The HyperKron Graph Model for higher-order features

Nicole Eikmeier and David F. Gleich
Purdue University
West Lafayette, USA
Email: {eikmeier, dgleich}@purdue.edu

Arjun S. Ramani
Stanford University
Stanford, USA
Email: aramani3@stanford.edu

***Abstract*—In this manuscript we present the HyperKron Graph model: an extension of the Kronecker Model, but with a distribution over hyperedges. We prove that we can efficiently generate graphs from this model in time proportional to the number of edges times a small log-factor, and find that in practice the runtime is linear with respect to the number of edges. We illustrate a number of useful features of the HyperKron model including non-trivial clustering and highly skewed degree distributions. Finally, we fit the HyperKron model to real-world networks, and demonstrate the model's flexibility with a complex application of the HyperKron model to networks with coherent feed-forward loops.**

***Keywords*-kronecker graph, graph model, hyperedges**

I. Introduction

One of the long-running challenges with network analysis is that there remains a gap between the features of real-world network data and the types of network data that are produced by most efficient synthetic network generators. For instance, simple models such as configuration model [4] and Chung-Lu [1] are designed to capture the degree distribution of a network, but typically fail to capture any higher-order pattern such as a clustering coefficient. Conversely, models that are designed to capture arbitrary features including clustering, such as exponential random graph models, often have exponential computational complexities due to the difficulty of the sampling procedure [7], [16]. More structural models, such as stochastic block model, are often designed to test extremely specific hypotheses involving communities and may not be appropriate as more general models. Pragmatic models such as BTER explicitly place clustering in a carefully designed pattern [24] at the cost of a larger description of the network.

Recently, there has been a surge of interest in higher-order network analysis [5], [6], [18], [34], [39], [40]. At a high-level, this constitutes an analysis of network of data in terms of multi-node patterns such as motifs and also in terms of stochastic processes that depend on more history. One of the origins of these studies is Milo's celebrated paper on the presence of higher-order interactions [27], which showed that some subgraphs appear more frequently than others. While there are plenty of network models (such as those mentioned above), the efficient ones often cannot model arbitrary higher-order interactions such as motifs and their interactions.

The primary contribution of this manuscript is a simple and flexible network model that has the ability to capture a single type of higher-order interaction. We call it the HyperKron model. (This is introduced formally in §III.) As might be guessed from the name, the model is a generalization of the extremely parsimonious Kronecker graph model [25], [26], [35]. Instead of edges, it uses hypergraph modeling [8] to directly model the higher-order interactions—which is where our inspiration came from. In comparison with many of the network models above, the key difference is that the probability model underlying it specifies a distribution on *hyperedges* rather than edges. We then associate each hyperedge with a specific network motif (such as a triangle or feed-forward motif).

One of the challenges with this model is that an exact and efficient sampling procedure for the desired hyperedge probability distribution is non-trivial to create. We explain our efficient procedure in §IV. We show when using a 3 node motif, the HyperKron model generates substantial triadic clustering in fitting real-world network data, far beyond what is possible with Kronecker models, but *lacks* clustering structure in four cliques that is present in real-world networks (§VI-A). We finally show that the model is flexible enough to model directed and signed interactions. (§VII).

II. Preliminaries

Here we present the background, terminology, and notation to understand the HyperKron model presented in §III.

Graphs and matrices. Let $G = (V, E)$ be an unweighted, undirected graph, where V is the set of vertices and E is the set of edges. Graphs are associated with adjacency matrices $\boldsymbol{A}$, where $\boldsymbol{A}_{ij}$ is equal to 1 if i and j are connected by an edge, and 0 otherwise, and also $\boldsymbol{A}_{ij} = \boldsymbol{A}_{ji}$.

Sampling Graphs from Probability Matrices. There is a large body of work on modeling graphs, many of which were mentioned in the introduction and further detailed in §VIII. What is relevant here is the class of graph generators which involves sampling edges from a probability matrix. Examples include the Erdős-Rényi model [14], the Chung-Lu model [12], the Stochastic Block Model [21], and the Kronecker Model [11], [25], [26]. These generators begin with a matrix of probabilities, $\boldsymbol{P}$, with the number of rows and columns equal to the number of nodes desired in the graph. For each entry i, j of $\boldsymbol{P}$, set $\boldsymbol{A}_{ij}$ equal to 1 with probability $\boldsymbol{P}_{ij}$, and set $\boldsymbol{A}_{ij}$ equal to 0

978-1-5386-9159-5/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDM.2018.00115

otherwise. This allows for generating many instances of a graph from a single generator matrix $\boldsymbol{P}$.

