

SPECTRAL PROPERTIES AND GENERATION OF REALISTIC NETWORKS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Nicole Eikmeier

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2019

Purdue University

West Lafayette, Indiana

**THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL**

Dr. David F. Gleich, Chair

Department of Computer Science

Dr. Jianlin Xia

Department of Mathematics

Dr. Greg Buzzard

Department of Mathematics

Dr. Jennifer Neville

Department of Computer Science

Approved by:

Dr. Greg Buzzard

Head of the Department of Mathematics

ACKNOWLEDGMENTS

Thank you to my PhD advisor David Gleich for research ideas, patient assistance in programming, walking through my proofs, all of those letters of recommendation over the years, and helping enormously to improve my writing. His very generous funding allowed me to focus on my research, and sent me to conferences around the world in which I had the opportunity to share and learn from others in my field.

Thank you to the other members of my thesis committee, Jianlin Xia, Greg Buzzard, and Jennifer Neville. I appreciate their time in giving feedback on earlier versions of my work.

I'd like to thank the past and present student members of my research group at Purdue for all their support big and small: Kyle Kloster, Yangyang Hou, Tao Wu, Huda Nassar, Nate Veldt, Varun Vasudevan, Bryan Rainey, Meng Liu, Yanfei Ren, Rania Ibrahim, Caitlin Kennedy, Charles Colley, and Cameron Ruggles.

Thank you to my collaborators and co-authors during my time as a graduate student for including me in your great work. David Gleich, Arjun Ramani, Anthony Bonato, Ed Zelnio, Tenzing Joshi, and Brian Quiter. Also to all the members of the Applied Nuclear Physics group at Lawrence Berkeley National Lab, and the folks that run the Automatic Target Recognition Center at Wright State University. I value the variety of people I've been fortunate to work with and learn from.

Many staff at Purdue made life as a grad student easier. In particular, Rebecca Lank, Shannon Cassidy, Kristi Stroud, and Jennifer Deno.

Outside of my research a number of students offered me moral support and friendship. Thank you to Brittney Miller, Phil Husom, RB McGee, Huda Nassar, Kyle Kloster, Kate Brubaker and Rachel Lynn. Thank you to every single person who has ensured the Purdue Math Department is a supportive and welcoming community, and in particular Greg Buzzard, Steve Bell, Donatella Danielli and David Goldberg. Thank you also to all past and current cabinet members of the Purdue AWM chapter, a group that has ensured I never felt alone in this journey.

Finally, thank you to my family. My mom, Lisa Rutt, and my grandparents, Jay Wright and Barbara Wright told me that I can accomplish anything and never doubted it for a second. Thank you to my husband and best friend Mike Eikmeier, who made so many sacrifices to help me get here. He has given me constant encouragement, from when I first considered applying for graduate school through the difficult realities of earning a PhD. It is impossible to put into words how grateful I am to have him by my side.

TABLE OF CONTENTS

	Page
LIST OF TABLES	vi
LIST OF FIGURES	vii
ABSTRACT	viii
1 INTRODUCTION	1
2 BACKGROUND	3
2.1 Graphs and Matrices	3
2.2 Graph Modeling	3
2.2.1 Sampling Graphs from Probability Matrices	4
2.2.2 Preferential Attachment	5
2.3 Higher Order Features in Graphs	6
2.4 Clustering	6
2.5 Power-laws	7
3 EMPIRICAL STUDY OF REAL WORLD NETWORKS	8
3.1 Data Sets and Models	10
3.1.1 Real-world data	10
3.1.2 Models	11
3.2 The details of our fitting methodology	13
3.2.1 Computing degrees, singular values, and eigenvalues	13
3.2.2 Fitting power-law parameters	14
3.2.3 Exceptions to our methodology	14
3.2.4 Limitations	14
3.3 Presence of Power-Laws	15
3.3.1 Many classes of networks have power-laws	15
3.3.2 Relationships between power-laws	17
3.4 Analysis of the Power-Laws	18
3.4.1 Characteristics of the power-laws	19
3.4.2 Consistency across network samples	21
3.4.3 Behavior of the cut-off	22
3.5 Large Networks	23
3.6 Summary	24
4 THE HYPERKRON GRAPH MODEL	26
4.1 Kronecker products of tensors	27
4.2 The Symmetric HyperKron Model with Triangles	29
4.3 Efficient Generation	29
4.3.1 A small number of Erdős-Rényi blocks	31
4.3.2 Multiplication tables and HyperKron tensors	32
4.3.3 Grass-Hopping Kronecker Tensors	33
4.3.4 Morton Codes	34
4.3.5 Runtime performance	36
4.4 HyperKron Properties	37
4.4.1 Summation Formulas	38

	Page
4.4.2 Exact Expectation of Edges	38
4.4.3 Duplicate Edges Motif	39
4.4.4 Approximate Expectation of Edges	40
4.4.5 Extra Formulas	43
4.4.6 Non-trivial Clustering	44
4.5 Fitting HyperKron to Real Data	45
4.5.1 Clustering Coefficients	46
4.5.2 Skewed Degrees	47
4.5.3 Improvements to the model	47
4.6 Model flexibility	52
4.7 Summary	53
5 THE TRIANGLE GENERALIZED PREFERENTIAL ATTACHMENT MODEL	54
5.1 Related Work	55
5.1.1 Generalized Preferential Attachment	55
5.1.2 Triad Formation	56
5.2 Eigenvalue Power-law in GPA	57
5.3 TGPA	69
5.3.1 TGPA(p, q)	69
5.3.2 TGPA(p_t, r_t, q_t)	70
5.4 Analysis of TGPA(p, q)	70
5.5 Analysis of TGPA(p_t, r_t, q_t)	83
5.5.1 Recursive relation for $m_{k,t}$	83
5.5.2 Power-law in TGPA(p_t, r_t, q_t)	85
5.6 Significant Clustering coefficients	87
5.7 The Eigenvalue Power-law is robust	88
5.8 Summary	89
6 CONCLUSIONS AND FUTURE WORK	92
REFERENCES	95

LIST OF TABLES

Table	Page
3.1 The types of networks we use in our studies, along with a rough order of magnitude of the sizes in vertices	12
3.2 Presence of power-laws	16
3.3 Conditional probabilities that a power-law distribution in one feature gives a power-law distribution in another	18
3.4 Fitting a power-law distribution to large graphs	24
4.1 Fitting real world data to the HyperKron model	48
4.2 Higher order global clustering coefficients of the HyperKron model	49
5.1 Clustering coefficients in TGPA model	87

LIST OF FIGURES

Figure	Page
3.1 Exponents of statistically significant power-law distributions	19
3.2 Relationship between power-law in degrees and eigenvalues of the Laplacian	20
3.3 Relationship between power-law in degrees and spectra	21
3.4 Density estimates of the power-law exponent	22
3.5 Network size versus size of power-law tail	22
4.1 An example graph generated with the HyperKron model	30
4.2 Pseudocode for fast hyperedge sampling algorithm on a HyperKron model	31
4.3 3-dimensional Morton Codes	36
4.4 Time taken to generate hyperedges for a HyperKron model	37
4.5 Two hyperedges which share an edge	41
4.6 Expected number of edges in the HyperKron model	42
4.7 Global clustering coefficients of the HyperKron model	45
4.8 Global clustering coefficients of the Kronecker model	46
4.9 HyperKron preserves highly skewed degree distribution	49
4.10 Improved degree distribution in HyperKron	51
4.11 HyperKron fit to yeast network	52
5.1 Example TGPA networks compared to existing models	57
5.2 Power-law in spectra is more robust to sampling	90
5.3 Forest Fire Sampling graphs generated using the preferential attachment model	91
5.4 Effects of Sampling a Real World Network	91

ABSTRACT

Eikmeier, Nicole PhD, Purdue University, August 2019. Spectral Properties and Generation of Realistic Networks. Major Professor: David F. Gleich.

Picture the life of a modern person in the western world: They wake up in the morning and check their social networking sites; they drive to work on roads that connect cities to each other; they make phone calls, send emails and messages to colleagues, friends, and family around the world; they use electricity flowing through power-lines; they browse the Internet, searching for information. All of these typical daily activities rely on the structure of *networks*. A network, in this case, is a set of nodes (people, web pages, etc) connected by edges (physical connection, collaboration, etc). The term graph is sometimes used to represent a more abstract structure - but here we use the terms graph and network interchangeably. The field of network analysis concerns studying and understanding networks in order to solve problems in the world around us. Graph models are used in conjunction with the study of real-world networks. They are used to study how well an algorithm may do on a real-world network, and for testing properties that may further produce faster algorithms. The first piece of this dissertation is an experimental study which explores features of real data, specifically power-law distributions in degrees and spectra. In addition to a comparison between features of real data to existing results in the literature, this study resulted in a hypothesis on power-law structure in spectra of real-world networks being more reliable than that in the degrees. The theoretical contributions of this dissertation are focused primarily on generating realistic networks through existing and novel graph models. The two graph models presented are called *HyperKron* and the *Triangle Generalized Preferential Attachment model*. Both of the models incorporate *higher-order structure* - leading to more sophisticated properties not examined in traditional models. We use the second of our models to further validate the hypothesis on power-laws in the spectra. Due to the structure of our model, we show that the power-law in the spectra is more resilient to sub-sampling. This gives some explanation for why we see power-laws more frequently in the spectra in real world data.

1. INTRODUCTION

The word *Network* is used widely across many fields, and the popularity of the word is growing. According to Google’s ngram tool [NGr], the use of the word network nearly doubled between the years 1980 and 2000. By network, we (and many others) mean a set of relationships between objects or ideas. For example, a physical network of telephone poles connected by wires. Or a network of people, connected through work relationships.

As the use of computers has grown, so has their storage and computation abilities. Businesses are incentivized to record as much data as possible, in order to analyze it for optimizing profits in the future. For example, a social networking website may run analysis on their stored data in order to optimize their advertisement placements, thus making the company more money overall. Much of this type of work falls into the field of *network analysis* - which is just as it sounds. From a high level view, there are a few primary goals of the research in network analysis: 1) Designing algorithms which can run efficiently on very large network data; 2) Understanding features, behavior, and structure of networks which arise from the real world. These two goals are not independent.

Modeling networks (also called *graphs*) is its own subfield of network analysis in which we aim to construct network models which resemble real world data. Graph modeling works towards *both* of the goals mentioned above. One of the things that makes a model useful is the ability to generate large amounts of synthetic data quickly. This is important where real data is expensive and limited. The model’s data is then used for testing algorithms, in order to predict how they will behave on the real data. Furthermore, developing models which obey the properties which occur naturally provide more realistic studies when using synthetic data in place of real data, since network models are widely used from industry to medical applications. On the flipside, studying the properties of the data generated by models may lead to insights into the structure of the real data. Network models are also used in statistical hypothesis testing [Moreno and Neville, 2013], and for benchmarking high performance computers [Murphy et al., 2010].

As the techniques for analyzing networks have improved, so has the study of more complex features of networks. If in the past we were concerned with studying groupings of two objects, now we are interested in the groupings of three or more objects. Understanding and replicating these more complex features can be described as higher-order network analysis. Further focus and study of higher-order features may bring about better understanding of real world data, leading to more accurate network models [Lambiotte et al., 2019].

When the work of this thesis began, it started with a question in epidemics. Consider a network of people (nodes) and their interactions (edges). If disease breaks out in one or more people, we may ask how to prevent an epidemic from breaking out in the network. One solution to this problem is to find the people with the biggest influence in the network and quarantine them (remove them and their edges from the network). Phrased mathematically, this is finding the nodes or edges of a network with the largest number of length- k walks through them. This problem is easily solved on paper by computing the Hadamard, or entry-wise, product of \mathbf{A}^k and \mathbf{A} , and finding the largest entries in the resulting matrix. But as A becomes large, computing k products of \mathbf{A} is not straightforward.

There are two strategies then for this problem. First, we could do the computation of $B = \text{Had}(\mathbf{A}^k, \mathbf{A})$ anyway, attempting to be as memory and time efficient as possible. The second option is to find approximation algorithms to estimate the largest entries of B . One approximation algorithm involves computing the first components of the singular value decomposition of \mathbf{A} , if \mathbf{A} has rank approximately equal to 1 [Mahoney, 2011]. This occurs when the largest degree of a network is *much* larger than the rest of the degrees. In other words, this occurs if the degrees follow a *power-law* distribution. This led to the question of whether or not to expect real-world data. Chapter 3 details an empirical study on features of real world networks. In particular, we focus on the existence of power-laws in degrees and spectra, and find that power-laws are actually more prevalent in the spectra than in the degrees. We suspect the results of the reliable power-law in the spectra to be a useful property for characterizing the extremely fast convergence of many matrix-based algorithms on these types of networks.

Faced with a better understanding of the behavior of networks, the remainder of this thesis is dedicated to studying and constructing new graph models. In Chapter 4 we present an extension of a Kronecker model, extended to tensors. Our work makes use of higher-order structure to impose desirable features in network samples. In Chapter 5, we present another model based on a preferential attachment scheme, where new nodes form triangular structure with existing nodes. This new model has a provable power-law distribution in both the degrees and spectra, making it ideal to explore further the hypothesis garnered from our empirical study. Specifically we wonder whether there are changes to the presence of power-laws after sub-sampling these networks. It turns out that the spectral power-laws seem to be much more resilient to perturbation, giving one explanation for why they appear so frequently in real-world data.

The work of this thesis has been joint work with my adviser David Gleich. The work in Chapter 4 is also joint with Arjun Ramani.

2. BACKGROUND

In this chapter we will define many terms and ideas which are used throughout the thesis.

2.1 Graphs and Matrices

Let $G = (V, E)$ be an unweighted, undirected graph without any loops. That is, V is a set of nodes, and E is a set of edges between the nodes. Let $|V| = n$ be the number of vertices and $|E| = m$ the number of edges of G . The adjacency matrix of G is the symmetric matrix \mathbf{A} , where entry $A_{ij} = A_{ji}$ is equal to 1 if there is an edge between vertices i and j , and 0 otherwise. The degree of a vertex i , d_i , is equal to the number of vertices which have an edge connecting to vertex i . Let d_{\max} be the largest degree. We will occasionally consider directed graphs, which correspond to an adjacency matrix which is not necessarily symmetric.

The Laplacian of a graph is $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where \mathbf{D} is the degree matrix, which is a diagonal matrix with $\mathbf{D}_{ii} = d_i$. Notice that the diagonal elements of \mathbf{L} are simply the degrees d_i and the off diagonal elements are either 0 or -1 . It is well known that the eigenvalues of \mathbf{L} , $0 = \mu_1 \leq \mu_2 \leq \dots \leq \mu_n$, are non-negative.

Any real-valued symmetric matrix has a set of n eigenvalues and an orthogonal set of n eigenvectors. The eigenvalues of the adjacency matrix range between $-d_{\max}$ and d_{\max} ; the eigenvalues of the Laplacian range from 0 to $2d_{\max}$. For a real-valued symmetric matrix, the singular values are the absolute values of the eigenvalues, so for an adjacency matrix, they range from 0 to d_{\max} . These topics can be studied further in Golub and Van Loan [2013] for example.

2.2 Graph Modeling

Much of this thesis is on modeling graphs, often called networks in this context. Simple models such as the configuration model [Bender and Canfield, 1978] and Chung-Lu [Aiello et al., 2000] are designed to capture the degree distribution of a network, but typically fail to capture any higher-order pattern such as a clustering coefficient. Conversely, models that are designed to capture arbitrary features including clustering, such as exponential random graph models, often have exponential computational complexities due to the difficulty of the sampling procedure [Bhamidi et al., 2011]. See for instance a recent survey on the difficulty in generating samples of graphs with the same degree

distribution [Fosdick et al., 2018]. More structural models, such as stochastic block model, are often designed to test extremely specific hypotheses involving communities and may not be appropriate as more general models. Pragmatic models such as BTER explicitly place clustering in a carefully designed pattern [Kolda et al., 2014] at the cost of a larger description of the network. Moreno et al. [2010] illuminated this issue in Kronecker graphs as well, and propose a mixed model which allows for more variation. As an area of active work, this gap between network models and real network data has implications for both studies on the performance of network algorithms when synthetic graphs are used as benchmarks—is it relevant if an algorithm scales well on an unrealistic model of networks?—as well as in the space of hypothesis testing on networks where the synthetic graphs are used as null-models—is it relevant if a feature of a network is a low-probability event with respect to an unrealistic model?

2.2.1 Sampling Graphs from Probability Matrices

One of the models of interest comes from the class of graph generators that involves sampling edges from a probability matrix. Examples include the Erdős-Rényi model [Erdős and Rényi, 1959], Chung-Lu model [Chung and Lu, 2002], Stochastic Block Model [Holland et al., 1983], and the Kronecker Model [Leskovec et al., 2010, 2005b, Chakrabarti et al., 2004]. This type of generator starts with a matrix of probabilities, \mathbf{P} , with the number of rows and columns equal to the number of nodes desired in the graph. For each entry i, j of \mathbf{P} , set \mathbf{A}_{ij} equal to 1 with probability \mathbf{P}_{ij} , and set \mathbf{A}_{ij} equal to 0 otherwise. This type of model allows for generating many instances of a graph from a single generator matrix \mathbf{P} .

Kronecker Graph. The HyperKron model which will be presented in Section 4.1 is built on many of the same motivations of the Kronecker Graph Model [Leskovec et al., 2010, 2005b, Chakrabarti et al., 2004]. For the Kronecker Graph, let \mathbf{P} be an $n \times n$ matrix of probabilities called an *initiator* matrix, with n small (n between 2-5 is typical). The Kronecker Product of \mathbf{P} with itself is the $n^2 \times n^2$ matrix constructed by multiplying every entry of \mathbf{P} with itself. For example, if \mathbf{P} is the 2×2 matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$, then the Kronecker product $\mathbf{P} \otimes \mathbf{P}$ is

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

Define the r th Kronecker product $\mathbf{P}^r : n^r \times n^r$ to be r Kronecker products of \mathbf{P} with itself:

$$\mathbf{P}^r = \underbrace{\mathbf{P} \otimes \mathbf{P} \otimes \cdots \otimes \mathbf{P}}_{r \text{ times}}.$$

Then \mathbf{P}^r is the matrix of probabilities used to sample a graph. It is worth being explicit here that we are abusing notation and use \mathbf{P}^r to indicate “powering-up” Kronecker products rather than the

standard notation of “powering-up” by repeated matrix multiplication. We never multiply matrices and only multiply by Kronecker products.

Kronecker Graph Properties. The Kronecker graph model became popular because of a number of desirable properties such as skewed degrees [Seshadhri et al., 2013] and similar properties to real-world networks [Leskovec et al., 2010]. Additionally, storage of the initiator matrix \mathbf{P} is very cheap, at just n^2 entries (where n is often taken to be 2). It has been used as a synthetic generator for parallel graph benchmarks [Murphy et al., 2010] (the Graph500 benchmark). Choosing parameters in the Kronecker model to *fit* a given graph has been studied using maximum-likelihood methods [Leskovec et al., 2010] and method-of-moments estimators [Gleich and Owen, 2012].

2.2.2 Preferential Attachment

Another type of model of interest falls under the term *Preferential Attachment*. Preferential attachment (PA) describes a mechanism of graph evolution in which nodes with higher degree tend to continue gaining neighbors. Consider G_0 to be some initial graph at time $t = 0$. In an evolutionary graph model, new nodes or edges (or both) are added at each successive time step to generate a sequence of graphs: G_1, G_2, \dots . Specifically in PA, at time t we add a new node u to the graph. Also, choose an existing vertex v from G_{t-1} with probability proportional to its degree. Formally, choose vertex v with probability

$$\gamma_t(v) = \frac{d_{t-1}(v)}{\sum_{w \in V_{t-1}} d_{t-1}(w)}, \quad (2.1)$$

where $d_t(w)$ is the degree of vertex w at time t , and V_t is the set of vertices at time t . Then add an edge connecting u to v . PA is meant to model the *power-law* behavior that is often seen in real-world networks [Faloutsos et al., 1999, Huberman, 2001, Medina et al., 2000], that is a few vertices tend to have very large degree while most vertices have fairly low degree. (See Section 2.5 for more on power-laws.)

Preferential attachment as a mechanism for the appearance of power-laws was described by [Yule, 1925], and later applied to networks in [Price, 1976] described as the “rich get richer”. The PA graph model itself is found in a few different forms. In the model by Barabási and Albert [1999], often called the BA model, at every new time step, a new vertex is formed with m edges. Each of the edges is then connected to an existing node chosen using PA, i.e. based on their degrees.

In a slight variation [Chung et al., 2006, Cooper and Frieze, 2001], at each time step t , a new node is added with probability p . Along with the new node is an edge between the new node and an existing node picked via PA. With probability $1 - p$ a new edge is added between two existing nodes, both chosen via PA. These two models generate slightly different distributions, but fundamentally give very similar graphs.

2.3 Higher Order Features in Graphs

Recently, there has been interest in higher order network analysis [Yin et al., 2018, Grilli et al., 2017, Rosvall et al., 2014, Xu et al., 2016, Benson et al., 2016, 2017]. At a high-level, this constitutes an analysis of network of data in terms of multi-node patterns such as motifs and also in terms of stochastic processes that depend on more history. One of the earlier motivations for this direction is the famous paper by Milo on the presence of motifs in real world networks [Milo et al., 2002]. Likewise, there are new models which aim to match these higher order features. For example the triad formation model described in Section 5.1.2 [Holme and Kim, 2002], and the family of PA models [Ostroumova et al., 2013] discussed in Chapter 5.

2.4 Clustering

A common graph feature to measure are called the *clustering coefficients*. The clustering coefficient for a node u is

$$\frac{2|K_3(u)|}{|W(u)|},$$

where $|K_3(u)|$ is the number of triangles for which u is a member, and $|W(u)|$ is the number of wedges for which u is a member. The average local clustering coefficient is the average over the local clustering coefficient defined for each node.

The global clustering coefficient is

$$\frac{6|K_3|}{|W|},$$

where $|K_3|$ is the number of triangles, and $|W|$ is the number of wedges. This gives a way to measure how much network nodes tend to form triangles [Watts and Strogatz, 1998].

We can define *higher-order* clustering coefficients, given by Yin et al. [2018] to be the fraction of appropriate motifs which are closed. Formally, the l -th order global clustering coefficient is

$$\frac{(l^2 + l)|K_{l+1}|}{|W_l|},$$

where $|K_{l+1}|$ is the number of $(l + 1)$ -cliques, and $|W_l|$ is the set of l -wedges. (An l -wedge is a $l - 1$ clique plus one edge). Similarly, the local clustering coefficient is the average over the local clustering coefficients for each node: $|K_{l+1}(u)|/|W_l(u)|$. Note that our earlier definition of clustering corresponds with the 2nd order clustering coefficient.

These metrics are used throughout this thesis to compare how well various graph models compare to real world networks.

2.5 Power-laws

A set of values x_1, \dots, x_k satisfies a power-law if it is drawn from a probability distribution where $p(x) \propto x^{-\alpha}$ for some α . A set of values satisfies a power-law with a cutoff if $p(x) \propto x^{-\alpha}$ for all $x \geq x_{\min}$. The second case can be considered to describe a distribution with a power-law tail. Power-laws appear as *linear* relationships on a *log-log* plot because of the equivalent formulation:

$$\log p(y) = -\alpha \log y + c.$$

Traditional methods to test for a power-law fit take advantage of this relationship and find a linear fit in the *log-log* plot. Unfortunately this type of graphical method is subject to errors [Goldstein et al., 2004]. A better methodology is due to Clauset et al. [2009] which uses a maximum likelihood fitting to determine whether data follows a power-law fit.

There is much debate on the presence of power-laws in real-world data [Broido and Clauset, 2018, Sala et al., 2010b, Kwak et al., 2010, Achlioptas et al., 2005, Gjoka et al., 2010, Meusel et al., 2015, Schoenebeck, 2013]. The details of why researchers are skeptical on the results on power-laws is discussed further in section 3.2.4, but include potential biases in sampled or crawled data. Nevertheless, it is standard practice to assume that power-laws are present in a variety of data. That is to say, we acknowledge that power-law distributions may appear in the our studied network data, which may not actually be a part of the ‘true’ data. Yet, this is the data that ultimately we wish to model and work with.

3. EMPIRICAL STUDY OF REAL WORLD NETWORKS ¹

Power-laws are a key component in many characterizations of the networks gathered from the world wide web and other large information sources. These include web-crawls, online social networks, recommender systems, and many other examples [Faloutsos et al., 1999, Medina et al., 2000, Huberman, 2001]. There are quite a few places that power-laws may arise in the description of these networks. For instance, the degree distributions of these networks are often observed to have a power-law. Additionally, the eigenvalues of these networks are often observed to obey a power-law. Power-laws also arise in other types of structural statistics about the networks [Papalexakis et al., 2016]. Properties of these power-laws are then used to generate realistic synthetic network models [Aiello et al., 2000, Bonato et al., 2012] and to establish theory about why various algorithms work better than expected in networks of this type [Kurauskas and Bloznelis, 2013, Gleich and Seshadhri, 2012, Cooper et al., 2012, Latapy, 2008, Watts, 2003].

Towards these dual goals of building realistic models and generating useful theory, it is useful to have accurate information about the presence of power-laws in these real-world networks. The folk-lore about this is that networks have a power-law in their degree distribution with exponent between 2 and 3 (see, e.g., [Chung et al., 2003b]). This finding does not always hold [Sala et al., 2010a] and there is even contradictory evidence that these networks have power-law distributions [Broido and Clauset, 2018, Sala et al., 2010b, Kwak et al., 2010]. Moreover, there is a diverse literature on the implications of a power-law in the degree distribution for the behavior of the eigenvalues of the adjacency matrix and Laplacian matrix [Mihail and Papadimitriou, 2002, Chung et al., 2003a, Goh et al., 2001]. This literature argues that in specific models of a network, a power-law in the degree distribution implies a power-law in the eigenvalue distribution. It also hypothesizes that this may hold more broadly outside the specific model.

We revisit many of these empirical findings with the goal of providing new guidance on the presence of power-laws in three features of real-world networks:

1. the degree distribution;
2. the singular value distribution of the adjacency matrix;
3. the eigenvalue distribution of the Laplacian matrix;

¹This chapter was previously published by the ACM [Eikmeier and Gleich, 2017]

We built a large collection of real-world networks from the Stanford network analysis project, Facebook, and various other sources (See Section 3.1 for more about where our data originates.) We have computed the singular value distribution and Laplacian eigenvalue distribution exactly using a large cluster of high-performance computers [Gleich, 2019]. To each distribution on each network, we fit the coefficients of a power-law distribution with cutoff in the tail following the methodology of Clauset et al. [2009]. (More specifically, we used the implementation by Nepusz [2016], see the details our methods in Section 3.2). This fitting also included a test of significance, which allows us to gauge the reliability of the results. We call a power-law fit *significant* if it passes this test. This methodology resulted in the following observations.

1. Many networks have a significant power-law in the tail of the degree distribution corresponding to the largest degree vertices, as well as the singular values of the adjacency matrix, and the eigenvalues of the Laplacian matrix. (Section 3.3.1)
2. A significant power-law distribution is more likely to occur in the singular values of the adjacency matrix compared with the degree distribution. This means it is *more accurate* to assume a model where the singular values of the adjacency matrix have a power-law compared with the degree distribution. (Section 3.3.1)
3. A significant power-law distribution in the degrees means there is a high probability for a significant power-law distribution in the singular values of the adjacency matrix and the eigenvalues of the Laplacian matrix. The converse does not hold. (Section 3.3.2)
4. The coefficients of these power-laws vary from 2–10 for all three distributions (degrees, adjacency singular values, and Laplacian eigenvalues). This is a much larger range than has been observed previously. (Section 3.4.1)
5. The tail of the degree distribution and the Laplacian eigenvalues appear to behave identically and have essentially the same power-law distribution. That is, the power-law exponent and cutoff value are almost identical between the fitted distributions. (Section 3.4.1)
6. The region of the distribution where the power-law fits appears to scale as $n^{2/3}$ for the degrees and Laplacian eigenvalues and between $n^{2/3}$ and $n^{1/2}$ for the singular values. (Section 3.4.3)
7. We use observation 6 to test a number of large networks beyond those used to make observations 1-6 because it shows we would not have to compute the entire singular value and eigenvalue spectra. We find these observations hold on eight graphs up to 2 magnitudes larger than those used to form our hypotheses. (Section 3.5)

Overall, these findings refine our view of the power-laws and their relationships in real-world data of relevance to the community.

The presence of a power-law distribution in the singular values is an extremely powerful analytic property for understanding the real-world behavior of many types of matrix-based computations on large social networks (and why we would expect it to be far better than the worst case scenario). We believe the observation that the power-law in the singular values of the adjacency matrix is a more consistently observable feature than the power-law in the degree distribution to be a novel and useful outcome from this study, which is explored further in Chapter 5.

In comparison with past studies revisiting power-laws in networks [Sala et al., 2010b], our focus is on the power-laws in the eigenvalues and singular values across a broad spectrum of networks. Regarding other conjectures and findings about the lack of power-laws in data [Meusel et al., 2015, Gjoka et al., 2010], we detail a few differences in our methodology in Section 3.2.4.

We provide the results of our power-law fits as well as our analytical tools in the github repository: <https://github.com/eikmeier/powerlaw-spectra>.

3.1 Data Sets and Models

Table 3.1 shows a quick view on all of our datasets. We have divided them into a number of groups based on common types of data. For some types of networks, we have a large number of samples (Facebook, Erdős, AS, Oregon, P2P), which we expect to be more highly related than the more general categories, and so these become their own categories. We also investigate four network models: graphs with a prescribed power-law degree distribution, graphs sampled from the copying model of graph evolution, graphs sampled from the preferential attachment model, and graphs sampled from the forest fire process. See 3.1.2 for more information on these models.

3.1.1 Real-world data

The real-world datasets are broken into three categories in Table 3.1 (the fourth category contains the models). The first category is real-world data where a power-law might be a possibility such as in collaboration networks, biology networks, citation networks, etc. The second (small) category is a subset of graphs where we do not expect power-law fits as the data comes from a low-dimensional space (low-dim). These include road networks and meshes. The third group is a number of *sets* of networks that have similar structure.

This large collection of real-world networks comes from a number of publicly available sources, but primary among them are the SNAP repository [Leskovec, 2016], Pajek collection [Batagelj

and Mrvar, 2006], the University of Florida sparse matrix collection [Davis and Hu, 2011], and the Facebook100 [Traud et al., 2012]. There is overlap and duplication of networks between these groups. We also used a number of smaller collections of networks. We have attempted to cite a large subset of the suggested sources for the networks we have used.

Small collections. *Fictional social networks* [Alberich et al., 2002]; *Collaboration* [Bonchi et al., 2012]; *Relational (Dictionary) blondel2004-graph-similarity*; *Biology* [Singh et al., 2008, Klau, 2009]; *Technological* [(The Cooperative Association for Internet Data Analyais), 2005]; *Web* [Constantine and Gleich, 2007]; *Low-dim. (Mesh)* [Gilbert and Teng, 2002].

Newman’s collection [Newman, 2006, 2001] *lesmis* [Knuth, 1993]; *dolphins* [Lusseau et al., 2003]

Arenas’s collection: *Jazz* [Gleiser and Danon, 2003], *email* [Guimerà et al., 2003], *PGP* [Boguñá et al., 2004],

SNAP. We used the following networks from SNAP. **Collaboration** *ca-AstroPh*, *ca-CondMat*, *ca-GrQc*, *ca-HepPh*, *ca-HepTh* [Leskovec et al., 2007]; **Social** *email-Enron*, *soc-Epinions1*, *soc-Slashdot0811*, *soc-Slashdot0902*, *wiki-Vote* [Leskovec et al., 2009]; **Web** *web-NotreDame* [Barabási et al., 1999].

Pajek. We used the following networks from Pajek. **Citation** *Kohonen*, *Lederberg*, *patents_main*, *SciMet*, *SmaGri*, *Zewail*; **Collaboration** *geom*; **Relational** *CSPhd*, *EVA*; **Technological** *USpowerGrid* **Web** *California*, *EPA*; **Word** *dictionary28*, *EAT_RS*, *FA*, *foldoc*, *ODLIS*, *Reuters911*, *Roget*, *Wordnet3*.

For the third group, (of sets of similar networks) we provide a bit of detail next. The AS type is autonomous systems network of routers on the internet, with edges as communications between two vertices [Leskovec et al., 2005a]. The Oregon graphs are also autonomous systems [Leskovec, 2016]. Each of the Facebook networks are social networks where nodes represent people, and edges are a “friendship” between two nodes [Traud et al., 2012]. The P2P graphs are peer-to-peer networks from Gnutella, where nodes are agents and edges are again communication between two nodes [Leskovec et al., 2007]. Erdős is a collection of networks of Erdős’s co-authors [Batagelj and Mrvar, 2006] collected over a few years.

3.1.2 Models

We now describe some relevant details about the models as there are often a variety of construction details that can vary, and we wish to be precise about our methods. Each model has a number of parameters. We picked parameters to explore a diversity of graphs generated by each model. We did not find any characteristic behavior in terms of the parameters.

Table 3.1.
The types of networks we use in our studies, along with a rough order of magnitude of the sizes in vertices.

Type	Description	Sizes
Collab.	Co-authorship or collaboration networks defined by co-occurrence in author lists	100-100k
Biology	Protein-protein interaction networks	100-10k
Citation	Citations or references between a set of papers or other objections	1k-230k
Fiction	Networks drawn from fictional works	100-20k
Relational	A catch-all category for non-specific relational links including recommender system similarities, sports teams, trust networks, and others	100-20k
Social	Networks that model social interactions	100-100k
Tech.	Edges represent physical infrastructure including routers or power grid	5k-200k
Web	Hyperlink networks	1k-300k
Word	Various types of associations between words	100-100k
Low-dim	Networks with low-dimensional geometry (which should not have power-laws)	100-100k
Facebook	The Facebook 100 collection of networks	1k-50k
Erdős	9 collaboration networks centered on Erdős	100-5k
AS	(Autonomous systems) A large set of autonomous systems networks	100-25k
Oregon	Another set of AS networks	10k-11k
P2P	(Peer to peer) networks from Gnutella	1k-100k
RPL	(Random power-law) Random networks generated with a prescribed power-law degree distribution	13k
Copying	Networks from the copying model of graph evolution	1k-100k
PA	(Preferential attachment) networks	1k-10k
Forest fire	Networks generated from a forest-fire process	1k-100k

In the copying model, we start with an initial clique graph and add vertices with the following process. A vertex arrives and picks a parent vertex uniformly at random. This new vertex then copies connections from the parent, but makes mistakes with probability α . A mistake drops a possible link. The graph is always undirected, so nodes can acquire new links via the copying process.

The forest fire model is similar [Leskovec et al., 2007]. We start with an initial clique. A vertex arrives and picks a parent uniformly at random. This new vertex then explores the local neighborhood of its parent via a forest-fire process that is akin to a randomly truncated breadth-first search. This process explores each node in the search frontier with probability q . The new node generates edges to any node that is explored in the process.

The preferential attachment model is the standard model [Barabási and Albert, 1999] where new nodes connect to k -nodes chosen with probability proportional to their degree. The random power-law

models generate a power-law distribution and then sample a graph using the Bayati-Saberi-Kim routine [Bayati et al., 2010] with this degree distribution (or a slight perturbation necessary to ensure a graphical sequence).

3.2 The details of our fitting methodology

Our overall methodology is to fit power-law distributions to the degrees, singular values of the adjacency matrix, and the eigenvalues of the Laplacian matrix. Note that power-laws are not usually described for a mixture of positive and negative values. For this reason, we look at fitting power-law distributions to the *singular values* of the adjacency matrix (rather than the eigenvalues, which may be negative. See Section 2.1). For each distribution, we seek to estimate the power-law coefficient α and the cutoff value x_{\min} , as well as a measure of the significance that we will discuss shortly. To simplify the setting, we consider only undirected, connected graphs without any self-loops. Thus, for any network with directed edges, we remove the directionality of the relationships, and extract the largest connected component.

3.2.1 Computing degrees, singular values, and eigenvalues

For each resulting undirected graph, we compute the degree distribution, all of the eigenvalues of the adjacency matrix (and by taking absolute values, all the the singular values as well), and all of the eigenvalues of the Laplacian matrix. The degree distribution is straightforward. To compute the eigenvalues, we used the MRRR algorithm as implemented in ScaLAPACK [Dhillon et al., 2006, Vömel, 2010], and executed an eigenvalue computation using a cluster of high performance computers at Sandia National Labs where we could load the entire matrix as a dense matrix and execute the $O(n^3)$ algorithm to find them. The description of these computations in more detail is in production [Gleich, 2019].

At this point, we have three collections of non-negative values: the degrees, the singular values of the adjacency matrix, and the eigenvalues of the Laplacian matrix. We remove small elements of the singular values and eigenvalues because the power-law distributions stated above cannot model values of 0. More specifically, due to floating point approximation, we remove any value that is smaller than $2^{-52}n$.

3.2.2 Fitting power-law parameters

To fit the power-law parameters, α and x_{\min} , we use the maximum-likelihood algorithm developed by Clauset et al. [2009]. Using this method is more accurate than the traditional method of fitting the slope of the *log-log* plot. More specifically, we use the implementation by Nepusz [2016] that uses the BFGS algorithm to estimate the parameters. Additionally, this method and software calculates a goodness-of-fit parameter p that indicates whether the power-law fit is likely to be significant. This score is based on a randomized procedure. If the value $p > 0.1$, then this is evidence that the presence of a power-law is justified. We adopt the term *significant* to describe power-laws that pass this threshold.

3.2.3 Exceptions to our methodology

We note that this procedure worked for the vast majority of networks we mentioned in Section 3.1. All told, we ran these procedures successfully for over 5000 distributions. We were able to fit the coefficients of the power-law for every single distribution. However, the goodness-of-fit computation reliably failed for three degree distributions (all synthetic networks); thus, we discarded these results as we cannot be confident in their significance. This is due to an issue of numerical precision in the software. Specifically, the implementation by Nepusz [2016] uses the L-BFGS algorithm (see for example [Nocedal and Wright, 2006]) to estimate the α parameter. It involves the calculation of the difference between the Hurwitz zeta function at two points, and division. The problem here is reduced to a classic issue in Numerical Analysis: How to avoid error when your computation resembles the following:

$$\frac{\text{small} - \text{small}}{\text{small}}.$$

In any case, we still have an extremely large database of results to study.

