Kronecker graphs: an approach to modeling networks

Jure Leskovec
Computer Science Department, Cornell University
Computer Science Department, Stanford University

Deepayan Chakrabarti
Yahoo! Research

Jon Kleinberg
Computer Science Department, Cornell University

Christos Faloutsos
Computer Science Department, Carnegie Mellon University

Zoubin Gharamani
Department of Engineering, University of Cambridge
Machine Learning Department, Carnegie Mellon University

Editor: Editor

Abstract

How can we generate realistic networks? In addition, how can we do so with a mathematically tractable model that allows for rigorous analysis of network properties? Real networks exhibit a long list of surprising properties: Heavy tails for the in- and out-degree distribution; heavy tails for the eigenvalues and eigenvectors; small diameters; and densification and shrinking diameters over time. The present network models and generators either fail to match several of the above properties, are complicated to analyze mathematically, or both. In this chapter we propose a generative model for networks that is both mathematically tractable and can generate networks that have all the above mentioned structural properties. Our main idea here is to use a non-standard matrix operation, the Kronecker product, to generate graphs that we refer to as “Kronecker graphs”.

First, we show that Kronecker graphs naturally obey common network properties; in fact, we rigorously prove that they do so. We also provide empirical evidence showing that Kronecker graphs can effectively model the structure of real networks.

We then present KRONFIT, a fast and scalable algorithm for fitting the Kronecker graph generation model to large real networks. A naive approach to fitting would take super-exponential time. In contrast, KRONFIT takes linear time, by exploiting the structure of Kronecker matrix multiplication and by using statistical simulation techniques.

Experiments on large real and synthetic networks show that KRONFit finds accurate parameters that indeed very well mimic the properties of target networks. Once fitted, the model parameters can be used to gain insights about the network structure, and the resulting synthetic graphs can be used for null-models, anonymization, extrapolations, and graph summarization.

Keywords: Kronecker graphs, Network analysis, Network models, Social networks, Graph generators, Graph mining, Network evolution

©2008 Jure Leskovec et al.
1. Introduction

What do real graphs look like? How do they evolve over time? How can we generate synthetic, but realistic looking, time-evolving graphs? Recently network analysis has been attracting much interest, with an emphasis on finding patterns and abnormalities in social networks, computer networks, e-mail interactions, gene regulatory networks, and many more. Most of the work focuses on static snapshots of graphs, where fascinating “laws” have been discovered, including small diameters and heavy-tailed degree distributions.

As such structural “laws” have been discovered a natural next question is to find a model that produces networks with such structure. Thus, a good realistic network generation model is important for at least two reasons. The first is that it can generate graphs for extrapolations, “what-if” scenarios, and simulations, when real graphs are difficult or impossible to collect. For example, how well will a given protocol run on the Internet five years from now? Accurate network models can produce more realistic models for the future Internet, on which simulations can be run. The second reason is more subtle: it forces us to think about the network properties that a graph models should obey, to be realistic.

In this paper we introduce Kronecker graphs, a network generative model which obeys all the main static network patterns that have appeared in the literature. Our model also obeys recently discovered the temporal evolution patterns (Leskovec et al., 2005b, 2007a). And, contrary to other models that match this combination of network properties, Kronecker graphs also lead to tractable analysis and rigorous proofs. Furthermore, the Kronecker graphs generative process also has a nice natural interpretation and justification.

Our model is based on a matrix operation, the Kronecker product. There are several known theorems on Kronecker products, which correspond exactly to a significant portion of what we want to prove: heavy-tailed distributions for in-degree, out-degree, eigenvalues, and eigenvectors. We also demonstrate how a Kronecker Graph can match the behavior of several real networks (social networks, citations, web, internet, and others). While Kronecker products have been studied by the algebraic combinatorics community (see, e.g., (Chow, 1997)), the present work is the first to employ this operation in the design of network models to match real data.

Then we also make a step further and tackle the following problem: Given a large real network, we want to generate a synthetic graph, so that our resulting synthetic graph matches the properties of the real network as well as possible.

Ideally we would like: (a) A graph generation model that naturally produces networks with many properties that are also found in real networks. (b) The model parameter estimation should be fast and scalable, so that we can handle networks with millions of nodes. (c) The resulting set of parameters should generate realistic-looking networks that match the statistical properties of the target, real networks.

In general the problem of modeling network structure presents several conceptual and engineering challenges: Which generative model should we choose, among the many in the literature? How do we measure the goodness of the fit? (Least squares don’t work well for power laws, for subtle reasons!) If we use likelihood, (that we do), how to estimate it faster than in time quadratic on the number of nodes? How do we solve the node correspondence problem (which node of the real network corresponds to what node of the synthetic one)?

To answer the above questions we present KRONFit, a fast and scalable algorithm for fitting Kronecker graphs by using the maximum likelihood principle. When calculating the likelihood
there are two challenges: First, one needs to solve the node correspondence problem by matching the nodes of the real and the synthetic network. Essentially, one has to consider all mappings of nodes of the network to the rows and columns of the graph adjacency matrix. This becomes intractable for graphs with more than tens of nodes. Even when given the “true” correspondences, just evaluating the likelihood is still prohibitively expensive for the size of graphs we want to consider here. We present solutions to both of these problems: We develop a Metropolis sampling algorithm for sampling node correspondences, and approximate the likelihood to obtain a linear time algorithm that scales to large networks with millions of nodes and edges. **KRONFit** gives orders of magnitude speed-ups against older methods (20 minutes on a commodity PC, versus 2 days on a 50-machine cluster).

Our extensive experiments on synthetic and real networks show that Kronecker Graph can efficiently model statistical properties of networks, like degree distribution and diameter, while using only four parameters.

Once the model is fitted to the real network, there are several benefits and applications:

(a) The parameters give us insight into the structure of the network itself;

(b) **Null-model**: when working with network data we would often like to assess the significance or the extent to which a certain network property is expressed. We can use the fitted Kronecker graph as an accurate null-model.

(c) **Simulations**: given an algorithm working on a graph we would like to evaluate how its performance depends on various properties of the network. Using our model one can generate graphs that exhibit various combinations of such properties, and then evaluate the algorithm.

(d) **Extrapolations**: we can use the model to generate a larger graph, to help us understand how the network will look like in the future.

(e) **Sampling**: conversely, we can also generate a smaller graph, which may be useful for running simulation experiments (e.g., simulating routing algorithms in computer networks, or virus/worm propagation algorithms), when these algorithms may be too slow to run on large graphs.

(f) **Graph similarity**: to compare the similarity of the structure of different networks (even of different sizes) one can use the differences in estimated parameters as a similarity measure.

(g) **Graph visualization and compression**: we can compress the graph, by storing just the model parameters, and the deviations between the real and the synthetic graph. Similarly, for visualization purposes one can use the structure of the parameter matrix to visualize the backbone of the network, and then display the edges that deviate from the backbone structure.

(h) **Anonymization**: suppose that the real graph cannot be publicized, like, e.g., corporate e-mail network. customer-product sales in a recommendation system. Yet, we would like to share our network. Our work gives ways to such a realistic, ’similar’ network.

The current paper builds on our previous work on Kronecker graphs (Leskovec et al., 2005a; Leskovec and Faloutsos, 2007) and is organized as follows: Section 2 briefly surveys the related literature. In section 3 we introduce the Kronecker graphs model, and give formal statements about
the properties of networks it generates. We investigate the model using simulation in Section 4 and continue by introducing KRONFIT, the Kronecker graphs parameter estimation algorithm, in Section 5. We present experimental results on real and synthetic networks in Section 6. We close with discussion and conclusions in sections 7 and 8.

2. Relation to previous work on network modeling

Networks across a wide range of domains present surprising regularities, like power laws, small diameters, communities, and so on. We use these patterns as sanity checks, that is, our synthetic graphs should match those properties of the real target graph.

Most of the related work in this field has concentrated on two aspects: properties and patterns found in real-world networks, and then ways to find models to build understanding about the emergence of these properties. First, we will discuss the commonly found patterns in (static and temporally evolving) graphs, and finally, the state of the art in graph generation methods.

2.1 Graph Patterns

Here we briefly introduce the network patterns (also referred to as properties or statistics) that we will later use to compare the similarity between the real networks and their synthetic counterparts produced by Kronecker graphs model. While many patterns have been discovered, two of the principal ones are heavy-tailed degree distributions and small diameters.

Degree distribution: The degree-distribution of a graph is a power law if the number of nodes $N_d$ with degree $d$ is given by $N_d \propto d^{-\gamma}$ ($\gamma > 0$) where $\gamma$ is called the power law exponent. Power laws have been found in the Internet (Faloutsos et al., 1999), the Web (Kleinberg et al., 1999; Broder et al., 2000), citation graphs (Redner, 1998), online social networks (Chakrabarti et al., 2004) and many others.

Small diameter: Most real-world graphs exhibit relatively small diameter (the “small-world” phenomenon, or “six degrees of separation”): A graph has diameter $D$ if every pair of nodes can be connected by a path of length at most $D$ edges. The diameter $D$ is susceptible to outliers. Thus, a more robust measure of the pair wise distances between nodes in a graph is the effective diameter (Tauro et al., 2001), which is the minimum number of links (steps/hops) in which some fraction (or quantile $q$, say $q = 0.9$) of all connected pairs of nodes can reach each other. The effective diameter has been found to be small for large real-world graphs, like Internet, Web, and online social networks (Albert and Barabási, 2002; Milgram, 1967; Leskovec et al., 2005b).

Hop-plot: extends the notion of diameter by plotting the number of reachable pairs $g(h)$ within $h$ hops, as a function of the number of hops $h$ (Palmer et al., 2002). It gives us a sense of how quickly nodes’ neighborhoods expand with the number of hops.

Scree plot: This is a plot of the eigenvalues (or singular values) of the graph adjacency matrix, versus their rank, using the logarithmic scale. The scree plot is also often found to approximately obey a power law (Chakrabarti et al., 2004; Farkas et al., 2001). Moreover, this pattern was also found analytically for random power law graphs (Chung et al., 2003).

Network values: The distribution of eigenvector components (indicators of “network value”) associated to the largest eigenvalue of the graph adjacency matrix has also been found to be skewed (Chakrabarti et al., 2004).

Node triangle participation: is a measure of transitivity in networks. It counts the number of triangles a node participates in, i.e., the number of connections between the neighbors of a node.
The plot of the number of triangles $\Delta$ versus the number of nodes participating in $\Delta$ triangles has also been found to be skewed (Tsourakakis, 2008).

**Densification Power Law:** The relation between the number of edges $E(t)$ and the number of nodes $N(t)$ in evolving network at time $t$ obeys the densification power law (DPL), which states that $E(t) \propto N(t)^a$. The densification exponent $a$ is typically greater than 1, implying that the average degree of a node in the network is increasing over time (as the network gains more nodes and edges). This means that real networks tend to sprout many more edges than nodes, and thus densify as they grow (Leskovec et al., 2005b, 2007a).

**Shrinking diameter:** The effective diameter of graphs tends to shrink or stabilize as the number of nodes in a network grows over time (Leskovec et al., 2005b, 2007a). This is somewhat counterintuitive since from common experience as one would expect that as the volume of the object (a graph) grows, the size (i.e., the diameter) would also grow. But for networks it seems this does not hold as the diameter shrinks and then stabilizes as the network grows.

### 2.2 Generative models of network structure

The earliest probabilistic generative model for graphs was the Erdős-Rényi (Erdős and Rényi, 1960) random graph model, where each pair of nodes has an identical, independent probability of being joined by an edge. The study of this model has led to a rich mathematical theory; however, as the model was not developed to model real-world networks it produces graphs that fail to match real networks in a number of respects (for example, it does not produce heavy-tailed degree distributions).

The vast majority of recent network models involve some form of preferential attachment (Barabási and Albert, 1999; Albert and Barabási, 2002; Winick and Jamin, 2002; Kleinberg et al., 1999; Kumar et al., 1999) that employs a simple rule: new node joins the graph at each time step, and then creates a connection to an existing node $u$ with the probability proportional to the degree of the node $u$. This leads to the “rich get richer” phenomena and to power law tails in degree distribution. However, the diameter in this model grows slowly with the number of nodes $N$, which violates the “shrinking diameter” property mentioned above.

There are also many variations of preferential attachment model, all somehow employing the “rich get richer” type mechanism, e.g., the “copying model” (Kumar et al., 2000), the “winner does not take all” model (Pennock et al., 2002), the “forest fire” model (Leskovec et al., 2005b), the “random surfer model” (Blum et al., 2006), etc.

A different family of network methods strives for small diameter and local clustering in networks. Examples of such models include the small-world model (Watts and Strogatz, 1998) and the Waxman generator (Waxman, 1988). Another family of models shows that heavy tails emerge if nodes try to optimize their connectivity under resource constraints (Carlson and Doyle, 1999; Fabrikant et al., 2002).

In summary, most current models focus on modeling only one (static) network property, and neglect the others. In addition, it is usually hard to analytically analyze properties of the network model. On the other hand, the Kronecker graphs model we describe in the next section addresses these issues as it matches multiple properties of real networks at the same time, while being analytically tractable and lending itself to rigorous analysis.
2.3 Parameter estimation of network models

Until recently relatively little effort was made to fit the above network models to real data. One of the difficulties is that most of the above models usually do not have a probabilistic interpretation, but rather define a mechanism or a principle by which a network is constructed.

Most work in estimating network models comes from the area of social sciences, statistics and social network analysis where the exponential random graphs, also known as $p^*$ model, were introduced (Wasserman and Pattison, 1996). The model essentially defines a log linear model over all possible graphs $G$, $p(G|\theta) \propto \exp(\theta^T s(G))$, where $G$ is a graph, and $s$ is a set of functions, that can be viewed as summary statistics for the structural features of the network. The $p^*$ model usually focuses on “local” structural features of networks (like, e.g., characteristics of nodes that determine a presence of an edge, link reciprocity, etc.). As exponential random graphs have been very useful for modeling small networks, and individual nodes and edges, our goal here is different in a sense that we aim to accurately model the structure of the network as a whole. Moreover, we aim to model and estimate parameters of networks with millions of nodes, while even for graphs of small size (> 100 nodes) the number of model parameters in exponential random graphs usually becomes too large, and estimation prohibitively expensive, both in terms of computational time and memory.

Regardless of a particular choice of a network model, a common theme when estimating the likelihood $P(G)$ of a graph $G$ under some model is the challenge of finding the correspondence between the nodes of the true network and its synthetic counterpart. The node correspondence problem results in the factorially many possible matchings of nodes. One can think of the correspondence problem as a test of graph isomorphism. Two isomorphic graphs $G$ and $G'$ with differently assigned node ids should have same likelihood $P(G) = P(G')$ so we aim to find an accurate mapping between the nodes of the two graphs.

An ordering or a permutation defines the mapping of nodes in one network to nodes in the other network. For example, Butts (Butts, 2005) used permutation sampling to determine similarity between two graph adjacency matrices, while Bezákova et al. (Bezákova et al., 2006) used permutations for graph model selection. Recently, an approach for estimating parameters of the “copying” model was introduced (Wiuf et al., 2006), however authors also note that the class of “copying” models may not be rich enough to accurately model real networks. As we show later, Kronecker graphs model seems to have the necessary expressive power to mimic real networks well.

3. Kronecker graphs model

The Kronecker graphs model we propose here is based on a recursive construction. Defining the recursion properly is somewhat subtle, as a number of standard, related graph construction methods fail to produce graphs that densify according to the patterns observed in real networks, and they also produce graphs whose diameters increase. To produce densifying graphs with constant/shrinking diameter, and thereby match the qualitative behavior of a real network, we develop a procedure that is best described in terms of the Kronecker product of matrices.

