# Sampling of Attributed Networks from Hierarchical Generative Models

Pablo Robles
Purdue University
West Lafayette, IN USA
problesg@purdue.edu

Sebastian Moreno
Universidad Adolfo Ibañez
Viña del Mar, Chile
sebastian.moreno@uai.cl

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

## ABSTRACT

Network sampling is a widely used procedure in social network analysis where a random network is sampled from a generative network model (GNM). Recently proposed GNMs, allow generation of networks with more realistic structural characteristics than earlier ones. This facilitates tasks such as hypothesis testing and sensitivity analysis. However, sampling of networks with correlated vertex attributes remains a challenging problem. While the recent work of [16] has provided a promising approach for attributed-network sampling, the approach was developed for use with relatively simple GNMs and does not work well with more complex hierarchical GNMs (which can model the range of characteristics and variation observed in real world networks more accurately). In contrast to simple GNMs where the probability mass is spread throughout the space of edges more evenly, hierarchical GNMs concentrate the mass to smaller regions of the space to reflect dependencies among edges in the network—this produces more realistic network characteristics, but also makes it more difficult to identify candidate networks from the sampling space.

In this paper, we propose a novel sampling method, CSAG, to sample from hierarchical GNMs and generate networks with correlated attributes. CSAG constrains every step of the sampling process to consider the structure of the GNM—in order to bias the search to regions of the space with higher likelihood. We implemented CSAG using *mixed Kronecker Product Graph Models* and evaluated our approach on three real-world datasets. The results show that CSAG jointly models the correlation and structure of the networks better than the state of the art. Specifically, CSAG maintains the variability of the underlying GNM while providing a $\geq$ 5X reduction in attribute correlation error.

## Keywords

Complex Networks; Network Sampling; Statistical Network Models; Attributed Networks



*KDD '16, August 13-17, 2016, San Francisco, CA, USA*



DOI: http://dx.doi.org/10.1145/2939672.2939808

## 1 Introduction

Since many complex systems can be represented as networks (where nodes and edges represent entities and their relations, respectively), there has been increased scientific interest in how to model observed network structure effectively—to analyze the properties of the associated systems. Recently many network analyses have focused on data that comprise a single large network (e.g., Facebook, Twitter). Moreover, algorithms are often developed using a small number of network datasets gathered from different domains. Thus, it is critical to have generative network models that can produce data samples that exhibit the natural characteristics of real world networks. Models that generate realistic networks can be used for a variety of analytic tasks. For instance, they enable hypothesis testing (by facilitating assessment of data variability), sensitivity analysis (by facilitating model evaluation under different settings), and benchmark testing (by providing semi-synthetic datasets that preserve privacy) [5].

The ability to generate network structure has served to support research in the basic area of network science but to date there are few methods to generate networks with correlated node attributes [10][16]. Furthermore, few can replicate the range of complex characteristics that occur in real world networks. More precisely, the problem of modeling graph structure with correlated node attributes corresponds to modeling the joint distribution $P(G, \mathbf{X})$ where $P(G)$ and $P(\mathbf{X})$ refer to the marginal distributions of the network structure $G$ and node attributes $\mathbf{X}$, respectively. Most of current methods in network science focus on the problem of modeling $P(G)$ without considering node-attributes $P(\mathbf{X})$[11][18]. Other methods in statistical relational learning focus on the problem of modeling the attribute distribution *conditioned* on the network structure [4]. However, no method effectively represents and samples from the joint distribution $P(G, \mathbf{X})$, due to the complexities of modeling both structure and correlated node attributes (e.g., high-dimensionality, sparse graphs, few sample graphs, etc.). While $P(G, \mathbf{X})$ could be described by combining these efforts to use $P(\mathbf{X}|G)P(G)$, it is typically impractical to draw samples from $P(\mathbf{X}|G)$ due to cyclic attribute dependencies.

More formally, in this work we explore the following–
**Problem Definition:** Let $G_{IN} = (\mathbf{V}, \mathbf{E})$ be an input network with node attributes $\mathbf{X}_{IN}$, where $\mathbf{x}_i$ refers to the attribute value of node $i$ (e.g., $\mathbf{x}_i \in \{0, 1\}$ for binary attributes). Let $\mathbf{P} = P(G, \mathbf{X})$ be the unknown target distribution which generated $< G_{IN}, \mathbf{X}_{IN} >$. If $P(G)$ is the marginal distribution of the network structure estimated with a generative network model $\mathcal{M}$ with parameter $\Theta^G$, and $P(\mathbf{X})$ is

the marginal distribution of node attributes with estimated parameter $\Theta^X$, then our objective is to learn a model using the marginals $P(\mathbf{X}), P(G)$ to construct a proposal distribution $Q$ and sample a pair $< G_{OUT}, \mathbf{X}_{OUT} > \sim Q$ such that it is likely to have been drawn from $P$.[1]

The *Attributed Graph Model* (AGM [16]) is a new approach to solve this problem, which approximates sampling from the joint distribution of structure and attributes $P(G, \mathbf{X})$ using rejection sampling. The AGM uses $Q = P(G)P(\mathbf{X})$ as a proposal distribution—by first sampling the node attributes from $P(\mathbf{X})$ and then using a probabilistic *generative network model* (GNM) to sample edges efficiently from $P(G)$. The proposed edges are accepted with probability proportional to $P(G|\mathbf{X})$ (learned from observed data). The AGM is simple to use and performs well with several edge-based GNMs, such as the Chung Lu (CL) model [2] and the Kronecker product graph model (KPGM) [11].

However, the AGM approximation is only accurate when all edge probabilities in the associated GNM are non-zero, and performance degrades if the edge probability mass is more concentrated over a subset of node pairs, instead of being spread more evenly over all pairs of nodes. As we will discuss later, some hierarchical GNMs that model more complex dependencies among graph edges produce more concentrated probability mass over smaller regions of node pairs. Thus, the AGM will not perform as well with these GNMs.

In this paper, we explore the issue of using hierarchical GNMs to generate attributed networks with correlated attributes and network structure that reflects the characteristics of real world graphs. Namely, we use the *mixed Kronecker product graph model* (mKPGM [14]), a sub-class of hierarchical generative models that overcome known limitations [19] of KPGMs. This hierarchical GNM creates networks by *iteratively* sampling from a hierarchy of latent variables that represent edges, blocks, and superblocks of edges. The hierarchical process restricts the space of possible networks, conditioned on the sampling of the blocks. Specifically, if a block is not sampled, the associated edges have zero probability (i.e., they cannot be sampled). This has significant impact on the model's effective sampling space and makes it difficult to use AGM's rejection sampling process.

We propose *Constrained Sampling for Attributed Graphs* (CSAG), a new method to approximate sampling structure and attributes from $P(G, \mathbf{X})$ using mKPGMs as the generative network model in the proposal distribution. As in AGM, we use a proposal $Q$ composed by the network distribution and the attribute distribution, i.e. $Q = P(G)P(\mathbf{X})$. However, due to the complex GNM hierarchy, we cannot model the target joint $\mathbf{P} = P(G|\mathbf{X})P(\mathbf{X})$ directly (i.e., we cannot efficiently calculate the likelihood of a sample, nor can we effectively use rejection sampling). Instead, we develop a two-stage constrained sampling process to generate from a distribution $Q' \sim \mathbf{P}$: a block-sampling stage to generate from the hierarchy, and an edge-sampling stage to generate edges, using constraints based on the maximum entropy principle [7]. The combination increases the accuracy of the sampling process, producing samples with correlated node-attribute values and complex network structure that match the marginal models.

To summarize, our contributions are twofold. First, we show both a discussion and empirical evidence of the reduction in sampling space that results from the use of a hierarchical GNM such as mKPGM—which makes it difficult to model $P(G, \mathbf{X})$ with the recently developed AGM sampling method. Second, to address the issue above, we propose CSAG, a 2-stage constrained sampling method that samples a set of blocks from a region of feasibility and then samples edges from the selected block-space. CSAG is appropriate to use with mKPGM and other hierarchical GNMs. We provide a detailed implementation of CSAG for mKPGMs. Our experiments on three real datasets show that CSAG outperforms the state of the art by jointly matching attribute correlations and network structure—using orders of magnitude fewer parameters than competing methods. The experiments also show a $\geq$ 5X reduction in the attribute correlation error with respect to the state-of-the-art.

[1] This likelihood depends on the goodness of fit of $P(G)$ and $P(\mathbf{X})$.

