

Kronecker graphs: An Approach to Modeling Networks

Jure Leskovec

Computer Science Department, Stanford University

JURE@CS.STANFORD.EDU

Deepayan Chakrabarti

Yahoo! Research

DEEPAY@YAHOO-INC.COM

Jon Kleinberg

Computer Science Department, Cornell University

KLEINBER@CS.CORNELL.EDU

Christos Faloutsos

Computer Science Department, Carnegie Mellon University

CHRISTOS@CS.CMU.EDU

Zoubin Ghahramani

Department of Engineering, University of Cambridge

Machine Learning Department, Carnegie Mellon University

ZOUBIN@ENG.CAM.AC.UK

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. Current network models and generators either fail to match several of the above properties, are complicated to analyze mathematically, or both. Here 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 which 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 a wide range of large real and synthetic networks show that KRONFIT finds accurate parameters that very well mimic the properties of target networks. In fact, using just four parameters we can accurately model several aspects of global network structure. 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

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.

In parallel with discoveries of such structural “laws” there has been effort to find mechanisms and models of network formation that generate networks with such structures. So, a good realistic network generation model is important for at least two reasons. The first is that it can generate graphs for extrapolations, hypothesis testing, “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 network properties that generative models should obey to be realistic.

In this paper we introduce Kronecker graphs, a generative network model which obeys all the main static network patterns that have appeared in the literature (Faloutsos et al., 1999; Albert et al., 1999; Chakrabarti et al., 2004; Farkas et al., 2001; Mihail and Papadimitriou, 2002; Watts and Strogatz, 1998). Our model also obeys recently discovered temporal evolution patterns (Leskovec et al., 2005b, 2007a). And, contrary to other models that match this combination of network properties (as for example, (Bu and Towsley, 2002; Klemm and Eguíluz, 2002; Vázquez, 2003; Leskovec et al., 2005b; Zheleva et al., 2009)), 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. They 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 graphs 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; Imrich, 1998; Imrich and Klavžar, 2000; Hammack, 2009)), 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 the 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 where many properties that are also found in real networks naturally emerge. (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, how do we estimate it faster than in time quadratic on the number

of nodes? How do we solve the node correspondence problem, i.e., 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” node correspondences, just evaluating the likelihood is still prohibitively expensive for large graphs that we consider, as one needs to evaluate the probability of each possible edge. 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 for Kronecker graph model parameter estimation 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 graphs 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) *Network structure*: the parameters give us insight into the global 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 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 or 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 graph 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 a wide range of 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, such as 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 the 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 γ 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” (Milgram, 1967)): **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 *integer 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. Here we make use of *effective diameter* which we define as follows (Leskovec et al., 2005b). For each natural number h , let $g(h)$ denote the fraction of connected node pairs whose shortest connecting path has length at most h , *i.e.*, at most h hops away. We then consider a function defined over all positive real numbers x by linearly interpolating between the points $(h, g(h))$ and $(h + 1, g(h + 1))$ for each h , where $h = \lfloor x \rfloor$, and we define the *effective diameter* of the network to be the value x at which the function $g(x)$ achieves the value 0.9. 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: It 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 (Mihail and Papadimitriou, 2002; 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: Edges in real-world networks and especially in social networks tend to cluster (Watts and Strogatz, 1998) and form triads of connected nodes. 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 Δ versus the number of nodes that participate in Δ 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 real networks this does not hold as the diameter shrinks and then seems to stabilize 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; Flaxman et al., 2007) 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 graph 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 define a mechanism or a principle by which a network is constructed, and thus parameter estimation is either trivial or almost impossible.

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 Bezáková *et al.* (Bezáková et al., 2006) used permu-

SYMBOL	DESCRIPTION
G	Real network
N	Number of nodes in G
E	Number of edges in G
K	Kronecker graph (synthetic estimate of G)
K_1	Initiator of a Kronecker graphs
N_1	Number of nodes in initiator K_1
E_1	Number of edges in K_1 (the expected number of edges in \mathcal{P}_1 , $E_1 = \sum \theta_{ij}$)
$G \otimes H$	Kronecker product of adjacency matrices of graphs G and H
$K_1^{[k]} = K_k = K$	k^{th} Kronecker power of K_1
$K_1[i, j]$	Entry at row i and column j of K_1
$\Theta = \mathcal{P}_1$	Stochastic Kronecker initiator
$\mathcal{P}_1^{[k]} = \mathcal{P}_k = \mathcal{P}$	k^{th} Kronecker power of \mathcal{P}_1
$\theta_{ij} = \mathcal{P}_1[i, j]$	Entry at row i and column j of \mathcal{P}_1
$p_{ij} = \mathcal{P}_k[i, j]$	Probability of an edge (i, j) in \mathcal{P}_k , i.e., entry at row i and column j of \mathcal{P}_k
$K = R(\mathcal{P})$	Realization of a Stochastic Kronecker graph \mathcal{P}
$l(\Theta)$	Log-likelihood. Log-prob. that Θ generated real graph G , $\log P(G \Theta)$
$\hat{\Theta}$	Parameters at maximum likelihood, $\hat{\Theta} = \operatorname{argmax}_{\Theta} P(G \Theta)$
σ	Permutation that maps node IDs of G to those of \mathcal{P}
a	Densification power law exponent, $E(t) \propto N(t)^a$
D	Diameter of a graph
N_c	Number of nodes in the largest weakly connected component of a graph
ω	Proportion of times SwapNodes permutation proposal distribution is used

Table 1: Table of symbols.

tations 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 graph model seems to have the necessary expressive power to mimic real networks well.

3. Kronecker graph model

The Kronecker graph 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

graphs K_2, K_3, \dots 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 yield graphs satisfying the densification power law.

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

$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B} \doteq \begin{pmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \dots & a_{1,m}\mathbf{B} \\ a_{2,1}\mathbf{B} & a_{2,2}\mathbf{B} & \dots & a_{2,m}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}\mathbf{B} & a_{n,2}\mathbf{B} & \dots & a_{n,m}\mathbf{B} \end{pmatrix} \quad (1)$$

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 (Weichsel, 1962)) *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)

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

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.

The last observation is 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:*

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

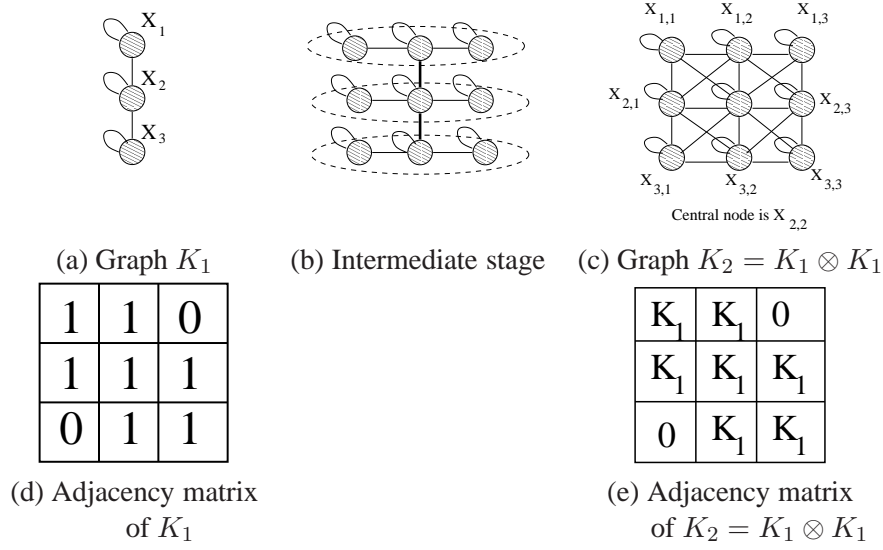


Figure 1: *Example of Kronecker multiplication:* Top: a “3-chain” initiator graph and its Kronecker product with itself. Each of the X_i nodes gets expanded into 3 nodes, which are then linked using Observation 1. Bottom row: the corresponding adjacency matrices. See figure 2 for adjacency matrices of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

Definition 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.

Note that there are many different names to refer to Kronecker product of graphs. Other names for the Kronecker product are tensor product, categorical product, direct product, cardinal product, relational product, conjunction, weak direct product or just product, and even Cartesian product (Imrich and Klavžar, 2000).

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.

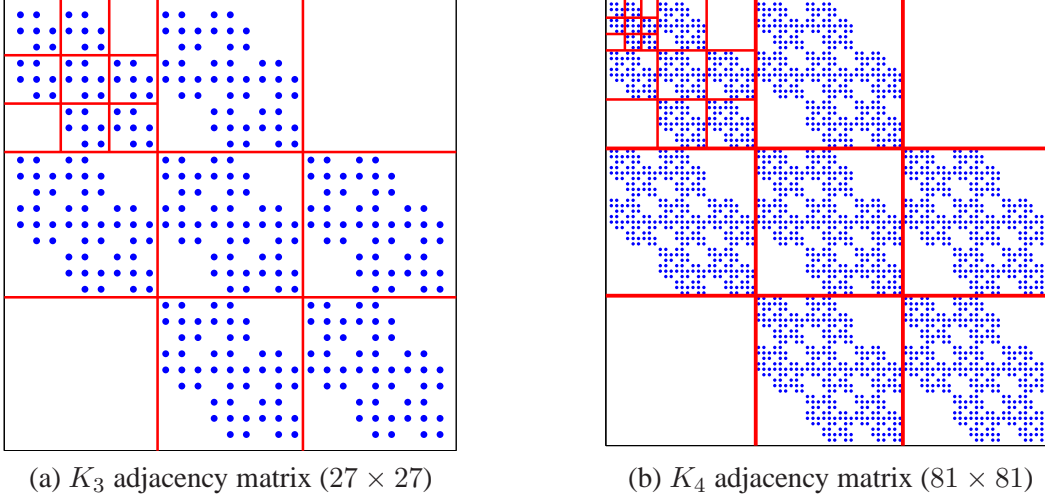


Figure 2: Adjacency matrices of K_3 and K_4 , the 3^{rd} and 4^{th} Kronecker power of K_1 matrix as defined in Figure 1. Dots represent non-zero matrix entries, and white space represents zeros. Notice the recursive self-similar structure of the adjacency matrix.

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 Discrete Gaussian Exponential (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, \dots, d_{N_1} . Kronecker multiplication of a node with degree d expands it into N_1 nodes, with the corresponding degrees being $d \times d_1, d \times d_2, \dots, d \times d_{N_1}$. After Kronecker powering, the degree of each node in graph K_k is of the form $d_{i_1} \times d_{i_2} \times \dots \times d_{i_k}$, with $i_1, i_2, \dots, i_k \in (1 \dots N_1)$, 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: $d_1^{i_1} d_2^{i_2} \dots d_{N_1}^{i_{N_1}}$ (where $\sum_{j=1}^{N_1} i_j = k$) and event (degree) probabilities will be proportional to $\binom{k}{i_1 i_2 \dots i_{N_1}}$. 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 $(d_1, d_2, \dots, d_{N_1})$. ■

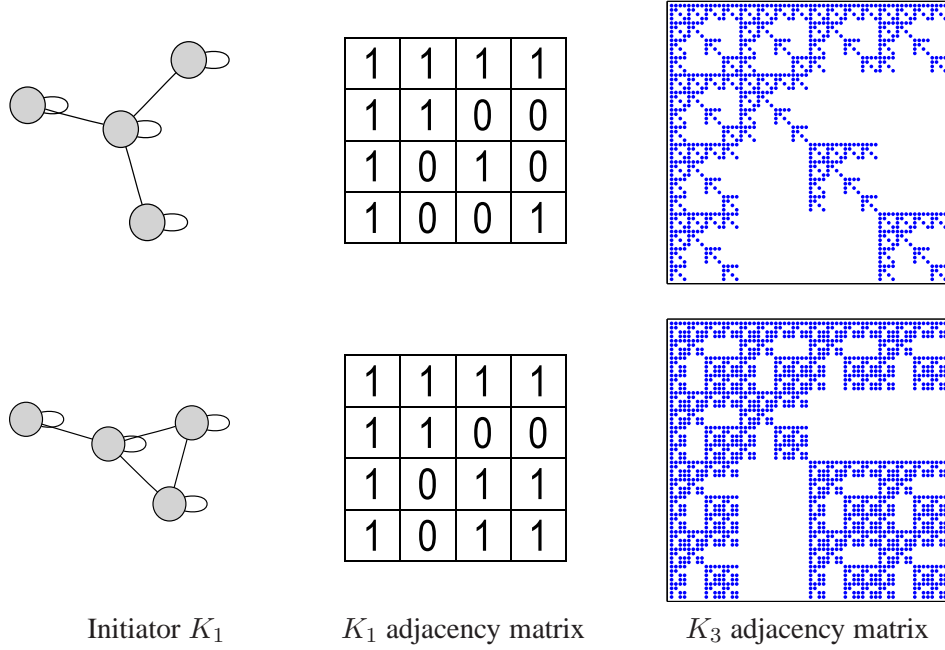


Figure 3: Two examples of Kronecker initiators on 4 nodes and the self-similar adjacency matrices they produce.