3.1 Main idea

The main intuition behind the model is to create self-similar graphs, recursively. We begin with an initiator graph $K_1$, with $N_1$ nodes and $E_1$ edges, and by recursion we produce successively larger
<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>Real network</td>
</tr>
<tr>
<td>N</td>
<td>Number of nodes in G</td>
</tr>
<tr>
<td>E</td>
<td>Number of edges in G</td>
</tr>
<tr>
<td>K</td>
<td>Kronecker graph (synthetic estimate of G)</td>
</tr>
<tr>
<td>$K_1$</td>
<td>Initiator of a Kronecker Graph</td>
</tr>
<tr>
<td>$N_1$</td>
<td>Number of nodes in initiator $K_1$</td>
</tr>
<tr>
<td>$E_1$</td>
<td>Number of edges in initiator $K_1$</td>
</tr>
<tr>
<td>$G \otimes H$</td>
<td>Kronecker product of adjacency matrices of graphs G and H</td>
</tr>
<tr>
<td>$K_1^{[k]} = K_k = K$</td>
<td>$k$th Kronecker power of $K_1$</td>
</tr>
<tr>
<td>$K_1[i,j]$</td>
<td>Entry at row $i$ and column $j$ of $K_1$</td>
</tr>
<tr>
<td>$\Theta = P_1$</td>
<td>Stochastic Kronecker initiator</td>
</tr>
<tr>
<td>$P_1^{[k]} = P_k = P$</td>
<td>$k$th Kronecker power of $P_1$</td>
</tr>
<tr>
<td>$p_{ij} = P_1[i,j]$</td>
<td>Entry at row $i$ and column $j$ of $P_1$</td>
</tr>
<tr>
<td>$\theta_{ij} = P_k[i,j]$</td>
<td>Probability of an edge $(i,j)$ in $P_k$, i.e., entry at row $i$ and column $j$ of $P_k$</td>
</tr>
<tr>
<td>$K = R(P)$</td>
<td>Realization of a Stochastic Kronecker graph $P$</td>
</tr>
<tr>
<td>$l(\Theta)$</td>
<td>Log-likelihood. Log-prob. that $\Theta$ generated real graph $G$, $\log P(G</td>
</tr>
<tr>
<td>$\hat{\Theta}$</td>
<td>Parameters at maximum likelihood, $\hat{\Theta} = \arg\max_{\Theta} P(G</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Permutation that maps node ids of $G$ to those of $P$</td>
</tr>
<tr>
<td>$a$</td>
<td>Densification power law exponent, $E(t) \propto N(t)^a$</td>
</tr>
<tr>
<td>$D$</td>
<td>Diameter of a graph</td>
</tr>
<tr>
<td>$N_c$</td>
<td>Number of nodes in the largest weakly connected component of a graph</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Proportion of times SwapNodes permutation proposal distribution is used</td>
</tr>
</tbody>
</table>

Table 1: Table of symbols.

graphs $K_2, K_3, \ldots$ such that the $k$th graph $K_k$ is on $N_k = N_1^k$ nodes. If we want these graphs to exhibit a version of the Densification Power Law (Leskovec et al., 2005b), then $K_k$ should have $E_k = E_1^k$ edges. This is a property that requires some care in order to get right, as standard recursive constructions (for example, the traditional Cartesian product or the construction of (Barabási et al., 2001)) do not satisfy it.

It turns out that the **Kronecker product** of two matrices is the right tool for this goal. The Kronecker product is defined as follows:
Definition 2 (Kronecker product of graphs) Given two graphs $G$ and $H$ with adjacency matrices $A(G)$ and $A(H)$ respectively, the Kronecker product $G \otimes H$ is defined as the graph with adjacency matrix $A(G) \otimes A(H)$.

Example of Kronecker multiplication: Top: a “3-chain” initiator graph and its Kronecker product with itself; each of the $X_i$ nodes gets expanded into 3 nodes, which are then linked using Observation 1. Bottom row: the corresponding adjacency matrices. See figure 2 for adjacency matrices of $K_3$ and $K_4$.

Figure 1: Example of Kronecker multiplication: Top: a “3-chain” initiator graph and its Kronecker product with itself; each of the $X_i$ nodes gets expanded into 3 nodes, which are then linked using Observation 1. Bottom row: the corresponding adjacency matrices. See figure 2 for adjacency matrices of $K_3$ and $K_4$.

Definition 1 (Kronecker product of matrices) Given two matrices $A = [a_{i,j}]$ and $B$ of sizes $n \times m$ and $n' \times m'$ respectively, the Kronecker product matrix $C$ of dimensions $(n \cdot n') \times (m \cdot m')$ is given by

$$C = A \otimes B = \begin{pmatrix} a_{1,1}B & a_{1,2}B & \ldots & a_{1,m}B \\ a_{2,1}B & a_{2,2}B & \ldots & a_{2,m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}B & a_{n,2}B & \ldots & a_{n,m}B \end{pmatrix}$$

We then define the Kronecker product of two graphs simply as the Kronecker product of their corresponding adjacency matrices.

Definition 2 (Kronecker product of graphs) If $G$ and $H$ are graphs with adjacency matrices $A(G)$ and $A(H)$ respectively, then the Kronecker product $G \otimes H$ is defined as the graph with adjacency matrix $A(G) \otimes A(H)$.

Observation 1 (Edges in Kronecker-multiplied graphs)

Edge $(X_{ij}, X_{kl}) \in G \otimes H$ iff $(X_i, X_k) \in G$ and $(X_j, X_l) \in H$

where $X_{ij}$ and $X_{kl}$ are nodes in $G \otimes H$, and $X_i$, $X_j$, $X_k$ and $X_l$ are the corresponding nodes in $G$ and $H$, as in Figure 1.
Figure 2: Adjacency matrices of $K_3$ and $K_4$, the 3rd and 4th Kronecker power of $K_1$ matrix as defined in Figure 1. Dots represent non-zero matrix entries, and white space represents zeros. Notice the recursive self-similar structure of the adjacency matrix.

The last observation is subtle, but crucial, and deserves elaboration. Basically, each node in $G \otimes H$ can be represented as an ordered pair $X_{ij}$, with $i$ a node of $G$ and $j$ a node of $H$, and with an edge joining $X_{ij}$ and $X_{kl}$ precisely when $(X_i, X_k)$ is an edge of $G$ and $(X_j, X_l)$ is an edge of $H$. This is a direct consequence of the hierarchical nature of the Kronecker product. Figure 1(a–c) further illustrates this by showing the recursive construction of $G \otimes H$, when $G = H$ is a 3-node chain. Consider node $X_{1,2}$ in Figure 1(c): It belongs to the $H$ graph that replaced node $X_1$ (see Figure 1(b)), and in fact is the $X_2$ node (i.e., the center) within this small $H$-graph.

We propose to produce a growing sequence of matrices by iterating the Kronecker product:

**Definition 3 (Kronecker power)** The $k^{th}$ power of $K_1$ is defined as the matrix $K_1^{[k]}$ (abbreviated to $K_k$), such that:

$$K_1^{[k]} = K_k = \underbrace{K_1 \otimes K_1 \otimes \ldots \otimes K_1}_{k \text{ times}} = K_{k-1} \otimes K_1$$

**Definition 4 (Kronecker graph)** Kronecker graph of order $k$ is defined by the adjacency matrix $K_1^{[k]}$, where $K_1$ is the Kronecker initiator adjacency matrix.

The self-similar nature of the Kronecker graph product is clear: To produce $K_k$ from $K_{k-1}$, we “expand” (replace) each node of $K_{k-1}$ by converting it into a copy of $K_1$, and we join these copies together according to the adjacencies in $K_{k-1}$ (see Figure 1). This process is very natural: one can imagine it as positing that communities within the graph grow recursively, with nodes in the community recursively getting expanded into miniature copies of the community. Nodes in the sub-community then link among themselves and also to nodes from other communities.
3.2 Analysis of Kronecker Graphs

We shall now discuss the properties of Kronecker graphs, specifically, their degree distributions, diameters, eigenvalues, eigenvectors, and time-evolution. Our ability to prove analytical results about all of these properties is a major advantage of Kronecker graphs over other network models.

3.2.1 Degree Distribution

The next few theorems prove that several distributions of interest are multinomial for our Kronecker graph model. This is important, because a careful choice of the initial graph $K_1$ makes the resulting multinomial distribution to behave like a power law or DGX distribution (Bi et al., 2001; Clauset et al., 2007).

**Theorem 5 (Multinomial degree distribution)**  Kronecker graphs have multinomial degree distributions, for both in- and out-degrees.

**Proof**  Let the initiator $K_1$ have the degree sequence $d_1, d_2, \ldots, d_{N_1}$. Kronecker multiplication of a node with degree $d$ expands it into $N_1$ nodes, with the corresponding degrees being $d \times d_1, d \times d_2, \ldots, d \times d_{N_1}$. After Kronecker powering, the degree of each node in graph $K_k$ is of the form $d_{i_1} \times d_{i_2} \times \ldots \times d_{i_k}$, with $i_1, i_2, \ldots, i_k \in (1 \ldots N_1)$, and there is one node for each ordered combination. This gives us the multinomial distribution on the degrees of $K_k$. So, graph $K_k$ will have
multinomial degree distribution where the “events” (degrees) of the distribution will be combinations of degree products: \(d_1^{i_1}d_2^{i_2} \ldots d_{N_1}^{i_{N_1}}\) (where \(\sum_{j=1}^{N_1} i_j = k\)) and event (degree) probabilities will be proportional to \(\binom{k}{i_1 \ldots i_{N_1}}\). Note also that this is equivalent to noticing that the degrees of nodes in \(K_k\) can be expressed as the \(k^{th}\) Kronecker power of the vector \((d_1, d_2, \ldots, d_{N_1})\).

### 3.2.2 Spectral properties

Next we analyze the spectral properties of adjacency matrix of a Kronecker graph. We show that both the distribution of eigenvalues and the distribution of component values of eigenvectors of the graph adjacency matrix follow multinomial distributions.

**Theorem 6 (Multinomial eigenvalue distribution)** The Kronecker graph \(K_k\) has a multinomial distribution for its eigenvalues.

**Proof** Let \(K_1\) have the eigenvalues \(\lambda_1, \lambda_2, \ldots, \lambda_{N_1}\). By properties of the Kronecker multiplication (Loan, 2000; Langville and Stewart, 2004), the eigenvalues of \(K_k\) are the \(k^{th}\) Kronecker power of the vector of eigenvalues of the initiator matrix, \((\lambda_1, \lambda_2, \ldots, \lambda_{N_1})^k\). As in Theorem 5, the eigenvalue distribution is a multinomial.

A similar argument using properties of Kronecker matrix multiplication shows the following.

**Theorem 7 (Multinomial eigenvector distribution)** The components of each eigenvector of the Kronecker graph \(K_k\) follow a multinomial distribution.

**Proof** Let \(K_1\) have the eigenvectors \(\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_{N_1}\). By properties of the Kronecker multiplication (Loan, 2000; Langville and Stewart, 2004), the eigenvectors of \(K_k\) are given by the \(k^{th}\) Kronecker power of the vector: \((\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_{N_1})\), which gives a multinomial distribution for the components of each eigenvector in \(K_k\).

We have just covered several of the static graph patterns. Notice that the proofs were a direct consequences of the Kronecker multiplication properties.

### 3.2.3 Connectivity of Kronecker graphs

We now present a series of results on the connectivity of Kronecker graphs. We show, maybe a bit surprisingly, that even if a Kronecker initiator graph is connected its Kronecker power can in fact be disconnected.

**Lemma 8** If at least one of \(G\) and \(H\) is a disconnected graph, then \(G \otimes H\) is also disconnected.

**Proof** Without loss of generality we can assume that \(G\) has two connected components, while \(H\) is connected. Figure 4(a) illustrates the corresponding adjacency matrix for \(G\). Using the notation from observation 1 let graph let \(G\) have nodes \(X_1, \ldots, X_n\), where nodes \(\{X_1, \ldots, X_r\}\) and
Figure 4: Graph adjacency matrices. Dark parts present connected (filled with ones) and white parts present empty (filled with zeros) parts of the adjacency matrix. (a) When $G$ is disconnected, Kronecker multiplication with any matrix $H$ will result in $G \otimes H$ being disconnected. (b) Adjacency matrix of a connected bipartite graph $G$ with partitions $A$ and $B$. (c) Adjacency matrix of a connected bipartite graph $G$ with partitions $C$ and $D$. (e) Kronecker product of two bipartite graphs $G$ and $H$. (d) After rearranging the adjacency matrix $G \otimes H$ we clearly see the resulting graph is disconnected.

\[ \{X_{r+1}, \ldots, X_n\} \] form the two connected components. Now, note that $(X_{ij}, X_{kl}) \notin G \otimes H$ for $i \in \{1, \ldots, r\}$, $k \in \{r + 1, \ldots, n\}$, and all $j, l$. This follows directly from observation 1 as $(X_i, X_k)$ are not edges in $G$. Thus, $G \otimes H$ must at least two connected components.

Actually it turns out that both $G$ and $H$ can be connected but $G \otimes H$ is still disconnected. The following theorem analyzes this case.

**Theorem 9** If both $G$ and $H$ are connected but bipartite, then $G \otimes H$ is disconnected, and each of the two connected components is again bipartite.

**Proof** Again without loss of generality let $G$ be bipartite with two partitions $A = \{X_1, \ldots, X_r\}$ and $B = \{X_{r+1}, \ldots, X_n\}$, where edges exist only between the partitions, and no edges exist inside the partition: $(X_i, X_k) \notin G$ for $i, k \in A$ or $i, k \in B$. Similarly, let $H$ also be bipartite with
two partitions $C = \{X_1, \ldots X_s\}$ and $D = \{X_{s+1}, \ldots X_m\}$. Figures 4(b) and (c) illustrate the structure of the corresponding adjacency matrices.

Now, there will be two connected components in $G \otimes H$: 1st component will be composed of nodes $\{X_{ij}\} \in G \otimes H$, where $(i \in A, j \in D)$ or $(i \in B, j \in C)$. And similarly, 2nd component will be composed of nodes $\{X_{ij}\}$, where $(i \in A, j \in C)$ or $(i \in B, j \in D)$. Basically, there exist edges between node sets $(A, D)$ and $(B, C)$, and similarly between $(A, C)$ and $(B, D)$ but not across the sets. To see this we have to analyze the cases using observation 1. For example, in $G \otimes H$ there exist edges between nodes $(A, C)$ and $(B, D)$ as there exist edges $(i, k) \in G$ for $i \in A, k \in B$, and $(j, l) \in H$ for $j \in C$ and $l \in D$. Similar is true for nodes $(A, C)$ and $(B, D)$. However, there are no edges cross the two sets, e.g., nodes from $(A, D)$ do not link to $(A, C)$, as there are no edges between nodes in $A$ (since $G$ is bipartite). See Figures 4(d) and 4(e) for a visual proof.

Note that bipartite graphs are triangle free and have no self-loops. For example, stars, chains, trees and cycles of even length are all examples of bipartite graphs. Thus, for the remainder of the paper we will focus on the initiator graphs $K_1$ that have self loops on all of their nodes so that we ensure $K_k$ to be connected.

### 3.2.4 Temporal Properties of Kronecker Graphs

We continue with the analysis of temporal patterns of evolution of Kronecker graphs: the densification power law, and shrinking/stabilizing diameter (Leskovec et al., 2005b, 2007a).

**Theorem 10 (Densification Power Law)** Kronecker graphs follow the Densification Power Law (DPL) with densification exponent $a = \log(E_1) / \log(N_1)$.

**Proof** Since the $k$th Kronecker power $K_k$ has $N_k = N_1^k$ nodes and $E_k = E_1^k$ edges, it satisfies $E_k = N_k^a$, where $a = \log(E_1) / \log(N_1)$. The crucial point is that this exponent $a$ is independent of $k$, and hence the sequence of Kronecker powers follows an exact version of the Densification Power Law.

We now show how the Kronecker product also preserves the property of constant diameter, a crucial ingredient for matching the diameter properties of many real-world network datasets. In order to establish this, we will assume that the initiator graph $K_1$ has a self-loop on every node; otherwise, its Kronecker powers may be disconnected.

**Lemma 11** If $G$ and $H$ each have diameter at most $D$, and each has a self-loop on every node, then the Kronecker graph $G \otimes H$ also has diameter at most $D$.

**Proof** Each node in $G \otimes H$ can be represented as an ordered pair $(v, w)$, with $v$ a node of $G$ and $w$ a node of $H$, and with an edge joining $(v, w)$ and $(x, y)$ precisely when $(v, x)$ is an edge of $G$ and $(w, y)$ is an edge of $H$. (Note this exactly the Observation 1.) Now, for an arbitrary pair of nodes $(v, w)$ and $(v', w')$, we must show that there is a path of length at most $D$ connecting them. Since $G$ has diameter at most $D$, there is a path $v = v_1, v_2, \ldots, v_r = v'$, where $r \leq D$. If $r < D$, we can convert this into a path $v = v_1, v_2, \ldots, v_D = v'$ of length exactly $D$, by simply repeating $v'$ at the end for $D - r$ times. By an analogous argument, we have a path $w = w_1, w_2, \ldots, w_D = w'$. Now
by the definition of the Kronecker product, there is an edge joining \((v_i, w_i)\) and \((v_{i+1}, w_{i+1})\) for all \(1 \leq i \leq D - 1\), and so \((v, w) = (v_1, w_1), (v_2, w_2), \ldots, (v_D, w_D) = (v', w')\) is a path of length \(D\) connecting \((v, w)\) to \((v', w')\), as required.

**Theorem 12** If \(K_1\) has diameter \(D\) and a self-loop on every node, then for every \(k\), the graph \(K_k\) also has diameter \(D\).

**Proof** This follows directly from the previous lemma, combined with induction on \(k\).

As defined in section 2 we also consider the effective diameter \(D^*\); we defined the \(q\)-effective diameter as the minimum \(D^*\) such that, for at least a \(q\) fraction of the reachable node pairs, the path length is at most \(D^*\). The \(q\)-effective diameter is a more robust quantity than the diameter, the latter being prone to the effects of degenerate structures in the graph (e.g., very long chains); however, the \(q\)-effective diameter and diameter tend to exhibit qualitatively similar behavior. For reporting results in subsequent sections, we will generally consider the \(q\)-effective diameter with \(q = 0.9\), and refer to this simply as the effective diameter.