## 2 Background and Related Work

### 2.1 Generative Network Models

Let $G = (\mathbf{V}, \mathbf{E})$ be a graph with set of vertices $\mathbf{V}$ and edges $\mathbf{E} \subset \mathbf{V} \times \mathbf{V}$. Generative network models sample random networks $G$ from a distribution $P(G)$. Let $\mathcal{M}(\Theta^G)$ refer to a generative network model with parameters $\Theta^G$, which generates a network $G$ through a random sampling process as follows. We define $E_{ij}$ to be a binary random variable, where $E_{ij} = 1$ indicates that an edge between $V_i, V_j \in \mathbf{V}$ exists (i.e., $e_{ij} \in \mathbf{E}$), and $E_{ij} = 0 \Rightarrow e_{ij} \notin \mathbf{E}$. Many GNMs sample networks from a $|\mathbf{V}| \times |\mathbf{V}|$ probability matrix $\mathcal{P}$, where $\mathcal{P}_{ij} = p_{ij}$ is the probability of the edge $e_{ij}$ existing in the network, i.e., $P(E_{ij} = 1) = p_{ij}$. We will denote as $\mathcal{P}^+$ the subset $\mathcal{P}_{ij} > 0$. Some examples of generative network models appear below.

***Unique Bernoulli probabilities***. An important abstraction/property of GNMs that use the probability matrix $\mathcal{P}$ is the fact that they can be simply represented with the pair $< \mathbf{U}, \mathbf{T} >$. $\mathbf{U}$ is the set of unique Bernoulli probabilities that appear in the matrix $\mathcal{P}$, i.e. $\mathbf{U} = \{\pi_1, \pi_2, \ldots, \pi_u, \ldots\} = \texttt{unique}(\mathcal{P})$. The function `unique` returns the set of all $p_{ij}$ (i.e. without repetitions). $\mathbf{T}$ is a set of lists $\mathbf{T}_u$, which lists the positions where the probability $\pi_u$ appears in $\mathcal{P}$. Later we will see that $\mathbf{T}$ can be indexed also by other criteria (e.g. by the node attributes). For now, it is enough to understand that $u$ indexes $\mathbf{T}$ ($\mathbf{T}_u$).

***Erdős-Rényi Model (ER)*** [3]. In an ER model, random graphs are formed by creating edges between each pair of nodes independently with probability $p$. It is easy to see that $\mathbf{U} = \{\pi = p\}$ has only one unique probability and $\mathbf{T}$ consists of a single list with all the positions in $\mathcal{P}$.

***Chung-Lu Model (CL)*** [2]. In a CL model $\mathcal{P}_{ij} = w_i w_j / \sum w_k$ for a sequence of expected degrees of nodes $\mathbf{w} = (w_1, \ldots, w_{|\mathbf{V}|})$ (CL assumes $\max_k w_k^2 < \sum w_k$ so that $0 \leq \mathcal{P}_{ij} \leq 1$). In the worst case, although very unlikely, there could be $|\mathbf{V}|^2$ unique probabilities $\pi_i$ and $\mathbf{T}$ would consist of $|\mathbf{V}|^2$ lists of a single element each.

***Transitive Chung Lu Model (TCL)*** [17]. TCL is a generalization of CL. Both CL and TCL model the expected degree via a set of weights $w_i$ proportional to degree of node $i$. However, while CL samples each edge independently with probability proportional to $w_i w_j$, TCL rewires edges to match clustering coefficient of the original network. The same criteria applies for the size of $\mathbf{U}$ and $\mathbf{T}$ as for CL.

***Block two-level Erdős-Rényi Model (BTER)*** [18]. Sampling for BTER is done in two phases. First, affinity blocks

are created in a preprocessing step, where edges in each block are linked with an ER model. Second, connections between blocks are created with a CL model. Since the phase-2 uses a CL model $\mathbf{U}$ and $\mathbf{T}$ could have the same worst case scenario than CL.

***Kronecker Product Graph Model (KPGM)*** [11]. Given a $b \times b$ parameter matrix $\Theta^G$ where $\forall\, i,j\ \theta^G_{ij} \in [0,1]$, and $K$ (the number of Kronecker products), KPGM samples graphs as follows. First, it computes $\mathcal{P}$ a matrix equal to $K-1$ Kronecker products of $\Theta^G$ with itself. Second, it samples $G=(\mathbf{V},\mathbf{E})$ from $\mathcal{P}$ by sampling each cell independently from $Bernoulli(\mathcal{P}_{ij})$. In KPGM, $\mathbf{U}$ is $|\mathbf{U}| = \binom{b^2+K-1}{K}$ and $\mathbf{T}$ consists of $|\mathbf{U}|$ lists of edges (more details in [15]).

***mixed Kronecker Product Graph Model (mKPGM)*** [14]. mKPGM is a hierarchical generative network model that overcomes known limitations [19] of the KPGM. Given a $b \times b$ parameters $\Theta^G$ where $\forall\, i,j\ \theta^G_{ij} \in [0,1]$, the number of Kronecker products $K$, and the number of untied levels $\ell$, mKPGM samples a network as follows. First, it computes $\mathcal{P}^{[0]}$, a $b^\ell \times b^\ell$ matrix, equal to $\ell-1$ Kronecker products of $\Theta^G$ with itself, i.e., it computes a KPGM model with parameters $\Theta^G, \ell$. Second, it samples $G^{[0]}=(\mathbf{V}^{[0]},\mathbf{E}^{[0]})$ from $\mathcal{P}^{[0]}$ by sampling each cell independently via $Bernoulli(\mathcal{P}^{[0]}_{ij})$. Note, $|\mathbf{V}^{[0]}| = b^\ell$. Third, it calculates $\mathcal{P}^{[l]} = G^{[l-1]} \otimes \Theta^G$ and samples $G^{[l]}$ for $l=1 \ldots K-\ell$. The iterations increase the variability in the sampled network $G^{[K-\ell]}$ (where $|\mathbf{V}^{[K-\ell]}| = b^K$).

Since $G^{[0]} \ldots G^{[K-\ell-1]}$ represent auxiliary graphs (in matrices of increasing size), where each edge influences a *block* of possible edges in the next iteration of the hierarchical sampling process (and they are not present in the final output network) we refer to them as block matrices: $\mathbf{B}^{[0]} \ldots \mathbf{B}^{[K-\ell-1]}$. To simplify notation we will refer to the final sampled network $G^{[K-\ell]}=(\mathbf{V}^{[K-\ell]},\mathbf{E}^{[K-\ell]})$ as $G_{OUT}=(\mathbf{V}_{OUT},\mathbf{E}_{OUT})$. Figure 1 illustrates the hierarchical sampling process from $\mathcal{P}$s to $\mathbf{B}$s, of increasing size. In mKPGMs, the size of $\mathbf{U}$ is $b^2$ for each level of the hierarchy and $\mathbf{T}$ consists of $b^2$ lists of edges.

## 2.2 Attributed Networks and Related Work

When developing models to sample attributed networks, it is worth highlighting the importance of accurately modeling the correlation observed among the attributes of linked nodes (or *autocorrelation*). Autocorrelation is typically observed in networks due to processes such as *social influence* and *homophily* [13, 20], and is an important feature exploited in statistical relational learning to improve classification and other statistical inference tasks in network domains [4].

The Attributed Graph Model (AGM) [16] approximates sampling from the joint distribution of network structure and node attributes $P(G,\mathbf{X})$ using rejection sampling. AGM combines structure and attributes independently in a proposal distribution $Q = P(G)P(\mathbf{X})$, by first sampling the node attributes from a distribution $P(\mathbf{X};\Theta^X)$ with parameters $\Theta^X$ and then using a generative network model $P(G;\Theta^G)$ with parameters $\Theta^G$ to sample attributed edges, based on:

$$P(E_{ij}|\mathbf{X}) \approx P(E_{ij}|\Theta^G) \cdot A(f(\mathbf{x}_i,\mathbf{x}_j)|\Theta^G,\Theta^X)$$

where $P(E_{ij}=1|\Theta^G) = \mathcal{P}_{ij}$ is the edge probability defined by the generative network model and $A(\cdot)$ is the acceptance probability based on features between nodes $f(\mathbf{x}_i,\mathbf{x}_j)$. The acceptance probability ensures that we add edges $e_{ij}$ with correlated attribute values.

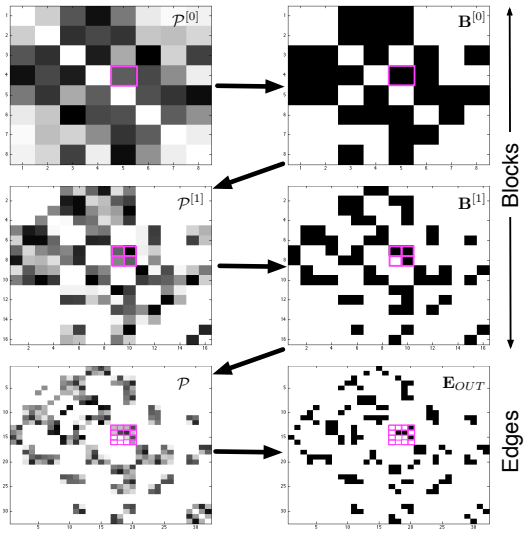


Figure 1: mKPGM sampling process with $b=2$, $\ell=3$, $K=5$. $\mathcal{P}^{[0]}$ is generated as a KPGM. Left: Matrices of probabilities $\mathcal{P}^{[l]}$ (white: $\mathcal{P}_{ij}=0$ black: $\mathcal{P}_{ij}=1$). Right: Adjacency matrices $\mathbf{B}^{[0]} \ldots \mathbf{B}^{[K-\ell-1]}, E_{OUT}$ (black $\Rightarrow$ block/edge sampled).