Kronecker Graph. The HyperKron model is built on many of the same motivations of the Kronecker Graph Model [11], [25], [26], so we briefly cover that model. Let $\boldsymbol{P}$ be an $n \times n$ matrix of probabilities called an *initiator* matrix, with n small (n between 2-5 is typical). The Kronecker product of $\boldsymbol{P}$ with itself is the $n^2 \times n^2$ matrix constructed by multiplying every entry of $\boldsymbol{P}$ with itself. For example, if $\boldsymbol{P}$ is the 2×2 matrix $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$, then the Kronecker product $\boldsymbol{P} \otimes \boldsymbol{P}$ is

$$\boldsymbol{P}^2 = \boldsymbol{P} \otimes \boldsymbol{P} = \begin{bmatrix} a \cdot \boldsymbol{P} & b \cdot \boldsymbol{P} \\ c \cdot \boldsymbol{P} & d \cdot \boldsymbol{P} \end{bmatrix} = \begin{bmatrix} aa & ab & ba & bb \\ ac & ad & bc & bd \\ ca & cb & da & db \\ cc & cd & dc & dd \end{bmatrix}.$$

Define the rth Kronecker product $\boldsymbol{P}^r : n^r \times n^r$ to be r Kronecker products of $\boldsymbol{P}$ with itself:[1] $\boldsymbol{P}^r = \boldsymbol{P} \otimes \boldsymbol{P}^{r-1}$. Then $\boldsymbol{P}^r$ is the matrix of probabilities used to sample a graph.

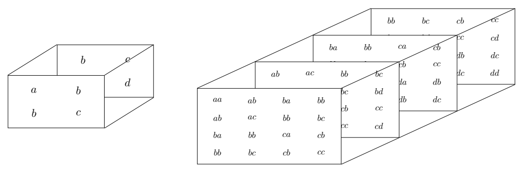
Kronecker Graph Properties. The Kronecker graph model has a number of desirable properties such as skewed degrees [35] and similar properties to real-world networks [26]. Additionally, storage of the initiator matrix $\boldsymbol{P}$ is very cheap, at just n^2 entries (where n is often taken to be 2). It has been used as a synthetic generator for parallel graph benchmarks [30] (the Graph500). Choosing parameters in the Kronecker model to *fit* a given graph has been studied using maximum-likelihood methods [26] and method-of-moments estimators [17].

Hyperedges. The HyperKron model will rely on the notion of *hyperedges*. A hyperedge is just a set of vertices, generalizing edges—sets of two vertices. All hyperedges in our model will have the same cardinality. (For simplicity, we describe our model where each hyperedge has three vertices.) When we create a graph from a hypergraph in the HyperKron model, we associate each hyperedge with a motif. For most of our paper, this motif is a triangle (§III - VI), yet HyperKron is flexible enough to handle other hyperedge structures (§VII). Though HyperKron uses the concept of hypergraphs, it is important to note that we simply use mechanisms to generate hyperedges to impose higher-order structure on a *traditional* graph.

III. The HyperKron Model

The HyperKron Model starts with a tensor and Kronecker-powers it to get a massive tensor of probabilities corresponding to *hyperedges*, a generalization of the Kronecker model which starts with a matrix and Kronecker-powers it to get a large matrix corresponding to *edges*. For the sake of simplicity in our discussion, we consider hyperedges with up to three nodes (3d tensors) although the ideas extend beyond this setting. To generate a graph, we associate the hyperedge with a triangle. The set-up extends to other motifs on three nodes (see §VII).

In more detail, we start with a 3d initiator tensor, $\underline{\boldsymbol{P}}$, with dimensions $n \times n \times n$. Just like in the Kronecker model, the value of n should be small, between 2 to 5, and entries of $\underline{\boldsymbol{P}}$ should be between 0 and 1. Symmetric in the case of a tensor means any permutation of indices has the same value. $\underline{\boldsymbol{P}}_{112} = \underline{\boldsymbol{P}}_{121} = \underline{\boldsymbol{P}}_{211}$. (Our model is not restricted to symmetric tensors, it merely simplifies the exposition.) The Kronecker product of tensors, $\underline{\boldsymbol{P}} \otimes \underline{\boldsymbol{P}}$, works just like the Kronecker product of matrices: every element gets multiplied by every other element giving way to a $n^2 \times n^2 \times n^2$ tensor [2], [32]. For example take a $2 \times 2 \times 2$ symmetric initiator tensor, and compute $\underline{\boldsymbol{P}}^2 = \underline{\boldsymbol{P}} \otimes \underline{\boldsymbol{P}}$



(1)

The Symmetric HyperKron Model with Triangles. Given an $n \times n \times n$ initiator tensor $\underline{\boldsymbol{P}}$ of probabilities, construct the rth Kronecker Product of $\underline{\boldsymbol{P}}$, $\underline{\boldsymbol{P}}^r = \underline{\boldsymbol{P}} \otimes \underline{\boldsymbol{P}}^{r-1}$. Then $\underline{\boldsymbol{P}}^r$ is of dimension $n^r \times n^r \times n^r$. Generate a set of hyperedges where we include hyperedge (i, j, k) with probability $\underline{\boldsymbol{P}}^r_{ijk}$. For each generated hyperedge, insert three undirected edges (i, j), (j, k), and (i, k). Duplicate edges are coalesced into a single edge. This results in an undirected graph on n^r vertices. The values of i, j, k need not be unique, so that we may just place an edge (or a loop). Because we insert undirected edges, it makes the most sense to consider this model with *symmetric* tensors, then we can restrict our generation to cases where $i \leq j \leq k$ (for instance) to minimize the number of duplicates.