**Theorem 13 (Effective Diameter)** If \(K_1\) has diameter \(D\) and a self-loop on every node, then for every \(q\), the \(q\)-effective diameter of \(K_k\) converges to \(D\) (from above) as \(k\) increases.

**Proof** To prove this, it is sufficient to show that for two randomly selected nodes of \(K_k\), the probability that their distance is \(D\) converges to 1 as \(k\) goes to infinity.

We establish this as follows. Each node in \(K_k\) can be represented as an ordered sequence of \(k\) nodes from \(K_1\), and we can view the random selection of a node in \(K_k\) as a sequence of \(k\) independent random node selections from \(K_1\). Suppose that \(v = (v_1, \ldots, v_k)\) and \(w = (w_1, \ldots, w_k)\) are two such randomly selected nodes from \(K_k\). Now, if \(x\) and \(y\) are two nodes in \(K_1\) at distance \(D\) (such a pair \((x, y)\) exists since \(K_1\) has diameter \(D\)), then with probability \(1 - (1 - 2/N_1^2)^k\), there is some index \(j\) for which \(\{v_j, w_j\} = \{x, y\}\). If there is such an index, then the distance between \(v\) and \(w\) is \(D\). As the expression \(1 - (1 - 2/N_1^2)^k\) converges to 1 as \(k\) increases, it follows that the \(q\)-effective diameter is converging to \(D\).

**3.3 Stochastic Kronecker Graphs**

While the Kronecker power construction discussed so far yields graphs with a range of desired properties, its discrete nature produces “staircase effects” in the degrees and spectral quantities, simply because individual values have large multiplicities. For example, degree distribution and distribution of eigenvalues of graph adjacency matrix and the distribution of the principal eigenvector components (i.e., the “network” value) are all impacted by this. These quantities are multinomially distributed which leads to individual values with large multiplicities. Figure 5 illustrates the staircase effect.

Here we propose a stochastic version of Kronecker graphs that eliminates this effect. There are many possible ways how one could introduce stochasticity into Kronecker graphs model. Before introducing the proposed model, we introduce two simple ways of introducing randomness to Kronecker graphs and describe why they do not work.
Probably the simplest (but wrong) idea is to generate a large deterministic Kronecker graph $K_k$, and then uniformly at random flip some edges, i.e., uniformly at random select entries of the graph adjacency matrix and flip them ($1 \rightarrow 0$, $0 \rightarrow 1$). However, this will not work, as it will essentially superimpose a Erdős-Rényi random graph, which would, for example, corrupt the degree distribution – real networks usually have heavy tailed degree distributions, while random graphs have Binomial degree distributions. A second idea could be to allow a weighted initiator matrix, i.e., values of entries of $K_1$ are not restricted to values $\{0, 1\}$ but rather can be any non-negative real number. Using such $K_1$ one would generate $K_k$ and then threshold the $K_k$ matrix to obtain a binary adjacency matrix $K$, i.e., for a chosen value of $\epsilon$ set $K[i, j] = 1$ if $K_k[i, j] > \epsilon$ else $K[i, j] = 0$. This also would not work as the mechanism would selectively remove edges and thus the low degree nodes which would have low weight edges would get isolated first.

Now we define Stochastic Kronecker Graphs model that overcomes the above issues. A more natural way to introduce stochasticity to Kronecker graphs is to relax the assumption that entries of the initiator matrix take only binary values. Now, we will allow entries of the initiator to take values on the interval $[0, 1]$. This means now each entry of the initiator matrix encodes the probability of that particular edge appearing. We then Kronecker power such initiator matrix to obtain a large stochastic adjacency matrix, where again each entry of the large matrix gives the probability of that particular edge appearing in a big graph. Such stochastic adjacency matrix effectively defines a probability distribution over all graphs. To obtain a graph we simply sample an instance from this distribution by sampling individual edges, where each edge appears independently with probability given by the entry of the large stochastic adjacency matrix. More formally, we define:

**Definition 14 (Stochastic Kronecker Graph)** Let $\mathcal{P}_1$ be a $N_1 \times N_1$ probability matrix: the value $\theta_{ij} \in \mathcal{P}_1$ denotes the probability that edge $(i, j)$ is present, $\theta_{ij} \in [0, 1]$.

Then $k^{th}$ Kronecker power $\mathcal{P}_1^{[k]} = \mathcal{P}_k$, where each entry $p_{uv} \in \mathcal{P}_k$ encodes the probability of an edge $(u, v)$.

To obtain a graph, an instance (or realization), $K = R(\mathcal{P}_k)$ we include edge $(u, v)$ in $K$ with probability $p_{uv}$, $p_{uv} \in \mathcal{P}_k$. 

Figure 5: The “staircase” effect. Kronecker initiator and the degree distribution and network value plot for the $6^{th}$ Kronecker power of the initiator. Notice the non-smoothness of the curves.
First, note that sum of the entries of $P_1$, $\sum_{ij} \theta_{ij}$, can be greater than 1. Second, notice that in principle it takes $O(N_1^{2k})$ time to generate an instance $K$ of a Stochastic Kronecker graph from the probability matrix $P_k$. This means the time to get a realization $K$ is quadratic in the size of $P_k$ as one has to flip a coin for each possible edge in the graph. Later we show how to generate Stochastic Kronecker graphs much faster, in the time linear in the expected number of edges in $P_k$.

3.3.1 Probability of an Edge

For the size of the graphs we aim to model and generate here taking $P_1$ (or $K_1$) and then explicitly performing the Kronecker product of the initiator matrix is infeasible. The reason for this is that $P_1$ is usually dense, so $P_k$ is also dense and one cannot store it in memory. However, due to the structure of Kronecker multiplication one can easily compute the probability of an edge in $P_k$.

The probability $p_{uv}$ of an edge $(u,v)$ occurring in $k$-th Kronecker power $P = P_k$ can be calculated in $O(k)$ time as follows:

$$p_{uv} = \prod_{i=0}^{k-1} P\left[\left\lfloor \frac{u-1}{N_1}\right\rfloor (\text{mod} N_1) + 1, \left\lfloor \frac{v-1}{N_1}\right\rfloor (\text{mod} N_1) + 1\right]$$

The equation imitates recursive descent into the matrix $P$, where at every level $i$ the appropriate entry of $P_1$ is chosen. Since $P$ has $N_1^k$ rows and columns it takes $O(k \log N_1)$ to evaluate the equation. Refer to figure 6 for the illustration of the recursive structure of $P$.

3.4 Additional properties of Kronecker graphs

Stochastic Kronecker Graphs with initiator matrix of size $N_1 = 2$ were studied by Mahdian and Xu (Mahdian and Xu, 2007). The authors showed a phase transition for the emergence of the giant component and another phase transition for connectivity, and proved that such graphs have constant diameters beyond the connectivity threshold, but are not searchable using a decentralized algorithm (Kleinberg, 1999).

Moreover, recently (Tsourakakis, 2008) gave a closed form expression for the number of triangles in a Kronecker graph that depends on the eigenvalues of the initiator graph $K_1$.

3.5 Two interpretations of Kronecker graphs

Next, we present two natural interpretations of the generative process behind the Kronecker Graphs that go beyond the purely mathematical construction of Kronecker Graphs as introduced so far.

We already mentioned the first interpretation when we first defined Kronecker Graphs. One intuition is that networks and communities in them grow recursively, creating miniature copies of themselves. Figure 1 depicts the process of the recursive community expansion. In fact, several researchers have argued that real networks are hierarchically organized (Ravasz et al., 2002; Ravasz and Barabási, 2003) and algorithms to extract the network hierarchical structure have also been developed (Sales-Pardo et al., 2007; Clauset et al., 2008). Moreover, especially web graphs (Dill et al., 2002; Dorogovtsev et al., 2002; Crovella and Bestavros, 1997) and biological networks (Ravasz and Barabási, 2003) were found to be self-similar and “fractal”.

The second interpretation comes from viewing every node of $P_k$ as being described with an ordered sequence of $k$ nodes from $P_1$. (This is similar to the Observation 1 and the proof of Theorem 13.)
### Kronecker Graphs

**Figure 6:** Stochastic Kronecker initiator $P_1$ and the corresponding $2^{nd}$ Kronecker power $P_2$. Notice the recursive nature of the Kronecker product, with edge probabilities in $P_2$ simply being products of entries of $P_1$.

Let’s label nodes of the initiator matrix $P_1$, $u_1, \ldots, u_{N_1}$, and nodes of $P_k$ as $v_1, \ldots, v_{N_k}$. Then every node $v_i$ of $P_k$ is described with a sequence $(v_i(1), \ldots, v_i(k))$ of node labels of $P_1$, where $v_i(l) \in \{u_1, \ldots, u_k\}$. Similarly, consider also a second node $v_j$ with the label sequence $(v_j(1), \ldots, v_j(k))$. Then the probability $p_e$ of an edge $(v_i, v_j)$ in $P_k$ is exactly:

$$p_e(v_i, v_j) = P_k[v_i, v_j] = \prod_{l=1}^{k} P_1[v_i(l), v_j(l)]$$

(Note this is exactly the Equation 2.)

Now one can look at the description sequence of node $v_i$ as a $k$ dimensional vector of attribute values $(v_i(1), \ldots, v_i(k))$. Then $p_e(v_i, v_j)$ is exactly the coordinate-wise product of appropriate entries of $P_1$, where the node description sequence selects which entries to multiply. Thus, the $P_1$ matrix can be thought of as the attribute similarity matrix, i.e., it encodes the probability of linking given that two nodes agree/disagree on the attribute value. Then the probability of an edge is simply a product of individual attribute similarities over the $k$ $N_1$-ary attributes that describe each of the two nodes.

This gives us a very natural interpretation of Stochastic Kronecker graphs: Each node is described by a sequence of categorical attribute values or features. And then the probability of two nodes linking depends on the product of individual attribute similarities. This way Kronecker graphs can effectively model homophily (nodes with similar attribute values are more likely to link) by $P_1$ having high value entries on the diagonal; or heterophily (nodes that differ are more likely to link) by $P_1$ having high entries off the diagonal.

Figure 6 shows an example. Let’s label nodes of $P_1$ $u_1, u_2$ as in Figure 6(a). Then every node of $P_k$ is described with an ordered sequence of $k$ binary attributes. For example, Figure 6(b) shows an instance for $k = 2$ where node $v_2$ of $P_2$ is described by $(u_1, u_2)$, and similarly $v_3$ by $(u_2, u_1)$. Then as shown in Figure 6(b), the probability of edge $p_e(v_2, v_3) = b \cdot c$, which is exactly $P_1[u_2, u_1] \cdot P_1[u_1, u_2] = b \cdot c$ — the product of entries of $P_1$, where the corresponding elements of the description of nodes $v_2$ and $v_3$ act as selectors of which entries of $P_1$ to multiply.

![Kronecker Graphs](image-url)
Figure 6(c) further illustrates the recursive nature of Kronecker graphs. One can see Kronecker product as recursive descent into the big adjacency matrix where at each stage one of the entries or blocks is chosen. For example, to get to entry \((v_2, v_3)\) one first needs to dive into quadrant \(b\) following by the quadrant \(c\). This intuition will help us in section 3.6 to devise a fast algorithm for generating Kronecker graphs.

However, there are also two notes to make here. First, using a single initiator \(P_1\) we are implicitly assuming that there is one single and universal attribute similarity matrix that holds across all \(k\) \(N_1\)-ary attributes. One can easily relax this assumption by taking a different initiator matrix for each attribute (initiator matrices can even be of different sizes as attributes are of different arity), and then Kronecker multiplying them to obtain a large network. Here each initiator matrix plays the role of attribute similarity matrix for that particular attribute.

For simplicity and convenience we will work with a single initiator matrix but all our methods can be trivially extended to handle multiple initiator matrices. Moreover, as we will see later in section 6 even a single \(2 \times 2\) initiator matrix seems to be enough to capture large scale statistical properties of real-world networks.

The second assumption is harder to relax. When describing every node \(v_i\) with a sequence of attribute values we are implicitly assuming the values of all attributes are uniformly distributed (have same proportions), and that every node has a unique combination of attribute values. So, all possible combinations of attribute values are taken. For example, node \(v_1\) in a large \(P_1\) has attribute sequence \((u_1, u_1, \ldots, u_1)\), \(v_{N_1}\) has \((u_1, u_1, \ldots, u_1, u_{N_1})\), while the “last” node \(v_{N_1^k}\) is has attribute values \((u_{N_1}, u_{N_1}, \ldots, u_{N_1})\). One can think of this as counting in \(N_1\)-ary number system, where node attribute descriptions range from 0 (i.e., “leftmost” node with attribute description \((u_1, u_1, \ldots, u_1)\)) to \(N_1^k\) (i.e., “rightmost” node attribute description \((u_{N_1}, u_{N_1}, \ldots, u_{N_1})\)).

A simple way to relax the above assumption is to take a larger initiator matrix with a smaller number of parameters than the number of entries. This means that multiple entries of \(P_1\) will share the same value (parameter). For example, if attribute \(u_1\) takes one value 66% of the times, and the other value 33% of the times, then one can model this by taking a \(3 \times 3\) initiator matrix with only four parameters. Adopting the naming convention of Figure 6 this means that parameter \(a\) now occupies a \(2 \times 2\) block, which then also makes \(b\) and \(c\) occupy \(2 \times 1\) and \(1 \times 2\) blocks, and \(d\) a single cell. This way one gets a four parameter model with uneven feature value distribution.

We note that the view of Kronecker graphs where every node is described with a set of features and the initiator matrix encodes the probability of linking given the attribute values of two nodes somewhat resembles the Random dot product graphs model (Young and Scheinerman, 2007; Nickel, 2008). The important difference here is that we multiply individual linking probabilities, while in Random dot product graphs one takes the sum of individual probabilities which seems somewhat less natural.

3.6 Fast generation of Stochastic Kronecker Graphs

The intuition for fast generation of Stochastic Kronecker Graphs comes from the recursive nature of the Kronecker product and is closely related to the R-MAT graph generator (Chakrabarti et al., 2004). Generating a Stochastic Kronecker graph \(K\) on \(N\) nodes naively takes \(O(N^2)\) time. Here we present a linear time \(O(E)\) algorithm, where \(E\) is the (expected) number of edges in \(K\).
Figure 6(c) shows the recursive nature of the Kronecker product. To “arrive” to a particular edge \((v_i, v_j)\) of \(P_k\) one has to make a sequence of \(k\) (in our case \(k = 2\)) decisions among the entries of \(P_1\), multiply the chosen entries of \(P_1\), and then placing the edge \((v_i, v_j)\) with the obtained probability.

Instead of flipping \(O(N^2) = O(N^{2k})\) biased coins to determine the edges, we can place \(E\) edges by directly simulating the recursion of the Kronecker product. Basically we recursively choose sub-regions of matrix \(K\) with probability proportional to \(\theta_{ij}\), \(\theta_{ij} \in P_1\) until in \(k\) steps we descend to a single cell of the matrix and place an edge. For example, for \((v_2, v_3)\) in Figure 6(c) we first have to choose \(b\) following by \(c\).

The probability of each individual edge of \(P_k\) follows a Bernoulli distribution, as the edge occurrences are independent. By the Central Limit Theorem the number of edges in \(P_k\) tends to a normal distribution with mean \((\sum_{i,j=1}^{N_1} \theta_{ij})^k = E_1^k\), where \(\theta_{ij} \in P_1\). So, given a stochastic initiator matrix \(P_1\) we first sample the expected number of edges \(E\) in \(P_k\). Then we place \(E\) edges in a graph \(K\), by applying the recursive descent for \(k\) steps where at each step we choose entry \((i, j)\) with probability \(\theta_{ij}/E_1\) where \(\theta_{ij} \in P_1\) and \(E_1 = \sum_{ij} \theta_{ij}\). Since we add \(E = E_1^k\) edges, the probability that edge \((v_i, v_j)\) appears in \(K\) is exactly \(P_k[v_i, v_j]\). This basically means that in Stochastic Kronecker Graphs the initiator matrix encodes both the total number of edges in a graph and their structure. \(\sum \theta_{ij}\) encodes the number of edges in the graph, while the proportions (ratios) of values \(\theta_{ij}\) define how many edges each part of graph adjacency matrix will contain.

In practice it can happen that more than one edge lands in the same \((v_i, v_j)\) cell of \(K\). Even though values of \(P_1\) are usually skewed, adjacency matrices of real network are sparse which mitigates the problem.

### 3.7 Observations and connections

Next, we describe several observations about the properties of Kronecker graphs and make connections to other network models.