For training, AGM uses an input network $G_{IN}$ and attributes $\mathbf{X}_{IN}$, to learn $\Theta^X$, $\Theta^G$, and $A(\cdot)$. To illustrate AGM's process consider a network where nodes have a single attribute. AGM learns a GNM model from the network structure (i.e., $P(G)$), fits a Bernoulli distribution to the observed attribute values (i.e., $P(\mathbf{X})$), and learns $A(\cdot)$ by comparing the attribute correlations observed in $G_{IN}$ to those produced by an independent sampling process (i.e., $P(G)P(\mathbf{X})$). Then, AGM produces a new set of nodes, samples attribute values from $P(\mathbf{X})$ for each node, generates candidate edges from the GNM, and accepts/rejects based on the attributes of the incident nodes using $A(\cdot)$. AGM has been implemented for GNMs: FCL [17], TCL, and KPGM.

Another approach to generate samples of attributed-networks is to develop a probabilistic model that jointly incorporates both attributes and structure. There is work in this direction. One example is the *multiplicative attribute graph* (MAG) model [10]. However, the evaluation of AGM [16] showed that MAG underperformed AGM for sampling new networks because MAG considers latent node attributes instead of learning from observed node attributes.

As we will describe below, the insight of our CSAG method is that sampling from the full joint is difficult due to its high dimensional space, but much of the space has essentially zero probability in some hierarchical GNMs, such as mKPGMs. Our CSAG approach uses constrained sampling to effectively explore only parts of the network space that are likely. It discards random variables with zero probability and exploits information about the desired level of autocorrelation to further constrain the search using the maximum entropy (MaxEnt) principle. We will compare CSAG only to AGM since [16] showed that AGM was significantly better than MAG and other methods.

Finally, it is important to note some existing applications of MaxEnt to related problems. While it has not been used for sampling attributed networks, MaxEnt has been exten-

sively used in sampling of structure alone, and for inference in attributed networks. The earliest example of MaxEnt for sampling network structure is the Erdős-Rényi model [3]. Modeling and sampling of scale-free networks with MaxEnt was explored by [8], [5] studied anomaly detection (i.e., inference), and [21] learned the structure in attributed networks. We note that [21] is, however, not easy to extend to model the joint distribution for sampling because it prunes away data and scales poorly without this pruning.

## 3 Sampling Attributed Networks

### *Naive application of AGM with mKPGM*

A full description of the joint distribution of structure and attributes $P(G, \mathbf{X})$ remains an open problem [16] and AGM is one of the first methods to approximate attributed-network sampling (i.e. sampling from $P(G, \mathbf{X})$). Thus, one could naively apply AGM using mKPGMs as the component GNM as follows. First, learn an mKPGM from $G_{IN}$ to model $P(G)$, fit a distribution $P(\mathbf{X})$ to the observed attributes $\mathbf{X}_{IN}$, and learn $A(\cdot)$ as usual. Next, produce a new set of nodes and sample their attribute values from $P(\mathbf{X})$. Then, sample blocks from the mKPGM and generate candidate edges based on the sampled blocks (i.e., the results of the penultimate iteration of mKPGM sampling process: $\mathbf{B}^{[K-\ell-1]}$). Finally, accept/reject the candidate edges based on the attribute values of the incident nodes.

While this implementation will preserve the network structure modeled by the mKPGM, it will be difficult for the process to match the correlations observed in $G_{IN}$. AGM assumes the set of non-zero edge-probabilities in $\mathcal{P}$ where $p_{ij} > 0$, which we will denote $\mathcal{P}^+$, has a large cardinality and is as close as possible to $|\mathbf{V}|^2$ (i.e., $|\mathcal{P}^+| \simeq |\mathcal{P}|$). However, in mKPGMs there is a high probability that many blocks will not be sampled (see Fig. 1), thus $E[|\mathcal{P}^+|] << |\mathcal{P}|$, as shown by Lemma 1 below. The restricted space of possible edges in mKPGM limits the possibilities considered in the AGM's rejection process and as a result degrades its ability to match the targeted attribute correlations.

### *Impact of the Generative Sampling Space*

As we discussed before, mKPGM sampling comprises a hierarchical structure of edges, blocks, and super-blocks of edges. A block $ij$ at level $l$ has an associated random variable (r.v.) $B_{ij}^{[l]}$ which represents its state: sampled ($B_{ij}^{[l]} = 1$) or not ($B_{ij}^{[l]} = 0$). Here $l \in \{0, \ldots, K-\ell\}$, with $l = 0$ corresponding to the root of the hierarchy and $l = K-\ell$ to the level of edges. A block or edge can be sampled iff its parent is sampled—if a superblock is not sampled, sampling of sub-blocks or edges is inhibited. In this case, the conditional $P(E_{ij} \mid pa(E_{ij}) = 0) = 0$, but the marginal $P(E_{ij}) > 0$. As a consequence, mKPGMs have a reduced conditional space. Specifically, $|\mathcal{P}^+| < |\mathcal{P}|$ given the set of sampled blocks in the hierarchy of an mKPGM model:

**Lemma 1.** *Let $\mathcal{P}$ be the matrix of edge probabilities and $\mathcal{P}^+$ the matrix of non-zero edge-probabilities conditioned on the sampled blocks in the level $K-\ell-1$ of an mKPGM. If $\sum \Theta^G < b^2$, then $E[|\mathcal{P}^+|] << |\mathcal{P}|$, where $K-\ell-1$ is the last block-iteration (before the generation of edges) (figure 1).*

*Proof.* Let $\Theta_s = \sum \Theta^G$ be the sum of the mKPGM parameters, then $E[|\mathcal{P}^+|] = (\Theta_s)^{K-1} \cdot b^2$ and $|\mathcal{P}| = (b^2)^K$. Then $\frac{E[|\mathcal{P}^+|]}{|\mathcal{P}|} = \frac{(\Theta_s)^{K-1}}{(b^2)^{K-1}}$. Since $\Theta_s < b^2$, then $\frac{E[|\mathcal{P}^+|]}{|\mathcal{P}|} << 1$, i.e. mKPGM has a reduced sampling space. □

In sparse networks $|\mathbf{E}| = O(|\mathbf{V}|)$ and for KPGM-family models $E[|\mathbf{E}|] = (\Theta_s)^K$. Thus, $(\Theta_s)^K = O(|\mathbf{V}|) \Rightarrow (\Theta_s)^{K-1} \approx \frac{|\mathbf{V}|}{\Theta_s}$. Consequently, for mKPGMs $\frac{E[|\mathcal{P}^+|]}{|\mathcal{P}|} \approx \frac{|\mathbf{V}|/\Theta_s}{|\mathbf{V}|^2} = \frac{1}{\Theta_s|\mathbf{V}|}$. Our empirical analysis in Table 2 shows that $|\mathcal{P}^+|$ for mKPGMs is up to 6 orders of magnitude smaller than $|\mathcal{P}^+|$ for other GNM models. In general, for non-hierarchical GNMs: $|\mathcal{P}| \simeq |\mathcal{P}^+| \simeq |\mathbf{V}|^2$.

The reduced sampling space for mKPGMs constrains the ability to generate edges with correlation between the attributes of the incident nodes (e.g., autocorrelation). This is because the correlation is jointly produced by the attribute distribution and the sampling of edges from the GNM. When $|\mathcal{P}^+| < |\mathcal{P}|$, the potential edges may not exhibit the combinations of attribute values needed to produce (auto)correlation. For instance, if an unsampled block contain nodes with particular values of an attribute, the correlation that can be obtained will be determined by the values in the remaining blocks. In comparison to GNMs with matrices where $|\mathcal{P}^+| \simeq |\mathcal{P}|$, for iterative sampling methods like AGM, it will be difficult to produce the target level of correlation when using hierarchical GNMs like mKPGM.

### *Our approach*

As we describe next, we develop a 2-stage constrained sampling method to address the problem of sampling from $P(G, \mathbf{X})$ with hierarchical GNMs. First, we use linear programming (LP) to sample blocks from the latent space. Then, we sample edges from the sampled blocks. We use MaxEnt [7] via constraints, maintained through the method of moments.