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 $\lambda_1, \lambda_2, \dots, \lambda_{N_1}$** . 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, $(\lambda_1, \lambda_2, \dots, \lambda_{N_1})^{[k]}$** . 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 $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_{N_1}$** . 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: $(\vec{v}_1, \vec{v}_2, \dots, \vec{v}_{N_1})$, 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 G have nodes X_1, \dots, X_n , where nodes $\{X_1, \dots, X_r\}$ and $\{X_{r+1}, \dots, X_n\}$ form the two connected components. Now, note that $(X_{ij}, X_{kl}) \notin G \otimes H$ for $i \in \{1, \dots, r\}$, $k \in \{r+1, \dots, n\}$, 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 while $G \otimes H$ is 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 $A = \{X_1, \dots, X_r\}$ and $B = \{X_{r+1}, \dots, X_n\}$, 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 $C = \{X_1, \dots, X_s\}$ and $D = \{X_{s+1}, \dots, X_m\}$. 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, D) and (B, C) . 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. Stars, chains, trees and cycles of even length are all examples of bipartite graphs. In order to ensure that K_k is connected, for the remainder of the paper we focus on initiator graphs K_1 with self loops on all of the vertices.

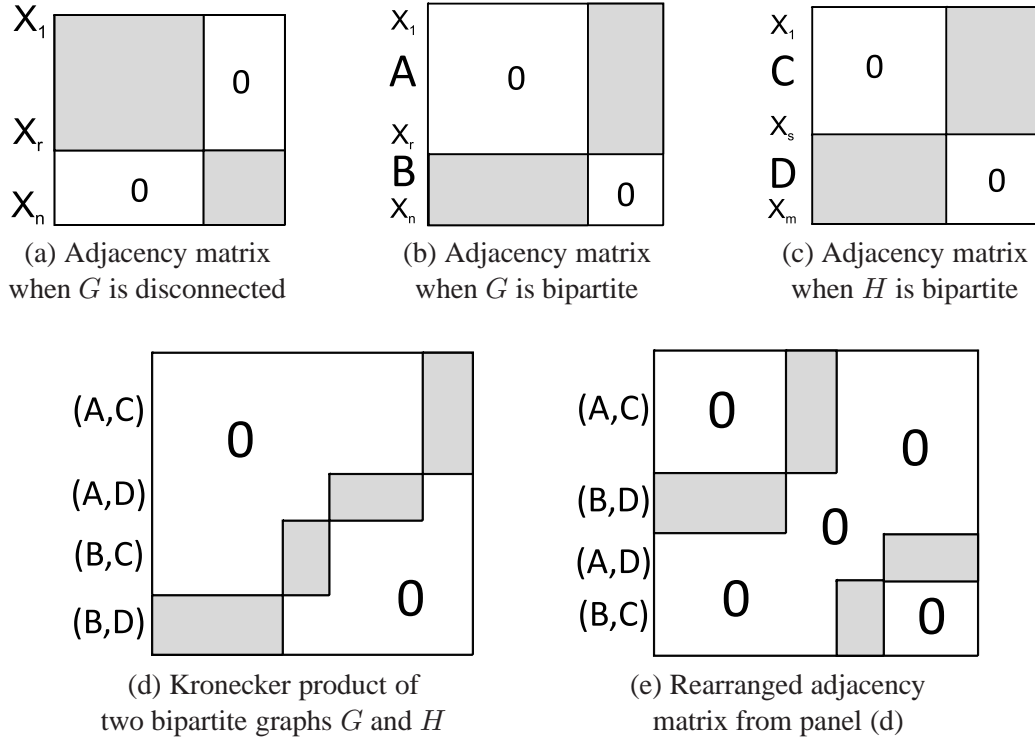


Figure 4: Graph adjacency matrices. Dark parts represent connected (filled with ones) and white parts represent empty (filled with zeros) parts of the adjacency matrix. (a) When G is disconnected, Kronecker multiplication with any matrix H will result in $G \otimes H$ being disconnected. (b) Adjacency matrix of a connected bipartite graph G with node partitions A and B . (c) Adjacency matrix of a connected bipartite graph G with node 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.

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 $v = v_1, v_2, \dots, v_r = v'$, where $r \leq D$. If $r < D$, we can convert this into a path $v = v_1, v_2, \dots, v_D = v'$ of length exactly D , by simply repeating v' at the end for $D - r$ times. By an analogous argument, we have a path $w = w_1, w_2, \dots, w_D = w'$. 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 $(v, w) = (v_1, w_1), (v_2, w_2), \dots, (v_D, w_D) = (v', w')$ 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 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 below) 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 $v = (v_1, \dots, v_k)$ and $w = (w_1, \dots, w_k)$ 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 $1 - (1 - \frac{1}{N_1^2})^k$, 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 $1 - (1 - \frac{1}{N_1^2})^k$ converges to 1 as k increases, it follows that the q -effective diameter is converging to D . ■

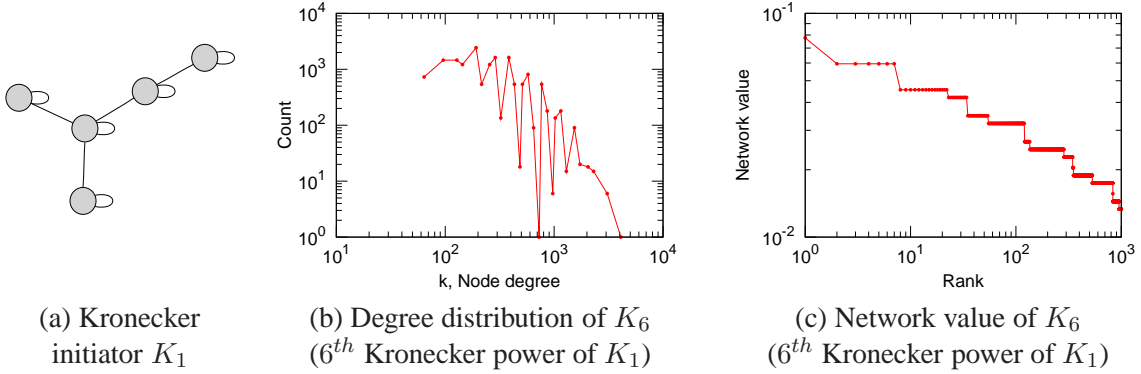


Figure 5: The “staircase” effect. Kronecker initiator and the degree distribution and network value plot for the 6^{th} Kronecker power of the initiator. Notice the non-smoothness of the curves.

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 graph 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 an 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 ϵ 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 graph model* which 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. Instead we 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 a stochastic adjacency matrix 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} , $p_{uv} \in \mathcal{P}_k$.

First, note that sum of the entries of \mathcal{P}_1 , $\sum_{i,j} \theta_{ij}$, 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 \mathcal{P}_k . This means the time to get a realization K is quadratic in the size of \mathcal{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 \mathcal{P}_k .

3.3.1 PROBABILITY OF AN EDGE

For the size of graphs we aim to model and generate here taking \mathcal{P}_1 (or K_1) and then explicitly performing the Kronecker product of the initiator matrix is infeasible. The reason for this is that \mathcal{P}_1 is usually dense, so \mathcal{P}_k is also dense and one can not explicitly store it in memory to directly iterate the Kronecker product. However, due to the structure of Kronecker multiplication one can easily compute the probability of an edge in \mathcal{P}_k .

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

$$p_{uv} = \prod_{i=0}^{k-1} \mathcal{P}_1 \left[\left\lfloor \frac{u-1}{N_1^i} \right\rfloor (\text{mod } N_1) + 1, \left\lfloor \frac{v-1}{N_1^i} \right\rfloor (\text{mod } N_1) + 1 \right] \quad (2)$$

The equation imitates recursive descent into the matrix \mathcal{P} , where at every level i the appropriate entry of \mathcal{P}_1 is chosen. Since \mathcal{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 \mathcal{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).

General overview of Kronecker product is given in (Imrich and Klavžar, 2000) and properties of Kronecker graphs related to graph minors, planarity, cut vertex and cut edge have been explored in (Bottreau and Metivier, 1998). 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 .

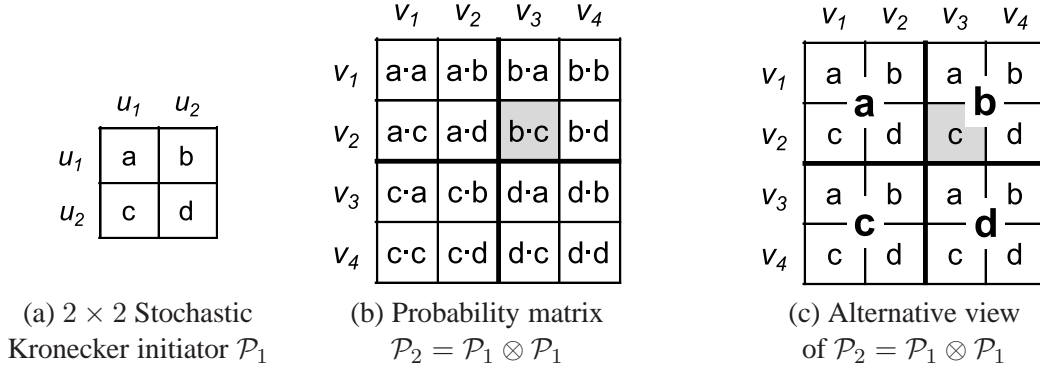


Figure 6: Stochastic Kronecker initiator \mathcal{P}_1 and the corresponding 2^{nd} Kronecker power \mathcal{P}_2 . Notice the recursive nature of the Kronecker product, with edge probabilities in \mathcal{P}_2 simply being products of entries of \mathcal{P}_1 .

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 are hierarchically organized into communities (clusters). Communities then 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 intuition comes from viewing every node of \mathcal{P}_k as being described with an ordered sequence of k nodes from \mathcal{P}_1 . (This is similar to the Observation 1 and the proof of Theorem 13.)

Let’s label nodes of the initiator matrix \mathcal{P}_1 , u_1, \dots, u_{N_1} , and nodes of \mathcal{P}_k as $v_1, \dots, v_{N_1^k}$. Then every node v_i of \mathcal{P}_k is described with a sequence $(v_i(1), \dots, v_i(k))$ of node labels of \mathcal{P}_1 , where $v_i(l) \in \{u_1, \dots, u_k\}$. Similarly, consider also a second node v_j with the label sequence $(v_j(1), \dots, v_j(k))$. Then the probability p_e of an edge (v_i, v_j) in \mathcal{P}_k is exactly:

$$p_e(v_i, v_j) = \mathcal{P}_k[v_i, v_j] = \prod_{l=1}^k \mathcal{P}_1[v_i(l), v_j(l)]$$

(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 $(v_i(1), \dots, v_i(k))$. Then $p_e(v_i, v_j)$ is exactly the coordinate-wise product of appropriate entries of \mathcal{P}_1 , where the node description sequence selects which entries of \mathcal{P}_1 to multiply. Thus, the \mathcal{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 -valued 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 \mathcal{P}_1 having high value entries on the diagonal. Or heterophily (nodes that differ are more likely to link) by \mathcal{P}_1 having high entries off the diagonal.

Figure 6 shows an example. Let's label nodes of \mathcal{P}_1 u_1, u_2 as in Figure 6(a). Then every node of \mathcal{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 \mathcal{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 $\mathcal{P}_1[u_2, u_1] \cdot \mathcal{P}_1[u_1, u_2] = b \cdot c$ — the product of entries of \mathcal{P}_1 , where the corresponding elements of the description of nodes v_2 and v_3 act as selectors of which entries of \mathcal{P}_1 to multiply.

<--- Binary bro:
 - v1 = u1, u1
 - v2 = u1, u2
 - v3 = u2, u1
 - v4 = u2, u2

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 \mathcal{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 2×2 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 that 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 matrix \mathcal{P}_k has attribute sequence (u_1, u_1, \dots, u_1) , and v_{N_1} has $(u_1, u_1, \dots, u_1, u_{N_1})$, while the “last” node $v_{N_1^k}$ is has attribute values $(u_{N_1}, u_{N_1}, \dots, u_{N_1})$. 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, \dots, u_1)) to N_1^k (i.e., “rightmost” node attribute description $(u_{N_1}, u_{N_1}, \dots, u_{N_1})$).

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 \mathcal{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 3×3 initiator matrix with only four parameters. Adopting the naming convention of Figure 6 this means that parameter a now occupies a 2×2 block, which then also makes b and c occupy 2×1 and 1×2 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 graph 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 \mathcal{P}_k one has to make a sequence of k (in our case $k = 2$) decisions among the entries of \mathcal{P}_1 , multiply the chosen entries of \mathcal{P}_1 , and then placing the edge (v_i, v_j) with the obtained probability.