We also computed clustering coefficients (see Section 2.4 for definitions) for each of the networks and models. We did not find many instances of power-laws in these distributions.

3.2.4 Limitations

There are two weaknesses with this study that slightly temper our conclusions and we wish to address them. First, our observations 1-6 from the beginning of this chapter originate with data up to size 300k vertices, beyond which point it became computationally difficult to compute entire spectra of the networks. These networks originate from a variety of sources and include crawled as well as sampled networks. Recently, there have been studies on potential biases in

power-law observations in networks of crawled data [Achlioptas et al., 2005, Gjoka et al., 2010, Meusel et al., 2015], which particularly apply to smaller networks. There are instances where the sampling procedure applied to the network causes properties to emerge that are not present in the underlying network [Schoenebeck, 2013, Achlioptas et al., 2005]. We agree our methodology cannot distinguish if the power-law originates due to the network collection methodology or reflects an underlying phenomenon. Although we note that just because there can be biases with crawling networks doesn't mean there will be problems.

To address these limitations, and towards the goal of studying larger networks, we include experiments on a set of large networks in Section 3.5 to investigate what happens for data two orders of magnitude larger than what we used to generate our hypotheses. These experiments support our observations. Furthermore, two of our large networks, *cit-Patents* and *wikipedia* are generated from a network data-dump, rather than a crawl. Additionally, we present new findings on the behavior of power-laws in sampled networks in Chapter 5.

Second, we wanted to study relationships between these power-laws, which meant we only used undirected graphs (and removed direction of edges in directed graphs), and we only considered the largest connected component. For this reason, existing negative results may not be directly comparable to ours. Meusel et al. for example, find that the degree distribution of a 3.56 Billion node web graph does not fit a power-law [Meusel et al., 2015]. Nevertheless, our main interest is not in terms of power-laws in the degrees, but power-laws in the singular values and eigenvalues. Independent of the results with degree distribution power-laws, the observations about power-laws in singular values appear to be more robust than within the degrees – which has the potential to better inform future theoretical models of these networks.

3.3 Presence of Power-Laws

In this section we present the results of the fitting method on our data only in terms of whether or not the distributions support a power-law hypothesis via the goodness-of-fit test. In subsequent section, we will study these power-laws in more detail. This section serves to support the first three findings reported at the beginning of this Chapter.

3.3.1 Many classes of networks have power-laws

First, many networks have a significant power-law in the tail of the degree distribution corresponding to the largest degree vertices, as well as the singular values of the adjacency matrix, and the eigenvalues of the Laplacian matrix. Table 3.2 lists the types of real-world networks and models that

Table 3.2.

Presence of power-laws. For each type of real-world network or graph model, we list the total number of networks, and the number which have significant power-law fits in the degrees, adjacency singular value, Laplacian eigenvalues, and combinations.

	Num.	Distribution						Combinations						All	
		Degrees		Adj.		Lap.		Degrees		Degrees		Adj. &			
				Sing. vals		Eig. vals		& Adj.		& Lap.		Lap.			
Biology	6	4	67%	6	100%	5	83%	4	67%	4	67%	5	83%	4	67%
Citation	6	4	67%	6	100%	5	83%	4	67%	4	67%	5	83%	4	67%
Collab.	13	5	38%	8	62%	5	38%	3	23%	4	31%	3	23%	2	15%
Fiction	3	1	33%	2	67%	0	0%	0	0%	0	0%	0	0%	0	0%
Relational	6	3	50%	3	50%	4	67%	2	33%	3	50%	2	33%	2	33%
Social	9	7	78%	8	89%	6	67%	6	67%	5	56%	5	56%	4	44%
Tech.	4	3	75%	4	100%	2	50%	3	75%	1	25%	2	50%	1	25%
Web	5	4	80%	4	80%	4	80%	3	60%	4	80%	3	60%	3	60%
Word	10	5	50%	9	90%	7	70%	5	50%	4	40%	6	60%	4	40%
Low-dim	2	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Facebook	100	74	74%	75	75%	80	80%	56	56%	73	73%	62	62%	56	56%
AS	855	831	97%	847	99%	846	99%	823	96%	824	96%	838	98%	816	95%
P2P	9	1	11%	6	67%	2	22%	1	11%	1	11%	2	22%	1	11%
Erdős	7	1	14%	7	100%	1	14%	1	14%	0	0%	1	14%	0	0%
Oregon	18	15	83%	16	89%	12	67%	13	72%	11	61%	11	61%	10	56%
Copying	163	69	42%	120	74%	81	50%	50	31%	60	37%	60	37%	43	26%
Forest fire	188	130	69%	82	44%	135	72%	59	31%	108	57%	60	32%	49	26%
PA	81	66	81%	68	84%	81	100%	53	65%	66	81%	68	84%	53	65%
RPL	20	17	85%	16	80%	18	90%	15	75%	17	85%	16	80%	15	75%

were described in the last section. For each type of network, we list the total number of networks and how many of each have a statistically significant power-law fit in the tail of the degree distribution, the singular values of the adjacency matrix, and eigenvalues of the Laplacian. We also list how many networks have a power-law distribution in two or more of those sets.

For example, from Table 3.2 we can see that of the 18 networks in category Oregon, 15 (83%) were found to have a power-law distribution in the degrees. Out of those with a power-law distribution in the degrees, 13 also have a power-law distribution in the eigenvalues of the adjacency matrix, which amounts to 72% of the total 18 networks.

As expected, the *low-dimensional networks* do not have power-law fits. Other classes with only a few significant power-law fits in the degrees include: fiction networks, P2P networks and Erdős’s collaboration networks.

Note that our methodology is not perfectly sensitive as we only identify 85% of the networks with planted power-law distributions from the RPL experiments.

Note also that networks are far more likely to have significant power-laws in the singular values of the adjacency matrix than the degrees. Striking examples of this include the P2P and Erdős classes. Exceptions to this include forest fire networks, and random power-law networks, where this is almost true. This supports our point that it is more consistently true that real-world networks have a power-law in their adjacency singular values compared with their degrees. This observation is much weaker for the Laplacian eigenvalues, for a reason that we will discuss shortly when we analyze the power-law fits themselves (Section 3.4.1).

3.3.2 Relationships between power-laws

In our second analysis, we study combinations of power-laws. We only do this for classes of networks where at least 40% of networks had *all three* power-laws to avoid drawing conclusions from small sample sizes. This removes the classes: Collaboration, Fiction, Technological, Relational, P2P, and Erdős. We still include the network models for reference.

We study these relationships in terms of conditional probabilities. Consider the probability that, given a power-law fit in the degrees, there is a power-law fit in the singular values of the adjacency matrix; denote this as $P[\mathbf{A}|\mathbf{D}]$. In contrast consider the likelihood that given a power-law fit in the singular values of the adjacency matrix, there is a power-law fit in the degrees. Denote this as $P[\mathbf{D}|\mathbf{A}]$. We use a similar notation regarding the Laplacian eigenvalues.

We list the probabilities in Table 3.3. Observe that $P[\mathbf{A}|\mathbf{D}]$ is almost always larger than $P[\mathbf{D}|\mathbf{A}]$ which is to say that a power-law fit in the tail of the degrees gives a high likelihood for a power-law fit in the tail of the singular values, but not vice-versa. Similarly $P[\mathbf{L}|\mathbf{D}]$ is usually larger than $P[\mathbf{D}|\mathbf{L}]$,

Table 3.3.

Conditional probabilities that a power-law distribution in one feature gives a power-law distribution in another. D stands for degrees, A stands for the singular values of the adjacency matrix, and L stands for eigenvalues of the Laplacian. The first column for example is the probability that there is a significant power-law distribution in the singular values of the adjacency matrix given that there is a significant power-law distribution in the degrees. For the first group of measurements, we combine the data and compute probabilities in the summary of *Other* class.

Type	$P[A D]$	$P[D A]$	$P[L D]$	$P[D L]$	$P[L A]$	$P[A L]$
Biology	1.0	0.67	1.0	0.8	0.83	1.0
Citation	1.0	0.67	1.0	0.8	0.83	1.0
Social	0.86	0.75	0.71	0.83	0.62	0.83
Web	0.75	0.75	1.0	1.0	0.75	0.75
Word	1.0	0.56	0.8	0.57	0.67	0.86
<i>Sum. of Other</i>	0.92	0.67	0.88	0.78	0.73	0.89
Facebook	0.76	0.75	0.99	0.91	0.83	0.78
AS	0.99	0.97	0.99	0.97	0.99	0.99
Oregon	0.87	0.81	0.73	0.92	0.69	0.92
Copying	0.72	0.42	0.87	0.74	0.5	0.74
Forest fire	0.45	0.72	0.83	0.8	0.73	0.44
PA	0.8	0.78	1.0	0.81	1.0	0.84
RPL	0.88	0.94	1.0	0.94	1.0	0.89

which means that a power-law fit in the tail of the degrees likely implies a power-law fit in the tail of the Laplacian eigenvalues. Both of these relationships have been studied in a variety of theoretical settings in graph models such as the Chung-Lu graphs [Mihail and Papadimitriou, 2002, Chung et al., 2003b, Elsässer, 2006]. Given the diversity of real-world data explored here, it is reassuring to see that these theoretical predictions have meaningful real-world evidence.

3.4 Analysis of the Power-Laws

In this section we consider the exponents of the power-law distributions, and we elaborate on several observations (points 4-6) from the beginning of this chapter. For some of these studies, we wish to draw conclusions over multiple types of networks. For this reason, we create a new class of

network called “Other” that consists of the classes Biology, Citation, Social, Web, and Word. (These are all classes where at least 40% of networks had a significant power-law in all three distributions. The findings are robust to nearby choices for this 40% threshold and the goal is to exclude classes of networks that seem to reliably *lack* power-laws.)

3.4.1 Characteristics of the power-laws

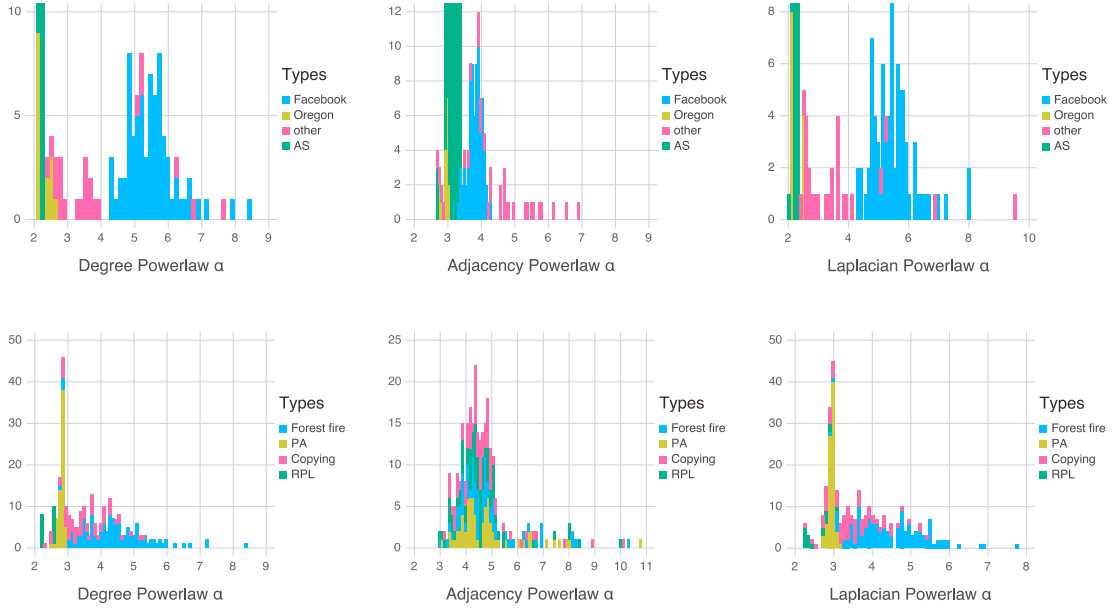


Figure 3.1. **Exponents of statistically significant power-law distributions.** In the first column is the exponent for the degree distribution, in the middle column the exponent in singular values of the adjacency matrix, and in the third column the exponent for the Laplacian eigenvalues. The top row is of the real networks, while the bottom row is of the graph models. The class “other” includes all small classes of real-world networks where power-laws are common (see the text).

The first characteristic of the power-law fits we examine are the exponents α . In Figure 3.1 we plot the exponents of power-law distributions in the degrees, singular values of the adjacency matrix, and eigenvalues of the Laplacian matrix. We consider both real world networks and models. We see that the majority of exponents of the power-law distributions vary from 2 – 10, and notice particularly that they are often greater than 3, which is much larger than observed previously (e.g., [Chung et al., 2003b, Sala et al., 2010a]).

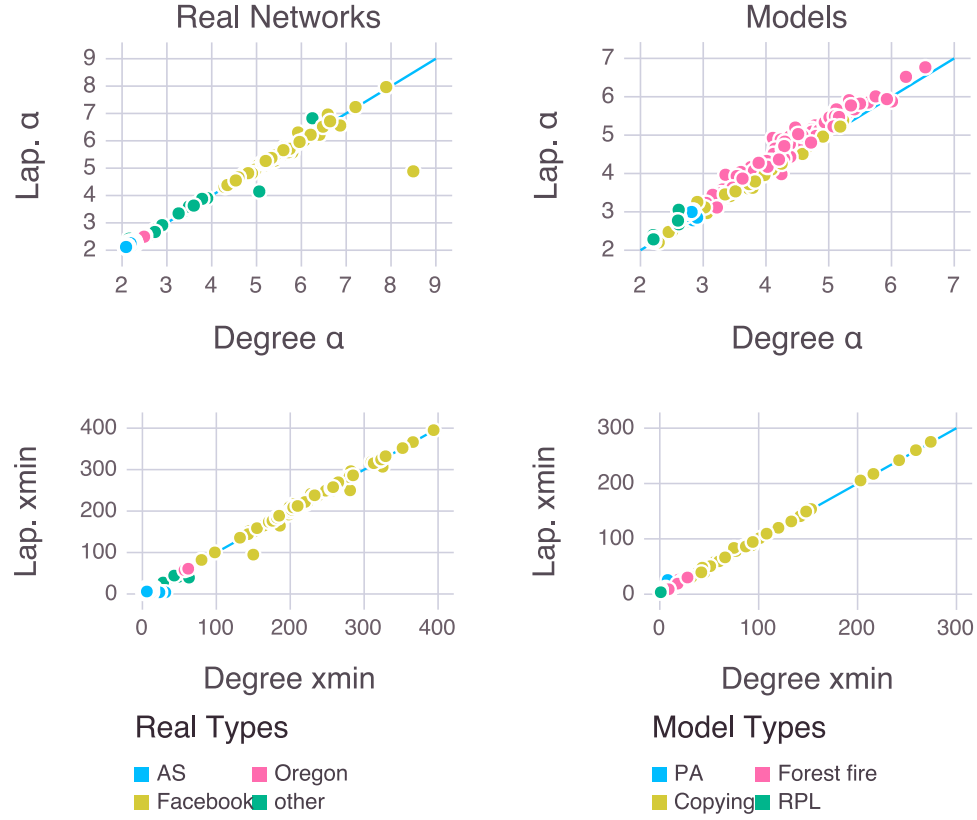


Figure 3.2. **Relationship between power-law in degrees and eigenvalues of the Laplacian.** Given here are networks with statistically significant power-law distributions in both. Top row: the exponent of the power-law fit in the degrees vs. the exponent of the power-law fit in the Laplacian eigenvalues. Bottom row: the cutoff of the power-law fit in the degrees vs. the power-law fit in the Laplacian eigenvalues. On the left are the real world networks, and on the right are the models. We plot the line $y = x$ on the same axes. A few networks are outliers from the line: Newman’s *netscience*, biology (protein-protein) network *dmela*, and social network *Caltech*. We could not find any common features of these outliers.

Next, we notice that the exponent for power-law of Laplacian eigenvalues is often nearly identical to the exponent of the power-law of degree. This has been conjectured for a variety of models [Elsässer, 2006]. Figure 3.2 shows the relationship between these power-law fits, which almost perfectly fits to the line $\alpha_{\text{Laplacian}} = \alpha_{\text{degree}}$, matching the theory well outside of its regime where it should apply. Furthermore, the values for the cutoff value (x_{\min}) in the degrees and Laplacian eigenvalues are nearly identical. This is to say that not only do the power-law fits have the same exponent, but they also fit to the same range of values. Thus, the tail of the degree distribution and the Laplacian eigenvalues appear to have essentially the same power-law distribution.

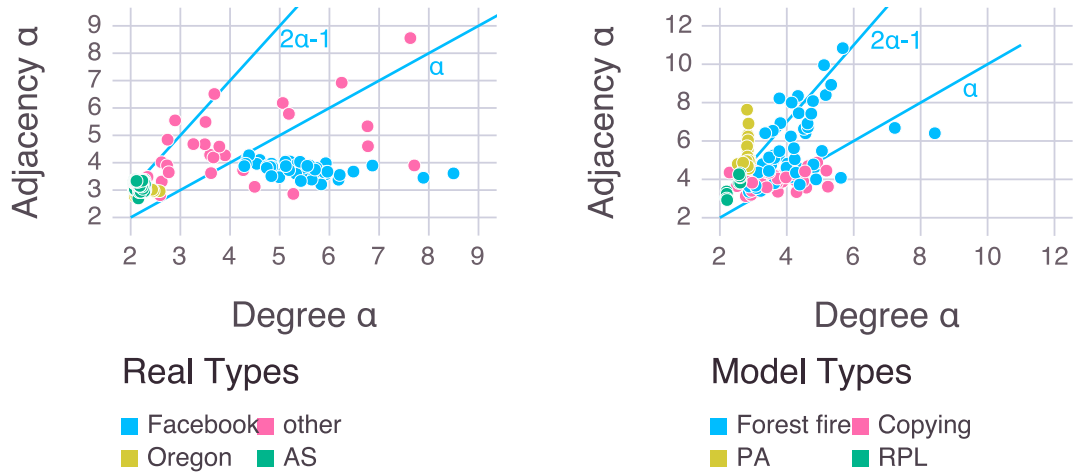


Figure 3.3. **Relationship between power-law in degrees and spectra.** We also plot the line $\alpha_{\text{Adj.}} = 2\alpha_{\text{Deg.}} - 1$, for comparison with results in the literature. The class “other” includes all small classes of real-world networks where power-laws are common (see the text).

Finally, we study the conjecture that the power-law in the adjacency singular values should be $2\alpha_{\text{degree}} - 1$ when there is a power-law in both [Chung et al., 2003a]. Figure 3.3 shows the relationships between these exponents. We see no hints of this scaling law in the real-world data. But, both the random power law graphs and the forest fire graphs show some agreement with this scaling. Thus, whereas the Laplacian result appears accurate, the adjacency result is not.

3.4.2 Consistency across network samples

Another observation we make is that the power-law distribution in the singular values of the adjacency matrix often appears to be more consistent when given multiple samples of the same network. We illustrate this via studying the density of the exponents over instances with multiple similar types of networks: Facebook, AS, and Oregon (Figure 3.4). We include the models Copying and PA (preferential attachment) as well. In the Facebook networks, for instance, the degree power-law exponents vary considerably, whereas the singular value power-law has a sharp distribution about 4. For Oregon and Copying, we see similar behavior. For AS networks, the singular values may have a slightly larger region, the preferential attachment networks are a counter-example.

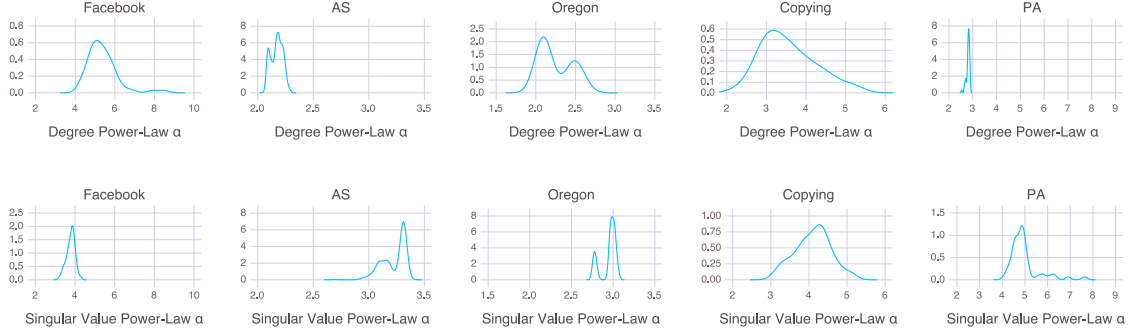


Figure 3.4. **Density estimates of the power-law exponent.** Considered for five classes: Facebook, AS, Oregon, Copying, and PA. On the top are the exponents of the power-law fit in the degree distribution, and on the bottom are the power-law exponents of the singular values of the adjacency matrix. The singular value exponents are more consistent in 3 of the 5 types (Facebook, Oregon, Copying), slightly less consistent in one (AS), and more variable in one (PA).

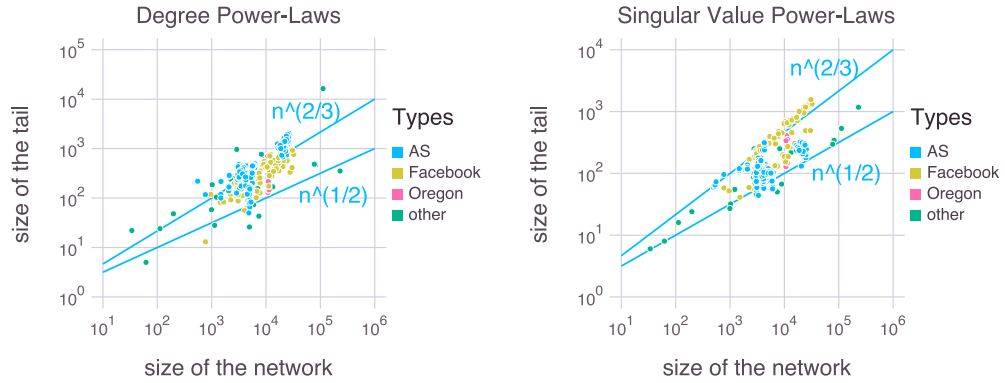


Figure 3.5. **Network size versus size of power-law tail.** Log-Log plots of the size of the network versus the size of the tail in the power-law distributions, (the number of values greater than x_{\min}). The lines $n^{1/2}$ and $n^{2/3}$ are given on the same plot for reference. On the left are the degree distributions and on the right are the singular value distributions. The class “other” includes all small classes of real-world networks where power-laws are common (see the text).

3.4.3 Behavior of the cut-off

Our final observation about the nature of these power-laws reflects the number of entries where the power-law applies. Recall that x_{\min} is the smallest value contained in the power-law distribution. Given a cutoff value x_{\min} , we compute the size of the tail of the distribution, i.e., the number of values larger than x_{\min} . We show the size of the tail relates to the size of the network in Figure 3.5

for both the degree and singular value power-laws. (The Laplacian power-law will behave almost identically to the degree power-law based on the discussion in Section 3.4.1).

The size of the tail appears to scale as $n^{2/3}$ in the degree distribution. A least-squares fit produces essentially the same result ($n^{0.67}$). For the singular values tails, the Facebook class shows the same $n^{2/3}$ scaling; but the other networks show scaling closer to $n^{1/2}$. (The least-squares fit chooses $n^{0.51}$). Both of these scenarios ($n^{2/3}, n^{1/2}$) indicate a shrinking fraction of the network where the power-law applies as the network size increases. However, they also provide useful practical advice about the region where “large degrees” and “large singular values” lie – which is important to understand for analyzing algorithms on these networks as well as designing models.

3.5 Large Networks

The networks discussed up to this point have been relatively small, topping out at around 300k vertices. There is a computational hurdle in computing entire eigenvalue and singular value spectra for graphs with a million or more vertices in that most approaches need $O(n^2)$ memory. The results from the previous section, however, offers an approach: as discussed in Section 3.4.3, a lower bound on the number of degrees or singular values we expect to be included in a power-law tail is $n^{(1/2)}$. This suggests we need not compute all eigenvalues when testing for a power-law distribution.

We considered 8 large graphs from SNAP [Leskovec, 2016], MPI [Mislove et al., 2007], and Wikipedia, listed in Table 3.4 along with information about the power-laws in the data. The *friendster*, *orkut*, *youtube*, *flickr*, *livejournal* data are all social networks, *skitter* is a technological router graph, *patents* a citation network, *wikipedia* a web network formed by Wikipedia articles and their categories where an edge occurs when there is a link between a pair of articles. Note that *wikipedia*, and *patents* are created from database dumps rather than crawls, whereas *orkut*, *youtube*, *flickr*, *livejournal* are all crawled. These results support our findings from previous observations: the singular values of the adjacency matrix are more likely to have a power-law than the degrees. With regards to the cutoff, the vast majority of values are included in these power-law, with exceptions noted below. Thus, these results show that hypotheses formed from graphs up to 300k vertices are also supported on data 100 times larger.

Details of experiment. We chose to restrict our analysis of the large networks to the top $n^{(1/2)}$ degrees and top $n^{(1/3)}$ singular values due to the computational complexity of testing more, and on the assumption that both of these regions would contain power-laws if they are present. The top $n^{(1/3)}$ singular-values of the adjacency matrix of each network were computed using ARPACK [Lehoucq and Sorensen, 1996] (set to a tolerance of 10^{-8}). A power-law distribution was fit using the plfit method discussed in Section 3.2.2. In most cases, the cutoff (see Section 3.4.3) included all or nearly all of

Table 3.4.

Fitting a power-law distribution to large graphs. A power-law was found to be significant with exponent α if it is labeled α^* , and insignificant if labeled α^- . Entries labeled α^\diamond are special cases, and discussed in the text.

Graph	Nodes	Edges	Deg. PL	Adj. PL
<i>youtube</i>	1.13M	2.99M	2.5*	4.2*
<i>flickr</i>	1.62M	15.5M	3.89*	3.09*
<i>skitter</i>	1.69M	11.1M	2.25 ⁻	3.5*
<i>orkut</i>	3.1M	117M	2.62*	4.76*
<i>patents</i>	3.76M	16.5M	4.18*	5.19*
<i>livejournal</i>	5.2M	48.9M	3.3*	3.55*
<i>wikipedia</i>	9.47M	107M	2.46*	4.03*
<i>friendster</i>	65.6M	1.8B	10.5 [◇]	5.04*

the data points which we included for testing. Exceptions are the degree distribution of *livejournal* and the singular value distribution of *flickr*, which only included about half of the values. There is one network, *friendster*, for which the plfit software fails on the degrees when fitting to a discrete distribution (see Section 3.2.3 for a short discussion on this). We instead fit *friendster* to a continuous distribution. This gives similar parameters to the discrete fitting procedure in the other cases we evaluated. The *friendster* network is found to have a significant power-law distribution, but there are only 45 values greater than x_{\min} .

3.6 Summary

By studying a large number of real world graphs, we find empirical evidence that most real world graphs have a statistically significant power-law distribution with a cutoff in the singular values of the adjacency matrix and eigenvalues of the Laplacian matrix in addition to the commonly conjectured power-law in the degrees. Among these results, power-laws in the singular values appear more consistently than in the degree distribution. The exponents of the power-law distributions are much larger than previously observed. We find a surprising direct relationship between the power-law in the degree distribution and the power-law in the eigenvalues of the Laplacian that was theorized in simple models but is extremely accurate in practice. We investigate these findings in large networks by studying the cutoff value itself, which shows a scaling law for the number of elements involved in these power-laws. Using the scaling law enables us to compute only a subset of eigenvalues of large

networks, up to tens of millions of vertices and billions of edges, where we find that those too show evidence of statistically significant power-laws.

4. THE HYPERKRON GRAPH MODEL ¹

While there are many network models (such as those mentioned in the Background of this thesis), the efficient ones often cannot model arbitrary higher-order interactions such as motifs and their interactions. Recall that one of the goals of this thesis is to consider *higher-order structure*, by which we mean patterns on three or more nodes, when generating models of networks. We believe higher-order analysis of networks to be a critical step towards better network analysis techniques, and this thought is supported by recent work [Lambiotte et al., 2019]. It is not obvious how to generate these types of structures for models based on matrices of probabilities such as Erdős-Rényi, Chung-Lu, or kernel functions [Hagberg and Lemons, 2015]. The same critique holds for evolutionary models such as the copying model or forest-fire model.

The primary result of this chapter is a simple and flexible network model that has the ability to capture a single type of higher-order interaction. We call it the HyperKron model. (This is introduced formally in Section 4.1.) For arbitrary higher-order interactions, a more appropriate primitive is hypergraph modeling [Bollobás et al., 2011] to directly model the higher-order interactions—which is where our inspiration came from. In comparison with many other network models, the key difference is that the probability model underlying it specifies a distribution on *hyperedges* rather than edges. To generate a network, we then associate each hyperedge with a specific network motif (such as a triangle or feed-forward motif). As we will show, this model exhibits clustering coefficients which can be closely aligned with real-world data. As might be guessed from the name, the model is a generalization of the extremely parsimonious Kronecker graph model described in Section 2.2.1 [Leskovec et al., 2005b, 2010, Seshadhri et al., 2013].

The HyperKron model will rely on the notion of *hyperedges*. A hyperedge is just a set of vertices. It generalizes an edge which is just a set of two vertices. In this case all hyperedges in our model will have the same cardinality, which is often called a regular hypergraph. (For simplicity, we describe our model where each hyperedge has three vertices.) When we create a graph from a hypergraph in the HyperKron model, we associate each hyperedge with a motif. For most of this chapter, this motif is simply a triangle which enables us to analyze some of the model properties. The HyperKron model is flexible enough to handle other hyperedge structures, as discussed in Section 4.6. Though the HyperKron model uses the concept of hypergraphs, it is important to note that we are not concerned with the generation of hypergraphs. We simply use mechanisms to generate hyperedges to impose

¹This chapter was previously published in an abbreviated form by the IEEE [Eikmeier et al., 2018]

higher-order structure on a traditional graph. This was used as well in Bollobás et al. [2011] where they associated hyperedges with triangles. HyperKron is related to another generalization of the Kronecker Model, RTM [Akoglu et al., 2008], in the sense that it too uses a 3 dimensional tensor, but RTM does not incorporate higher-order structure like HyperKron.

One of the challenges with this model is that an exact and efficient sampling procedure for the desired hyperedge probability distribution is non-trivial to create. The Kronecker model, for instance, has historically been only approximately generated [Moreno et al., 2014]. The situation is even more complex for the HyperKron model and we need to employ techniques including multidimensional Morton codes in order to generate these graphs in time proportional to the number of edges. (Our procedure is explained in Section 4.3).

An advantage to working with the HyperKron model is it admits an analytical characterization of simple properties. We show, for instance, the number of hyperedges that share an edge (Section 4.4.3). This enables us to get accurate estimates of the number of edges the resulting model has for sparse graphs (Section 4.4.4).

Our work also continues to evolve the space of Kronecker models. In fact, our HyperKron model is fairly easy to combine with the majority of other ideas that have been proposed to extend Kronecker models. It would be easy, for instance, to adapt the mKPGM model [Moreno et al., 2010] to our setting as it simply involves a deterministic choice for some of the early tensors. Likewise, the MAG model uses a set of Kronecker models to handle attributed graphs [Kim and Leskovec, 2010].

We conclude the technical portion of this chapter with case-studies on the HyperKron model. We show that the HyperKron model, when using a 3 node motif, generates substantial triadic clustering in fitting real-world network data, far beyond what is possible with Kronecker models (Section 4.5.1). We also illustrate the same generated graphs *lack* clustering structure in four cliques that is present in real-world networks. We finally show that the model is flexible enough to model other types of interactions including directed and signed interactions when the motif associated with the hyperedge is one of the coherent-feed forward motifs (Section 4.6).

4.1 Kronecker products of tensors

The HyperKron Model mimics the ideas of the original Kronecker Model, introduced in Section 2.2.1. The difference is that instead of starting with a matrix and Kronecker-powering it to get a large matrix of probabilities corresponding to edges, we start with a tensor and Kronecker-power it to get a massive tensor of probabilities corresponding to *hyperedges*. For the sake of simplicity in our discussion and analysis, we consider hyperedges with up to three nodes (3d tensors) although

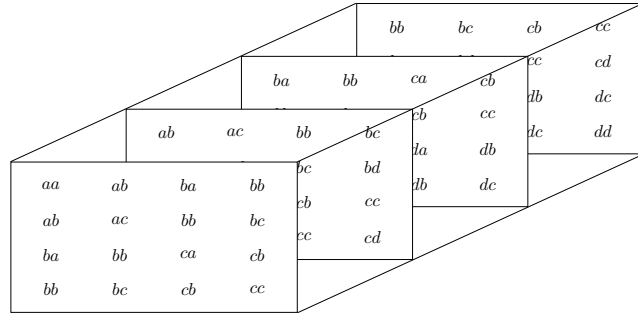
the ideas extend beyond this setting. To generate a graph, we then associate the hyperedge with a triangle. The set-up extends to other motifs on three nodes as discussed in Section 4.6.

In more detail, start with a 3 dimensional initiator tensor, $\underline{\mathbf{P}}$, with dimensions $n \times n \times n$. Just like in the Kronecker model, the value of n should be small, between 2 to 5. For example take a $2 \times 2 \times 2$ symmetric initiator tensor:

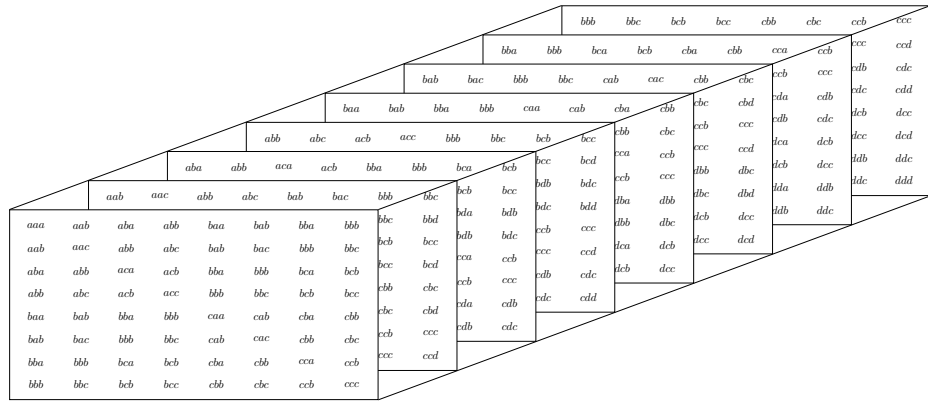

(4.1)

Note that symmetric in the case of a tensor means any permutation of indices has the same value. $\underline{\mathbf{P}}_{112} = \underline{\mathbf{P}}_{121} = \underline{\mathbf{P}}_{211}$. The entries of $\underline{\mathbf{P}}$ should be probability values. (Again, the model is not restricted to symmetric tensors, it merely simplifies the exposition.)

The Kronecker product of tensors, $\underline{\mathbf{P}} \otimes \underline{\mathbf{P}}$, works just like the Kronecker product of matrices: every element gets multiplied by every other element giving way to a $n^2 \times n^2 \times n^2$ tensor [Phan et al., 2012, Akoglu et al., 2008]. In the example above, $\underline{\mathbf{P}}^2 = \underline{\mathbf{P}} \otimes \underline{\mathbf{P}}$ has dimension $4 \times 4 \times 4$


(4.2)

And the third Kronecker product, $\underline{\mathbf{P}}^3$ has dimension $n^3 \times n^3 \times n^3 (8 \times 8 \times 8)$.


(4.3)

4.2 The Symmetric HyperKron Model with Triangles

Given an $n \times n \times n$ initiator tensor \underline{P} of probabilities, construct the r th Kronecker Product of \underline{P} ,

$$\underline{P}^r = \underbrace{\underline{P} \otimes \underline{P} \otimes \cdots \otimes \underline{P}}_{r \text{ times}}.$$

Then \underline{P}^r is of dimension $n^r \times n^r \times n^r$. Note that if \underline{P} is symmetric then so is \underline{P}^r . Generate a set of hyperedges where we include hyperedge (i, j, k) with probability \underline{P}_{ijk}^r . For each generated hyperedge, insert three undirected edges (i, j) , (j, k) , and (i, k) . Duplicate edges are coalesced into a single edge. This results in an undirected graph on n^r vertices. The values of i, j, k need not be unique, so that we may just place an edge (or a loop). An example of the result is show in Figure 4.1. Because we insert undirected edges, it makes the most sense to consider this model with *symmetric* tensors, in this case, we can restrict our generation to cases where $i \leq j \leq k$ (for instance) to minimize the number of duplicates.

The model allows placing triangles with repeated indices, i.e. edges. This is realistic for a real world network, because we would expect some nodes to join the network with only 1 neighbor. That being said, in most cases, we expect that all nodes in the final graph are members of triangles. This is due to the fact that every edge (i, j) has n^k opportunities to be placed.

Instead of inserting 3 undirected edges for each generated hyperedge, we could insert some combination of directed or weighted edges between the three relevant nodes. An application using directed and signed edges is explored in Section 4.6.

4.3 Efficient Generation

A simple algorithm to generate a HyperKron model is to explicitly generate the tensor \underline{P}^r and then to explicitly sample Bernoulli random variables (coin-flips) for each entry in the tensor. If $N = n^r$ is the dimension of the tensor, this is an $O(N^3)$ algorithm, and will not enable us to efficiently generate realistically large networks. Consequently, we seek a more efficient algorithm.

The ideal case for a generation algorithm is that we should do $O(m)$ or $O(m \log N)$ work where m is the number of edges in the output. Note that $r = \log N$. We will show how to get an $O(mr^2)$ method, which can be achieved by adapting the idea of *grass hopping* from our recent paper on generating graphs from matrices of probabilities [Ramani et al., 2017]. (This paper is written with a tutorial style and provides a gentle introduction to many of the topics from this section for those unfamiliar.)

