop plot distribution. Moreover, the learned parameters also produce a fairly good match on characteristics that were not included in the objective function (eigenvalues, network values, and K-core). We note that the SMM method is flexible enough to include these additional measures into the objective function directly if they are deemed important enough to explicitly optimize and computational resources allow.

We also demonstrated that SMM learning can be applied to a single network by setting $\ell = K$. mKPGMs learned with SMM offer a significant improvement over current KPGM learning algorithms, not only by reducing the error over the measures (10-40% reduction in KS_{3D} distance) but also by avoiding the difficulties of search over factorial permutation space and the complexities of deriving analytical moments.

For evaluation, we proposed a new measure—the 3-D Kolmogorov-Smirnov distance. This measure considers the correlation among graph distributions (e.g., multiple graph features per node) and enables a more accurate assessment of the characteristics of generated networks by considering the empirical joint distribution rather than independently evaluating marginal distributions.

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).

7. ACKNOWLEDGEMENTS

This research is supported by NSF, ARO, and DARPA under contract number(s) IIS-0916686, IIS-1219015, IIS-1017898, W911NF-08-1-0238, and N660001-1-2-4014. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of NSF, ARO, and DARPA or the U.S. Government.

8. REFERENCES

[1] J. I. Alvarez-Hamelin, A. Barrat, A. Vespignani, and et al. k-core decomposition of internet graphs: hierarchies, self-similarity and measurement biases. *Networks and Heterogeneous Media*, 3(2):371, 2008.

[2] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[3] F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *PNAS*, 99(25):15879–15882, 2002.

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

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

[6] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *FOCS*, 2000.

[7] A. N. Langville and C. D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings.* Princeton University Press, 2006.

[8] J. Leskovec, D. Chakrabarti, J. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In *PKDD*, pages 133–145. Springer-Verlag, 2005.

[9] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *JMLR*, 11(Feb):985–1042, 2010.

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

[11] S. Moreno and J. Neville. An investigation of the distributional characteristics of generative graph models. In *Proceedings of WIN'09*, 2009.

[12] A. Pakes and D. Pollard. Simulation and the asymptotics of optimization estimators. *Econometrica*, 57(5):1027–1057, 1989.

[13] C. Seshadhri, A. Pinar, and T. G. Kolda. An in-depth study of stochastic Kronecker graphs. *ICDM*, pages 587–596, 2011.

[14] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in Facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 37–42, New York, NY, USA, 2009.

[15] S. Wasserman and P. E. Pattison. Logit models and logistic regression for social networks: I. An introduction to Markov graphs and p*. *Psychometrika*, 61:401–425, 1996.

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

APPENDIX

A. MKPGM GENERATION TIME

The running time is given by the KPGM network $G' = (\mathbf{V}', \mathbf{E}')$ and the generation of the $K - \ell$ tied levels for each edge of the set $\mathbf{E}'$. The running time for KPGM is $O(\mathbf{E}')$ [9], while one subnetwork of $K - \ell$ tied takes a time equal to the total number of bernoulli trial. Considering that the expected number of edges for the level $i - 1$ is given by S_Θ^{i-1}, and b^2 bernoulli trials are realized for each edge, the number of trials is:

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

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', the final running time for the entire network is $O(S_\Theta^\ell + S_\Theta^\ell b^2 S_\Theta^{K-\ell}) \approx b^2 S_\Theta^K \approx O(|\mathbf{E}|)$.

B. VARIANCE FOR CL MODEL

Theorem: The independent edge generation process of the Chung-Lu model produces a low variance in the number of edges.
Proof: The variance of the number of edges for a graph $G = (\mathbf{V}, \mathbf{E})$, generated by the Chung-Lu model, is:

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

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

Given that $\sum_{i=1}^{N}\sum_{j=1}^{N} w_i^2 w_j^2 \geq 0$ then $Var\left(|\mathbf{E}|\right) \leq E[|\mathbf{E}|]$.