Instead of flipping $O(N^2) = O(N_1^{2k})$ 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 θ_{ij} , $\theta_{ij} \in \mathcal{P}_1$ until in k steps we descend to a single cell of the big adjacency matrix K 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 \mathcal{P}_k follows a Bernoulli distribution, as the edge occurrences are independent. By the Central Limit Theorem (Petrov, 1995) the number of edges in \mathcal{P}_k tends to a normal distribution with mean $(\sum_{i,j=1}^{N_1} \theta_{ij})^k = E_1^k$, where $\theta_{ij} \in \mathcal{P}_1$. So, given a stochastic initiator matrix \mathcal{P}_1 we first sample the expected number of edges E in \mathcal{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 θ_{ij}/E_1 where $E_1 = \sum_{i,j} \theta_{ij}$. Since we add $E = E_1^k$ edges, the probability that edge (v_i, v_j) appears in K is exactly $\mathcal{P}_k[v_i, v_j]$. 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 θ_{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) entry of big adjacency matrix K . If an edge lands in a already occupied cell we insert it again. Even though values of \mathcal{P}_1 are usually skewed, adjacency matrices of real networks are so sparse that this is not really a problem in practice. Empirically we note that around 1% of edges collide.

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:* \mathcal{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 (2×2 initiator matrix) or 9 (3×3 initiator), is enough to accurately model the structure of large networks.
- *Extension of Erdős-Rényi random graph model:* Stochastic Kronecker graphs represent an extension of Erdős-Rényi (Erdős and Rényi, 1960) random graphs. If one takes $\mathcal{P}_1 = [\theta_{ij}]$, where every $\theta_{ij} = p$ then we obtain exactly the Erdős-Rényi model of random graphs $G_{n,p}$, where every edge 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 \mathcal{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 θ_{ij} of the initiator matrix \mathcal{P}_1 so that $\sum_{ij} \theta_{ij} > N_1$ in order for the resulting network to densify.

4. Simulations of Kronecker graphs

Next we perform a set of simulation experiments to demonstrate the ability of Kronecker graphs to match the patterns of real-world networks. We will tackle the problem of estimating the Kronecker graph model from real data, *i.e.*, finding the most likely initiator \mathcal{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 \mathcal{P}_1 separately. However, we reduce the number of parameters from N_1^2 to just two: α and β . Let K_1 be the initiator matrix (binary, deterministic). Then we create the corresponding stochastic initiator matrix \mathcal{P}_1 by replacing each “1” and “0” of K_1 with α and β 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 automatically estimate \mathcal{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 to some degree 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 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, and 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-ROUTEVIEWS) 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.

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 (shown in top row of Figure 3), setting each “1” of K_1 to α and each “0” to $\beta = 0$ to obtain \mathcal{P}_1 that we then use to generate a growing sequence of graphs. We plot the effective diameter of each $R(\mathcal{P}_k)$ as we generate a sequence of growing graphs $R(\mathcal{P}_2), R(\mathcal{P}_3), \dots, R(\mathcal{P}_{10})$. $R(\mathcal{P}_{10})$ 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 α) we obtain graphs with slowly increasing effective diameter (Figure 9(a)). For intermediate values of α we get graphs with constant diameter (Figure 9(b)) and that in our case also slowly densify with densification exponent $a = 1.05$. 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 $a = 1.2$. 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 graphs where we choose a star on 4 nodes as a initiator graph and parameterized with α and β as before. The initiator graph and the structure of the corresponding (deterministic) Kronecker graph adjacency matrix is shown in top row of Figure 3.

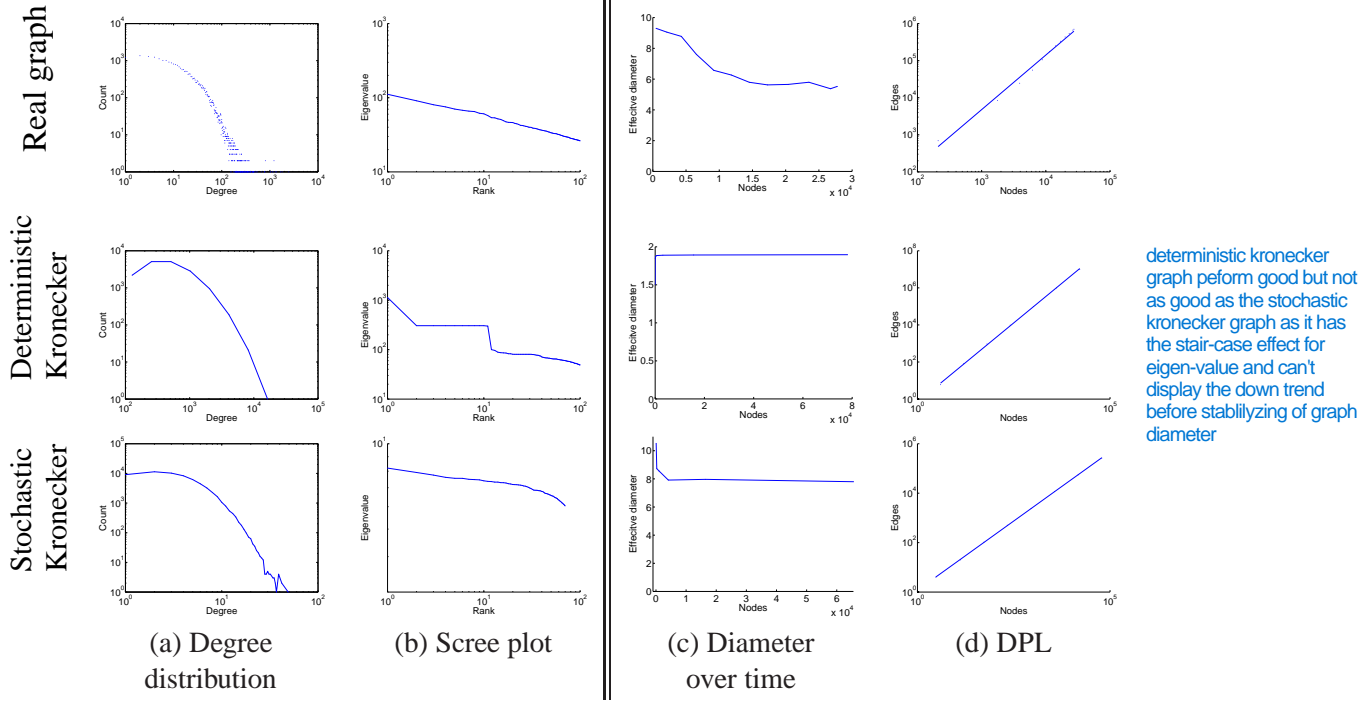


Figure 7: *Citation network (CIT-HEP-TH)*: Patterns from the real graph (top row), the deterministic Kronecker graph with K_1 being a star graph on 4 nodes (center + 3 satellites) (middle row), and the Stochastic Kronecker graph ($\alpha = 0.41$, $\beta = 0.11$ – bottom row). *Static* patterns: (a) is 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 graphs qualitatively matches all the patterns very well.

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 α . 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 (α, β) 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.

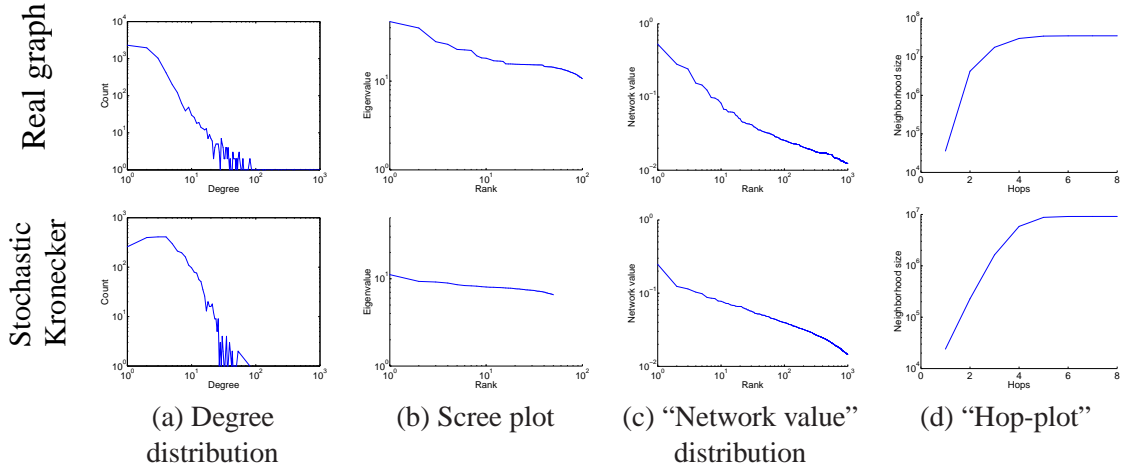


Figure 8: *Autonomous systems (AS-ROUTEVIEWS)*: Real (top) versus Kronecker (bottom). Columns (a) and (b) show the degree distribution and the scree plot, as before. Columns (c) and (d) show two more static patterns (see text). Notice that, again, the Stochastic Kronecker graph matches well the properties of the real graph.

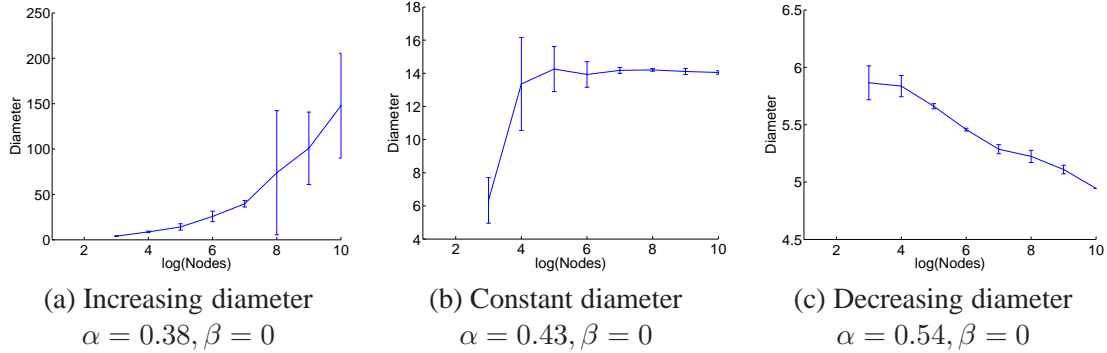


Figure 9: Effective diameter over time for a 4-node chain initiator graph. After each consecutive Kronecker power we measure the effective diameter. We use different settings of α parameter. $\alpha = 0.38, 0.43, 0.54$ and $\beta = 0$, respectively.

5. Kronecker graph model estimation

In previous sections we investigated various properties of networks generated by the (Stochastic) Kronecker graphs model. Many of these properties were also observed in real networks. Moreover, we also gave closed form expressions (parametric forms) for values of these statistical network properties, which allow us to calculate a property (*e.g.*, diameter, eigenvalue spectrum) of a network directly from just the initiator matrix. So in principle, one could invert these equations and directly get from a property (*e.g.*, shape of degree distribution) to the values of initiator matrix.

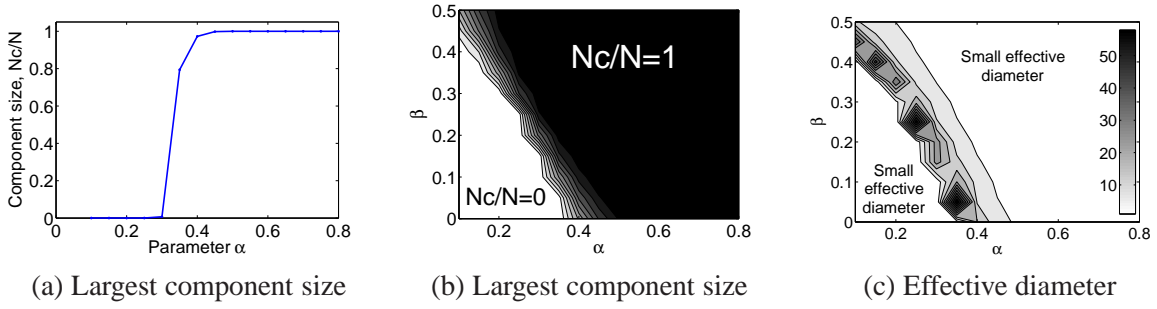


Figure 10: Fraction of nodes in the largest weakly connected component (N_c/N) and the effective diameter for 4-star initiator graph. (a) We fix $\beta = 0.15$ and vary α . (b) We vary both α and β . (c) Effective diameter of the network, if network is disconnected or very dense path lengths are short, the diameter is large when the network is barely connected.

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 two network properties are mutually exclusive. For instance, perhaps only could 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 \mathcal{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 network properties (statistics) to match, then define a quality of fit 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 to maximize the quality of fit 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 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 will still hold.

5.1 Preliminaries

Stochastic graph models induce 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) generates the graph G . We concentrate on the Stochastic Kronecker graphs model, and consider fitting it to a real graph G , our data. We use the maximum likelihood approach, *i.e.*, we aim to find parameter values, the initiator \mathcal{P}_1 , that maximize $P(G)$ under the Stochastic Kronecker graph model.

This presents several challenges:

- **Model selection:** a graph is a single structure, and not a set of items drawn independently and identically-distributed (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. The graph G has a set of N nodes, and each node has a unique label (index, 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 labels should have the same likelihood. A permutation σ 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 σ 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 \mathcal{P} .
- **Likelihood estimation:** Even if we assume one can efficiently solve the node correspondence problem, 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 graphs initiator matrix \mathcal{P}_1 . Here \mathcal{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 \mathcal{P}_1 we create a Stochastic Kronecker graphs probability matrix \mathcal{P}_k , where every entry p_{uv} of \mathcal{P}_k contains a probability that node u links to node

v. We then evaluate the probability that G is a realization of \mathcal{P}_k . The task is to find such \mathcal{P}_1 that has the highest probability of realizing (generating) G .