- **Bipartite graphs:** Kronecker Graphs can naturally model bipartite graphs. Instead of starting with a square \(N_1 \times N_1\) initiator matrix, one can choose arbitrary \(N_1 \times M_1\) initiator matrix, where rows define “left”, and columns the “right” side of the bipartite graph. Kronecker multiplication will then generate bipartite graphs with partition sizes \(N_1^k\) and \(M_1^k\).

- **Graph distributions:** \(P_k\) defines a distribution over all graphs, as it encodes the probability of all possible \(N_1^{2k}\) edges appearing in a graph by using an exponentially smaller number of parameters (just \(N_1^2\)). As we will later see, even a very small number of parameters, e.g., 4 (\(2 \times 2\) initiator matrix) or 9 (\(3 \times 3\) initiator), is enough to accurately model the structure of large networks.

- **Natural extension of Erdős-Rényi random graph model:** Stochastic Kronecker Graphs represent a natural extension of Erdős-Rényi (Erdős and Rényi, 1960) random graphs. If one takes \(P_1 = [\theta_{ij}]\), where every \(\theta_{ij} = p\) then we obtain exactly the Erdős-Rényi model of random graphs \(G_{n,p}\), where every node appears independently with probability \(p\).

- **Relation to the R-MAT model:** The recursive nature of Stochastic Kronecker Graphs makes them related to the R-mat generator (Chakrabarti et al., 2004). The difference between the two models is that in R-mat one needs to separately specify the number of edges, while in Stochastic Kronecker Graphs initiator matrix \(P_1\) also encodes the number of edges in the
graph. Section 3.6 built on this similarity to devise a fast algorithm for generating Stochastic Kronecker graphs.

- **Densification:** Similarly as with deterministic Kronecker graphs the number of nodes in a Stochastic Kronecker Graph grows as $N_1^k$, and the expected number of edges grows as $(\sum_{ij} \theta_{ij})^k$. This means one would want to choose values $b_{ij}$ of the initiator matrix $P_1$ so that $\sum_{ij} \theta_{ij} > N_1$ in order for the resulting network to densify.

4. Simulations of Kronecker graphs

In previous section we proved and now we demonstrate using simulation the ability of Kronecker graphs to match the patterns of real-world networks. We will tackle the problem of estimating the Kronecker Graphs model from real data, *i.e.*, finding the most likely initiator $P_1$, in the next section. Instead here we present simulation experiments using Kronecker graphs to explore the parameter space, and to compare properties of Kronecker Graphs to those found in large real networks.

4.1 Comparison to real graphs

We observe two kinds of graph patterns — “static” and “temporal.” As mentioned earlier, common static patterns include degree distribution, scree plot (eigenvalues of graph adjacency matrix vs. rank) and distribution of components of the principal eigenvector of graph adjacency matrix. Temporal patterns include the diameter over time, and the densification power law. For the diameter computation, we use the effective diameter as defined in Section 2.

For the purpose of this section consider the following setting. Given a real graph $G$ we want to find Kronecker initiator that produces qualitatively similar graph. In principle one could try choosing each of the $N_1^2$ parameters for the matrix $P_1$ separately. However, we reduce the number of parameters from $N_1^2$ to just two: $\alpha$ and $\beta$. Let $K_1$ be the initiator matrix (binary, deterministic); we create the corresponding stochastic initiator matrix $P_1$ by replacing each “1” and “0” of $K_1$ with $\alpha$ and $\beta$ respectively ($\beta \leq \alpha$). The resulting probability matrices maintain — with some random noise — the self-similar structure of the Kronecker graphs in the previous section (which, for clarity, we call *deterministic Kronecker graphs*). We defer the discussion of how to estimate $P_1$ from data $G$ to the next section.

The datasets we use here are:

- **CIT-HEP-TH:** This is a citation graph for High-Energy Physics Theory research papers from pre-print archive ArXiv, with a total of $N = 29,555$ papers and $E = 352,807$ citations (Gehrke et al., 2003). We follow its evolution from January 1993 to April 2003, with one data-point per month.

- **AS-ROUTEVIEWS:** We also analyze a static dataset consisting of a single snapshot of connectivity among Internet Autonomous Systems (RouteViews, 1997) from January 2000, with $N = 6,474$ and $E = 26,467$.

Results are shown in Figure 7 for the CIT-HEP-TH graph which evolves over time. We show the plots of two static and two temporal patterns. We see that the deterministic Kronecker model already captures the qualitative structure of the degree and eigenvalue distributions, as well as the temporal patterns represented by the Densification Power Law and the stabilizing diameter. However, the
Figure 7: Citation network (CIT-HEP-TH): Patterns from the real graph (top row), the deterministic Kronecker graph with $K_1$ being a star graph on 4 nodes (center + 3 satellites) (middle row), and the Stochastic Kronecker graph ($\alpha = 0.41$, $\beta = 0.11$ – bottom row). Static patterns: (a) is the PDF of degrees in the graph (log-log scale), and (b) the distribution of eigenvalues (log-log scale). Temporal patterns: (c) gives the effective diameter over time (linear-linear scale), and (d) is the number of edges versus number of nodes over time (log-log scale). Notice that the Stochastic Kronecker Graph qualitatively matches all the patterns very well.

The deterministic nature of this model results in “staircase” behavior, as shown in scree plot for the deterministic Kronecker graph of Figure 7 (column (b), second row). We see that the Stochastic Kronecker Graphs smooth out these distributions, further matching the qualitative structure of the real data; they also match the shrinking-before-stabilization trend of the diameters of real graphs.

Similarly, Figure 8 shows plots for the static patterns in the Autonomous systems (AS-ROUTEVIEW) graph. Recall that we analyze a single, static network snapshot in this case. In addition to the degree distribution and scree plot, we also show two typical plots (Chakrabarti et al., 2004): the distribution of network values (principal eigenvector components, sorted, versus rank) and the hop-plot (the number of reachable pairs $g(h)$ within $h$ hops or less, as a function of the number of hops $h$). Notice that, again, the Stochastic Kronecker graph matches well the properties of the real graph.
Figure 8: Autonomous systems (AS-ROUTEVIEWS): Real (top) versus Kronecker (bottom). Columns (a) and (b) show the degree distribution and the scree plot, as before. Columns (c) and (d) show two more static patterns (see text). Notice that, again, the Stochastic Kronecker graph matches well the properties of the real graph.

Figure 9: Diameter over time for a 4-node chain initiator graph. After each consecutive Kronecker power we measure the effective diameter. We use different settings of \( \alpha \) parameter. \( \alpha = 0.38, 0.43, 0.54 \) and \( \beta = 0 \), respectively.

4.2 Parameter space of Kronecker Graphs

Last we present simulation experiments that investigate the parameter space of Stochastic Kronecker Graphs.

First, in Figure 9 we show the ability of Kronecker Graphs to generate networks with increasing, constant and decreasing/stabilizing effective diameter. We start with a 4-node chain initiator graph, setting each “1” of \( K_1 \) to \( \alpha \) and each “0” to \( \beta = 0 \) to obtain \( P_1 \) that we then use to generate a growing sequence of graphs. We plot the effective diameter of each \( R(P_k) \) as we generate a
sequence of growing graphs $R(P_2), R(P_3), \ldots, R(P_{10})$. $R(P_{10})$ has exactly 1,048,576 nodes. Notice Stochastic Kronecker graphs is a very flexible model. When the generated graph is very sparse (low value of $\alpha$) we obtain graphs with slowly increasing effective diameter (Figure 9(a)). For intermediate values of $\alpha$ we get graphs with constant diameter (Figure 9(b)) and that in our case also slowly densify with densification exponent in $a = 1.05$. Last, we see an example of a graph with shrinking/stabilizing effective diameter. Here we set the $\alpha = 0.54$ which results in a densification exponent of 1.2. Note that these observations are not contradicting Theorem 11. Actually, these simulations here agree well with the analysis of (Mahdian and Xu, 2007).

Next, we examine the parameter space of a Stochastic Kronecker graph where we choose a star on 4 nodes as a initiator graph and use the familiar parameterization, using $\alpha$ and $\beta$. The initiator graph and the structure of the corresponding (deterministic) Kronecker graph adjacency matrix is shown in top row of Figure 3.

Figure 10(a) shows the sharp transition in the fraction of the number of nodes that belong to the largest weakly connected component as we fix $\beta = 0.15$ and slowly increase $\alpha$. Such phase transitions on the size of the largest connected component also occur in Erdős-Rényi random graphs. Figure 10(b) further explores this by plotting the fraction of nodes in the largest connected component ($N_c/N$) over the full parameter space. Notice a sharp transition between disconnected (white area) and connected graphs (dark).

Last, Figure 10(c) shows the effective diameter over the parameter space ($\alpha, \beta$) for the 4-node star initiator graph. Notice that when parameter values are small, the effective diameter is small, since the graph is disconnected and not many pairs of nodes can be reached. The shape of the transition between low-high diameter closely follows the shape of the emergence of the connected component. Similarly, when parameter values are large, the graph is very dense, and the diameter is small. There is a narrow band in parameter space where we get graphs with interesting diameters.
5. Kronecker graph model estimation

In previous sections we proved that shapes (parametric forms) of various network properties of Kronecker graphs follow those found in real networks. Moreover, we also gave closed form expressions that allow us to calculate a property (e.g., diameter, eigenvalue spectrum) of a network given just the initiator matrix. So in principle, one could invert the equations and directly get from a property (e.g., shape of degree distribution) to the values of initiator matrix.

However, in previous sections we did not say anything about how various network properties of a Kronecker graph correlate and interdepend. For example, it could be the case that they are mutually exclusive. So one could, for instance, only match the network diameter but not the degree distribution or vice versa. However, as we show later this is not the case.

Now we turn our attention to automatically estimating the Kronecker initiator graph. The setting is that we are given a real network \( G \) and would like to find a Stochastic Kronecker initiator \( P_1 \) that produces a synthetic Kronecker graph \( K \) that is “similar” to \( G \). One way to measure similarity is to compare statistical network properties, like diameter and degree distribution, of graphs \( G \) and \( K \).

Comparing statistical properties already suggests a very direct approach to this problem: One could first identify the set of statistics to match, then define an error metric and somehow optimize over it. For example, one could use the KL divergence (Kullback and Leibler, 1951), or the sum of squared differences between the degree distribution of the real network \( G \) and its synthetic counterpart \( K \). Moreover, as we are interested in matching several such statistics between the networks one would have to meaningfully combine these individual error metrics into a global error metric. So, one would have to specify what kind of properties he or she cares about and then combine them accordingly. This would be a hard task as the patterns of interest have very different magnitudes and scales. Moreover, as new network patterns are discovered, the error functions would have to be changed and models re-estimated. And even then it is not clear how to define the optimization procedure and how to perform optimization over the parameter space.

Our approach here is different. Instead of committing to a set of network properties ahead of time, we will try to directly match the adjacency matrices of the real network \( G \) and its synthetic counterpart \( K \). The idea is that if the adjacency matrices are similar then the global statistical properties (statistics computed over \( K \) and \( G \)) will also match. Moreover, by directly working with the graph itself (and not summary statistics), we do not commit to any particular set of network statistics (network properties/patterns) and as new statistical properties of networks are discovered our models and estimated parameters still hold.

5.1 Preliminaries

Stochastic graph models introduce probability distributions over graphs. A generative model assigns a probability \( P(G) \) to every graph \( G \). \( P(G) \) is the likelihood that a given model (with a given set of parameters) generated graph \( G \). We concentrate on the Stochastic Kronecker Graph model, and consider fitting it to a real graph \( G \), our data. We use the maximum likelihood approach, \textit{i.e.}, we aim to find parameter values, the initiator \( P_1 \), that maximize the \( P(G) \) under the Stochastic Kronecker model.

This presents several challenges:

- \textbf{Model selection:} a graph is a single structure, and not a set of items drawn i.i.d. from some distribution. So one cannot split it into independent training and test sets. The fitted
parameters will thus be best to generate a particular instance of a graph. Also, overfitting could be an issue since a more complex model generally fits better.

- **Node correspondence**: The second challenge is the node correspondence or node labeling problem. Graph $G$ has a set of $N$ nodes, and each node has a unique index (label, id). Labels do not carry any particular meaning, they just uniquely denote or identify the nodes. One can think of this as the graph is first generated and then the labels (node ids) are randomly assigned. This means that two isomorphic graphs that have different node ids should have the same likelihood. A permutation $\sigma$ is sufficient to describe the node correspondences as it maps labels (ids) to nodes of the graph. To compute the likelihood $P(G)$ one has to consider all node correspondences $P(G) = \sum_\sigma P(G|\sigma)P(\sigma)$, where the sum is over all $N!$ permutations $\sigma$ of $N$ nodes. Calculating this super-exponential sum explicitly is infeasible for any graph with more than a handful of nodes. Intuitively, one can think of this summation as some kind of graph isomorphism test where we are searching for best correspondence (mapping) between nodes of $G$ and $P$.

- **Likelihood estimation**: Calculating $P(G|\sigma)$ naively takes $O(N^2)$ as one has to evaluate the probability of each of the $N^2$ possible edges in the graph adjacency matrix. Again, for graphs of size we want to model here, approaches with quadratic complexity are infeasible.

To develop our solution we use sampling to avoid the super-exponential sum over the node correspondences. By exploiting the structure of the Kronecker matrix multiplication we develop an algorithm to evaluate $P(G|\sigma)$ in linear time $O(E)$. Since real graphs are sparse, i.e., the number of edges is roughly of the same order as the number of nodes, this makes fitting of Kronecker Graphs to large networks feasible.

### 5.2 Problem formulation

Suppose we are given a graph $G$ on $N = N_1^k$ nodes (for some positive integer $k$), and an $N_1 \times N_1$ Stochastic Kronecker Graph initiator matrix $P_1$. Here $P_1$ is a parameter matrix, a set of parameters that we aim to estimate. For now also assume $N_1$, the size of the initiator matrix, is given. Later we will show how to automatically select it. Next, using $P_1$ we create a Stochastic Kronecker Graph probability matrix $P_k$, where every entry $p_{uv}$ of $P_k$ contains a probability that node $u$ links to node $v$. We then evaluate the probability that $G$ is a realization of $P_k$. The task is to find such $P_1$ that has the highest probability of realizing (generating) $G$.

Formally, we are solving:

$$\arg \max_{P_1} P(G|P_1)$$

(3)

To keep the notation simpler we use standard symbol $\Theta$ to denote the parameter matrix $P_1$ that we are trying to estimate. We denote entries of $\Theta = P_1 = [\theta_{ij}]$, and similarly we denote $P = P_k = [p_{ij}]$. Note that here we slightly simplified the notation: we use $\Theta$ to refer to $P_1$, and $\theta_{ij}$ are elements of $\Theta$. Similarly, $p_{ij}$ are elements of $P$ (i.e., $P_k$). Moreover, we denote $K = R(P)$, i.e., $K$ is a realization of the Stochastic Kronecker graph sampled from probabilistic adjacency matrix $P$.

As noted before, the node ids are assigned arbitrarily and they carry no significant information, which means that we have to consider all the mappings of nodes from $G$ to rows and columns of
stochastic adjacency matrix $\mathcal{P}$. A priori all labelings are equally likely. A permutation $\sigma$ of the set \{1, \ldots, N\} defines this mapping of nodes from $G$ to stochastic adjacency matrix $\mathcal{P}$. To evaluate the likelihood of $G$ one needs to consider all possible mappings of $N$ nodes of $G$ to rows (columns) of $\mathcal{P}$. For convenience we work with log-likelihood $l(\Theta)$, and solve $\hat{\Theta} = \arg \max \Theta \ l(\Theta)$, where $l(\Theta)$ is defined as:

$$l(\Theta) = \log P(G|\Theta) = \log \sum_{\sigma} P(G|\Theta, \sigma)P(\sigma|\Theta)$$

$$= \log \sum_{\sigma} P(G|\Theta, \sigma)P(\sigma) \quad (4)$$

The likelihood that a given initiator matrix $\Theta$ and permutation $\sigma$ gave rise to the real graph $G$, $P(G|\Theta, \sigma)$, is calculated naturally as follows. First, by using $\Theta$ we create the Stochastic Kronecker graph adjacency matrix $\mathcal{P} = \mathcal{P}_k = \Theta^{[k]}$. Permutation $\sigma$ defines the mapping of nodes of $G$ to the rows and columns of stochastic adjacency matrix $\mathcal{P}$. (See Figure 11 for an illustration.)

We then model edges as independent Bernoulli random variables parameterized by the parameter matrix $\Theta$. So, each entry $p_{uv}$ of $\mathcal{P}$ gives exactly the probability of edge $(u,v)$ appearing.

We then define the likelihood:

$$P(G|\mathcal{P}, \sigma) = \prod_{(u,v) \in G} \mathcal{P}[\sigma_u, \sigma_v] \prod_{(u,v) \notin G} (1 - \mathcal{P}[\sigma_u, \sigma_v]), \quad (5)$$

where we denote $\sigma_i$ as the $i^{th}$ element of the permutation $\sigma$, and $\mathcal{P}[i, j]$ is the element at row $i$, and column $j$ of matrix $\mathcal{P} = \Theta^{[k]}$.