IV. Efficient Generation

A simple algorithm to generate a HyperKron model is to explicitly generate the tensor $\underline{\boldsymbol{P}}^r$ and then to explicitly sample Bernoulli random variables (coin-flips) for each entry in the tensor. If $N = n^r$ is the dimension of the tensor, this is an $O(N^3)$ algorithm, and does not yield efficient generation of realistically large networks. The ideal case for a generation algorithm should do $O(m)$ or $O(m \log N)$ work where m is the number of edges in the output. Note that $r = \log N$. We will show how to get an $O(mr^2)$ method, which can be achieved by adapting the idea of *grass hopping* from our recent paper on generating graphs from matrices of probabilities [33].

Moreno et al. [29] first pointed out that Kronecker matrices have Erdős-Rényi sub-regions, and our approach is based on this idea. Recall that an Erdős-Rényi graph is sampled from a matrix where every edge has the same probability of occurring. An Erdős-Rényi region is a set of entries in $\underline{\boldsymbol{P}}^r$ where all the probability values are the same. For instance, note that the probability ab occurs multiple times in (1). Edges in these regions can be generated by a waiting time, geometric variable, or grass-hopping method [3], [15], [20], [33]. That is, we sample a geometric random variable to find the gap between successive edges. Thus, the method only does work proportional to the number of edges within the region. What is difficult is to identify *where* these Erdős-Rényi regions occur and then how to map from these regions back to entries in $\underline{\boldsymbol{P}}^r$.

In the remainder of this section, we show (i) that the number of Erdős-Rényi blocks is sufficiently small that this approach will work given that we have to at least *look* at each block; (ii) how to sample edges in a multiplication table

[1]We are abusing notation and use $\boldsymbol{P}^r$ to indicate "powering-up" Kronecker products rather than the standard notation of repeated matrix multiplication. We only multiply by Kronecker products in this paper.

The HyperKron sampling algorithm.

```
· For each length-$r$ multiset of $\{0, 1, \dots, n^3 - 1\}$ (call it $s$)
· Compute the probability $p$ for region $s$.
· Let $t$ be the total length of the region $s$.
· Set the index $I$ to $-1$
· While the index $I$ is less than $t$
    · Sample a geometric rand. var. with prob $p$.
    · Increment the index $I$ by the sample.
    · If the index $I$ is still less than $t$
        · Identify the multiset permutation $p$ for $I$
        · Compute the multiplication table index $J$ for $p$
        · MortonDecode($J, n, n, n$) gives a hyperedge in $\underline{\boldsymbol{P}}^r$
```

Fig. 1. The pseudocode for our fast hyperedge sampling algorithm on a HyperKron model. An implementation is at www.cs.purdue.edu/homes/dgleich/codes/hyperkron

view of the repeated Kronecker product by grass-hopping (that is, sampling geometric random variables) and unranking multiset permutations; and (iii) how to identify entries in $\underline{\boldsymbol{P}}^r$ by translating the multiplication table indices through a Morton code procedure. The final algorithm is given in Figure 1 for reference. Our procedure discussed in this section assumes a general initiator tensor $\underline{\boldsymbol{P}}$ that need not be symmetric.

A. A small number of Erdős-Rényi blocks

First, we show that there are fewer than $O(N)$ Erdős-Rényi regions in the graph. Let $\underline{\boldsymbol{P}}$ be $n \times n \times n$, and r be the number of HyperKron products, so that there are $N = n^r$ nodes in the graph. Notice that each probability value in $\underline{\boldsymbol{P}}^r$ is the product of r values from $\underline{\boldsymbol{P}}$. Entries of $\underline{\boldsymbol{P}}$ can appear more than once in the product, and the entries of $\underline{\boldsymbol{P}}^r$ are only unique up to permutation. Thus, the total number of unique probability values in $\underline{\boldsymbol{P}}^r$ is n^3 *multi-choose* r: $\binom{n^3+r-1}{r}$. This goes to $O(r^{n^3-1})$, which is less than the $O(n^r)$ nodes of the graph. (This can be verified by a simple argument similar to [33]).

B. Multiplication tables & HyperKron tensors

Each Erdős-Rényi region is identifiable by its unique product of elements from $\underline{\boldsymbol{P}}^r$, a probability value. Let us order these probability values. To that end, first number the entries of $\underline{\boldsymbol{P}}$ from 0 through $n^3 - 1$. Then when writing an entry of $\underline{\boldsymbol{P}}^r$, associate each element in the product with the sequence of r integers. For example, if $\underline{\boldsymbol{P}}$ is $2 \times 2 \times 2$ as in (1), map each of the 8 entries to an index between 0 and 7. Probability ada in $\underline{\boldsymbol{P}}^3$ would be mapped to 070, where $a = 0$ and $d = 7$.