Formally, we are solving:

$$\arg \max_{\mathcal{P}_1} P(G|\mathcal{P}_1) \quad (3)$$

To keep the notation simpler we use standard symbol Θ to denote the parameter matrix \mathcal{P}_1 that we are trying to estimate. We denote entries of $\Theta = \mathcal{P}_1 = [\theta_{ij}]$, and similarly we denote $\mathcal{P} = \mathcal{P}_k = [p_{ij}]$. Note that here we slightly simplified the notation: we use Θ to refer to \mathcal{P}_1 , and θ_{ij} are elements of Θ . Similarly, p_{ij} are elements of $\mathcal{P} (\equiv \mathcal{P}_k)$. Moreover, we denote $K = R(\mathcal{P})$, i.e., K is a realization of the Stochastic Kronecker graph sampled from probabilistic adjacency matrix \mathcal{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 σ of the set $\{1, \dots, N\}$ 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:

$$\begin{aligned} l(\Theta) &= \log P(G|\Theta) = \log \sum_{\sigma} P(G|\Theta, \sigma) P(\sigma|\Theta) \\ &= \log \sum_{\sigma} P(G|\Theta, \sigma) P(\sigma) \end{aligned} \quad (4)$$

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

We then model edges as independent Bernoulli random variables parameterized by the parameter matrix Θ . 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 σ_i as the i^{th} element of the permutation σ , 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 $O(N^2)$ time.

We further illustrate the process of estimating Stochastic Kronecker initiator matrix Θ in Figure 11. We search over initiator matrices Θ to find the one that maximizes the likelihood $P(G|\Theta)$. To estimate $P(G|\Theta)$ we are given a concrete Θ 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

$$\arg \max_{\Theta} P(G | \Theta^{[k]}) \quad \leftarrow \text{Kronecker} \quad (\Theta)$$

Figure 11: Kronecker parameter estimation as an optimization problem. We search over the initiator matrices $\Theta (\equiv \mathcal{P}_1)$. Using Kronecker multiplication we create probabilistic adjacency matrix $\Theta^{[k]}$ that is of same size as real network G . Now, we evaluate the likelihood by simultaneously traversing and multiplying entries of G and $\Theta^{[k]}$ (see Eq. 5). As shown by the figure permutation σ plays an important role, as permuting rows and columns of G could make it look more similar to $\Theta^{[k]}$ and thus increase the likelihood.

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 (σ_u, σ_v) of \mathcal{P} . Then in case that edge (u, v) exists in G (i.e., $G[u, v] = 1$) the likelihood that particular edge existing is $\mathcal{P}[\sigma_u, \sigma_v]$, and similarly, in case the edge (u, v) does not exist the likelihood is simply $1 - \mathcal{P}[\sigma_u, \sigma_v]$. This also demonstrates the importance of permutation σ , as permuting rows and columns of G could make the adjacency matrix look more “similar” to $\Theta^{[k]}$, and would increase the likelihood.

So far we showed how to assess the quality (likelihood) of a particular Θ . So, naively one could perform some kind of grid search to find best Θ . However, this is very inefficient. A better way of doing it is to compute the gradient of the log-likelihood $\frac{\partial}{\partial \Theta} l(\Theta)$, and then use the gradient to update the current estimate of Θ 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 find a good solution, 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 now takes $O(N^2)$ time and needs to be evaluated $N!$ times. So, given a concrete Θ just naively calculating the likelihood takes $O(N!N^2)$ time, and then one also has to optimize over Θ .

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*.

input : size of parameter matrix N_1 , graph G on $N = N_1^k$ nodes, and learning rate λ
output: MLE parameters $\hat{\Theta}$ ($N_1 \times N_1$ probability matrix)

- 1 initialize $\hat{\Theta}_1$
- 2 **while** *not converged* **do**
- 3 evaluate gradient: $\frac{\partial}{\partial \hat{\Theta}_t} l(\hat{\Theta}_t)$
- 4 update parameter estimates: $\hat{\Theta}_{t+1} = \hat{\Theta}_t + \lambda \frac{\partial}{\partial \hat{\Theta}_t} l(\hat{\Theta}_t)$
- 5 **end**
- 6 **return** $\hat{\Theta} = \hat{\Theta}_t$

Algorithm 1: KRONFIT algorithm.

5.3 Summing over the node labelings

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

$$\begin{aligned}
 \frac{\partial}{\partial \Theta} l(\Theta) &= \frac{\sum_{\sigma} \frac{\partial}{\partial \Theta} P(G|\sigma, \Theta) P(\sigma)}{\sum_{\sigma'} P(G|\sigma', \Theta) P(\sigma')} \\
 &= \frac{\sum_{\sigma} \frac{\partial \log P(G|\sigma, \Theta)}{\partial \Theta} P(G|\sigma, \Theta) P(\sigma)}{P(G|\Theta)} \\
 &= \sum_{\sigma} \frac{\partial \log P(G|\sigma, \Theta)}{\partial \Theta} P(\sigma|G, \Theta)
 \end{aligned} \tag{6}$$

Note we are still summing over all $N!$ permutations σ , 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 Θ to obtain equivalent estimates. Therefore Θ is not strictly identifiable exactly because of these permutations. Since the space of permutations on N nodes is very large (grows as $N!$) the **permutation sampling algorithm 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

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

```

input : Parameter matrix  $\Theta$ , and graph  $G$ 
output: Log-likelihood  $l(\Theta)$ , and gradient  $\frac{\partial}{\partial \Theta} l(\Theta)$ 

1 for  $t := 1$  to  $T$  do
2    $\sigma_t := \text{SamplePermutation}(G, \Theta)$ 
3    $l_t = \log P(G|\sigma^{(t)}, \Theta)$ 
4    $\text{grad}_t := \frac{\partial}{\partial \Theta} \log P(G|\sigma^{(t)}, \Theta)$ 
5 end
6 return  $l(\Theta) = \frac{1}{T} \sum_t l_t$ , and  $\frac{\partial}{\partial \Theta} l(\Theta) = \frac{1}{T} \sum_t \text{grad}_t$ 
    
```

Algorithm 2: Calculating log-likelihood and gradient

where Z 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 σ and σ' (Equation 7) the normalizing constants nicely cancel out:

$$\frac{P(\sigma'|G, \Theta)}{P(\sigma|G, \Theta)} = \prod_{(u,v) \in G} \frac{\mathcal{P}[\sigma_u, \sigma_v]}{\mathcal{P}[\sigma'_u, \sigma'_v]} \prod_{(u,v) \notin G} \frac{(1 - \mathcal{P}[\sigma_u, \sigma_v])}{(1 - \mathcal{P}[\sigma'_u, \sigma'_v])} \quad (7)$$

$$= \prod_{\substack{(u,v) \in G \\ (\sigma_u, \sigma_v) \neq (\sigma'_u, \sigma'_v)}} \frac{\mathcal{P}[\sigma_u, \sigma_v]}{\mathcal{P}[\sigma'_u, \sigma'_v]} \prod_{\substack{(u,v) \notin G \\ (\sigma_u, \sigma_v) \neq (\sigma'_u, \sigma'_v)}} \frac{(1 - \mathcal{P}[\sigma_u, \sigma_v])}{(1 - \mathcal{P}[\sigma'_u, \sigma'_v])} \quad (8)$$

This immediately suggests the use of a 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 σ to a new permutation σ' . Probability of accepting the move to σ' 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 σ from the proposal distribution. One way to do this would be to simply generate a random permutation σ' 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 permutations (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 the permutation space longer. We do this by making samples dependent, i.e., given σ' we want to generate next candidate permutation σ'' 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 σ' from σ'' and, $\pi(\sigma')$ is the stationary distribution.