Historically, sampling Kronecker graphs was done using a ball-dropping approach [Leskovec et al., 2005b, Seshadhri et al., 2013] that inserted edges one at a time by simulating where a ball-dropped

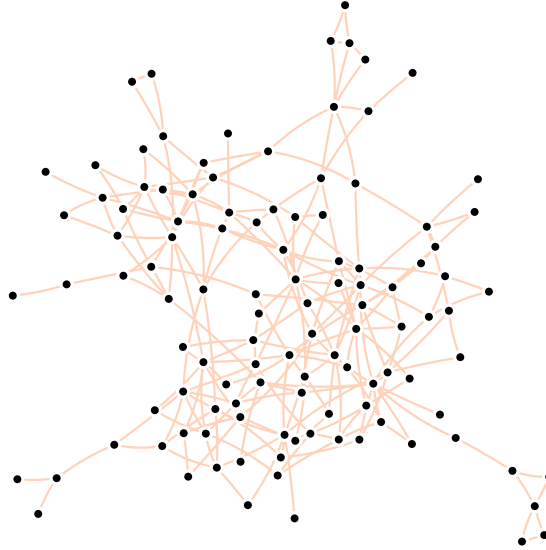


Figure 4.1. **An example graph generated with the HyperKron model.** Symmetric initiator $a=0.8, b=0.115, c=0.215, d=0.61$ was used with $r = 7$ (only the large component is shown). Note the presence of individual edges not involved in triangles.

through successive initiator matrices would land. This gives an $O(mr)$ algorithm, at the cost of an approximate distribution. A faster method to generating an exact Kronecker graph was finally achieved when Moreno et al. [2014] showed that a Kronecker matrix is comprised of a small number of Erdős-Rényi regions and illustrated a way to efficiently sample edges within these regions.

Our approach is also based on the idea of looking at the Erdős-Rényi regions within the Kronecker-powered tensor $\underline{\mathbf{P}}^r$. Recall that an Erdős-Rényi graph is sampled from a matrix where every edge has the same probability of occurring. Define an Erdős-Rényi region as a set of entries in $\underline{\mathbf{P}}^r$ where all the probability values are the same. For instance, note that the probability ab occurs multiple times in (4.2). Edges in these regions can be generated by a waiting time, geometric variable, or grass-hopping method [Fan et al., 1962, Batagelj and Brandes, 2005, Hagberg and Lemons, 2015, Ramani et al., 2017]. That is, sample a geometric random variable to find the gap between successive edges. Thus, the method only does work proportional to the number of edges within the region. What is difficult is to identify *where* these Erdős-Rényi regions occur and then how to map from these regions back to entries in $\underline{\mathbf{P}}^r$.

In the remainder of this section, we show (i) that the number of Erdős-Rényi blocks is sufficiently small that this approach will work given that we have to at least *look* at each block – this analysis will show that the number of such blocks is the number of length r multisetsets of integers $\{0, \dots, n^3 - 1\}$; (ii) how to sample edges in a multiplication table view of the repeated Kronecker product by grass-

hopping (that is, sampling geometric random variables) and unranking multiset permutations; and (iii) how to identify entries in \underline{P}^r by translating the multiplication table indices through a Morton code procedure. The final algorithm is given in Figure 4.2 for reference. Note that our procedure discussed in this section assumes a general initiator tensor \underline{P} that need not be symmetric.

The HyperKron sampling algorithm.

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

Figure 4.2. **Pseudocode for fast hyperedge sampling algorithm on a HyperKron model.** An implementation is at www.cs.purdue.edu/homes/dgleich/codes/hyperkron

4.3.1 A small number of Erdős-Rényi blocks

First, we show that there are fewer than $O(N)$ Erdős-Rényi regions in the graph. (If not, then this procedure would have trouble efficiently generating sparse graphs where $m = O(N)$.) Let the initiator tensor \underline{P} be $n \times n \times n$, and let r be the number of HyperKron products, so that there are $N = n^r$ nodes in the graph. Notice that each probability value in \underline{P}^r is the product of r values from \underline{P} . Entries of \underline{P} can appear more than once in the product, and the entries of \underline{P}^r are only unique up to permutation (that is, the multiplication of these probabilities from \underline{P} is commutative). Thus, the total number of unique probability values in \underline{P}^r is n^3 multi-choose r : $\binom{n^3+r-1}{r}$. We will show that this term goes to $O(n^{n^3-1})$ as r grows, which is less than the $O(n^r)$ nodes of the graph. The statement and proof follow closely from a similar result in [Ramani et al., 2017].

Lemma 4.3.1 *Let \underline{P} be of dimension $n \times n \times n$, and let r be the number of HyperKron products. Then the number of Erdős-Rényi regions in \underline{P}^r is less than $O(N) = O(n^r)$.*

Proof First consider n as a parameter. Sterling's approximation gives $\binom{r+n^3-1}{r} \leq \left(\frac{e(r+n^3-1)}{r}\right)^r$. If $r \geq e(n^3-1)$ then we have at most $(e+1)^r$ regions. In other words, we have the inequality when $e+1 \leq n$. So it remains to show the result for $n=2$ and $n=3$.

We will show that $\binom{r+n^3-1}{r} \leq 2^k$ for $n=2$. Consider the equality $\binom{r+n^3-1}{r} = \binom{r+n^3-1}{n^3-1} = \binom{r+7}{7}$. When $r=20$, $\binom{r+7}{7} = 880,030$ and $2^r = 1,048,576$. Using induction, we'll show that $\binom{r+8}{7} \leq 2^{r+1}$.

$$\begin{aligned} \binom{r+8}{7} &= \frac{(r+8)(r+7)(r+6)}{7!} \\ &= \frac{r+5}{r+5} \cdot \frac{(r+8)(r+7)(r+6)}{7!} \\ &= \frac{r+8}{r+1} \binom{r+5}{7} \\ &\leq 2 \cdot 2^r = 2^{r+1} \end{aligned}$$

The last statement holds when $r \geq 6$.

A similar argument holds when $n=3$. The remaining thing to consider is the behavior as r increases.

$$\begin{aligned} \binom{r+n^3-1}{r} &= \frac{(r+n^3-1)!}{r!(n^3-1)!} \\ &= O\left(\frac{(r+n^3-1)!}{r!}\right) \\ &= (r+n^3-1)^{n^3-1} \end{aligned}$$

because n is a constant. Taking logarithm of both sides,

$$\begin{aligned} \log \binom{r+n^3-1}{r} &\leq (n^3-1) \log r + n^3-1 \\ &\leq (n^3-1) \log r + (n^3-1) \log n^3 - 1. \end{aligned}$$

The last inequality holds if $r \geq 2$ and $n^3-1 \geq 2$. Finally, exponentiating both sides, as k goes to infinity this term goes to r^{n^3-1} which is less than the number of nodes. \blacksquare

4.3.2 Multiplication tables and HyperKron tensors

How can we distinguish each of the Erdős-Rényi regions? Each region is identifiable by its unique product of elements from $\underline{\mathbf{P}}^r$, a probability value. Let us order these probability values. To that end, first number the entries of $\underline{\mathbf{P}}$ 0 through n^3-1 . Then when writing an entry of $\underline{\mathbf{P}}^r$, associate each element in the product with the sequence of r integers. For example, if $\underline{\mathbf{P}}$ is $2 \times 2 \times 2$ as in (4.1), map each of the 8 entries to an index between 0 and 7. Probability ada in $\underline{\mathbf{P}}^3$ would be mapped to 070, where $a=0$ and $d=7$.

It isn't obvious how to easily identify each of the locations of ada in $\underline{\mathbf{P}}^3$. (Or more generally, elements in $\underline{\mathbf{P}}^r$.) We first solve an easier problem and then later determine how to translate back to $\underline{\mathbf{P}}^r$.

If we re-order the entries of \underline{P}^3 so that $ada = 070$ occurred exactly in locations $[0, 0, 7]$, $[0, 7, 0]$, $[7, 0, 0]$, then the locations would be very easy to find—they are just the permutations of $[070]$.

We will call this re-ordering a *multiplication table*. Define $v = \text{vec}(\underline{P})$ to be the initiator tensor as a length- n^3 vector proceeding in a column-major fashion, e.g. the vectorized version of (4.1) is $[a, b, b, c, b, c, c, d]$. Define a r -dimensional multiplication table:

$$M(\underbrace{i, j, \dots, k}_{r \text{ indices}}) = \underbrace{v_i v_j \dots v_k}_{r \text{ terms}}. \quad (4.4)$$

For instance, $M(0, 0, 7) = M(0, 7, 0) = M(7, 0, 0) = aad$. Note that a HyperKron tensor \underline{P}^r is always 3-dimensional, while a multiplication table is r -dimensional.

The start of our strategy is: for each unique probability in \underline{P}^r , given by a multiset of indices as in Section 4.3.1, “grass-hop” sample through the locations in the multiplication table where the probability is all the same. We will see how to do this efficiently next in Section 4.3.3, and finally see how to convert between entries of the multiplication table M and \underline{P}^r in Section 4.3.4.

4.3.3 Grass-Hopping Kronecker Tensors

Given an Erdős-Rényi region in \underline{P}^r or M , we now discuss how to “grass-hop” within that region to find successive hyperedges. (For Kronecker graphs, see the treatment in Ramani et al. [2017]). Let us say that our Erdős-Rényi region corresponds with a probability which is mapped to indices $p = v(i_1)v(i_2)\dots v(i_r)$ where $v = \text{vec}(\underline{P})$ as described in Section 4.3.2. Recall that each of the elements of v are mapped to a numerical index between 0 and $n^3 - 1$. As established in Section 4.3.2, the locations of p in the r -dimensional multiplication table correspond exactly with permutations of i_1, i_2, \dots, i_r . Note that these are permutations of *multisets*, or sets in which elements can occur more than once. We label each permutation lexicographically from 0 to $t - 1$ where

$$t = \frac{m!}{a_1! a_2! \dots a_r!}$$

is the number of permutations of the multiset i_1, \dots, i_r , m is the cardinality of the multiset, and a_i is the number of times that the i th element appears.

The idea for generation then, is that we can easily identify indices between 0 and $t - 1$ where edges occur because each hyperedge occurs with the same probability p . As previously hinted, this is done by sampling a geometric random variable to compute the gap until the next edge (hence we “grass-hop” from edge-to-edge). See the discussions in Ramani et al. [2017], Fan et al. [1962], Batagelj and Brandes [2005], and Hagberg and Lemons [2015] for more about this technique.

Given the indices where hyperedges occurred, we then need to map them to entries of the multiplication table. This can be done by *unranking* multiset permutations [Knuth, 1997, Bonet,

2008]. We wish to order the multi-set permutations, in order to grass-hop among them. To determine this ordering on multiset permutations, we can use the ranking and unranking on combinatorial enumeration as described in [Bonet, 2008].

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

$$\begin{aligned} [0, 1, 1, 2] &\rightarrow 0 & [0, 2, 1, 1] &\rightarrow 2 & \dots & [2, 2, 0, 1] &\rightarrow 10 \\ [0, 1, 2, 1] &\rightarrow 1 & \dots & & \dots & [2, 2, 1, 0] &\rightarrow 11 \end{aligned} \quad (4.5)$$

The unranking of this multiset corresponds to taking one of the indices $0, \dots, 11$ and generating the corresponding permutation. This step can be done in time $O(r^2)$ without any precomputation. The inverse of this process is called *ranking*: finding the index associated with a permutation of a multiset. The details of this process is not difficult to understand, and can be found in our online code and in Ramani et al. [2017].

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

4.3.4 Morton Codes

The last detail is how to map between the r -dimensional multiplication table entry and the 3-dimensional HyperKron tensor $\underline{\mathbf{P}}^r$. The relationship between their locations depends on *Morton codes* as was described for the case of matrices [Ramani et al., 2017]. We extend that analysis to the 3-dimensional HyperKron tensor. A Morton code reflects a particular way of ordering multidimensional data. Because of its shape for a matrix, it is sometimes called a “Z-order.” For a 3d space (such as a tensor), the order is given by a more complicated structure illustrated in Figure 4.3. The particular relationship we use is established by the following theorem.

Theorem 4.3.1 *Let $\underline{\mathbf{P}}^r$ be an $n \times n \times n$ tensor, and \mathbf{v} be the column major representation of $\underline{\mathbf{P}}$. Consider an element in the vectorized multiplication table*

$$\underbrace{\mathbf{v} \otimes \mathbf{v} \otimes \dots \otimes \mathbf{v}}_{r \text{ times}}$$

with index (p_1, p_2, \dots, p_r) . Let I be the lexicographical index of the element (p_1, p_2, \dots, p_r) . Then the base 3-dimensional Morton Decoding of I in base r provides the row, column, and slice indices of an element in

$$\underbrace{\underline{\mathbf{P}} \otimes \underline{\mathbf{P}} \otimes \dots \otimes \underline{\mathbf{P}}}_{r \text{ times}}$$

with the same value. Converting the row, column, and slice indices by using Morton ordering, we can build a permutation matrix $M_{n,r}$, and we can write

$$\text{vec}(\underbrace{\underline{P} \otimes \underline{P} \otimes \dots \otimes \underline{P}}_{r \text{ times}}) = M_{n,r}(\underbrace{\mathbf{v} \otimes \mathbf{v} \otimes \dots \otimes \mathbf{v}}_{r \text{ times}})$$

Proof See Ramani et al. [2017] for the proof for the 2-dimensional Morton decoding. The key idea for the proof is that we can induct over r and show that given that it works for $r = 1$ (trivially) and given that it works for (p_1, \dots, p_r) , then the Morton code including p_{r+1} will “zoom in” on the right region of the full tensor \underline{P}^r .

For brevity, we will write $\underline{P}^{\otimes r} = \underbrace{\underline{P} \otimes \underline{P} \otimes \dots \otimes \underline{P}}_{r \text{ times}}$ and $\mathbf{v}^{\otimes r} = \underbrace{\mathbf{v} \otimes \mathbf{v} \otimes \dots \otimes \mathbf{v}}_{r \text{ times}}$ be defined in the same way. (In the rest of this chapter we simply used \underline{P}^r instead of $\underline{P}^{\otimes r}$, but we wish to use the same notation between \underline{P} and \mathbf{v} .) In the proof, we will show that the Morton code for I gives the correct row, column, and slice indices for $\underline{P}^{\otimes r}$. The resulting permutation matrix follows by converting those indices to a column-major index and building the full permutation matrix. We will prove this statement by induction on the power of the Kronecker tensor product, r .

Base case: For the base case, $r = 1$. This means that $\underline{P}^{\otimes r}$ is the initiator tensor. Then the equality is $\text{vec}(\underline{P}) = M_{n,1}$ which gives $M_{n,1}$ to be the identity.

Inductive step: Assume that $M_{n,r}$ is the correct map. This means that we can find the correct row, column, and slice indices, (R_r, C_r, S_r) for a given index of the r dimensional multiplication table (p_1, p_2, \dots, p_r) .

Let $(p_1, p_2, \dots, p_r, p_{r+1})$ be an index in $v^{\otimes r+1}$. Then by the assumption, (p_1, p_2, \dots, p_r) is an index in $v^{\otimes r}$ where the Morton code correctly gives the correct row, column, and slice values in $\underline{P}^{\otimes r} : (R_r, C_r, S_r)$. To understand the next step, it will be helpful to keep in mind the structure of $\underline{P}^{\otimes r+1}$. Each slice i is of the form:

$$\begin{bmatrix} \underline{P}_{11i} \underline{P}^{\otimes r} & \underline{P}_{12i} \underline{P}^{\otimes r} & \dots & \underline{P}_{1ni} \underline{P}^{\otimes r} \\ \underline{P}_{21i} \underline{P}^{\otimes r} & \underline{P}_{22i} \underline{P}^{\otimes r} & \dots & \underline{P}_{2ni} \underline{P}^{\otimes r} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{P}_{n1i} \underline{P}^{\otimes r} & \underline{P}_{n2i} \underline{P}^{\otimes r} & \dots & \underline{P}_{nni} \underline{P}^{\otimes r} \end{bmatrix} \quad (4.6)$$

Then the row, column, and slice values for $(r_1, r_2, \dots, r_k, r_{k+1})$ are

$$\begin{aligned} R_{r+1} &= R_r + (p_{r+1} \bmod n)n^r \\ C_{r+1} &= C_r + \left(\left\lfloor \frac{p_{r+1}}{n} \right\rfloor \bmod n \right) n^r \\ S_{r+1} &= S_r + \left(\left\lfloor \frac{p_{r+1}}{n^2} \right\rfloor \bmod n \right) n^r \end{aligned} \quad (4.7)$$

So far in this inductive step, we have found the row, column, and slice index which corresponds to a multiplication table index. Now we must show that the Morton decoding process gives the same

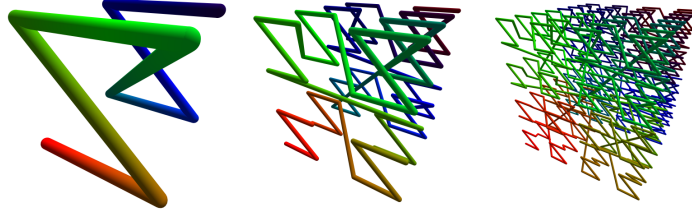


Figure 4.3. **3-dimensional Morton Codes.** Starting from the left, a $2 \times 2 \times 2$ tensor, a $4 \times 4 \times 4$ tensor, and an $8 \times 8 \times 8$ tensor. Image obtained from Hoedt [2012] and used with permission.

correct row, column, and slice index. Lets say that the multiplication table index (p_1, p_2, \dots, p_r) , has lexicographical index I_r . Then we need to find the lexicographical index I_{r+1} which corresponds to $(p_1, p_2, \dots, p_r, p_{r+1})$. This update comes from the fact that the lexicographical index is calculated by converting the multiplication table index from base n^3 to base 10:

$$I_{r+1} = I_r + p_{r+1}(n^3)^r \quad (4.8)$$

Finally, we need to show that I_{r+1} maps correctly to R_{r+1} , C_{r+1} , and S_{r+1} . We'll use the following equalities:

$$p_{r+1} = \left\lfloor \frac{p_{r+1}}{n} \right\rfloor n + (p_{r+1} \bmod n) p_{r+1} = \left\lfloor \frac{p_{r+1}}{n^2} \right\rfloor n^2 + (p_{r+1} \bmod n^2) \quad (4.9)$$

which comes from the division and remainder theorem when calculating p_{r+1}/n and p_{r+1}/n^2 . Substituting these into equation (4.8) gives

$$\begin{aligned} I_{r+1} &= I_r + \left\lfloor \frac{p_{r+1}}{n} \right\rfloor n^{3r+1} + (p_{r+1} \bmod n) n^{3r} \\ &= I_r + \left\lfloor \frac{p_{r+1}}{n} \right\rfloor n^{3r+1} + \left(\left\lfloor \frac{p_r + 1}{n^2} \right\rfloor \bmod n \right) n^{3r+2} + ((p_{r+1} \bmod n^2) \bmod n) n^{3r} \end{aligned}$$

Thus, the Morton decoding of I_{r+1} is equivalent to decoding I_r and adding on the digits for the row, column and slice indices. This finished the proof. ■

4.3.5 Runtime performance

We implemented the generation procedure (Figure 4.2) in the Julia language with a goal towards optimizing easy-to-avoid computational overhead. The resulting program, which is available from www.cs.purdue.edu/homes/dgleich/codes/hyperkron, generates hyperedges at about 1,000,000 per second on a single-thread of modern desktop computer. We evaluated the scalability of the code up to 20 million edges in three scenarios. All three scenarios use a $2 \times 2 \times 2$ symmetric

initiator tensor \underline{P} . This gives four parameters a, b, c, d as in equation (4.1). In the first case, we set $a = 0.05, b = 0.3, c = 0.4$, and choose d such that the expected number of hyperedges $((a+3b+3c+d)^r)$ is 5 times the number of nodes. (For reference, when $r = 10$ then $d = 0.199$ and when $r = 20$ then $d = 0.018$.) In the second case, we set $a = 0.9, b = 0.3, d = 0.0$ and choose c such that the expected number of hyperedges is 10 times the number of nodes. In the third case, we set $a = 0.3, c = 0.3, d = 0.1$ and choose b so there are 20 times the number of hyperedges as nodes. The time it takes to generate graphs as r varies from 10 to 20 is shown in Figure 4.4. Although the theoretical scaling of our procedure is $O(mr^2)$, we observe linear scaling in this regime because the r^2 work can be done efficiently within an array of $4r$ bytes that typically fits in $L1$ cache. Hence, it operates faster than many other steps of the algorithm. The slopes of the lines in Figure 4.4 are approximately 0.09, 0.11 and 0.15 for $\text{avgdeg} = 5, 10$, and 20 respectively.

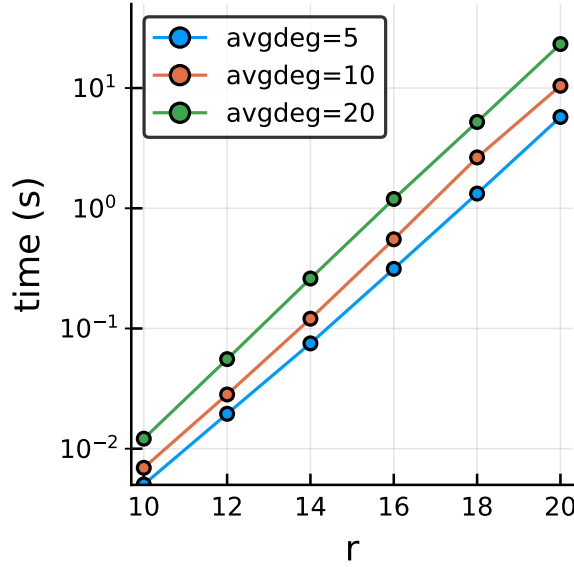


Figure 4.4. **Time taken to generate hyperedges for a HyperKron model.** As r grows, the time shows linear scaling in 2^r . The *average degree* is the expected number of hyperedges per node. See Section 4.3.5 for more about the choice of parameters.

4.4 HyperKron Properties

Next we discuss analytic results and features of the HyperKron model. We compute various expected properties of the model such as the number of hyperedges that share an edge and practical estimates of the total number of generated edges. We also show that the model has non-trivial clustering for a wide set of parameters.

4.4.1 Summation Formulas

Many of the quantities we analytically compute require the following summation formulas. Define \sum^* to be a summation over all combinations of indicated indices except when two or more have the same value. While it will be easy to write down formulas relating to the model in terms of \sum^* , it is easier to sum over the original \sum . So here we present formulas that relate the two. Further explanation for these formulas can be found in Gleich and Owen [2012].

$$\begin{aligned}\sum_{ij}^* f_{ij} &= \sum_{ij} f_{ij} - \sum_i f_{ii} \\ \sum_{ijk}^* f_{ijk} &= \sum_{ijk} f_{ijk} - \sum_{ij} (f_{ijj} + f_{iji} + f_{iij}) + 2 \sum_i f_{iii}\end{aligned}$$

4.4.2 Exact Expectation of Edges

First note that the actual number of edges is

$$E = \frac{1}{2} \sum_{ij}^* \mathbf{A}_{ij} \quad (4.10)$$

where \mathbf{A} is the symmetric adjacency matrix and \sum^* is defined in Section 4.4.1. Let $F_{ijt} = \text{Bernoulli}(\underline{\mathbf{P}}_{ijt}^r)$, so F_{ijt} is 1 with probability $\underline{\mathbf{P}}_{ijt}^r$ and 0 otherwise. The probability that $\mathbf{A}_{ij} = 1$ is equal to the probability that *any* of the Bernoulli samples F_{rst} contain the indices i and j .

Now define a random variable, X_{ij} , for the number of successful trials of F_{ijt} for all t and permutations. Note $X_{ij} \in \{0, 1, 2, \dots\}$, and we can calculate the probability of edge (i, j) as the probability that X_{ij} is greater than or equal to 1. $X_{ij} = \sum_t (F_{ijt} + F_{itj} + F_{jit} + F_{jti} + F_{tij} + F_{tji}) = 6 \sum_t F_{ijt}$. Then we can calculate the expected value for each entry of \mathbf{A} .

$$\mathbb{E}[X_{ij}] = 6 \sum_t \mathbb{E}[F_{ijt}] = 6 \sum_{t=1}^n P_{ijt}$$

$$\mathbb{E}(\mathbf{A}_{ij}) = \mathbb{P}(X_{ij} \geq 1) = 1 - \mathbb{P}(X_{ij} = 0) = 1 - \prod_{t=1}^n \mathbb{P}(F_{ijt} = 0) = 1 - \prod_{t=1}^n (1 - \underline{\mathbf{P}}_{ijt}^r)$$

Then the expected number of edges in the graph is the sum over all entries in the adjacency matrix. We divide by two since the summation counts each edge twice.

$$2\mathbb{E}(E) = \sum_{ij}^* \mathbb{E}(\mathbf{A}_{ij}) = \sum_{ij} \left[1 - \prod_{t=1}^n (1 - \underline{\mathbf{P}}_{ijt}^r) \right] - \sum_i \left[1 - \prod_{t=1}^n (1 - \underline{\mathbf{P}}_{iit}^r) \right]$$

The very last equality comes from Section 4.4.1. This formula, unfortunately, is not computationally helpful.

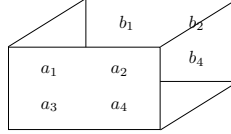
4.4.3 Duplicate Edges Motif

Next we explicitly compute the expected number of duplicate edges placed in the HyperKron model, which we will put to use for a computationally efficient approximation to the number of edges in Section 4.4.4. We expect to see duplicates if two hyper-edges are dropped with a single repeated edge. The number of such features in the model is the sum over all hyper-edges which share an edge, $\sum_{k_1 k_2}^* (\sum_{ij}^* \underline{\mathbf{P}}_{ijk_1}^r \underline{\mathbf{P}}_{ijk_2}^r)$. We split the sums in this way because $k_1 \neq k_2$ and $i \neq j$, but other equalities among indices can occur. Using the relationships in Section 4.4.1,

$$\begin{aligned} 4 \cdot \text{duplicates} &= \sum_{k_1 k_2}^* \left(\sum_{ij}^* \underline{\mathbf{P}}_{ijk_1}^r \underline{\mathbf{P}}_{ijk_2}^r \right) \\ &= \sum_{ijk_1 k_2} \underline{\mathbf{P}}_{ijk_1}^r \underline{\mathbf{P}}_{ijk_2}^r - \sum_{ik_1 k_2} \underline{\mathbf{P}}_{iik_1}^r \underline{\mathbf{P}}_{iik_2}^r - \sum_{ijk_1} (\underline{\mathbf{P}}_{ijk_1}^r)^2 + \sum_{ik_1} (P_{iik_1})^2 \end{aligned}$$

The factor of 4 is due to counting $\{\{ijk_1\}, \{ijk_2\}\}$ 4 times.

We can derive formulas for each of these sums in terms of the values of $\underline{\mathbf{P}}$. The formulas will be based off of a $2 \times 2 \times 2$ non-symmetric initiator matrix, hence is more general than the HyperKron paradigm we've presented. It is easily adjusted to the symmetric case.



We will show how to derive the summation for arguably the most difficult of those presented here, due to the 4 summation indices. The others are similar and we give their values in Eq. (4.11).

Lemma 4.4.1 *Let the entries of the initiator tensor $\underline{\mathbf{P}}$ be*

$$\begin{aligned} \underline{\mathbf{P}}_{111} &= a_1 & \underline{\mathbf{P}}_{121} &= a_2 & \underline{\mathbf{P}}_{211} &= a_3 & \underline{\mathbf{P}}_{221} &= a_4 \\ \underline{\mathbf{P}}_{112} &= b_1 & \underline{\mathbf{P}}_{122} &= b_2 & \underline{\mathbf{P}}_{212} &= b_3 & \underline{\mathbf{P}}_{222} &= b_4 \end{aligned}, \quad \text{then}$$

$$\sum_{ijk_1 k_2} \underline{\mathbf{P}}_{ijk_1}^r \underline{\mathbf{P}}_{ijk_2}^r = [(a_1 + b_1)^2 + (a_2 + b_2)^2 + (a_3 + b_3)^2 + (a_4 + b_4)^2]^r$$

PROOF. We proceed inductively. If $r = 1$, then $\underline{\mathbf{P}}^r = \underline{\mathbf{P}}$ is the initiator tensor, then the indices i, j, k_1, k_2 all range between 1-2, and we can easily write out the sum:

$$\sum_{ijk_1 k_2} \underline{\mathbf{P}}_{ijk_1} \underline{\mathbf{P}}_{ijk_2} = (a_1 + b_1)^2 + (a_2 + b_2)^2 + (a_3 + b_3)^2 + (a_4 + b_4)^2$$

Assume the formula for $r - 1$, we use $\underline{P}^r = \underline{P} \otimes \underline{P}^{r-1}$ to get:

$$\begin{aligned}
\sum_{ijk_1k_2} \underline{P}_{ijk_1}^r \underline{P}_{ijk_2}^r &= \sum_{ijk_1k_2} \left(\underline{P}_{ijk_1} \otimes \underline{P}_{ijk_1}^{r-1} \right) \left(\underline{P}_{ijk_2} \otimes \underline{P}_{ijk_2}^{r-1} \right) \\
&= \sum_{ijk_1k_2} \underline{P}_{ijk_1} \underline{P}_{ijk_2} \left(\sum_{ijk_1k_2} \underline{P}_{ijk_1}^{r-1} \underline{P}_{ijk_2}^{r-1} \right) \\
&= [(a_1 + b_1)^2 + (a_2 + b_2)^2 + (a_3 + b_3)^2 + (a_4 + b_4)^2] \\
&\quad \cdot [(a_1 + b_1)^2 + (a_2 + b_2)^2 + (a_3 + b_3)^2 + (a_4 + b_4)^2]^{r-1} \\
&= [(a_1 + b_1)^2 + (a_2 + b_2)^2 + (a_3 + b_3)^2 + (a_4 + b_4)^2]^r
\end{aligned}$$

4.4.4 Approximate Expectation of Edges

The formula for edges presented in Section 4.4.2 is exact in expectation, but it is computationally *expensive*. We instead offer an approximation for the number of edges that is appropriate when there aren't too many hyperedges. Our estimate comes from the basic idea that 3 times the number of hyper-edges dropped in the model, with small adjustments, should be a good estimate on the total number of edges assuming a sparse set of hyperedges. So to be more precise, our estimate is 3 times the number of 3-edges dropped *plus* 2 times the number of two-edges dropped *minus* duplicates expected at random (Section 4.4.3).

The number of 3-edges dropped is number of hyper-edges dropped with unique indices. This quantity is just the sum over \underline{P}^r :

$$6 \cdot \text{3-edges} = \sum_{ijk}^* \underline{P}_{ijk}^r = \sum_{ijk} \underline{P}_{ijk}^r - \sum_{ij} (\underline{P}_{ijj}^r + \underline{P}_{jij}^r + \underline{P}_{jji}^r) + 2 \sum_i \underline{P}_{iii}^r$$

The factor of 6 is because there are six permutations of (i, j, k) .

The number of 2-edges dropped is the number of hyper-edges dropped with a repeated index, which is exactly the end pieces of the formula above:

$$2 \cdot \text{2-edges} = \sum_{ij}^* (\underline{P}_{ijj}^r + \underline{P}_{jij}^r + \underline{P}_{jji}^r) = \sum_{ij} (\underline{P}_{ijj}^r + \underline{P}_{jij}^r + \underline{P}_{jji}^r) - 3 \sum_i \underline{P}_{iii}^r$$

The factor of 2 is for counting (i, j) twice.

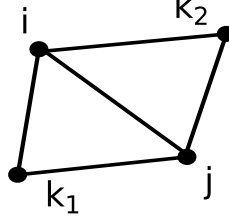


Figure 4.5. **Two hyperedges which share an edge.**

One of the sums from the duplicates formula is explicitly calculated in Section 4.4.3, and given the same setup as in lemma 4.4.1 the remaining sums can be found in similar fashion:

$$\begin{aligned}
 \sum_{ijk} (\underline{\mathbf{P}}_{ijk}^r)^m &= (a_1^m + a_2^m + a_3^m + a_4^m + b_1^m + b_2^m + b_3^m + b_4^m)^r \\
 \sum_{ijk} \underline{\mathbf{P}}_{iij}^r \underline{\mathbf{P}}_{iik}^r &= ((a_1 + b_1)^2 + (a_4 + b_4)^2)^r \\
 \sum_{ij} (\underline{\mathbf{P}}_{ijj}^r)^m &= (a_1^m + b_2^m + a_3^m + b_4^m)^r \\
 \sum_{ij} (\underline{\mathbf{P}}_{iji}^r)^m &= (a_1^m + a_2^m + b_3^m + b_4^m)^r \\
 \sum_{ij} (\underline{\mathbf{P}}_{iij}^r)^m &= (a_1^m + b_1^m + a_4^m + b_4^m)^r \\
 \sum_i (\underline{\mathbf{P}}_{iii}^r)^m &= (a_1^m + b_4^m)^r
 \end{aligned} \tag{4.11}$$

So all together, the estimate for the number of edges is

$$\begin{aligned}
 \mathbb{E}(E) &= 3(3\text{-hyperedges}) + 2(2\text{-hyperedges}) - \text{duplicates} \\
 &= 1/2(a_1 + a_2 + a_3 + a_4 + b_1 + b_2 + b_3 + b_4)^r + 1/2(a_1 + b_2 + a_3 + b_4)^r \\
 &\quad + 1/2(a_1 + a_2 + b_3 + b_4)^r + 1/2(a_1 + b_1 + a_4 + b_4)^r - 2(a_1 + b_4)^r \\
 &\quad - 1/4 \left((a_1 + b_1)^2 + (a_2 + b_2)^2 + (a_3 + b_3)^2 + (a_4 + b_4)^2 \right)^r \\
 &\quad + 1/4(a_1^2 + a_2^2 + a_3^2 + a_4^2 + b_1^2 + b_2^2 + b_3^2 + b_4^2)^r \\
 &\quad + 1/4((a_1 + b_1)^2 + (a_4 + b_4)^2)^r - 1/4(a_1^2 + b_1^2 + a_4^2 + b_4^2)^r
 \end{aligned} \tag{4.12}$$

To verify that the estimate (4.12) is accurate, we test the expected edges count against the true number of edges generated in HyperKron models with the same parameters used as in the experiment in Section 4.3.5. The true number of edges compared to the estimate is presented in Figure (4.6). If you look closely, particularly for small r , (4.12) is not exact. Nevertheless, it is quite accurate. It is important to note that (4.12) will not work for all choices of parameters, particularly those that result in *dense* graphs. For example, choosing parameters $a, b, c, d = (0.99, 0.43, 0.4, 0.009)$ with $r = 13$ leads to an average of 4 million edges generated in a HyperKron graph, while the expected edge count from (4.12) is only 1.98 million. Nonetheless, it is still very useful when fitting sparse real-data to have a way to predict the number of edges which will appear in the HyperKron model.

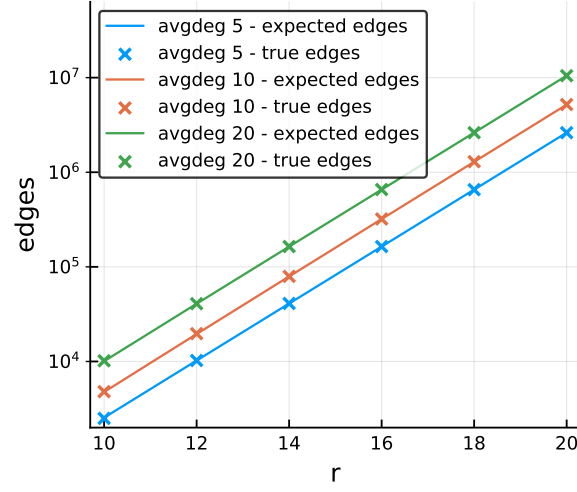


Figure 4.6. **Expected number of edges in the HyperKron model.** The true number of edges generated in the model versus the count from (4.12). See Section 4.3.5 for more about the choice of parameters and average degree.