It isn't obvious how to easily identify each of the locations of ada in $\underline{\boldsymbol{P}}^3$. (Or more generally, in $\underline{\boldsymbol{P}}^r$.) We first solve an easier problem and then later determine how to translate back to $\underline{\boldsymbol{P}}^r$. If we re-order the entries of $\underline{\boldsymbol{P}}^3$ so that $ada = 070$ occurred exactly in locations $[0, 0, 7], [0, 7, 0], [7, 0, 0]$, then the locations are easy to find—they are the permutations of [070].

We will call this re-ordering a *multiplication table*. Define $\mathbf{v} = \text{vec}(\underline{\boldsymbol{P}})$ to be the initiator tensor as a length-n^3 vector proceeding in a column-major fashion, e.g. the vectorized version of (1) is $[a, b, b, c, b, c, c, d]$. Define a r-dimensional multiplication table: $M(\underbrace{i, j, \dots, k}_{\text{r indices}}) = \underbrace{v_i v_j \cdots v_k}_{r \text{ terms}}$. For instance, $M(0, 0, 7) = M(0, 7, 0) = M(7, 0, 0) = aad$.

The start of our strategy is: for each unique probability in $\underline{\boldsymbol{P}}^r$, given by a multiset of indices as in §IV-A, "grass-hop" sample through the locations in the multiplication table where the probability is all the same. We will see how to do this efficiently next in §IV-C, and finally see how to convert between entries of the multiplication table M and $\underline{\boldsymbol{P}}^r$ in §IV-D.

C. Grass-Hopping Kronecker Tensors

Given an Erdős-Rényi region in $\underline{\boldsymbol{P}}^r$ or M, we now discuss how to "grass-hop" within that region to find successive hyperedges. Say that our Erdős-Rényi region corresponds with a probability which is mapped to indices $p = v(i_1)v(i_2) \cdots v(i_r)$ where $\mathbf{v} = \text{vec}(\underline{\boldsymbol{P}})$ as in §IV-B. Recall that each of the elements of $\mathbf{v}$ are mapped to a numerical index between 0 and $n^3 - 1$. As established in §IV-B, the locations of p in the r-dimensional multiplication table correspond exactly with permutations of $i_1, i_2, \dots, i_r$. Note that these are permutations of *multisets*, or sets in which elements can occur more than once. We label each permutation lexicographically from 0 to $t - 1$ where $t = m!/(a_1!a_2! \dots a_r!)$ is the number of permutations of the multiset $i_1, \dots, i_r$, m is the cardinality of the multiset, and a_i is the number of times that the ith element appears.

The idea for generation then, is that we can easily identify indices between 0 and $t - 1$ where edges occur because each hyperedge occurs with the same probability p. As previously hinted, this is done by sampling a geometric random variable to compute the gap until the next edge. See the discussions in [33], and [3], [15], [20] for more about this technique.

Given the indices where hyperedges occurred, we then need to map them to entries of the multiplication table. This can be done by *unranking* multiset permutations [10].

For example, suppose that the Erdős-Rényi region corresponds to a probability with indices $[0, 1, 1, 2]$. The permutations of this multiset are (lexicographically):

$$\begin{array}{llll} [0,1,1,2] \to 0 & [0,2,1,1] \to 2 & \dots & [2,2,0,1] \to 10 \\ [0,1,2,1] \to 1 & \dots & \dots & [2,2,1,0] \to 11 \end{array} \qquad (2)$$

Unranking this multiset corresponds to taking one of the indices $0, \dots, 11$ and generating the corresponding permutation. This step can be done in time $O(r^2)$ without any precomputation.

We now can use a geometric random variable to repeatedly "hop" to the next successful hyperedge, through the labels 0-(number of permutations -1), until the end is reached. This gives a list of entries in the Multiplication table.

D. Morton Codes

The relationship between the locations of the r-dimensional multiplication table entry and the 3-dimensional HyperKron tensor $\underline{\boldsymbol{P}}^r$ depends on *Morton codes* as was described for the case of matrices [33]. We extend that analysis to the 3-dimensional HyperKron tensor. A Morton code reflects a particular way of ordering multidimensional data. The particular relationship we use is established by the following theorem.

Theorem 1. *Let $\underline{\boldsymbol{P}}^r$ be an $n \times n \times n$ tensor, and $\mathbf{v}$ be the column major representation of $\underline{\boldsymbol{P}}$. Consider an element in the vectorized multiplication table M with index $(p_1, p_2, \dots, p_r)$. Let I be the lexicographical index of the element $(p_1, p_2, \dots, p_r)$. Then the 3-dimensional Morton Decoding of I in base n provides*

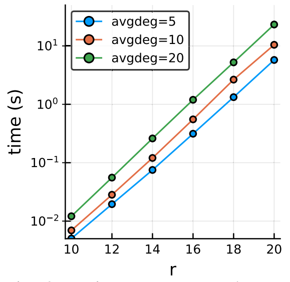


Fig. 2. Time to generate hyperedges for a HyperKron model as r varies shows linear scaling in 2^r.