In Algorithm 3 we use a simple proposal where given permutation σ' we generate σ'' by swapping elements at two uniformly at random chosen positions of σ' . 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

```

input : Kronecker initiator matrix  $\Theta$  and a graph  $G$  on  $N$  nodes
output: Permutation  $\sigma^{(i)} \sim P(\sigma|G, \Theta)$ 
1  $\sigma^{(0)} := (1, \dots, N)$ 
2  $i = 1$ 
3 repeat
4   Draw  $j$  and  $k$  uniformly from  $(1, \dots, N)$ 
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, Θ): Metropolis sampling of the node permutation.

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 σ' will swap low degree nodes (as they form the majority of nodes).

A possibly more efficient way would be to swap elements of σ biased based on corresponding node degree, so that high degree nodes would get swapped more often. 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 nodes at the edge 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 ω and SwapEdgeEndpoints with probability $1 - \omega$.

To summarize we consider the following two permutation proposal distributions:

- $\sigma'' = \text{SwapNodes}(\sigma')$: we obtain σ'' by taking σ' , uniformly at random selecting a pair of elements and swapping their positions.
- $\sigma'' = \text{SwapEdgeEndpoints}(\sigma')$: we obtain σ'' from σ' by first sampling an edge (j, k) from G uniformly at random, then we take σ' 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 σ and σ' differ only at two positions, *i.e.* elements at position j and k are swapped, *i.e.*, σ and σ' 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 \mathcal{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 \mathcal{P} where σ and σ' differ.

Graphs we are working with here are too large to allow us to explicitly create and store the stochastic adjacency matrix \mathcal{P} by Kronecker powering the initiator matrix Θ . Every time probability $\mathcal{P}[i, j]$ 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 σ from $P(\sigma|G, \Theta)$ takes $O(kN)$.

This 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 $O(N^2)$. This is exactly what we investigate next and how to calculate the likelihood in *linear time*.

5.4 Efficiently approximating 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 \mathcal{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:

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

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 \mathcal{P} generated by the Kronecker product. We substitute the $\log(1 - p_{ij})$ with its Taylor approximation, which gives a sum over elements of \mathcal{P} and their squares. Next, we notice the sum of elements of \mathcal{P} forms a multinomial series, and thus $\sum_{i,j} p_{ij} = (\sum_{i,j} \theta_{ij})^k$, where θ_{ij} denotes an element of Θ , and p_{ij} element of $\Theta^{[k]}$.

Calculating log-likelihood of G now takes $O(E)$: First, we approximate 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:

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

We note that by using the second order Taylor approximation to the log-likelihood of an empty graph, the error term of the approximation is $\frac{1}{3}(\sum_i \theta_{ij}^3)^k$, which can diverge for large k . For typical values of initiator matrix \mathcal{P}_1 (that we present in Section 6.5) we note that one needs about fourth or fifth order Taylor approximation for the error of the approximation actually go to zero as k approaches infinity, *i.e.*, $\sum_{ij} \theta_{ij}^{n+1} < 1$, where n is the order of Taylor approximation employed.

5.5 Calculating the gradient

Calculation of the gradient of the log-likelihood follows exactly the same pattern as described above. First by using the Taylor approximation we calculate the 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 σ and σ' 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 $\partial \mathcal{P} / \partial \Theta$ 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 matrix Θ ? 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 Θ , 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 graph 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 Θ . This is then a *curved exponential family* (Efron, 1975), and BIC naturally applies:

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

where $\hat{\Theta}_{N_1}$ are the maximum likelihood parameters of the model with $N_1 \times N_1$ parameter matrix, 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.

As an alternative 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

Next we described our experiments on a range of real and synthetic networks. 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 synthetic Kronecker graphs to see whether KRONFIT is able to recover the parameters used to generate the network. Last, we consider fitting Stochastic Kronecker graphs 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 $\mathcal{P}_1^* = [0.8, 0.6; 0.5, 0.3]$, and $k = 14$ which gives us a graph G on $N = 16,384$ nodes and $E = 115,741$ edges. We chose this particular \mathcal{P}_1^* as it resembles the typical initiator for real networks analyzed later in this section.

6.1.1 CONVERGENCE OF THE LOG-LIKELIHOOD AND THE GRADIENT

First, we examine the convergence of Metropolis permutation sampling, where permutations are sampled sequentially. A new permutation is obtained by modifying the previous one which creates a Markov chain. We want to assess the convergence and mixing of the chain. We aim to determine how many permutations one needs to draw to reliably estimate the likelihood and the gradient, and also how long does it take until the samples converge to the stationary distribution. For the experiment we generated a synthetic Stochastic Kronecker graphs using \mathcal{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 a Stochastic Kronecker graphs 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 graphs).

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 θ_{11} and θ_{22} of Θ' and \mathcal{P}_1^* match, so gradients of these two parameters should converge to zero and indeed they do. On the other hand, θ_{12} and θ_{21} differ between Θ' and \mathcal{P}_1^* . Notice, the gradient for one is positive as the parameter θ_{12} of Θ' should be decreased, and similarly for θ_{21} the gradient is negative as the parameter value should be increased to match the Θ' . In summary, this shows that log-likelihood and gradients rather quickly converge to their true values.

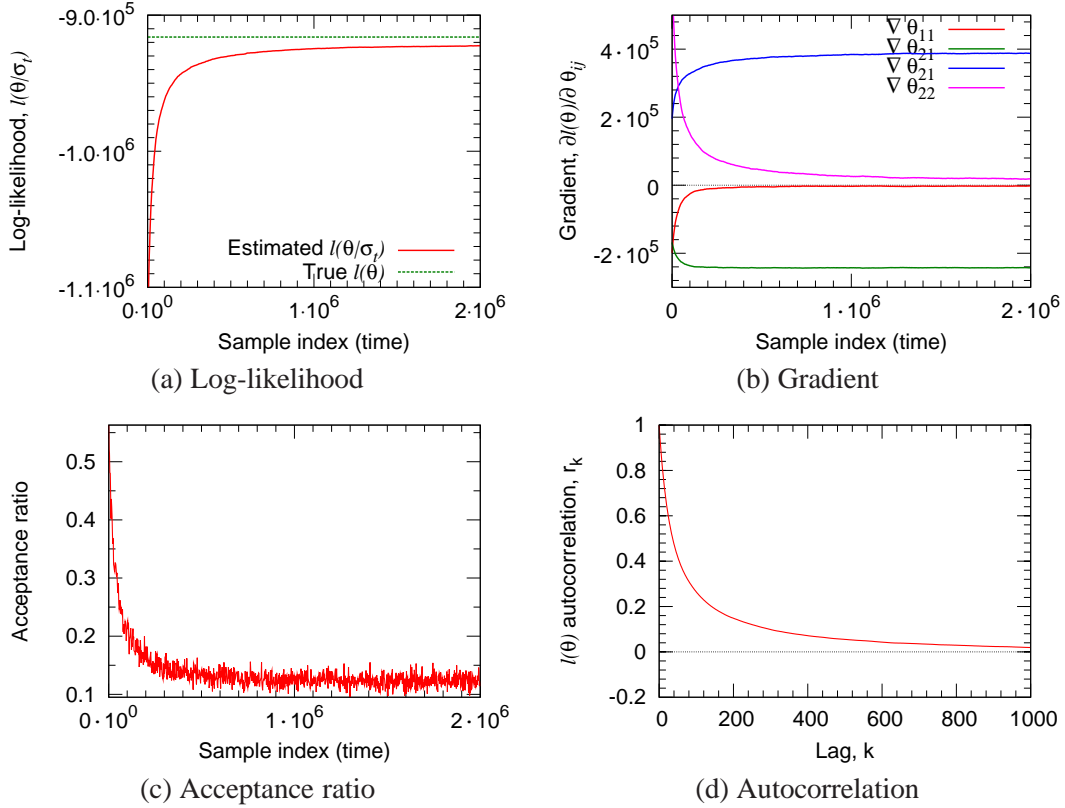


Figure 12: Convergence of the log-likelihood and components of the gradient towards their true values for Metropolis permutation sampling (Algorithm 3) with the number of samples.

In Figures 12(c) and (d) we also 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 implies better the mixing and thus one needs less samples to accurately estimate the gradient or the likelihood. Notice the rather fast autocorrelation decay.

All in all, these experiments show that one needs to sample on the 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 the previous step to initialize the permutation at the current step of the gradient descent. Intuitively, small changes in parameter space Θ also mean small changes in $P(\sigma|G, \Theta)$.

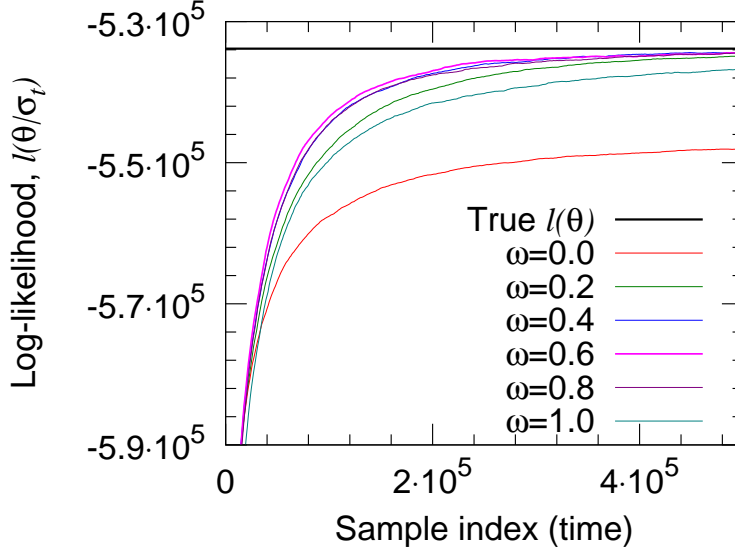


Figure 13: Convergence of the log-likelihood and gradients for Metropolis permutation sampling (Algorithm 3) for different choices of ω that interpolates between the `SwapNodes` ($\omega = 1$) and `SwapEdgeEndpoints` ($\omega = 0$) permutation proposal distributions. Notice fastest convergence of log-likelihood for $\omega = 0.6$.

6.1.2 DIFFERENT PROPOSAL DISTRIBUTIONS

In section 5.3.1 we defined two permutation sampling strategies: `SwapNodes` where we pick two nodes uniformly at random and swap their labels (node ids), and `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 `SwapNodes` with probability ω , and `SwapEdgeEndpoints` with probability $1 - \omega$.

So, given a Stochastic Kronecker graphs G on $N = 16,384$ and $E = 115,741$ generated from $\mathcal{P}_1^* = [0.8, 0.7; 0.5, 0.3]$ 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 ω which interpolates between the two permutation proposal distributions. The quicker the convergence 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 implies better permutation proposal distribution. When we use only `SwapNodes` ($\omega = 1$) or `SwapEdgeEndpoints` ($\omega = 0$) convergence is rather slow. We obtain best convergence for ω around 0.6.

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

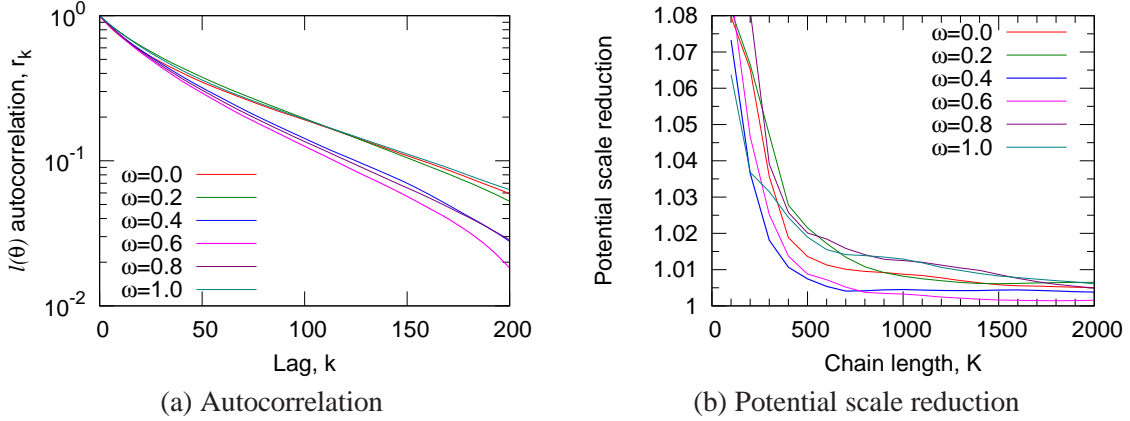


Figure 14: (a) Autocorrelation plot of the log-likelihood for the different choices of parameter ω . Notice we get best mixing with $\omega \approx 0.6$. (b) The potential scale reduction that compares the variance inside- and across- independent Markov chains for different values of parameter ω .

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 shorter 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 $j = 1, \dots, J$ and $k = 1, \dots, K$. 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 $l_j^{(k)}$.

First, we compute between and within chain variances $\hat{\sigma}_B^2$ and $\hat{\sigma}_W^2$, where between-chain variance is obtained by

$$\hat{\sigma}_B^2 = \frac{K}{J-1} \sum_{j=1}^J (\bar{l}_{\cdot j} - \bar{l}_{\cdot\cdot})^2$$