## 4 CSAG Sampling Method

To solve the problem defined in Section 1, we propose a novel sampling method to draw the network structure conditioned on the attribute values. Specifically, we fit a distribution $P(\mathbf{X})$ to the attributes $\mathbf{X}_{IN}$, and estimate the structure $P(G)$ by learning a model $\mathcal{M}$ from $G_{IN}$, where $G_{IN}$ and $X_{IN}$ are the input data by problem definition. Then we use the structure conditioned on the attributes, as a proposal $Q = P(G)P(\mathbf{X})$. Although $Q$ is easy to sample from (because assumes independence) it is not a good approximation of the joint $\mathbf{P}$ because $P(G)$ and $P(\mathbf{X})$ are not independent. To solve this, we calculate the moments $\mathbf{f} = \{f_o(G_{IN}, \mathbf{X}_{IN})\}$ of the training data (attributes and structure), and sample a network from $Q$ that matches those moments i.e. we sample from $Q' \subset Q : Q' = P(G|\mathbf{X}, \mathbf{f})P(\mathbf{X})$.

The general goal of the algorithm is to maintain the marginals $P(G)$ and $P(\mathbf{X})$ and keep only the sampled networks that maintain the characteristics of networks in the joint distribution $\mathbf{P}$. In other words, from the candidate edges we keep only those that match the moments $\mathbf{f}$.

Here we describe the intuition behind our 2-stage sampling algorithm while its full description is provided in 4.1. First, let us define the moments used for sampling. The set of attributes $\mathbf{X} = \{\mathbf{X}^1, \mathbf{X}^2, ..., \mathbf{X}^m\}$ create the pairs $(\mathbf{x}_i^m, \mathbf{x}_j^{m'})$ for $e_{ij} \in \mathbf{E}$ $\forall i, j, m, m'$. We refer the set of unique labels $(\mathbf{x}_i^m, \mathbf{x}_j^{m'})$ as the edge-types $\Psi$ (e.g., for an undirected network with a single binary node attribute $\Psi = \{00, 01, 11\}$), and the fraction of edges of each type $\Psi$

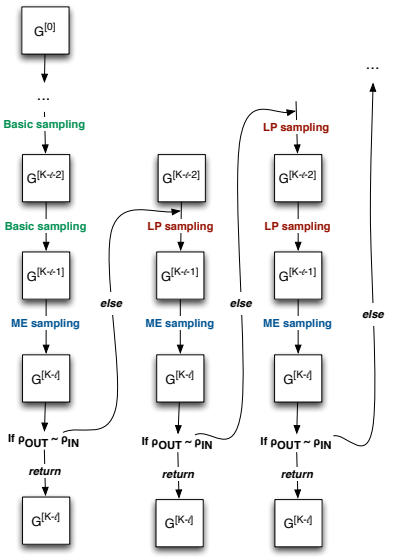


Figure 2: Example illustration of the sampling process which combines basic sampling with MaxEnt sampling, and LP search at different levels of the hierarchy, depending on whether the output network meets the desired constraints.

**Algorithm** *GraphSampling*

**Input:** $G_{IN} = (\mathbf{V}, \mathbf{E})$, node attrs $\mathbf{X}_{IN}$, GNM $\mathcal{M}$, error $\epsilon$
**Ouput:** $G_{OUT}, \mathbf{X}_{OUT}, \rho_{OUT}$
$[\Psi, \beta, \Theta^{\mathbf{X}}, \Theta^{G}] \leftarrow$ *LearnParameters*$(G_{IN}, \mathbf{X}_{IN})$
Sample $\mathbf{X}_{OUT}$ from $P(\mathbf{X}|\Theta^{\mathbf{X}})$
Initialize $\rho_{OUT} = \infty$ and $l_o = K - \ell - 1$
**while** $(|\rho_{IN} - \rho_{OUT}| > \epsilon)$ AND $(l_o \geq 0)$ **do**
  Sample $\mathbf{B}^{[l_o]}_{sample} \sim \mathcal{M}(\Theta^{G})$ using *basic sampling* [2]
  **for** $l = l_o + 1$ **to** $K - \ell - 1$ **do**
    $\mathbf{B}^{[l+1]}_{sample} \leftarrow$ *LPBlockSearch*$(\mathcal{M}, \Theta^{G}, \mathbf{B}^{[l]}_{sample}, \Psi, \beta)$
  $G_{OUT} \leftarrow$ *MEEdgeSampling*$(\mathcal{M}, \Theta^{G}, \mathbf{B}^{[K-\ell-1]}_{sample}, \Psi, \beta)$
  Calculate $\rho_{OUT}$ using $G_{OUT}$ and $\mathbf{X}_{OUT}$
  $l_o = l_o - 1$

in $G$ as $\beta = \{\beta_1, \ldots, \beta_{|\Psi|}\}$. The moments $\mathbf{f} = \beta$ model the Pearson correlation $\rho$ of the attribute vectors $\mathbf{X}^m = \{\mathbf{x}^m_i\}$ and $\mathbf{X}^{m'} = \{\mathbf{x}^{m'}_j\}$ over the pairs $e_{ij} \in \mathbf{E}\ \forall i, j$. When $m = m'$, this corresponds to the autocorrelation of a single attribute across edges in the network. We will denote the target correlation as $\rho_{IN}$ and the sample correlation as $\rho_{OUT}$. Since $\rho$ is function of $\beta$ (and $|\mathbf{E}|$) we will use $\beta$ instead of $\rho$ in our discussion. It is worth to note that $\rho$ is jointly defined by the GNM and the distribution of node attribute values. Thus in hierarchical GNMs where the graph structure is constrained, the possible values of $\rho$ in the sample are also constrained.

Given an input graph and its attributes $(G_{IN}, \mathbf{X}_{IN})$, and a GNM $\mathcal{M}$, we sample in a two stage process to generate the attributed network $(G_{OUT}, \mathbf{X}_{OUT})$ with correlation $\rho_{OUT}$, as outlined in Algorithm *GraphSampling*. The task of *GraphSampling* is to sample the network structure conditioned on the attribute values to match $\rho_{IN}$. The general idea of this two-stage sampling method is to apply moment matching whenever possible while sampling edges. When the sampling space is too restrictive however (i.e., it is not possible to sample edges with the target correlations), a second stage sampling is used to steer the search towards parts of the space with blocks of edges that make the target correlations more likely. Specifically, when the *basic unconstrained sampling* of the GNM[2] results in a space (of sampled blocks) where it is possible to match $\mathbf{f}$, then the first stage sampling (with MaxEnt moment matching) will be sufficient to sample edges with the target correlations. Otherwise, sampling will require a second stage linear program (LP sampling), to sample blocks before the first stage—replacing the basic sampling of that iteration. This is repeated for as many previous iterations as needed (see Figure 2). Since the number of iterations where second stage sampling is needed is generally small ($\leq K/2$), the process is relatively efficient.

The first-stage sampling, described by Algorithm *MEEdgeSampling*, is a general procedure to sample edges and works under the assumption that $\beta$ can be achieved with the sampled blocks. Consider the case where sampling from an mKPGM leads to a space where a network with the target correlation is feasible. In such case, the edges can be sampled from a set of blocks $\mathbf{B}^{K-\ell-1}_{sample}$ sampled at the iteration $K-\ell-1$ of the sampling process. The **constraints** that must be enforced to sample edges (at iteration $K-\ell$) are two-fold: (1) the *structure of the sampled network* is the one defined by the GNM, and (2) the *correlations obtained* are the same as the input correlations. CSAG constrains every step of the sampling process by including these two restrictions. The former restriction (to maintain the structure of the GNM) is guaranteed by using group probability sampling [15] as follows. Given the set of unique Bernoulli probabilities $\pi_u$ from the GNM[3], we need to guarantee that the resulting samples follow the exact same Bernoulli distribution of the edges. This is accomplished by ensuring the number of edges sampled per $\pi_u$ follows a binomial distribution, as proved in [15]. The later restriction (to maintain the correlation) is guaranteed by matching the moments $\beta_j$ of edges of each type $\psi_j$ to the moments of the input data.

The second-stage sampling, described in Algorithm *LPBlockSearch*, is used to draw blocks, i.e. it can be applied to *all iterations except the last*, as opposed to the previous stage that only applies to the *last iteration* for sampling edges. This stage is necessary to deal with cases where the network-space obtained using the generative network model does not allow for networks with the target correlation. In such cases the set of blocks from which edges will be sampled must be realized from regions of high(er) likelihood such that the edges sampled fulfill the constraints that produce the target correlation.

The objective in the second stage is to find the set $\chi$ of blocks to sample in a particular iteration. As we will describe, $\chi$ must fulfill **constraints** so that the blocks selected will maintain both the *network structure* of the GNM and the *attribute structure* of $P(\mathbf{X})$, based on the training data. To express these two types of constraints mathematically, we will consider certain conditions that the sampled blocks must follow. Thus, our objective function $f(\chi)$ is an error function (to minimize) that measures how far the samples are from a region that satisfies the conditions.

Since both sampling stages respect the original distribution of the GNM, the structure and the variability of the GNM is maintained in the output network. Both stages are also designed to sample from a region that has non-trivial

[2]Section 2.1 describes mKPGM's *basic unconstrained sampling*.
[3]As described in 2.1, GNMs are equivalent to a set of unique Bernoulli probabilities $\pi_u \in \mathbf{U}$ and their positions $\mathbf{T}_u$.

mass in the joint distribution—because many networks with high likelihood w.r.t. the GNM will typically not exhibit the desired attribute correlations.

