1-1-2013

Spectral Analysis of Randomly Generated Networks With Prescribed Degree Sequences

Clifford Davis Gaddy
University of South Carolina

Spectral Analysis of Randomly Generated Networks with Prescribed Degree Sequences

by

Clifford Davis Gaddy

Bachelor of Arts
Washington and Lee University 2010

Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Arts in

Mathematics

College of Arts and Sciences

University of South Carolina

2013

Accepted by:

Éva Czabarka, Major Professor

Linyuan Lu, Second Reader

Lacy Ford, Vice Provost and Dean of Graduate Studies

Acknowledgments

First of all, I want to say thank you to my advisor Dr. Éva Czabarka. From my first semester here, she has been extremely helpful and eager to discuss any topic I was curious about. She has given me much personal, career, and mathematical advice during the last two years. Thank you for leading me through this thesis and the enormous amount of time that you have invested. You have made this whole process and my time at the University of South Carolina a very enjoyable experience. Thank you to Dr. Linyuan Lu for being the second reader of my thesis. I enjoyed taking *Probabilistic Methods* with you in the fall. It was an extremely interesting course and during this whole year, you have been very willing to discuss anything with me. I thank you for that. Thank you to Dr. Zoltán Toroczkai for introducing me to this project and giving me the network data that was fundamental to this project. Thanks to Dr. Aaron Duttle for the discussions that we have had this year regarding graphs and Markov chains. Also thanks for introducing me to the program Geogebra that allows one to draw graphs. Thanks to Dr. László Székely for helpful discussions throughout the thesis process, especially with regards to enumeration of graphs with a given degree sequence and Markov chains. I also enjoyed your Discrete mathematics class this year. Thank you to Dr. Richard Anstee, an extremely friendly person full of advice. Thanks to Dr. George Mcnulty and Dr. Francisco Blanca-Silva for help with Latex. Thanks to the technical staff of the mathematics department for giving me extra hard drive space to store all my data. Lastly, thanks to all of my friends and family for the loving support during the last two years: my Mom and Dad, Janie, Jacob, Ira, Jocy, my office mates John and Dan, my tennis mates

Toby, Pat and Steph; and my two best friends Jordan and Asel, who have supported me all along. And lastly, thanks to the other graduate students for long hours spent struggling through problems together.

Abstract

Network science attempts to capture real-world phenomenon through mathematical models. The underlying model of a network relies on a mathematical structure called a graph. Having seen its early beginnings in the 1950's, the field has seen a surge of interest over the last two decades, attracting interest from a range of scientists including computer scientists, sociologists, biologists, physicists, and mathematicians. The field requires a delicate interplay between real-world modeling and theory, as it must develop accurate probabilistic models and then study these models from a mathematical perspective. In my thesis, we undertake a project involving computer programming in which we generate random network samples with fixed degree sequences and then record properties of these samples. We begin with a real-world network, from which we extract a sample of at least one-hundred vertices through the use of snowball sampling. We record the degree sequence, D, of this sample and then generate random models with this same degree sequence. To generate these models, we use a well-known graph algorithm, the Havel-Hakimi algorithm, to produce an initial non-random sample G_D. We then run a Monte Carlo Markov Chain (MCMC) on the sample space of graphs with this degree sequence beginning at G_D in order to produce a random graph in this space. Denote this random graph by H_D. Lastly, we compute the eigenvalues of the Laplacian matrix of H_D, as these eigenvalues are intricately connected with the structure of the graph. In doing this, we intend to capture local properties of the network captured by the degree sequence alone. The programming of this project is done in Python and Matlab.

Table of Contents

List of Tables

List of Figures

Chapter 1

Background

1.1 Introduction

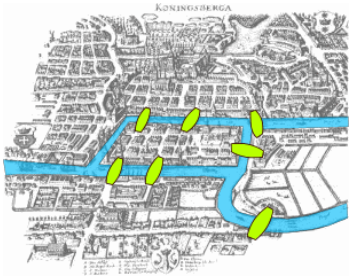
The field of complex networks is an exciting area of current study. Having been a topic of interest since the middle of the last century, the field has very recently seen a surge in interest from a broad range of disciplines. To help solve important problems, the field has attracted attention from biologists, physicists, sociologists, computer scientists, and mathematicians [5]. Thus its development has seen a delicate interplay between theory and applications. Its beauty lies in its ability to model and understand real-world phenomenon as well as demonstrate deep theoretical results.

At first glance, a network ultimately is an abstract representation of a set of discrete objects and connections between these objects. In mathematical terms, a network is a graph and thus the objects are referred to as vertices, and the connections between them are referred to as edges. A network abstracts away from properties of the underlying system it represents, and captures only topological and connectivity properties of the system. One of the goals of network science is to understand and explain properties of a system by the structural connectedness properties of its network representation. Many scientific phenomenon have very natural representations as networks. The internet and world wide web are two of the most studied and prominent examples of networks. The world wide web is an example of a "directed" network. Each webpage represents a vertex. Then if page u has a hyper-link to another page v, we say that there is a directed edge from u to v . Note that this does not necessarily

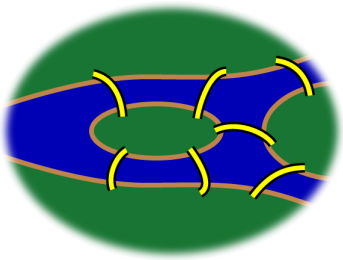
mean there is an edge from v to u. Thus the edge is "directed". Many networks do not distinguish direction in their edges, and thus, in these cases, two vertices sharing an edge is not an ordered relationship. We would call this a "simple" network. Other examples of networks include protein interaction networks, metabolic networks, and social networks. In a social network, it could be the case that vertices are people and edges represent relations among the people such as friendship or familial ties. See [3] for many more examples.

One could claim that graph theory and network science have their origins in the same story. In 1736, the mathematician Leonard Euler attempted to solve a question involving bridges in the old city of Königsberg, Prussia. The city lie on the banks of the Pregel River, and consisted of two islands in the middle of the river. Seven bridges connected the islands with each other and both sides of the river. The problem was to determine if there was a path along the network that crosses each bridge exactly once. The is known as the Königsberg Bridge Problem. This is often cited as the first problem of graph theory. By abstracting the network of different landmasses and bridges connecting them to just represent vertices and edges, respectively, the problem then turns into a graph theoretical question. It turns out that the graph representing this network does not permit an Eulerian path, which is the correct mathematical concept to describe the desired path that crosses each bridge exactly once. Thus by considering this fact, one attains a negative answer to the question. [5] Figure 1.1 shows the city [13], the network of bridges [31], and graph representation of this network [32].

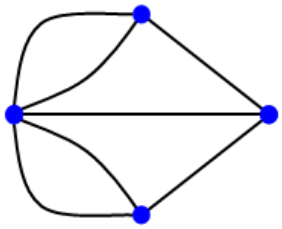
Since this time, both fields have been the subject of intense theoretical and practical study. Barabási et al. present a detailed summary of the field in the book *The Structure and Dynamics of Networks* [5]. In this introduction, I present a brief history of the field along with a discussion of major theoretical developments. Most of my introductory discussion is inspired by this book, and thus the interested reader is



City of Königsberg [13]



Bridges of Königsberg [31]



Graph representation of bridges of Königsberg [32]

Figure 1.1 Representation of real life problem as a network

invited to consult the book for further details.

In the 1950's, social scientists saw a need to quantify the methods of their sciences, such as sociology and anthropology. Thus many adopted terms and concepts from graph theory to help describe problems they faced. They used these ideas to help create models as well as analyze collected empirical data, linking graph theoretical ideas with social ideas such as "status, influence, cohesiveness, social roles, and identities in social networks," [5] page 3. Around the same time, graphs became an accepted model for means of disease and information transmission. Furthermore, it was around this time that the notion of random graphs saw its early developments. This new approach saw graphs as stochastic, or probabilistic, objects rather than deterministic. All these events thus produced a surge of interest in the new field of network science. [5] chapter 1. Before any further discussion of the historical development of network science, it is important to introduce several basic concepts from graph theory. For a complete introduction to graph theory, see e.g. Diestel's book *Graph Theory* [9].

1.2 Graph Theory

Notation 1.1. Let $n \in \mathbb{N}$. $[n] = \{z \in \mathbb{Z} : 1 \leq z \leq n\}$

Notation 1.2. If V is a set and k is a positive integer, then $[V]^k$ is the set of all k-element subsets of V.

Notation 1.3. If V is a set and k is a positive integer, then V^k is the set of all k-tuples of elements of V.

Definition 1.4. A *simple, non-directed graph* G is a pair of sets (V, E) where V is a set of objects which we call vertices and $E \subseteq [V]^2$, elements of which are referred to as edges.

Note that we do not allow for more than one edge between any two vertices.

Definition 1.5. A *directed graph* is a graph $G = (V, A)$, where $A \subseteq V^2$.

For this work, we always assume that V is of finite cardinality. If $|V| = n$, we can create a bijection between $[n]$ and V and refer to the vertices as v_i, where $i \in [n]$. We assume that a graph is non-directed and simple, unless otherwise stated.

Now we introduce several definitions associated with a graph. In the following, assume $G = (V, E)$ is a graph with n vertices.

Definition 1.6. A vertex $v \in V$ is said to be *incident* upon an edge $e \in E$ if $e = \{v, w\}$ for some $w \in V$. In words, v is incident upon e if v is an end-vertex of e. Conversely, e is also said to be incident upon v.

Definition 1.7. A vertex $v \in V$ is said to be *adjacent* to a vertex $w \in V$ if they share an edge, that is, if $\{v, w\} \in E$.

Definition 1.8. Two vertices are said to be *neighbors* if they are adjacent.

Definition 1.9. Given a vertex $v \in V$, its *neighborhood* is the set of its neighbors. We denote it by $N(v) = \{y \in V : \{y, v\} \in E\}$.

Definition 1.10. Given a vertex v, its *degree*, denoted by $d(v)$, is the number of edges that it is incident with, or equivalently the number of vertices it is adjacent with.

Definition 1.11. Two edges e and f are said to be *independent* if they do not share an end-vertex.