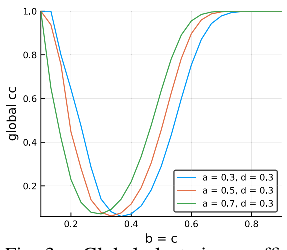


Fig. 3. Global clustering coefficients vary with changing HyperKron Parameters. Here $r = 10$.

the row, column, and slice indices of an element in $\underline{\boldsymbol{P}}^r$ *with the same value.*

The proof of this result is uninsightful, so we omit it here. See [33] for the proof for the 2d Morton decoding.

E. Runtime performance

We implemented the generation procedure (Figure 1) in the Julia language with a goal towards optimizing easy-to-avoid computational overhead. The resulting program generates hyperedges at about 1,000,000 per second on a single-thread of modern desktop computer. We evaluated the scalability of the code up to 20 million edges in three scenarios. All three scenarios use a $2 \times 2 \times 2$ symmetric initiator tensor $\underline{\boldsymbol{P}}$. This gives four parameters a, b, c, d as in equation (1). In the first case, we set $a = 0.05, b = 0.3, c = 0.4$, and choose d such that the expected number of hyperedges $(a+3b+3c+d)^r$ (*average degree* in Figure 2) is 5 times the number of nodes. In the second case, we set $a = 0.9, b = 0.3, d = 0.0$ and choose c such that the expected number of hyperedges is 10 times the number of nodes. In the third case, we set $a = 0.3, c = 0.3, d = 0.1$ and choose b so there are 20 times the number of hyperedges as nodes. The time it takes to generate graphs as r varies from 10 to 20 is shown in Figure 2. Although the theoretical scaling of our procedure is $O(mr^2)$, we observe linear scaling in this regime because the r^2 work can be done efficiently within an array of $4r$ bytes that typically fits in $L1$ cache.

V. HyperKron Properties: Non-trivial Clustering

The HyperKron model allows for generating models with significant *clustering* even with few edges, an improvement over the Kronecker model, (see §VI). We use the global clustering coefficient: $6|K_3|/|W|$, where $|K_3|$ is the number of triangles, and $|W|$ is the number of wedges, to measure how much network nodes tend to form triangles [38]. Fixing a and d parameters, and using $r = 10$, Figure 3 demonstrates that for varying all values of b and c the global clustering is always above 0.05, and is often much larger. It is large initially because all edges are in triangles. As the network becomes denser (b, c get larger), the wedges emerge causing the coefficient to drop. Finally, as the network becomes quite dense, these wedges combine into triangles. But throughout, clustering remains. This is significant because we can still achieve good clustering with sparse networks (the real-world behavior) with our model.

In the long version [13] we also compute practical estimates of the total number of generated edges.

VI. Fitting HyperKron to Real Data

We demonstrate now the HyperKron model can be fit to real-world data by hand-tuning the coefficients. Four real-world networks were chosen: *email* is a list of email exchanges between members of a University (1133 nodes) [19]; *Villanova62* (7772) and *MU68* (15k) are from the facebook 100 data set [37] where nodes represent people and edges are friendships; and *homo* is a biology network of protein interactions (8887) [36].

To fit real-world data to our HyperKron model, we choose to fit the model to just the set of triangles in the network as this is the natural structure for HyperKron to generate. (See §VI-B where we consider the full network.) See our parameter choices for a symmetric HyperKron model with a $2 \times 2 \times 2$ initiator matrix in Table I. For comparison, we fit the data sets to the Kronecker model using the method-of-moments [17] (KGMome for short) and maximum likelihood [26] (KGFit). We fit those models to the full edge data in addition to the extracted triangle data. While there are other models that would also capture clustering [24], [31], these require far more parameters and so we don't compare against them.

A. Clustering Coefficients

Global clustering coefficients are described in §V. The average local clustering coefficient is the average over the local clustering coefficient defined for each node u: $2|K_3(u)|/|W(u)|$, where $|K_3(u)|$ is the number of triangles for which u is a member, and $|W(u)|$ is the number of wedges for which u is a member. A big improvement of the HyperKron model over other graph models such as the Kronecker model, is the ability to capture clustering. Regardless of using the full data, or restricted triangle data, the Kronecker models do not capture clustering properties as closely as HyperKron (see Table I).

There remain properties of the real-world networks that HyperKron does not possess, including higher-order clustering. We use the methodology and code presented in [40] to compute *higher order clustering coefficients*. The precise details are not relevant for our case, but these extend clustering coefficients to larger cliques. We find that the HyperKron model does not display clustering in terms of four cliques, five cliques, or six cliques (3rd, 4th, and 5th order).

B. Skewed Degrees

The HyperKron model also preserves is a highly skewed degree distribution. Figure 4(a) shows the degree distributions in log-scale for two of our networks: *Villanova62*, and *MU78*, along with their HyperKron fits. We also show Loess smoothed estimates to show broader properties. There are two notable behaviors in the HyperKron degree distribution. First, there are two "tails" in nodes with small degree. The tail with larger counts are nodes with even degree. They occur in higher frequency since the model most often adds two neighbors to a node at once. Conversely, the tail with smaller counts are nodes with odd degree, since a single edge is placed infrequently.