where $\bar{l}_{\cdot j} = \frac{1}{K} \sum_{k=1}^K l_j^{(k)}$ and $\bar{l}_{\cdot\cdot} = \frac{1}{J} \sum_{j=1}^J \bar{l}_{\cdot j}$

Similarly the within-chain variance is defined by

$$\hat{\sigma}_W^2 = \frac{1}{J(K-1)} \sum_{j=1}^J \sum_{k=1}^K (l_j^{(k)} - \bar{l}_{\cdot j})^2$$

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

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

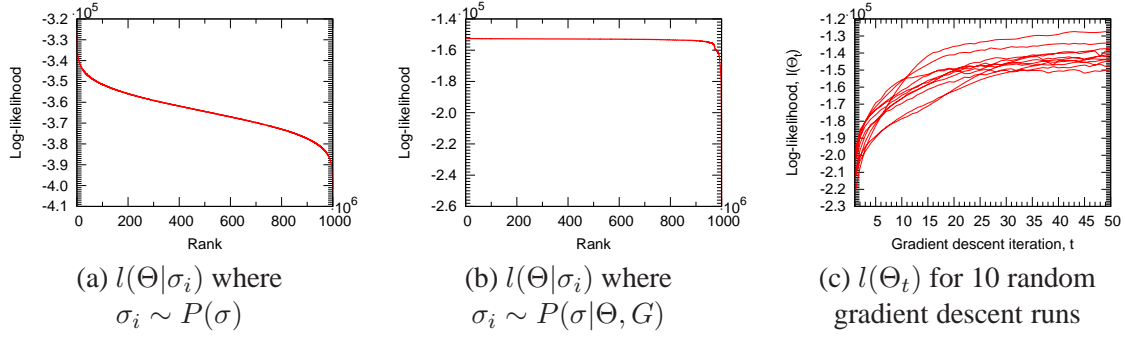


Figure 15: (a) Distribution of log-likelihood of permutations sampled uniformly at random, and (b) when sampled from $P(\sigma|\Theta, G)$. Notice the space of good permutations is rather small but our sampling quickly finds permutations of high likelihood. (c) Convergence of log-likelihood for 10 runs of gradient descent, each from a different random starting point.

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

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

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

Figure 14(b) gives the Gelman-Rubin-Brooks plot, where we plot the potential scale reduction $\sqrt{\hat{R}}$ over the increasing chain length K for different choices of parameter ω . Notice that the potential scale reduction quickly decays towards 1. Similarly as in Figure 14 the extreme values of ω give slow decay, while we obtain the 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 are they discovered. For the experiment we took a real network G (AS-ROUTEVIEWS network) and the MLE parameters $\hat{\Theta}$ for it that we estimated before hand ($l(\hat{\Theta}) \approx -150,000$). 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 σ_i uniformly at random, *i.e.*, $P(\sigma_i) = 1/(6,474!)$ and for each evaluated its log-likelihood $l(\sigma_i|\Theta) = \log P(\Theta|G, \sigma_i)$. We ordered the permutations in decreasing log-likelihood and plotted $l(\sigma_i|\Theta)$ 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 using permutations sampled from the permutation distribution $\sigma_i \sim P(\sigma|\Theta, G)$ via 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 likelihood of $\approx -150,000$. This suggests 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, and spends the most time there.

6.2 Properties of the optimization space

In maximizing the likelihood we use a 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 Θ all have the same log-likelihood. This means that we have several global minima that correspond to permutations of parameter matrix Θ , and then between them the log-likelihood drops. This means that the optimization problem is non-convex. ■

The above observation does not seem promising for estimating $\hat{\Theta}$ using gradient descent as it is prone to finding local minima. To test for this behavior 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, each with a randomly chosen 2×2 parameter matrix Θ^* . For each of the 100 graphs we run a single trial of gradient descent starting from a random parameter matrix Θ' , and try to recover Θ^* . 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 Θ^* . 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 Θ_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 Θ_0 . Notice that in all runs gradient descent always converges towards the optimum, and none of the runs gets stuck in some local maxima.

6.3 Convergence of the graph properties

We approached the problem of estimating Stochastic Kronecker initiator matrix Θ by defining the likelihood over the individual entries of the graph adjacency matrix. However, what we would really

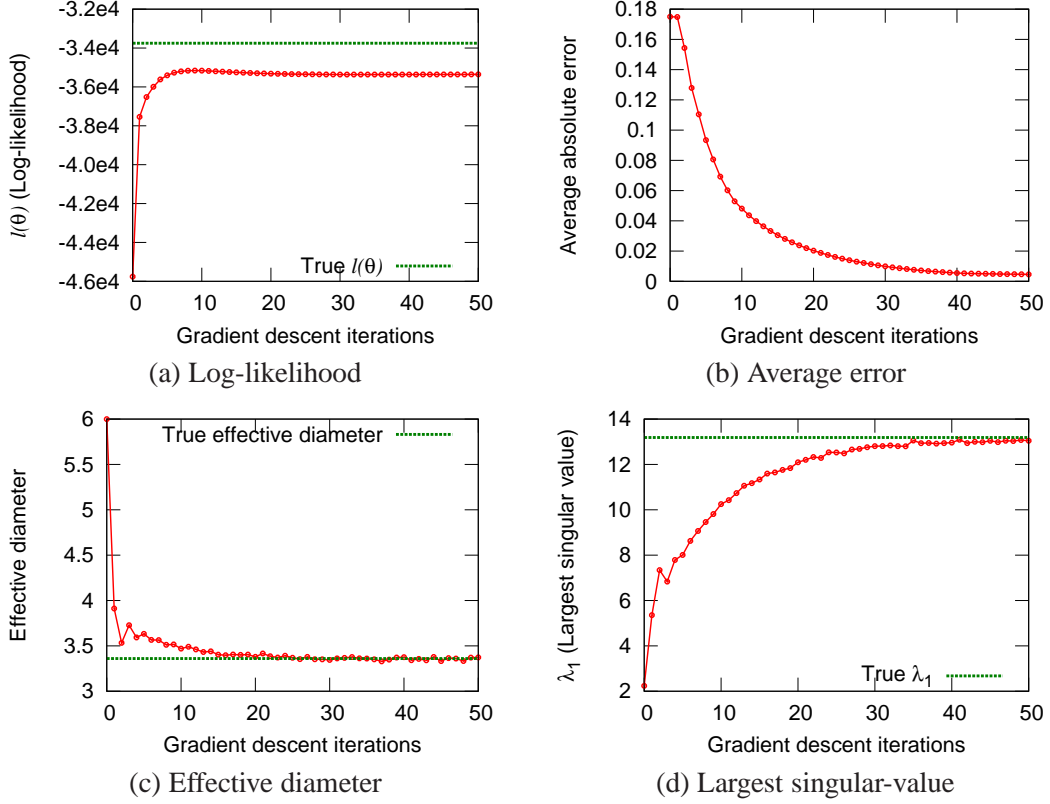


Figure 16: Convergence of graph properties with the number of iterations of gradient descent using the synthetic dataset. We start with a random choice of parameters and with steps of gradient descent the Kronecker graph better and better matches network properties of the target graph.

like is to be given a real graph G and then generate a synthetic graph K that has similar properties as the real 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:

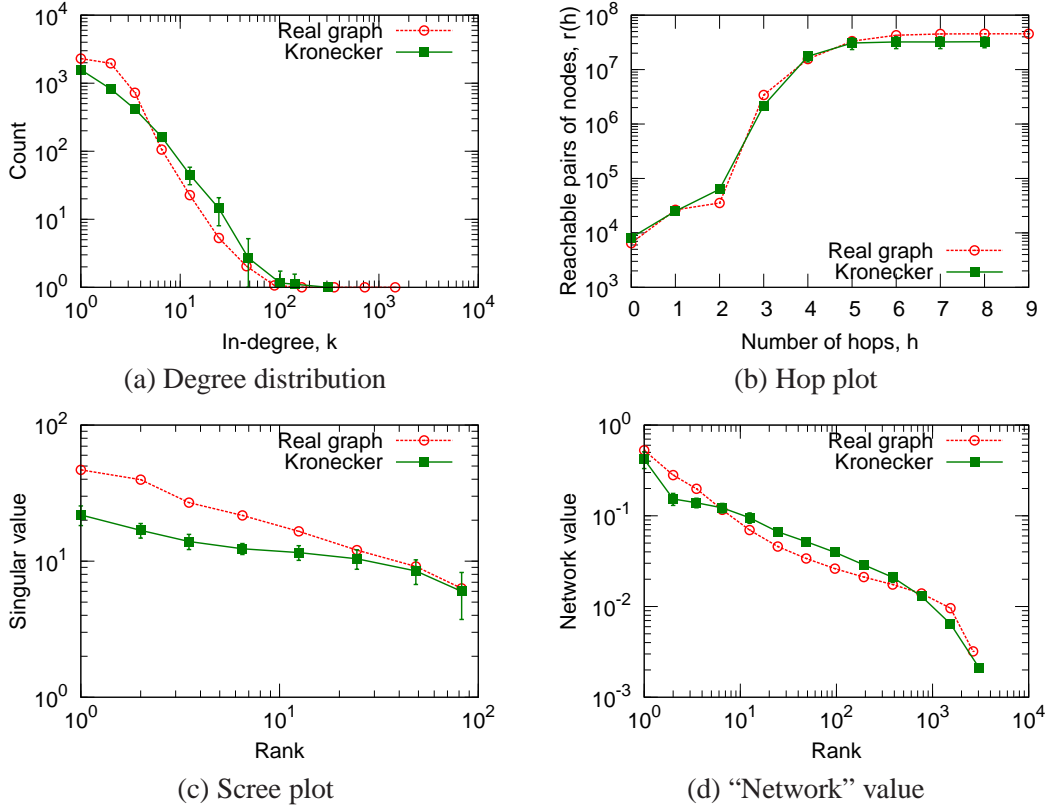


Figure 17: *Autonomous Systems (AS-ROUTEVIEWS)*: Overlaid patterns of real graph and the fitted Kronecker graph. Notice that the fitted Kronecker graph matches patterns of the real graph while using only four parameters (2×2 initiator matrix).

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 graph 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 are 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 graph 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 2×2 . 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 to 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 the good agreement with the real graph while using only four parameters.

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 $\hat{\Theta}$, 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

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
AS-ROUTEVIEWS			26,467	6,474	

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.

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. The 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 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 $l(\hat{\theta})$ at MLE. Similarly, notice (column N_1^k) that while 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 AS-ROUTEVIEWS network. This shows that KRONFIT is actually fitting the graph well and 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 AS-ROUTEVIEWS graph.

Last, comparing the log-likelihood at the 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

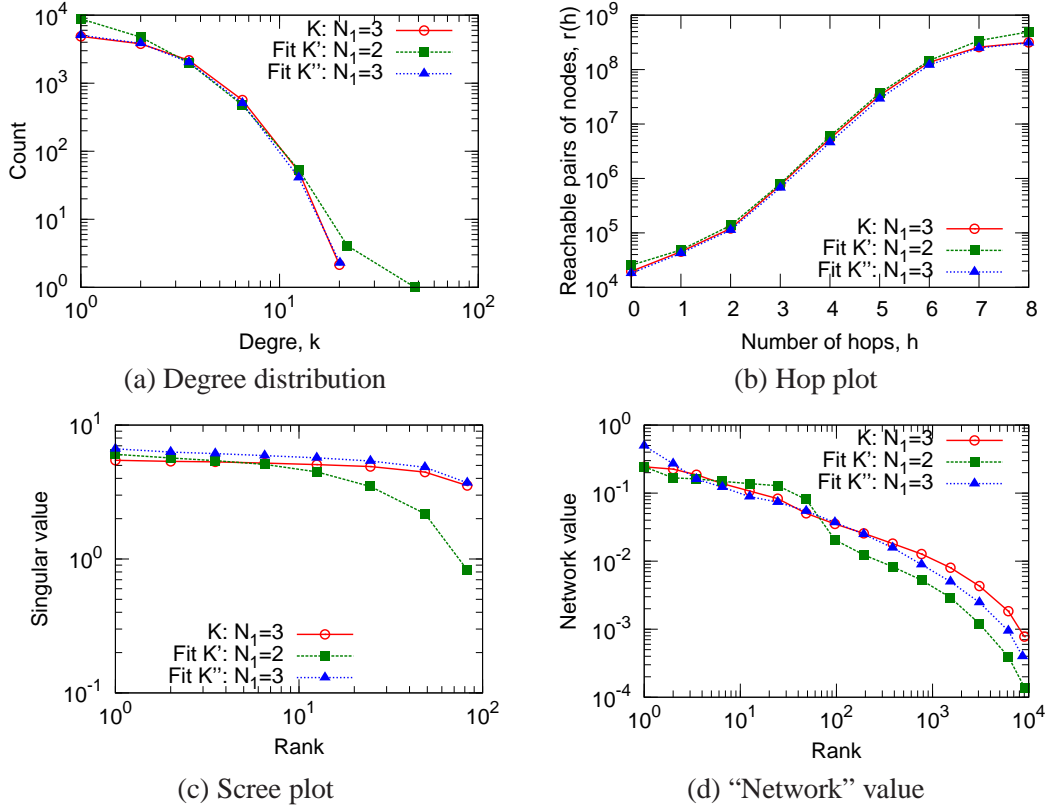


Figure 18: *3 by 3 Stochastic Kronecker graphs*: Given a Stochastic Kronecker graphs G generated from $N_1 = 3$ (red curve), we fit a Kronecker graph K' with $N'_1 = 2$ (green) and K'' with $N''_1 = 3$ (blue). Not surprisingly K'' fits the properties of K perfectly as the model is the of same complexity. On the other hand K' has only 4 parameters (instead of 9 as in K and K'') and still fits well.

(number of parameters) is so small that overfitting is not a concern. 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 graphs 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

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

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

too simple model still fits the data well. In general we observe empirically that by increasing the size of the 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 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 AS-ROUTEVIEWS network. Notice the 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 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_1^k 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 graphs 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, 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

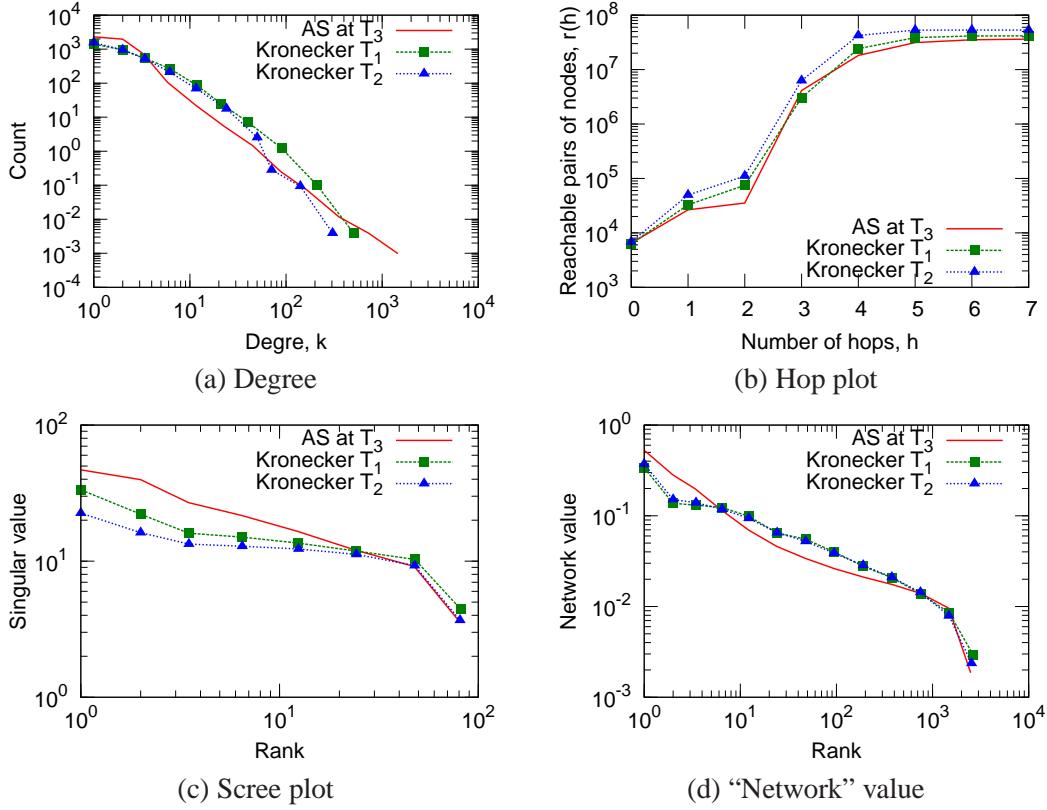


Figure 19: *Autonomous systems network over time* (AS-ROUTEVIEWS): Overlaid patterns of real AS-ROUTEVIEWS network at time T_3 and the Kronecker graphs with parameters estimated from AS-ROUTEVIEWS at time T_1 and T_2 . Notice good fits which means that parameters estimated on historic snapshots can be used to estimate the graph in the future.

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. They are available for download at <http://snap.stanford.edu>.

For each dataset we started gradient descent from a random point (random initiator matrix) and ran 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 matrices $\hat{\Theta}$ seem to have almost universal structure with a large value in the top left entry, a very small 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 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 sepa-

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

Table 4: Results of parameter estimation for 20 different networks. Table 5 gives the description and basic properties of the above network datasets. Networks are available for download at <http://snap.stanford.edu>.

rately 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.)

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 graphs with just 4 parameters we match the scree plot, degree distributions, triangle participation, hop plot and network values.

Given the previous experiments from the Autonomous systems graph 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 2×2 model as a baseline and estimate the log-likelihood at the MLE for larger initiator matrices. Again, models with more parameters tend to fit better. However, sometimes due to zero-padding of a graph adjacency matrix they actually have lower log-likelihood (as for example seen in Table 2).

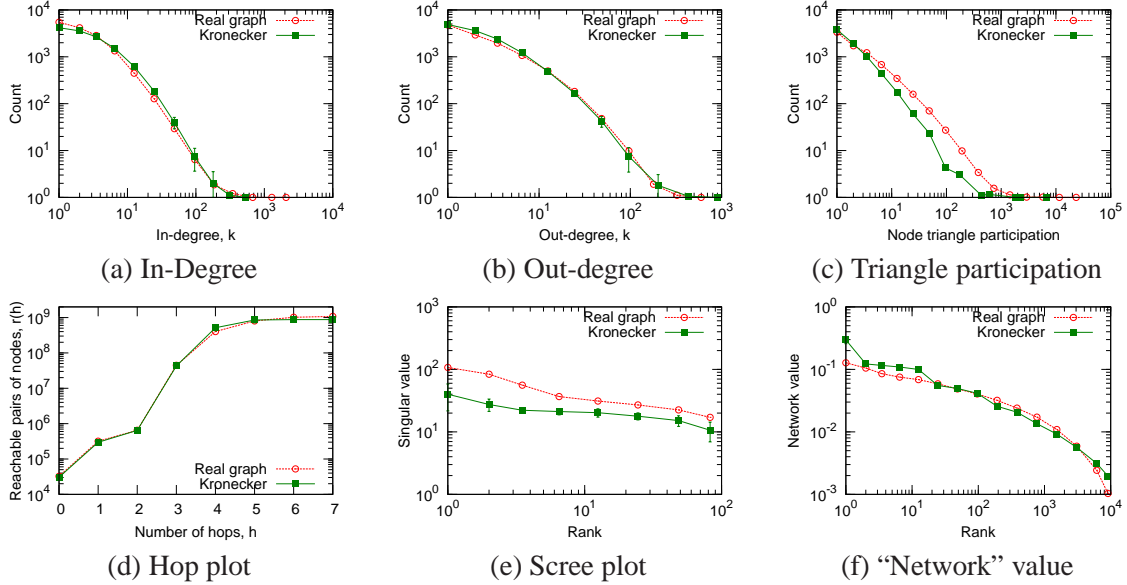


Figure 20: *Blog network* (BLOG-NAT06ALL): Overlaid patterns of real network and the estimated Kronecker graph using 4 parameters (2×2 initiator matrix). Notice that the Kronecker graph matches all properties of the real network.

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 the network. We plot wall-clock time vs. size of the graph. The dashed line gives a linear fit to the data points.

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 the 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 2×2 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.

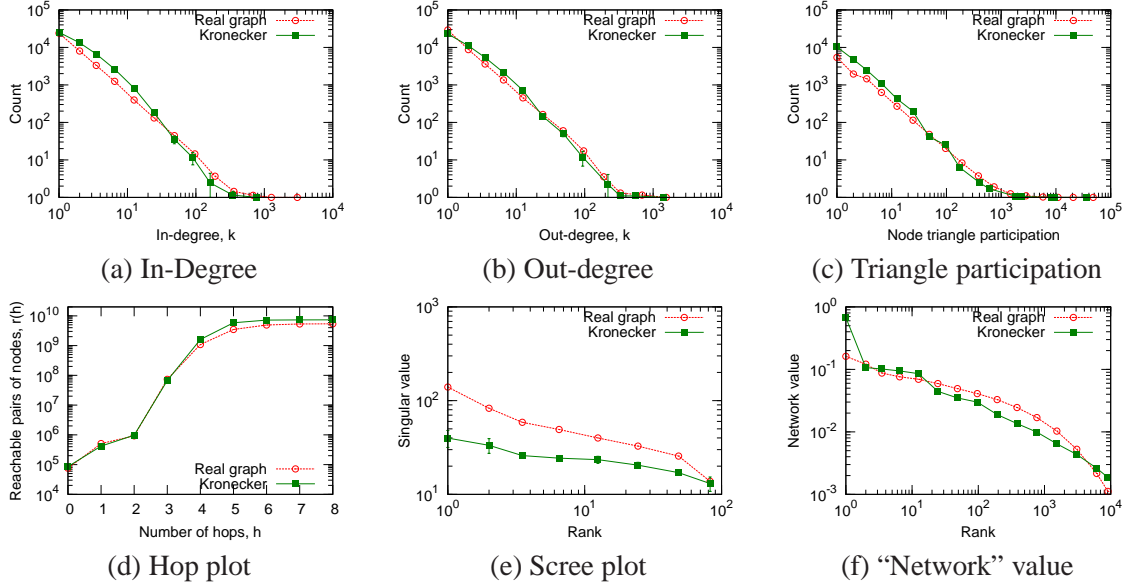