Definition 1.12. The *order* of G refers to the number of vertices in G. It is denoted by $|G| := |V|$.

Definition 1.13. The *size* of G refers to the number of edges in G. It is denoted by $||G|| := |E|$.

Definition 1.14. A *path* in G is a sequence of vertices $v_0, v_1, \ldots, v_m$ where $m \geq 0$ such that $\{\{v_i, v_{i+1}\} : 0 \leq i \leq m-1\}$ is a set of m different edges of G, and no vertices are repeated with the exception that v_0 could equal v_m. Denote such a path as a $v_0 - v_m$ path.

Definition 1.15. A *cycle* is a path $P = v_0, v_1, \ldots, v_m$ in which $v_0 = v_m$ and $m > 0$.

Note that it follows that in a cycle, $m \geq 3$.

Definition 1.16. The *length* of a path is the number of edges in the path.

Definition 1.17. The *girth* of G is the length of the shortest cycle in G.

Definition 1.18. Given two vertices, u and v, their *distance* is the length of the shortest $u - v$ path in G.

Definition 1.19. A subset $U \subseteq V$ of vertices is connected if for every pair of vertices $v, w \in U$, there exists a $v - w$ path in G.

Definition 1.20. A graph G is *connected* if and only if the entire vertex set V is connected.

Definition 1.21. A maximally connected set of vertices of a graph is called a *component*. Thus a connected graph has one component.

Definition 1.22. A *tree* is a connected graph with no cycles.

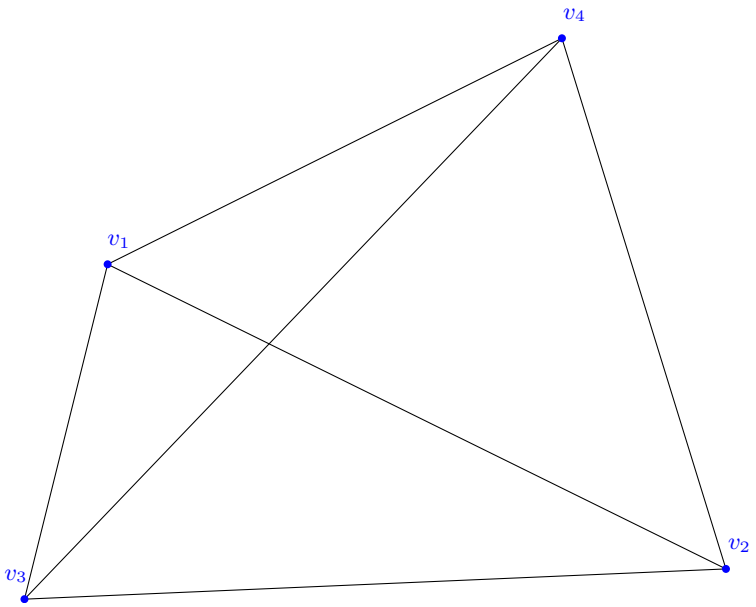


Figure 1.2 Example Graph

Definition 1.23. A *dominating set* D is a subset of V such that every vertex not in D has at least one neighbor in D. A dominating component is a connected dominating set.

Definition 1.24. Two graphs $G = (V_1, E_1)$ and $H = (V_2, E_2)$ are *isomorphic* if there exists a bijection $\phi : V_1 \to V_2$ such that $\{u, w\} \in E_1$ if and only if $\{\phi(u), \phi(v)\} \in E_2$.

Definition 1.25. If Ω is a set, then $\mathcal{P}(\Omega)$ is the set of all subsets of Ω, called its *power set*.

Definition 1.26. : Given a graph G on vertex set $\{v_1, \ldots, v_n\}$, there exists an associated *degree sequence* D. D is the list of non-negative integers $(d_1, d_2, \ldots, d_n)$ such that $d(v_i) = d_i$.

Figure 1.2 is an example of a graph that has degree sequence $(3, 3, 3, 3)$.

As the order and size of a graph becomes very large, it is often computationally impractical to obtain exact measurements of graph parameters, such as the diameter, girth, connectivity, etc. To mitigate this difficulty, it helps to look at the space of

all graphs on a fixed number of vertices, n, as a probability space. Letting n go to infinity, one can often obtain average measurements of these parameters in terms of n. One can obtain a probability measure of the set of graphs on n vertices having properties of interest. Thus a random graph in this space will have these properties of interest with this associated probability. Thus it is important to understand some basic notions from probability.

Modern probability relies on the mathematical axioms, definitions, and results of set theory and mathematical analysis, specifically measure theory. By rigorously defining probability spaces, one is able to use powerful tools from analysis. To begin with, a probability space is a special kind of mathematical object called a measure space. Several background definitions are needed. Many of the following definitions are taken from Resnick's book *A Probability Path*. See this for further reference [27].

1.3 Probability

Definition 1.27. An *experiment* is a hypothetical or realistic scenario in which a range of outcomes are possible.

Examples include rolling a six-sided dice or flipping a two-sided coin.

Definition 1.28. The set of outcomes that can occur in an experiment is referred to as the *sample space*. A sample space is typically denoted by Ω. Flipping a coin, the sample space could be $\{0, 1\}$ where 0 represents a tails and 1 represents a heads.

Definition 1.29. Given an experiment, we can associate probabilities to each outcome in the sample space. This is called a *probability distribution*.

Definition 1.30. If Ω is a an arbitrary set, then a set, $\mathcal{A}$, of subsets of Ω is a *σ-algebra* on Ω provided that the following conditions hold:

(i) $\Omega \in \mathcal{A}$;

(ii) If $\{C_i, i \in \mathbb{N}\}$ is a countable collection of elements of $\mathcal{A}$, then $\cup_{i\in\mathbb{N}} C_i \in \mathcal{A}$;

(iii) If $C \in \mathcal{A}$, then its complement $C' = \Omega - C \in \mathcal{A}$.

Condition (ii) is referred to as being closed under countable unions. It quickly follows from these axioms that $\emptyset \in \Omega$, and $\mathcal{A}$ is closed under countable intersections.

$\mathcal{P}(\Omega)$ is a simple example of a σ-algebra on Ω.

Definition 1.31. : A *measurable space* is a pair $(\Omega, \mathcal{A})$ where Ω is a set and $\mathcal{A}$ is a σ-algebra over Ω.

Definition 1.32. Let $\mathcal{A}$ be a σ-algebra on a set Ω. A *measure* on $\mathcal{A}$ is a function $\mu : \mathcal{A} \to \mathbb{R} \cup \{+\infty\}$ which obeys the following axioms:

(i) Empty set: $\mu(\emptyset) = 0$;

(ii) Non-negativity: $\mu(E) \geq 0$ for all $E \in \Omega$;

(iii) Countable additivity: For a countable collection $\{C_i\}_{i \in \mathbb{N}}$ of pairwise disjoint elements in Ω, $\mu(\cup_{i=1}^{\infty} C_i) = \sum_{i=1}^{\infty} \mu(C_i)$.

Definition 1.33. A *measure space* is a triple $(\Omega, \mathcal{A}, \mu)$ where Ω is a set, $\mathcal{A}$ is a σ-algebra of subsets of Ω, and μ is a measure on $\mathcal{A}$.

Definition 1.34. A *probability space* is a measure space $(\Omega, \mathcal{A}, P)$ where $P(\Omega) = 1$. A probability distribution can be thought of as a probability measure on the sample space.

Definition 1.35. : Given a probability space $(\Omega, \mathcal{A}, P)$, each element of $\mathcal{A}$ is called an *event*.

So if Ω is the set of all possible outcomes of an experiment, an event $A \in \mathcal{A}$ is a subset of outcomes. We say that the event A occurs if the outcome of the experiment is an element of A.

Definition 1.36. : Given a sample space Ω, a *random variable* is a measurable function $X : \Omega \to \mathbb{R}$. A random variable is often used to represent an experiment.

For example, suppose that a two-sided coin is tossed n times. Using the same encoding as before, let the n-tuple $(o_1, o_2, \ldots, o_n)$ represent a possible outcome of the n tosses where $o_i \in \{0, 1\}$ for $1 \leq i \leq n$. Thus Ω consists of all 2^n such $n-$tuples. Then the variable $X : \Omega \to \{0, \ldots, n\}$ is a random variable where X outputs the number of heads of a given outcome. Given a random variable X and a probability measure P on Ω, there is an associated probability distribution (measure), D, on $\mathbb{R}$. That is, for $y \in \mathbb{R}$, $D(y) = P(X^{-1}(y))$. In the example above, supposing each outcome has an equal probability of $\frac{1}{2^n}$, $D(k) = \frac{\binom{n}{k}}{2^n}$ for $0 \leq k \leq n$. Several common distributions arise frequently and thus are given names. When necessary, we will define such a distribution.

Definition 1.37. Given a random variable X on a probability space $(\Omega, \mathcal{A}, P)$, the *expected value* of X is defined formally as the Lebesgue integral $E[X] = \int_\Omega X dP$. In the discrete case, this turns out to be $\sum_i x_i p_i$ where x_i ranges over all possible values of X and X takes the value of x_i with probability p_i.

The expected value of a random variable can be viewed as its average value. Here we talk about the notion of dependence between events in a probability space, as this is important in understanding Markov Chains, to be discussed later.

Definition 1.38. If E, F are events in some sample space Ω with $P(F) \neq 0$, we define the *conditional probability* of the event E occurring given that F occurs as $P(E|F) = \frac{P(E \cap F)}{P(F)}$. This can be generalized to an arbitrary number of events. If $A, E_1, E_2, \ldots, E_n$ are events with $P(\cap_{i=1}^n E_i) \neq 0$, then $P(A|E_1, E_2, \ldots, E_n) = \frac{P(A \cap (\cap_{i=1}^n E_i))}{P(\cap_{i=1}^n E_i)}$ is the probability of A given the occurrence of the events $E_1, \ldots, E_n$.

Definition 1.39. : Two events E, F in a probability space are said to be *independent* if $P(E \cap F) = P(E) \cdot P(F)$. If $P(F) \neq 0$, this is equivalent with $P(E|F) = P(E)$.