4.4.5 Extra Formulas

Similar to equation 4.11, other counts the tensors can be made. For fun I computed a few more and record them here.

$$\begin{aligned}
\sum_{ij} \underline{\mathbf{P}}_{ij}^r \underline{\mathbf{P}}_{iii}^r &= (a_1^2 + a_1 b_1 + a_4 b_4 + b_4^2)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iji}^r \underline{\mathbf{P}}_{iii}^r &= (a_1^2 + a_1 a_2 + b_3 b_4 + b_4^2)^r \\
\sum_{ij} \underline{\mathbf{P}}_{ijj}^r \underline{\mathbf{P}}_{iii}^r &= (a_1^2 + a_1 b_2 + a_3 b_4 + b_4^2)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iji}^r \underline{\mathbf{P}}_{ijj}^r &= (a_1^2 + a_2 b_1 + a_4 b_3 + b_4^2)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iji}^r \underline{\mathbf{P}}_{ijj}^r &= (a_1^2 + a_2 b_2 + a_3 b_3 + b_4^2)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iij}^r \underline{\mathbf{P}}_{ijj}^r &= (a_1^2 + a_2 b_1 + a_4 b_3 + b_4^2)^r \\
\sum_{ijk} \underline{\mathbf{P}}_{iii}^r \underline{\mathbf{P}}_{ijk}^r &= (a_1^2 + a_1 b_1 + a_1 a_2 + a_1 b_2 + a_3 b_4 + b_3 b_4 + a_4 b_4 + b_4^2)^2 \\
\sum_{ijk} \underline{\mathbf{P}}_{ijj}^r \underline{\mathbf{P}}_{iki}^r &= ((a_1 + b_1)(a_1 + a_2) + (a_4 + b_4)(b_3 + b_4))^r \\
\sum_{ijk} \underline{\mathbf{P}}_{iji}^r \underline{\mathbf{P}}_{iki}^r &= ((a_1 + a_2)^2 + (b_3 + b_4)^2)^r \\
\sum_{ijk} \underline{\mathbf{P}}_{ijk}^r \underline{\mathbf{P}}_{iji}^r &= (a_1^2 + a_1 b_1 + a_2^2 + a_2 b_2 + b_3^2 + a_3 b_3 + a_4 b_4 + b_4^2)^r \\
\sum_{ijk} \underline{\mathbf{P}}_{ijk}^r \underline{\mathbf{P}}_{iki}^r &= (a_1^2 + a_1 a_2 + a_2 b_1 + a_2 b_2 + a_3 b_3 + a_4 b_3 + b_3 b_4 + b_4^2)^r \\
\sum_{ijk} \underline{\mathbf{P}}_{ijk}^r \underline{\mathbf{P}}_{ijj}^r &= (a_1^2 + a_1 b_1 + b_2^2 + a_2 b_2 + a_3^2 + a_3 b_3 + a_4 b_4 + b_4^2)^r \\
\sum_{ijk} \underline{\mathbf{P}}_{ijk}^r \underline{\mathbf{P}}_{ijj}^r &= (a_1^2 + a_1 a_2 + b_1 b_2 + b_2^2 + a_3^2 + a_3 a_4 + b_3 b_4 + b_4^2)^r \\
\sum_{ijk} \underline{\mathbf{P}}_{ijk}^r \underline{\mathbf{P}}_{iij}^r &= (a_1^2 + a_1 b_1 + a_2 b_1 + b_1 b_2 + a_3 a_4 + a_4 b_3 + a_4 b_4 + b_4^2)^r \\
\sum_{ijk} \underline{\mathbf{P}}_{ijk}^r \underline{\mathbf{P}}_{iik}^r &= (a_1^2 + a_1 a_2 + b_1^2 + b_1 b_2 + a_3 a_4 + a_4^2 + b_3 b_4 + b_4^2)^r \\
\sum_{ijk} \underline{\mathbf{P}}_{ijk}^r \underline{\mathbf{P}}_{ikj}^r &= (a_1^2 + 2a_2 b_1 + b_2^2 + a_3^2 + 2a_4 b_3 + b_4^2)^r \\
\sum_{ijk} \underline{\mathbf{P}}_{ijk}^r \underline{\mathbf{P}}_{ikk}^r &= (a_1^2 + a_1 a_2 + b_1 b_2 + b_2^2 + a_3^2 + a_3 a_4 + b_3 b_4 + b_4^2)^r \\
\sum_{ijk} \underline{\mathbf{P}}_{ijj}^r \underline{\mathbf{P}}_{ikk}^r &= ((a_1 + b_2)^2 + (a_3 + b_4)^2)^r \\
\sum_{ijk} \underline{\mathbf{P}}_{iji}^r \underline{\mathbf{P}}_{ikk}^r &= ((a_1 + a_2)(a_1 + b_2) + (a_3 + b_4)(b_3 + b_4))^r \\
\sum_{ijk} \underline{\mathbf{P}}_{iij}^r \underline{\mathbf{P}}_{ikk}^r &= ((a_1 + b_1)(a_1 + b_2) + (a_3 + b_4)(a_4 + b_4))^r \\
\sum_{ijkt} \underline{\mathbf{P}}_{ijk}^r \underline{\mathbf{P}}_{ijt}^r &= ((a_1 + b_1)^2 + (a_2 + b_2)^2 + (a_3 + b_3)^2 + (a_4 + b_4)^2)^r
\end{aligned}$$

$$\begin{aligned}
\sum_{ij} \underline{\mathbf{P}}_{iji}^r (\underline{\mathbf{P}}_{iii}^r)^2 &= (a_1^3 + a_1^2 a_2 + b_3 b_4^2 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{ijj}^r (\underline{\mathbf{P}}_{iii}^r)^2 &= (a_1^3 + a_1^2 b_1 + a_4 b_4^2 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iii}^r (\underline{\mathbf{P}}_{ijj}^r)^2 &= (a_1^3 + a_1 b_1^2 + a_4^2 b_4 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{ijj}^r (\underline{\mathbf{P}}_{ijj}^r)^2 &= (a_1^3 + a_3 a_4^2 + b_1^2 b_2 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iji}^r (\underline{\mathbf{P}}_{ijj}^r)^2 &= (a_1^3 + a_2 b_1^2 + a_4^2 b_3 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{ijj}^r (\underline{\mathbf{P}}_{iji}^r)^2 &= (a_1^3 + a_2^2 b_2 + a_3 b_3^2 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iii}^r (\underline{\mathbf{P}}_{iji}^r)^2 &= (a_1^3 + a_1 a_2^2 + b_3^2 b_4 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{jjj}^r (\underline{\mathbf{P}}_{ijj}^r)^2 &= (a_1^3 + a_1 a_3^2 + b_2^2 b_4 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iji}^r (\underline{\mathbf{P}}_{ijj}^r)^2 &= (a_1^3 + b_1 b_2^2 + a_3^2 a_4 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iji}^r (\underline{\mathbf{P}}_{ijj}^r)^2 &= (a_1^3 + a_2 b_2^2 + a_3^2 b_3 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{jii}^r (\underline{\mathbf{P}}_{iji}^r)^2 &= (a_1^3 + a_2^2 a_3 + b_2 b_3^2 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{ijj}^r (\underline{\mathbf{P}}_{iji}^r)^2 &= (a_1^3 + a_2^2 b_1 + a_4 b_3^2 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iii}^r \underline{\mathbf{P}}_{ijj}^r \underline{\mathbf{P}}_{iji}^r &= (a_1^3 + a_1 a_2 b_1 + a_4 b_3 b_4 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iii}^r \underline{\mathbf{P}}_{iji}^r \underline{\mathbf{P}}_{ijj}^r &= (a_1^3 + a_1 a_2 b_2 + a_3 b_3 b_4 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iii}^r \underline{\mathbf{P}}_{iji}^r \underline{\mathbf{P}}_{jii}^r &= (a_1^3 + a_1 a_2 a_3 + b_2 b_3 b_4 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iii}^r \underline{\mathbf{P}}_{ijj}^r \underline{\mathbf{P}}_{ijj}^r &= (a_1^3 + a_1 b_1 b_2 + a_3 a_4 b_4 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iii}^r \underline{\mathbf{P}}_{ijj}^r \underline{\mathbf{P}}_{jji}^r &= (a_1^3 + a_1 a_4 b_2 + a_3 b_1 b_4 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{jjj}^r \underline{\mathbf{P}}_{ijj}^r \underline{\mathbf{P}}_{ijj}^r &= (a_1^3 + a_1 a_3 a_4 + b_1 b_2 b_4 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{ijj}^r \underline{\mathbf{P}}_{iji}^r \underline{\mathbf{P}}_{ijj}^r &= (a_1^3 + a_2 b_1 b_3 + a_2 a_4 b_3 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{iji}^r \underline{\mathbf{P}}_{jij}^r \underline{\mathbf{P}}_{ijj}^r &= (a_1^3 + a_2 b_2 b_3 + a_2 a_3 b_3 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{ijj}^r \underline{\mathbf{P}}_{iji}^r \underline{\mathbf{P}}_{iij}^r &= (a_1^3 + a_2 b_1 b_2 + a_3 a_4 b_3 + b_4^3)^r \\
\sum_{ij} \underline{\mathbf{P}}_{ijj}^r \underline{\mathbf{P}}_{jji}^r \underline{\mathbf{P}}_{iji}^r &= (a_1^3 + a_2 a_4 b_2 + a_3 b_1 b_3 + b_4^3)^r \\
\sum_{ijk} \underline{\mathbf{P}}_{iii}^r \underline{\mathbf{P}}_{ijj}^r \underline{\mathbf{P}}_{iik}^r &= (a_1(a_1 + b_1)^2 + b_4(a_4 + b_4)^2)^r \\
\sum_{ijk} \underline{\mathbf{P}}_{iik}^r (\underline{\mathbf{P}}_{iij}^r)^2 &= ((a_1 + b_1)(a_1^2 + b_1^2) + (a_4 + b_4)(a_4^2 + b_4^2))^r \\
\sum_{ijk} \underline{\mathbf{P}}_{ijj}^r \underline{\mathbf{P}}_{iik}^r \underline{\mathbf{P}}_{iit}^r &= ((a_1 + b_1)^3 + (a_4 + b_4)^3)^r
\end{aligned}$$

4.4.6 Non-trivial Clustering

Next we present the case the HyperKron model allows for generating models with significant *clustering* even with few edges. This is an improvement over the original Kronecker model, which we compare to here and explore further in Section 3.2.2. We use the global clustering coefficient, which was explained in 2.4. We generate the HyperKron model for fixed a and d parameters, using $r = 10$. Figure 4.7 demonstrates that for varying all values of b and c the global clustering is always above 0.05, and is often much larger. It is large initially because all edges are in triangles. As the network becomes denser (b, c get larger), then wedges emerge causing the coefficient to drop. Finally,

as the network becomes quite dense, these wedges combine into triangles. But throughout, clustering remains. This is significant because we can still achieve good clustering with sparse networks (the real-world behavior) with the HyperKron model.

In comparison, global clustering coefficients for varying parameters in the Kronecker model are in Figure 4.8. Note first that the y-axis is on a different scale. Second, notice that we use larger parameters for d , in order to find larger global clustering coefficients. The parameters for which the clustering coefficients are greater than 0.05 are all large, resulting in Kronecker graphs which will be quite dense. This makes the parameters not very useful to model real world data.

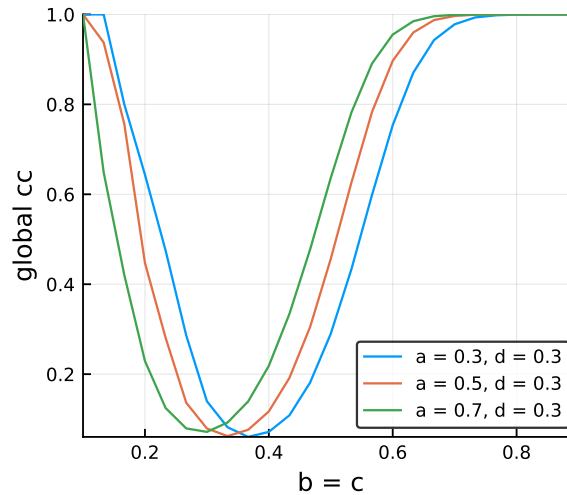


Figure 4.7. **Global clustering coefficients of the HyperKron model.** The coefficients vary with changing HyperKron parameters. Here $r = 10$.

4.5 Fitting HyperKron to Real Data

We demonstrate now that the HyperKron model can be fit to real-world data by hand-tuning the coefficients. Four real-world networks were chosen: *email* comes from Arena’s collection, and is a list of email exchanges between members of the University Rovira i Virgili with 1133 nodes [Guimerà et al., 2003]; *Villanova62* (7772 nodes) and *MU68* (15k nodes) come from the Facebook 100 data set [Traud et al., 2012] where nodes represent people and edges are friendships; and *homo* is a biology network of protein interactions with 8887 nodes [Singh et al., 2008].

To fit real-world data to our HyperKron model, we choose to fit the model to just the set of triangles in the network as this is the natural structure for HyperKron to generate. (See Section 4.5.3 where we consider the full network.) Fitting the coefficients to a symmetric HyperKron model with

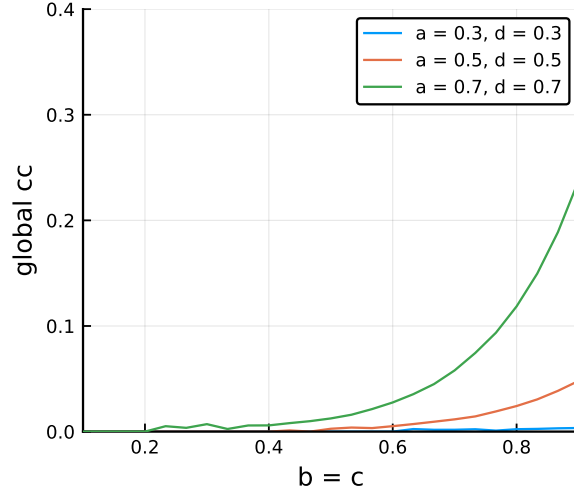


Figure 4.8. **Global clustering coefficients of the Kronecker model.** Coefficients vary with changing Kronecker parameters. $r = 10$.

a $2 \times 2 \times 2$ initiator matrix (four parameters) was done by hand. See our choices a, b, c , and d in Table 4.1.

For comparison, we fit the same data sets to the Kronecker model as well using the method-of-moments [Gleich and Owen, 2012] (called KGMoment in the table) and maximum likelihood [Leskovec et al., 2010] (called KGFit in the table). (These methods often picked different r .) For reference, we also fit those models to the full edge data in addition to the extracted triangle data. While there are other models that would also capture clustering [Kolda et al., 2014, Newman, 2009], these require far more parameters and so we don’t compare against them.

4.5.1 Clustering Coefficients

The traditional global clustering coefficient and the average local clustering coefficient are both described in Section 2.4. One of the biggest improvements of the HyperKron model over other graph models such as the original Kronecker model, is the ability to capture clustering in the model. Regardless of using the full data, or restricted triangle data, the Kronecker models do not capture clustering properties as closely as the HyperKron model does (see Table 4.1).

There remain properties of the real-world networks that HyperKron does not possess. For instance, the HyperKron model lacks higher-order clustering. We use the methodology and code presented in Yin et al. [2018] to compute *higher order clustering coefficients*, described in Section 2.4. We find

that the HyperKron model does not display clustering in terms of four cliques, five cliques, or six cliques (3rd, 4th, and 5th order) and the coefficients for higher orders are small, as seen in Table 4.2.

4.5.2 Skewed Degrees

Another desirable property that the HyperKron model preserves is a highly skewed degree distribution. That is, there are a few nodes with very large degree, with the average degree being much smaller. Figure 4.9 shows the degree distributions in log-scale for two of the networks: *Villanova62*, and *MU78*, along with their HyperKron fits. We also show Loess smoothed estimates to show broader properties.

There are two notable behaviors in the HyperKron degree distribution. First, there are two “tails” in nodes with lower degree. The tail with larger counts are nodes with even degree. They occur in higher frequency since the model most often adds two neighbors to a node at once when we add triangles. Conversely, the tail with smaller counts are nodes with odd degree, since a single edge is placed infrequently. Second, the HyperKron model shows an interesting pattern in the high-degree vertices. This is a known problem with Kronecker models and occurs in the original version as well. The peaks can be smoothed out in the Kronecker model by perturbing the probability matrix as demonstrated in Seshadhri et al. [2013]. We implemented a similar procedure which we explain next.

4.5.3 Improvements to the model

We made several tweaks to the HyperKron generation to address these issues. First of all, we add a *noise* parameter to the HyperKron model in a generalization of the method in Seshadhri et al. [2013]. Recall that the probability of placing a particular hyperedge is the multiplication of r entries of the initiator tensor, \underline{P} . To incorporate noise we perturb each of the r tensors involved in the Kronecker product with two noise parameters, μ_i, ν_i , in the following way. Instead of using the tensor in (4.1), for each level $i = 1, 2, \dots, r$ we use

$$\underline{P}_i = \left[\begin{array}{cc|cc} a - \frac{3a\mu_i}{a+d} - \frac{3a\nu_i}{a+d} & b + \mu_i & b + \mu_i & c + \nu_i \\ b + \mu_i & c + \nu_i & c + \nu_i & d - \frac{3d\mu_i}{a+d} - \frac{3d\nu_i}{a+d} \end{array} \right], \quad (4.13)$$

where μ_i, ν_i are uniformly randomly sampled within an appropriate range, $[-\sigma, \sigma]$ and $\sigma \leq \min(b, c)$ (akin to the case for Kronecker in Seshadhri et al. [2013]). For each noisy \underline{P}_i , the sum of the entries is equal to the sum of the entries of \underline{P} (that is, $a + 3b + 3c + d$). Using these added noise parameters, we fit HyperKron to the set of edges involved in triangles, using the same initiator parameters as before.

Table 4.1.

Fitting real world data to the HyperKron model. Note that the only model presented here with non-trivial global and local clustering are the HyperKron (HKron) fits. See the text for some of the details. We list the number of nodes, number of edges, global clustering coefficient, mean-local clustering coefficient, and the size of the largest connected component.

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

Table 4.2.
Higher order global clustering coefficients of the HyperKron model.

Network name	3^{rd}	4^{th}	5^{th}
<i>email triangles</i>	0.137	0.156	0.223
HKron: (0.999, 0.31, 0.2, 0.0001), $r = 10$	0.065	0.045	0.033
<i>homo triangles</i>	0.113	0.184	0.261
HKron: (0.8, 0.115, 0.15, 0.83), $r = 12$	0.002	0.0	0.0
<i>Villanova62 triangles</i>	0.109	0.115	0.131
HKron: (0.9, 0.4, .24, .001), $r = 13$	0.050	0.037	0.031
<i>MU78 triangles</i>	0.137	0.164	0.175
HKron: (0.9, 0.42, 0.20, 0.001), $r = 14$	0.052	0.040	0.033

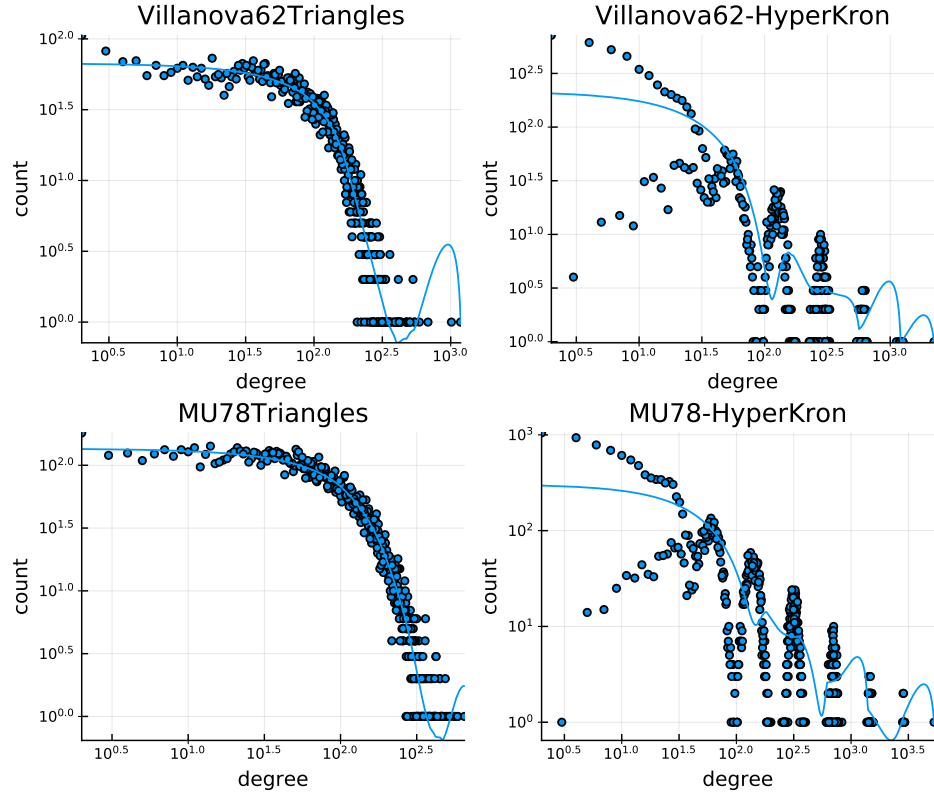


Figure 4.9. **HyperKron preserves highly skewed degree distribution.** Though the distribution is skewed, there is notable behavior which is discussed in the text.

The second adjustment that is to also account for the set of remaining edges (those not involved in triangles). We fit this residual set of edges to a Kronecker model using the method of moments in Gleich and Owen [2012], with an added noise parameter as in Seshadhri et al. [2013]. Note that when we add the Kronecker graph to the HyperKron graph, many of the edges overlap. So finally, we add in an Erdős-Rényi graph with an expected number of edges set to add enough edges to get back to the number of edges of the original graph. Adding on additional edges in these ways is natural, as both the Kronecker and Erdős-Rényi graphs can be derived from subsets of the HyperKron tensor where two indices are equal. That is, we can view these as instances of sums of HyperKron models.

Let us elaborate on this point. Consider the following initiator Kronecker matrix, and it's second power:

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

Next consider the following symmetric HyperKron initiator tensor:

$$\underline{\mathbf{P}} = \left[\begin{array}{cc|cc} a & b & b & c \\ b & c & c & d \end{array} \right],$$

and it's 2nd power:

$$\underline{\mathbf{P}}^2 = \left[\begin{array}{cccc|cccc|cccc|cccc} aa & ab & ba & bb & ab & ac & bb & bc & ba & bb & ca & cb & bb & bc & cb & cc \\ ab & ac & bb & bc & ac & ad & bc & bd & bb & bc & cb & cc & bc & bd & cc & cd \\ ba & bb & ca & cb & bb & bc & cb & cc & ca & cb & da & db & cb & cc & db & dc \\ bb & bc & cb & cc & bc & bd & cc & cd & cb & cc & db & dc & cc & cd & dc & dd \end{array} \right].$$

Each element which has two indices equal is bold in $\underline{\mathbf{P}}^2$. Notice that if the expectation of an edge in \mathbf{P}^2 is E , then the expectation of that edge from the bold items in $\underline{\mathbf{P}}^2$ is $3E$. For example, the expectation of placing edge $(1, 2)$ in \mathbf{P}^2 is

$$\mathbf{P}_{21}^2 + \mathbf{P}_{12}^2 = ac + ab.$$

And in $\underline{\mathbf{P}}^2$ it is

$$\underline{\mathbf{P}}_{112}^2 + \underline{\mathbf{P}}_{121}^2 + \underline{\mathbf{P}}_{211}^2 + \underline{\mathbf{P}}_{122}^2 + \underline{\mathbf{P}}_{212}^2 + \underline{\mathbf{P}}_{221}^2 = 3ac + 3ab.$$

This shouldn't be surprising. Given edge indices (i, j) , the number of times it appears in $\underline{\mathbf{P}}^k$ is the number of multiset permutations of i, j of length 3, which is 6 total. So, we can think of a graph generated by a Kronecker model as a subset of edges generated from a HyperKron model.

To summarize, given a real-world network the overall procedure for this improved fitting is:

1. Determine which edges are involved in triangles, and which are not.

2. Fit the set of triangles to a perturbed HyperKron model.
3. Fit the set of edges which are not involved in triangles to a perturbed Kronecker model.
4. After generating samples of each of these, and adding them together, bring the total number of edges up to the correct expected number of edges by adding in an Erdős-Rényi sample.

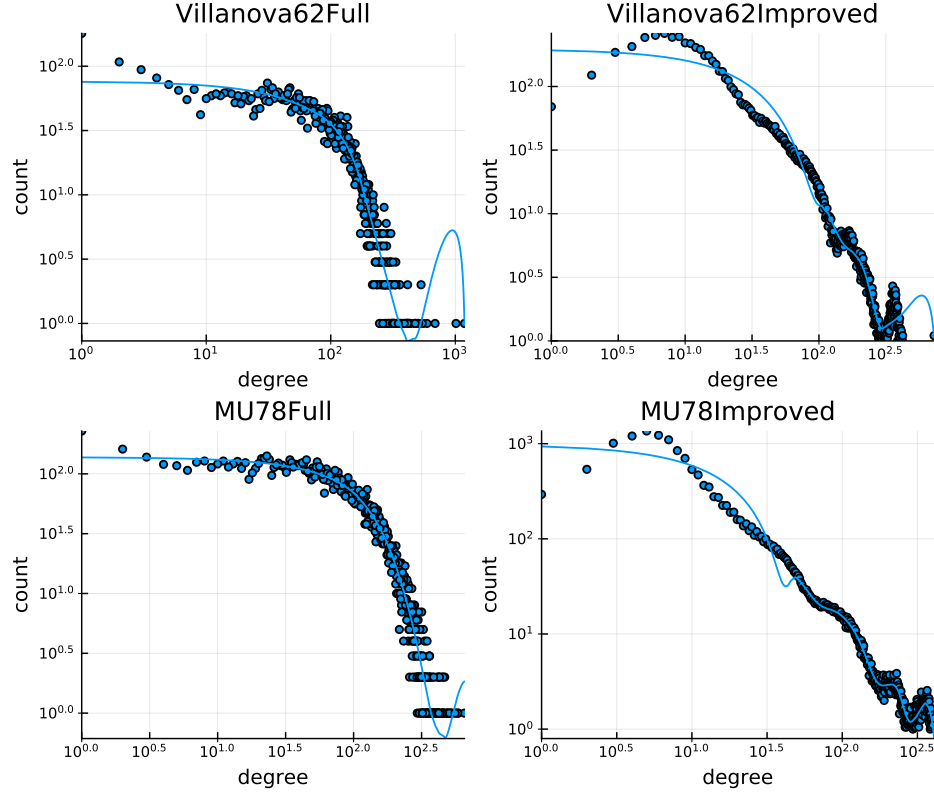


Figure 4.10. **Improved degree distribution in HyperKron.** Improvements to the HyperKron model eliminates two-tailed behavior and almost entirely removes oscillation and improves the fit to data. See the text for details.

Figure 4.10 gives the degree distributions of the full original network data in log-scale, along with the improved fitting. For Villanova62, the HyperKron noise (σ) was set to 0.15, and the Kronecker noise was set to 0.1. For MU78 the HyperKron noise was set to .2 and the Kronecker noise was set to 0.05. The two tailed behavior is eliminated by fitting to the non-triangle edges, and the oscillation behavior is almost entirely removed by the noise. In both cases, the fittings retain non-trivial clustering coefficients.

4.6 Model flexibility

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

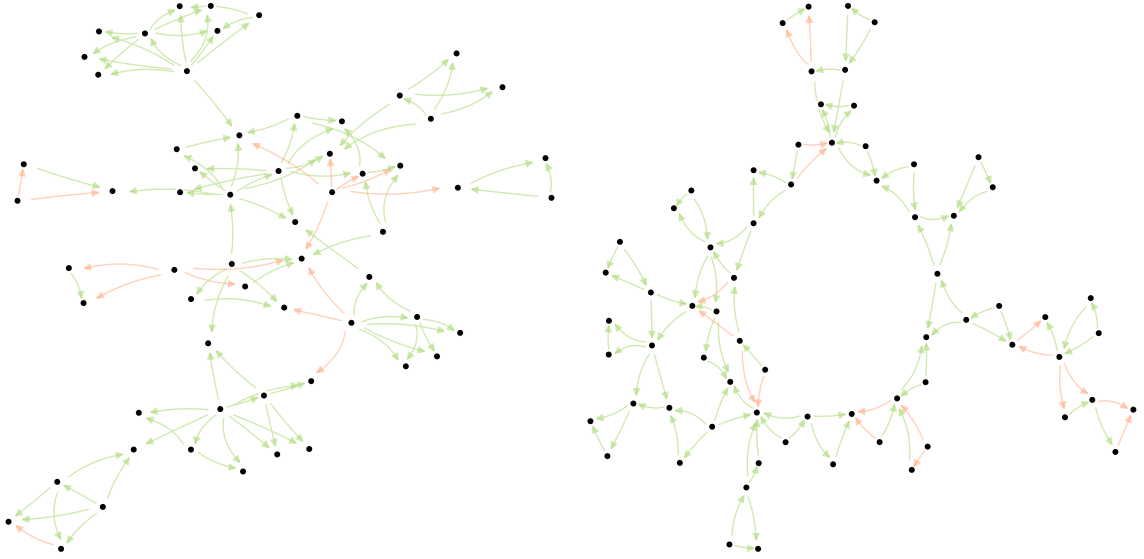


Figure 4.11. **HyperKron fit to yeast network.** On the left, a network drawing of the nodes involved in the feed forward loops in *S. cerevisiae*. Green edges denote promotion (positive sign) and orange edges denote repression (negative sign). On right, a network drawing of the HyperKron model described in the text.

For instance, the *S. cerevisiae* transcription regulatory network is a directed, signed graph that describes promotion or repression of gene expression in the common yeast organism. Coherent feed-forward loops are an important higher-order structure in this network [Milo et al., 2002]. We extract all nodes involved in coherent feed forward loops, leaving a network with 61 nodes and 108 directed, signed edges (92 positive, 16 negative). By manually tweaking entries to get the number of edges to match, we generated HyperKron model using a $2 \times 2 \times 2$ tensor with parameters:

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

and $r = 7$. We associated each hyperedge with one of the four coherent feed-forward loops based on a biased random choice. More specifically:

- the type 1, all positive motif, had probability $1/2$

- the type 2 motif had probability $1/4$
- the type 3 and 4 motifs had probability $1/8$

See Milo et al. [2002] for more about these types. The probabilities were chosen in this way because the real network doesn't have any type 3 and 4 feed-forward loops. When we assemble the motifs placed via these hyperedges into a network, any two motifs that share an edge with the same direction will be coalesced by summing the signs. The largest connected component of the resulting network had 69 nodes and 108 directed, signed edges (90 positive, 18 negative).

We show graph drawings of the two networks in Figure 4.11. The real network has 38 coherent feed forward loops and 2 incoherent feed forward loops. The HyperKron model has 36 coherent feed forward loops and 1 incoherent feed forward loop. Note that the presence of incoherent feed forward loops is an emergent behavior because we only ever generated coherent loops. In this case, we might ask if finding 2 incoherent feed forward loops in the real network is likely to occur or not. By generating 10000 instances of our model, we find at least 2 incoherent feed forward loops around 1006 times (roughly 10%). Consequently, the presence of these two loops in the real data could easily have occurred by chance.

4.7 Summary

Graph models have long been used in lieu of real data which can be expensive and hard to come by. A common class of models constructs a matrix of probabilities, and samples an adjacency matrix by flipping a weighted coin for each entry. Examples include the Erdős-Rényi model, Chung-Lu model, and the Kronecker model. In this chapter we presented the HyperKron Graph model: an extension of the Kronecker Model, but with a distribution over hyperedges. We prove that we can efficiently generate graphs from this model in order proportional to the number of edges times a small log-factor, and find that in practice the runtime is linear with respect to the number of edges. We illustrate a number of useful features of the HyperKron model including non-trivial clustering and highly skewed degree distributions. Finally, we fit the HyperKron model to real-world networks, and demonstrate the model's flexibility with a complex application of the HyperKron model to networks with coherent feed-forward loops.

5. THE TRIANGLE GENERALIZED PREFERENTIAL ATTACHMENT MODEL

As introduced in Section 2.2.2, the idea of preferential attachment (PA) has a lengthy history in explaining “rich-get-richer” models [Yule, 1925, Price, 1976]. In the context of networks, a preferential attachment model suggests that when agents join a network, they form links to existing nodes with large degrees. These models offer a simple local rule that helps explain the presence of highly-skewed or power-law degree distributions in real-world networks [Barabási and Albert, 1999]. While a simple and compelling mathematical model, there are weaknesses in the relationship between PA models and real-world data. One of the most striking is the lack of clustering in PA network models. Consequently, there has been a line of work on generalized PA models that include ways to address the lack of clustering. First, Holme and Kim [Holme and Kim, 2002] proposed a triangle PA model, where agents arrive and link to a node based on its degree and also link to a neighbor of that node to form a triangle. Later, Ostroumova [Ostroumova et al., 2013] generalized a family of PA models and showed that they had power-law degree distributions and high-clustering.

The work here follows in this vein, although we adapt a slightly different notion of a triangle PA model that builds on a recent proposal to show how preferential attachment could give a power-law with any exponent [Avin et al., 2017]. The specific *Triangle Generalized Preferential Attachment* model we use has two slightly different forms as explained in Section 5.3. The two forms are used to greatly simplify the analysis of the resulting properties. We do not believe there to be qualitative differences between them. Formally, we show that these models have a power-law in the degree distribution (Theorem 5.4.1, Corollary 5.5.2) as well as a power-law in the eigenvalues of the adjacency matrix (Theorem 5.4.2).

We also find empirically that our TGPA model has higher-order clustering in terms of higher-order clique closures [Yin et al., 2018] that is characteristic of real-world data (Section 4.5.1).

Our interest in the TGPA model stems from the findings on the reliable presence of power-laws in the eigenvalue spectrum of the adjacency matrix [Eikmeier and Gleich, 2017], presented in Chapter 3. Recall that we found that real-world networks of a variety of types were more likely to have a statistically significant power-law in the eigenvalues of the adjacency matrix than in the degree distribution. This observation presents a simple question, might this behavior be expected in light of how real-world network data are collected? To be specific, real-world network data reflect two types of sampling artifacts. They are often built from a process run on a larger dataset. Consider how

web and social networks are often *crawled* by programs that use breadth-first or related crawling strategies. Second, the crawled data itself represents a sample of some underlying (and unknown) latent network [Schoenebeck, 2013]. Again, note that the social links on networks such as Facebook and Twitter only represent a sample of some unobserved *true* social network. Because of the way that individuals join these networks, forest-fire models are often used to simulate this type of artifact.

Consequently, we study how often *samples* of TGPA models have statistically significant power laws in their degrees and eigenvalues (Section 5.7). These results (Figure 5.2) offer compelling evidence that the eigenvalues of the adjacency matrix robustly indicate the presence of a power-law, with more reliability than the degrees. It should be noted that the presence (or lack thereof) of power-laws in real world data has been often debated [Meusel et al., 2015, Gjoka et al., 2010, Broido and Clauset, 2018]. For that reason, we study models where they are unambiguously present. Although other PA models have the needed property of power-laws in both spectra and degrees, we find that the differences in behavior between the sampled eigenvalues and degrees are less clear than in TGPA.

In summary, the primary contributions of this chapter are:

1. We extend the results presented on the Generalized Preferential Attachment Model (in [Avin et al., 2017]), to show the eigenvalues follow a power-law distribution. (Section 5.2)
2. We present the Triangle Generalized Preferential Attachment Model (TGPA): A model which imposes higher order structure directly into the network. (Section 5.3)
3. We conduct extensive analysis of TGPA to show that the degrees follow a power-law distribution with an exponent which can range between $(1, \infty)$ (Section 5.5), and that the eigenvalues follow a power-law distribution. (Section 5.4)
4. We use TGPA to support a conjecture on why power-laws are observed more often in spectra of networks, and study the results of perturbing the TGPA model. (Section 5.7)

5.1 Related Work

5.1.1 Generalized Preferential Attachment

The Generalized Preferential Attachment Model (GPA) was defined by Avin et al. [2017]. In this model, in addition to adding new vertices and edges, there is also an option in each time step of adding a new *component*. Furthermore, the parameters may change over time, if desired. Start with an arbitrary initial non-empty graph G_0 . For time $t \geq 1$, the graph G_t is constructed by performing either a *node event* with probability $p_t \in [0, 1]$, an *edge event* with probability $r_t \in [0, 1 - p_t]$, or a *component event* with probability $q_t = 1 - p_t - r_t$. In a node event, a new vertex v is added to the graph, along with an edge (u, v) where u is chosen from G_{t-1} with probability $\gamma_t(u)$. In an

edge event, a new edge (u, w) is added, with u and w both nodes in G_{t-1} , and they are chosen with probability $\gamma_t(u) \cdot \gamma_t(w)$. And in a component event, two new nodes v_1, v_2 are added along with edge (v_1, v_2) . Exactly one edge is added at each time step, so the number of edges in G_t is equal to $e_0 + t$. An example GPA graph is shown in Figure 5.1.

The key difference of this model defined by Avin et al. [2017] over the PA model discussed in Section 2.2.2 is the ability to add new components to the graph. To elaborate, in a traditional PA model, the graph at every time step is one large connected component. In GPA, at some time steps new small components will be created. Depending on the parameters of the model, much of the graph may still end up in one large component by the end of the graph evolution, but there is opportunity for different structure. In Avin et al. [2017], it is proved that the degree distribution follows a power-law. In Section 5.2 we further prove that the eigenvalues follow a power-law distribution.

We will also work with a slight variation of the GPA model, along the lines of the alternate version of the PA model defined in Flaxman et al. [2005], Barabási and Albert [1999] and discussed in Section 2.2.2. Start with an empty graph. At time $t = 1, 2, \dots$ do one of the following: With probability p add a new vertex v_t and an edge from v_t to some other vertex in u where u is chosen with probability

$$Pr[u = v_i] = \begin{cases} \frac{d_t(v_i)}{2t-1}, & \text{if } v_i \neq v_t \\ \frac{1}{2t-1}, & \text{if } v_i = v_t \end{cases}; \quad (5.1)$$

And with probability $1 - p$ add two new vertices and an edge between them. For some constant m , every m steps contract the most recent m vertices added through the PA step to form a super vertex. Notice that equation (5.1) is not quite the same as γ_t in equation (2.1). equation (5.1) allows for nodes to be added with self loops. In both versions loops are allowed in the edge step. Regardless, the allowance of self loops has little effect as the graph becomes large, and we remove all self-loops in our final graph for experimental analysis.

5.1.2 Triad Formation

Holme and Kim [2002] introduced a Triad Formation step into the BA version of the PA model (see Section 2.2.2). To be more specific, each time step t has two pieces. First, conduct a PA step in which a new vertex v is added and an edge (v, u) is added with probability given in equation (5.1). Second, with probability p_t add another edge which will create a triangle. In the case that the edge will be added, choose a neighbor of u , u_2 uniformly at random and add edge (v, u_2) . As described in Section 2.2.2, this model also contracts every m steps, for some constant m .

An example network sample is shown in Figure 5.1 under ‘Holme’. The average number of triad closures per added vertex is $m_t = (m - 1)p_t$. It is shown in Holme and Kim [2002] that the network

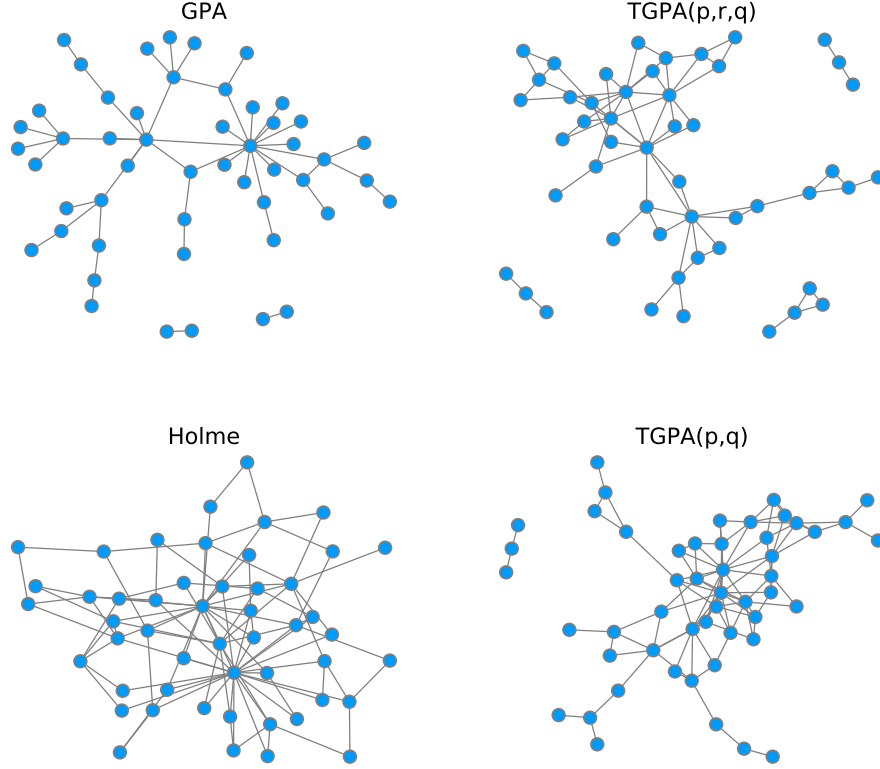


Figure 5.1. **Example TGPA networks compared to existing models.** Each graph has 50 nodes. The top two figures were generated using $p = 0.8, r = 0.1, q = 0.1$. The graphs on the bottom were generated using $m = 2$, and $\text{TGPA}(p, q)$ used $p = 0.85$. See the text for the details on these parameters.

follows a power-law in the degrees with an exponent of 3, and has clustering coefficients which can be tuned by the parameter m_t . Our model incorporates something very similar to this triad formation, but with less regular structure due to an added component step, and with a larger range of possible power-law exponents. See Sections 5.3, 5.5.