Figure 21: EPINIONS *who-trusts-whom* social network: Overlaid patterns of real network and the fitted Kronecker graph using only 4 parameters (2×2 initiator matrix). Again, the synthetic Kronecker graph matches all the properties of the real network.

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” or phase transition (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 2×2 initiator ($N_1 = 2$) fits well the properties of real-world networks. Moreover, given a 2×2 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 Θ 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.

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 and few edges crossing them (Leskovec, 2009). 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

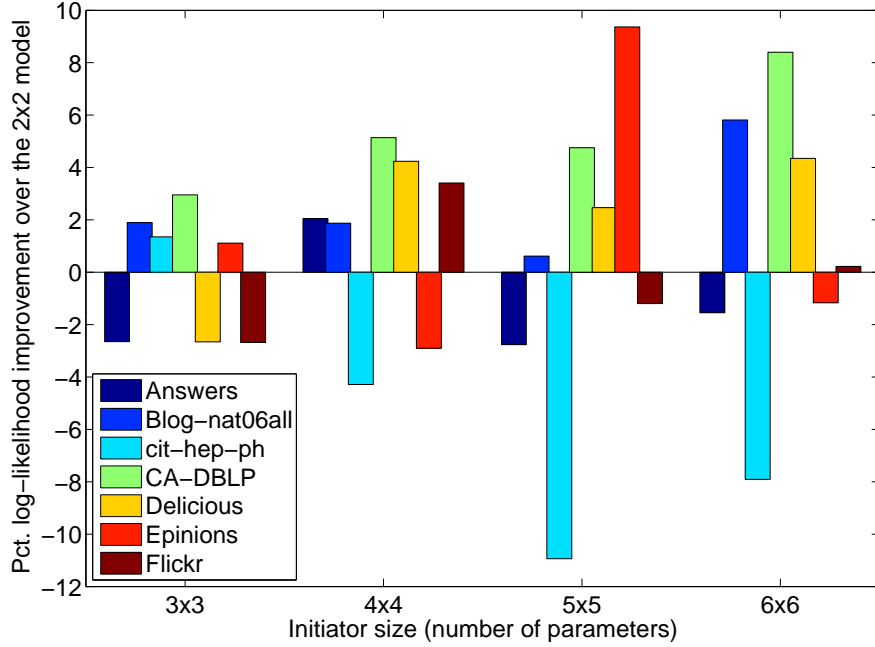


Figure 22: Percent improvement in log-likelihood over the 2×2 model as we increase the model complexity (size of initiator matrix). In general larger initiator matrices that have more degrees of freedom help improving the fit of the model.

$\hat{\Theta}$ also holds when fitting 3×3 or 4×4 models. Always the top left element is the largest and then the values on the diagonal decay faster than off the diagonal (Leskovec, 2009).

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 ($b, c > d$) (Leskovec, 2009). 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 *nested core-periphery* (Leskovec et al., 2008a,b) 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 3×3 or 4×4 initiator matrix. The diagonal elements have large but decreasing values with off diagonal elements following same decreasing pattern.

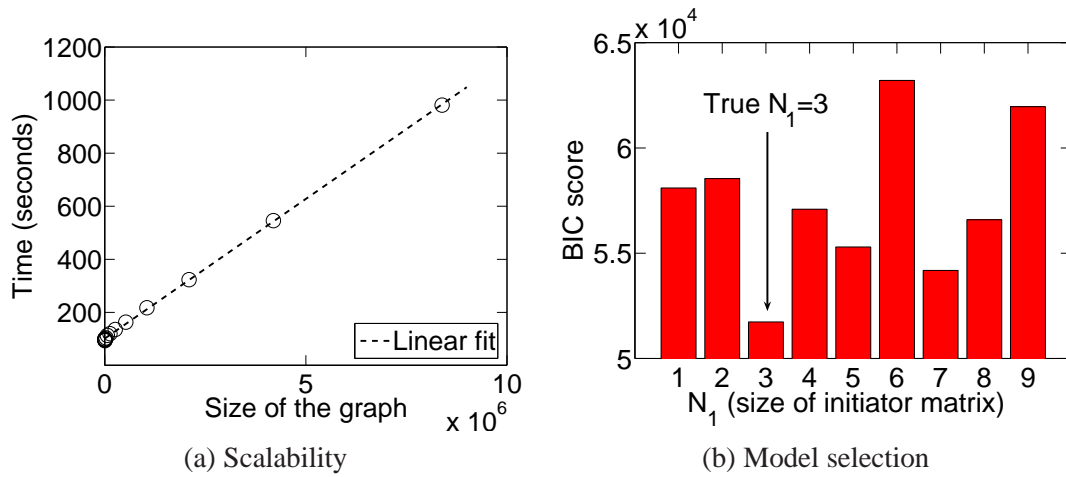


Figure 23: (a) Processor time to sample 1 million gradients as the graph grows. Notice the algorithm scales linearly with the graph size. (b) BIC score for model selection.

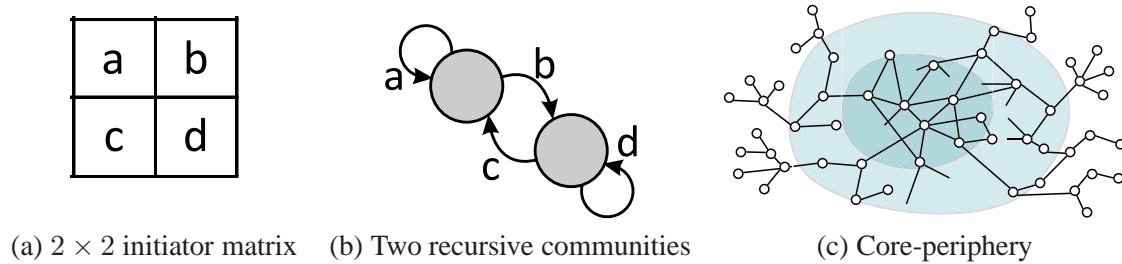


Figure 24: 2×2 Kronecker initiator matrix (a) can be thought of as two communities where there are a and d edges inside each of the communities and b and c edges crossing the two communities as illustrated in (b). Each community can then be recursively divided using the same pattern. (c) The onion like core-periphery structure where the network gets denser and denser as we move towards the center of the network.

One of the implications of this is that networks do not break nicely into hierarchically organized sets of communities that lend themselves to graph partitioning and community detection 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 based on graph partitioning) about the clustering and community structure of large real-world 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 Θ 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*: Estimated parameters give insight into the global network and community structure of the network.
- *Forecasting*: As we demonstrated one can obtain Θ from a graph G_t at time t such that G is mimicked by $\Theta^{[k]}$. Then Θ 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 .

Future work could include extensions of Kronecker graphs to evolving networks. We envision formulating a dynamic Bayesian network with first order Markov dependencies, where parameter matrix at time t depends on the graph G_t at current time t and the parameter matrix at time $t - 1$. Given a series of network snapshots one would then aim to estimate initiator matrices at individual time steps and the parameters of the model governing the evolution of the initiator matrix. We

expect that based on the evolution of initiator matrix one would gain greater insight in the evolution of large networks.

Second direction for future work is to explore connections between Kronecker graphs and Random Dot Product graphs (Young and Scheinerman, 2007; Nickel, 2008). This also nicely connects with the “attribute view” of Kronecker graphs as described in Section 3.5. It would be interesting to design methods to estimate the individual node attribute values as well as the attribute-attribute similarity matrix (i.e., the initiator matrix). As for some networks node attributes are already given one could then try to infer “hidden” or missing node attribute values and this way gain insight into individual nodes as well as individual edge formations. Moreover, this would be interesting as one could further evaluate how realistic is the “attribute view” of Kronecker graphs.

Last, we also mention possible extensions of Kronecker graphs for modeling weighted and labeled networks. Currently Stochastic Kronecker graphs use a Bernoulli edge generation model, i.e., an entry of big matrix \mathcal{P} encodes the parameter of a Bernoulli coin. In similar spirit one could consider entries of \mathcal{P} to encode parameters of different edge generative processes. For example, to generate networks with weights on edges an entry of \mathcal{P} could encode the parameter of an exponential distribution, or in case of labeled networks one could use several initiator matrices in parallel and this way encode parameters of a multinomial distribution over different node attribute values.

References

- Network data. <http://www-personal.umich.edu/~mejn/netdata>, July 16, 2007.
- R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.
- R. Albert, H. Jeong, and A.-L. Barabási. Diameter of the world-wide web. *Nature*, 401:130–131, September 1999.
- L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54, 2006. ISBN 1-59593-339-5.
- P. Bak. *How Nature Works: The Science of Self-Organized Criticality*. Springer, September 1996.
- A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
- A.-L. Barabási, E. Ravasz, and T. Vicsek. Deterministic scale-free networks. *Physica A*, 299: 559–564, 2001.
- I. Bezáková, A. Kalai, and R. Santhanam. Graph model selection using maximum likelihood. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 105–112, 2006.
- Z. Bi, C. Faloutsos, and F. Korn. The DGX distribution for mining massive, skewed data. In *KDD '01: Proceedings of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 17–26, 2001.

- A. Blum, H. Chan, and M. Rwebangira. A random-surfer web-graph model. In *ANALCO '06: Proceedings of the 3rd Workshop on Analytic Algorithmics and Combinatorics*, 2006.
- S. P. Borgatti and M. G. Everett. Models of core/periphery structures. *Social Networks*, 21(4): 375–395, 2000.
- A. Bottreau and Y. Metivier. Some remarks on the kronecker product of graphs. *Information Processing Letters*, 68(2):55 – 61, 1998.
- A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web: experiments and models. In *WWW '00: Proceedings of the 9th international conference on World Wide Web*, 2000.
- T. Bu and D. F. Towsley. On distinguishing between internet power law topology generators. In *INFOCOM*, 2002.
- C. T. Butts. Permutation models for relational data. (tech. rep. mbs 05-02, Univ. of California, Irvine, 2005.
- J. M. Carlson and J. Doyle. Highly optimized tolerance: a mechanism for power laws in designed systems. *Physical Review E*, 60(2):1412–1427, 1999.
- D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-mat: A recursive model for graph mining. In *SDM '04: SIAM Conference on Data Mining*, 2004.
- T. Chow. The Q-spectrum and spanning trees of tensor products of bipartite graphs. *Proc. Amer. Math. Soc.*, 125:3155–3161, 1997.
- F. R. K. Chung and L. Lu. *Complex Graphs and Networks*, volume 107 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 2006.
- F. R. K. Chung, L. Lu, and V. Vu. Eigenvalues of random power law graphs. *Annals of Combinatorics*, 7:21–33, 2003.
- A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *ArXiv*, arXiv:0706.1062, Jun 2007.
- A. Clauset, C. Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.
- V. Colizza, A. Flammini, M. A. Serrano, and A. Vespignani. Characterization and modeling of protein protein interaction networks. *Physica A Statistical Mechanics and its Applications*, 352: 1–27, 2005.
- M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: evidence and possible causes. *IEEE /ACM Transactions on Networking*, 5(6):835–846, 1997.
- S. Dill, R. Kumar, K. S. Mccurley, S. Rajagopalan, D. Sivakumar, and A. Tomkins. Self-similarity in the web. *ACM Trans. Interet Technology*, 2(3):205–223, 2002. ISSN 1533-5399.

- S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes. Pseudofractal scale-free web. *Physical Review E*, 65(6):066122, Jun 2002.
- B. Efron. Defining the curvature of a statistical problem (with applications to second order efficiency). *Ann. Statist.*, 3(6):1189–1242, 1975.
- P. Erdős and A. Rényi. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Science*, 5:17–67, 1960.
- A. Fabrikant, E. Koutsoupias, and C. H. Papadimitriou. Heuristically optimized trade-offs: A new paradigm for power laws in the internet. In *ICALP '02: Proceedings of the 29th International Colloquium on Automata, Languages, and Programming*, volume 2380, 2002.
- M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 251–262, 1999.
- I. Farkas, I. Derényi, A.-L. Barabási, and T. Vicsek. Spectra of “real-world” graphs: beyond the semicircle law. *Physical Review E*, 64(026704), 2001.
- A. D. Flaxman, A. M. Frieze, and J. Vera. A geometric preferential attachment model of networks II. In *WAW '07: Proceedings of the 5th Workshop On Algorithms And Models For The Web-Graph*, pages 41–55, 2007.
- D. Gamerman. *Markov chain Monte Carlo, Stochastic simulation for Bayesian inference*. Chapman & Hall, London, 1997.
- J. Gehrke, P. Ginsparg, and J. M. Kleinberg. Overview of the 2003 kdd cup. *SIGKDD Explorations*, 5(2):149–151, 2003.