Independence is important in understanding random graph processes. As said before, with the types of probability spaces one is typically interested in in combina-

torics and graph theory, the asymptotic limiting behavior of a random process is of most importance. Also, one often speaks of an event occurring with high probability. We will define this formally.

Definition 1.40. Given some property A and a probability space $(\Omega, \mathcal{A}, P)$, it is said that A holds *almost surely* if there exists a set $N \in \mathcal{A}$ such that $P(N) = 0$ and for $w \in N'$, A holds. Thus A will hold always, except for on a set of probability zero.

Definition 1.41. Given an arbitrary function real-valuled f, $\lim_{n\to\infty} f(n) = L$ means that given an arbitrary $\epsilon > 0$, there exists $N \in \mathbb{N}$ such that for all $n \geq N$, $|f(n)-L| < \epsilon$. L is the *asymptotic limit* of f.

Sometimes, an exact limit of a function is not known. But it is known that its limiting behavior is similar to that of a simpler function which is better understood. Thus a function is spoken about with reference to its asymptotic order. Let $f(n), g(n)$ be two non-negative functions. The following notation is used:

$f(n) \in O(g(n))$ means that there exists n_0 such that for all $n \geq n_0, f(n) \leq kg(n)$ for some positive constant k.

$f(n) \in \Omega(g(n))$ means that there exists n_0 such that for all $n \geq n_0, f(n) \geq kg(n)$ for some positive constant k.

$f(n) \in \Theta(g(n))$ means that there exists n_0 such that for all $n \geq n_0, k_2 g(n) \leq f(n) \leq k_1 g(n)$ for some positive constants k_1, k_2.

$f(n) \in o(g(n))$ means that for every $\epsilon > 0$, there exists n_0 such that for all $n \geq n_0, f(n) \leq \epsilon g(n)$.

$f(n) \sim g(n)$ means that $f(n) - g(n) \in o(g(n))$.

These definitions can be found in [2].

1.4 History

Here we will discuss several developments historically important to the field, developments coming from biologists, sociologists, and mathematicians. Graph theorists are familiar with the Erdős-Rényi random graph model, introduced by Paul Erdős and Alfréd Rényi in the 1960 paper, "On The Evolution of Random Graphs" [11]. It is interesting to note, though, that several of the basic notions were actually introduced a decade earlier by the biologists Anatol Rapoport and Ray Solomonoff in their 1951 paper, "Connectivity of Random Nets" [29]. Rapoport, one of the earliest mathematical biologists, was interested in statistical aspects of networks. This was a new notion at the time, looking at averages over random models rather than individual graphs. In their paper, they considered a random graph model. They studied the component structure of networks, and predicted a deep theoretical result proven by Erdős-Rényi, that is the phase transition of the component structure. Their model is essentially the same as the Erdős-Rényi model in that it considers a set of vertices with edges randomly placed between them. For a randomly chosen vertex, they looked at its expected component size, referred to by them as "weak connectivity". They concluded that the weak connectivity is dependent on the mean degree a of the vertices. For $a < 1$, the network has many small components, but for $a > 1$, a giant, dominating component arises. This is essentially the phase transition of the component structure proven rigorously by Erdős-Rényi. The authors suggest several problems for which their model might be useful. These problems involve neural networks, epidemiology, and genetics [5], page 11-12.

Erdős and Rényi were not aware of the contributions by Solomonoff and Rapoport and treated the subject independently of them. Nonetheless, the Erdős-Rényi treatment was mathematically thorough and delved deeper into the subject than their predecessors. They produced many papers and results involving random graphs over the next decade, and this work serves as the foundation for much future research in

network science as well as for the purely mathematical topic of probabilistic combinatorics, or rather the so-called "probabilistic method". See Alon and Spencer's book *The Probabilistic Method* for an exhaustive treatment of this subject [2]. This random graph model is one of the points where network science shares an unclear boundary with deep theoretical, mathematical results. We will discuss particulars of this model later when we also introduce several other network models [5], page 12.

One of the most influential sociological papers was put forth in 1978 by the political scientist Ithiel de Sola Pool and the mathematician Manfred Kochen in their paper "Contacts and Influence" [8]. The paper concerns social network structure and patterns. Written in 1958, it was not published until twenty years later because the authors were not satisfied with the depth of their work. A manuscript was produced though, and the paper was un-officially circulated amongst the relevant research community. The paper put forth many of the crucial issues that came to define the field for some time after. In this network, they treat people as vertices and acquaintances between two people as an edge. Issues addressed involve an individual person's degree, degree distributions, average and range of degrees, high-degree people, network structure, probability of a random acquaintance, probability of shared acquaintances or neighbors, and shortest path lengths. Due to the insufficiency of empirical data, they used the random graph model. They were also the first to scientifically introduce the notion of the small-world effect, popularized as "6 degrees of separation", which refers to the idea that everybody in the world is within a relatively short social distance of everybody else [5], page 15.

The scientific community itself also gives rise to several networks. One of these was first studied by Derek de Solla Price in his 1965 paper "Networks of Scientific Papers" [24]. In this directed network, each paper is a vertex, and a citation of another paper is represented by a directed edge from the first to the cited. Each paper has an in-degree and an out-degree. He studied statistical properties of these two degrees,

namely the distributions of them and discovered that they have power-law tails. This phenomenon has later been observed in many naturally occurring networks, and a network with such a degree distribution is referred to as a "scale-free network." In 1976, he published another paper which proposed an explanation for the existence of power-laws in certain networks. This explanation has been widely accepted and adopted and is known today as "preferential attachment" [25]. The explanation claims that "citations receive further citations in proportion to the number they already have" [5], page 18. Thus vertices with a large degree are more likely to accumulate more neighbors. Preferential attachment and scale-free networks are notions of much interest in network science today [5], page 17-18.

As for the current state of the field, networks have seen a very recent surge in scientific interest over the last two decades. The most influential reason for this is probably the advent of the use of computers, the internet, and the world wide web in everyday society. With this surge, the field has also undergone some changes in the way research is conducted. Barabási highlights three major points in this change in [5]. He claims that there is more emphasis on empirical questions than before; networks are seen as evolving rather than static; and the dynamical structure of the underlying system on which the network lies is being taken into consideration more, thus viewing the network no longer as a purely topological object [5], chapter 1.2. There is now a concerted effort to find accurate models of the empirical networks at hand. One reason this has not been a practical concern up until now is the inability to sufficiently collect meaningful data. Obviously, we need the computer to facilitate this [5], page 4-5. Networks evolve in the real sense that a person's number of acquaintances might change over time in a social network, the number of citations of a paper can change in a citation network, and a web-page can add or delete hyper-links with time. Thus models capturing time-evolving networks rather than static models are necessary [5], page 7. When considering dynamical structures

of the system underlying a network, it is important to take real properties of the network into account. For example, in epidemiology, it should not be assumed that all people have equal interaction probabilities. Social rules will dynamically affect the probability of two specific vertices interacting. Corresponding models need to account for that [5], page 8.

It is important to look at the theoretical models that have been developed and used over time. Given real networks with certain properties of interest, we want models that accurately capture these properties. To find such models and then analyze them from a theoretical standpoint is obviously one of the tasks of network science. We discuss theoretical properties of several models and highlight where they have fallen short. [5] highlights three significant models: the Erdős-Rényi random graph, the small-world model, and the scale-free model.

The Erdős-Rényi random graph model takes two different forms: $G(n,p)$ and $G(n,m)$. $G(n,m)$ refers to the space of all graphs with n vertices and m edges, equipped with uniform probability. For $0 \leq p \leq 1$, $G(n,p)$ refers to the space of all graphs on $[n]$, in which for $i,j \in [n]$, $\Pr[\{i,j\} \in E(G)] = p$. For any two possible edges e, f, the events that $e \in E(G)$ and $f \in E(G)$ are independent. We will focus on $G(n,p)$. In this space, each graph has a certain probability which is a function of n, p, and its size m: $p^m(1-p)^{\binom{n}{2}-m}$. One is concerned with the behavior of the space asymptotically, as $n \to \infty$. p is typically treated as a function of n, denoted $p(n)$, and in this sense we can define threshold functions. Given some graph property A, one wishes to determine whether there is a threshold function $r(n)$, such that when $r(n) \in o(p(n))$, the probability that $G(n,p)$ satisfies A goes to 1 and when $p(n) \in o(r(n))$, this probability goes to 0. Thus when we say that a certain property holds in $G(n,p)$ for certain p, we are saying that the property holds almost surely as n goes to infinity. The most noted result regarding this model is the phase transition its component structure undergoes as p changes. Let $\epsilon > 0$ be a small constant. The

evolution of $G(n,p)$ goes as follows: (1) when $p < \frac{1-\epsilon}{n}$, all components are trees or unicyclic, having one cycle, and of size $\leq o(\ln(n))$; (2) when $p > \frac{1+\epsilon}{n}$, there exists a giant component of size $\Theta(n)$, and all other components are of size $o(\ln(n))$; (3) when $p = \frac{1+\lambda n^{\frac{1}{3}}}{n}$ for λ a real constant, the largest component has size $\Theta(n^{\frac{2}{3}})$; (4) $p \sim (1+o(1))\frac{\ln(n)}{n}$ is the threshold for $G(n,p)$ being connected [2].

Also of note in this model is that it can be proven that its underlying probabilistic structure only allows for the existence of one large component. This, as well as the predicted component framework, is reflected in many real-world networks [5], page 230. The problem with this model though is that it does not accurately reflect degree distributions. Given a random vertex v, the probability that it has degree k is given by $p_k = \binom{n-1}{k} p^k (1-p)^{n-k-1}$, i.e. the binomial distribution. This is because for each choice of k neighbors from the remaining $n-1$ vertices, there is a probability of p^k that all k of them are neighbors and a $(1-p)^{n-k-1}$ probability that the remaining $n-k-1$ vertices are non-neighbors. Note that the expected value of the degree is $(n-1)p$. If a model assumes that the average (or expected) value of the degree is constant, i.e. $p(n) = \frac{c}{n-1}$, then as $n \to \infty, p_k \to \frac{(pn)^k e^{-pn}}{k!}$. Thus the degree distribution of any vertex v approaches the Poisson distribution as $n \to \infty$. [5] claims that this distribution is not the degree distribution seen in many empirical networks. This is an important issue, since the degree distribution significantly affects the network structure [5], page 233. We discuss random graphs with given degree distributions or sequences at a further point, as this is directly related to our project.