5.2 Eigenvalue Power-law in GPA

In this section, we present results for the Generalized Preferential Attachment model presented in [Avin et al., 2017] and discussed in Section 5.1.1. Our result relates to the distribution of the eigenvalues of a graph formed in the model. Note that in order to get our desired result that the eigenvalues follow a power-law distribution (Theorem 5.2.2), we also prove that the degree distribution has a power-law distribution (Theorem 5.2.1). The degree power-law was already proven in [Avin

et al., 2017], but the version of our proof is useful in order to obtain Theorem 5.2.2. The results presented here and the proofs follow closely those presented in Flaxman et al. [2005].

We use the variation of the GPA model which has the parameter m , for the number of nodes which contract together. (See Section 5.1.1.) In Lemma 5.2.1, we prove a helpful bound on the degree of a node at any time. The proof steps of Lemma 5.2.1 are nearly identical to those in Flaxman et al. [2005], except that a factor of p appears early on, because the probability of a node s increasing in degree at time t is $p \cdot d_t(s)/(2t - 1)$. In Lemma 5.2.2, we prove a bound on how much a set of nodes will increase in degree over time. Again, the difference in this proof versus the one in Flaxman et al. [2005] is the probability of a node increasing in degree, which adds a p factor to the analysis and result.

Theorem 5.2.1 is a bound on the largest degrees. The factor of $t^{p/2}$ in Theorem 5.2.1 implies a power-law distribution in the largest degrees with exponent $\beta = (2 + p)/p$. This is calculated using a martingale argument, as described in van der Hofstad [2016] for instance. β comes from solving

$$\frac{p}{2} = \frac{1}{\beta - 1}$$

Notice that depending on the value chosen for p , we can obtain a power-law fit with exponents ranging between 3 and ∞ . The statement of Theorem 5.2.1 also uses the phrase whp. What we mean by this is that as $t \rightarrow \infty$, the probability of the result occurring goes to 1.

The final result, Theorem 5.2.2, relates maximum eigenvalues and maximal degrees in the GPA model. It is similar to results found in Mihail and Papadimitriou [2002], Chung et al. [2003a,b], Flaxman et al. [2005]. It says that if the degrees follows a power-law with exponent β , then the spectra follows a power-law as well. As t goes to infinity, the exponent of the spectral power-law will be $2\beta - 1$, as argued in Chung et al. [2003a]. Again, the proofs of both of these theorems are structurally very similar to the ones in Flaxman et al. [2005], with the added p parameter.

Fix parameter p . Denote G_t^m as the Generalized Preferential Attachment Graph at time t with contractions of size m .

Lemma 5.2.1 *Let $d_t(s)$ be the degree of vertex s in G_t , for any time t after s has been added to the graph. Let $a^{(k)} = a(a + 1)(a + 2) \cdots (a + k - 1)$ be the rising factorial function. Let s' be the time at which node s arrives in the graph. Then for any positive integer k ,*

$$\mathbb{E}[(d_t(s))^{(k)}] \leq (2m)^{(k)} 2^{pk/2} \left(\frac{t}{s'} \right)^{pk/2}$$

Proof Denote G_t^m as the graph at time t with contractions of size m . Let $Z_t = d_t^m(s)$ be the degree of vertex s at time t , and Y_t an indicator for the event that the edge added at time t is incident to s . Then we can write the expectation of Z_t in terms of Z_{t-1} :

$$\begin{aligned}\mathbb{E}[Z_t^{(k)}] &= \mathbb{E}[\mathbb{E}[(Z_{t-1} + Y_t)^{(k)} | Z_{t-1}]] \\ &= \mathbb{E}\left[Z_{t-1}^{(k)} \left(1 - p \cdot \frac{Z_{t-1}}{2t-1}\right) + (Z_{t-1} + 1)^{(k)} \left(p \cdot \frac{Z_{t-1}}{2t-1}\right)\right] \\ &= \mathbb{E}[Z_{t-1}^{(k)}] \left(1 + \frac{pk}{2t-1}\right)\end{aligned}\tag{5.2}$$

where the last equality is an easy algebra step. Apply this relationship iteratively, down to the time when node s was added (recall we denoted that time as s'). Also note that the degree of s at time s' is bounded by $2m$ (if all m edges were added as self loops). Thus:

$$\mathbb{E}(Z_t^{(k)}) = \prod_{t'=s'+1}^t \left(1 + \frac{pk}{2t'-1}\right) \leq (2m)^{(k)} \prod_{t'=s'+1}^t \left(1 + \frac{pk}{2t'-1}\right)\tag{5.3}$$

Use $1 + x \leq e^x$ to write the product as a sum, and bound the sum with an integral:

$$\sum_{t'=s'+1}^t \frac{1}{t'-1/2} \leq \int_{x=s'}^t \frac{1}{x-1/2} dx = \log \frac{t-1/2}{s'-1/2}.\tag{5.4}$$

So finally,

$$\begin{aligned}\mathbb{E}(Z_t^{(k)}) &\leq (2m)^{(k)} \left(\frac{t-1/2}{s'-1/2}\right)^{pk/2} \\ &= (2m)^{(k)} \left(\frac{t}{s'}\right)^{pk/2} \left(\frac{2-1/t}{2-1/s'}\right)^{pk/2} \\ &\leq (2m)^{(k)} \left(\frac{t}{s'}\right)^{pk/2} 2^{pk/2}.\end{aligned}\tag{5.5}$$

■

Now define a *supernode* to be a collection of nodes viewed as one. The degree of a supernode is the sum of the degrees of the vertices in the supernode.

Lemma 5.2.2 *Let $S = (S_1, S_2, \dots, S_l)$ be a disjoint collection of supernodes at time t_0 . Assume that the degree of S_i at time t_0 is $d_{t_0}(S_i) = d_i$. Let t be a time later than t_0 . Let $p_S(\mathbf{r}; \mathbf{d}, t_0, t)$ be the probability that each supernode S_i has degree $r_i + d_i$ at time t . Let $d = \sum_{i=1}^l d_i, r = \sum_{i=1}^l r_i$. If $d = o(t^{1/2})$ and $r = o(t^{2/3})$, then*

$$p_S(\mathbf{r}; \mathbf{d}, t_0, t) \leq \left(\prod_{i=1}^l \binom{r_i + d_i - 1}{d_i - 1}\right) \left(\frac{t_0 + 1}{t}\right)^{pd/2} \exp\left\{2 + t_0 - \frac{pd}{2} + \frac{3pr}{t^{p/2}}\right\}$$

Proof Let $\tau^{(i)} = (\tau_1^{(i)}, \dots, \tau_{r_i}^{(i)})$, where $\tau_j^{(i)}$ is the time when we add an edge incident to S_i and increase the degree of S_i from $d_i + j - 1$ to $d_i + j$. Define $\tau = (\tau_0, \tau_1, \dots, \tau_{r+1})$ to be the ordered

union of $\tau^{(i)}$, with $\tau_0 = t_0$ and $\tau_{r+1} = t$. Let $p(\tau; \mathbf{d}, t_0, t)$ be the probability that supernodes S_i increase in degree at exactly the times specified by τ between time t_0 and t :

$$\begin{aligned}
 p(\tau; \mathbf{d}, t_0, t) &= \left(\prod_{i=1}^l \prod_{k=1}^{r_i} p \frac{d_i + k - 1}{2\tau_k^{(i)} - 1} \right) \left(\prod_{k=0}^r \prod_{j=\tau_k+1}^{\tau_{k+1}-1} \left(1 - p \frac{d+k}{2j-1} \right) \right) \\
 &\quad \text{for each supernode } S_i, \text{ the prob. of } \tau \text{ aligning with } \tau^{(i)}. \quad \text{for each timestep inbetween the relevant ones, the probability of picking any edge outside of } S_1, \dots, S_l. \\
 &= \left(\prod_{i=1}^l \frac{(r_i + d_i - 1)!}{(d_i - 1)!} \right) \left(\prod_{k=1}^r \frac{p}{2\tau_k - 1} \right) \exp \left\{ \sum_{k=0}^r \sum_{j=\tau_k+1}^{\tau_{k+1}-1} \log \left(1 - p \left(\frac{d+k}{2j-1} \right) \right) \right\}
 \end{aligned} \tag{5.6}$$

Now we can bound the inner most sum of the exponential term.

$$\sum_{j=\tau_k+1}^{\tau_{k+1}-1} \log \left(1 - p \left(\frac{d+k}{2j-1} \right) \right) \leq \sum_{j=\tau_k+1}^{\tau_{k+1}-1} \log \left(1 - \frac{p(d+k)}{2j} \right) \tag{5.7}$$

which is less than or equal to

$$\begin{aligned}
 \int_{\tau_k+1}^{\tau_{k+1}} \log \left(1 - \frac{p(d+k)}{2x} \right) dx &= -\tau_{k+1} \log(2\tau_{k+1}) + (\tau_k + 1) \log(2\tau_k + 2) \\
 &\quad + 1/2(2\tau_{k+1} - p(d+k)) \log(2\tau_{k+1} - p(d+k)) \\
 &\quad - 1/2(2\tau_k + 2 - p(d+k)) \log(2\tau_k + 2 - p(d+k)).
 \end{aligned} \tag{5.8}$$

Note that $\tau_0 = t_0$ and $\tau_{r+1} = t$. We can write

$$\sum_{k=0}^r \int_{\tau_k+1}^{\tau_{k+1}} \log \left(1 - \frac{p(d+k)}{2x} \right) dx = A + \sum_{k=1}^r B_k \tag{5.9}$$

where

$$\begin{aligned}
 A &= (t_0 + 1) \log(2t_0 + 2) - 1/2(2t_0 + 2 - pd) \log(2t_0 + 2 - pd) \\
 &\quad - t \log(2t) + 1/2(2t - p(d+r)) \log(2t - p(d+r))
 \end{aligned} \tag{5.10}$$

and

$$\begin{aligned}
 B_k &= \tau_k \log(1 + 1/\tau_k) + \log(2\tau_k + 2) - \frac{2-p}{2} \log(2\tau_k + p - p(d+k)) \\
 &\quad + 1/2(2\tau_k + 2 - p(d+k)) \log \left(1 - \frac{2-p}{2\tau_k + 2 - p(d+k)} \right).
 \end{aligned} \tag{5.11}$$

We will bound each of A and B_k , starting with B_k . Since $1 + x \leq e^x$, $\tau_k \log(1 + 1/\tau_k) \leq 1$, and $\frac{1}{2}(2\tau_k + 2 - p(d+k)) \log \left(1 - \frac{2-p}{2\tau_k + 2 - p(d+k)} \right) \leq -1 + p/2$. Rearranging the other two terms of equation (5.11) and combining with these inequalities we get

$$B_k \leq \frac{p}{2} \log(2\tau_k + 2) - \frac{2-p}{2} \log \left(1 - \frac{p(d+k) + 2-p}{2\tau_k + 2} \right) + \frac{p}{2}. \tag{5.12}$$

Now rearranging terms of A from equation (5.10),

$$\begin{aligned}
A &= -(t_0 + 1) \log \left(1 - \frac{pd}{2t_0 + 2} \right) + \frac{pd}{2} \log(2t_0 + 2 - pd) \\
&\quad + t \log \left(1 - \frac{p(d+r)}{2t} \right) - \frac{p(d+r)}{2} \log(2t - p(d+r)) \\
e^A &= \left(1 - \frac{pd}{2t_0 + 2} \right)^{-(t_0+1)} (2t_0 + 2 - pd)^{pd/2} \left(1 - \frac{p(d+r)}{2t} \right)^t (2t - p(d+r))^{\frac{-p(d+r)}{2}} \\
&= \left(1 - \frac{pd}{2t_0 + 2} \right)^{-(1 - \frac{pd}{2(t_0+1)})(t_0+1)} \left(1 - \frac{p(d+r)}{2t} \right)^{t - \frac{p(d+r)}{2}} \left(\frac{t_0 + 1}{t} \right)^{\frac{pd}{2}} (2t)^{\frac{-pr}{2}}.
\end{aligned} \tag{5.13}$$

Using the bound $1 - x \leq e^{-x - x^2/2}$ for $0 < x < 1$,

$$\left(1 - \frac{p(d+r)}{2t} \right)^{t - p(d+r)/2} \leq \exp \left\{ -\frac{p(d+r)}{2} + \frac{p^2(d+r)^2}{8t} + \frac{p^3(d+r)^3}{16t^2} \right\} \tag{5.14}$$

Putting the bounds on A and B_k together, we get

$$\begin{aligned}
e^{A + \sum B_k} &\leq \left(1 - \frac{pd}{2t_0 + 2} \right)^{-(1 - \frac{pd}{2(t_0+1)})(t_0+1)} \exp \left\{ -\frac{p(d+r)}{2} + \frac{p^2(d+r)^2}{8t} + \frac{p^3(d+r)^3}{16t^2} \right\} \\
&\quad \times \left(\frac{t_0 + 1}{t} \right)^{pd/2} (2t)^{-pr/2} \prod_{k=1}^r \left(\left(1 - \frac{p(d+k) + 2 - p}{2\tau_k + 2} \right)^{-(2-p)/2} (2\tau_k + 2)^{p/2} \right) e^{pr/2}.
\end{aligned} \tag{5.15}$$

Using

$$\text{err}(r, d, t_0, t) = \left(1 - \frac{pd}{2t_0 + 2} \right)^{-(1 - \frac{pd}{2(t_0+1)})(t_0+1)} \exp \left\{ -\frac{pd}{2} + \frac{p^2(d+r)^2}{8t} + \frac{p^3(d+r)^3}{16t^2} \right\}, \tag{5.16}$$

we can write equation (5.15) as

$$\text{err}(r, d, t_0, t) \left(\frac{t_0 + 1}{t} \right)^{\frac{pd}{2}} (2t)^{\frac{-pr}{2}} \prod_{k=1}^r \left(\left(1 - \frac{p(d+k) + 2 - p}{2\tau_k + 2} \right)^{\frac{-(2-p)}{2}} (2\tau_k + 2)^{\frac{p}{2}} \right). \tag{5.17}$$

So we finally finish with the bound on $p(\tau; \mathbf{d}, t_0, t)$ by substituting equation (5.15) into equation (5.6):

$$\begin{aligned}
p(\tau; \mathbf{d}, t_0, t) &\leq \left(\prod_{i=1}^l \frac{(r_i + d_i - 1)!}{(d_i - 1)!} \right) \text{err}(r, d, t_0, t) \left(\frac{t_0 + 1}{t} \right)^{\frac{pd}{2}} (2t)^{\frac{-pr}{2}} \\
&\quad \times \prod_{k=1}^r \left(\left(1 - \frac{p(d+k) + 2 - p}{2\tau_k + 2} \right)^{\frac{-(2-p)}{2}} (2\tau_k + 2)^{\frac{p}{2}} \frac{p}{2\tau_k - 1} \right),
\end{aligned} \tag{5.18}$$

which can be re-arranged as

$$\begin{aligned}
&= \left(\prod_{i=1}^l \frac{(r_i + d_i - 1)!}{(d_i - 1)!} \right) \text{err}(r, d, t_0, t) \left(\frac{t_0 + 1}{t} \right)^{pd/2} (2t)^{-pr/2} \\
&\quad \times \prod_{k=1}^r \left(p(2\tau_k + p - p(d+k))^{-(2-p)/2} \left(1 + \frac{3}{2\tau_k - 1} \right) \right).
\end{aligned} \tag{5.19}$$

Now, we will sum $p(\tau; \mathbf{d}, t_0, t)$ over all ordered choices of τ .

$$\begin{aligned}
p(\mathbf{r}; \mathbf{d}, t_0, t) &\leq \sum_{\tau^{(1)}, \dots, \tau^{(l)}} p(\tau; \mathbf{d}, t_0, t) \\
&\leq \binom{r}{r_1, \dots, r_l} \sum_{t_0+1 \leq \tau_1 < \dots < \tau_r \leq t} \prod_{i=1}^l \frac{(r_i + d_i - 1)!}{(d_i - 1)!} \text{err}(r, d, t_0, t) \left(\frac{t_0 + 1}{t} \right)^{\frac{pd}{2}} \\
&\quad \times (2t)^{-\frac{pr}{2}} p \prod_{k=1}^r (2\tau_k + p - p(d+k))^{-(2-p)/2} \left(1 + \frac{3}{2\tau_k - 1} \right) \\
&= r! \left(\prod_{i=1}^l \binom{r_i + d_i - 1}{d_i - 1} \right) \text{err}(r, d, t_0, t) \left(\frac{t_0 + 1}{t} \right)^{pd/2} (2t)^{-pr/2} \\
&\quad \times \sum_{t_0+1 \leq \tau_1 < \dots < \tau_r \leq t} p \prod_{k=1}^r (2\tau_k + p - p(d+k))^{-(2-p)/2} \left(1 + \frac{3}{2\tau_k - 1} \right)
\end{aligned} \tag{5.20}$$

Now let $\tau'_k = \tau_k - \lceil p(d+k)/2 \rceil$. Since $d \geq 1$ and $k \geq 1$, we have $2\lceil p(d+k)/2 \rceil \geq 2$. So, the last term in equation (5.20) is less than or equal to

$$\begin{aligned}
&\sum_{(t_0 - p\lceil d/2 \rceil + 1) \leq \tau'_1 \leq \dots \leq \tau'_r \leq (t - p\lceil (d+r)/2 \rceil)} \left(p \prod_{k=1}^r (2\tau'_k + p)^{-(2-p)/2} \left(1 + \frac{3}{2\tau'_k + 1} \right) \right) \\
&\leq \frac{p}{r!} \left(\sum_{\tau' = (t_0 - p\lceil d/2 \rceil + 1)}^{t - p\lceil (d+r)/2 \rceil} \left((2\tau' + p)^{-(2-p)/2} + 3(2\tau' + 1)^{-(4-p)/2} \right) \right)^r \\
&\leq \frac{p}{r!} \left(\int_0^{t - p\lceil (d+r)/2 \rceil} \left((2x + p)^{-(2-p)/2} + 3(2x + 1)^{-(4-p)/2} \right) dx \right)^r \\
&\leq \frac{p}{r!} \left(\frac{1}{p} (2t - p(d+r) + p)^{p/2} - p^{(p-2)/2} - \frac{3}{2-p} (2t - p(d+r) + 1)^{-(2-p)/2} + \frac{3}{2-p} \right)^r \\
&\leq \frac{p}{r!} \left(\frac{1}{p} (2t - p(d+r) + p)^{p/2} + 3 \right)^r \\
&= \frac{p}{r!} \left(\frac{1}{p} (2t)^{p/2} \left(1 - \frac{p(d+r) - p}{2t} \right)^{p/2} \left(1 + \frac{3p}{(2t - p(d+r) + p)^{p/2}} \right) \right)^r \\
&\leq \frac{1}{r!} (2t)^{pr/2} \underbrace{\left(1 - \frac{p(d+r) - p}{2t} \right)^{pr/2}}_{\leq \exp\left\{ -\frac{rp(p(d+r)-p)}{4t} \right\}} \underbrace{\left(1 + \frac{3p}{(2t - p(d+r) + p)^{p/2}} \right)^r}_{\leq \exp\left\{ \frac{3pr}{(2t - p(d+r) + p)^{p/2}} \right\}},
\end{aligned} \tag{5.21}$$

where the last inequalities come from $1 + x \leq e^x$. So finally,

$$\begin{aligned}
p_S(\mathbf{r}; \mathbf{d}, t_0, t) &\leq \left(\prod_{i=1}^l \binom{r_i + d_i - 1}{d_i - 1} \right) \text{err}(r, d, t_0, t) \left(\frac{t_0 + 1}{t} \right)^{pd/2} \\
&\quad \times \exp \left\{ \frac{-rp((d+r) - p)}{4t} + \frac{3pr}{(2t - p(d+r) + p)^{p/2}} \right\}.
\end{aligned}$$

Since $d = o(t^{1/2})$ and $r = o(t^{2/3})$,

$$\begin{aligned} & \text{err}(r, d, t_0, t) \exp \left\{ \frac{-rp((d+r)-p)}{4t} + \frac{3pr}{(2t-p(d+r)+p)^{p/2}} \right\} \\ & \leq \left(1 - \frac{pd}{2(t_0+1)} \right)^{-(1+pd/2(t_0+1))(t_0+1)} \exp \left\{ 1 - \frac{pd}{2} - \frac{r^2}{8t} + \frac{3pr}{t^{p/2}} \right\} \\ & \leq \underset{\text{since } x^{-x} \leq e}{e^{(t_0+1)}} \exp \left\{ 1 - \frac{pd}{2} + \frac{3pr}{t^{p/2}} \right\} = \exp \left\{ 2 + t_0 - \frac{pd}{2} + \frac{3pr}{t^{p/2}} \right\} \end{aligned}$$

This concludes the proof. ■

Theorem 5.2.1 *Let m, k be fixed positive integers, and let $f(t)$ be a function with $f(t) \rightarrow \infty$ as $t \rightarrow \infty$. Let $\Delta_1 \geq \Delta_2 \geq \dots \geq \Delta_k$ denote the degrees of the k highest degree vertices of G_t^m . Then*

$$\frac{t^{p/2}}{f(t)} \leq \Delta_1 \leq t^{p/2} f(t) \quad \text{and} \quad \frac{t^{p/2}}{f(t)} \leq \Delta_i \leq \Delta_{i-1} - t^{p/2} f(t)$$

for $i = 1, 2, \dots, k$ with high probability (whp).

Proof Partition the vertices into those added before time t_0 , before time t_1 , and after t_1 , with $t_0 = \log \log \log f(t)$, $t_1 = \log \log f(t)$. We will argue about the maximum degree vertices in each set.

Claim 5.2.1 *In G_t^m , the degree of the supernode of vertices added before time t_0 is at least $t_0^{1/3} t^{p/2}$ whp.*

Proof Consider all vertices added before time t_0 as a supernode. Let A_1 denote the event that this supernode has degree less than $t_0^{1/3} t^{p/2}$ at time t . We will use Lemma 5.2.2 with $l = 1$, and $d = 2t_0$ (because the supernode has all edges at time t_0).

$$\begin{aligned} Pr[A_1] & \leq \sum_{r_1=0}^{t_0^{1/3} t^{p/2} - 2t_0} \binom{r_1 + 2t_0 - 1}{2t_0 - 1} \left(\frac{t_0 + 1}{t} \right)^{pd/2} e^{2+t_0-pd/2+3pr/t^{p/2}} \\ & \leq \sum_{r_1=0}^{t_0^{1/3} t^{p/2} - 2t_0} \binom{t_0^{1/3} t^{p/2} - 1}{2t_0 - 1} \left(\frac{t_0 + 1}{t} \right)^{pt_0} e^{2+t_0-pt_0+3pt_0^{1/3}-6pt_0/t^{p/2}} \\ & \quad \text{By substituting } r_1 = t_0^{1/3} t^{p/2} \\ & = (t_0^{1/3} t^{p/2} - 2t_0) \frac{(t_0^{1/3} t^{p/2} - 1)!}{(2t_0 - 1)!(t_0^{1/3} t^{p/2} - 2t_0)!} \left(\frac{t_0 + 1}{t} \right)^{pt_0} e^{2+t_0(1-p)+3pt_0^{1/3}-\frac{6pt_0}{t^{p/2}}} \\ & \leq t_0^{1/3} t^{p/2} \frac{(t_0^{1/3} t^{p/2})^{2t_0-1}}{(2t_0 - 1)!} \left(\frac{t_0 + 1}{t} \right)^{pt_0} e^{2+t_0(1-p)+3pt_0^{1/3}-6pt_0/t^{p/2}} \\ & \leq t_0^{2t_0/3} \frac{e^{2t_0-1}}{(2t_0 - 1)^{2t_0-1}} (t_0 + 1)^{pt_0} e^{2+t_0(1-p)+3pt_0^{1/3}-6pt_0/t^{p/2}} \underset{\text{since } 1/x! \leq e^x/x^x}{\leq} \frac{e^{1+(3-p)t_0+3pt_0^{1/3}-6pt_0/t^{p/2}}}{(2t_0 - 1)^{t_0(4/3-p)-1}} \end{aligned} \tag{5.22}$$

which goes to 0 as t goes to infinity. Thus A_1 does not hold with high probability, and the claim is proved. ■

Claim 5.2.2 In G_t^m , no vertex added after time t_1 has degree exceeding $t_0^{-2}t^{p/2}$ whp.

Proof Let A_2 denote the event that some vertex added after time t_1 has degree exceeding $t_0^{-2}t^{p/2}$.

$$\begin{aligned}
 \Pr[A_2] &\leq \sum_{s=t_1}^t \Pr[d_t(s) \geq t_0^{-2}t^{p/2}] = \sum_{s=t_1}^t \Pr[(d_t(s))^{(l)} \geq (t_0^{-2}t^{p/2})^{(l)}] \leq \sum_{s=t_1}^t t_0^{2l} t^{-lp/2} \mathbb{E}[(d_t(s))^{(l)}] \\
 &\quad \text{by Markov} \\
 &= \sum_{s=t_1}^t t_0^{2l} t^{-lp/2} (2m)^{(l)} 2^{lp/2} \left(\frac{t}{s}\right)^{lp/2} = 2^{lp/2} (2m)^{(l)} t_0^{2l} \int_{t_1-1}^t x^{-lp/2} dx \\
 &\quad \text{by Lemma 5.2.1}
 \end{aligned} \tag{5.23}$$

We compute the integral in equation (5.23),

$$\int_{t_1-1}^t x^{-lp/2} dx = \left. \frac{x^{-lp/2+1}}{-lp/2+1} \right|_{t_1-1}^t = (-lp/2+1)^{-1} \left(t^{-lp/2+1} - (t_1-1)^{-lp/2+1} \right) \tag{5.24}$$

We want to choose l so that $-lp/2+1$ is less than 0. So choose $l > 2/p$. Then the integral in equation (5.24) is less than or equal to $(lp/2-1)^{-1}(t_1-1)^{-lp/2+1}$, and plugging in the computation from equation (5.24) into equation (5.23),

$$\Pr[A_2] \leq \frac{2^{lp/2} (2m)^{(l)} t_0^{2l}}{(lp/2-1)(t_1-1)^{lp/2-1}} \tag{5.25}$$

which goes to 0 as t increases. ■

Claim 5.2.3 In G_t^m , no vertex added before time t_1 has degree exceeding $t_0^{1/6}t^{p/2}$ whp.

Proof Let A_3 denote the event that some vertex added before time t_1 has degree exceeding $t_0^{1/6}t^{p/2}$.

Using the exact argument as in Claim 5.2.2, $\Pr[A_3]$ goes to 0 as t increases. ■

Claim 5.2.4 The k highest degree vertices of G_t^m are added before time t_1 and have degree Δ_i bounded by $t_0^{-1}t^{p/2} \leq \Delta_i \leq t_0^{1/6}t^{p/2}$

Proof First lets summarize the results of the last three claims:

- Bound on degrees of vertices added after time t_1 : $t_0^{-2}t^{p/2}$
- Bound on degrees of vertices added before time t_1 : $t_0^{1/6}t^{p/2}$
- Sum of all degrees added before time t_0 is at least: $t_0^{1/3}t^{p/2}$

So the upper bound of the claim is immediately clear from the second item. Suppose that the lower bound does not hold. Then one of the top k vertices has degree less than $t_0^{-1}t^{p/2}$ and the total degree of vertices added before time t_0 is bounded by

$$\begin{aligned}
 &\underbrace{(k-1)t_0^{1/6}t^{p/2}}_{\text{largest possible degrees of } (k-1) \text{ vertices}} + \underbrace{\left(\frac{t_0}{m} - k + 1\right) \left(t_0^{-1}t^{p/2}\right)}_{\text{largest possible degrees of remaining vertices}} \leq kt_0^{1/6}t^{p/2} + t_0(t_0^{-1}t^{p/2}) \\
 &= kt_0^{1/6}t^{p/2} + t^{p/2} = t^{p/2}(kt_0^{1/6} + 1) \leq t^{p/2}(2kt_0^{1/6}) \leq t^{p/2}t_0^{1/3}.
 \end{aligned} \tag{5.26}$$

Finally, since we have the lower bound, and we know that $t^{p/2}/t_0 \geq t^{p/2}/t_0^2$, then none of the largest degree vertices could be added after time t_1 . \blacksquare

Claim 5.2.5 *The k highest degree vertices of G_t^m have $\Delta_i \leq \Delta_{i-1} - \frac{t^{p/2}}{f(t)}$ whp.*

Proof Let A_4 denote the event that there are two vertices among the first t_1 time steps with degrees exceeding $t_0^{-1}t^{p/2}$ and within $t^{p/2}/f(t)$ of each other. Let

$$p_{l,s_1,s_2} = \Pr [d_t(s_1) - d_t(s_2) = l \mid \overline{A_3}], \text{ for } |l| \leq t^{p/2}/f(t),$$

where $\overline{A_3}$ means the opposite of event A_3 from Claim 5.2.3. Then

$$\Pr [A_4 | \overline{A_3}] \leq \sum_{1 \leq s_1 < s_2 \leq t_1} \sum_{l=-t^{p/2}/f(t)}^{t^{p/2}/f(t)} p_{l,s_1,s_2} \quad (5.27)$$

Now

$$p_{l,s_1,s_2} \leq \sum_{r_1=t_0^{-1}t^{p/2}}^{t_0^{1/6}t^{p/2}} \sum_{d_1,d_2=1}^{2t_1} p_{(s_1,s_2)}((r_1, r_1-l); (d_1, d_2), t_1, t) \quad (5.28)$$

Notation from Lemma 5.2.2.

Using Lemma 5.2.2,

$$\begin{aligned} &\leq \sum_{r_1=t_0^{-1}t^{p/2}}^{t_0^{1/6}t^{p/2}} \sum_{d_1,d_2=1}^{2t_1} \binom{r_1+d_1-1}{d_1-1} \binom{r_1-l+d_2-1}{d_2-1} \left(\frac{t_1+1}{t}\right)^{\frac{p(d_1+d_2)}{2}} e^{\left\{2+t_1-\frac{p(d_1+d_2)}{2}+\frac{3p(r_1-l)}{t^{p/2}}\right\}} \\ &\leq t_0^{1/6}t^{p/2} \sum_{d_1,d_2=1}^{2t_1} \binom{t_0^{1/6}t^{p/2}+d_1-1}{d_1-1} \binom{t_0^{1/6}t^{p/2}-l+d_2-1}{d_2-1} \left(\frac{t_1+1}{t}\right)^{\frac{p(d_1+d_2)}{2}} e^{\left\{2+t_1-\frac{p(d_1+d_2)}{2}+\frac{3pt_0^{1/6}t^{p/2}}{t^{p/2}}\right\}} \\ &\leq t_0^{1/6}t^{p/2} \sum_{d_1,d_2=1}^{2t_1} \binom{2t_0^{1/6}t^{p/2}}{d_1-1} \binom{2t_0^{1/6}t^{p/2}}{d_2-1} \left(\frac{t_1+1}{t}\right)^{\frac{p(d_1+d_2)}{2}} e^{2+t_1+3pt_0^{1/6}} \\ &\leq t_0^{1/6}t^{p/2} \sum_{d_1,d_2=1}^{2t_1} (2t_0^{1/6}t^{p/2})^{d_1+d_2-2} (t_1+1)^{2pt_1} t^{-p(d_1+d_2)/2} e^{2+t_1+3pt_0^{1/6}} \\ &= t^{-p/2}t_0^{1/6}(2t_1)^{24t_1}t_0^{2t_1/3}(t_1+1)^{2pt_1}e^{2+t_1+3pt_0^{1/6}} \end{aligned} \quad (5.30)$$

Denote the last equation as $h(t)$ and note $h(t)$ is a polynomial in $\log(f(t))$ times a factor of $t^{-p/2}$.

Then going back to equation (5.27),

$$\Pr [A_4 | \overline{A_3}] \leq \binom{t_1}{2} 2 \frac{t^{p/2}}{f(t)} h(t) = \binom{t_1}{2} 2 \frac{\text{poly}(\log(f(t)))}{f(t)} \quad (5.31)$$

which goes to 0 as t increases. This concludes the proof of this final claim. \blacksquare

And this concludes the proof of the theorem. \blacksquare

Theorem 5.2.2 *Let k be a fixed integer, and let $f(t)$ be a function with $f(t) \rightarrow \infty$ as $t \rightarrow \infty$. Let $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k$ be the k largest eigenvalues of the adjacency matrix of G_t . Then for $i = 1, \dots, k$, we have $\lambda_i = (1 + o(1))\Delta_i^{1/2}$, where Δ_i is the i^{th} largest degree.*

We will show, that with high probability, G contains a star forest F , with stars of degree asymptotic to the maximum degree vertices of G . Then we show that $G \setminus F$ has small eigenvalues. Then we can use Rayleigh's principle to say that the large eigenvalues of G cannot be too different than the large eigenvalues of F .

Proof Let S_i be the vertices added after time t_{i-1} and at or before time t_i , for $t_0 = 0, t_1 = t^{1/8}, t_2 = t^{9/16}, t_3 = t$. Let $G = G_t$. We start by finding bounds on the degrees of G .

Claim 5.2.6 *For any $\varepsilon > 0$, and any $f(t)$ with $f(t) \rightarrow \infty$ as $t \rightarrow \infty$ the following holds whp: for all s with $f(t) \leq s \leq t$, for all vertices $v \in G_s$, if v was added at time r , then $d_s(v) \leq s^{p/2+\varepsilon} r^{-p/2}$.*

Proof

$$\begin{aligned} \Pr \left[\bigcup_{s=f(t)}^t \bigcup_{r=1}^s \left\{ d_s^m(r) \geq s^{p/2+\varepsilon} r^{-p/2} \right\} \right] &\leq \sum_{s=f(t)}^t \sum_{r=1}^s \Pr \left[d_s^m(r) \geq s^{p/2+\varepsilon} r^{-p/2} \right] \\ &= \sum_{s=f(t)}^t \sum_{r=1}^s \Pr \left[(d_s^m(r))^{(l)} \geq (s^{p/2+\varepsilon} r^{-p/2})^{(l)} \right] \end{aligned} \quad (5.32)$$

which is bounded using Markov:

$$\leq \sum_{s=f(t)}^t \sum_{r=1}^s s^{-l(p/2+\varepsilon)} r^{lp/2} \mathbb{E} \left[(d_s^m(r))^{(l)} \right]$$

which we can bound using Lemma 5.2.1

$$\leq \sum_{s=f(t)}^t \sum_{r=1}^s s^{-l(p/2+\varepsilon)} r^{lp/2} (2m)^{(l)} 2^{lp/2} \left(\frac{s}{r} \right)^{lp/2} = (2m)^{(l)} 2^{lp/2} \sum_{s=f(t)}^t s^{1-\varepsilon l}$$

Take $l \geq 3/\varepsilon$. Then we can bound the sum by an integral,

$$\sum_{s=f(t)}^t s^{1-\varepsilon l} \leq \int_{f(t)-1}^{\infty} x^{1-\varepsilon l} dx = \frac{1}{2-\varepsilon l} x^{2-\varepsilon l} \Big|_{f(t)-1}^{\infty} = \frac{1}{\varepsilon l - 2} (f(t) - 1)^{2-\varepsilon l} \quad (5.33)$$

which goes to zero as t increases, since $l \geq 3/\varepsilon$. ■

Claim 5.2.7 *Let S'_3 be the set of vertices in S_3 that are adjacent to more than one vertex of S_1 in G . Then $|S'_3| \leq t^{7p/16}$ with high probability.*

Proof Let B_1 be the event that the conditions of Claim 5.2.6 hold with $f(t) = t_2$ and $\varepsilon = 1/16$. Then for a vertex $v \in S_3$ added at time s , the probability that v picks at least one neighbor in S_1 is less than or equal to

$$\frac{\sum_{w \in S_1} d_s(w)}{2s-1} \leq \frac{\sum_{w \in S_1} s^{p/2+\varepsilon}}{2s-1} = \frac{t_1 s^{p/2+\varepsilon}}{2s-1} \quad (5.34)$$

So, the probability of having two or more neighbors in S_1 can be bounded by,

$$\Pr[|N(v) \cap S_1| \geq 2 \mid B_1] \leq \left(\frac{t_1 s^{p/2+\varepsilon}}{2s-1} \right)^2 \cdot \binom{m}{2} \leq m^2 t^{1/4} s^{(-15+8p)/8} \quad (5.35)$$

Let X denote the number of $v \in S_3$ adjacent to more than one vertex of S_1 . Then

$$\begin{aligned} \mathbb{E}[X|B_1] &\leq \sum_{t_2+1}^t m^2 s^{(-15+8p)/8} t^{1/4} \leq m^2 t^{1/4} \int_{t_2}^t x^{(-15+8p)/8} dx \\ &= m^2 t^{1/4} \left[\frac{8}{-7+8p} x^{(-7+8p)/8} \right]_{t_2}^t \leq \frac{8m^2 t^{1/4}}{-7+8p} t^{(-7+8p)/8} \end{aligned} \quad (5.36)$$

Then by Markov,

$$\Pr[X \geq t^{7p/16} | B_1] \leq \frac{\mathbb{E}[X|B_1]}{t^{7p/16}} \leq \frac{8m^2}{8p-7} \frac{t^{p-5/8}}{t^{7p/16}} = \frac{8m^2}{8p-7} \frac{t^{-5/8}}{t^{-9p/16}} \quad (5.37)$$

And $\frac{t^{-5/8}}{t^{-9p/16}} = \frac{t^{9p/16}}{t^{5/8}} \leq \frac{t^{9/16}}{t^{5/8}}$ which goes to zero. ■

Let $F \subseteq G$ be the star forest consisting of edges between S_1 and $S_3 \setminus S'_3$.