- A. Gelman, J. B. Carlin, H. S. Stern, and D.B. Rubin. *Bayesian Data Analysis, Second Edition*. Chapman & Hall, London, July 2003. ISBN 158488388X.
- R. H. Hammack. Proof of a conjecture concerning the direct product of bipartite graphs. *European Journal of Combinatorics*, 30(5):1114 – 1118, 2009. ISSN 0195-6698. Part Special Issue on Metric Graph Theory.
- P. Holme. Core-periphery organization of complex networks. *Physical Review E*, 72:046111, 2005.
- W. Imrich. Factoring cardinal product graphs in polynomial time. *Discrete Mathematics*, 192(1-3): 119–144, 1998. ISSN 0012-365X.
- W. Imrich and S. Klavžar. *Product Graphs: Structure and Recognition*. Wiley, 2000. ISBN 1-4020-0489-3.
- J. M. Kleinberg. The small-world phenomenon: an algorithmic perspective. Technical Report 99-1776, Cornell Computer Science Department, 1999.
- J. M. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web as a graph: Measurements, models and methods. In *COCOON '99: Proceedings of the International Conference on Combinatorics and Computing*, 1999.

- K. Klemm and V. M. Eguíluz. Highly clustered scale-free networks. *Phys. Rev. E*, 65(3):036123, Feb 2002.
- S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.
- R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 57, 2000.
- R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 611–617, 2006.
- S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scale knowledge bases from the web. In *Proceedings of the 25th VLDB Conference*, Edinburgh, Scotland, 1999.
- A. N. Langville and W. J. Stewart. The Kronecker product and stochastic automata networks. *Journal of Computation and Applied Mathematics*, 167:429–447, 2004.
- J. Leskovec. Networks, communities and kronecker products. In *CNIKM '09: Complex Networks in Information and Knowledge Management*, 2009.
- J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, 2007.
- J. Leskovec, D. Chakrabarti, J. M. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In *PKDD '05: Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 133–145, 2005a.
- J. Leskovec, J. M. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD '05: Proceeding of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, 2005b.
- J. Leskovec, J. M. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007a.
- J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst. Cascading behavior in large blog graphs. In *SDM '07: Proceedings of the SIAM Conference on Data Mining*, 2007b.
- J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW '08: Proceedings of the 17th International Conference on World Wide Web*, 2008a.
- J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *ArXiv*, arXiv:0810.1355, Oct 2008b.
- C. F. Van Loan. The ubiquitous Kronecker product. *Journal of Computation and Applied Mathematics*, 123:85–100, 2000.

- M. Mahdian and Y. Xu. Stochastic kronecker graphs. In *WAW '07: Proceedings of the 5th Workshop On Algorithms And Models For The Web-Graph*, pages 179–186, 2007.
- M. Mihail and C. H. Papadimitriou. On the eigenvalue power law. In *RANDOM*, pages 254–262, 2002.
- S. Milgram. The small-world problem. *Psychology Today*, 2:60–67, 1967.
- C. L. M. Nickel. Random dot product graphs: A model for social networks. Ph.D. Thesis, Dept. of Applied Mathematics and Statistics, Johns Hopkins University, 2008.
- C. R. Palmer, P. B. Gibbons, and C. Faloutsos. Anf: a fast and scalable tool for data mining in massive graphs. In *KDD '02: Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 81–90, 2002.
- D. M. Pennock, G. W. Flake, S. Lawrence, E. J. Glover, and C. L. Giles. Winners don't take all: Characterizing the competition for links on the Web. *Proceedings of the National Academy of Sciences*, 99(8):5207–5211, 2002.
- V. V. Petrov. *Limit Theorems of Probability Theory*. Oxford University Press, 1995.
- E. Ravasz and A.-L. Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67(2):026112, 2003.
- E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555, 2002.
- S. Redner. How popular is your paper? an empirical study of the citation distribution. *European Physical Journal B*, 4:131–134, 1998.
- M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *ISWC*, 2003.
- M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1):50–57, Jan/Feb 2002.
- J. Rissanen. Modelling by the shortest data description. *Automatica*, 14:465–471, 1978.
- RouteViews. University of Oregon Route Views Project. Online data and reports. <http://www.routeviews.org>, 1997.
- M. Sales-Pardo, R. Guimera, A. A. Moreira, and L. A. Amaral. Extracting the hierarchical organization of complex systems. *Proceedings of the National Academy of Sciences*, 104(39):15224–15229, September 2007.
- G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.
- G. Siganos, S. L. Tauro, and M. Faloutsos. Jellyfish: A conceptual model for the as internet topology. *Journal of Communications and Networks*, 8:339–350, 2006.

- R. Sole and B. Goodwin. *Signs of Life: How Complexity Pervades Biology*. Perseus Books Group, New York, NY, 2000.
- S. L. Tauro, C. Palmer, G. Siganos, and M. Faloutsos. A simple conceptual model for the internet topology. In *GLOBECOM '01: Global Telecommunications Conference*, volume 3, pages 1667 – 1671, 2001.
- C. E. Tsourakakis. Fast counting of triangles in large real networks, without counting: Algorithms and laws. In *ICDM '08 : IEEE International Conference on Data Mining*, 2008.
- A. Vázquez. Growing network with local rules: Preferential attachment, clustering hierarchy, and degree correlations. *Phys. Rev. E*, 67(5):056104, May 2003.
- S. Wasserman and P. Pattison. Logit models and logistic regressions for social networks. *Psychometrika*, 60:401–425, 1996.
- D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.
- B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, December 1988.
- P. M. Weichsel. The kronecker product of graphs. In *American Mathematical Society*, volume 13, pages 37–52, 1962.
- J. Winick and S. Jamin. Inet-3.0: Internet Topology Generator. Technical Report CSE-TR-456-02, University of Michigan, Ann Arbor, 2002.
- C. Wiuf, M. Brameier, O. Hagberg, and M. P. Stumpf. A likelihood approach to analysis of network data. *Proceedings of the National Academy of Sciences*, 103(20):7566–7570, 2006.
- Stephen J. Young and Edward R. Scheinerman. Random dot product graph models for social networks. In *WAW '07: Proceedings of the 5th Workshop On Algorithms And Models For The Web-Graph*, pages 138–149, 2007.
- E. Zheleva, H. Sharara, and L. Getoor. Co-evolution of social and affiliation networks. In *KDD*, pages 1007–1016, 2009.

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. Most of the networks are available for download from <http://snap.stanford.edu>.

Network	N	E	N_c	N_c/N	\bar{C}	D	\bar{D}	Description
Social networks								
ANSWERS	598,314	1,834,200	488,484	0.82	0.11	22	5.72	Yahoo! Answers social network (Leskovec et al., 2008a)
DELICIOUS	205,282	436,735	147,567	0.72	0.3	24	6.28	del.icio.us social network (Leskovec et al., 2008a)
EMAIL-INSIDE	986	32,128	986	1.00	0.45	7	2.6	European research organization email network (Leskovec et al., 2007a)
EPINIONS	75,879	508,837	75,877	1.00	0.26	15	4.27	Who-trusts-whom graph of epinions.com (Richardson et al., 2003)
FLICKR	584,207	3,555,115	404,733	0.69	0.4	18	5.42	Flickr photo sharing social network (Kumar et al., 2006)
Information (citation) networks								
BLOG-NAT05-6M	31,600	271,377	29,150	0.92	0.24	10	3.4	Blog-to-blog citation network (6 months of data) (Leskovec et al., 2007b)
BLOG-NAT06ALL	32,443	318,815	32,384	1.00	0.2	18	3.94	Blog-to-blog citation network (1 year of data) (Leskovec et al., 2007b)
CIT-HEP-PH	30,567	348,721	34,401	1.13	0.3	14	4.33	Citation network of ArXiv hep-th papers (Gehrke et al., 2003)
CIT-HEP-TH	27,770	352,807	27,400	0.99	0.33	15	4.2	Citations network of ArXiv hep-ph papers (Gehrke et al., 2003)
Collaboration networks								
CA-DBLP	425,957	2,696,489	317,080	0.74	0.73	23	6.75	DBLP co-authorship network (Backstrom et al., 2006)
CA-GR-QC	5,242	28,980	4,158	0.79	0.66	17	6.1	Co-authorship network in gr-qc category of ArXiv (Leskovec et al., 2005b)
CA-HEP-PH	12,008	237,010	11,204	0.93	0.69	13	4.71	Co-authorship network in hep-ph category of ArXiv (Leskovec et al., 2005b)
CA-HEP-TH	9,877	51,971	8,638	0.87	0.58	18	5.96	Co-authorship network in hep-th category of ArXiv (Leskovec et al., 2005b)
Web graphs								
WEB-NOTREDAME	325,729	1,497,134	325,729	1.00	0.47	46	7.22	Web graph of University of Notre Dame (Albert et al., 1999)
Internet networks								
AS-NEWMAN	22,963	96,872	22,963	1.00	0.35	11	3.83	AS graph from Newman (new, July 16, 2007)
AS-ROUTEVIEWS	6,474	26,467	6,474	1.00	0.4	9	3.72	AS from Oregon Route View (Leskovec et al., 2005b)
GNUTELLA-25	22,687	54,705	22,663	1.00	0.01	11	5.57	Gnutella P2P network on 3/25 2000 (Ripeanu et al., 2002)
GNUTELLA-30	36,682	88,328	36,646	1.00	0.01	11	5.75	Gnutella P2P network on 3/30 2000 (Ripeanu et al., 2002)
Bi-partite networks								
ATP-GR-QC	19,177	26,169	14,832	0.77	0	35	11.08	Affiliation network of gr-qc category in ArXiv (Leskovec et al., 2007b)
Biological networks								
BIO-PROTEINS	4,626	29,602	4,626	1.00	0.12	12	4.24	Yeast protein interaction network (Colizza et al., 2005)

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 \bar{C} ; diameter D , and average path length \bar{D} . Networks are available for download at <http://snap.stanford.edu>.