### *Learning*

Learning $\Theta^{\mathbf{X}}$ and $\Theta^{G}$ is straightforward. In our evaluation, we estimate $\Theta^{\mathbf{X}}$ using the MLE of the multinomial distribution for $\mathbf{X}$, while for $\Theta^{G}$ we use the simulated method of moments of [14]. Likewise, estimating $\rho_{IN}$ is a trivial MLE. In our example, for instance, it simply consists in calculating the fraction of edges with labels $\Psi = \{00, 01, 11\}$.

**Algorithm** *LearnParameters*

1: **Input:** $G_{IN} = (\mathbf{V}, \mathbf{E})$, node attrs $\mathbf{X}_{IN}$, model $\mathcal{M}$
2: **Ouput:** $\Psi$, $\beta$, $\Theta^{\mathbf{X}}$, $\Theta^{G}$
3: Compute $\Psi$ and $\beta$ (that define $\rho_{IN}$) from $G_{IN}$ and $\mathbf{X}_{IN}$
4: Compute the MLE of $\Theta^{\mathbf{X}}$ using $\mathbf{X}_{IN} \sim f(X)$
5: Estimate $\Theta^{G}$ from $G_{IN} \sim \mathcal{M}$

## 4.1 Algorithmic details

Algorithm *GraphSampling* describes CSAG's overall framework. *GraphSampling* first determines the correlation $\rho_{IN}$ of the input graph (the edge-types $\Psi$, and the fraction of edges per type $\beta$), and learns $\Theta^{\mathbf{X}}$ and $\Theta^{G}$ for the attributes $P(\mathbf{X})$ and model $\mathcal{M}(\Theta^{G})$ (line 3). $\Theta^{\mathbf{X}}$ is estimated using the MLE of the multinomial distribution of $\mathbf{X}$, while $\Theta^{G}$ is estimated with the learning algorithm of the GNM (e.g. [14] is used to learn $\Theta^{G}$ for mKPGM). The algorithm samples attribute values $X_{OUT}$ for the output nodes with $\Theta^{\mathbf{X}}$ (line 4). $G_{OUT}$ is sampled using the two stage-sampling based on $\Theta^{G}$. The LP-search starts from the set of blocks $\mathbf{B}^{[l_o]}_{sample}$ sampled from $\mathcal{M}$ at layer $l_o$ (line 7). Then, it searches through the hierarchy ($l = l_o \ldots K-\ell-1$) to ultimately sample $\mathbf{B}^{[K-\ell-1]}_{sample}$ (lines 8-9). Then the method of moments uses $\mathbf{B}^{[K-\ell-1]}_{sample}$ to sample edges (line 10). $l_o$ moves one level up in the hierarchy if the space cannot match $\rho_{IN}$. This process is repeated until $|\rho_{IN} - \rho_{OUT}| \leq \epsilon$, for some predefined error $\epsilon$. The LP search is not realized if $\rho_{IN}$ is feasible to obtain (e.g. in non-hierarchical GNMs)—then $G_{OUT}$ is sampled directly. It is important to note that there are configurations of attribute values, and target correlations, for which sampling may not be possible. If this is the case, *GraphSampling* will not satisfy the constraints and the algorithm will terminate (with $l_o = 0$). In practice, because the networks are sparse and the attributes are low-dimensional it is generally possible to find a sample of edges that satisfy the constraints.

Algorithm *LPBlockSearch* describes the LP block-sampling stage. This stage focuses the sampling on relevant regions of the space where networks with the target correlation of node attributes are possible to be drawn. Given the GNM $\mathcal{M}$ with parameters $\Theta^{G}$, the set of sampled blocks $\mathbf{B}^{[l]}_{sample}$, the list of edge-types $\Psi$, and the fraction of edges of each type $\beta$, *LPBlockSearch* returns the set of sample blocks $\mathbf{B}^{[l+1]}_{sample}$ at level $l+1$ such that all possible edges descendent from $\mathbf{B}^{[l+1]}_{sample}$ have the same proportions as $\beta$. *LPBlockSearch* first calculates $\mathbf{U} = \{\pi_1, \ldots, \pi_k\}$, and $\mathbf{T}$—the set of unique probabilities of $\mathcal{M}$ and the locations of possible sample blocks per unique probability $\pi_u$, respectively, which can be generated by $\mathbf{B}^{[l]}_{sample}$. The FOR loop (lines 4-14) performs a linear search per $\pi_u$. Specifically, it calculates the number of blocks to be sampled (line 5). The next FOR loop computes

**Algorithm** *LPBlockSearch*

1: **Input:** $\mathcal{M}, \Theta^{G}, \mathbf{B}^{[l]}_{sample}, \Psi, \beta$
2: **Ouput:** $\mathbf{B}^{[l+1]}_{sample}$ sampled blocks in $l+1$:
3: $(\mathbf{U}, \mathbf{T})$=*getUniquePr_BLocations*$(\mathcal{M}, \Theta^{G}, \mathbf{B}^{[l]}_{sample})$
4: **for** $u = 1$ **to** $|\mathbf{U}|$ **do**
5: Draw $n_u \sim Bin(|\mathbf{T}_u|, \pi_u)$ {# of blocks to sample per $\pi_u$}
6: **for** $j = 1$ **to** $|\Psi|$ **do**
7: $\mathbf{e}_j = \beta_j \times n_u$ {fraction of possible edges leading to $\rho_{IN}$}
8: Determine $A_{jk}$ {# of descendent edges of type $\psi_j \in \Psi$ per position $t_k \in \mathbf{T}_u$}
9: **for** $j = 1$ **to** $N_\Omega$ **do**
10: $ub_j = \sum_{k=1}^{|\Psi|} A_{jk}$ {max # of sampled blocks per $A_{\cdot k}$}
11: Solve the LP of eq 1: find min $\chi$ using $n_u$, $\mathbf{e}$, $A$, and $ub$
12: **for** $j = 1$ **to** $N_\Omega$ **do**
13: $\mathbb{B}'_{sample}$=Randomly sample $\chi_j$ blocks from $ub_j$ places
14: $\mathbf{B}^{[l+1]}_{sample} = \mathbf{B}^{[l+1]}_{sample} \cup \mathbb{B}'_{sample}$

the fraction of edges per type $\psi_j \in \Psi$ that would lead to $\rho_{IN}$ (lines 6-7). Then, the algorithm determines the matrix $A$ with the number of descendent edges of type $\psi_j \in \Psi$ per position $t_k \in \mathbf{T}_u$ (line 8). Thus, $A$ is a look-ahead structure for the possible edges of the blocks sampled with probability $\pi_u$. In theory the dimension of $A$ is $|\Psi| \times |\mathbf{T}_u|$. However, in practice we take only the set of unique columns which makes $A$ much smaller ($|\Psi| \times N_\Omega$). Let $\chi$ be the indices of the blocks to be sampled, then $A \cdot \chi$ is (a $|\Psi|$-dimensional vector with) the total number of possible descendent edges per edge-type $\psi_j \in \Psi$. Next, the algorithm computes the maximum number of blocks that can be sampled per column in $A$, which we refer to as a configuration (lines 9-10). Using these restrictions, the number of edges to sample $\chi$ is obtained through LP search (line 11, we used Matlab's interior-point algorithm *fmincon* [9]). Given $\chi$, the last FOR loop (lines 12-14) samples the desired number of blocks per configuration (line 13), generating $\mathbf{B}^{[l+1]}_{sample}$ (line 14).

In summary, *LPBlockSearch* applies the following process per probability $\pi_u$.