The small-world model attempts to capture two properties that appear in many real-world networks. The first of these is the condition that the distance between two vertices in a network is relatively short. Specifically, the average distance is a logarithmic function of the order of the network and thus is growing slowly with the number of vertices in the network. Secondly, the network experiences high clustering, which means that the probability of two vertices being adjacent increases if they also

share a neighbor. A noteworthy fact about this model is that it has some connections to statistical physics [5], page 286. See Watts and Strogatz for the first appearance of and further details to this model [33].

The last model of discussion, the scale-free network, has only recently been of intense interest. A scale-free network is characterized by the fact that its degree distribution follows a power law. A power law is a function which outputs a power of the input value. For example, $F(k) = k^{-2}$ is a power law function. There has been recent interest in them, because it has been shown that many real-world networks follow power laws. The random graph and small-world models do not follow power laws, in their general form, even though the random graph model can be modified to follow any desired distribution [1]. Thus the need for models which follow power laws along with their mathematical analysis has arisen. This problem has attracted attention from some of the most well-known network scientists and mathematicians. The celebrated first paper on scale-free networks was by Barabási and Albert in 1999, titled "Emergence of Scaling in Random Networks" [4]. The paper put forth several ideas. First, they argued that power-law distributions are common among many networks, evidencing this with the World Wide Web, a network of film actors [33], and citation networks [26]. Next, they put forth a model with allows for growth and claim that models of this kind help explain the existence of power-laws [5], page 335. The explaining properties of such a model are that they grow, continuously adding new vertices, and that a vertex gains new edges at a rate proportional to its current degree. This second property is known as preferential attachment in the literature. In the real world, it makes sense that the aforementioned networks follow these properties. The model put forth by Barabási and Albert treats the network as a discrete growth process, in which at each time step a new vertex is added along with m edges incident with this vertex and already existent vertices randomly chosen proportionally to their current degrees. This model has degree distribution $P(k) \approx k^{-3}$. Variations of the

preferential attachment model have been developed since, and give rise to exponents in the power law varying from 2 to ∞ depending on the parameters of the model, thus allowing for a good deal of flexibility [5], page 335-337. Barabási and Albert's analytical treatment of their model falls short from a mathematical perspective. Thus Bollobás et al. took on the role of performing such an analysis [6]. They gave a mathematical confirmation of the results of Barabási and Albert, rigorously defining the model and proving the main claim of [4]. They later went on to derive new results as well, such as the calculation of average path lengths between all pairs of vertices in Barabási-Albert's model. This is an interesting example in which phenomenon were predicted heuristically to later be proven mathematically. Likewise, Aiello, Chung, and Lu [1] devoted attention to random scale-free networks, thus signifying an effort from the mathematical community to develop theory pertaining to real-world networks [5], page 345-346.

Chapter 2

Project

2.1 Degree Sequence

Any graph on vertex set $\{v_1, \ldots, v_n\}$ has an associated degree sequence, i.e. a finite sequence of integers $(d_1, \ldots, d_n)$ such that $d(v_i) = d_i$. The converse is not always true though: given an arbitrary finite sequence of non-negative integers L, there does not necessarily exist a simple graph which has L as its degree sequence. We give a name to sequences which are the degree sequence of some simple graph.

Definition 2.1. A finite sequence D of n non-negative integers is called *graphical* if there exists a graph on n vertices which has D as its degree sequence. Likewise, we call such a sequence non-graphical if no such graph exists.

Definition 2.2. If a degree sequence, D, is graphical, we call a *realization* of D a graph which has D as its degree sequence.

For example, $D = (3, 3, 1, 1)$ is non-graphical. To see this, assume G_D is a realization of D with vertex set $\{v_1, v_2, v_3, v_4\}$. Then v_1 and v_2 are attached to all others. Then since v_3 and v_4 are adjacent with both v_1 and v_2, the degree of both is greater than 1. Thus we have a contradiction and see that D is not graphical.

In general, a realization of a graphical sequence, $D = (d_1, \ldots, d_n)$, is not unique. Let S_D refer to the set of graphs which have D as its degree sequence. Obviously S_D is finite. Generating all graphs in S_D is in general not an easy problem. But once at least one graph in S_D is generated, there is an operation called "switching" (sometimes referred to as "swapping") which theoretically allows one to generate every graph in

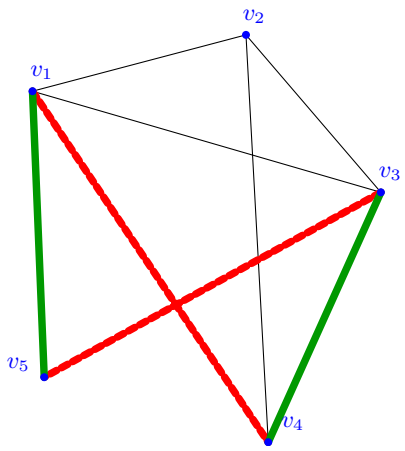


Figure 2.1 Switch in which $\{v_1, v_4\}$ and $\{v_3, v_5\}$ are deleted, and $\{v_1, v_5\}$ and $\{v_3, v_4\}$ are added

S_D. That is, given a graph G with a prescribed degree sequence D, performing a "switch" produces a different graph with the same degree sequence, given that G is not the unique realization of D. Switching goes as follows:

1. Find two independent edges $\{u, v\}, \{w, x\}$ such that both $\{u, w\}$ and $\{v, x\}$ are non-edges.

2. Switch end-vertices to produce a graph G' in which $\{u, v\}$ and $\{w, x\}$ are no longer edges and $\{u, w\}$ and $\{v, x\}$ are edges.

Note that all involved vertices maintain the same degree, and thus G' has the same degree sequence as G. Thus this operation is closed with respect to the state space S_D. Figure 2.1 shows an example of a switch.

As stated above, provided that one has a realization, switching allows one to generate every graph in S_D. That is, every realization in S_D can be reached by every other realization in S_D by a finite sequence of switches. We prove this below. We can think of the state space S_D as a graph, in which each event (realization) is a vertex, and there is an edge between two realizations if and only if one can be reached from the other by one switch. Note that switching is commutative, in that if realization A can be reached from realization B then B can be reached from A. Thus this graph

is non-directed. Since there is an $A - B$ path from each realization to every other realization, we say that this state space is connected with respect to switching.

Lemma 2.3. *Given a realizable degree sequence D on $n \leq 3$ vertices, the state space S_D is connected with respect to switching.*

Proof: For $n = 1, 2$, and 3, for any degree sequence on n vertices, the corresponding state space is a single point, and so it is connected. □

Claim 2.4. *Let G be a graph with degree sequence $(d_1, d_2, \ldots, d_n)$ where $d_1 \geq d_2 \geq \ldots \geq d_{n-1}, d_n > 0$. For a graph with such a degree sequence, let its vertex set be $\{v_1, v_2, \ldots, v_n\}$, where $deg(v_i) = d_i$. Let $\mathcal{G}(G)$ be the set of graphs that can be realized from G by a sequence of switches. Then there exists a graph G_0 in $\mathcal{G}(G)$ such that $N_{G_0}(v_n) = \{v_1, v_2, \ldots, v_{d_n}\}$.*

Proof: Clearly $d_n \leq n-1$. If $d_n = n-1$, then $N_G(v_n) = \{v_1, \ldots, v_{n-1}\}$ and so we are done. So assume $d_n < n-1$. Let $G_0 \in \mathcal{G}(G)$ be such that $|N_{G_0}(v_n) \cap \{v_1, \ldots, v_{d_n}\}|$ is maximal. Assume $N_{G_0}(v_n) \neq \{v_1, \ldots, v_{d_n}\}$. Let i be the smallest index such that $v_i \notin N_{G_0}(v_n)$. There exists $j > d_n$ such that $v_j v_n \in E(G_0)$. Now note that $i < j$ implies that $d_i \geq d_j$. Since $v_n \in N_{G_0}(v_j) \setminus N_{G_0}(v_i)$, there must exist some $k \notin \{i, j, n\}$ such that $v_k \in N_{G_0}(v_i) \setminus N_{G_0}(v_j)$. Then $\{v_i, v_k\}, \{v_n, v_j\} \in E(G_0)$ while $\{v_i, v_n\}, \{v_k, v_j\} \notin E(G_0)$. So we can perform a switch, making $v_i \in N_{G_0}(v_n)$, contradicting the maximality of G_0. Thus the claim is proven. □

Lemma 2.5. *Given a realizable degree sequence D on n vertices, the state space S_D is connected with respect to switching.*

Proof: The proof goes by induction on n. By Lemma 2.3, we know the statement holds for $n = 1, 2$, and 3. These serve as the base cases. So now assume the statement holds for any graph on $n-1$ vertices. Let $D = (d_1, d_2, \ldots, d_n)$ be a realizable sequence. Without loss of generality, we can assume that $d_1 \geq d_2 \geq \ldots \geq d_n$.