The likelihood is defined very naturally. We traverse the entries of adjacency matrix $G$ and then based on whether a particular edge appeared in $G$ or not we take the probability of edge occurring (or not) as given by $\mathcal{P}$, and multiply these probabilities. As one has to touch all the entries of the stochastic adjacency matrix $\mathcal{P}$ evaluating Equation 5 takes $O(N^2)$.

We further illustrate the process of estimating Stochastic Kronecker initiator matrix $\Theta$ in Figure 11. We search over initiator matrices $\Theta$ to find the one that maximizes the likelihood $P(G|\Theta)$.
To estimate $P(G|\Theta)$ we are given a concrete $\Theta$ and now we use Kronecker multiplication to create probabilistic adjacency matrix $\Theta^k$ that is of same size as real network $G$. Now, we evaluate the likelihood by traversing the corresponding entries of $G$ and $\Theta^k$. Equation 5 basically traverses the adjacency matrix of $G$, and maps every entry $(u, v)$ of $G$ to a corresponding entry $(\sigma_u, \sigma_v)$ of $\Theta^k$. This also demonstrates the importance of permutation $\sigma$, as permuting rows and columns of $G$ could make the adjacency matrix looking more “similar” to $\Theta^k$, and would increase the likelihood.

So far we showed how to asses the quality (likelihood) of a particular $\Theta$. So, naively one could perform some kind of exhaustive grid search to find best $\Theta$. However, this is very inefficient. A better way of doing it is to compute the gradient of the log-likelihood $\frac{\partial}{\partial \Theta} \log l(\hat{\Theta})$, and then use the gradient to update the current estimate of $\Theta$ and move towards a solution of higher likelihood. Algorithm 1 gives an outline of the optimization procedure.

However, there are several difficulties with this algorithm. First, we are assuming gradient descent type optimization will work, i.e. the problem does not have (too many) local minima. Second, we are summing over exponentially many permutations in equation 4. Third, the evaluation of equation 5 as it is written takes $O(N^2)$ and needs to be evaluated $N!$ times. So, just naively calculating the likelihood takes $O(N!N^2)$.

**Observation 2** The complexity of calculating the likelihood $P(G|\Theta)$ of the graph $G$ naively is $O(N!N^2)$, where $N$ is the number of nodes in $G$.

Next, we show that all this can be done in linear time.

### 5.3 Summing over the node labelings

To maximize equation 3 using algorithm 1 we need to obtain the gradient of the log-likelihood $\frac{\partial}{\partial \Theta} \log l(\Theta)$. We can write:

$$
\frac{\partial}{\partial \Theta} \log l(\Theta) = \sum_\sigma \frac{\partial}{\partial \Theta} \log P(G|\sigma, \Theta) P(\sigma)
$$

$$
= \sum_\sigma \frac{\partial}{\partial \Theta} \log P(G|\sigma, \Theta) \frac{P(G)}{P(\sigma)}
$$

$$
= \sum_\sigma \frac{\partial}{\partial \Theta} \log P(G|\sigma, \Theta) P(\sigma|G, \Theta)
$$

Note we are still summing over all $N!$ permutations $\sigma$, so calculating eq. 6 is computationally intractable for graphs with more than a handful of nodes. However, the equation has a nice form which allows for use of simulation techniques to avoid the summation over super-exponentially many node correspondences. Thus, we simulate draws from the permutation distribution $P(\sigma|G, \Theta)$, and then evaluate the quantities at the sampled permutations to obtain the expected values of log-likelihood and gradient. Algorithm 2 gives the details.

Note that we can also permute the rows and columns of the parameter matrix $\Theta$ to obtain equivalent estimates; therefore $\Theta$ is not strictly identifiable exactly because of these permutations. Since
**Algorithm 1: KRONFit algorithm.**

**Algorithm 2: Calculating log-likelihood and gradient**

The space of permutations on $N$ nodes is very large (grows as $N!$) the MCMC sampler will explore only a small fraction of the space of all permutations and may converge to one of the global maxima (but may not explore all $N_1!$ of them) of the parameter space. As we empirically show later our results are not sensitive to this and multiple restarts result in equivalent (but often permuted) parameter estimates.

### 5.3.1 Sampling Permutations

Next, we describe the Metropolis algorithm to simulate draws from the permutation distribution $P(\sigma | G, \Theta)$, which is given by

$$P(\sigma | G, \Theta) = \frac{P(\sigma, G, \Theta)}{\sum_{\sigma} P(\sigma, G, \Theta)} = \frac{\sum_{\sigma} P(\sigma, G, \Theta)}{Z_{\sigma}}$$

where $Z_{\sigma}$ is the normalizing constant that is hard to compute since it involves the sum over $N!$ elements. However, if we compute the likelihood ratio between permutations $\sigma$ and $\sigma'$ (Equation 7) the normalizing constants nicely cancel out:

$$\frac{P(\sigma' | G, \Theta)}{P(\sigma | G, \Theta)} = \prod_{(u,v) \in G} \frac{P[\sigma_u, \sigma_v]}{P[\sigma'_u, \sigma'_v]} \prod_{(u,v) \notin G} \frac{(1 - P[\sigma_u, \sigma_v])}{(1 - P[\sigma'_u, \sigma'_v])}$$

$$= \prod_{(u,v) \in G} \frac{P[\sigma_u, \sigma_v]}{P[\sigma'_u, \sigma'_v]} \prod_{(u,v) \notin G} \frac{(1 - P[\sigma_u, \sigma_v])}{(1 - P[\sigma'_u, \sigma'_v])}$$

(Equation 8)
**Kronecker Graphs**

**input**: Kronecker initiator matrix $\Theta$ and a graph $G$ on $N$ nodes

**output**: Permutation $\sigma(i) \sim P(\sigma|G, \Theta)$

1. $\sigma(0) := (1, \ldots, N)$
2. $i = 1$
3. **repeat**
4. Draw $j$ and $k$ uniformly from $(1, \ldots, N)$
5. $\sigma(i) := \text{SwapNodes}(\sigma(i-1), j, k)$
6. Draw $u$ from $U(0, 1)$
7. **if** $u > \frac{P(\sigma(i)|G, \Theta)}{P(\sigma(i-1)|G, \Theta)}$ **then**
8. $\sigma(i) := \sigma(i-1)$
9. **end**
10. $i = i + 1$
11. **until** $\sigma(i) \sim P(\sigma|G, \Theta)$
12. **return** $\sigma(i)$
13. Where $U(0, 1)$ is a uniform distribution on $[0, 1]$, and $\sigma' := \text{SwapNodes}(\sigma, j, k)$ is the permutation $\sigma'$ obtained from $\sigma$ by swapping elements at positions $j$ and $k$.

**Algorithm 3**: SamplePermutation $(G, \Theta)$: Metropolis sampling of the node permutation.

This immediately suggests to use of Metropolis sampling algorithm (Gamerman, 1997) to simulate draws from the permutation distribution since Metropolis is solely based on such ratios (where normalizing constants cancel out). In particular, suppose that in the Metropolis algorithm (Algorithm 3) we consider a move from permutation $\sigma$ to a new permutation $\sigma'$. Probability of accepting the move to $\sigma'$ is given by Equation 7 (if $\frac{P(\sigma'|G, \Theta)}{P(\sigma|G, \Theta)} \leq 1$) or 1 otherwise.

Now we have to devise a way to sample permutations $\sigma$ from the proposal distribution. One way to do this would be to simply generate a random permutation $\sigma'$ and then check the acceptance condition. This would be very inefficient as we expect the distribution $P(\sigma|G, \Theta)$ to be heavily skewed, *i.e.*, there will be a relatively small number of good node mappings. Even more so as the degree distributions in real networks are skewed there will be many bad permutations with low likelihood, and few good ones that do a good job in matching nodes of high degree.

To make the sampling process “smoother”, *i.e.*, sample permutations that are not that different (and thus are not randomly jumping across the permutation space) we design a Markov chain. The idea is to stay in high likelihood part of permutation space longer. We do this by making samples dependent, *i.e.*, given $\sigma'$ we want to generate next candidate permutation $\sigma''$ to then evaluate the likelihood ratio. When designing the Markov chain step one has to be careful so that the proposal distribution satisfies the detailed balance condition: $\pi(\sigma')P(\sigma'|\sigma'') = \pi(\sigma'')P(\sigma'|\sigma')$, where $P(\sigma'|\sigma'')$ is the transition probability of obtaining permutation $\sigma'$ from $\sigma''$ and, $\pi(\sigma')$ is the stationary distribution. In our case we use a bit stronger condition of transition probabilities being symmetric which means that probability of a generating a candidate $\sigma''$ from $\sigma'$ has to be same as transition in the opposite way, $P(\sigma'|\sigma'') = P(\sigma''|\sigma')$.

In algorithm 3 we use a simple proposal where given permutation $\sigma'$ we generate $\sigma''$ by swapping elements at two uniformly at random chosen positions of $\sigma'$. We refer to this proposal as
SwapNodes. While this is simple and clearly satisfies the detailed balance condition it is also inefficient in a way that most of the times low degree nodes will get swapped (a direct consequence of heavy tailed degree distributions). This has two consequences, (a) we will slowly converge to good permutations (accurate mappings of high degree nodes), and (b) once we reach a good permutation, very few permutations will get accepted as most proposed permutations $\sigma'$ will swap low degree nodes (as they form the majority of nodes).

A possibly more efficient way would be to swap elements of $\sigma$ biased based on corresponding node degree. However, doing this directly does not satisfy the detailed balance condition. A way of sampling labels biased by node degrees that at the same time satisfies the detailed balance condition is the following: we pick an edge in $G$ uniformly at random and swap the labels of the endpoints. Notice this is biased towards swapping labels of nodes with high degrees simply as they have more edges. The detailed balance condition holds as edges are sampled uniformly at random. We refer to this proposal as SwapEdgeEndpoints.

However, the issue with this proposal is that if the graph $G$ is disconnected, we will only be swapping labels of nodes that belong to the same connected component. This means that some parts of the permutation space will never get visited. To overcome this problem we execute SwapNodes with some probability $\omega$ and SwapEdgeEndpoints with probability $1 - \omega$.

To summarize we consider the following two permutation proposal distributions:

- $\sigma'' = \text{SwapNodes}(\sigma')$: we obtain $\sigma''$ by taking $\sigma'$, uniformly at random selecting a pair of elements and swapping their positions.
- $\sigma'' = \text{SwapEdgeEndpoints}(\sigma')$: we obtain $\sigma''$ from $\sigma'$ by first sampling an edge $(j, k)$ from $G$ uniformly at random, then we take $\sigma'$ and swap the labels at positions $j$ and $k$.

5.3.2 SPEEDING UP THE LIKELIHOOD RATIO CALCULATION

We further speed up the algorithm by using the following observation. As written the equation 7 takes $O(N^2)$ to evaluate since we have to consider $N^2$ possible edges. However, notice that permutations $\sigma$ and $\sigma'$ differ only at two positions, i.e. elements at position $j$ and $k$ are swapped, i.e., $\sigma$ and $\sigma'$ map all nodes except the two to the same locations. This means those elements of equation 7 cancel out. Thus to update the likelihood we only need to traverse two rows and columns of matrix $P$, namely rows and columns $j$ and $k$, since everywhere else the mapping of the nodes to the adjacency matrix is the same for both permutations. This gives equation 8 where the products now range only over the two rows/columns of $P$ where $\sigma$ and $\sigma'$ differ.

Graphs we are working with here are too large to allow us to explicitly create and store the stochastic adjacency matrix $P$ by Kronecker powering the initiator matrix $\Theta$. Every time probability $P[i, j]$ of edge $(i, j)$ is needed the equation 2 is evaluated, which takes $O(k)$. So a single iteration of algorithm 3 takes $O(kN)$.

**Observation 3** Sampling a permutation $\sigma$ from $P(\sigma|G, \Theta)$ takes $O(kN)$.

This is gives us an improvement over the $O(N!)$ complexity of summing over all the permutations. So far we have shown how to obtain a permutation but we still need to evaluate the likelihood and find the gradients that will guide us in finding good initiator matrix. The problem here is that naively evaluating the network likelihood (gradient) as written in equation 6 takes time $O(N^3)$. This is exactly what we investigate next and how to calculate the likelihood in linear time.
5.4 Efficiently evaluating likelihood and gradient

We just showed how to efficiently sample node permutations. Now, given a permutation we show how to efficiently evaluate the likelihood and its gradient. Similarly as evaluating the likelihood ratio, naively calculating the log-likelihood $l(\Theta)$ or its gradient $\frac{\partial}{\partial \Theta} l(\Theta)$ takes time quadratic in the number of nodes. Next, we show how to compute this in linear time $O(E)$.

We begin with the observation that real graphs are sparse, which means that the number of edges is not quadratic but rather almost linear in the number of nodes, $E \ll N^2$. This means that majority of entries of graph adjacency matrix are zero, i.e., most of the edges are not present. We exploit this fact. The idea is to first calculate the likelihood (gradient) of an empty graph, i.e., a graph with zero edges, and then correct for the edges that actually appear in $G$.

To naively calculate the likelihood for an empty graph one needs to evaluate every cell of graph adjacency matrix. We consider Taylor approximation to the likelihood, and exploit the structure of matrix $P$ to devise a constant time algorithm.

First, consider the second order Taylor approximation to log-likelihood of an edge that succeeds with probability $x$ but does not appear in the graph:

$$\log(1 - x) \approx -x - \frac{1}{2} x^2$$

Calculating $l_e(\Theta)$, the log-likelihood of an empty graph, becomes:

$$l_e(\Theta) = \sum_{i=1}^{N} \sum_{j=1}^{N} \log(1 - p_{ij}) \approx - \left( \sum_{i=1}^{N_1} \sum_{j=1}^{N_1} \theta_{ij} \right)^k - \frac{1}{2} \left( \sum_{i=1}^{N_1} \sum_{j=1}^{N_1} \theta_{ij}^2 \right)^k$$  \hspace{1cm} (9)

Notice that while the first pair of sums ranges over $N$ elements, the last pair only ranges over $N_1$ elements ($N_1 = \log_k N$). Equation 9 holds due to the recursive structure of matrix $P$ generated by the Kronecker product. We substitute the $\log(1 - p_{ij})$ with its Taylor approximation, which gives a sum over elements of $P$ and their squares. Next, we notice the sum of elements of $P$ forms a multinomial series, and thus $\sum_{ij} p_{ij} = (\sum_{ij} \theta_{ij})^k$, where $\theta_{ij}$ denotes an element of $\Theta$, and $p_{ij}$ element of $\Theta^{[k]}$.

Calculating log-likelihood of $G$ now takes $O(E)$: First, we calculate the likelihood of an empty graph in constant time, and then account for the edges that are actually present in $G$, i.e., we subtract no-edge likelihood and add the edge likelihoods:

$$l(\Theta) = l_e(\Theta) + \sum_{(u,v)\in G} - \log(1 - P[\sigma_u, \sigma_v]) + \log(P[\sigma_u, \sigma_v])$$

5.5 Calculating the gradient

Calculation of the gradient of log-likelihood follows exactly the same pattern as described above. We first calculate gradient as if graph $G$ would have no edges. Then we correct the gradient for the edges that are present in $G$. As in previous section we speed up the calculations of the gradient by exploiting the fact that two consecutive permutations $\sigma$ and $\sigma'$ differ only at two positions, and thus given the gradient from previous step one only needs to account for the swap of the two rows and columns of the gradient matrix $\partial P / \partial \Theta$ to update to the gradients of individual parameters.
5.6 Determining the size of initiator matrix

The question we answer next is how to determine the right number of parameters, *i.e.*, what is the right size of $\Theta$ matrix? This is a classical question of model selection where there is a tradeoff between the complexity of the model, and the quality of the fit. Bigger model with more parameters usually fits better, however it is also more likely to overfit the data.

For model selection to find the appropriate value of $N_1$, the size of matrix $\Theta$, and choose the right tradeoff between the complexity of the model and the quality of the fit, we propose to use the Bayes Information Criterion (BIC) (Schwarz, 1978). Stochastic Kronecker Graphs model the presence of edges with independent Bernoulli random variables, where the canonical number of parameters is $N_1^{2k}$, which is a function of a lower-dimensional parameter $\Theta$. This is then a *curved exponential family* (Efron, 1975), and BIC naturally applies:

$$\text{BIC} = -l(\hat{\Theta}) + \frac{1}{2}N_1^2 \log(N^2)$$

where $\hat{\Theta}$ are maximum likelihood parameters under the model with $\hat{\Theta}$ of size $N_1 \times N_1$, and $N$ is the number of nodes in $G$. Note that one could also additional term to the above formula to account for multiple global maxima of the likelihood space but as $N_1$ is small the additional term would make no real difference.