*(a) Number of samples according to the GNM $P(G)$.* Since $\pi_u$ is Bernoulli, the number of blocks to sample per $\pi_u$ is $n_u \sim Bin(|\mathbf{T}_u|, \pi_u)$, where $\mathbf{T}$ is the *set* of locations of possible blocks per unique probability $\pi_u$

*(b) Number of samples according to correlation* in $P(\mathbf{X})$. Let $\mathbf{e}_j = \beta_j \times n_u$ be the fraction of edges per type $\psi_j \in \Psi$ that would lead to $\rho_{IN}$. Let $A \cdot \chi$ be (a $|\Psi|$-dimensional vector with) the total number of possible descendent edges per edge-type $\psi_j \in \Psi$, where matrix $A$ is a look-ahead structure on the possible edges for the blocks sampled with probability $\pi_u$. Then, $e_j - A_{j\cdot} \cdot \chi$ must be as small as possible $\forall j \in [1, |\Psi|]$; where $\chi$ is a $N_\Omega$-dimensional vector with the set of blocks to sample per configuration $A_{\cdot k}$. The value of $\chi$ can be found by minimizing the objective function:

$$\begin{aligned} \text{minimize} \quad & f(\chi) = \sum_{j}^{|\Psi|} (\mathbf{e}_j - A_{j\cdot} \cdot \chi) \times \mathbb{1}[(\mathbf{e}_j - A_{j\cdot} \cdot \chi) > 0] \\ \text{subject to} \quad & \textstyle\sum_j \chi_j = n_u \\ & 0 \leq \chi_j \leq ub_j \quad \forall j \end{aligned} \tag{1}$$

Minimizing $f(\chi)$ gives a solution where the fraction of possible descendent edges per type is similar to the fraction of edges $\beta$ of the input network, which leads to the target correlation $\rho_{IN}$. Notice $f(\chi)$ is the summation of the differences

**Algorithm** *MEEdgeSampling*

1: **Input:** $\mathcal{M}, \Theta^G, \mathbf{B}_{sample}^{[K-\ell-1]}, \Psi, \beta$
2: **Output:** $G_{OUT}$
3: $(\mathbf{U}, \mathbf{T})$=*getUniquePr_ELocations*$(\mathcal{M}, \Theta^G, \mathbf{B}_{sample}^{[l]}, \Psi)$
4: **for** $u = 1$ **to** $|\mathbf{U}|$ **do**
5: Draw $n_u \sim Bin(|\mathbf{T}_{u\cdot}|, \pi_u)$ {# edges per unique probabil}
6: Set $N_e = N_e + n_u$ {Total # edges to be sample}
7: Draw $\mathbf{\Gamma} = [\gamma_1, \ldots, \gamma_{|\Psi|}] \sim Mult\,(N_e; \beta)$ {# edges per edge-type to match $\rho_{IN}$}
8: **for** $u = 1$ **to** $|\mathbf{U}|$ **do**
9: Draw $\mathbf{Y} = [Y_1, ..., Y_{|\Psi|}] \sim Mult\left(n_u; \frac{\mathbf{\Gamma}}{N_e}\right)$ {# edges per edge-type for $\pi_u$}
10: **for** $j = 1$ **to** $|\Psi|$ **do**
11: $\mathbf{E}' =$ Sampling $Y_j$ edges at random from $\mathbf{T}_{uj}$ locations.
12: $\mathbf{E}_{OUT} = \mathbf{E}_{OUT} \cup \mathbf{E}'$
13: $\gamma_j = \gamma_j - Y_j$ {Adaptative process to match the moments}
14: $N_e = N_e - Y_j$

between the expected number of possible edges according to $\beta$ and the actual number of descendent edges per edge-type. If $(\mathbf{e}_j - A_{j\cdot} \cdot \chi)$ is negative, the value $(\mathbb{1}[(\mathbf{e}_j - A_{j\cdot} \cdot \chi) > 0])$ is zero and the term is not considered (i.e., enough edges exist satisfying the condition). This forces the function to apply constraints for all conditions where not enough edges exist.

The constraint $\sum_j \chi_j = n_u$ guarantees that the sum of the number of blocks to sample, per edge type, is equal to the desired number of blocks to sample $n_u$. This constraint forces sampling a network with a high likelihood with respect to $P(G|\mathcal{M}(\Theta^G))$, because $n_u$ edges are sampled from the unique $\pi_u$. The inequalities $0 \leq \chi_j \leq ub_j \ \forall j$ ensure that $\chi$ is a valid solution, i.e. each $\chi_j$ is greater than zero and less than the number of possible blocks.

Algorithm *MEEdgeSampling* describes the edge sampling stage. In this stage of the sampling process, the network is sampled by drawing edges from the graph-model $\mathcal{M}$ conditioned on the node attributes. The edges accepted must respect a sampling-invariant of the sampling-process, namely, that the fraction of edges of each type are the same as those in the input-network. Given the GNM $\mathcal{M}$ with parameter $\Theta^G$, the set of sample blocks $\mathbf{B}_{sample}^{[K-\ell-1]}$, the list of edge-types $\Psi$, and the fraction of edges of each type $\beta$, *MEEdgeSampling* returns the sampled graph $G_{OUT}$. *MEEdgeSampling* calculates $\mathbf{U} = \{\pi_1, \ldots, \pi_k\}$ and $\mathbf{T}$—the set of unique probabilities and the locations of possible sample edges. In order to optimize sampling we transform $\mathbf{T}$ from a vector to a matrix indexed not only per unique probability $\pi_u$ but also per edge type $\psi_j \in \Psi$. Thus, $\mathbf{T}_{uj}$ contains a matrix with the position of edges sampled with probability $\pi_u$ that correspond to edge type $\psi_j$, i.e $\dim(\mathbf{T}) = |\mathbf{U}| \times |\Psi|$. Then, *MEEdgeSampling* computes the number of edges $n_u$ per unique probability and the total number of edges $N_e$ to sample in the final network (lines 4-6). With $N_e$ and $\beta$, it determines $\mathbf{\Gamma}$ (the total number of edges per type that must be sampled to match $\rho_{IN}$). Edge sampling is realized by the FOR loop (lines 8-14). This loop calculates the number of edges per edge-type $\mathbf{Y}$ to be sampled given a unique probability $\pi_u$ (line 9). This matches moments by sampling networks with the original proportion of edges $\beta$, i.e. $\rho_{IN}$. Lines 10 to 14 sample edges per edge-type given $\mathbf{Y}$. Specifically, $Y_j$ edges are sampled from the $|\mathbf{T}_{uj}|$ possible locations with probability $\pi_u$ and edge-type $\Psi_j$. The sampled edges are added to the sampled $G_{OUT}$. Finally, the number of edges per edge-type $\gamma_j$, and $N_e$ are updated to avoid possible errors given the multinomial sampling process (lines 13-14). Lines 13 and 14 are crucial to generate networks able to match $\rho_{IN}$ for all unique probabilities and edge-types, unless the number of edges to sample ($\mathbf{Y}$) is smaller than the possible positions $T_{uj}$. For a unique probability $\pi_u$ consider $Y_j > T_{uj}$, (i.e. it is impossible to sample the required number of edges). These lines distribute $Y_j - T_{uj}$ edges among the other edge-types, to obtain the number of edges $n_u$ for $\pi_u$. Since other edge-types may be oversampled for $\pi_u$, the lines recalculate the proportion of edges for the next unique probabilities such that $\rho_{OUT} \approx \rho_{IN}$.

In summary, algorithm *MEEdgeSampling* performs the edge sampling stage, as follows:

*(a) The structure of the GNM $P(G)$* is maintained by sampling the target number of edges $n_u \sim Bin(|\mathbf{T}_{u\cdot}|, \pi_u)$, because $\pi_u$ is Bernoulli-distributed [15].

*(b) The attribute distribution $P(\mathbf{X})$* is guaranteed by maintaining the correlation as defined by $\beta$, $N_e$: $\mathbf{\Gamma} = [\gamma_1, \ldots, \gamma_{|\Psi|}] \sim Mult\,(N_e; \beta)$.

These two criteria are combined to sample the number of edges per unique probability $\pi_u$ and per type $\psi_j$: $\mathbf{Y} = [Y_1, ..., Y_{|\Psi|}] \sim Mult\left(n_u; \frac{\mathbf{\Gamma}}{N_e}\right)$.

As in *LPBlockSearch*, $G_{OUT}$ has high probability with respect to $P(G|\mathcal{M}(\Theta^G))$. By respecting the original distribution of the GNM, the structure is maintained as well as the variability of the sample networks.

# 5 Experimental Results

We compare our sampling method, CSAG, using mixed Kronecker Product Graph Model (CSAG-mKPGM), with AGM-mKPGM, AGM Chung Lu (AGM-CL), and AGM Transitive Chung Lu (AGM-TCL)[4]. We note that AGM-TCL is the state of the art, outperforming AGM-CL and MAG [16]. However, AGM-TCL can not be applied to directed networks [17]. The results confirm that CSAG-mKPGM accurately captures the structure and correlation of the input networks, while creating networks with enough variability.

## 5.1 Datasets

We evaluate our framework on three real world datasets: CoRA citations network [12], Facebook wall postings from Purdue University, and the US patent citation network [6]. The CoRA network comprises 11,881 papers with 31,482 citations between them. We use the categorical attribute *AI* (*AI*=1 iff a paper's topic is in the field of Artificial Intelligence). The Facebook network contains wall postings of 449,748 users with 1,016,621 posting edges. We use two profile attributes: Religion (*R*=1 indicates a user's *Religious Views* contains "christ") and Politics (*P*=1 indicates a user's *Political Views* contains "conservative"). The US patent citation network contains 3,774,768 patents with 16,528,948 citation edges. We use the attribute Category with six possible values: Chemical, Computers and Communications, Drugs and Medical, Electrical and Electronic, Mechanical, and Others. For all networks, we model the attribute distributions with multinomials and use MLE estimation.