Let $G, H \in S_D$. If $d_n = 0$, then v_n is isolated in both G and H. Define $G^* = G \setminus \{v_n\}$ and $H^* = H \setminus \{v_n\}$. By the inductive hypothesis, there exists a sequence of switches from G^* to H^* that is also a sequence of switches from G to H. In this case, we are done. So assume $d_n > 0$. By Claim 2.4, we know that there exists a graph $G_0 \in S_D$ such that $N_{G_0}(v_n) = \{v_1, \dots v_{d_n}\}$, and there exists a sequence $\mathcal{S}_1$ of switches that take G to G_0. Likewise, there is such a graph H_0 with the same property, and there exists a sequence $\mathcal{S}_3$ of switches that take H to H_0. Now observe that $N_{G_0}(v_n) = N_{H_0}(v_n) = \{v_1, \dots, v_{d_n}\}$. This if we remove v_n from both G_0 and H_0, we remove the same edges from both, that is $\{v_n, v_i\}$ for $1 \leq i \leq d_n$. Then both $G_0 \setminus \{v_n\}$ and $H_0 \setminus \{v_n\}$ have the same degree sequences $d_1 - 1, d_2 - 1, \dots, d_{d_n} - 1, d_{d_n+1}, \dots, d_{n-1}$. So by the induction hypothesis, there exists a sequence of switches $\mathcal{S}_2$ from $G_0 \setminus \{v_n\}$ to $H_0 \setminus \{v_n\}$. Note that the switches in $\mathcal{S}_2$ are also switches from G_0 to H_0 since the switches do not involve any edges with v_n as an end-vertex. Let $\mathcal{S}_3'$ be the reversal of $\mathcal{S}_3$. Then $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3'$ is a sequence of switches from G to H. Since G and H are arbitrary, we see that S_D is connected. Since D is arbitrary, we see that for any degree sequence on n vertices, its state space is connected. By induction, the statement holds for all $n \in \mathbb{N}$. □

A natural question involving degree sequences is how to determine whether a given finite sequence of non-negative integers, L, is graphical or not. Call this the Graphicality Problem. One quick way to attain a negative answer to this question relies on one of the most fundamental theorems of graph theory:

Lemma 2.6. *(Handshaking [9]) - The sum of the degrees of the vertices of G equals twice the size of G and thus is always even. Notationally, $\sum_{v \in V(G)} deg(v) = 2 \cdot |E|$.*

So if the sum of the elements of L is odd, L must be non-graphical. A more substantial answer to the Graphicality Problem is given by the Erdős-Gallai theorem which uses this fact plus an extra condition to establish a complete characterization of graphical sequences.

Theorem 2.7. *(Erdős-Gallai [10]) A sequence of non-negative integers* $d_1 \geq \ldots \geq d_n$ *is graphical if and only if* $\sum_{i=1}^{n} d_i$ *is even and* $\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} min(d_i, k)$ *holds for* $1 \leq k \leq n$.

This theorem answers the Graphicality problem. It does not provide a concrete realization in the affirmative case though. The Havel-Hakimi Theorem provides a simple algorithm to determine whether a sequence is graphical, and in the affirmative case actually produces a realization of the prescribed degree sequence. The proof was given independently by Havel in 1955 [15] and Hakimi in 1962 [14].

Theorem 2.8. *(Havel-Hakimi) A sequence of integers* $d_1 \geq d_2 \geq \ldots \geq d_n > 0$ *is graphical if and only if* $(d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, d_{d_1+3}, \ldots, d_n)$ *is graphical.*

Proof: This is a quick corollary of Claim 2.4 (we need to renumber vertices). □

As stated, if the sequence is graphical, this theorem provides an iterative algorithm to produce a realization. The algorithm follows the following steps:

1. Start with an empty graph, G, and a dictionary L of n vertices as the keys and a residual degree requirement assigned as the values. The starting list of residual values will be the degree sequence.

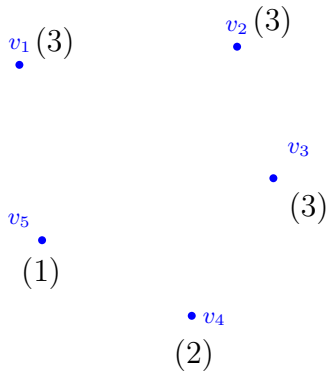
2. If all residual degrees are zero, output G and stop. If any residual degree is less than 0, output FAILURE and stop.

3. Maintain the dictionary in non-increasing order according to the residual degree values.

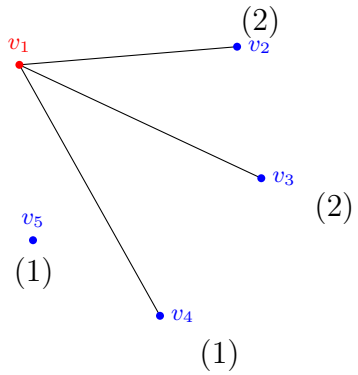
4. Assume the first vertex in the dictionary has a degree requirement of d. For the next d vertices in the list, attach an edge between each vertex and the first vertex.

5. Remove the first vertex from the dictionary and decrease the degree requirement of these next d vertices by 1.

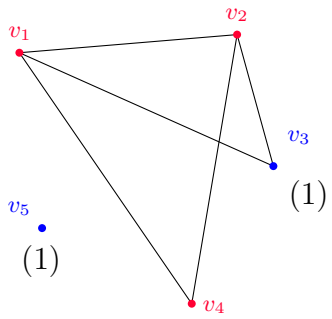
6. Go back to Step 2.



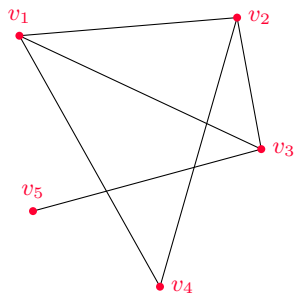
Empty graph after Step 1



After 1st iteration



After 2nd Iteration



Output graph

Figure 2.2 Iterations of Havel-Hakimi algorithm for degree sequence $(3, 3, 3, 2, 1)$

Figure 2.2 shows the graphs that are produced after each iteration of the Havel-Hakimi algorithm when $(3, 3, 3, 2, 1)$ is the input degree sequence. The numbers in parentheses next to the vertices represent the residual degree requirement of the corresponding vertex after that iteration.

When dealing with S_D for a given D, it is natural to ask how many graphs are realized by this sequence and thus lie in this space. We can easily characterize when this space only has one graph. Note that for every $n \in \mathbb{N}$, there is a uniquely realizable graph on n vertices, that is, a graph which is the only graph which realizes its degree

sequence. The complete graph on n vertices provides a trivial example. There is a way to characterize all such uniquely realizable graphs.

Define a graph G to be a *threshold graph* if for every vertex $x, y \in V(G), N(x) \subseteq N(y) \setminus \{x\}$ or $N(y) \subseteq N(x) \setminus \{y\}$. The name of threshold graphs comes from the fact that they can also be obtained as follows: assign non-negative weights to vertices and set a threshold T. Let $\{i, j\} \in E(G)$ if and only if $w(i) + w(j) > T$.

Lemma 2.9. *Threshold graphs are precisely those that are uniquely realizable.*

Proof: Suppose that a graph G is not a threshold graph. Then there is $x, y \in V(G)$ such that neither $N(x) \not\subseteq N(y) \setminus \{x\}$ nor $N(y) \not\subseteq N(x) \setminus \{y\}$. Then neither x nor y is isolated, and there exists a vertex $w \in N(x) \setminus N(y)$ and a vertex $v \in N(y) \setminus N(x)$. This means that $\{x, w\}, \{v, y\} \in E(G)$ and $\{x, v\}, \{y, w\} \notin E(G)$; also $x \neq v$ and $y \neq w$. Thus we can perform a switch from $\{x, w\}$ and $\{y, v\}$ to $\{x, v\}$ and $\{y, w\}$, making a new graph with the same degree sequence. So G is not uniquely realizable.

On the other hand, suppose that G is a threshold graph. Let $\{x, w\}$ and $\{y, v\}$ be two independent edges in G. Since $N(x) \subseteq N(y) \setminus \{x\}$ or $N(y) \subseteq N(x) \setminus \{y\}$, we have $v \in N(x) \setminus \{y\}$ or $w \in N(y) \setminus \{x\}$. In either case, a switch from these two edges is not permitted. Since these are arbitrary independent edges, we see that no switch is permitted and thus G is uniquely realizable. □

Thus we may speak of uniquely realizable graphs in terms of threshold graphs. We make several claims.

Claim 2.10. *Let I be the set of isolated vertices of the graph G. G is a threshold graph if and only if $G \setminus I$ is a threshold graph.*

This is self-evident.

Claim 2.11. *If G is a threshold graph and contains no isolated vertices, then there exists a vertex $x \in V(G)$ such that $d(x) = |V(G)| - 1$, i.e. $N(x) = V(G) \setminus \{x\}$.*

Proof: Let x be a vertex of maximal degree in G. Suppose that $d(x) < |V(G)| - 1$, i.e. $N(x) \neq V(G) \setminus \{x\}$. Then there exists a vertex y such that $y \notin N(x) \cup \{x\}$. Since there are no isolated vertices, there is an edge $\{y, z\}$ in G for some vertex $z \neq x$. Since G is a threshold graph, either $N(z) \subseteq N(x) \setminus \{z\}$ or $N(x) \subseteq N(z) \setminus \{x\}$. Since $y \in N(z) \setminus N(x), N(z) \not\subseteq N(x) \setminus \{z\}$. Suppose $N(x) \subseteq N(z) \setminus \{x\}$. Then $|N(x)| \leq |N(z)|$. Then since the degree of x is maximal, $|N(x)| = |N(z)|$ which further implies that $N(x) = N(z)$. But $y \in N(z)$, and $y \notin N(x)$, which is a contradiction. Thus we see that $N(x) = V(G) \setminus \{x\}$. □

From the previous two claims and the definition, the following claim follows:

Claim 2.12. *If G is a graph and x is adjacent with every other vertex in the graph, then G is a threshold graph if and only if $G \setminus \{x\}$ is a threshold graph.*

Thus from these three claims, we have an iterative algorithm to check whether a graph G is uniquely realizable or not:

1. Delete all isolated vertices. If the remaining graph is empty, stop and output that G is a threshold graph.

2. Search for a vertex that is adjacent with every other vertex. If not found, then stop and output that G is not uniquely realizable. If found, then delete that vertex and all its incident edges. Go back to Step 1.

This characterizes situations in which the state space consists of a single graph. There are several results regarding the enumeration of graphs with certain degree sequences. Linyuan Lu and László Székely apply the Lovász Local Lemma to the space of random multi-graphs with a given degree sequence to obtain asymptotic enumeration results in [19]. See [2] for an overview of the Lovász Local Lemma, a celebrated tool in the probabilistic method. They also outline a brief history of results involving enumeration of d-regular graphs, that is graphs in which every vertex has degree d. They attribute the strongest result to McKay and Wormald (strongest in the sense that the range of d is the largest). In [21], McKay and Wormald give the

following best asymptotic boundary on the regularity d. Let $(N-1)!! = \frac{N!}{N/2)!2^{N/2}}$ for N even. They show that the number of simple graphs in which all vertices have degree d for $d \in o(n^{1/2})$ is

$$(1+o(1))e^{\frac{1-d^2}{4}-\frac{d^3}{12n}+O(\frac{d^2}{n})}\frac{(dn-1)!!}{(d!)^n}.$$

Lu and Székely first state a result involving d-regular graphs and then generalize to a larger class of degree sequences. We state the generalization here. Let x_i be real numbers and assume $x_i > 0$ for each $i \in [n]$ for some $n \in \mathbb{N}$. Define $\overline{x} = (\sum_{i=1}^n x_i)/n$ and $\tilde{x} = (\sum_{i=1}^n x_i^2)/\overline{x}$.