Similarly, to BIC one could also consider the Minimum Description Length (MDL) (Rissanen, 1978) principle where the model is scored by the quality of the fit plus the size of the description that encodes the model and the parameters.

6. Experiments on real and synthetic data

We divide the experiments into several subsections. First we examine the convergence and mixing of the Markov chain of our permutation sampling scheme. Then we consider estimating the parameters of the synthetic Kronecker graphs to see whether KRONFit is able to recover the parameters used to generate the network. Last, we consider fitting Stochastic Kronecker Graph to large real world networks.

6.1 Permutation sampling

In our experiments we considered both synthetic and real graphs. Unless mentioned otherwise all synthetic Kronecker graphs were generated using $P_1^* = [0.8, 0.6; 0.5, 0.3]$, and $k = 14$ which gives us a graph $G$ on $N = 16,384$ nodes and $E = 115,741$ edges. We chose this particular $P_1^*$ as it closely resembles a typical initiator for real networks (that we show later).

6.1.1 Convergence of the log-likelihood and the gradient

First, we examine the convergence of Metropolis permutation sampling. As every next permutation is obtained from the previous one by locally modifying it this creates a Markov chain. We want to assess the convergence and mixing of the chain, *i.e.*, determine how many permutations one needs to draw to reliably estimate the likelihood and the gradient, and also how long does it take till the samples converge to the stationary distribution. For the experiment we generated a synthetic Stochastic Kronecker Graph using $P_1^*$ as defined above. Then, starting with a random permutation we run algorithm 3, and measure how the likelihood and the gradients converge to their true values.
In this particular case we first generated Stochastic Kronecker Graph $G$ as described above, but then calculated the likelihood and the parameter gradients for $\Theta' = [0.8, 0.75; 0.45, 0.3]$. We average the likelihoods and gradients over buckets of 1,000 consecutive samples, and plot how the log-likelihood calculated over the sampled permutations approaches the true log-likelihood (that we can compute since $G$ is a Stochastic Kronecker Graph).

First, we present experiments that aim to answer how many samples (i.e., permutations) does one need to draw to obtain a reliable estimate of the gradient (see Equation 6). Figure 12(a) shows how the estimated log-likelihood approaches the true likelihood. Notice that estimated values quickly converge to their true values, i.e., Metropolis sampling quickly moves towards “good” permutations. Similarly, Figure 12(b) plots the convergence of the gradients. Notice that $\theta_{11}$ and $\theta_{22}$ of $\Theta'$ and $P^*_1$ match, so gradients of these two parameters should converge to zero and indeed they do. On the other hand, $\theta_{12}$ and $\theta_{21}$ differ between $\Theta'$ and $P^*_1$. Notice, the gradient for one is positive as the parameter $\theta_{12}$ of $\Theta'$ should be decreased, and similarly for $\theta_{21}$ the gradient is negative as the parameter value should be increased to match the $\Theta'$. In summary, this shows that log-likelihood and gradients rather quickly converge to their true values.
Moreover, in Figures 12(c) and (d) we investigate the properties of the Markov Chain Monte Carlo sampling procedure, and assess convergence and mixing criteria. First, we plot the fraction of accepted proposals. It stabilizes at around 15%, which is quite close to the rule-of-a-thumb of 25%. Second, Figure 12(d) plots the autocorrelation of the log-likelihood as a function of the lag. Autocorrelation $r_k$ of a signal $X$ is a function of the lag $k$ where $r_k$ is defined as the correlation of signal $X$ at time $t$ with $X$ at $t + k$, i.e., correlation of the signal with itself at lag $k$. High autocorrelations within chains indicate slow mixing and, usually, slow convergence. On the other hand fast decay of autocorrelation means better the mixing and thus one needs less samples to accurately estimate the gradient or the likelihood. Notice rather fast autocorrelation decay.

All in all, these experiments show that one needs to sample an order of tens of thousands of permutations for the estimates to converge. We also verified that the variance of the estimates is sufficiently small. In our experiments we start with a random permutation and use long burn-in time. Then when performing optimization we use the permutation from previous step to initialize the permutation at current step of gradient descent. The intuition is that small changes in $P(\sigma|G, \Theta)$ also mean small changes in $\Theta$.

6.1.2 DIFFERENT PROPOSAL DISTRIBUTIONS

In section 5.3.1 we defined two permutation sampling proposal distributions: \texttt{SwapNodes} where we pick two nodes uniformly at random and swap their labels (node ids); and \texttt{SwapEdgeEndpoints} where we pick a random edge in a graph and then swap the labels of the edge endpoints. We also discussed that one can interpolate between the two strategies by executing \texttt{SwapNodes} with probability $\omega$, and \texttt{SwapEdgeEndpoints} with probability $1 - \omega$. 
So, given a Stochastic Kronecker Graph $G$ on $N = 16,384$ and $E = 115,741$ generated from $P_1^* = [0.8, 0.7; 0.5, 0.3]$ we evaluate the likelihood of $\Theta' = [0.8, 0.75; 0.45, 0.3]$. As we sample permutations we observe how the estimated likelihood converges to the true likelihood. Moreover we also vary parameter $\omega$ that interpolates between the two permutation proposal distributions. The quicker the converge towards the true log-likelihood the better the proposal distribution.

Figure 13 plots the convergence of the log-likelihood with the number of sampled permutations. We plot the average over non-overlapping buckets of 1,000 consecutive permutations. Faster convergence means better permutation proposal distribution. When we use only $\text{SwapNodes}$ ($\omega = 1$) or $\text{SwapEdgeEndpoints}$ ($\omega = 0$) convergence is rather slow. We obtain best convergence for $\omega$ around 0.6.

Similarly, Figure 14(a) plots the autocorrelation as a function of the lag $k$ for different choices of $\omega$. Faster autocorrelation decay means better mixing of the Markov chain. Again, notice that we get best mixing for $\omega = 0.6$. (Notice logarithmic y-axis.)

Last, we diagnose how long the sampling procedure must be run before the generated samples can be considered to be drawn (approximately) from the stationary distribution. We call this the burn-in time of the chain. There are various procedures for assessing convergence. Here we adopt the approach of Gelman et al. (Gelman et al., 2003), that is based on running multiple Markov chains each from a different starting point, and then comparing the variance within the chain and between the chains. The sooner the within- and between-chain variances become equal the faster the burn-in time, i.e., the sooner the samples are drawn from the stationary distribution.

Let $l$ be the parameter that is being simulated with $J$ different chains, and then let $l_j^{(k)}$ denote the $k^{th}$ sample of the $j^{th}$ chain, where $j = 1, \ldots, J$ and $k = 1, \ldots, K$. More specifically, in our case we run separate permutation sampling chains. So, we first sample permutation $\sigma_j^{(k)}$ and then calculate the corresponding log-likelihood $l_j^{(k)}$. 

Figure 14: (a) Autocorrelation plot of the log-likelihood for the different choices of parameter $\omega$. Notice we get best mixing with $\omega = 0.6$. (b) The potential scale reduction that compares the variance inside- and across- independent Markov chains for different values of parameter $\omega$.
First, we compute between and within chain variances \( \hat{\sigma}_B^2 \) and \( \hat{\sigma}_W^2 \), where between-chain variance is obtained by

\[
\hat{\sigma}_B^2 = \frac{K}{J-1} \sum_{j=1}^{J} (\bar{l}_j - \bar{l})^2
\]

where \( \bar{l}_j = \frac{1}{K} \sum_{k=1}^{K} l_{j}^{(k)} \) and \( \bar{l} = \frac{1}{J} \sum_{j=1}^{J} \bar{l}_j \).

Similarly the within-chain variance is defined by

\[
\hat{\sigma}_W^2 = \frac{1}{J(K-1)} \sum_{j=1}^{J} \sum_{k=1}^{K} (l_{j}^{(k)} - \bar{l}_j)^2
\]

Then, the marginal posterior variance of \( \hat{l} \) is calculated using

\[
\hat{\sigma}^2 = \frac{K-1}{K} \hat{\sigma}_W^2 + \frac{1}{K} \hat{\sigma}_B^2
\]

And, finally, we estimate the potential scale reduction (Gelman et al., 2003) of \( l \) by

\[
\sqrt{R} = \sqrt{\frac{\hat{\sigma}^2}{\hat{\sigma}_W^2}}
\]

Note that as the length of the chain \( K \to \infty \), \( \sqrt{R} \) converges to 1 from above. A recommendation for convergence assessment from (Gelman et al., 2003) is that potential scale reduction is below 1.2.

Figure 14(b) gives the Gelman-Rubin-Brooks plot, where we plot the potential scale reduction \( \sqrt{R} \) over the increasing chain length \( K \) for different choices of parameter \( \omega \). Notice that the potential scale reduction quickly decays towards 1. Similarly as in Figure 14 the extreme values of \( \omega \) give slow decay, while we obtain fastest potential scale reduction when \( \omega \approx 0.6 \).

### 6.1.3 Properties of the Permutation Space

Next we explore the properties of the permutation space. We would like to quantify what fraction of permutations are “good” (have high likelihood), and how quickly do we discover them. For the experiment we took a real network \( G \) (AS-ROUTEVIEWS network) and the MLE parameters \( \Theta \) for it that we estimated beforehand \( l(\hat{\Theta}) \approx -150,000 \). The network \( G \) has 6,474 nodes which means the space of all permutations has \( \approx 10^{22,000} \) elements.

First, we sampled 1 billion \( (10^9) \) permutations \( \sigma_i \) uniformly at random, i.e., \( P(\sigma_i) = 1/(6,474!) \) and for each evaluated its log-likelihood \( l(\sigma|\Theta_i) = \log P(\Theta_i|G, \sigma) \). We ordered the permutations in decreasing log-likelihood and plotted \( l(\sigma|\Theta_i) \) vs. rank. Figure 15(a) gives the plot. Notice that very few random permutations are very bad (i.e., they give low likelihood), similarly few permutations are very good, while most of them are somewhere in between. Notice that best “random” permutation has log-likelihood of \( \approx -320,000 \), which is far below true likelihood \( l(\hat{\Theta}) \approx -150,000 \). This suggests that only a very small fraction of all permutations gives good node labelings.

On the other hand, we also repeated the same experiment but now sampled permutations from the permutation distribution \( \sigma_i \sim P(\sigma|\Theta, G) \) using our Metropolis sampling scheme. Figure 15(b) gives the plot. Notice the radical difference. Now the \( l(\sigma|\Theta_i) \) very quickly converges to the true
Figure 15: (a) Distribution of log-likelihood of permutations sampled uniformly at random, and (b) when sampled from $P(\sigma|\Theta, G)$. Notice the space of good permutations is rather small but our sampling quickly finds permutations of high likelihood. (c) Convergence of log-likelihood for 10 runs of gradient descent, each from a different random starting point.

likelihood of $\approx -150,000$. This suggest that while the number of “good” permutations (accurate node mappings) is rather small, our sampling procedure quickly converges to the “good” part of the permutation space where node mappings are accurate.

### 6.2 Properties of the optimization space

In maximizing the likelihood we use stochastic approximation to the gradient. This adds variance to the gradient and makes efficient optimization techniques, like conjugate gradient, highly unstable. Thus we use gradient descent, which is slower but easier to control. First, we make the following observation:

**Observation 4** Given a real graph $G$ then finding the maximum likelihood Stochastic Kronecker initiator matrix $\hat{\Theta}$

$$\hat{\Theta} = \arg \max_{\Theta} P(G|\Theta)$$

is a non-convex optimization problem.

**Proof** By definition permutations of the Kronecker graphs initiator matrix $\hat{\Theta}$ all have the same log-likelihood. This means that we have several global minima that correspond to permutations of parameter matrix $\hat{\Theta}$, and then between them the log-likelihood drops. This means that the optimization problem is non-convex.

The above observation seem not to give much promise to estimating $\hat{\Theta}$ using gradient descent as it is prone to local minima. To check for the presence of other local minima where gradient descent could get stuck we run the following experiment: we generated 100 synthetic Kronecker graphs on 16,384 ($2^{14}$) nodes and 1.4 million edges on the average, with a randomly chosen $2 \times 2$ parameter matrix $\Theta^*$. For each of the 100 graphs we run gradient descent starting from a different random
parameter matrix $\Theta'$, and try to recover $\Theta^*$. In 98% of the cases the gradient descent converged to the true parameters. Many times the algorithm converged to a different global minima, i.e., $\hat{\Theta}$ is a permuted version of original parameter matrix $\Theta^*$. Moreover, the median number of gradient descent iterations was only 52.

This suggests surprisingly nice structure of our optimization space: it seems to behave like a convex optimization problem with many equivalent global minima. Moreover, this experiment is also a good sanity check as it shows that given a Kronecker graph we can recover and identify the parameters that were used to generate it.

Moreover, Figure 15(c) plots the log-likelihood $l(\Theta_t)$ of the current parameter estimate $\Theta_t$ over the iterations $t$ of the stochastic gradient descent. We plot the log-likelihood for 10 different runs of gradient descent, each time starting from a different random set of parameters $\Theta_0$. Notice that in all runs gradient descent always converges towards the optimum, and none of the runs gets stuck is some local maxima.
6.3 Convergence of the graph properties

We approached the problem of estimating Stochastic Kronecker initiator matrix $\Theta$ by defining the likelihood over the individual entries of the graph adjacency matrix. However, what we would really like is to be given a real graph $G$ and then generate a synthetic graph $K$ that has similar network properties as $G$. By properties we mean network statistics that can be computed from the graph, e.g., diameter, degree distribution, clustering coefficient, etc. A priori it is not clear that our approach which tries to match individual entries of graph adjacency matrix will also be able to reproduce these global network statistics. However, as show next this is not the case.

To get some understanding of the convergence of the gradient descent in terms of the network properties we performed the following experiment. After every step $t$ of stochastic gradient descent, we compare the true graph $G$ with the synthetic Kronecker graph $K_t$ generated using the current parameter estimates $\hat{\Theta}_t$. Figure 16(a) gives the convergence of log-likelihood, and (b) gives absolute error in parameter values ($\sum |\hat{\theta}_{ij} - \theta_{ij}^*|$, where $\hat{\theta}_{ij} \in \hat{\Theta}_t$, and $\theta_{ij}^* \in \Theta^*$). Similarly, Figure 16(c) plots the effective diameter, and (d) gives the largest singular value of graph adjacency matrix $K$ as it converges to largest singular value of $G$.

Note how with progressing iterations of gradient descent properties of graph $K_t$ quickly converge to those of $G$ even though we are not directly optimizing the similarity in network properties: log-likelihood increases, absolute error of parameters decreases, diameter and largest singular value of $K_t$ both converge to $G$. This is a nice result as it shows that through maximizing the likelihood the resulting graphs become more and more similar also in their structural properties (even though we are not directly optimizing over them).

6.4 Fitting to real-world networks

Next, we present experiments of fitting Kronecker Graphs model to real-world networks. Given a real network $G$ we aim to discover the most likely parameters $\hat{\Theta}$ that ideally would generate a synthetic graph $K$ having similar properties as real $G$. This assumes that Kronecker Graphs is a good model of the network structure, and that KRONFit is able to find good parameters. In previous section we showed that KRONFit can efficiently recover the parameters. Now we examine how well can Kronecker graphs model the structure of real networks.

We consider several different networks, like a graph of connectivity among Internet Autonomous Systems (AS-ROUTEVIEWS) with $N = 6,474$ and $E = 26,467$; a who-trusts-whom type social network from Epinions (Richardson et al., 2003) (EPINIONS) with $N = 75,879$ and $E = 508,960$ and many others. The largest network we consider for fitting is FLICKR photo-sharing online social network with 584,207 nodes and 3,555,115 edges.

For the purpose of this section we take a real network $G$, find parameters $\hat{\Theta}$ using KRONFit, generate a synthetic graph $K$ using $\hat{\Theta}$, and then compare $G$ and $K$ by comparing their properties that we introduced in section 2. In all experiments we started from a random point (random initiator matrix) and run gradient descent for 100 steps. At each step we estimate the likelihood and the gradient based on 510,000 sampled permutations where we discard first 10,000 samples to allow the chain to burn-in.

6.4.1 Fitting to Autonomous Systems network

First, we focus on the Autonomous Systems network obtained from the University of Oregon Route Views project (RouteViews, 1997). Given the AS network $G$ we run KRONFit to obtain parameter
estimates $\hat{\Theta}$. Using the $\hat{\Theta}$ we then generate a synthetic Kronecker graph $K$, and compare the properties of $G$ and $K$.

Figure 17 shows properties of AS-ROUTEVIEWS, and compares them with the properties of a synthetic Kronecker graph generated using the fitted parameters $\hat{\Theta}$ of size $2 \times 2$. Notice that properties of both graphs match really well. The estimated parameters are $\hat{\Theta} = [0.987, 0.571; 0.571, 0.049]$.