Claim 5.2.8 *Let $\Delta_1 \geq \Delta_2 \geq \dots \geq \Delta_k$ denote the degrees of the k highest degree vertices of G . Then $\lambda_i(F) = (1 - o(1))\Delta_i^{1/2}$.*

Proof Denote K_{1,d_i} to be a star of degree d_i . Let H be the star forest $H = K_{1,d_1} \cup \dots \cup K_{1,d_k}$ with $d_1 \geq d_2 \geq \dots \geq d_k$. Then for $i = 1, \dots, k$, $\lambda_i(H) = d_i^{1/2}$. So it will be sufficient to show that $\Delta_i(F) = (1 - o(1))\Delta_i(G)$. Claim 5.2.4 shows that the k highest degree vertices G are added before time t_1 . So these vertices are all in F . The only edges to those vertices that are not in F are those added before time t_2 and those incident to S'_3 .

By Theorem 5.2.1 we can choose $f(t)$ such that, $\Delta_1(G_{t_2}^m) \leq t_2^{p/2} f(t) \leq t^{7p/16}$. Also by Theorem 5.2.1, $\Delta_i(G) \geq t^{p/2} / \log t$. Finally, Claim 5.2.7 says that $|S'_3| \leq t^{7p/16}$ whp. And so, with high probability,

$$\begin{aligned} \Delta_i(F) &\geq \Delta_i(G) - t^{7p/16} - m t^{7p/16} \geq \frac{t^{p/2}}{\log t} - t^{7p/16}(1+m) = \frac{t^{p/2}}{\log t} \left[1 - t^{7p/16}(1+m) \frac{\log t}{t^{p/2}} \right] \\ &= \frac{t^{p/2}}{\log t} \left[1 - (1+m) \frac{\log t}{t^{p/2-7p/16}} \right] = \frac{t^{p/2}}{\log t} \left[1 - (1+m) \frac{\log t}{t^{p/16}} \right] = (1 - o(1))\Delta_i(G) \end{aligned} \quad (5.38)$$
■

Let $H = G \setminus F$. Denote $\mathbf{A}_G, \mathbf{A}_F$ and \mathbf{A}_H to be the adjacency matrices for graphs G, F and H . In the following claim, we'll show that $\lambda_1(\mathbf{A}_H)$ is $o(\lambda_k(\mathbf{A}_F))$. Consider the fact that if \mathbf{A} and $\mathbf{A} + \mathbf{E}$ are symmetric $n \times n$ matrices, then $\lambda_k(\mathbf{A}) + \lambda_n(\mathbf{E}) \leq \lambda_k(\mathbf{A} + \mathbf{E})$ (see for instance theorem 8.1.5 in Golub and Van Loan [2013]). That implies that for any subspace L ,

$$\max_{\mathbf{x} \in L, \mathbf{x} \neq 0} \frac{\mathbf{x}^T \mathbf{A}_G \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = \max_{\mathbf{x} \in L, \mathbf{x} \neq 0} \frac{\mathbf{x}^T \mathbf{A}_F \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \pm O \left(\max_{\mathbf{x} \neq 0} \frac{\mathbf{x}^T \mathbf{A}_H \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \right).$$

That will be enough to finish the proof because by Rayleigh's Principle [Golub and Van Loan, 2013], $\lambda_i(\mathbf{A}_G) = \lambda_i(\mathbf{A}_F)(1 \pm o(1))$.

Claim 5.2.9 $\lambda_1(\mathbf{A}_H) \leq 6mt^{15/64}$ *whp.*

Proof Let H_i denote the subgraph of H induced by S_i , and let H_{ij} denote the subgraph of H containing only edges with one vertex in S_i and the other in S_j . That is, write \mathbf{A}_H in the following way:

$$\mathbf{A}_H = \begin{bmatrix} \mathbf{H}_1 & \mathbf{H}_{12} & \mathbf{H}_{13} \\ \mathbf{H}_{21} & \mathbf{H}_2 & \mathbf{H}_{23} \\ \mathbf{H}_{31} & \mathbf{H}_{32} & \mathbf{H}_3 \end{bmatrix}$$

We will use this to bound the maximal eigenvalue of \mathbf{A}_H as

$$\begin{aligned} \lambda_1(\mathbf{A}_H) &= \lambda_1 \left(\begin{bmatrix} \mathbf{H}_1 & \mathbf{H}_{12} & \mathbf{H}_{13} \\ \mathbf{H}_{21} & \mathbf{H}_2 & \mathbf{H}_{23} \\ \mathbf{H}_{31} & \mathbf{H}_{32} & \mathbf{H}_3 \end{bmatrix} \right) \\ &\leq \lambda_1(\mathbf{H}_1) + \lambda_1(\mathbf{H}_2) + \lambda_1(\mathbf{H}_3) + \lambda_1(\mathbf{H}_{12}) + \lambda_1(\mathbf{H}_{23}) + \lambda_1(\mathbf{H}_{13}). \end{aligned} \quad (5.39)$$

Note that the maximum eigenvalue of a graph is at most the maximum degree of a graph. By Claim 5.2.6 with $f(t) = t_1$ and $\varepsilon = 1/64$,

$$\begin{aligned} \lambda_1(\mathbf{H}_1) &\leq \Delta_1(\mathbf{H}_1) = \max_{v \leq t_1} \{d_{t_1}^m(v)\} \leq t_1^{p/2+\varepsilon} \leq t^{33/512} \\ \lambda_1(\mathbf{H}_2) &\leq \Delta_1(\mathbf{H}_2) \leq \max_{t_1 \leq v \leq t_2} \{d_{t_2}^m(v)\} \leq t_2^{p/2+\varepsilon} / t_1^{p/2} \leq t^{233/1024} \\ \lambda_1(\mathbf{H}_3) &\leq \Delta_1(\mathbf{H}_3) \leq \max_{t_2 \leq v \leq t_3} \{d_{t_3}^m(v)\} \leq t_3^{p/2+\varepsilon} / t_2^{p/2} \leq t^{15/64} \end{aligned} \quad (5.40)$$

To bound $\lambda_1(\mathbf{H}_{ij})$, start with $m = 1$. For $i < j$, this implies that each vertex in S_j has at most one edge in \mathbf{H}_{ij} , i.e. H_{ij} is a star forest. Then we have a bound on \mathbf{H}_{ij} by Claim 5.2.8. For $m > 1$, let G' be one of our generated graphs with t edges and $m = 1$. Think now of contracting vertices in G' (only the ones added using preferential attachment) into a single vertex. We can write \mathbf{A}_G in terms of \mathbf{A}'_G : $\mathbf{A}_G = \mathbf{C}^T \mathbf{A}'_G \mathbf{C}$, where \mathbf{C} is a contraction matrix with t rows and the number of columns equal to the number of vertices in \mathbf{A}_G (at most t/m). The i^{th} column is equal to 1 at indices j in which (i, j) are identified. Similarly, we can write \mathbf{H}_{ij} in terms of \mathbf{H}'_{ij} .

Note that if $\mathbf{y} = \mathbf{C}\mathbf{x}$, then $\mathbf{y}^T \mathbf{y} = \mathbf{x}^T \mathbf{C}^T \mathbf{C} \mathbf{x}$, where $\mathbf{C}^T \mathbf{C}$ is a diagonal matrix with 1 's and m 's on the diagonal. So $\mathbf{x}^T \mathbf{x} \leq \mathbf{y}^T \mathbf{y} \leq m \mathbf{x}^T \mathbf{x}$.

$$\begin{aligned} \lambda_1(\mathbf{H}_{ij}) &= \max_{\mathbf{x} \neq 0} \frac{\mathbf{x}^T \mathbf{H}_{ij} \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = \max_{\mathbf{x} \neq 0} \frac{\mathbf{x}^T \mathbf{C}^T \mathbf{H}'_{ij} \mathbf{C} \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = \max_{\mathbf{x} \neq 0, \mathbf{y} = \mathbf{C}\mathbf{x}} \frac{\mathbf{y}^T \mathbf{H}'_{ij} \mathbf{y}}{\mathbf{x}^T \mathbf{x}} \\ &= \max_{\mathbf{x} \neq 0, \mathbf{y} = \mathbf{C}\mathbf{x}} \frac{m \mathbf{y}^T \mathbf{H}'_{ij} \mathbf{y}}{m \mathbf{x}^T \mathbf{x}} \leq \max_{\mathbf{x} \neq 0, \mathbf{y} = \mathbf{C}\mathbf{x}} \frac{m \mathbf{y}^T \mathbf{H}'_{ij} \mathbf{y}}{\mathbf{y}^T \mathbf{y}} \end{aligned} \quad (5.41)$$

Now using Claim 5.2.6 with $f(t) = t_1$ and $\varepsilon = 1/64$,

$$\begin{aligned}\Delta_1(\mathbf{H}'_{12}) &= \max_{\mathbf{v} \leq t_2} \{d'_{t_2}(v)\} \leq t_2^{p/2+\varepsilon} \leq t^{297/1024} \\ \Delta_1(\mathbf{H}'_{23}) &= \max_{t_1 \leq v \leq t_3} \{d'_{t_3}(v)\} \leq t_3^{p/2+\varepsilon}/t_1^{p/2} \leq t^{29/64}\end{aligned}\tag{5.42}$$

Finally, all edges in \mathbf{H}'_{13} are between S_1 and S_3 , so Claim 5.2.7 shows $\Delta_1(\mathbf{H}'_{13}) \leq t^{p-9/16} \leq t^{7/16}$ whp. Putting together equations (5.41) and (5.42), we get $\lambda_1(\mathbf{H}_{ij}) \leq m\lambda_1(\mathbf{H}'_{ij}) \leq m\Delta_1(\mathbf{H}'_{ij})^{1/2} \leq mt^{15/64}$. And so we get the final bound,

$$\lambda_1(\mathbf{A}_H) \leq \sum_{i=1}^3 \lambda_1(\mathbf{H}_i) + \sum_{i < j} \lambda_1(\mathbf{H}_{ij}) \leq 6mt^{15/64}$$

This shows that $\lambda_i(\mathbf{A}_H)$ is $o(\lambda_k(\mathbf{A}_F))$, which implies $\lambda_i(\mathbf{A}_G) = \lambda_i(\mathbf{A}_F)(1 \pm o(1))$. ■

■

5.3 TGPA

In this section we present our model which we call Triangle Generalized Preferential Attachment (TGPA). This model is motivated by the purpose of adding higher order structure into the resulting graph as discussed in Section 2.3, and the recent paper by Avin et al. [2017] which shows a model of preferential attachment with any power-law exponent (section 5.1.1). We present two different versions of the model. The first, in Section 5.3.1 follows the PA model as described by Barabási and Albert [1999], Flaxman et al. [2005], and the second in Section 5.3.2 follows the PA model as described in Chung et al. [2006], Avin et al. [2017]. Though these models are not the same, they share similar properties. In Sections 5.4 and 5.5 we'll see each formulation is useful for the analysis of the models.

5.3.1 TGPA(p, q)

Start with an empty graph. At time $t = 1, 2, \dots$ do one of the following:

1. (node event) With probability p , add a new vertex v_t , and an edge from v_t to some other vertex u where u is chosen with probability

$$Pr[u = v_i] = \begin{cases} \frac{d_t(v_i)}{4t-2}, & \text{if } v_i \neq v_t \\ \frac{2}{4t-2}, & \text{if } v_i = v_t \end{cases}\tag{5.43}$$

Then pick a neighbor of u , call it w , and also add an edge from v_t to w . We pick w with the following probability:

$$Pr[w = v_i] = \begin{cases} \frac{\# \text{ edges between } u, w}{d_{t-1}(u)}, & \text{if } v_i \neq u \\ \frac{2 \cdot \# \text{ self-loops of } u}{d_{t-1}(u)}, & \text{if } v_i = u \end{cases} \quad (5.44)$$

2. (component event) With probability $q = 1 - p$ add a wedge to the graph (3 new nodes with 2 edges)
3. For some constant m , every m steps contract the most recently added vertices through the preferential attachment steps (in step 1) to form a super vertex.

Note that vertex w (chosen in step 1) is *also* chosen via preferential attachment. The probability of picking w is the probability of picking u as a neighbor of w times the probability of picking w .

$$Pr[w = v_i] = \frac{\sum_{u \in N(w)} d_{t-1}(u)}{4t - 2} \cdot \frac{\text{num edges between } u, w}{d_{t-1}(u)} = \frac{d_t(w)}{4t - 2}$$

5.3.2 TGPA(p_t, r_t, q_t)

Start with a graph with e_0 edges. At time $t = 1, 2, \dots$ do one of the following:

1. (node event) With probability p_t , add a new vertex v_t , and an edge from v_t to some other vertex u where u is chosen with probability given in Equation 5.43. Then pick a neighbor of u , call it w , and also add edge an edge from v_t to w . We pick w with the probability given in Equation 5.44.
2. (wedge event) With probability r_t add a wedge to the graph by picking two nodes using preferential attachment: v_1, v_2 . Pick the third node uniformly from neighbors of v_1 , call it w . Add edges (v_1, v_2) and (v_1, w) .
3. (component event) With probability q_t add a wedge to the graph (3 new nodes with 2 edges).

5.4 Analysis of TGPA(p, q)

In this section we present results on the degrees and spectra of the TGPA(p, q) model. The proofs follow very closely from the proofs of the results presented in Section 5.2. The primary difference is the probability of adding a node at any time, which results in a p term in the results instead of $p/2$. With analysis as in Section 5.2, by solving a martingale equation we get power-laws with exponent

$$\frac{1 + p}{p}$$

which ranges between $(1, \infty)$. Fix parameter p . Denote $G_t^m(p, q)$ as the Triangle Generalized Preferential Attachment Graph at time t with contractions of size m .

Lemma 5.4.1 *Let $d_t(s)$ be the degree of vertex s in G_t , for any time t after s has been added to the graph. Let $a^{(\bar{k})} = a(a+2)(a+4) \cdots (a+k-2)$ be a modified rising factorial function. Let s' be the time at which node s arrives in the graph. Then for any positive integer k ,*

$$\mathbb{E}[(d_t(s))^{(\bar{k})}] \leq (4m)^{(\bar{k})} 2^{pk} \left(\frac{t}{s'} \right)^{pk}$$

Proof Denote G_t^m as the graph at time t with contractions of size m . Let $Z_t = d_t^m(s)$ be the degree of vertex s at time t . Let Y_t be an indicator for the event that only one edge added at time t is incident to s , and let X_t be an indicator variable for the event that both of the edges added at time t are incident to s . First, let's calculate the probability of placing exactly one edge incident to node s at time t :

$$\begin{aligned} & p \left[\frac{d_t(s)}{4t-2} \left(1 - \frac{2(\text{num of self loops})}{d_t(s)} \right) + \frac{\sum_{u \in N(s), u \neq s} d_t(u)}{4t-2} \left(\frac{\text{num edges btwn } u, s}{d_t(u)} \right) \right] \\ & \quad \begin{array}{l} \text{probability of picking } s \text{ first, and then} \\ \text{not picking it second} \end{array} \quad \begin{array}{l} \text{probability of not picking } s \text{ first, but pick-} \\ \text{ing a neighbor, and then picking } s \text{ second} \end{array} \\ & = p \left[\frac{d_t(s)}{2t-1} - \frac{2(\text{num self loops})}{2t-1} \right]. \end{aligned}$$

Also the probability of placing two (both) edges incident to node s at time t :

$$p \frac{d_t(s)}{4t-2} \cdot \Pr[\text{picking it second} \mid \text{picked it first}] = \frac{d_t(s)}{4t-2} \cdot \frac{2(\text{num self loops})}{d_t(s)} = \frac{\text{num self loops}}{2t-1}.$$

Then we can write the expectation of Z_t in terms of Z_{t-1} using the above calculations:

$$\begin{aligned}
\mathbb{E}[Z_t^{(\bar{k})}] &= \mathbb{E}[\mathbb{E}[(Z_{t-1} + Y_t + 2X_t)^{(\bar{k})}] | Z_{t-1}] \\
&= \mathbb{E} \left[(Z_{t-1} + 2)^{(\bar{k})} p \cdot \left(\frac{\text{num self loops}}{2t-1} \right) + (Z_{t-1} + 1)^{(\bar{k})} p \cdot \left(\frac{d_t(s)}{2t-1} - \frac{2(\text{num self loops})}{2t-1} \right) \right. \\
&\quad \left. + Z_{t-1}^{(\bar{k})} \left(1 - p \frac{d_t(s)}{2t-1} + p \frac{\text{num self loops}}{2t-1} \right) \right] \\
&\leq \mathbb{E} \left[(Z_{t-1} + 2)^{(\bar{k})} p \cdot \left(\frac{\text{num self loops}}{2t-1} \right) + (Z_{t-1} + 2)^{(\bar{k})} p \cdot \left(\frac{d_t(s)}{2t-1} - \frac{2(\text{num self loops})}{2t-1} \right) \right. \\
&\quad \left. + Z_{t-1}^{(\bar{k})} \left(1 - p \frac{d_t(s)}{2t-1} + p \frac{\text{num self loops}}{2t-1} \right) \right] \\
&= \mathbb{E} \left[(Z_{t-1} + 2)^{(\bar{k})} p \cdot \left(\frac{d_t(s)}{2t-1} - \frac{\text{num self loops}}{2t-1} \right) + Z_{t-1}^{(\bar{k})} \left(1 - p \frac{d_t(s)}{2t-1} + p \frac{\text{num self loops}}{2t-1} \right) \right] \\
&= \mathbb{E} \left[Z_{t-1}^{(\bar{k})} \left(1 - p \frac{Z_{t-1}}{2t-1} + p \frac{\text{num self loops}}{2t-1} + \frac{Z_{t-1} + k}{Z_{t-1}} \left(p \frac{Z_{t-1}}{2t-1} - p \frac{\text{num self loops}}{2t-1} \right) \right) \right] \\
&= \mathbb{E} \left[Z_{t-1}^{(\bar{k})} \left(1 - p \frac{Z_{t-1} - \text{num self loops}}{2t-1} + \frac{Z_{t-1} + k}{Z_{t-1}} \left(p \frac{Z_{t-1} - \text{num self loops}}{2t-1} \right) \right) \right] \\
&\leq \mathbb{E} \left[Z_{t-1}^{(\bar{k})} \left(1 - p \frac{Z_{t-1} - \text{num self loops}}{2t-1} + p \frac{Z_{t-1} + k}{2t-1} \right) \right] \\
&= \mathbb{E} \left[Z_{t-1}^{(\bar{k})} \left(1 + \frac{p}{2t-1} (k + \text{num self loops}) \right) \right].
\end{aligned}$$

Now if $k \geq \text{num self loops}$ we can move on to:

$$\mathbb{E}[Z_t^{(\bar{k})}] \leq \mathbb{E} \left[Z_{t-1}^{(\bar{k})} \left(1 + \frac{2pk}{2t-1} \right) \right] \quad (5.45)$$

Apply this relationship iteratively, down to the time when node s was added (recall we denoted that time as s'). Also note that the degree of s at time s' is bounded by $4m$ (if all m edges were added as self loops). Thus:

$$\mathbb{E}(Z_t^{(\bar{k})}) = \prod_{t'=s'}^t \left(1 + \frac{2pk}{2t'-1} \right) \leq (4m)^{(\bar{k})} \prod_{t'=s'+1}^t \left(1 + \frac{2pk}{2t'-1} \right) \quad (5.46)$$

Use $1 + x \leq e^x$ to write the product as a sum, and bound the sum with an integral:

$$\sum_{t'=s'+1}^t \frac{1}{t'-1/2} \leq \int_{x=s'}^t \frac{1}{x-1/2} dx = \log \frac{t-1/2}{s'-1/2}. \quad (5.47)$$

So finally,

$$\begin{aligned}
\mathbb{E}(Z_t^{(\bar{k})}) &\leq (4m)^{(\bar{k})} \left(\frac{t-1/2}{s'-1/2} \right)^{pk} \\
&= (4m)^{(\bar{k})} \left(\frac{t}{s'} \right)^{pk/2} \left(\frac{2-1/t}{2-1/s'} \right)^{pk} \\
&\leq (4m)^{(\bar{k})} \left(\frac{t}{s'} \right)^{pk} 2^{pk}.
\end{aligned} \quad (5.48)$$

■

Lemma 5.4.2 *Let $S = (S_1, S_2, \dots, S_l)$ be a disjoint collection of supernodes at time t_0 . Assume that the degree of S_i at time t_0 is $d_{t_0}(S_i) = d_i$. Let t be a time later than t_0 . Let $p_S(\mathbf{r}; \mathbf{d}, t_0, t)$ be the probability that each supernode S_i has degree $r_i + d_i$ at time t . Let $d = \sum_{i=1}^l d_i, r = \sum_{i=1}^l r_i$. If $d = o(t^{1/2})$ and $r = o(t^{2/3})$, then*

$$p_S(\mathbf{r}; \mathbf{d}, t_0, t) \leq \left(\prod_{i=1}^l \binom{r_i + d_i - 1}{d_i - 1} \right) \left(\frac{t_0}{t - 1} \right)^{pd} \exp \left\{ 3 + 2t_0 - pd + \frac{19pr}{4t^p} \right\}$$

Proof Let $\tau^{(i)} = (\tau_1^{(i)}, \dots, \tau_{r_i}^{(i)})$, where $\tau_j^{(i)}$ is the time when we add an edge incident to S_i and increase the degree of S_i from $d_i + j - 1$ to $d_i + j$. Note that we could have repeated times in $\tau^{(i)}$. Define $\tau = (\tau_0, \tau_1, \dots, \tau_{r+1})$ to be the ordered union of $\tau^{(i)}$, with $\tau_0 = t_0$ and $\tau_{r+1} = t$. Again, there may be up to two moves per time-step. Let $p(\tau; \mathbf{d}, t_0, t)$ be the probability that supernodes S_i increase in degree at exactly the times specified by τ between time t_0 and t . Define all time-steps to be $T := \{t_0, t_1, t_1, t_2, t_2, \dots, t_r, t_r\}$. Time steps involving the sets S_i we defined to be τ . So the remaining time-steps are $T - \tau$. Then

$$\begin{aligned} p(\tau; \mathbf{d}, t_0, t) &\leq \left(\prod_{i=1}^l \prod_{k=1}^{r_i} 2p \frac{d_i + k - 1}{4\tau_k^{(i)} - 2} \right) \left(\prod_{k=0}^r \prod_{\substack{j \in T - \tau \\ j \geq \tau_k \\ j < \tau_{k+1}}} \left(1 - 2p \frac{d + k}{4j - 2} \right) \right) \\ &\quad \text{for each supernode } S_i, \text{ the prob. of } \tau \text{ aligning with } \tau^{(i)}. \quad \text{for each timestep inbetween the relevant ones, the probability of picking any edge outside of } S_1, \dots, S_l. \end{aligned} \quad (5.49)$$

$$= \left(\prod_{i=1}^l \frac{(r_i + d_i - 1)!}{(d_i - 1)!} \right) \left(\prod_{k=1}^r \frac{p}{2\tau_k - 1} \right) \exp \left\{ \sum_{k=0}^r \sum_{\substack{j \in T - \tau \\ j \geq \tau_k \\ j < \tau_{k+1}}} \log \left(1 - p \left(\frac{d + k}{2j - 1} \right) \right) \right\}$$

Now we can bound the inner most sum of the exponential term.

$$\sum_{\substack{j \in T - \tau \\ j \geq \tau_k \\ j < \tau_{k+1}}} \log \left(1 - p \left(\frac{d + k}{2j - 1} \right) \right) \leq 2 \sum_{j=\tau_k+1}^{\tau_{k+1}-1} \log \left(1 - \frac{p(d + k)}{2j} \right) \quad (5.50)$$

which is less than or equal to

$$\begin{aligned} 2 \int_{\tau_k+1}^{\tau_{k+1}} \log \left(1 - \frac{p(d + k)}{2x} \right) dx &= -2\tau_{k+1} \log(2\tau_{k+1}) + 2(\tau_k + 1) \log(2\tau_k + 2) \\ &\quad + (2\tau_{k+1} - p(d + k)) \log(2\tau_{k+1} - p(d + k)) \\ &\quad - (2\tau_k + 2 - p(d + k)) \log(2\tau_k + 2 - p(d + k)). \end{aligned} \quad (5.51)$$

Note that $\tau_0 = t_0$ and $\tau_{r+1} = t$. We can write

$$\sum_{k=0}^r \int_{\tau_k+1}^{\tau_{k+1}} \log \left(1 - \frac{p(d + k)}{2x} \right) dx = A + \sum_{k=1}^r B_k \quad (5.52)$$

where

$$A = 2(t_0 + 1) \log(2t_0 + 2) - (2t_0 + 2 - pd) \log(2t_0 + 2 - pd) - 2t \log(2t) + (2t - p(d + r)) \log(2t - p(d + r)) \quad (5.53)$$

and

$$B_k = 2\tau_k \log(1 + 1/\tau_k) + 2 \log(2\tau_k + 2) - (2 - p) \log(2\tau_k + p - p(d + k)) + (2\tau_k + 2 - p(d + k)) \log \left(1 - \frac{2 - p}{2\tau_k + 2 - p(d + k)} \right). \quad (5.54)$$

We will bound each of A and B_k , starting with B_k . Since $1 + x \leq e^x$, $1\tau_k \log(1 + 1/\tau_k) \leq 2$, and $(2\tau_k + 2 - p(d + k)) \log \left(1 - \frac{2 - p}{2\tau_k + 2 - p(d + k)} \right) \leq p - 2$. Rearranging the other two terms of equation (5.54) and combining with these inequalities we get

$$B_k \leq p \log(2\tau_k + 2) - (2 - p) \log \left(1 - \frac{p(d + k) + 2 - p}{2\tau_k + 2} \right) + p. \quad (5.55)$$

Now rearranging terms of A from equation (5.53),

$$\begin{aligned} A &= -2(t_0 + 1) \log \left(1 - \frac{pd}{2t_0 + 2} \right) + pd \log(2t_0 + 2 - pd) \\ &\quad + 2t \log \left(1 - \frac{p(d + r)}{2t} \right) - p(d + r) \log(2t - p(d + r)) \\ e^A &= \left(1 - \frac{pd}{2t_0 + 2} \right)^{-2(t_0 + 1)} (2t_0 + 2 - pd)^{pd} \left(1 - \frac{p(d + r)}{2t} \right)^{2t} (2t - p(d + r))^{-p(d + r)} \\ &= \left(1 - \frac{pd}{2t_0 + 2} \right)^{-2(1 - \frac{pd}{2(t_0 + 1)})(t_0 + 1)} \left(1 - \frac{p(d + r)}{2t} \right)^{2t - p(d + r)} \left(\frac{t_0 + 1}{t} \right)^{pd} (2t)^{-pr}. \end{aligned} \quad (5.56)$$

Using the bound $1 - x \leq e^{-x - x^2/2}$ for $0 < x < 1$,

$$\left(1 - \frac{p(d + r)}{2t} \right)^{2t - p(d + r)} \leq \exp \left\{ -p(d + r) + \frac{p^2(d + r)^2}{4t} + \frac{p^3(d + r)^3}{8t^2} \right\} \quad (5.57)$$

Putting the bounds on A and B_k together, we get

$$\begin{aligned} e^{A + \sum B_k} &\leq \left(1 - \frac{pd}{2t_0 + 2} \right)^{-2(1 - \frac{pd}{2(t_0 + 1)})(t_0 + 1)} \exp \left\{ -p(d + r) + \frac{p^2(d + r)^2}{4t} + \frac{p^3(d + r)^3}{8t^2} \right\} \\ &\quad \times \left(\frac{t_0 + 1}{t} \right)^{pd} (2t)^{-pr} \prod_{k=1}^r \left(\left(1 - \frac{p(d + k) + 2 - p}{2\tau_k + 2} \right)^{-(2 - p)} (2\tau_k + 2)^p \right) e^{pr}. \end{aligned} \quad (5.58)$$

Using

$$\text{err}(r, d, t_0, t) = \left(1 - \frac{pd}{2t_0 + 2} \right)^{-2(1 - \frac{pd}{2(t_0 + 1)})(t_0 + 1)} \exp \left\{ -pd + \frac{p^2(d + r)^2}{4t} + \frac{p^3(d + r)^3}{8t^2} \right\}, \quad (5.59)$$

we can write equation (5.58) as

$$e^{A + \sum B_k} \leq \text{err}(r, d, t_0, t) \left(\frac{t_0 + 1}{t} \right)^{pd} (2t)^{-pr} \prod_{k=1}^r \left(\left(1 - \frac{p(d + k) + 2 - p}{2\tau_k + 2} \right)^{-(2 - p)} (2\tau_k + 2)^p \right). \quad (5.60)$$

So we finally finish with the bound on $p(\tau; \mathbf{d}, t_0, t)$ by substituting equation (5.60) into equation (5.49):

$$p(\tau; \mathbf{d}, t_0, t) \leq \left(\prod_{i=1}^l \frac{(r_i + d_i - 1)!}{(d_i - 1)!} \right) \text{err}(r, d, t_0, t) \left(\frac{t_0 + 1}{t} \right)^{pd} (2t)^{-pr} \\ \times \prod_{k=1}^r \left(\left(1 - \frac{p(d+k) + 2 - p}{2\tau_k + 2} \right)^{-(2-p)} (2\tau_k + 2)^p \frac{p}{2\tau_k - 1} \right), \quad (5.61)$$

which can be re-arranged as

$$= \left(\prod_{i=1}^l \frac{(r_i + d_i - 1)!}{(d_i - 1)!} \right) \text{err}(r, d, t_0, t) \left(\frac{t_0 + 1}{t} \right)^{pd} (2t)^{-pr} \\ \times \prod_{k=1}^r \left(p(2\tau_k + p - p(d+k))^{-(2-p)} \left(2\tau_k + 5 + \frac{9}{2\tau_k - 1} \right) \right). \quad (5.62)$$

Now, we will sum $p(\tau; \mathbf{d}, t_0, t)$ over all ordered choices of τ .

$$p(\mathbf{r}; \mathbf{d}, t_0, t) \leq \sum_{\tau^{(1)}, \dots, \tau^{(l)}} p(\tau; \mathbf{d}, t_0, t) \\ \leq \binom{r}{r_1, \dots, r_l} \sum_{t_0+1 \leq \tau_1 < \dots < \tau_r \leq t} \prod_{i=1}^l \frac{(r_i + d_i - 1)!}{(d_i - 1)!} \text{err}(r, d, t_0, t) \left(\frac{t_0 + 1}{t} \right)^{pd} \\ \times (2t)^{-pr} p \prod_{k=1}^r (2\tau_k + p - p(d+k))^{-(2-p)} \left(2\tau_k + 5 + \frac{9}{2\tau_k - 1} \right) \\ = r! \left(\prod_{i=1}^l \binom{r_i + d_i - 1}{d_i - 1} \right) \text{err}(r, d, t_0, t) \left(\frac{t_0 + 1}{t} \right)^{pd} (2t)^{-pr} \\ \times \sum_{t_0+1 \leq \tau_1 < \dots < \tau_r \leq t} p \prod_{k=1}^r (2\tau_k + p - p(d+k))^{-(2-p)} \left(2\tau_k + 5 + \frac{9}{2\tau_k - 1} \right) \quad (5.63)$$

Now let $\tau'_k = \tau_k - \lceil p(d+k)/2 \rceil$. Since $d \geq 1$ and $k \geq 1$, we have $2\lceil p(d+k)/2 \rceil \geq 2$. So, the last term in equation (5.63) is less than or equal to

$$\begin{aligned}
& \sum_{(t_0-p\lceil d/2 \rceil+1) \leq \tau'_1 \leq \dots \leq \tau'_r \leq (t-p\lceil (d+r)/2 \rceil)} \left(p \prod_{k=1}^r (2\tau'_k + p)^{-(2-p)} \left(2\tau'_k + 5 + \frac{9}{2\tau'_k + 1} \right) \right) \\
& \leq \frac{p}{r!} \left(\sum_{\tau'=(t_0-p\lceil d/2 \rceil+1)}^{t-p\lceil (d+r)/2 \rceil} \left(9(2\tau' + p)^{-(3-p)} + (2\tau'_k + 5)(2\tau'_k + p)^{-(2-p)} \right) \right)^r \\
& \leq \frac{p}{r!} \left(\int_0^{t-p\lceil (d+r)/2 \rceil} \left(9(2x + p)^{-(3-p)} + (2\tau'_k + 5)(2x + 1)^{-(2-p)} \right) dx \right)^r \\
& \leq \frac{p}{r!} \left(\frac{9}{2(2-p)p^{2-p}} + \frac{5}{(1-p)p^{1-p}} + \frac{2}{(1-p)p} (2t - p(d+r) + 1)^p \right)^r \tag{5.64} \\
& \leq \frac{2p}{r!(1-p)p^{2-p}} \left(\frac{19}{4} + (2t - p(d+r) + p)^p \right)^r \\
& = \frac{2}{r!(1-p)p^{1-p}} \left((2t)^p \left(1 - \frac{p(d+r) - p}{2t} \right)^p \left(1 + \frac{19p}{4(2t - p(d+r) + p)^p} \right) \right)^r \\
& \leq \frac{2}{r!(1-p)p^{1-p}} (2t)^{pr} \underbrace{\left(1 - \frac{p(d+r) - p}{2t} \right)^{pr}}_{\leq \exp\left\{ -\frac{pr(p(d+r)-p)}{2t} \right\}} \underbrace{\left(1 + \frac{19p}{4(2t - p(d+r) + p)^p} \right)^r}_{\leq \exp\left\{ \frac{19pr}{4(2t - p(d+r) + p)^p} \right\}}
\end{aligned}$$

where the last inequalities come from $1 + x \leq e^x$. So finally,

$$\begin{aligned}
p_S(\mathbf{r}; \mathbf{d}, t_0, t) & \leq \left(\prod_{i=1}^l \binom{r_i + d_i - 1}{d_i - 1} \right) \text{err}(r, d, t_0, t) \left(\frac{t_0 + 1}{t} \right)^{pd} \\
& \times \exp \left\{ \frac{-pr((d+r) - p)}{2t} + \frac{19pr}{4(2t - p(d+r) + p)^p} \right\}.
\end{aligned}$$

Since $d = o(t^{1/2})$ and $r = o(t^{2/3})$,

$$\begin{aligned}
& \text{err}(r, d, t_0, t) \exp \left\{ \frac{-rp((d+r) - p)}{2t} + \frac{19pr}{4(2t - p(d+r) + p)^p} \right\} \\
& \leq \left(1 - \frac{pd}{2(t_0 + 1)} \right)^{-2(1-pd/2(t_0+1))(t_0+1)} \exp \left\{ 1 - pd - \frac{r^2}{4t} + \frac{19pr}{4t^p} \right\} \\
& \leq \underset{\text{since } x^{-x} \leq e}{e^{2(t_0+1)}} \exp \left\{ 1 - pd + \frac{19pr}{4t^p} \right\} \\
& = \exp \left\{ 3 + 2t_0 - pd + \frac{19pr}{4t^p} \right\}
\end{aligned}$$

This concludes the proof. ■

Theorem 5.4.1 *Let m, k be fixed positive integers, and let $f(t)$ be a function with $f(t) \rightarrow \infty$ as $t \rightarrow \infty$. Let $\Delta_1 \geq \Delta_2 \geq \dots \geq \Delta_k$ denote the degrees of the k highest degree vertices of G_t^m . Then*

$$\frac{t^p}{f(t)} \leq \Delta_1 \leq t^p f(t) \quad \text{and} \quad \frac{t^p}{f(t)} \leq \Delta_i \leq \Delta_{i-1} - t^p f(t)$$

for $i = 1, 2, \dots, k$ whp.

The factor of t^p in Theorem 5.4.1 implies a power-law distribution with exponent $\alpha = (1 + p)/p$. This can be seen by using a martingale argument, which has been done a number of times. See for instance [van der Hofstad, 2016]. Notice that depending on the value chosen for p , we can obtain a power-law fit with exponents ranging between 2 and ∞ .

Proof Partition the vertices into those added before time t_0 , before time t_1 , and after t_1 , with $t_0 = \log \log \log f(t)$, $t_1 = \log \log f(t)$. We will argue about the maximum degree vertices in each set.