Theorem 2.13. *Let $d_i \in \mathbb{N}$ for $i \in [n]$. Assume $N = d_1 + d_2 + \ldots + d_n$ is even, $\overline{d} \geq 3$, and every $d_i \geq 2$. Let $D_i = d_i(d_i - 1)$. Then the number of graphs with degree sequence $d_1, d_2, \ldots, d_n$ and girth $g \geq 3$ is*

$$(1+o(1))\frac{(N-1)!!}{\prod_i d_i!}\exp(-\sum_{i=1}^{g-1}\frac{1}{2i}(\frac{\overline{D}}{\overline{d}})^i)$$

assuming that

$$(\frac{g^2}{n}+\frac{g^2}{n^2}\frac{\tilde{D}}{\overline{d}})(\frac{\overline{D}}{\overline{d}})^{g-1} = o(1)$$

and

$$g^6(\frac{\overline{D}}{\overline{d}})^{2g-4}d_n^2 = o(N).$$

This is the state of the art with respect to the enumeration of graphs with general degree sequences.

There has been significant attention devoted to random generation of graphs with given degree distributions, a task of importance to network science. Much significant work has been done by treating degree sequences rather than distributions. This distinction is not problematic though in that they correspond to each other: one determines the other. Since the Havel-Hakimi algorithm produces a graph in S_D and we theoretically can attain every other graph in S_D by switches, these methods serve as useful tools in this modeling and generation effort. In our work, for a fixed D,

we generate a realization, G, of D via the Havel-Hakimi algorithm. We then run a Monte-Carlo Markov Chain on S_D with starting point G_D via the use of switches. We define Markov Chains in the next section.

2.2 Markov Chains

The following definitions are taken from [16], chapters 1,2, and 3. A Markov chain is a specific kind of probabilistic model. It is an example of a more general notion called a stochastic process.

Definition 2.14. A *stochastic process* is a family of random variables defined on some sample space Ω.

For our purposes, there is a countable number of random variables, and so we enumerate them as $X_0, X_1, X_2, X_3, \ldots$. Intuitively, the indices of the random variables refer to the time-step of the process. When we say that the process lies in state i at the n_{th} step, we really mean that X_n takes on the value i. This is called a discrete-time process. We define the state space of the stochastic process as the union of the possible values of the individual random variables.

Definition 2.15. A *Markov chain*, $\mathcal{C}$, consisting of the random variables $\{X_k\}, k = 0, 1, 2, \ldots$ with state space S is a stochastic process that satisfies the following properties:

1. It is a discrete-time process.
2. S is countable or finite.
3. Markov property: If for every m and all states $i_1, i_2, \ldots, i_m$ where i_j is a possible value of X_j, it is true that

$$P(X_m = i_m | X_{m-1} = i_{m-1}, X_{m-2} = i_{m-2}, \ldots, X_1 = i_1) = P(X_m = i_m | X_{m-1} = i_{m-1})$$

In words, the proximate value of a random variable at a point in time only depends on its current value and does not depend on previous values in the process. The process is "memoryless." Intuitively, a Markov chain travels along a state space. We will only work with finite state spaces, so we may say $S = [N]$ for some $N \in \mathbb{N}$. At each time-step, the random variable corresponding to that step takes on values of

the state space with a given probability. The factors determining the movement of a Markov chain are governed by the set of conditional probabilities

$$P(X_m = j|X_{m-1} = k)$$

for every pair of states k, j in the state space. We can also refer to these as transitional probabilities at time m. We can further classify these processes according to whether these conditional probabilities are time-dependent or not.

Definition 2.16. A Markov chain is *homogeneous* in time if the transition probability from any given state to any other given state is independent of the time-step. That is, for arbitrary m and every pair $j, i \in S$,

$$P(X_m = j|X_{m-1} = i) = P(X_{m+k} = j|X_{m-1+k} = i)$$

for all k such that $-(n-1) \leq k$.

The Markov Chain we use is finite and homogeneous. Set $N = |S|$. We can describe the transition probabilities by an $N \times N$ *transition matrix*, $T = (p_{i,j})$, where

$$p_{i,j} = P[X_1 = j|X_0 = i] = P(X_{m+1} = j|X_m = i).$$

It follows from this definition that

$$1 = P[X_1 \in [N]|X_0 = i] = \sum_{j=1}^{N} p_{i,j}.$$

The transition matrix serves as a very useful tool in understanding a Markov chain. In fact, the analysis of a Markov chain can often be done completely by looking at its transition matrix. One might ask what is the probability that the chain lies in a given state at a given time. Transition matrices aid in answering this question.

First, given a current state at time k, the transition matrix can be used to calculate the probability of being in all other states after m steps for $m \geq 1$. This probability is given by T^m. The entry $T^m_{i,j}$ represents $P(X_{k+m} = j|X_k = i)$. Now suppose

that we have some information about the starting point in the chain, specifically we have a probability distribution for X_0. We may encode this information as a vector $\alpha_0 = (p_1, p_2, \ldots, p_N)$ where p_i refers to the probability that X_0 is in state i. Note that this allows for a deterministic or non-deterministic starting point. If deterministic, then we assign a probability of 1 to its determined value in α_0 and a probability of 0 everywhere else. It turns out that T and α_0 is all that is needed to give a probability distribution for the state of $\mathcal{C}$ at a certain time, or rather a probability distribution of X_m. Let $\alpha_m = \alpha_0 \cdot T^m$. Then α_m gives this probability distribution. That is, α_{m_i} gives the probability that $\mathcal{C}$ lies in state i at the m_{th} step given the starting vector α_0. Note that this is not a contradiction of being homogeneous. The distinction is that the probabilities of transitioning from a specific state to another do not change with time, but rather the probability that $\mathcal{C}$ lies in a certain state do change with time. Often, one is not interested in the behavior of a Markov chain, $\mathcal{C}$, at a specific point in time but rather interested in its long-term behavior. Does α_m converge as $m \to \infty$? Some Markov chains do have this property while some do not.

Definition 2.17. State i is *accessible* from state j if $\Pr(X_n = i | X_0 = j) > 0$ for some $n \in \mathbb{N}$.

Definition 2.18. State i and j *communicate* if state i is accessible from state j and state j is accessible from state i.

Definition 2.19. A Markov chain is said to be irreducible if for every pair of states $i < j$, i and j communicate.

Using the language we used above with the degree sequence, this is the same as saying that the state space is connected.

Definition 2.20. The *period*, R, of a state i is defined as

$$gcd\{n : \Pr(X_n = i | X_0 = i) > 0\},$$

where *gcd* is the greatest common denominator.

Definition 2.21. A state i is said to be *aperiodic* if its period is 1.

Definition 2.22. A Markov chain is said to be *aperiodic* if each state is aperiodic.

Definition 2.23. A vector π that satisfies $\pi = \pi \cdot T$ is called a *steady-state distribution* of $\mathcal{C}$.

Theorem 2.24. *A time-homogeneous, irreducible and aperiodic Markov chain with a finite state space has a unique steady-state distribution.*

Note that for a time-homogeneous, irreducible and aperiodic Markov chain, $\pi = \lim_{m\to\infty} \alpha_0 \cdot T^m$ for any initial distribution α_0, i.e. the chain converges to the steady-state distribution regardless of the initial distribution.

Definition 2.25. Suppose a Markov Chain, $\mathcal{C}$, converges to a steady-state distribution π. Then the *mixing time* t of $\mathcal{C}$ is the minimum t such that $|\Pr(X_m = i) - \pi(i)| \leq 1/4$ for all $m \geq t$ and all states i in the sate space.

Intuitively, the mixing time refers to how many steps are necessary to get sufficiently close to the steady-state distribution. Before we describe our project exactly, it is important to give an introduction to one more theoretical concept, that is spectral graph theory.

The Markov Chain we use will have state space S_D for some realizable degree sequence D. The steps of the chain will be defined as follows: Given D, $X_0 = H$, where H is created by the Havel-Hakimi algorithm. The transition steps will be time-independent (thus, the chain is homogeneous); and each step will be defined either as a switch or staying in place, i.e. $P(X_{m+1} = G' | X_m = G) > 0$ if and only if $G' = G$ or G' can be reached from G by a single switch. We will describe these probabilities in Section 2.4.

2.3 Laplacian Spectrum

Many structural properties are captured by algebraic properties of a graph. Thus the study of certain matrices associated with graphs has proven fruitful. Spectral graph theory is the study of how properties of a graph related to the characteristic polynomial, eigenvalues and eigenvectors of matrices associated with the graph (such as the adjacency matrix or Laplacian matrix). In the following definitions, assume the graph has n vertices.

Definition 2.26. The *degree matrix* $\mathcal{D}$ of a graph is the $n \times n$ matrix with (i, j)-entry: $\deg(v_i)$ for $i = j$ and 0 for $i \neq j$. The degree matrix essentially encodes the degrees of the individual vertices along the diagonals.

Definition 2.27. The *adjacency matrix* of a graph is the $n \times n$ matrix with (i, j)-entry: 1 if v_i is adjacent with v_j and 0 otherwise.

Definition 2.28. If $\mathcal{D}$ is the degree matrix of a graph and A is its adjacency matrix, then its *Laplacian matrix* is the $n \times n$ matrix $L = \mathcal{D} - A$. Thus the (i, j)-entry of L is $\deg(v_i)$ for $i = j$; -1 for $i \neq j$ and v_i, v_j adjacent; 0 for $i \neq j$ and v_i, v_j non-adjacent.

Note that all three of these matrices are symmetric, that is the (i, j)-entry is equal to the (j, i)-entry for all i, j.