Figure 17(a) compares the degree distributions of the AS-ROUTEVIEWS network and its synthetic Kronecker estimate. In this and all other plots we use the exponential binning which is a standard procedure the de-noise the data when plotting on log–log scales. Notice a very close match in degree distribution between the real graph and its synthetic counterpart.

Figure 17(b) plots the cumulative number of pairs of nodes $g(h)$ that can be reached in $\leq h$ hops. The hop plot gives a sense about the distribution of the shortest path lengths in the network and about the network diameter. Last, Figures 17(c) and (d) plot the spectral properties of the graph adjacency matrix. Figure 17(c) plots largest singular values vs. rank, and (d) plots the components of left singular vector (the network value) vs. the rank. Again notice good agreement with the real graph while using only four parameters.
Table 2: Log-likelihood at MLE for different choices of the size of the initiator matrix $N_1$ for the AS-ROUTEVIEWS graph. Notice the log-likelihood $l(\hat{\theta})$ generally increases with the model complexity $N_1$. Also notice the effect of zero-padding, *i.e.*, for $N_1 = 4$ and $N_1 = 5$ the constraint of the number of nodes being an integer power of $N_1$ decreases the log-likelihood. However, the column $|\{\deg(u) > 0\}|$ gives the number of non-isolated nodes in the network which is much less than $N_1^k$ and is in fact very close to the true number of nodes in the AS-ROUTEVIEWS. Using the BIC scores we see that $N_1 = 3$ or $N_1 = 6$ are best choices for the size of the initiator matrix.

| $N_1$ | $l(\hat{\theta})$ | $N_1^k$ | $E_1^k$ | $|\{\deg(u) > 0\}|$ | BIC score |
|-------|------------------|---------|---------|-----------------|-----------|
| 2     | −152,499         | 8,192   | 25,023  | 5,675           | 152,506   |
| 3     | −127,066         | 6,561   | 28,790  | 5,683           | 127,083   |
| 4     | −153,260         | 16,384  | 24,925  | 8,222           | 153,290   |
| 5     | −149,949         | 15,625  | 29,111  | 9,822           | 149,996   |
| 6     | −128,241         | 7,776   | 26,557  | 6,623           | 128,309   |

Moreover, on all plots the error bars of two standard deviations show the variance of the graph properties for different realizations $R(\hat{\Theta}^{[k]})$. To obtain the error bars we took the same $\Theta$, and generated 50 realizations of a Kronecker graph. As for the most of the plots the error bars are so small to be practically invisible, this shows that the variance of network properties when generating a Stochastic Kronecker graph is indeed very small.

Also notice that the AS-ROUTEVIEWS is an undirected graph, and that the fitted parameter matrix $\hat{\Theta}$ is in fact symmetric. This means that without a priori biasing the fitting towards undirected graphs, the recovered parameters obey this aspect of the network. Fitting AS-ROUTEVIEWS graph from a random set of parameters, performing gradient descent for 100 iterations and at each iteration sampling half a million permutations, took less than 10 minutes on a standard desktop PC. This is a significant speedup over (Bezáková et al., 2006), where by using a similar permutation sampling approach for calculating the likelihood of a preferential attachment model on similar AS-ROUTEVIEWS graph took about two days on a cluster of 50 machines.

6.4.2 Choice of the Initiator Matrix Size $N_1$

As mentioned earlier for finding the optimal number of parameters, *i.e.*, selecting the size of initiator matrix, BIC criterion naturally applies to the case of Kronecker Graphs. Figure 23(b) shows BIC scores for the following experiment: We generated Kronecker graph with $N = 2,187$ and $E = 8,736$ using $N_1 = 3$ (9 parameters) and $k = 7$. For $1 \leq N_1 \leq 9$ we find the MLE parameters using gradient descent, and calculate the BIC scores. Model with the lowest score is chosen. As figure 23(b) shows we recovered the true model, *i.e.*, BIC score is the lowest for the model with the true number of parameters, $N_1 = 3$.

Intuitively we expect a more complex model with more parameters to fit the data better. Thus we expect larger $N_1$ to generally give better likelihood. On the other hand the fit will also depend on the size of the graph $G$. Kronecker graphs can only generate graphs on $N_1^k$ nodes, while real graphs
do not necessarily have \( N_1^k \) nodes (for some, preferably small, integers \( N_1 \) and \( k \)). To solve this problem we choose \( k \) so that \( N_1^{k-1} < N(G) \leq N_1^k \), and then augment \( G \) by adding \( N_1^k - N \) isolated nodes. Or equivalently, we pad the adjacency matrix of \( G \) with zeros until it is of the appropriate size, \( N_1^k \times N_1^k \). While this solves the problem of requiring the integer power of the number of nodes it also makes the fitting problem harder as when \( N \ll N_1^k \) we are basically fitting \( G \) plus a large number of isolated nodes.

Table 2 shows the results of fitting Kronecker graphs to \textsc{As-RouteViews} while varying the size of the initiator matrix \( N_1 \). First, notice that in general larger \( N_1 \) results in higher log-likelihood \( \ell(\hat{\Theta}) \) at MLE. Similarly, notice (column \( N_1^{k_1} \)) that while \textsc{As-RouteViews} has 6,474 nodes, Kronecker estimates have up to 16,384 nodes (16,384 = 4^7, which is the first integer power of 4 greater than 6,474). However, we also show the number of non-zero degree (non-isolated) nodes in the Kronecker graph (column \( |\{\deg(u) > 0\}| \)). Notice that the number of non-isolated nodes well corresponds to the number of nodes in \textsc{As-RouteViews} network. This shows that \textsc{KronFit} is actually fitting the graph well, it successfully fits the structure of the graph plus a number of isolated nodes. Last, column \( E_1^k \) gives the number of edges in the corresponding Kronecker graph which is close to the true number of edges of the \textsc{As-RouteViews} graph.

Last, comparing the log-likelihood at MLE and the BIC score in Table 2 we notice that the log-likelihood heavily dominates the BIC score. This means that the size of the initiator matrix (number of parameters) is so small that one does not really have to care about overfitting. Thus we can just choose the initiator matrix that maximizes the likelihood. A simple calculation shows that one would need to take initiator matrices with thousands of entries before the model complexity part of BIC score would start to play a significant role.

We further examine the sensitivity of the choice of the initiator size by the following experiment. We generate a Stochastic Kronecker Graph \( K \) on 9 parameters (\( N_1 = 3 \)), and then fit a Kronecker graph \( K' \) with a smaller number of parameters (4 instead of 9, \( N_1' = 2 \)). And also a Kronecker graph \( K'' \) of the same complexity as \( K \) (\( N_1'' = 3 \)).

Figure 18 plots the properties of all three graphs. Not surprisingly \( K'' \) (blue) fits the properties of \( K \) (red) perfectly as the initiator is of the same size. On the other hand \( K' \) (green) is a simpler model with only 4 parameters (instead of 9 as in \( K \) and \( K'' \)) and still generally fits well: hop plot and degree distribution match well, while spectral properties of graph adjacency matrix, especially scree plot, are not matched that well. This shows that nothing drastic happens and that even a bit too simple model still fits the data well. In general we observe empirically that by increasing the size of initiator matrix one does not gain radically better fits for degree distribution and hop plot. On the other hand there is usually an improvement in the scree plot and the plot of network values when one increases the initiator size.

### 6.4.3 Network Parameters over Time

Next we briefly examine the evolution of the Kronecker initiator for a temporally evolving graph. The idea is that given parameter estimates of a real-graph \( G_t \) at time \( t \), we can forecast the future structure of the graph \( G_{t+x} \) at time \( t + x \), i.e., using parameters obtained from \( G_t \) we can generate a larger synthetic graph \( K \) that will be similar to \( G_{t+x} \).

As we have the information about the evolution of the \textsc{As-RouteViews} network, we estimated parameters for three snapshots of the network when it had about \( 2^k \) nodes. Table 3 gives the results of the fitting for the three temporal snapshots of the \textsc{As-RouteViews} network. Notice the
Figure 18: 3 by 3 Stochastic Kronecker Graph: Given a Stochastic Kronecker Graph $G$ generated from $N_1 = 3$ (red curve), we fit a Kronecker graph $K'$ with $N'_1 = 2$ (green) and $K''$ with $N''_1 = 3$ (blue). Not surprisingly $K''$ fits the properties of $K$ perfectly as the model is of the same complexity. On the other hand $K'$ has only 4 parameters (instead of 9 as in $K$ and $K''$) and still fits well.

<table>
<thead>
<tr>
<th>Snapshot at time</th>
<th>$N$</th>
<th>$E$</th>
<th>$l(\hat{\Theta})$</th>
<th>Estimates at MLE, $\hat{\Theta}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>2,048</td>
<td>8,794</td>
<td>-40,535</td>
<td>[0.981, 0.633; 0.633, 0.048]</td>
</tr>
<tr>
<td>$T_2$</td>
<td>4,088</td>
<td>15,711</td>
<td>-82,675</td>
<td>[0.934, 0.623; 0.622, 0.044]</td>
</tr>
<tr>
<td>$T_3$</td>
<td>6,474</td>
<td>26,467</td>
<td>-152,499</td>
<td>[0.987, 0.571; 0.571, 0.049]</td>
</tr>
</tbody>
</table>

Table 3: Parameter estimates of the three temporal snapshots of the AS-ROUTEVIEWS network. Notice that estimates stay remarkably stable over time.

parameter estimates $\hat{\Theta}$ remain remarkably stable over time. This means that Kronecker graphs can be used to estimate the structure of the networks in the future, i.e., parameters estimated from the historic data can extrapolate the graph structure in the future.

Figure 19 further explores this. It overlays the graph properties of the real AS-ROUTEVIEWS network at time $T_3$ and the synthetic graphs for which we used the parameters obtained on historic
Figure 19: Autonomous systems network over time (AS-ROUTEVIEWS): Overlayed patterns of real AS-ROUTEVIEWS network at time $T_3$ and the Kronecker graphs with parameters estimated from AS-ROUTEVIEWS at time $T_1$ and $T_2$. Notice good fits which means that parameters estimated on historic snapshots can be used to estimate the graph in the future.

Snapshots of AS-ROUTEVIEWS at times $T_1$ and $T_2$. The agreements are good which demonstrates that Kronecker graphs can forecast the structure of the network in the future.

Moreover, this experiments also shows that parameter estimates do not suffer much from the zero padding of graph adjacency matrix (i.e., adding isolated nodes to make $G$ have $N^k_1$ nodes). Snapshots of AS-ROUTEVIEWS at $T_1$ and $T_2$ have close to $2^k$ nodes, while we had to add 26% (1,718) isolated nodes to the network at $T_3$ to make the number of nodes be $2^k$. Regardless of this we see the parameter estimates $\hat{\Theta}$ remain basically constant over time, which seems to be independent of the number of isolated nodes added. This means that the estimated parameters are not biased too much from zero padding the adjacency matrix of $G$.

### 6.5 Fitting to other large real-world networks

Last, we present results of fitting Stochastic Kronecker Graph to 20 large real-world networks: large online social networks, like EPINIONS, FLICKR and DELICIOUS, web and blog graphs (WEB-NOTREDAME, BLOG-NAT05-6M, BLOG-NAT06ALL), internet and peer-to-peer networks (AS-NEWMAN,
Kronecker Graphs

<table>
<thead>
<tr>
<th>Network</th>
<th>N</th>
<th>E</th>
<th>Estimated parameters $\Theta$</th>
<th>$l(\Theta)$</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS-ROUTE VIEWS</td>
<td>6,474</td>
<td>26,467</td>
<td>0.987, 0.571; 0.571, 0.049</td>
<td>−152,499</td>
<td>8m15s</td>
</tr>
<tr>
<td>ATP-GR-QC</td>
<td>19,177</td>
<td>26,169</td>
<td>0.902, 0.253; 0.221, 0.582</td>
<td>−242,493</td>
<td>7m40s</td>
</tr>
<tr>
<td>BIO-PROTEINS</td>
<td>4,626</td>
<td>29,602</td>
<td>0.847, 0.641; 0.641, 0.072</td>
<td>−185,130</td>
<td>43m41s</td>
</tr>
<tr>
<td>EMAIL-INSIDE</td>
<td>986</td>
<td>32,128</td>
<td>0.999, 0.772; 0.772, 0.257</td>
<td>−107,283</td>
<td>1h07m</td>
</tr>
<tr>
<td>CA-GR-QC</td>
<td>5,242</td>
<td>28,980</td>
<td>0.999, 0.245; 0.245, 0.691</td>
<td>−160,902</td>
<td>14m02s</td>
</tr>
<tr>
<td>AS-NEWMAN</td>
<td>22,963</td>
<td>96,872</td>
<td>0.954, 0.594; 0.594, 0.019</td>
<td>−593,747</td>
<td>28m48s</td>
</tr>
<tr>
<td>BLOG-NAT05-6M</td>
<td>31,600</td>
<td>271,377</td>
<td>0.999, 0.569; 0.502, 0.221</td>
<td>−1,994,943</td>
<td>47m20s</td>
</tr>
<tr>
<td>BLOG-NAT06ALL</td>
<td>32,443</td>
<td>318,815</td>
<td>0.999, 0.578; 0.517, 0.221</td>
<td>−2,289,009</td>
<td>52m31s</td>
</tr>
<tr>
<td>CA-HEP-PH</td>
<td>12,008</td>
<td>237,010</td>
<td>0.999, 0.437; 0.437, 0.484</td>
<td>−1,272,629</td>
<td>1h22m</td>
</tr>
<tr>
<td>CA-HEP-TH</td>
<td>9,877</td>
<td>51,971</td>
<td>0.999, 0.271; 0.271, 0.587</td>
<td>−343,614</td>
<td>21m17s</td>
</tr>
<tr>
<td>CIT-HEP-PH</td>
<td>30,567</td>
<td>348,721</td>
<td>0.994, 0.439; 0.355, 0.526</td>
<td>−2,607,159</td>
<td>51m26s</td>
</tr>
<tr>
<td>CIT-HEP-TH</td>
<td>27,770</td>
<td>352,807</td>
<td>0.990, 0.440; 0.347, 0.538</td>
<td>−2,507,167</td>
<td>15m23s</td>
</tr>
<tr>
<td>EPINIONS</td>
<td>75,879</td>
<td>508,837</td>
<td>0.999, 0.532; 0.480, 0.129</td>
<td>−3,817,121</td>
<td>45m39s</td>
</tr>
<tr>
<td>GNUTELLA-25</td>
<td>22,687</td>
<td>54,705</td>
<td>0.746, 0.496; 0.654, 0.183</td>
<td>−530,199</td>
<td>16m22s</td>
</tr>
<tr>
<td>GNUTELLA-30</td>
<td>36,682</td>
<td>88,328</td>
<td>0.753, 0.489; 0.632, 0.178</td>
<td>−919,235</td>
<td>14m20s</td>
</tr>
<tr>
<td>DELICIOUS</td>
<td>205,282</td>
<td>436,735</td>
<td>0.999, 0.327; 0.348, 0.391</td>
<td>−4,579,001</td>
<td>27m51s</td>
</tr>
<tr>
<td>ANSWERS</td>
<td>598,314</td>
<td>1,834,200</td>
<td>0.994, 0.384; 0.414, 0.249</td>
<td>−20,508,982</td>
<td>2h35m</td>
</tr>
<tr>
<td>CA-DBLP</td>
<td>425,957</td>
<td>2,696,489</td>
<td>0.999, 0.307; 0.307, 0.574</td>
<td>−26,813,878</td>
<td>3h01m</td>
</tr>
<tr>
<td>FLICKR</td>
<td>584,207</td>
<td>3,555,115</td>
<td>0.999, 0.474; 0.485, 0.144</td>
<td>−32,043,787</td>
<td>4h26m</td>
</tr>
<tr>
<td>WEB-NOTREDAME</td>
<td>325,729</td>
<td>1,497,134</td>
<td>0.999, 0.414; 0.453, 0.229</td>
<td>−14,588,217</td>
<td>0h59m</td>
</tr>
</tbody>
</table>

Table 4: Results of parameter estimation for 20 different networks. Table 5 gives the description and basic properties of the above network datasets.

GNUTELLA-25, GNUTELLA-30, collaboration networks of co-authorships from DBLP (CA-DBLP) and various areas of physics (CA-HEP-TH, CA-HEP-PH, CA-GR-QC), physics citation networks (CIT-HEP-PH, CIT-HEP-TH), an email network (EMAIL-INSIDE), a protein interaction network BIO-PROTEINS, and a bipartite affiliation network (authors-to-papers, ATP-GR-QC). Refer to table 5 in the appendix for the description and basic properties of these networks.

For each dataset we started gradient descent from a random point (random initiator matrix) and run it for 100 steps. At each step we estimate the likelihood and the gradient based on 510,000 sampled permutations where we discard first 10,000 samples to allow the chain to burn-in.