[4]Code and data available at www.dropbox.com/s/rdppd1qvqwjgj29/CSAG.zip?dl=0

Table 1: Correlation modeled for CoRA, Facebook, and Patents using AGM and CSAG. Best performers: Values closest to original are bolded. Values corresponding to non-rejected hypothesis under $\alpha = 0.01$ are marked with *

| Model | Correlations | | | | |
|---|---|---|---|---|---|
| | CoRA | Facebook | | | Patent |
| | AI | R | P | RP | |
| Original | 0.8373 | 0.1319 | 0.2231 | 0.1873 | 0.7097 |
| AGM-TCL | 0.8388* | 0.1369 | 0.2364 | 0.1945 | – |
| AGM-CL | – | – | – | – | **0.7111** |
| mKPGM | 0.2884 | 0.0311 | 0.0076 | 0.0391 | 0.6310 |
| AGM-mKPGM | 0.6945 | 0.1253 | 0.0286 | 0.0807 | 0.7260 |
| CSAG-mKPGM | **0.8370*** | **0.1317*** | **0.2225*** | **0.1870*** | 0.7074 |

## 5.2 Models

Since AGM is the best performer in [17], we compare CSAG-mKPGM against AGM-mKPGM and AGM-TCL (AGM-CL was used for directed network). We use TCL as described in [17]. We trained mKPGM using the simulated method of moments [14] with the following features: average number of edges, clustering coefficient, geodesic distance (approximated by a sample of nodes), size of largest connected component, and number of non-zero-degree nodes. Table 3 shows the values of the learned parameters. As we will discuss, due to its use of mKPGM, CSAG requires significantly fewer parameters than AGM.

Our sampling method introduces a constraint on the order of attribute values that are generated for the output nodes. To understand the rationale for this choice, we note that in practice random attribute samples can result in a scenario where networks with the desired structure and correlation have probability close to zero due to the specifics of $\Theta^G$. This is because the GNMs create edges at particular locations in the adjacency matrix. Thus it can be difficult for CSAG to produce high correlations with arbitrary orderings of the attribute values (over the random nodes). While this issue could be solved more generally by iteratively reassigning attribute values as part of the search process, for this work we randomly generated the attribute values and then partially sorted (over the nodes in the matrix) before randomly sampling the edges. Sorting starts the sampling from a location that might likely, although not always, result in a successful search. Although this process introduces a bias in a selection of edges geared to match the target correlation, sorting the labels does not affect the distribution nor the sampled graph because the nodes are all random.

## 5.3 Evaluation and Results

With the learned parameters (Table 3), we sampled 50 networks for CoRA with attribute AI, 20 networks for Facebook with both attributes (Religion and Politics), and a single network for Patent because of AGM's run-time limitations. We analyze both the attribute correlations and the graph structures of the sampled networks.

Figure 3 summarizes both network structure and attribute correlation errors in a single plot. The x-axis comprises the 3-dimensional Kolmogorov-Smirnov distance of the CDFs of the three network characteristics described before, between the models and the original data. The y-axis comprises the relative error of the correlation, i.e. $|\rho_{IN} - \rho_{OUT}|/\rho_{IN}$. As the figure shows, CSAG (enclosed by the triangle) has the lowest errors in all three datasets for both graph-structure and attribute correlation. As we will discuss, CSAG also provides relatively faster performance than AGM and uses orders of magnitude fewer parameters.

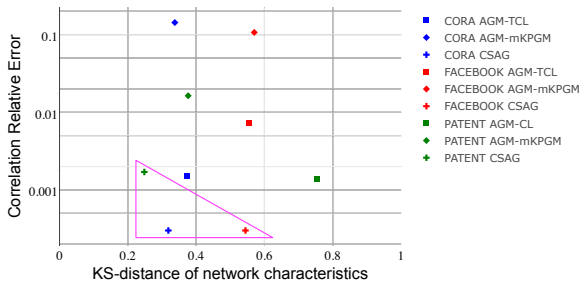


Figure 3: Performance of methods (shape-coded) per dataset (color-coded). Values towards the bottom-left are better.

Table 2: Non-zero edge-probabilities for mKPGM and TCL/CL modeling the three datasets.

| Dataset | count of $p(e_{ij}) \neq 0$ | |
|---|---|---|
| | mKPGM | CL and TCL |
| CoRA | $2.60 \times 10^5$ | $1.41 \times 10^8$ |
| Facebook | $6.35 \times 10^6$ | $2.02 \times 10^{11}$ |
| Patent | $4.53 \times 10^7$ | $1.42 \times 10^{13}$ |

### 5.3.1 *Feature Correlations*

Our first set of evaluations are based on the mean attribute-correlation of the sampled networks. The measurements of the mean-correlation are shown in Table 1. These results show that, though mKPGM has a very restricted sample space (Table 2), in all cases CSAG-mKPGM can capture attribute correlation. Indeed, CSAG-mKPGM is the best model to capture correlation except for AGM-CL in Patents data where CSAG-mKPGM gives competitive performance. The results confirm the importance of our CSAG framework to search over the space of possible networks to sample edges that match the target correlations. Moreover, we can observe that AGM-mKPGM performs badly when AGM's assumptions are violated, producing quite different correlations from the input. We verified these results statistically. Since population variance is unknown for the original networks, we used a T-test with $\alpha = 0.01$ and null hypothesis $H_0$: true correlation=model's correlation (assuming the true correlation to be that of the original network). $H_0$ is never rejected for our framework while $H_0$ is rejected for TCL (values with non-rejected $H_0$ are marked with $*$ in Table 1). The test cannot be applied to Patents because there is only one network available. This confirms that CSAG in combination with mKPGM provides the closest match to the correlations in the original network datasets.

### 5.3.2 *Graph Structure*

In our second set of evaluations, we investigated whether the models capture three important structure characteristics of real networks: degree, clustering coefficient (CC), and hop-plot (HP). To evaluate the ability of the models to capture these characteristics, we compared the cumulative distribution functions (CDFs) of the sampled vs. the original networks, visualizing not only the median but the variability of the sampled networks through the plot of the first and ninety nine percentiles (in the case of CoRA and Facebook). Additionally, we compared two more undirected-network characteristics, not included in the mKPGM learning process: the K-core distribution [1] and eigenvalue distances. We measured the maximum absolute distance between the K-core

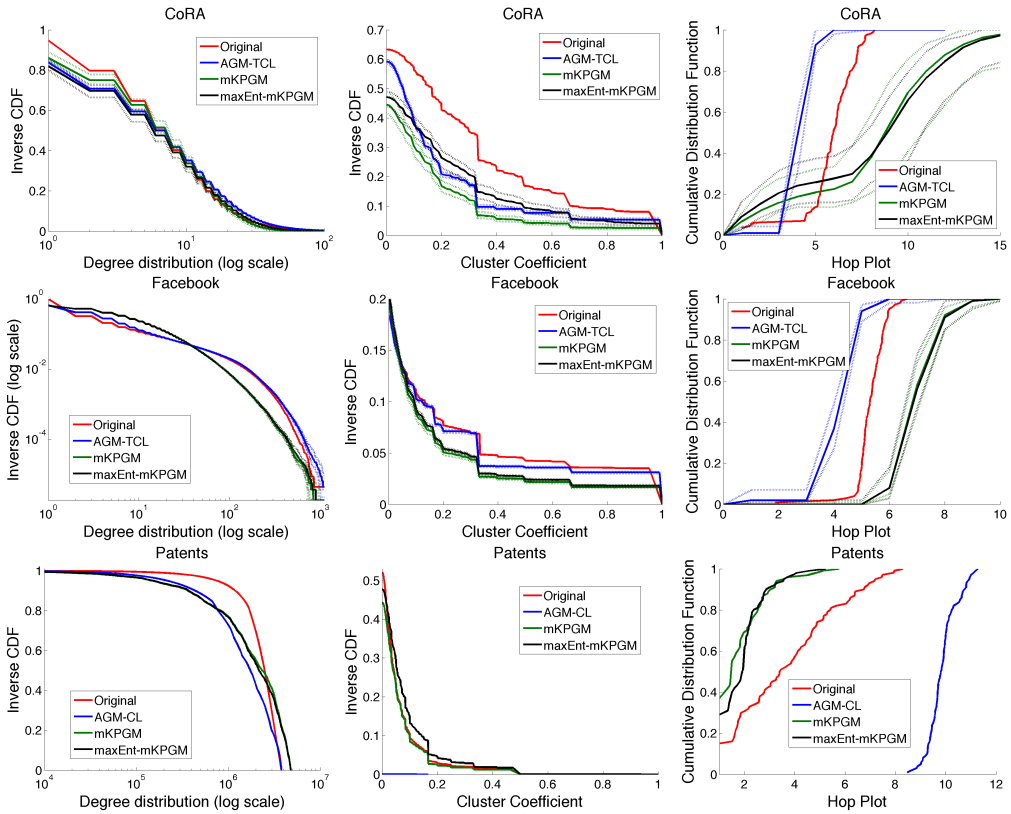


Figure 4: CDF of network characteristics (median, $1^{st}$, and $99^{th}$ percentile). Left-to-right: degree, cluster coefficient, and hop plot. Top-to-bottom: CoRA, Facebook, and Patents.