Definition 2.29. Given an n-dimensional square matrix M over $\mathbb{R}$, its *eigenvalues* are the values λ such that $MX = \lambda X$ for some vector $X \in \mathbb{R}^n \setminus \{0\}$. If $MX = \lambda X$ for some λ, then X is called an *eigenvector* of M. The spectrum of the matrix is the multiset of eigenvalues, where the eigenvalue λ has multiplicity $\dim(V)$, where $V \subseteq \mathbb{R}^n$ is the maximal subspace such that all $X \in V, MX = \lambda X$.

It is not necessarily true that such a matrix M will have eigenvalues, but for the Laplacian matrix of a graph, it is true. The follows results are summarized in [34].

Lemma 2.30. *([20]) The Laplacian L of a graph G is a singular, positive semi-definite, symmetric matrix, and thus has n nonnegative eigenvalues. $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. Since each row sum of L equals 0, for $V = (1, 1, \ldots, 1)$, we have $VL = (0, 0, \ldots, 0) = 0V$, i.e. $\lambda_1 = 0$.*

As said before, eigenvalues contain a lot of information about the structural properties of a graph. We list a few main results here to give the reader a flavor of the connection between a graph and the eigenvalues of its Laplacian.

One of the most celebrated and earliest results involves the number of spanning trees of a graph:

Theorem 2.31. *(Kirchhoff Matrix Tree Theorem [17]) Let $L(i|j)$ be the $(n-1) \times (n-1)$ submatrix of L, obtained by deleting the i_{th} row and j_{th} column. Denote by $\tau(G)$ the number of spanning trees in G. Then $\tau(G) = (-1)^{i+j} \det L(i|j) = \frac{1}{n} \prod_{i=2}^{n} \lambda_i$.*

As a corollary of Theorem 2.31, $\lambda_2 > 0$ if and only if G is connected, i.e. it has at least one spanning tree. Thus we call the second smallest eigenvalue of L the algebraic connectivity of G. This is because it gives an indication of whether G is connected or not. On a further note, the multiplicity of 0 as an eigenvalue gives the number of connected components of G [22]. The value of the second smallest eigenvalue has been shown to have many other connections with structural parameters of a graph, as we will see presently.

The question of how highly connected an arbitrary graph is has been a topic of intense study over the years. This notion has often been linked with various other structural properties of a graph. There are many interesting results relating connectivity and Laplacian eigenvalues. First though, we need to formally define the notion of vertex and edge connectivity, notions which quantify how highly "connected" a graph is.

Definition 2.32. The *(vertex) connectivity* $\mathcal{K}(K_n)$ of the complete graph K_n on n vertices is $n-1$. If G is a graph that is not complete, a vertex cut of G is a set of vertices whose removal disconnects the graph G, and $\mathcal{K}(G)$ is the minimum size of any vertex cut. The graph is k-connected if its vertex connectivity is at least k.

There is a similar notion involving the edges of a graph.

Definition 2.33. An edge cut of a graph G on at least two points is a set of edges whose removal disconnects the graph. The *edge connectivity*, $\mathcal{E}(G)$ is the minimum size of an edge cut.

The following result is due to Fielder:

Theorem 2.34. *([12]) Let G be any simple graph on n vertices other than the complete graph. Assume G has vertex connectivity $\mathcal{K}(G)$ and edge connectivity $\mathcal{E}(G)$. Then*

$$2\mathcal{E}(G)(1-\cos(\pi/n)) \leq \lambda_2 \leq \mathcal{K}(G) \leq \mathcal{E}(G).$$

Now we give a few results concerning some other structural aspects of a graph. Remember that a *dominating set* D is a subset of V such that every vertex not in D has at least one neighbor in D. Define $\gamma(G)$ to be the minimum size of a dominating set in G. Call this the domination number of G. Nikiforov [23] gives an upper bound on the second smallest eigenvalue in terms of the domination number:

Theorem 2.35. *([23]) If G is a connected graph other than the complete graph, then $\lambda_2(G) \leq n - \gamma(G)$.*

The diameter of G is an important graph parameter. The diameter of a connected graph G is the maximum distance among all pairs of vertices of G. Denote this value by $diam(G)$. Lu gives a bound on λ_2 from below in terms of $diam(G)$, its size, and order:

Theorem 2.36. *([20]) Let G be a simple connected graph of order n, size m, and diameter $diam(G)$. Then $\lambda_2 \geq \frac{2n}{2+n(n-1)(diam(G)-2m(diam(G)))}$ with equality if and only if G is a path of order 3 or the complete graph.*

Definition 2.37. A graph is called *bipartite* if its vertex set V can be partitioned into vertex sets A, B such that all edges of G connect a vertex of A to a vertex of B.

The following theorem gives a necessary and sufficient condition for a graph being bipartite and regular.

Theorem 2.38. *([28]) Assume that the degree sequence of G is $d_1 \geq d_2 \geq \ldots \geq d_n$. Then $\lambda_n \leq d_n + \frac{1}{2} + \sqrt{(d_n - \frac{1}{2})^2 + \sum_{i=1}^{n} d_i(d_i - d_n)}$ with equality if and only if G is a regular bipartite graph.*

2.4 Program

As a significant contribution to my thesis, we have undertaken a project whose goal is to determine how local structure of a network is determined by the degree sequence alone of local samples of the network. In determining this structure, we are interested in those structural properties captured by the Laplacian spectrum of the studied samples. To this end, we have taken four local snowball samples of the network and determined their degree sequence and spectrum. For each sample, we ran a Monte-Carlo Markov chain (MCMC) on the space of graphs with the corresponding degree sequence to produce a large number of random graphs with this corresponding degree sequence. We computed the spectrum of each random graph and then created an empirical probability density function for these spectra to see whether there are any observable localized eigenvalues. Appendix B contains this data, in which one can see the degree sequence and spectrum of each network sample as well as the empirical probability density function for the corresponding MCMC generated spectra.

We will now describe this project in more detail. We received a data representation of a massive empirical real-world scale-free network that was generously provided to us by Dr. Zoltán Toroczkai. As stated, we extract local connected sample of this network according to a technique called snowball sampling which goes as follows: we randomly select with uniform probability a vertex v from the network and add v to a list; we add all vertices within a distance of d from v to the list where d is the minimum integer such that at least 100 vertices are in the list. After this, we take the induced subgraph on the vertices added to the list, and we refer to this as a *local sample*.

The next step is to record the degree sequence of the local sample, D. Since D is taken directly from a graph, we know it is realizable. So we use the Havel-Hakimi algorithm to create a realization of this sequence, G_D. We run a Monte-Carlo Markov chain (MCMC), on S_D, in which each step is a switch between two graphs in S_D.

Denote the Markov chain by $X = \{X_i\}_{i=1}^{\infty}$. Each random variable $X_i : E_{i-1} \to S_D$ outputs a graph in S_D as its value, where E_{i-1} is the set of pairs of edges that permit a switch in the graph that X_{i-1} takes on as a value. G_D provides a starting state for the Markov chain. That is, X_0 takes on the value G_D with probability one. Suppose that D is of length n i.e. there are n vertices in the local sample. Then we run X for n^2 steps to produce a graph G_i^D. We run the MCMC t times, ultimately generating an ensemble of t graphs with the degree sequence D: $\{G_1^D, \ldots, G_t^D\}$. Denote this ensemble by $\mathcal{G}_D$.

Next, we generate the Laplacian matrix and compute the spectrum of this Laplacian for each graph $G_i^D \in \mathcal{G}_D$. Let λ_D be the multiset of all values λ such that λ is an eigenvalue of some graph $G_i^D \in \mathcal{G}_D$. λ_D has nt values counted with multiplicities. We divide the real line into bin widths of length .001. Then for each bin, b, let Z_b be the number of eigenvalues (counted with multiplicity) from λ_D that lie in b. Let $f(b) = \frac{Z_b}{nt \cdot .001}$. Let Q be the set of bins q such that $f(q) \neq 0$. Then $\sum_{q \in Q} f(q) \cdot .001 = 1$. This empirical function can be viewed as a representation of the following: let δ be the Dirac delta function. Formally, $\delta = 0$ everywhere except at 0, and $\int_{-\infty}^{\infty} \delta(x)dx = 1$. Then our function is a representation of $g(x) = \frac{1}{nt} \sum_{z \in \lambda_D} \delta(x - z)$. Appendix B shows the graph of this empirical probability distribution function for each of the four degree sequences. There is a normal and zoomed in picture of each graph, where the zoom is on the y-axis.

We will now describe practical concerns of the project. The majority of this work is done in the computer language Python. We take an object-oriented approach, the objects being graphs, as this allows for easy alteration of the graphs. We created a class object to represent a graph. We also created several methods that carry basic information about the graph and also allow one to alter the graph. In representing the empirical network, we first used the data form given to us by Dr. Toroczkai to create a graph object representing this network. To induce a subgraph, we create a

new graph object to represent the subgraph and only add appropriate edges. In all random selections used, we use Python's built in random number generator. From a programming aspect, extracting the snowball sample amounts to adding all vertices at a specific distance to a list, starting at a distance of one with v's neighbors, and then checking whether the order of the list has reached 100 yet. If not, then add all those vertices at the next distance from v, which are the neighbors of those added in the last step. The sample "snowballs" in this way. Since the program adds all vertices at a distance of d and does not stop when it has exactly reached 100, in practice the samples most likely will contain more than 100 vertices. All of the samples we extracted in fact do.

The Markov chain is implemented rather computationally easy due to the object-oriented nature of the graphs. We randomly select two edges $\{x, y\}, \{v, w\}$ and check if a switch is permitted. If two switches are permitted, that is, we can switch to $\{x, v\}, \{y, w\}$ or$\{x, w\}, \{y, v\}$, then we randomly choose between which switch. If only one is permitted, we perform that switch. If no switches are permitted, then we do not switch. Then we select two more random edges and repeat the process. To perform the switch, we simply call the methods to delete and add edges and leave the rest of the graph unchanged. This process only requires a delicate alteration of the graph and does not have to create a whole new object.