Table 4 gives the estimated parameters, the corresponding log-likelihoods and the wall clock times. All experiments were carried out on standard desktop computer. Notice that the estimated initiator matrix $\Theta$ seems to have almost universal structure with a big value in the top left entry, a very low value at the bottom right corner and intermediate values in the other two corners. We further discuss the implications of such structure of Kronecker initiator matrix on the global network structure in the next section.

Last, Figures 20 and 21 show overlays of various network properties of real and the estimated synthetic networks. In addition to the network properties we plotted in Figure 18, we also separately plot in- and out-degree distributions (as both networks are directed) and plot the node triangle participation in panel (c), where we plot the number of triangles a node participates in versus the number of such nodes. (Again the error bars show the variance of network properties over different realizations $R(\hat{\Theta}[k])$ of a Stochastic Kronecker graph.)
Figure 20: Blog network (BLOG-NAT06ALL): Overlayed patterns of real network and the estimated Kronecker graph using 4 parameters (2 × 2 initiator matrix). Notice that the Kronecker graph matches all properties of the real network.

Notice that for both networks and in all cases the properties of the real network and the synthetic Kronecker coincide really well. Using Stochastic Kronecker Graph with just 4 parameters we match the scree plot, degree distributions, triangle participation, hop plot and network values.

Given the experience from the Autonomous systems we only present the results for the simplest model with initiator size $N_1 = 2$. Empirically we also observe that $N_1 = 2$ gives surprisingly good fits and the estimation procedure is the most robust and converges the fastest. Using larger initiator matrices $N_1 > 2$ generally helps improve the likelihood but not dramatically. In terms of matching the network properties we also get a slight improvement by making the model more complex. Figure 22 gives the percent improvement in log-likelihood as we make the model more complex. We use the log-likelihood of a $2 \times 2$ model as a baseline and estimate the log-likelihood at MLE for larger initiator matrices. Again, models with more parameters tend to fit better. However, sometimes due to zero-padding of graph adjacency matrix they actually have lower log-likelihood.

6.6 Scalability

Last we also empirically evaluate the scalability of the KRONFIT. The experiment confirms that KRONFIT runtime scales linearly with the number of edges $E$ in a graph $G$. More precisely, we performed the following experiment.

We generated a sequence of increasingly larger synthetic graphs on $N$ nodes and $8N$ edges, and measured the time of one iteration of gradient descent, i.e., sample 1 million permutations and evaluate the gradients. We started with a graph on 1,000 nodes, and finished with a graph on 8 million nodes, and 64 million edges. Figure 23(a) shows KRONFIT scales linearly with the size of
Figure 21: Epinions who-trusts-whom social network: Overlayed patterns of real network and the fitted Kronecker graph using only 4 parameters ($2 \times 2$ initiator matrix). Again, the synthetic Kronecker graph matches all the properties of the real network.

7. Discussion

Here we discuss several of the desirable properties of the proposed Kronecker Graphs.

**Generality:** Stochastic Kronecker Graphs include several other generators as special cases: For $\theta_{ij} = c$, we obtain classical Erdős-Rényi random graph model; for $\theta_{i,j} \in \{0, 1\}$, we obtain a deterministic Kronecker graph; setting the $K_1$ matrix to a $2 \times 2$ matrix, we obtain the R-MAT generator (Chakrabarti et al., 2004). In contrast to Kronecker graphs, the R-MAT cannot extrapolate into the future, since it needs to know the number of edges to insert. Thus, it is incapable of obeying the densification power law.

**Phase transition phenomena:** The Erdős-Rényi graphs exhibit phase transitions (Erdős and Rényi, 1960). Several researchers argue that real systems are “at the edge of chaos” (Bak, 1996; Sole and Goodwin, 2000). Stochastic Kronecker Graphs also exhibit phase transitions (Mahdian and Xu, 2007) for the emergence of the giant component and another phase transition for connectivity.

**Implications to the structure of the large-real networks:** Empirically we found that $2 \times 2$ initiator ($N_1 = 2$) fits well the properties of real-world networks. Moreover, given a $2 \times 2$ initiator matrix, one can look at it as a recursive expansion of two groups into sub-groups. We introduced this recursive view of Kronecker graphs back in section 3. So, one can then interpret the diagonal values of $\Theta$ as the proportion of edges inside each of the groups, and the off-diagonal values give the fraction of edges connecting the groups. Figure 24 illustrates the setting for two groups.
Figure 22: Percent improvement in log-likelihood over the $2 \times 2$ model as we increase the model complexity (size of initiator matrix). In general larger initiator matrices that have more degrees of freedom help improving the fit of the model.

Figure 23: (a) Processor time to sample 1 million gradients as the graph grows. Notice the algorithm scales linearly with the graph size. (b) BIC score for model selection.

For example, as shown in Figure 24, large $a, d$ and small $b, c$ would imply that the network is composed of hierarchically nested communities, where there are many edges inside each community.
Figure 24: 2 × 2 Kronecker initiator matrix (a) can be thought of as two communities where there are $a$ and $d$ edges inside each of the communities and $b$ and $c$ edges crossing the communities as illustrated in (b). The each sub-community can then be recursively divided using the same pattern. (c) The onion like core-periphery structure where the network gets denser and denser as we move towards the center of the network.

and few edges crossing them. One could think of this structure as some kind of organizational or university hierarchy, where one expects the most friendships between people within same lab, a bit less between people in the same department, less across different departments, and the least friendships to be formed across people from different schools of the university.

However, parameter estimates for a wide range of networks presented in Table 4 suggests a very different picture of the network structure. Notice that for most networks $a \gg b > c \gg d$. Moreover, $a \approx 1$, $b \approx c \approx 0.6$ and $d \approx 0.2$. We empirically observed that the same structure of initiator matrix $\hat{\Theta}$ also holds when fitting $3 \times 3$ or $4 \times 4$ models. Always the top left element is the largest and then the values on the diagonal decay faster than off the diagonal.

This suggests a network structure which is also known as core-periphery (Borgatti and Everett, 2000; Holme, 2005), the jellyfish (Tauro et al., 2001; Siganos et al., 2006), or the octopus (Chung and Lu, 2006) structure of the network as illustrated in Figure 24(c).

All of the above basically say that the network is composed of a densely linked network core and the periphery. In our case this would imply the following structure of the initiator matrix. The core is modeled by parameter $a$ and the periphery by $d$. Most edges are inside the core (large $a$), and very few between the nodes of periphery (small $d$). Then there are many more edges between the core and the periphery than inside the periphery ($b, c > d$). This is exactly what we see as well. And in spirit of Kronecker graphs the structure repeats recursively — the core again has the dense core and the periphery, and so on. And similarly the periphery itself has the core and the periphery.

This suggest an “onion” like network structure as illustrated in Figure 24(c), where the network is composed of denser and denser layers as one moves towards the center of the network. We also observe similar structure of the Kronecker initiator when fitting $3 \times 3$ or $4 \times 4$ initiator matrix. The diagonal elements have large but decreasing values with off diagonal elements following same decreasing pattern.

One of the implications of this is that networks do not break nicely into hierarchically organized sets of communities that nicely allow themselves to partitioning and community identification algorithms. On contrary, this suggests that large networks can be decomposed into a densely linked core with many small periphery pieces hanging off the core. This is in accordance with our recent
results (Leskovec et al., 2008a,b), that make similar observation (but based on a completely different methodology) about the structure of large real-world networks. Refer to (Leskovec et al., 2008a) and (Leskovec et al., 2008b) for more details on large scale community structure of networks.

8. Conclusion

In conclusion, the main contribution of this work is a family of models of network structure that uses a non-traditional matrix operation, the Kronecker product. The resulting graphs (a) have all the static properties (heavy-tailed degree distribution, small diameter, etc.), (b) all the temporal properties (densification, shrinking diameter) that are found in real networks. And in addition, (c) we can formally prove all of these properties.

Several of the proofs are extremely simple, thanks to the rich theory of Kronecker multiplication. We also provide proofs about the diameter and effective diameter, and we show that Stochastic Kronecker Graphs can mimic real graphs well.

Moreover, we also presented KRONFIT, a fast, scalable algorithm to estimate Stochastic Kronecker initiator, which can be then used to create a synthetic graph that mimics the properties of a given real network.

In contrast to earlier work, our work has the following novelties: (a) it is among the few that estimates the parameters of the chosen generator in a principled way, (b) it is among the few that has a concrete measure of goodness of the fit (namely, the likelihood), (c) it avoids the quadratic complexity of computing the likelihood by exploiting the properties of the Kronecker graphs, and (d) it avoids the factorial explosion of the node correspondence problem, by using the Metropolis sampling.

The resulting algorithm matches well all the known properties of real graphs, as we show with the Epinions graph and the AS graph, it scales linearly on the number of edges, and it is orders of magnitudes faster than earlier graph-fitting attempts: 20 minutes on a commodity PC, versus 2 days on a cluster of 50 workstations (Bezáková et al., 2006).

The benefits of fitting a Kronecker graph model into a real graph are several:

- **Extrapolation**: Once we have the Kronecker generator $\Theta$ for a given real matrix $G$ (such that $G$ is mimicked by $\Theta[k]$), a larger version of $G$ can be generated by $\Theta[k+1]$.

- **Null-model**: When analyzing a real network $G$ one often needs to assess the significance of the observation. $\Theta[k]$ that mimics $G$ can be used as an accurate model of $G$.

- **Network structure**: fitted parameters give insight into the global network and community structure of the network.

- **Forecasting**: As we demonstrated one can obtain $\Theta$ from a graph $G_t$ at time $t$ such that $G$ is mimicked by $\Theta[k]$. Then $\Theta$ can be used to model the structure of $G_{t+x}$ in the future.

- **Sampling**: Similarly, if we want a realistic sample of the real graph, we could use a smaller exponent in the Kronecker exponentiation, like $\Theta[k-1]$.

- **Anonymization**: Since $\Theta[k]$ mimics $G$, we can publish $\Theta[k]$, without revealing information about the nodes of the real graph $G$. 
References


Kronecker Graphs


**Appendix A. Table of networks**

Table 5 lists all the network datasets that were used in this paper. We also computed some of the structural network properties.
<table>
<thead>
<tr>
<th>Network</th>
<th>$N$</th>
<th>$E$</th>
<th>$N_c$</th>
<th>$N_c/N$</th>
<th>$C$</th>
<th>$D$</th>
<th>$\bar{D}$</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Social networks</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANSWERS</td>
<td>598,314</td>
<td>1,834,200</td>
<td>488,484</td>
<td>0.82</td>
<td>0.11</td>
<td>22</td>
<td>5.72</td>
<td>Yahoo! Answers social network (Leskovec et al., 2008a)</td>
</tr>
<tr>
<td>DELICIOUS</td>
<td>205,282</td>
<td>436,735</td>
<td>147,567</td>
<td>0.72</td>
<td>0.3</td>
<td>24</td>
<td>6.28</td>
<td>del.icio.us social network (Leskovec et al., 2008a)</td>
</tr>
<tr>
<td>EMAIL-INSIDE</td>
<td>986</td>
<td>32,128</td>
<td>986</td>
<td>1.00</td>
<td>0.45</td>
<td>7</td>
<td>2.6</td>
<td>European research organization email network (Leskovec et al., 2007a)</td>
</tr>
<tr>
<td>EPINIONS</td>
<td>75,879</td>
<td>508,837</td>
<td>75,877</td>
<td>1.00</td>
<td>0.26</td>
<td>15</td>
<td>4.27</td>
<td>Who-trusts-whom graph of epinions.com (Richardson et al., 2003)</td>
</tr>
<tr>
<td>FLICKR</td>
<td>584,207</td>
<td>3,555,115</td>
<td>404,733</td>
<td>0.69</td>
<td>0.4</td>
<td>18</td>
<td>5.42</td>
<td>Flickr photo sharing social network (Kumar et al., 2006)</td>
</tr>
<tr>
<td><strong>Information (citation) networks</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOG-NAT05-6M</td>
<td>31,600</td>
<td>271,377</td>
<td>29,150</td>
<td>0.92</td>
<td>0.24</td>
<td>10</td>
<td>3.4</td>
<td>Blog-to-blog citation network (6 months of data) (Leskovec et al., 2007b)</td>
</tr>
<tr>
<td>BLOG-NAT06ALL</td>
<td>32,443</td>
<td>318,815</td>
<td>32,384</td>
<td>1.00</td>
<td>0.2</td>
<td>18</td>
<td>3.94</td>
<td>Blog-to-blog citation network (1 year of data) (Leskovec et al., 2007b)</td>
</tr>
<tr>
<td>CIT-HEP-PH</td>
<td>30,567</td>
<td>348,721</td>
<td>34,401</td>
<td>1.13</td>
<td>0.3</td>
<td>14</td>
<td>4.33</td>
<td>Citation network of ArXiv hep-th papers (Gehrke et al., 2003)</td>
</tr>
<tr>
<td>CIT-HEP-TTH</td>
<td>27,770</td>
<td>352,807</td>
<td>27,400</td>
<td>0.99</td>
<td>0.33</td>
<td>15</td>
<td>4.2</td>
<td>Citations network of ArXiv hep-ph papers (Gehrke et al., 2003)</td>
</tr>
<tr>
<td><strong>Collaboration networks</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA-DLBP</td>
<td>425,957</td>
<td>2,696,489</td>
<td>317,080</td>
<td>0.74</td>
<td>0.73</td>
<td>23</td>
<td>6.75</td>
<td>DBLP co-authorship network (Backstrom et al., 2006)</td>
</tr>
<tr>
<td>CA-GR-QC</td>
<td>5,242</td>
<td>28,980</td>
<td>4,158</td>
<td>0.79</td>
<td>0.66</td>
<td>17</td>
<td>6.1</td>
<td>Co-authorship network in gr-qc category of ArXiv (Leskovec et al., 2005b)</td>
</tr>
<tr>
<td>CA-HEP-PH</td>
<td>12,008</td>
<td>237,010</td>
<td>11,204</td>
<td>0.93</td>
<td>0.69</td>
<td>13</td>
<td>4.71</td>
<td>Co-authorship network in hep-ph category of ArXiv (Leskovec et al., 2005b)</td>
</tr>
<tr>
<td>CA-HEP-TTH</td>
<td>9,877</td>
<td>51,971</td>
<td>8,638</td>
<td>0.87</td>
<td>0.58</td>
<td>18</td>
<td>5.96</td>
<td>Co-authorship network in hep-th category of ArXiv (Leskovec et al., 2005b)</td>
</tr>
<tr>
<td><strong>Web graphs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WEB-NOTREDAIME</td>
<td>325,729</td>
<td>1,497,134</td>
<td>325,729</td>
<td>1.00</td>
<td>0.47</td>
<td>46</td>
<td>7.22</td>
<td>Web graph of University of Notre Dame (Albert et al., 1999)</td>
</tr>
<tr>
<td><strong>Internet networks</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS-NEWMAN</td>
<td>22,963</td>
<td>96,872</td>
<td>22,963</td>
<td>1.00</td>
<td>0.35</td>
<td>11</td>
<td>3.83</td>
<td>AS graph from Newman (new, July 16, 2007)</td>
</tr>
<tr>
<td>AS-ROUTEVIEW</td>
<td>6,474</td>
<td>26,467</td>
<td>6,474</td>
<td>1.00</td>
<td>0.4</td>
<td>9</td>
<td>3.72</td>
<td>AS from Oregon Route View (Leskovec et al., 2005b)</td>
</tr>
<tr>
<td>GNUTELLA-25</td>
<td>22,687</td>
<td>54,705</td>
<td>22,663</td>
<td>1.00</td>
<td>0.01</td>
<td>11</td>
<td>5.57</td>
<td>Gnutella P2P network on 3/25 2000 (Ripeanu et al., 2002)</td>
</tr>
<tr>
<td>GNUTELLA-30</td>
<td>36,682</td>
<td>88,328</td>
<td>36,646</td>
<td>1.00</td>
<td>0.01</td>
<td>11</td>
<td>5.75</td>
<td>Gnutella P2P network on 3/30 2000 (Ripeanu et al., 2002)</td>
</tr>
<tr>
<td><strong>Bi-partite networks</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATP-GR-QC</td>
<td>19,177</td>
<td>26,169</td>
<td>14,832</td>
<td>0.77</td>
<td>0.0</td>
<td>35</td>
<td>11.08</td>
<td>Affiliation network of gr-qc category in ArXiv (Leskovec et al., 2007b)</td>
</tr>
<tr>
<td><strong>Biological networks</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIO-PROTEINS</td>
<td>4,626</td>
<td>29,602</td>
<td>4,626</td>
<td>1.00</td>
<td>0.12</td>
<td>12</td>
<td>4.24</td>
<td>Yeast protein interaction network (Colizza et al., 2005)</td>
</tr>
</tbody>
</table>

Table 5: Network datasets we analyzed. Statistics of networks we consider: number of nodes $N$; number of edges $E$; number of nodes in largest connected component $N_c$; fraction of nodes in largest connected component $N_c/N$; average clustering coefficient $C$; diameter $D$; and average path length $\bar{D}$. 