TABLE I

Fitting real world data to the HyperKron model. The only model with non-trivial global and local clustering are the HyperKron (HKron) fits. We list the model parameters, number of edges, global clustering coefficient, mean-local clustering coefficient, and the size of the largest connected component.

Network name	edges	global clust	local clust	lcc size
***email* full**	5451	0.166	0.220	1133
KGFit: (.9538, .6196, .1463) r = 11	4941	0.032	0.060	1803
KGMome: (1.0, 0.5241, 0.2990), r = 11	5945	0.035	0.031	1351
***email* triangles**	4229	0.232	0.366	837
HKron: (0.999, 0.31, 0.2, 0.0001), r = 10	4546	0.140	0.346	735
KGFit: (.9036, .6946, .2056), r = 10	4736	0.052	0.076	949
KGMome: (1.0, 0.5132, 0.2688), r = 11	4651	0.034	0.032	1393
***homo* full**	33k	0.070	0.133	8887
KGFit: (.9895, .5569, 0.1147), r = 14	34k	0.013	0.025	6547
KGMome: (1.0, 0.5676, 0.0759), r=14	33k	0.015	0.033	6333
***homo* triangles**	19k	0.141	0.264	3783
HKron: (0.8, 0.115, 0.15, 0.83), r = 12	19k	0.101	0.164	4072
KGFit: (.9487, .6416, .1832), r = 12	20k	0.027	0.048	3194
KGMome: (1.0, 0.5227, 0.0882), r=14	20k	0.013	0.022	4502
***Villanova62* full**	315k	0.166	0.235	7755
KGFit: (.9999, .7064, .388), r = 13	326k	0.056	0.064	8187
KGMome: (1.0, 0.696, 0.4086), r = 13	326k	0.054	0.059	8185
***Villanova62* triangles**	311k	0.168	0.258	7476
HKron: (0.9, 0.4, .24, .001), r = 13	306k	0.111	0.265	7944
KGFit: (0.9999, .7058, .3865), r = 13	322k	0.055	0.064	8187
KGMome: (1.0, 0.6965, 0.4054), r = 13	323k	0.054	0.059	8186
***MU78* full**	649k	0.152	0.214	15k
KGFit: (.996, .675, .3992), r = 14	690k	0.034	0.037	16k
KGMome: (1.0, 0.6305, 0.4790), r = 14	672k	0.028	0.026	16k
***MU78* triangles**	637k	0.155	0.240	15k
HKron: (0.9, 0.42, 0.20, 0.001), r = 14	625k	0.097	0.295	16k
KGFit: (0.9993, 0.6721, 0.3973), r = 14	675k	0.037	0.034	16k
KGMome: (1.0, 0.6311, 0.4745), r = 14	661k	0.028	0.026	16k

Second, the high-degree vertices oscillate, an occurrence in the original Kronecker model as well. The peaks can be smoothed out by perturbing the probability matrix as demonstrated in [35].

We made several tweaks to the HyperKron generation to address these issues. Our explanation here is slightly abridged, more detail can be found in our long version [13]. First, we add a *noise* parameter to the HyperKron model in a generalization of the method in [35]. Using this added noise, we fit HyperKron to the set of edges involved in triangles, using the same initiator parameters as before. The second adjustment is to account for the set of remaining edges (those not involved in triangles). We fit this residual set of edges to a Kronecker model using the method of moments in [17], with an added noise parameter as in [35]. Note that when we add the Kronecker graph to the HyperKron graph, many of the edges overlap. So finally, we add in an Erdős-Rényi graph with an expected number of edges set to add enough edges to get back to the number of edges of the original graph.

Figure 4(b) gives the degree distributions of the full original network data in log-scale, along with the improved fitting. For Villanova62, the HyperKron noise was set to 0.15, and the Kronecker noise was set to 0.1. For MU78 the HyperKron noise was set to .2 and the Kronecker noise was set to 0.05. The two tailed behavior is eliminated by fitting to the non-triangle edges,

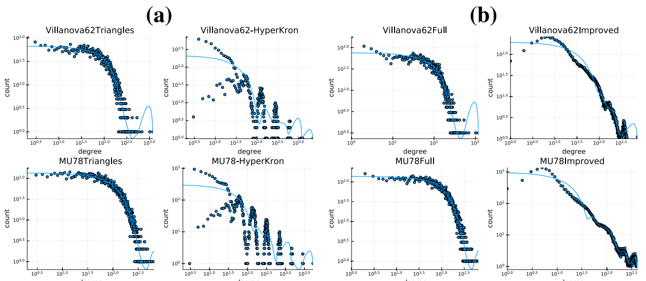


Fig. 4. (a) HyperKron preserves highly skewed degree distribution with notable behaviors discussed in the text; (b) Improvements to the HyperKron model eliminates two-tailed behavior, and almost entirely removes oscillation.

and the oscillation behavior is almost entirely removed by the noise. The fittings also retain non-trivial clustering coefficients.