Table 3: Learned parameters for mKPGMs in CoRA: $K = 9, \ell = 6$, Facebook: $K = 12, \ell = 9$, and Patent: $K = 14, \ell = 10$

$$\Theta^G_{CoRA} = \begin{bmatrix} .94 & .01 & .03 \\ - & .86 & .40 \\ - & - & .95 \end{bmatrix} \quad \Theta^G_{Facebook} = \begin{bmatrix} .99 & .01 & .07 \\ - & .01 & .65 \\ - & - & .98 \end{bmatrix} \quad \Theta^G_{Patents} = \begin{bmatrix} .99 & .01 & .01 \\ .01 & .81 & .01 \\ .46 & .01 & .98 \end{bmatrix}$$

CDFs of the sampled vs. the original networks, and compared the largest 10 eigenvalues of each network using the standardized absolute distance between them.

Figure 4 shows the cumulative distribution function (CDF) of three different characteristics. We compare CSAG with the best performing AGM (i.e. AGM-TCL, which outperforms MAG [10]). The results show CSAG with mKPGM captures most characteristics of the real networks using only 9 parameters. Moreover, the results also show variability in the generated networks, as can be observed in the dashed lines of the plots in the first two rows, which represent the $1^{st}$ and $99^{th}$ percentile of the set of samples. (Recall, we generated only a single network for Patents due to AGM's restrictive sampling time.)

While AGM also captures the characteristics of the original networks, AGM requires 11881, 449748, and 3774768 parameters to model CoRA, Facebook, and Patents respectively. One exception is the surprisingly poor fit of AGM for Patents HP and CC (recall, we did not use TCL because it is only defined for undirected networks). As expected, AGM-TCL also shows significantly less variability than CSAG with mKPGM.

We further analyze the variability in network characteristics in two ways. First, we compare CSAG-mKPGM's with mKPGM's variance, using an F-test ($\alpha = 0.01$) for distinct values (vertical slice) of the characteristics in Figure 4. We assume normality for each slice. In this case $H_0$: the variance of a model is equal to the variance of mKPGM. For CC, HP, and degree CSAG fails to reject $H_0$ (i.e. maintains mKPGM's variance) for 100% 80% and 91% of the slices for CoRA. Facebook's results were similar. Second, we asses the variance reduction of AGM vs. CSAG. AGM's average variance reduction (compared to CSAG) is as follows: (1) For CoRA, the CC, degree, and HP, is reduced by 88%, 76%, and 70% respectively; (22) For Facebook, CC is reduced by 19%, but degree and HP are similar for AGM and CSAG.

Table 4: K-core and eigVal distances [**undirected** networks] CoRA and Facebook (smallest distances to original bolded).

| Model | CoRA | | Facebook | |
|---|---|---|---|---|
| | K-core | eigenvalue | K-core | eigenvalue |
| AGM-TCL | 0.1069 | 0.5438 | **0.3281** | 0.6608 |
| mKPGM | **0.0838** | 0.2737 | 0.3398 | **0.2707** |
| AGM-mKPGM | 0.1073 | 0.3059 | 0.3458 | 0.2825 |
| CSAG-mKPGM | 0.1255 | **0.1647** | 0.3409 | 0.3008 |

Table 4 shows the K-core and eigenvalue distances for CoRA and Facebook using all models. The K-core distances for CoRA and Facebook are similar across models, confirming that both models (mKPGM and TCL) show similar connectivity and community structure in the sampled graphs compared to the original network. In contrast, mKPGMs have considerably smaller eigenvalue distance than AGM-TCL. In fact, CSAG-mKPGM obtains the lowest eigenvalue distance across models in CoRA, which indicates the similarity of the most important nodes compared to the original network. In contrast, AGM-TCL shows surprisingly high

eigenvalue distances for both CoRA and Facebook. Once again, there are no results for Patents dataset because these measures are calculated only for undirected networks.

Finally, sampling with CSAG is faster than with AGM. For Facebook the mean $\pm 1$ stdev runtimes using a MATLAB implementation on a laptop with 2.8 GHz Intel Core i7 and 16 GB in RAM are: CSAG:$12.42 \pm 0.55$s vs AGM:$59.15 \pm 4.07$s, respectively. In the more restricted scenario of CoRA, AGM is faster (CSAG:$9.9 \pm 0.65$s vs AGM:$1.5 \pm 0.15$s) but AGM does not match the correlation. CSAG is an order of magnitude faster in Patents (CSAG:100.7s vs AGM: 1361.6s).

In summary, the results show that CSAG achieves: (1) Overall best fit of attribute correlation (smallest relative error) and network structure (smallest KS-distance). Specifically, CSAG produces $\geq$ 5X reduction in attribute correlation error compared to AGM-TCL. Moreover, AGM-TCL cannot model correlation consistently and cannot be used with directed networks; (2) More effective sampling—the exponential complexity of sampling from the full joint is relieved in CSAG by having a proposal distribution and few rejected samples. This is reflected in the improvement in runtime compared to AGM; (3) Variability of the GNM is maintained for $80\% - 100\%$ of the values of the characteristics; (4) Fewer parameters required compared to AGM.

## 6 Discussion and Conclusion

There are two main contributions of this paper. First, we showed that hierarchical GNMs, such as mKPGM, have a limited size of the matrix of conditional edge-probabilities which restricts iterative sampling of edges with targeted attribute correlation. This constrains AGM's performance as a result. Second, we proposed CSAG, a novel method to samples networks using mKPGM as the structural component of a proposal distribution. Our method uses a two-stage constrained sampling: an LP stage to sample blocks and a moment matching stage to sample edges. While the LP stage is strictly for models with constrained space, the moment matching is general enough to work with other types of models as well. CSAG's implementation with mKPGM provides the following benefits, as verified experimentally with 3 real datasets: (1) It jointly models network structure and correlated attributes, obtaining cumulative distribution functions (CDFs) and correlations close to the original networks. (2) It does not affect the network variability of the original GNM, as observed in the CDFs. (3) It captures the attribute correlations better than AGM, when using the same GNM.

## Acknowledgements

The authors thank the reviewers for their useful comments. This research is supported by NSF under contracts IIS-1149789, IIS-1219015, and CCF-0939370. Sebastian Moreno acknowledges the support of "CONICYT + PAI/Concurso nacional de apoyo al retorno de investigadores/as desde el extranjero, convocatoria 2014 + folio 82140043".

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

[3] P. Erdos and A. Renyi. On the evolution of random graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[4] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Relational Data Mining*, pages 307–335. Springer-Verlag, 2001.

[5] Y. Gu, A. McCallum, and D. Towsley. Detecting anomalies in network traffic using maximum entropy estimation. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, IMC '05, pages 32–32, Berkeley, CA, USA, 2005. USENIX Association.

[6] B. Hall, A. Jaffe, and M. Trajtenberg. The NBER patent citation data file: Lessons, insights and methodological tools. *NBER working paper 8498*, 2001.

[7] E. T. Jaynes. On the rationale of maximum-entropy methods. *Proceedings of the IEEE*, 70(9):939–952, June 1982.

[8] K. Judd, M. Small, and T. Stemler. What exactly are the properties of scale-free and other networks? *EPL (Europhysics Letters)*, 103(5):58004, 2013.

[9] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, STOC '84, pages 302–311, 1984.

[10] M. Kim and J. Leskovec. Multiplicative attribute graph model of real-world networks. In *Algorithms and Models for the Web-Graph*, volume 6516 of *LNCS*, pages 62–73, 2010.

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

[12] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[13] M. McPherson, L. Smith-Lovin, and J. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27:415–445, 2001.

[14] S. Moreno, J. Neville, and S. Kirshner. Learning mixed kronecker product graph models with simulated method of moments. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1052–1060, 2013.

[15] S. Moreno, J. Pfeiffer III, S. Kirshner, and J. Neville. A scalable method for exact sampling from kronecker family models. In *IEEE 14th International Conference on Data Mining (ICDM)*, Dec 2014.

[16] J. J. Pfeiffer, III, S. Moreno, T. La Fond, J. Neville, and B. Gallagher. Attributed graph models: Modeling network structure with correlated attributes. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 831–842, 2014.

[17] J. J. Pfeiffer III, T. La Fond, S. Moreno, and J. Neville. Fast generation of large scale social networks while incorporating transitive closures. In *4th ASE/IEEE International Conference on Social Computing*, 2012.

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

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

[20] C. R. Shalizi and A. C. Thomas. Homophily and contagion are generically confounded in observational social network studies. *Sociological methods & research*, 40(2):211?239, May 2011.

[21] E. Spyropoulou, T. De Bie, and M. Boley. Interesting pattern mining in multi-relational data. *Data Mining and Knowledge Discovery*, 28(3):808–849, 2014.