Claim 5.4.1 *In G_t^m , the degree of the supernode of vertices added before time t_0 is at least $t_0^{(1-p)/2} t^p$ whp.*

Proof Consider all vertices added before time t_0 as a supernode. Let A_1 denote the event that this supernode has degree less than $t_0^{(1-p)/2} t^p$ at time t . We will use Lemma 5.4.2 with $l = 1$, and $d = 4t_0$ (because the supernode has all edges at time t_0).

$$\begin{aligned}
Pr[A_1] &\leq \sum_{r_1=0}^{t_0^{(1-p)/2} t^p - 4t_0} \binom{r_1 + 4t_0 - 1}{4t_0 - 1} \left(\frac{t_0 + 1}{t} \right)^{pd} e^{3+2t_0-pd+19pr/4t^p} \\
&\leq \sum_{r_1=0}^{t_0^{(1-p)/2} t^p - 4t_0} \binom{t_0^{(1-p)/2} t^p - 1}{4t_0 - 1} \left(\frac{t_0 + 1}{t} \right)^{4pt_0} e^{3+2t_0-4pt_0+(19/4)pt_0^{1/3}-\frac{19pt_0}{t^p}} \\
&\quad \text{By substituting } r_1 = t_0^{(1-p)/2} t^p \text{ because } r = r_1 \text{ and } d = 4t_0 \\
&= (t_0^{(1-p)/2} t^p - 4t_0) \frac{(t_0^{(1-p)/2} t^p - 1)!}{(4t_0 - 1)!(t_0^{(1-p)/2} t^p - 4t_0)!} \left(\frac{t_0 + 1}{t} \right)^{4pt_0} e^{3+2t_0(1-2p)+(19/4)pt_0^{1/3}-\frac{19pt_0}{t^p}} \\
&\leq t_0^{(1-p)/2} t^p \frac{(t_0^{(1-p)/2} t^p)^{4t_0-1}}{(4t_0 - 1)!} \left(\frac{t_0 + 1}{t} \right)^{4pt_0} e^{3+2t_0(1-2p)+(19/4)pt_0^{1/3}-19pt_0/t^p} \\
&\leq t_0^{2(1-p)t_0} \frac{e^{4t_0-1}}{(4t_0 - 1)^{4t_0-1}} (t_0 + 1)^{4pt_0} e^{3+2t_0(1-2p)+(19/4)pt_0^{1/3}-19pt_0/t^p} \\
&\quad \text{since } 1/x! \leq e^x/x^x \\
&\leq \frac{e^{2+2t_0(3-2p)+(19/4)pt_0^{1/3}-19pt_0/t^p}}{(4t_0 - 1)^{2t_0(1-p)-1}}
\end{aligned} \tag{5.65}$$

which goes to 0 as t goes to infinity. Thus A_1 does not hold with high probability, and the claim is proved. \blacksquare

Claim 5.4.2 *In G_t^m , no vertex added after time t_1 has degree exceeding $t_0^{-2} t^p$ whp.*

Proof Let A_2 denote the event that some vertex added after time t_1 has degree exceeding $t_0^{-2} t^p$.

$$\begin{aligned}
Pr[A_2] &\leq \sum_{s=t_1}^t Pr[d_t(s) \geq t_0^{-2} t^p] = \sum_{s=t_1}^t Pr[(d_t(s))^{(\bar{l})} \geq (t_0^{-2} t^p)^{(\bar{l})}] \leq \sum_{s=t_1}^t t_0^{2l} t^{-lp} \mathbb{E}[(d_t(s))^{(\bar{l})}] \\
&\quad \text{by Markov} \\
&= \sum_{s=t_1}^t t_0^{2l} t^{-lp} (4m)^{(\bar{l})} 2^{lp} \left(\frac{t}{s} \right)^{lp} = 2^{lp} (4m)^{(\bar{l})} t_0^{2l} \int_{t_1-1}^t x^{-lp} dx \\
&\quad \text{by Lemma 5.4.1}
\end{aligned} \tag{5.66}$$

We compute the integral in equation (5.66),

$$\int_{t_1-1}^t x^{-lp} dx = \frac{x^{-lp+1}}{-lp+1} \Big|_{t_1-1}^t = (-lp+1)^{-1} \left(t^{-lp+1} - (t_1-1)^{-lp+1} \right) \quad (5.67)$$

We want to choose l so that $-lp+1$ is less than 0. So choose $l > 1/p$. Then the integral in equation (5.67) is less than or equal to $(lp-1)^{-1}(t_1-1)^{-lp+1}$, and plugging in the computation from equation (5.67) into equation (5.66),

$$Pr[A_2] \leq \frac{2^{lp}(4m)^{(\bar{l})} t_0^{2l}}{(lp-1)(t_1-1)^{lp-1}} \quad (5.68)$$

which goes to 0 as t increases. ■

Claim 5.4.3 *In G_t^m , no vertex added before time t_1 has degree exceeding $t_0^{(1-p)/4} t^p$ whp.*

Proof Let A_3 denote the event that some vertex added before time t_1 has degree exceeding $t_0^{(1-p)/4} t^p$. Using the exact argument as in Claim 5.4.2, $Pr[A_3]$ goes to 0 as t increases. ■

Claim 5.4.4 *The k highest degree vertices of G_t^m are added before time t_1 and have degree Δ_i bounded by $t_0^{-1} t^p \leq \Delta_i \leq t_0^{(1-p)/4} t^p$*

Proof First lets summarize the results of the last three claims:

- Bound on degrees of vertices added after time t_1 : $t_0^{-2} t^p$
- Bound on degrees of vertices added before time t_1 : $t_0^{(1-p)/4} t^p$
- Sum of all degrees added before time t_0 is at least: $t_0^{(1-p)/2} t^p$

So the upper bound of the claim is immediately clear from the second item. Suppose that the lower bound does not hold. Then one of the top k vertices has degree less than $t_0^{-1} t^p$ and the total degree of vertices added before time t_0 is bounded by

$$\begin{aligned} & \underbrace{(k-1)t_0^{(1-p)/4} t^p}_{\text{largest possible degrees of } (k-1) \text{ vertices}} + \underbrace{\left(\frac{t_0}{m} - k + 1 \right) \left(t_0^{-1} t^p \right)}_{\text{largest possible degrees of remaining vertices}} \leq k t_0^{(1-p)/4} t^p + t_0 (t_0^{-1} t^p) \\ & = k t_0^{(1-p)/4} t^p + t^p = t^p (k t_0^{(1-p)/4} + 1) \leq t^p (2k t_0^{(1-p)/4}) \leq t^p t_0^{(1-p)/2}, \end{aligned} \quad (5.69)$$

which contradicts the third bulleted item. Finally, since we have the lower bound, and we know that $t^p/t_0 \geq t^p/t_0^2$, then none of the largest degree vertices could be added after time t_1 . ■

Claim 5.4.5 *The k highest degree vertices of G_t^m have $\Delta_i \leq \Delta_{i-1} - \frac{t^p}{f(t)}$ whp.*

Proof Let A_4 denote the event that there are two vertices among the first t_1 time steps with degrees exceeding $t_0^{-1}t^p$ and within $t^p/f(t)$ of each other. Let

$$p_{l,s_1,s_2} = \Pr [d_t(s_1) - d_t(s_2) = l \mid \overline{A_3}], \text{ for } |l| \leq t^p/f(t),$$

where $\overline{A_3}$ means the opposite of event A_3 from Claim 5.4.3. Then

$$\Pr [A_4 | \overline{A_3}] \leq \sum_{1 \leq s_1 < s_2 \leq t_1} \sum_{l=-t^p/f(t)}^{t^p/f(t)} p_{l,s_1,s_2} \quad (5.70)$$

Now

$$p_{l,s_1,s_2} \leq \sum_{r_1=t_0^{-1}t^p}^{t_0^{(1-p)/4}t^p} \sum_{d_1,d_2=1}^{4t_1} p_{(s_1,s_2)}((r_1, r_1 - l); (d_1, d_2), t_1, t) \quad (5.71)$$

Notation from Lemma 5.4.2.

Using Lemma 5.4.2,

$$\begin{aligned} &\leq \sum_{r_1=t_0^{-1}t^p}^{t_0^{(1-p)/4}t^p} \sum_{d_1,d_2=1}^{4t_1} \binom{r_1 + d_1 - 1}{d_1 - 1} \binom{r_1 - l + d_2 - 1}{d_2 - 1} \left(\frac{t_1 + 1}{t}\right)^{p(d_1+d_2)} \\ &\quad \times e^{\left\{3+2t_1-p(d_1+d_2)+\frac{19p(r_1-l)}{4t^p}\right\}} \\ &\leq t_0^{(1-p)/4}t^p \sum_{d_1,d_2=1}^{4t_1} \binom{t_0^{(1-p)/4}t^p + d_1 - 1}{d_1 - 1} \binom{t_0^{(1-p)/4}t^p - l + d_2 - 1}{d_2 - 1} \left(\frac{t_1 + 1}{t}\right)^{p(d_1+d_2)} \\ &\quad \times e^{\left\{3+2t_1-p(d_1+d_2)+\frac{19pt_0^{(1-p)/4}t^p}{4t^p}\right\}} \\ &\leq t_0^{(1-p)/4}t^p \sum_{d_1,d_2=1}^{4t_1} \binom{2t_0^{(1-p)/4}t^p}{d_1 - 1} \binom{2t_0^{(1-p)/4}t^p}{d_2 - 1} \left(\frac{t_1 + 1}{t}\right)^{p(d_1+d_2)} \\ &\quad \times e^{3+2t_1+(19/4)pt_0^{(1-p)/4}} \\ &\leq t_0^{(1-p)/4}t^p \sum_{d_1,d_2=1}^{4t_1} (2t_0^{(1-p)/4}t^p)^{d_1+d_2-2} (t_1 + 1)^{8pt_1} t^{-p(d_1+d_2)} e^{3+2t_1+(19/4)pt_0^{(1-p)/4}} \\ &= t^{-p}t_0^{(1-p)/4}(4t_1)^2 2^{8t_1} t_0^{2t_1(1-p)} (t_1 + 1)^{8pt_1} e^{3+2t_1+(19/4)pt_0^{(1-p)/4}} \end{aligned} \quad (5.73)$$

Denote the last equation as $h(t)$ and note $h(t)$ is a polynomial in $\log(f(t))$ times a factor of t^{-p} .

Then going back to equation (5.70),

$$\Pr [A_4 | \overline{A_3}] \leq \binom{t_1}{2} 2 \frac{t^p}{f(t)} h(t) = \binom{t_1}{2} 2 \frac{\text{poly}(\log(f(t)))}{f(t)} \quad (5.74)$$

which goes to 0 as t increases. This concludes the proof of this final claim. ■

And this concludes the proof of the theorem. ■

Theorem 5.4.2 *Let k be a fixed integer, and let $f(t)$ be a function with $f(t) \rightarrow \infty$ as $t \rightarrow \infty$. Let $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k$ be the k largest eigenvalues of the adjacency matrix of G_t . Then for $i = 1, \dots, k$, we have $\lambda_i = (1 + o(1))\Delta_i^{1/2}$, where Δ_i is the i^{th} largest degree.*

In the analysis of this proof, we will need to restrict p to be greater than $9/32$. This comes in Claim 5.4.7. This restricts the power of the exponent in the power-law to be between 2 and 5. We will show, that with high probability, G contains a star forest F , with stars of degree asymptotic to the maximum degree vertices of G . Then we show that $G \setminus F$ has small eigenvalues. Then we can use Rayleigh's principle to say that the large eigenvalues of G cannot be too different than the large eigenvalues of F .

Proof Let S_i be the vertices added after time t_{i-1} and at or before time t_i , for $t_0 = 0, t_1 = t^{1/8}, t_2 = t^{9/16}, t_3 = t$. Let $G = G_t$. We start by finding bounds on the degrees and co-degrees of G .

Claim 5.4.6 *For any $\varepsilon > 0$, and any $f(t)$ with $f(t) \rightarrow \infty$ as $t \rightarrow \infty$ the following holds whp: for all s with $f(t) \leq s \leq t$, for all vertices $v \in G_s$, if v was added at time r , then $d_s(v) \leq s^{p+\varepsilon} r^{-p}$.*

Proof

$$\begin{aligned} \Pr \left[\bigcup_{s=f(t)}^t \bigcup_{r=1}^s \left\{ d_s^m(r) \geq s^{p+\varepsilon} r^{-p} \right\} \right] &\leq \sum_{s=f(t)}^t \sum_{r=1}^s \Pr \left[d_s^m(r) \geq s^{p+\varepsilon} r^{-p} \right] \\ &= \sum_{s=f(t)}^t \sum_{r=1}^s \Pr \left[(d_s^m(r))^{(\bar{l})} \geq (s^{p+\varepsilon} r^{-p})^{(\bar{l})} \right] \end{aligned} \quad (5.75)$$

which is bounded using Markov:

$$\leq \sum_{s=f(t)}^t \sum_{r=1}^s s^{-l(p+\varepsilon)} r^{pl} \mathbb{E} \left[(d_s^m(r))^{(\bar{l})} \right]$$

which we can bound using Lemma 5.4.1

$$\leq \sum_{s=f(t)}^t \sum_{r=1}^s s^{-l(p+\varepsilon)} r^{lp} (4m)^{(\bar{l})} 2^{lp} \left(\frac{s}{r} \right)^{lp} = (4m)^{(\bar{l})} 2^{lp} \sum_{s=f(t)}^t s^{1-\varepsilon l}$$

Take $l \geq 3/\varepsilon$. Then we can bound the sum by an integral,

$$\sum_{s=f(t)}^t s^{1-\varepsilon l} \leq \int_{f(t)-1}^{\infty} x^{1-\varepsilon l} dx = \frac{1}{2-\varepsilon l} x^{2-\varepsilon l} \Big|_{f(t)-1}^{\infty} = \frac{1}{\varepsilon l - 2} (f(t) - 1)^{2-\varepsilon l} \quad (5.76)$$

which goes to zero as t increases, since $l \geq 3/\varepsilon$. ■

Claim 5.4.7 *Let S'_3 be the set of vertices in S_3 that are adjacent to more than one vertex of S_1 in G . Then $|S'_3| \leq t^{2p-9/16}$ with high probability.*

Proof Let B_1 be the event that the conditions of Claim 5.4.6 hold with $f(t) = t_2$ and $\varepsilon = 1/16$. Then for a vertex $v \in S_3$ added at time s , the probability that v picks at least one neighbor in S_1 is less than or equal to

$$\frac{2 \sum_{w \in S_1} d_s(w)}{4s - 2} \leq \frac{\sum_{w \in S_1} s^{p+\varepsilon}}{2s - 1} = \frac{t_1 s^{p+\varepsilon}}{2s - 1} \quad (5.77)$$

So, the probability of having two or more neighbors in S_1 can be bounded by,

$$\Pr[|N(v) \cap S_1| \geq 2 \mid B_1] \leq \left(\frac{t_1 s^{p+\varepsilon}}{2s-1} \right)^2 \cdot \binom{2m}{2} \leq m^2 t^{1/4} s^{(16p-15)/8} \quad (5.78)$$

Let X denote the number of $v \in S_3$ adjacent to more than one vertex of S_1 . Then

$$\begin{aligned} \mathbb{E}[X|B_1] &\leq \sum_{t_2+1}^t m^2 s^{(-15+16p)/8} t^{1/4} \leq m^2 t^{1/4} \int_{t_2}^t x^{(-15+16p)/8} dx \\ &= m^2 t^{1/4} \left[\frac{8}{-7+16p} x^{(-7+16p)/8} \right]_{t_2}^t \leq \frac{8m^2 t^{1/4}}{-7+16p} t^{(-7+16p)/8} \end{aligned} \quad (5.79)$$

Then by Markov,

$$\Pr[X \geq t^{2p-9/16} | B_1] \leq \frac{\mathbb{E}[X|B_1]}{t^{2p-9/16}} \leq \frac{8m^2}{16p-7} \frac{t^{2p-5/8}}{t^{2p-9/16}} = \frac{8m^2}{16p-7} \frac{t^{-5/8}}{t^{-9/16}} \quad (5.80)$$

And $\frac{t^{-5/8}}{t^{-9/16}} = \frac{t^{9/16}}{t^{10/16}}$ which goes to zero. ■

Let $F \subseteq G$ be the star forest consisting of edges between S_1 and $S_3 \setminus S'_3$.

Claim 5.4.8 *Let $\Delta_1 \geq \Delta_2 \geq \dots \geq \Delta_k$ denote the degrees of the k highest degree vertices of G . Then $\lambda_i(F) = (1 - o(1))\Delta_i^{1/2}$.*

Proof Denote K_{1,d_i} to be a star of degree d_i . Let H be the star forest $H = K_{1,d_1} \cup \dots \cup K_{1,d_k}$ with $d_1 \geq d_2 \geq \dots \geq d_k$. Then for $i = 1, \dots, k$, $\lambda_i(H) = d_i^{1/2}$. So it will be sufficient to show that $\Delta_i(F) = (1 - o(1))\Delta_i(G)$. Claim 5.4.4 shows that the k highest degree vertices G are added before time t_1 . So these vertices are all in F . The only edges to those vertices that are not in F are those added before time t_2 and those incident to S'_3 .

By Theorem 5.4.1 we can choose $f(t)$ such that, $\Delta_1(G_{t_2}^m) \leq t_2^p f(t) \leq t^{7p/16}$. Also by Theorem 5.4.1, $\Delta_i(G) \geq t^p / \log t$. Finally, Claim 5.4.7 says that $|S'_3| \leq t^{7p/16}$ whp. And so, with high probability,

$$\begin{aligned} \Delta_i(F) &\geq \Delta_i(G) - t^{7p/16} - m t^{7p/16} \geq \frac{t^p}{\log t} - t^{7p/16}(1+m) = \frac{t^{p/2}}{\log t} \left[1 - t^{7p/16}(1+m) \frac{\log t}{t^p} \right] \\ &= \frac{t^p}{\log t} \left[1 - (1+m) \frac{\log t}{t^{p-7p/16}} \right] = \frac{t^p}{\log t} \left[1 - (1+m) \frac{\log t}{t^{9p/16}} \right] = (1 - o(1))\Delta_i(G) \end{aligned} \quad (5.81)$$
■

Let $H = G \setminus F$. Denote $\mathbf{A}_G, \mathbf{A}_F$ and \mathbf{A}_H to be the adjacency matrices for graphs G, F and H . In the following claim, we'll show that $\lambda_1(\mathbf{A}_H)$ is $o(\lambda_k(\mathbf{A}_F))$. Consider the fact that if \mathbf{A} and $\mathbf{A} + \mathbf{E}$ are symmetric $n \times n$ matrices, then $\lambda_k(\mathbf{A}) + \lambda_n(\mathbf{E}) \leq \lambda_k(\mathbf{A} + \mathbf{E})$ (see for instance theorem 8.1.5 in Golub and Van Loan [2013]). That implies that for any subspace L ,

$$\max_{\mathbf{x} \in L, \mathbf{x} \neq 0} \frac{\mathbf{x}^T \mathbf{A}_G \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = \max_{\mathbf{x} \in L, \mathbf{x} \neq 0} \frac{\mathbf{x}^T \mathbf{A}_F \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \pm O \left(\max_{\mathbf{x} \neq 0} \frac{\mathbf{x}^T \mathbf{A}_H \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \right).$$

That will be enough to finish the proof because by Rayleigh's Principle [Golub and Van Loan, 2013], $\lambda_i(\mathbf{A}_G) = \lambda_i(\mathbf{A}_F)(1 \pm o(1))$.

Claim 5.4.9 $\lambda_1(\mathbf{A}_H) \leq 6mt^{29/64}$ *whp*.

Proof Let H_i denote the subgraph of H induced by S_i , and let H_{ij} denote the subgraph of H containing only edges with one vertex in S_i and the other in S_j . That is, write \mathbf{A}_H in the following way:

$$\mathbf{A}_H = \begin{bmatrix} \mathbf{H}_1 & \mathbf{H}_{12} & \mathbf{H}_{13} \\ \mathbf{H}_{21} & \mathbf{H}_2 & \mathbf{H}_{23} \\ \mathbf{H}_{31} & \mathbf{H}_{32} & \mathbf{H}_3 \end{bmatrix}$$

We will use this to bound the maximal eigenvalue of \mathbf{A}_H as

$$\begin{aligned} \lambda_1(\mathbf{A}_H) &= \lambda_1 \left(\begin{bmatrix} \mathbf{H}_1 & \mathbf{H}_{12} & \mathbf{H}_{13} \\ \mathbf{H}_{21} & \mathbf{H}_2 & \mathbf{H}_{23} \\ \mathbf{H}_{31} & \mathbf{H}_{32} & \mathbf{H}_3 \end{bmatrix} \right) \\ &\leq \lambda_1(\mathbf{H}_1) + \lambda_1(\mathbf{H}_2) + \lambda_1(\mathbf{H}_3) + \lambda_1(\mathbf{H}_{12}) + \lambda_1(\mathbf{H}_{23}) + \lambda_1(\mathbf{H}_{13}). \end{aligned} \quad (5.82)$$

Note that the maximum eigenvalue of a graph is at most the maximum degree of a graph. By Claim 5.4.6 with $f(t) = t_1$ and $\varepsilon = 1/64$,

$$\begin{aligned} \lambda_1(\mathbf{H}_1) &\leq \Delta_1(\mathbf{H}_1) = \max_{v \leq t_1} \{d_{t_1}^m(v)\} \leq t_1^{p+\varepsilon} \leq t^{65/512} \\ \lambda_1(\mathbf{H}_2) &\leq \Delta_1(\mathbf{H}_2) \leq \max_{t_1 \leq v \leq t_2} \{d_{t_2}^m(v)\} \leq t_2^{p+\varepsilon}/t_1^p \leq t^{457/1024} \\ \lambda_1(\mathbf{H}_3) &\leq \Delta_1(\mathbf{H}_3) \leq \max_{t_2 \leq v \leq t_3} \{d_{t_3}^m(v)\} \leq t_3^{p+\varepsilon}/t_2^p \leq t^{29/64} \end{aligned} \quad (5.83)$$

To bound $\lambda_1(\mathbf{H}_{ij})$, start with $m = 1$. For $i < j$, this implies that each vertex in S_j has at most one edge in \mathbf{H}_{ij} , i.e. H_{ij} is a star forest. Then we have a bound on \mathbf{H}_{ij} by Claim 5.4.8. For $m > 1$, let G' be one of our generated graphs with t edges and $m = 1$. Think now of contracting vertices in G' (only the ones added using preferential attachment) into a single vertex. We can write \mathbf{A}_G in terms of \mathbf{A}'_G : $\mathbf{A}_G = \mathbf{C}^T \mathbf{A}'_G \mathbf{C}$, where \mathbf{C} is a contraction matrix with t rows and the number of columns equal to the number of vertices in \mathbf{A}_G (at most t/m). The i^{th} column is equal to 1 at indices j in which (i, j) are identified. Similarly, we can write \mathbf{H}_{ij} in terms of \mathbf{H}'_{ij} .

Note that if $\mathbf{y} = \mathbf{C}\mathbf{x}$, then $\mathbf{y}^T \mathbf{y} = \mathbf{x}^T \mathbf{C}^T \mathbf{C} \mathbf{x}$, where $\mathbf{C}^T \mathbf{C}$ is a diagonal matrix with 1 's and m 's on the diagonal. So $\mathbf{x}^T \mathbf{x} \leq \mathbf{y}^T \mathbf{y} \leq m \mathbf{x}^T \mathbf{x}$.

$$\begin{aligned} \lambda_1(\mathbf{H}_{ij}) &= \max_{\mathbf{x} \neq 0} \frac{\mathbf{x}^T \mathbf{H}_{ij} \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = \max_{\mathbf{x} \neq 0} \frac{\mathbf{x}^T \mathbf{C}^T \mathbf{H}'_{ij} \mathbf{C} \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = \max_{\mathbf{x} \neq 0, \mathbf{y} = \mathbf{C} \mathbf{x}} \frac{\mathbf{y}^T \mathbf{H}'_{ij} \mathbf{y}}{\mathbf{x}^T \mathbf{x}} \\ &= \max_{\mathbf{x} \neq 0, \mathbf{y} = \mathbf{C} \mathbf{x}} \frac{m \mathbf{y}^T \mathbf{H}'_{ij} \mathbf{y}}{m \mathbf{x}^T \mathbf{x}} \leq \max_{\mathbf{x} \neq 0, \mathbf{y} = \mathbf{C} \mathbf{x}} \frac{m \mathbf{y}^T \mathbf{H}'_{ij} \mathbf{y}}{\mathbf{y}^T \mathbf{y}} \end{aligned} \quad (5.84)$$

Now using Claim 5.4.6 with $f(t) = t_1$ and $\varepsilon = 1/64$,

$$\begin{aligned}\Delta_1(\mathbf{H}'_{12}) &= \max_{\mathbf{v} \leq t_2} \{d'_{t_2}(v)\} \leq t_2^{p+\varepsilon} \leq t^{37/64} \\ \Delta_1(\mathbf{H}'_{23}) &= \max_{t_1 \leq v \leq t_3} \{d'_{t_3}(v)\} \leq t_3^{p+\varepsilon}/t_1^p \leq t^{29/64}\end{aligned}\tag{5.85}$$

Finally, all edges in \mathbf{H}'_{13} are between S_1 and S'_3 , so Claim 5.4.7 shows $\Delta_1(\mathbf{H}'_{13}) \leq t^{p-9/16} \leq t^{7/16}$ whp. Putting together equations (5.84) and (5.85), we get $\lambda_1(\mathbf{H}_{ij}) \leq m\lambda_1(\mathbf{H}'_{ij}) \leq m\Delta_1(\mathbf{H}'_{ij})^{1/2} \leq mt^{29/64}$. And so we get the final bound,

$$\lambda_1(\mathbf{A}_H) \leq \sum_{i=1}^3 \lambda_1(\mathbf{H}_i) + \sum_{i < j} \lambda_1(\mathbf{H}_{ij}) \leq 6mt^{29/64}$$

This shows that $\lambda_i(\mathbf{A}_H)$ is $o(\lambda_k(\mathbf{A}_F))$, which implies $\lambda_i(\mathbf{A}_G) = \lambda_i(\mathbf{A}_F)(1 \pm o(1))$. ■

■

5.5 Analysis of $\text{TGPA}(p_t, r_t, q_t)$

Consider $\text{TGPA}(p_t, r_t, q_t)$, which was described in Section 5.3.2. The parameters p_t, r_t, q_t can change over time, though we will restrict the ways in which the parameters can evolve in Section 5.5.2.

5.5.1 Recursive relation for $m_{k,t}$

Recall that $m_{k,t}$ is the number of nodes at time t with degree k . We wish to write down a relationship for $m_{k,t+1}$ in terms of $m_{k',t}$ for $k' \leq k$. Recall also that the number of edges at time t is $e_t = e_0 + 2t$, and the total sum of degrees at any time t is $2e_t$. Note that for this reason we need only focus on $m_{k,t}$ for $1 \leq k \leq 2e_t$.

Let \mathcal{F}_t denote the σ -algebra generated by the graphs G_0, G_1, \dots, G_t (\mathcal{F}_t holds the history of events up until time t). Fix $k \geq 2$. Since $0 \leq d_{t+1}(v) - d_t(v) \leq 4$ for every node v and time t , we have

$$\mathbb{E}[m_{k,t+1} | \mathcal{F}] = \sum_{\{v: k-4 \leq d_t(v) \leq k\}} \mathbb{P}[d_{t+1}(v) = k].\tag{5.86}$$

Recall $\gamma_t(v)$ from Equation 2.1. Denote $\theta_t(v)$ as 2 times the number of self loops in which v is involved divided by $\sum_{w \in V_{t-1}} d_{t-1}(w)$. (i.e. the proportion of edges which are self loops on v). If $d_{t+1}(v) = 4$, then there are at most 5 possible values for $d_t(v)$ when $k \geq 4$:

1. $d_t(v) = k$. In this case there must have either been a node event not involving v (this occurs with probability $p_{t+1}(1 - 2\gamma_{t+1}(v) + \theta_{t+1}(v))$), or a wedge event not involving v (with probability $r_{t+1}(1 - \gamma_{t+1}(v))(1 - 2\gamma_{t+1}(v) + \theta_{t+1}(v))$), or a component event (with probability q_{r+1}).

2. $d_t(v) = k - 1$. In this case there must have either been a node event where v is involved as the first node (probability $p_{t+1} \cdot \gamma_{t+1}(v) \cdot (1 - \theta_{t+1}(v))$), or where v is involved as the second node (probability $p_{t+1}(\gamma_{t+1}(v) - \theta_{t+1}(v))$), or a wedge event in which v is involved as the first node (with probability $r_{t+1}(\gamma_{t+1}(v) - \gamma_{t+1}(v))^2 - \theta_{t+1}(v) + \gamma_{t+1}(v) \cdot \theta_{t+1}(v)$) or as the third node (probability $r_{t+1}(1 - \gamma_{t+1}(v))(\gamma_{t+1}(v) - \theta_{t+1}(v))$).
3. $d_t(v) = k - 2$. In this case there must have either been a node event in which v is picked as both nodes involved (with probability $p_{t+1} \cdot \theta_{t+1}(v)$) or there must have been a wedge event in which v is involved as the second node (with probability $r_{t+1} \cdot \theta_{t+1}(v)(1 - \gamma_{t+1}(v))$) or as the first and third nodes (with probability $r_{t+1} \cdot \gamma_{t+1}(v)(1 - \gamma_{t+1}(v) + \theta_{t+1}(v))$).
4. $d_t(v) = k - 3$. In this case there must have been a wedge event where v was involved as the first and second nodes or there was a wedge event where v was involved as the second and third nodes (these events occur in combination with probability $2r_{t+1}\gamma_{t+1}(v)(\gamma_{t+1}(v) - \theta_{t+1}(v))$).
5. $d_t(v) = k - 4$. In this case there must have been a wedge event where v is picked for all three wedges, which happens with probability $r_{t+1} \cdot \gamma_{t+1}(v) \cdot \theta_{t+1}(v)$

Let $\alpha_{k,t} = k/(2e_t)$. Then for every v such that $d_t(v) = i$, $\gamma_{t+1}(v) = \alpha_{i,t}$. Define

$$\begin{aligned}
A_{k,t} &= p_{t+1,k}(1 - 2\alpha_{k,t} + \theta_{t+1}(v)) + r_{t+1}(1 - \alpha_{k,t})(1 - 2\alpha_{k,t} + \theta_{t+1}(v)) + q_{t+1}, \\
B_{k,t} &= 2p_{t+1}(\alpha_{k,t} - \theta_{t+1}(v)) + 2r_{t+1}(1 - \alpha_{k,t})(\alpha_{k,t} - \theta_{t+1}(v)), \\
C_{k,t} &= p_{t+1}\theta_{t+1}(v) + r_{t+1}(\alpha_{k,t} - \alpha_{k,t}^2 + \theta_{t+1}(v)), \\
D_{k,t} &= 2r_{t+1}\alpha_{k,t}(\alpha_{k,t} - \theta_{t+1}(v)), \text{ and } E_{k,t} = r_{t+1}\alpha_{k,t}\theta_{t+1}(v).
\end{aligned}$$

Then $A_{k,t} + B_{k,t} + C_{k,t} + D_{k,t} + E_{k,t} = 1$ and $A_{k,t}, B_{k,t}, C_{k,t}, D_{k,t}, E_{k,t} \geq 0$ for every $0 \leq k \leq 2e_t$. Also, by equation 5.86, for every $k \geq 4$

$$\begin{aligned}
\mathbb{E}[m_{k,t+1}|\mathcal{F}] &= m_{k,t}A_{k,t} + m_{k-1,t}B_{k-1,t} + m_{k-2,t}C_{k-2,t} \\
&\quad + m_{k-3,t}D_{k-3,t} + m_{k-4,t}E_{k-4,t}.
\end{aligned} \tag{5.87}$$

And for remaining values of k we have

$$\begin{aligned}
\mathbb{E}[m_{3,t+1}|\mathcal{F}] &= m_{3,t}A_{3,t} + m_{2,t}B_{2,t} + m_{1,t}C_{1,t} \\
\mathbb{E}[m_{2,t+1}|\mathcal{F}] &= m_{2,t}A_{2,t} + m_{1,t}B_{1,t} + p_{t+1} + q_{t+1} \\
\mathbb{E}[m_{1,t+1}|\mathcal{F}] &= m_{1,t}A_{1,t} + 2q_{t+1}.
\end{aligned} \tag{5.88}$$

Define

$$X_{k,t} = \begin{cases} m_{k-1,t}B_{k-1,t} + m_{k-2,t}C_{k-2,t} + m_{k-3,t}D_{k-3,t} + m_{k-4,t}E_{k-4,t} & k \geq 4 \\ m_{2,t}B_{2,t} + m_{1,t}C_{1,t} & k = 3 \\ m_{1,t}B_{1,t} + p_{t+1} + q_{t+1} & k = 2 \\ 2q_{t+1} & k = 1 \end{cases} \quad (5.89)$$

Then equations 5.87 and 5.88 can be re-written as

$$\mathbb{E}[m_{k,t+1}] = \mathbb{E}[m_{k,t}] \cdot A_{k,t} + \mathbb{E}[X_{k,t}] \quad (5.90)$$

5.5.2 Power-law in TGPA(p_t, r_t, q_t)

The following lemma is presented in Avin et al. [2017] and is a quick generalization of a result in Chung et al. [2006].

Lemma 5.5.1 ([Avin et al., 2017]) *Suppose that a sequence satisfies the recurrence relation $a_{t+1} = (1 - b_t/(t + t_1))a_t + c_t$ for $t \geq t_0$. Furthermore, let $\{s_t\}$ be a sequence of real numbers with $\lim_{t \rightarrow \infty} s_t/s_{t+1} = 1$, $d_t = t(1 - s_t/s_{t+1})$, $\lim_{t \rightarrow \infty} b_t = b$, $\lim_{t \rightarrow \infty} c_t \cdot t/s_t = c$, $\lim_{t \rightarrow \infty} d_t = d$, and $b + d > 1$. Then $\lim_{t \rightarrow \infty} a_t/s_t$ exists and $\lim_{t \rightarrow \infty} a_t/s_t = c/(b + d)$.*

The following theorem and corollary prove that TGPA(p_t, r_t, q_t) has a power-law in the degree distribution, which we can analyze.

Theorem 5.5.1 *Consider TGPA(p_t, r_t, q_t). Let $y_t = p_t + 3q_t$. Assume that $\lim_{t \rightarrow \infty} y_t = y < 3$, $\sum_{t=1}^{\infty} y_t = \infty$, and $\lim_{t \rightarrow \infty} t \cdot y_{t+1} / \sum_{j=1}^t y_j = \Gamma > 0$. Then letting $\beta = 1 + 2\Gamma/(3 - y)$, the limit $M_k = \lim_{t \rightarrow \infty} \mathbb{E}[m_{k,t}] / \mathbb{E}[n_t]$ exists for every $k \geq 1$ and*

$$M_k = \frac{\Gamma}{\Gamma + 3/2 - y/2} \prod_{j=1}^{k-1} \frac{j}{j + \beta}.$$

Proof This proof will be an induction on k . For $k = 1$ we use Lemma 5.5.1 setting $(t_1, s_t, a_t, b_t, c_t) = (e_0, \mathbb{E}[n_t], \mathbb{E}[m_{1,t}], e_t(1 - A_{1,t}), y_{t+1})$. Using Equation 5.90, this gives the limits $b = 3/2 - y/2$, and $c = d = \Gamma$, which concludes the base case. Now assume the Theorem holds for $k - 1$, we now prove it for k . Again use Lemma 5.5.1, this time with $(t_1, s_t, a_t, b_t, c_t) = (e_0, \mathbb{E}[n_t], \mathbb{E}[m_{k,t}], B_{k-1,t} \mathbb{E}[m_{k-1,t}] + C_{k-1,t} \mathbb{E}[m_{k-2,t}] + D_{k-3,t} \mathbb{E}[m_{k-3,t}] + E_{k-4,t} \mathbb{E}[m_{k-4,t}])$. Then we get $d = \Gamma$, $b = k \cdot (3/2 - y/2)$, and using the inductive hypothesis,

$$c = \lim_{t \rightarrow \infty} \frac{c_t \cdot t}{s_t} = (k - 1) \left(\frac{3}{2} - \frac{y}{2} \right) M_{k-1}.$$

Therefore M_k exists and

$$M_k = \frac{(k-1)(3/2 - y/2)M_{k-1}}{k(3/2 - y/2) + \Gamma} = \frac{k-1}{k-1+\beta} M_{k-1}.$$

■

The proof of the following corollary follows exactly from [Avin et al., 2017].

Corollary 5.5.1 *Under the assumptions in Theorem 5.5.1, M_k is proportional to $k^{-\beta}$.*

Finally, we can state which power-law exponents are obtainable.

Lemma 5.5.2 *For any $x \in (1, \infty)$, there exists a choice of p_t, r_t, q_t such that in $TGPA(p_t, r_t, q_t)$ the resulting network follows a power-law in the degree distribution with exponent $\beta = x$.*

Proof We can use three separate cases:

1. For $x \in (5/3, \infty)$, setting $y_t = 3 - 2/(x-1)$ gives exponent $\beta = 1 + 2/(3 - (3 - 2/(x-1))) = x$.
2. For $x \in (1, 5/3)$, set $y_t = t^{3/2(x-5/3)}$. Then

$$\begin{aligned} \Gamma &= \lim_{t \rightarrow \infty} \frac{y_{t+1} \cdot t}{\sum_{j=1}^t y_j} = \lim_{t \rightarrow \infty} \frac{(t+1)^{3/2(x-5/3)} \cdot t}{\sum_{j=1}^t (j^{3/2(x-5/3)})} = \lim_{t \rightarrow \infty} \frac{t^{3/2x-3/2}}{\int_{j=0}^t j^{3/2(x-5/3)} dj} \\ &= \lim_{t \rightarrow \infty} \frac{(3/2x - 3/2)t^{3/2x-3/2}}{j^{3/2x-3/2} \Big|_{j=0}^t} = 3/2x - 3/2 \end{aligned}$$

Then $\beta = 1 + (2\Gamma)/(3 - y) = 1 + 2(3/2x - 3/2)/(3 - 0) = x$.

3. For $x = 5/3$, set $y_t = 1/\ln(t+2)$ for every t . Then we have

$$\Gamma = \lim_{t \rightarrow \infty} \frac{y_{t+1} \cdot t}{\sum_{j=1}^t y_j} = \lim_{t \rightarrow \infty} \frac{t/\ln(t+3)}{\sum_{j=1}^t 1/\ln(j+2)} = \lim_{t \rightarrow \infty} \frac{t/\ln(t+3)}{t/\ln t} = 1$$

Then $TGPA(p_t, r_t, q_t)$ follows a power law degree distribution with exponent $\beta = 1 + 2\Gamma/(3 - y) = 1 + 2/(3 - 0) = 5/3$.

■

For a final analysis, we show that the component portion is necessary to obtain the full power-law exponent range $(1, \infty)$. Lemma 5.5.3 comes directly from Avin et al. [2017].

Lemma 5.5.3 (Avin et al. [2017]) *Assume $\lim_{t \rightarrow \infty} y_t = y$ and $\lim_{t \rightarrow \infty} y_{t+1} \cdot t / \sum_{j=1}^t j_j = \Gamma$. Then for $y > 0$ we have $\Gamma = 1$, and for $y = 0$ we have $\Gamma \leq 1$.*

Corollary 5.5.2 *Consider $TGPA(p_t, r_t, q_t)$. Assume that $\lim_{t \rightarrow \infty} q_t = 0$, $\lim_{t \rightarrow \infty} y_t = y$, and $y_{t+1}t / \sum_{j=1}^t y_j = \Gamma > 0$. Then the resulting graph follows a power law degree distribution with exponent $\beta \in (1, 3]$.*

Proof By Corollary 5.5.1, $TGPA(p_t, r_t, q_t)$ follows a power-law in the degree distribution with exponent $\beta = 1 + 2\Gamma/(3 - y) > 1$. By Lemma 5.5.3, for $0 < y \leq 1$, we have $\beta = 1 + 2/(3 - y) \in (5/3, 3]$ and for $y = 0$ we have $\beta = 1 + 2\Gamma/3 \leq 5/3$.

■

5.6 Significant Clustering coefficients

Table 5.1.

Clustering coefficients in TGPA model. TGPA is able to generate data with much larger clustering coefficients, compared to GPA.