VII. Model flexibility

Thus far, HyperKron was described in a setting where triangles are associated with each generated hyperedge. As we have seen, this is an appropriate choice for settings where we expect 2nd order (triangle-based) clustering in undirected networks. There are more complex types of network data, and we now show that HyperKron is also relevant here.

The *S. cerevisiae* transcription regulatory network is a directed, signed graph that describes gene expression in the common yeast organism. We extract all nodes involved in coherent feed forward loops (an important higher-order structure in this network [27]), leaving a network with 61 nodes and 108 directed, signed edges (92 positive, 16 negative). By manually tweaking entries to get the number of edges to match, we generated a HyperKron model using a $2 \times 2 \times 2$ tensor with

$$\begin{array}{llll} \underline{\boldsymbol{P}}_{111} = 0.14 & \underline{\boldsymbol{P}}_{121} = 0.25 & \underline{\boldsymbol{P}}_{211} = 0 & \underline{\boldsymbol{P}}_{221} = 0.45 \\ \underline{\boldsymbol{P}}_{112} = 0.55 & \underline{\boldsymbol{P}}_{122} = 0 & \underline{\boldsymbol{P}}_{212} = 0.31 & \underline{\boldsymbol{P}}_{222} = 0.06 \end{array}$$

and $r = 7$. We associated each hyperedge with one of the four coherent feed-forward loops based on a biased random choice, the type 1 had probability 1/2, the type 2 motif had probability 1/4, and the type 3 and 4 motifs had probability 1/8 (see [27] for more about these types). These were chosen because the real network doesn't have any type 3 and 4 feed-forward loops. When we assemble the motifs placed via these hyperedges into a network, any two motifs that share an edge with the same direction will be coalesced by summing the signs. The largest connected component of the resulting network had 69 nodes and 108 directed, signed edges (90 positive, 18 negative).

The real network has 38 coherent feed forward loops and 2 incoherent feed forward loops, while the HyperKron model has 36 and 1, respectively. The presence of incoherent feed forward loops is an emergent behavior because we only ever generated coherent loops. We might ask if finding 2 incoherent feed forward loops in the real network is likely to occur or not. By generating 10000 instances of our model, we find at least 2 incoherent feed forward loops in roughly 10%. Consequently, the presence of these two loops in the real data could easily have occurred by chance. Our code to reproduce this experiment will be posted online.

VIII. Related & future work & discussion

Let us be clear that we do not believe that HyperKron is a universal network model that is always appropriate. Rather, it provides some complementary directions in the large space of network models. One of the advantages of HyperKron is that it provides an easy and flexible means to incorporate higher-order structure. This was used as well in [8] where they associated hyperedges with triangles. We use this flexibility to model directed, signed networks in §VII and networks with non-trivial clustering coefficients in §VI-A. It is not obvious how to generate these types of structures for models based on matrices of probabilities such as Erdős-Rényi, Chung-Lu, or kernel functions [20], as well as for evolutionary models such as the copying model or forest-fire model. The HyperKron model is also easy to simulate in parallel – you can parallelize over the Erdős-Rényi regions, for instance.

That said, there are other types of network models that possess clustering. Newman [31] studied a configuration model that incorporated the triangle degree of each node. Kolda et al. [24] proposed the BTER model that has large clustering coefficients. These are both excellent models with clustering, but is unclear how to incorporate more complex types of structure such as signs into these models. Likewise, models that randomly generate points for each node and then connect nearby nodes based on a metric space are often known to have non-trivial local clustering [9], [22]. However, these models tend to be unrealistically dense.

HyperKron is easy to combine with the majority of other ideas that have been proposed to extend Kronecker models. For instance, adapting the mKPGM model [28] to our setting simply involves a deterministic choice for some of the early tensors. Likewise, the MAG model uses a set of Kronecker models to handle attributed graphs [23]. It remains open how to pragmatically fit HyperKron to real data similar to [17] or [26]. This remains one of our most important next steps.

Funding. Research supported in part by NSF CAREER CCF-1149756, IIS-1422918, IIS-1546488, NSF Center for Science of Information STC, CCF-0939370, DOE DE-SC0014543, DARPA SIMPLEX, and the Sloan Foundation.

References

[1] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *STOC*, pages 171–180. ACM, 2000.

[2] L. Akoglu, M. McGlohon, and C. Faloutsos. RTM: Laws and a recursive generator for weighted time-evolving graphs. In *ICDM'08. Eighth IEEE International Conference on Data Mining*, pages 701–706. IEEE, 2008.

[3] V. Batagelj and U. Brandes. Efficient generation of large random networks. *Physical Review E*, 71(3), March 2005.

[4] E. A. Bender and E. Canfield. The asymptotic number of labeled graphs with given degree sequences. *J. Comb. Theory, A*, 24(3):296 – 307, 1978.

[5] A. Benson, D. F. Gleich, and J. Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

[6] A. Benson, D. F. Gleich, and L.-H. Lim. The spacey random walk: a stochastic process for higher-order data. *SIAM Review*, 59(2):321–345, May 2017.