Network name	edges	global clust	local clust	HO global	HO local
<i>Auburn (18k vertices)</i>	974k	0.137	0.223	0.107	0.172
TGPA(18k,0.987,10,150):	640k	0.25	0.22	0.118	0.03
GPA(18k,0.001,0.999,2):	906k	0.021	0.030	0.005	0.014
<i>Berkeley (13k vertices)</i>	852k	0.114	0.207	0.0876	0.156
TGPA(13k,0.99, 10, 58)	502k	0.104	0.185	0.034	0.025
GPA(13k,0.001,0.999,2)	502k	0.024	0.034	0.005	0.015
<i>Princeton (7k vertices)</i>	293k	0.237	0.164	0.091	0.146
TGPA(7k,0.987,10,100):	207k	0.298	0.251	0.148	0.053
GPA(7k,0.001,0.999,2):	255k	0.038	0.054	0.009	0.025

We analyzed 3 networks from the Facebook 100 dataset [Traud et al., 2012], each of which is a set of users at a particular university. We computed the global clustering coefficient, average local clustering coefficient, and *higher-order* clustering coefficients (see Section 2.4 for the definitions of these coefficients).

To fit the TGPA(p, q) model (Section 5.3.1) to the real world networks, we noted that the average degree of our model, the total degrees divided by the number of nodes, is approximately $(2m(1-p) + 2m)/(m(1-p) + 1)$. Choosing the average degree gives a relationship between parameters m and p . We tested various sets of parameters to obtain the best possible fit. We started both TPGA and GPA with a k -node clique. Table 5.1 lists the parameters we chose for the TGPA model as TGPA(n, p, k, m), which produces an n node graph starting from a k node clique. For comparison we also fit the GPA model (Section 5.1.1). The parameters in Table 5.1 are GPA(n, p, r, k). Notice that TGPA maintains much more significant clustering coefficients across all measures.

5.7 The Eigenvalue Power-law is robust

As discussed at length already, preferential attachment has long been used to describe the reason why we find power-law distributions in the degrees of real world networks. There are many other empirical and theoretical studies on the presence of power-laws in spectra [Chung et al., 2003a, Goh et al., 2001, Mihail and Papadimitriou, 2002, Eikmeier and Gleich, 2017]. Given that many real-world networks should have power-laws in both the eigenvalues and the degrees, this suggests that one should be easier and more reliable to detect than the other. Chapter 3 [Eikmeier and Gleich, 2017] gives evidence that power-laws are more likely to be present in the spectra than in degree distributions. An explanation for this observation may come from the way in which we obtain data, rather than a true feature of the data itself. Consider for a moment that the “real data” that is used in so many studies is not the full set of data. Instead, due to sampling or missing data the “real data” is actually some perturbation of the true set. If the underlying graph has both a power-law in the degrees and eigenvalues, then it is possible this observation just reflects the robustness of the eigenvalue power-law to the type of network sampling that occurred. There are many methods of sampling graphs and studying properties of sampled graphs is a well-studied field [Leskovec and Faloutsos, 2006, Stumpf and Wiuf, 2005, Stumpf et al., 2005, Lovász et al., 1993, Lee et al., 2006, Ebrahimi et al., 2017, Schoenebeck, 2013].

Because the TGPA model produces graphs with reliable power-law exponents in *both* the degrees and spectra, as well as clustering, (Sections 5.5, 5.4) this makes it a good model to study this potential effect. Of course, TGPA isn’t the only model with power-laws in both the degrees and eigenvalues, as we’ve proved about the GPA model in this chapter. We generated 35 TGPA graphs of size 5000 with theoretical degree power-law exponents between 2 – 5, and separately 35 GPA graphs with the same parameters. For each graph, we detect that it has a statistically significant power-laws in both the degrees and spectra. The distributions were tested for power-laws using the method of Clauset et al. [2009]. We then perturbed each of the networks in three ways: In the first method we sampled random sets of edges of the graphs; in the second method we did a depth first search, starting at a random seed node; and third we did a forest fire sampling procedure from a random seed node (at each time step a fire “spreads” to each neighbor with some probability based on a burn rate). In each case, we ran the perturbation until a certain percentage of the nodes were obtained. And in each case we did the perturbation 50 times.

The results of this experiment on the TGPA model are shown in violin plots in Figure 5.2. Notice that the degree plots have a much larger spread in most cases, and the spectra almost always retains its power-law. Random edge sampling appears to be the only case where the degree power-law may

be argued to remain more intact, but then again sampling random edges is the least natural of the three.

When trying the same experiment on PA models, we don't see as much variation between the degrees and spectra. See Figure 5.3 for an example of the forest fire sampling procedure. The other sampling procedures give similar results. We believe that the local structure of TGPA is necessary to see the effects of sampling.

We additionally ran the same sampling experiment on a citations network which comes from [Leskovec et al., 2005a]. *HEP-PH* is a network of citations between papers in high energy physics with 34k nodes and 421k edges. We symmetrized the dataset, and found it to have a statistically significant power-law in both the degrees and spectra. Figure 5.4 shows the results of sub-sampling using the three same methods used previously. Note that there is no distribution in this case (no violin plot), since we are sampling only one graph. In all three sampling procedures, the spectra retained always retained the power-law (50 out of 50 sampled networks), while the degree distribution of the sampled networks retains a power-law much less frequently.

5.8 Summary

Preferential attachment models are a common class of graph models which have been used to explain why power-law distributions appear in the degree sequences of real network data. One of the things they lack, however, is higher-order network clustering, including non-trivial clustering coefficients. In this chapter we presented a specific Triangle Generalized Preferential Attachment Model (TGPA) that, by construction, has nontrivial clustering. We further prove that this model has a power-law in both the degree distribution and eigenvalue spectra. We use this model to investigate further the finding that power-laws are more reliably observed in the eigenvalue spectra of real-world networks than in their degree distribution (from Chapter 3). One conjectured explanation for this is that the spectra of the graph is more robust to various sampling strategies that would have been employed to collect the real-world data compared with the degree distribution.

Consequently, we generate random TGPA models that provably have a power-law in both, and sample subgraphs via forest fire, depth-first, and random edge models. We find that the samples show a power-law in the spectra even when only 30% of the network is seen. Whereas there is a large chance that the degrees will not show a power-law. Our TGPA model shows this behavior much more clearly than a standard preferential attachment model. This provides one possible explanation for why power-laws may be seen frequently in the spectra of real world data.

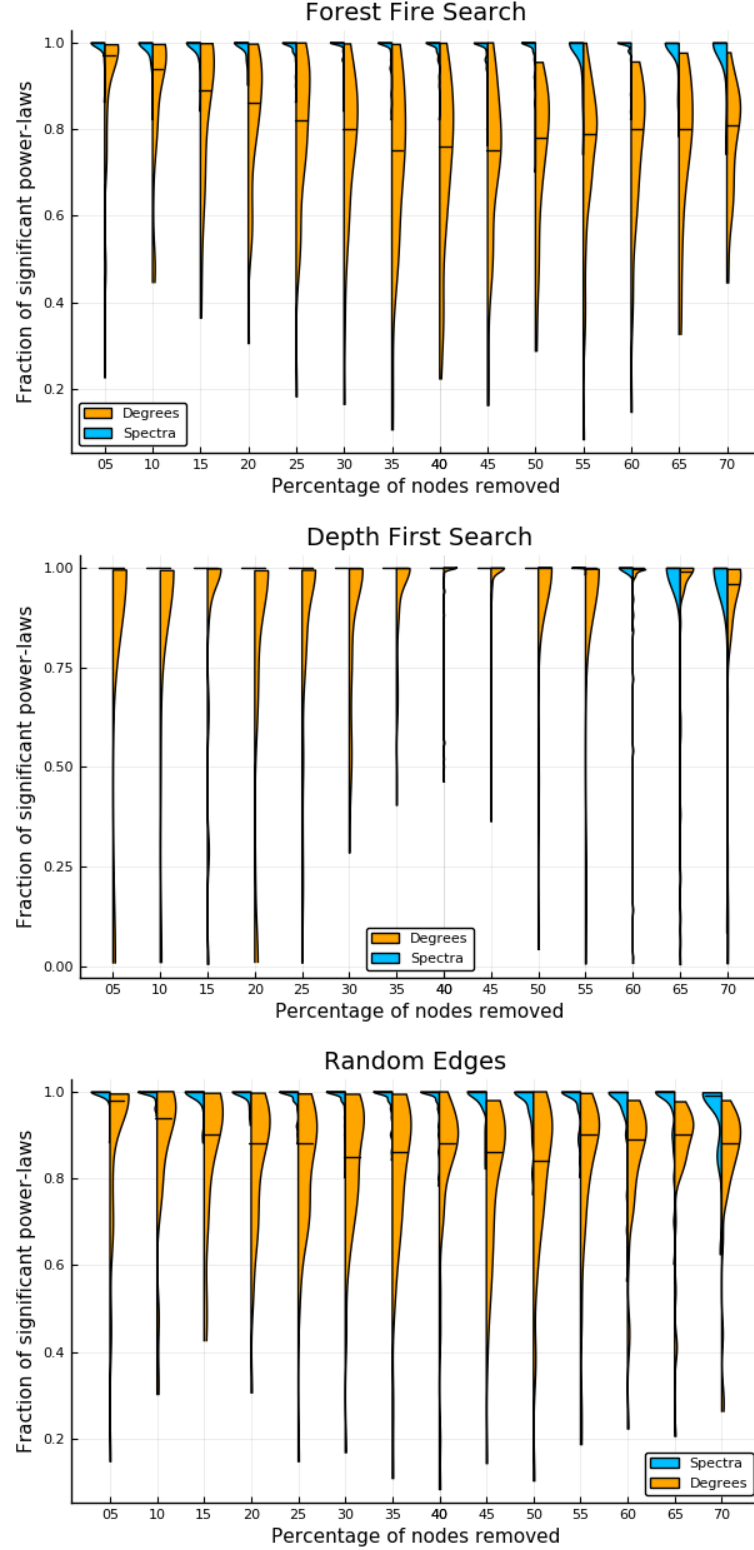


Figure 5.2. **Power-law in spectra is more robust to sampling.** 35 TGPA graphs with power-law exponents between 2-5, sub-sampled in various ways. On the left, the graphs were sampled using a forest fire search on a random seed node; in the middle a depth first search on a random seed node; and on the right, the graphs were perturbed by sampling random edges. Note that when there appears to be *no* violin plot (e.g. most spectra in DFS) that means 100% of the sampled graphs had significant power-laws. The horizontal lines give the median.

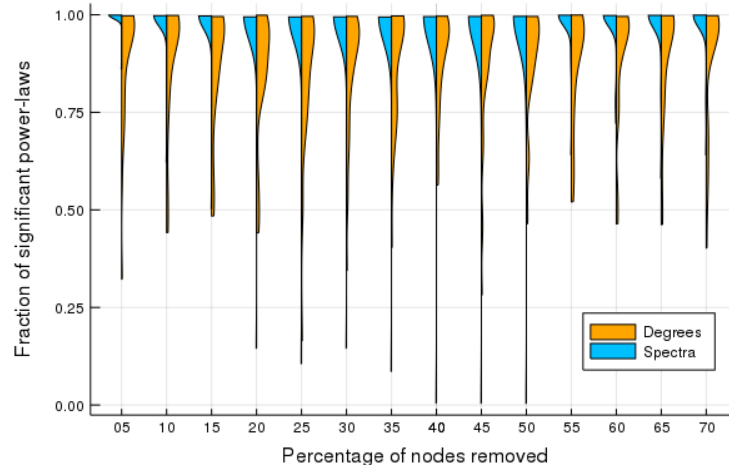


Figure 5.3. **Forest Fire Sampling** graphs generated using the preferential attachment model.

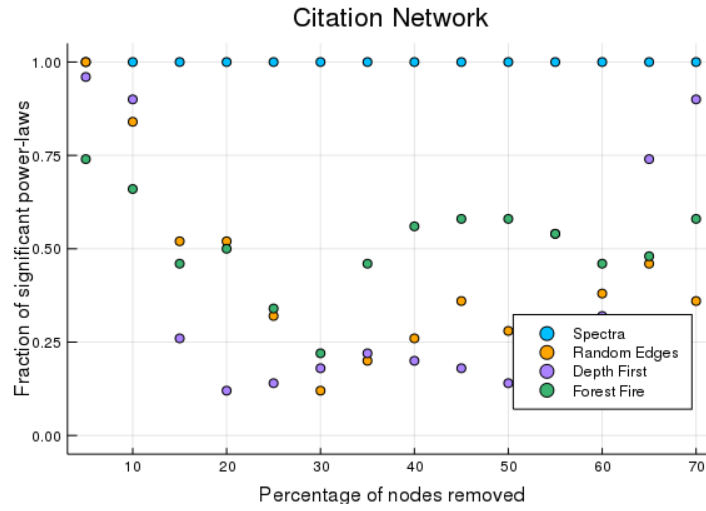


Figure 5.4. **Effects of Sampling a Real World Network.** When running the same sampling experiment on a real world dataset, we see similar results as on the TGPA model.

6. CONCLUSIONS AND FUTURE WORK

The foundation of this thesis was borne from the question: How well do features touted in graph models actually fit to real-world networks? A power-law in the degree distribution of a network was one of the hallmarks of the early study on web graphs and other types of information networks [Barabási and Albert, 1999]. This initial focus on power-laws then led to a number of theoretical studies about the presence of power-laws in other features of the network including the singular value distribution and Laplacian eigenvalues [Chung et al., 2003a, Elsässer, 2006, Mihail and Papadimitriou, 2002] – as well as work critical of this finding [Achlioptas et al., 2005]. In the start of this thesis, we have conducted a wide-ranging evaluation of these conjectured relationships and discovered that: (i) the presence of a power-law in the largest singular values is more reliable than power-laws in the degree distribution; (ii) this power-law often applies to at least $n^{(1/2)}$ largest singular values, and for some classes of networks, up to $n^{(2/3)}$. Moreover, we find compelling empirical evidence of the relationships from Elsässer [2006], which posits that the Laplacian eigenvalues and the degree distribution should have similar power-law exponents and behavior. Finally, we found that empirical power-law exponents are far larger than previously discussed, which may impact how we create random networks for statistical tests on networks and the applicability of existing results on power-law graphs (e.g. [Kurauskas and Bloznelis, 2013, Gleich and Seshadhri, 2012, Cooper et al., 2012, Latapy, 2008, Watts, 2003]).

Following the empirical study, we offer an updated Kronecker model and an updated Preferential attachment model. Both proposals maintain desirable features of their predecessors while offering new advantages, particularly through their use of higher-order structure.

When it comes to graph modeling, there is no model which is suited to every situation. Just as data comes in all different forms, graph models emphasize varying aspects of real world data. One thing that graph models *must* feature is usability, that is they must be efficient enough to generate instances of large networks. Naïvely implemented, our HyperKron model would be extremely inefficient to generate. We proved that the HyperKron model can be generated efficiently, and validated the run-time via experimentation. It is also easy to simulate in parallel – you can parallelize over the Erdős-Rényi regions, for instance. HyperKron associates hyper-edges with motifs, hence is particularly useful to model data with inherent motif structure (such as triangles). It is even flexible enough to model directed, signed motifs, as we showed the *C. Servisae* example. It is not obvious how to generate these types of structures for models based on matrices of probabilities such

as Erdős-Rényi, Chung-Lu, or kernel functions [Hagberg and Lemons, 2015]. The same critique holds for evolutionary models such as the copying model or forest-fire model. HyperKron is also well suited to model graphs with significant clustering coefficients (as is the case is much of real-world data).

Our novel Triangle Preferential Attachment model also possesses significant clustering coefficients, over traditional Preferential Attachment models. It directly incorporates a triangle structure into the generation process. It also has a provable power-law distribution in both the degrees and eigenvalues (something that HyperKron lacks, as well as others models Holme and Kim [2002], Lattanzi and Sivakumar [2009]).

That said, there are other types of network models that possess clustering. Newman [2009] studied a configuration model that incorporated the triangle degree of each node. Kolda et al. [2014] proposed the BTER model that has large clustering coefficients and a reasonable match. These are both excellent models with clustering, but is unclear how to incorporate more complex types of structure such as signs into these models. Likewise, models that randomly generate points for each node and then connect nearby nodes based on a metric space are often known to have non-trivial local clustering [Bonato et al., 2012, Jacob and Mörters, 2015]. However, these models tend to be unrealistically dense if the geometry is not sufficiently high dimensional, at which point you lose local clustering.

Since the Triangle Generalized Preferential Attachment model has provable power-laws in both the degrees and eigenvalues, it makes it an ideal scenario to explore further these distributions. We showed that the spectra is far more stable to network sampling than the degrees in this model, as hypothesized in the study of real-world data. We provide this experiment as evidence for one possibly reason why we may see power-law distributions in the spectra of real networks more often than in the degrees.

There are a number of open avenues to extend the work of this thesis. In the HyperKron model for example, we were able to hand tune parameters in Section 3.2.2, by getting the number of edges to match. However it would be nice to have an automated fitting technique similar to Gleich and Owen [2012] or Leskovec et al. [2010]. As discussed in Section 4.4, there are a number of duplicate edges placed in the HyperKron model, which makes it difficult to estimate features of HyperKron, so this area remains one of our most important next steps.

Another extension to the study of HyperKron is to fit the model to data sets with larger motifs, using 4th and 5th order tensors. We also illustrated the lack of higher-order clustering in the HyperKron models. We believe this aspect is a feature of the model as it enables testing specific sources of clustering structure. For instance, if the goal is to test hypotheses about 2nd order clustering structure in the network, then the lack of higher-order structure is useful. Additionally, the HyperKron model would also extend to larger size motifs such a four-clique and five-cliques through

a suitable adaptation of the HyperKron sampling procedure to 4th and 5th order tensors. These models will exhibit higher-order clustering. We also plan to explore these settings in the future.

Another area of open research is to study further generalizations of higher-order preferential attachment graphs. More interestingly, it would be beneficial to explore further the results on perturbation of graphs to the degree and eigenvalue distributions. We wonder if there may be some local properties of real-world graphs (or the TGPA model specifically) that can be used to analyze the stability of these distributions.

REFERENCES

- Google ngram viewer. <https://books.google.com/ngrams>. Accessed: 2019-2-20.
- Dimitris Achlioptas, Aaron Clauset, David Kempe, and Cristopher Moore. On the bias of traceroute sampling: or, power-law degree distributions in regular graphs. In *STOC 05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 694–703. ACM Press, 2005.
- William Aiello, Fan Chung, and Linyuan Lu. A random graph model for massive graphs. In *STOCSTOC '00 Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 171–180. ACM, 2000. URL <http://dl.acm.org/citation.cfm?id=335326>.
- Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Rtm: Laws and a recursive generator for weighted time-evolving graphs. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 701–706. IEEE, 2008.
- R. Alberich, J. Miro-Julia, and F. Rossello. Marvel universe looks almost like a real social network. *arXiv*, cond-mat.dis-nn:0202174, 2002. URL <http://arxiv.org/abs/cond-mat/0202174>.
- Chen Avin, Zvi Lotker, Yinon Nahum, and David Peleg. Improved degree bounds and full spectrum power laws in preferential attachment networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 45–53. ACM, 2017.
- Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999. URL <http://science.sciencemag.org/content/286/5439/509>.
- Albert-László Barabási, Hawoong Jeong, and Réka Albert. The www network. Accessed online via <http://www.nd.edu/~networks/resources.htm> in May 2009, 1999. URL <http://www.nd.edu/~networks/resources.htm>.
- Vladimir Batagelj and Ulrik Brandes. Efficient generation of large random networks. *Physical Review E*, 71(3), March 2005. URL <https://journals.aps.org/pre/abstract/10.1103/PhysRevE.71.036113>.
- Vladimir Batagelj and Andrej Mrvar. Pajek datasets. <http://vlado.fmf.uni-lj.si/pub/networks/data/>, 2006.
- Mohsen Bayati, Jeong Kim, and Amin Saberi. A sequential algorithm for generating random graphs. *Algorithmica*, 58(4):860–910, 2010. ISSN 0178-4617. doi: 10.1007/s00453-009-9340-1. 10.1007/s00453-009-9340-1.
- Edward A Bender and E.Rodney Canfield. The asymptotic number of labeled graphs with given degree sequences. *J. Comb. Theory, A*, 24(3):296 – 307, 1978. ISSN 0097-3165. doi: 10.1016/0097-3165(78)90059-6.
- Austin Benson, David F. Gleich, and Lek-Heng Lim. The spacey random walk: a stochastic process for higher-order data. *SIAM Review*, 59(2):321–345, May 2017. doi: 10.1137/16M1074023. URL <http://arxiv.org/abs/1602.02102>.
- Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

- Shankar Bhamidi, Guy Bresler, and Allan Sly. Mixing time of exponential random graphs. *Ann. Appl. Probab.*, 21(6):2146–2170, 12 2011. doi: 10.1214/10-AAP740.
- Marián Boguñá, Romualdo Pastor-Satorras, Albert Díaz-Guilera, and Alex Arenas. Models of social networks based on social distance attachment. *Phys. Rev. E*, 70(5):056122, Nov 2004. doi: 10.1103/PhysRevE.70.056122.
- Béla Bollobás, Svante Janson, and Oliver Riordan. Sparse random graphs with clustering. *Random Structures and Algorithms*, 38(3):269–323, 2011. ISSN 1098-2418. doi: 10.1002/rsa.20322. URL <http://dx.doi.org/10.1002/rsa.20322>.
- Anthony Bonato, Jeannette Janssen, and Paweł Prallat. Geometric protean graphs. *Internet Mathematics*, 8(1-2):2–28, 2012. doi: 10.1080/15427951.2012.625246.
- Francesco Bonchi, Pooya Esfandiar, David F. Gleich, Chen Greif, and Laks V.S. Lakshmanan. Fast matrix computations for pairwise and columnwise commute times and Katz scores. *Internet Mathematics*, 8(1-2):73–112, 2012. doi: 10.1080/15427951.2012.625256.
- Blai Bonet. Efficient algorithms to rank and unrank permutations in lexicographic order. *AAAI-workshop on Search in AI and Robotics*, 2008.
- Anna D Broido and Aaron Clauset. Scale-free networks are rare. *arXiv preprint arXiv:1801.03400*, 2018.
- Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-MAT: A recursive model for graph mining. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 442–446, 2004. doi: 10.1137/1.9781611972740.43.
- Fan Chung and Linyuan Lu. Connected components in random graphs with given expected degree sequences. *Annals of Combinatorics*, 6(2):125–145, 2002. ISSN 0219-3094. doi: 10.1007/PL00012580.
- Fan Chung, Linyuan Lu, and Van Vu. Eigenvalues of random power law graphs. *Annals of Combinatorics*, 7:21–33, 2003a.
- Fan Chung, Linyuan Lu, and Van Vu. Spectra of random graphs with given expected degrees. In *Proceedings of the National Academy of Sciences*, volume 100, pages 6313–6318. PNAS, May 2003b.
- Fan Chung, Fan RK Chung, Fan Chung Graham, Linyuan Lu, Kian Fan Chung, et al. *Complex graphs and networks*. Number 107. American Mathematical Soc., 2006.
- Aaron Clauset, Cosma Rohilla Shalizi, and M.E.J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009. URL <http://epubs.siam.org/doi/abs/10.1137/070710111>.
- Paul G. Constantine and David F. Gleich. Using polynomial chaos to compute the influence of multiple random surfers in the PageRank model. In *Proceedings of the 5th Workshop on Algorithms and Models for the Web Graph*, volume 4863 of *LNCS*, pages 82–95. Springer, 2007. doi: 10.1007/978-3-540-77004-6_7.
- Colin Cooper and Alan M Frieze. A general model of undirected web graphs. In *European Symposium on Algorithms*, pages 500–511. Springer, 2001.
- Colin Cooper, Tomasz Radzik, and Yiannis Siantos. A fast algorithm to find all high degree vertices in power law graphs. In *Proceedings of the 21st International Conference on World Wide Web*, pages 1007–1016. ACM, 2012.
- Timothy A. Davis and Yifan Hu. The University of Florida sparse matrix collection. *ACM Trans. Math. Softw.*, 38(1):1:1–1:25, December 2011. ISSN 0098-3500. doi: 10.1145/2049662.2049663.

- Inderjit S. Dhillon, Beresford N. Parlett, and Christof Vömel. The design and implementation of the MRRR algorithm. *ACM Trans. Math. Softw.*, 32(4):533–560, December 2006. ISSN 0098-3500. doi: 10.1145/1186785.1186788.
- Roozbeh Ebrahimi, Jie Gao, Golnaz Ghasemiefteh, and Grant Schoenbeck. How complex contagions spread quickly in preferential attachment models and other time-evolving networks. *IEEE Transactions on Network Science and Engineering*, 4(4):201–214, 2017.
- Nicole Eikmeier and David F. Gleich. Revisiting power-law distributions in spectra of real world networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 817–826, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4887-4. doi: 10.1145/3097983.3098128. URL <http://doi.acm.org/10.1145/3097983.3098128>.
- Nicole Eikmeier, Arjun S. Ramani, and David F. Gleich. The HyperKron graph model for higher order features. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 941–946. IEEE, 2018.
- Robert Elsässer. Toward the eigenvalue power law. In *International Symposium on Mathematical Foundations of Computer Science*, pages 351–362. Springer, 2006.
- Paul Erdős and Albert Rényi. On random graphs i. *Publicationes Mathematicae*, 6:290–297, 1959.
- Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, 1999.
- C. T. Fan, MerviMer. Muller, and Ivan Rezucha. Development of sampling plans by using sequential (item by item) selection techniques and digital computers. *Journal of the American Statistical Association*, 57(298):387–402, June 1962.
- Abraham Flaxman, Alan Frieze, and Trevor Fenner. High degree vertices and eigenvalues in the preferential attachment graph. *Internet Mathematics*, 2(1):1–19, 2005.
- Bailey K. Fosdick, Daniel B. Larremore, Joel Nishimura, and Johan Ugander. Configuring random graph models with fixed degree sequences. *SIAM Review*, 60(2):315–355, 2018. doi: 10.1137/16M1087175. URL <https://arxiv.org/abs/1608.00607>.
- John R. Gilbert and Shang-Hua Teng. MESHPART: Matlab mesh partitioning and graph separator toolbox. <http://www.cerfacs.fr/algor/Softs/MESHPART/>, February 2002. URL <http://www.cerfacs.fr/algor/Softs/MESHPART/>.
- Minas Gjoka, Maciej Kurant, Carter T. Butts, and Athina Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *IEEE INFOCOM*. IEEE, 2010. URL <http://ieeexplore.ieee.org/document/5462078/>.
- David F. Gleich. A repository of graph spectra. In preparation, 2019.
- David F. Gleich and Art B. Owen. Moment based estimation of stochastic Kronecker graph parameters. *Internet Mathematics*, 8(3):232–256, August 2012. doi: 10.1080/15427951.2012.680824.
- David F. Gleich and C. Seshadhri. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *KDD2012*, pages 597–605, August 2012. doi: 10.1145/2339530.2339628.
- PABLO M. Gleiser and LEON Danon. Community structure in jazz. *Advances in Complex Systems*, 06(04):565–573, 2003. doi: 10.1142/S0219525903001067.
- K-I Goh, Byungnam Kahng, and D Kim. Spectra and eigenvectors of scale-free networks. *Physical Review E*, 64(5):051903, 2001.

- Michel L Goldstein, Steven A Morris, and Gary G Yen. Problems with fitting to the power-law distribution. *The European Physical Journal B*, 41:255–258, September 2004. URL <http://link.springer.com/article/10.1140/epjb/e2004-00316-5>.
- Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 4th edition, 2013.
- Jacopo Grilli et al. Higher-order interactions stabilize dynamics in competitive network models. *Nature*, 548:210–213, 2017. doi: 10.1038/nature23273.
- R. Guimerà et al. Self-similar community structure in a network of human interactions. *Phys. Rev. E*, 68:065103, 2003. doi: 10.1103/PhysRevE.68.065103.
- Aric Hagberg and Nathan Lemons. Fast generation of sparse random kernel graphs. *PLOS ONE*, 10(9):e0135177, sep 2015. doi: 10.1371/journal.pone.0135177. URL <https://doi.org/10.1371/journal.pone.0135177>.
- Asger Hoedt. Morton codes. <http://asgerhoedt.dk/?p=276>, October 2012. URL <http://asgerhoedt.dk/?p=276>.
- Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983. ISSN 0378-8733. doi: 10.1016/0378-8733(83)90021-7.
- Petter Holme and Beom Jun Kim. Growing scale-free networks with tunable clustering. *Physical review E*, 65(2):026107, 2002.
- Bernardo A. Huberman. *The Laws of the Web*. The MIT Press, Cambridge, Massachusetts, 2001. URL <https://mitpress.mit.edu/books/laws-web>.
- Emmanuel Jacob and Peter Mörters. Spatial preferential attachment networks: Power laws and clustering coefficients. *The Annals of Applied Probability*, 25(2):632–662, 2015.
- Myunghwan Kim and Jure Leskovec. Multiplicative attribute graph model of real-world networks. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 62–73. Springer, 2010.
- Gunnar Klau. A new graph-based method for pairwise global network alignment. *BMC Bioinformatics*, 10(Suppl 1):S59, January 2009. ISSN 1471-2105. doi: 10.1186/1471-2105-10-S1-S59.
- Donald E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley, 1993.
- Donald Ervin Knuth. *The art of computer programming*, volume 3. 1997.
- Tamara G. Kolda, Ali Pinar, Todd Plantenga, and C. Seshadhri. A scalable generative graph model with community structure. *SIAM Journal on Scientific Computing*, 36(5):C424–C452, 2014. doi: 10.1137/130914218.
- Valentas Kurauskas and Mindaugas Bloznelis. Large cliques in sparse random intersection graphs. *arXiv*, math.CO:1302.4627, 2013. URL <http://arxiv.org/pdf/1302.4627.pdf>.
- Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In *WWW ’10: Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772751.
- Renaud Lambiotte, Martin Rosvall, and Ingo Scholtes. From networks to optimal higher-order models of complex systems. *Nature Physics*, March 2019. doi: 10.1038/s41567-019-0459-y. URL <https://doi.org/10.1038/s41567-019-0459-y>.

- Matthieu Latapy. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theoretical Computer Science*, 407:458–473, 2008.
- Silvio Lattanzi and D Sivakumar. Affiliation networks. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 427–434. ACM, 2009.
- Sang Hoon Lee, Pan-Jun Kim, and Hawoong Jeong. Statistical properties of sampled networks. *Physical Review E*, 73(1):016102, 2006.
- Richard B Lehoucq and Danny C Sorensen. Deflation techniques for an implicitly restarted arnoldi iteration. *SIAM J. Matrix Anal. Appl.*, 17(4):789–821, 1996. ISSN 0895-4798. doi: 10.1137/S0895479895281484.
- Jure Leskovec. The Stanford large network dataset collection. <http://snap.stanford.edu/data/index.html>, 2016.
- Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2006.
- Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 177–187, New York, NY, USA, 2005a. ACM. ISBN 1-59593-135-X. doi: 10.1145/1081870.1081893.
- Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1), Marsh 2007. URL <http://dl.acm.org/citation.cfm?id=1217301>.
- Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, September 2009. doi: 10.1080/15427951.2009.10129177.
- Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research*, 11:985–1042, February 2010. URL <http://www.jmlr.org/papers/volume11/leskovec10a/leskovec10a.pdf>.
- Jurij Leskovec, Deepayan Chakrabarti, Jon Kleinberg, and Christos Faloutsos. Realistic, mathematically tractable graph generation and evolution, using Kronecker multiplication. In *PKDD*, volume 3721 of *Lecture Notes in Computer Science*, pages 133–145. 2005b. doi: 10.1007/11564126_17.
- László Lovász et al. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty*, 2(1): 1–46, 1993.
- David Lusseau, Karsten Schneider, Oliver J. Boisseau, Patti Haase, Elisabeth Slooten, and Steve M. Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations: Can geographic isolation explain this unique trait? *Behavioral Ecology and Sociobiology*, 54(4):pp. 396–405, 2003. ISSN 03405443. URL <http://www.jstor.org/stable/25063281>.
- Michael W Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.
- Alberto Medina, Ibrahim Matta, and John Byers. On the origin of power laws in internet topologies. *ACM SIGCOMM Computer Communication Review*, 30(2), 2000.

Robert Meusel, Sebastiano Vigna, Oliver Lehmborg, and Christian Bizer. The graph structure in the web - analyzed on different aggregation levels. *The Journal of Web Science*, (1):33–47, 2015.

Milena Mihail and Christos Papadimitriou. On the eigenvalue power law. In *RANDOM '02 Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, pages 254–262, 2002.

Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002. ISSN 0036-8075. doi: 10.1126/science.298.5594.824.

Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet Measurement*, ICM '07, pages 29–42, New York, NY, October 2007. ACM.

Sebastian Moreno and Jennifer Neville. Network hypothesis testing using mixed kronecker product graph models. In *2013 IEEE 13th International Conference on Data Mining*, pages 1163–1168. IEEE, 2013.

Sebastian Moreno, Sergey Kirshner, Jennifer Neville, and SVN Vishwanathan. Tied kronecker product graph models to capture variance in network populations. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1137–1144, Sept 2010. doi: 10.1109/ALLERTON.2010.5707038.

Sebastian Moreno, Joseph J. Pfeffer, Jennifer Neville, and Sergey Kirshner. A scalable method for exact sampling from kronecker family models. *IEEE International Conference on Data Mining (ICDM)*, 2014.

Richard C Murphy, Kyle B Wheeler, Brian W Barrett, and James A Ang. Introducing the graph 500. Cray User’s Group, May 2010.

Tamás Nepusz. plfit software. <https://github.com/ntamas/plfit>, 2016.

M. E. J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001. doi: 10.1073/pnas.98.2.404.

Mark Newman. Network datasets. <http://www-personal.umich.edu/~mejn/netdata/>, 2006.

Mark EJ Newman. Random graphs with clustering. *Physical review letters*, 103(5):058701, 2009.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Liudmila Ostroumova, Alexander Ryabchenko, and Egor Samosvat. Generalized preferential attachment: tunable power-law degree distribution and clustering coefficient. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 185–202. Springer, 2013.

Evangelos Papalexakis, Bryan Hooi, Konstantinos Pelechrinis, and Christos Faloutsos. Power-hop: A pervasive observation for real complex networks. *PLoS ONE*, 11(3), 2016.

Anh Huy Phan, Andrzej Cichocki, Petr Tichavský, Danilo P. Mandic, and Kiyotoshi Matsuoaka. On revealing replicating structures in multiway data: A novel tensor decomposition approach. In *Latent Variable Analysis and Signal Separation*, pages 297–305, 2012. ISBN 978-3-642-28551-6. doi: 10.1007/978-3-642-28551-6_37.

Derek De Solla Price. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American Society for Information Science*, 27(5):292–306, 1976. doi: 10.1002/asi.4630270505.

Arjun S. Ramani, Nicole Eikmeier, and David F. Gleich. Coin-flipping, ball-dropping, and grass-hopping for generating random graphs from matrices of probabilities. *arXiv*, cs.SI:1709.03438, 2017. URL <https://arxiv.org/abs/1709.03438>.

Martin Rosvall et al. Memory in network flows and its effects on spreading dynamics and community detection. *Nat. Comm.*, 5(4630), 2014. doi: 10.1038/ncomms5630.

Alessandra Sala, Lili Cao, Christo Wilson, Robert Zablit, Haitao Zheng, and Ben Y. Zhao. Measurement-calibrated graph models for social network experiments. In *Proceedings of the 19th International Conference on World Wide Web, WWW '10*, pages 861–870, New York, NY, USA, 2010a. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772778.

Alessandra Sala, Haitao Zheng, Ben Y. Zhao, Sabrina Gaito, and Gian Paolo Rossi. Brief announcement: Revisiting the power-law degree distribution for social graph analysis. In *Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC '10*, pages 400–401. ACM, 2010b. ISBN 978-1-60558-888-9. doi: 10.1145/1835698.1835791.

Grant Schoenebeck. Potential networks, contagious communities, and understanding social network structure. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1123–1132. ACM, 2013.

C. Seshadhri, Ali Pinar, and Tamara G. Kolda. An in-depth analysis of stochastic kronecker graphs. *J. ACM*, 60(2):13:1–13:32, May 2013. ISSN 0004-5411. doi: 10.1145/2450142.2450149.

Rohit Singh, Jinbo Xu, and Bonnie Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *PNAS*, 105(35):12763–12768, 2008. doi: 10.1073/pnas.0806627105.

Michael PH Stumpf and Carsten Wiuf. Sampling properties of random graphs: the degree distribution. *Physical Review E*, 72(3):036118, 2005.

Michael PH Stumpf, Carsten Wiuf, and Robert M May. Subnets of scale-free networks are not scale-free: sampling properties of networks. *Proceedings of the National Academy of Sciences*, 102(12):4221–4224, 2005.

CAIDA (The Cooperative Association for Internet Data Analysis). Network datasets. http://www.caida.org/tools/measurement/skitter/router_topology/, 2005. Accessed in 2005.

Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. Social structure of facebook networks. *Physica A*, 391(16):4165–4180, 2012. ISSN 0378-4371. doi: 10.1016/j.physa.2011.12.021.

Remco van der Hofstad. *Random graphs and complex networks: Volume 1. Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge university press, 2016.

Christof Vömel. Scalapack’s MRRR algorithm. *ACM Trans. Math. Softw.*, 37(1):1:1–1:35, January 2010. ISSN 0098-3500. doi: 10.1145/1644001.1644002.

Duncan J. Watts. *Six Degrees: The Science of a Connected Age*. W. W. Norton and Company, 500 Fifth Avenue, New York, N.Y. 10110, 2003. URL <http://books.wwnorton.com/books/Six-Degrees/>.

Duncan J Watts and Steven H Strogatz. Collective dynamics of “small-world” networks. *Nature*, 393(6684):440, 1998.

J. Xu, T. L. Wickramaratne, and N. V. Chawla. Representing higher-order dependencies in networks. *Science Advances*, 2(5):e1600028–e1600028, may 2016. doi: 10.1126/sciadv.1600028.

Hao Yin, Austin R Benson, and Jure Leskovec. Higher-order clustering in networks. *Physical Review E*, 97(5):052306, 2018.

G. Udny Yule. II.—A mathematical theory of evolution, based on the conclusions of Dr. J. C. Willis, F. R. S. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 213 (402-410):21-87, 1925. ISSN 0264-3960. doi: 10.1098/rstb.1925.0002.