[7] S. Bhamidi, G. Bresler, and A. Sly. Mixing time of exponential random graphs. *Ann. Appl. Probab.*, 21(6):2146–2170, 12 2011.

[8] B. Bollobás, S. Janson, and O. Riordan. Sparse random graphs with clustering. *Random Structures and Algorithms*, 38(3):269–323, 2011.

[9] A. Bonato, J. Janssen, and P. Prałat. Geometric protean graphs. *Internet Mathematics*, 8(1-2):2–28, 2012.

[10] B. Bonet. Efficient algorithms to rank and unrank permutations in lexicographic order. *AAAI-workshop on Search in AI and Robotics*, 2008.

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

[12] F. Chung and L. Lu. Connected components in random graphs with given expected degree sequences. *Ann. Comb.*, 6(2):125–145, 2002.

[13] N. Eikmeier, A. S. Ramani, and D. F. Gleich. The hyperkron graph model for higher-order features. *arXiv*, cs.SI:1809.03488, 2018.

[14] P. Erdős and A. Rényi. On random graphs I. *Publicationes Mathematicae*, 6:290–297, 1959.

[15] C. T. Fan, M. E. Muller, and I. Rezucha. Development of sampling plans by using sequential (item by item) selection techniques and digital computers. *J. Am. Stat. Assoc.*, 57(298):387–402, 1962.

[16] B. K. Fosdick, D. B. Larremore, J. Nishimura, and J. Ugander. Configuring random graph models with fixed degree sequences. *SIAM Review*, In press.

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

[18] J. Grilli et al. Higher-order interactions stabilize dynamics in competitive network models. *Nature*, 548:210–213, 2017.

[19] R. Guimerà et al. Self-similar community structure in a network of human interactions. *Phys. Rev. E*, 68:065103, 2003.

[20] A. Hagberg and N. Lemons. Fast generation of sparse random kernel graphs. *PLOS ONE*, 10(9):e0135177, sep 2015.

[21] P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[22] E. Jacob and P. Mörters. Spatial preferential attachment networks: Power laws and clustering coefficients. *The Annals of Applied Probability*, 25(2):632–662, 2015.

[23] M. Kim and J. Leskovec. Multiplicative attribute graph model of real-world networks. In *WAW*, pages 62–73. Springer, 2010.

[24] T. G. Kolda, A. Pinar, T. Plantenga, and C. Seshadhri. A scalable generative graph model with community structure. *SIAM J. Sci. Comp.*, 36(5):C424–C452, 2014.

[25] J. Leskovec, D. Chakrabarti, J. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using Kronecker multiplication. In *PKDD*, volume 3721 of *Lecture Notes in Computer Science*, pages 133–145. 2005.

[26] J. Leskovec, D. Chakrabarti, J. Kleinberg, et al. Kronecker graphs: An approach to modeling networks. *JMLR*, 11:985–1042, 2010.

[27] R. Milo et al. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[28] S. Moreno, S. Kirshner, J. Neville, and S. V. N. Vishwanathan. Tied Kronecker product graph models to capture variance in network populations. In *Allerton*, pages 1137–1144, Sept 2010.

[29] S. Moreno, J. J. Pfieffer, J. Neville, and S. Kirshner. A scalable method for exact sampling from Kronecker family models. *ICDM*, 2014.

[30] R. C. Murphy, K. B. Wheeler, B. W. Barrett, and J. A. Ang. Introducing the graph 500. Cray User's Group, May 2010.

[31] M. E. Newman. Random graphs with clustering. *Physical review letters*, 103(5):058701, 2009.

[32] A. H. Phan, A. Cichocki, P. Tichavský, et al. On revealing replicating structures in multiway data: A novel tensor decomposition approach. In *Latent Variable Analysis and Signal Separation*, pages 297–305, 2012.

[33] A. S. Ramani, N. Eikmeier, and D. F. Gleich. Coin-flipping, ball-dropping, and grass-hopping for generating random graphs from matrices of probabilities. *arXiv*, cs.SI:1709.03438, 2017.

[34] M. Rosvall et al. Memory in network flows and its effects on spreading dynamics and community detection. *Nat. Comm.*, 5(4630), 2014.

[35] C. Seshadhri, A. Pinar, and T. G. Kolda. An in-depth analysis of stochastic kronecker graphs. *J. ACM*, 60(2):13:1–13:32, May 2013.

[36] R. Singh, J. Xu, and B. Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *PNAS*, 105(35):12763–12768, 2008.

[37] A. L. Traud, P. J. Mucha, and M. A. Porter. Social structure of facebook networks. *Physica A*, 391(16):4165–4180, 2012.

[38] D. J. Watts and S. H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393(6684):440, 1998.

[39] J. Xu, T. L. Wickramarathne, and N. V. Chawla. Representing higher-order dependencies in networks. *Science Advances*, 2(5):e1600028–e1600028, may 2016.

[40] H. Yin, A. R. Benson, and J. Leskovec. Higher-order clustering in networks. *arXiv preprint arXiv:1704.03913*, 2017.