To obtain the transition matrix T for this MCMC, one must look at the structure of this state space, which relies on the structure of the different realizations. Let $S_D = \{G_1, G_2, ..., G_N\}$. Let $G_i \in S_D$. For all graphs $G_j \neq G_i$ which are not adjacent to G_i, specifically graphs that can not be attained by one switch, the probability of transitioning from G_i to G_j and vice-versa is zero. That is, $p_{i,j} = p_{j,i} = 0$. The program randomly picks two edges from G_i. The way in which it does this allows for the same edge to be chosen twice. Suppose that there are e edges in G_i. Then there are e^2 choices of pairs of edges. Let p be the fraction of pairs of edges which

do not permit a switch. Then the set of edge-pairs which do permit a switch has probability $1-p$. Assume that we have s pairs of edges that allow for a switch and a total of u switches that can be made. Note that as some pairs of edges may allow for two switches, one immediately obtains that $u \leq s \leq 2u$. Let m be the number of times we select an edge pair that does not allow for a switch before we select one that does. Note m can be any non-negative integer. The probability of this is $p^m(1-p)$. If we want to see what the probability that we will make a fixed switch in this step, we need to divide our analysis into two parts. First, assume that this switch belongs to an edge pair that only allows a single switch. Then we will perform this switch with probability $\frac{1}{s}$ at this point (as we only have to have selected the edge-pair corresponding to this switch), so the probability that this switch is made after the $(m+1)$-st step is $\frac{p^m(1-p)}{s}$, and the probability that this is the switch that we will use to move from our graph to another one is $\sum_{m=0}^{\infty} \frac{p^m(1-p)}{s} = \frac{1}{s}$. Similarly, if we assume that the switch we are interested in belongs to an edge-pair that allows for two switches, the probability that our selected switch will be performed in the $(m+1)$-st step is $\frac{p^m(1-p)}{2s}$, and the probability that we will use this selected switch is $\frac{1}{2s}$. Thus, this program does not select all possible switches with equal probability.

We know that this Markov chain converges to a steady-state distribution. Theorem 2.24 gives sufficient conditions for a Markov chain with a finite state space to converge to a steady-state distribution. These conditions are aperiodicity and irreducibility. In Lemma 2.3, we proved that S_D is connected with respect to switching. Thus with positive probability, each state is reached from every other. In other words, the state space is irreducible. By [7], the state space is aperiodic. Thus we see that there is in fact convergence to the steady-state distribution. The hope is that if the MCMC is run "sufficiently" long enough, then each graph in S_D will be chosen according to the steady-state distribution. Note that the steady- state distribution is not known in general, and the mixing time of X is not known either. Supposing the

graphs in S_D have n vertices, we run X for n^2 steps as this is thought possibly to be sufficient time to allow for mixing.

We utilize an open-source software package NetworkX to create the Laplacian. This package is built on Python and contains an extensive library of tools for working with graphs [18]. To compute the eigenvalues, we use Matlab ®[30]. For Sample 1, we ran the Markov Chain approximately 300,000 times. For Samples 2, 3, and 4, we ran the Markov chain 400,000 times each. See Appendix A for the Python code and the next section for an analysis of the data.

2.5 Results

Table 2.1 summarizes the degree sequences of the four local samples that we extracted from the network. Appendix B displays the PDF eigenvalue spectrum data. For each network sample, we include the PDF of the spectrum of the actual sample. We then include a frequency count for each degree in the degree sequence. Lastly, we include a broad-range view and zoomed in view of the PDF of the spectrum of the MCMC generated samples with the degree sequence of the network sample.

Theorem 2.38 gives an upper bound on the largest Laplacian eigenvalue of a graph in terms of its degree sequence. For the degree sequence of Sample 1, we get a bound of 85.81. For the degree sequence of Sample 2, we get a bound of 67.45. For the degree sequence of Sample 3, we get a bound of 48.34. For the degree sequence of Sample 4, we get a bound of 56.10. Observing the spectral data in Appendix B, one sees that the range of the average spectrum for each sample falls within these respective bounds.

Comparing the spectrum of the network samples with the average spectrum from the ensemble MCMC generated samples, we see that the pictures look very similar. We will refer to the network sample as the "Sample" and the respective MCMC generated sample spectrum as the "Average". Sample and Average 1 have many eigenvalues clustered within the range of 0 to 14. Both the Sample and Average have a small spike around 21 and 71. Sample and Average 2 have a cluster of eigenvalues within the range of 0 to 11. The Average has several isolated eigenvalues around $14, 15.5, 17, 22, 23, 27$, and 31. In the Average, we see a triple hump between 20 and 25 and a double hump around 15. Sample 3 has eigenvalues dispersed throughout, more densely packed from 0 to 7 and becoming more spread out up till 21. Average 3 sees the same phenomenon with single humps around $17, 19, 21$. Sample 4 has eigenvalues packed tightly from 0 to 15 with a spike around 19 and 32. Average 4 sees the same tight packing as the sample, two double humps between 15 and 20, and a spike

Sample	Order	Degree Sequence
1	134	70, 21, 12, 12, 11, 11, 10, 9, 8, 8, 8, 8, 8, 8, 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
2	141	29, 26, 22, 22, 21, 17, 14, 14, 10, 9, 8, 8, 7, 7, 6, 6, 6, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
3	101	20, 18, 16, 13, 13, 12, 12, 9, 9, 8, 8, 7, 7, 6, 6, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
4	172	31, 17, 16, 13, 12, 11, 11, 9, 9, 9, 9, 8, 7, 7, 7, 7, 6, 6, 6, 6, 6, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

Table 2.1 Local Samples' Degree Sequences

around 31. It is highly plausible that localized spikes seen in the Samples correspond to humps localized at the same eigenvalue range in the Averages.

For all of the samples, there was not a significant contribution coming from integer eigenvalues except for $0, 1, 2, 3$, and 4. As said before, the multiplicity of 0 as an eigenvalue gives the number of components. For Average 1, 0 accounts for approximately $.01588 \approx \frac{2}{134}$ of all eigenvalues. Thus on average, graphs from this ensemble

have 2 components. For Average 2, 0 accounts for approximately .02035 $\approx \frac{3}{141}$ of all eigenvalues. Thus on average, graphs from this ensemble have 3 components. For Average 3, 0 accounts for approximately .03396 $\approx \frac{3.75}{101}$ of all eigenvalues. Thus on average, graphs from this ensemble have 3.75 components. For Average 4, 0 accounts for approximately .03876 $\approx \frac{7.6}{172}$ of all eigenvalues. Thus on average, graphs from this ensemble have 7.6 components.

A phenomenon we do see in the Averages is a hump starting approximately one value above the greatest degree. For each degree sequence, D of length n, we compute the weight, $W = m/1000$ of these humps, where m refers to the measure of the hump in the PDF. We compare this number to $1/n$, to get an approximate weight of how many eigenvalues are on average accounted for in this hump. For Average 1, this number is .0074 $\approx \frac{1}{134} = .0074$. Thus this hump accounts for 1 eigenvalue from each graph on average. For Average 2, this number is .0069 $\approx \frac{1}{141} = .0074$. For Average 3, this number is .0099 $\approx \frac{1}{101} = .0099$. The two humps between 15 and 20 each account for approximately one eigenvalue. For Average 4, this number is .0058 $\approx \frac{1}{172} = .0058$. The two double humps centered between 15 and 20 each account for half of an eigenvalue. So in conclusion, we see that the humps located at approximately one value larger than the largest degree account for approximately exactly one eigenvalue in each graph on average for each degree sequence. This could be investigated further. For each degree sequence, we see that the Sample spectra and the corresponding Average spectra do have many similar properties.

Bibliography

[1] William Aiello, Fan Chung, and Linyuan Lu, *A random graph model for massive graphs*, Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing (New York), ACM, 2000, pp. 171–180 (electronic). 2114530

[2] Noga Alon and H. Joel Spencer, *The probabilistic method*, 3rd ed., John Wiley and Sons, Inc., 2008.

[3] Albert-László Barabási, *Barabıasilab: Center for complex network research at northeastern university*.

[4] Albert-László Barabási and Réka Albert, *Emergence of scaling in random networks*, Science **286** (1999), no. 5439, 509–512. 2091634

[5] Albert-László Barabási, Duncan J. Watts, and Mark Newman, *The structure and dynamics of networks*, Princeton University Press, 41 William Street, Princeton, New Jersey 08540, 2006.

[6] Béla Bollobás, Oliver Riordan, Joel Spencer, and Gábor Tusnády, *The degree sequence of a scale-free random graph process*, Random Structures Algorithms **18** (2001), no. 3, 279–290. 1824277 (2002b:05121)

[7] Richard A. Brualdi, *Matrices of zeros and ones with fixed row and column sum vectors*, Linear Algebra Appl. **33** (1980), 159–231. 585770 (82c:15019)

[8] Ithiel de Sola Pool and Manfred Kochen, *Contacts and influence*, Social Networks **1** (1978/79), no. 1, 5–51. 506602 (80d:92039)

[9] Reinhard Diestel, *Graph theory*, 3rd ed., Springer, 2005.

[10] Paul Erdős and Tibor Gallai, *Graphs with prescribed degrees of vertices (hungarian)*, Mat. Lapok **11** (1960), 264–274.

[11] Paul Erdős and Alfréd Rényi, *On the evolution of random graphs*, Magyar Tud. Akad. Mat. Kutató Int. Közl. **5** (1960), 17–61. 0125031 (23 #A2338)

[12] Miroslav Fiedler, *Algebra connectivity of graphs*, Czechoslovake Mathematical Journal **23(98)** (1973), 298–305.

[13] Bogdan Guişcă, *The problem of the seven bridges of konisberg*.

[14] S Louis Hakimi, *On realizability of a set of integers as degrees of the vertices of a linear graph. I*, J. Soc. Indust. Appl. Math. **10** (1962), 496–506. 0148049 (26 #5558)

[15] Václav Havel, *A remark on the existence of finite graphs (czech)*, Časopis Pěst Mat. **80** (1955), 477–480.

[16] Dean L. Isaacson and Richard W. Madsen, *Markov chains theory and applications*, John Wiley and Sons, 1976.

[17] Gustav Kirchhoff, *Ęber die auflŽsung der gleichungen, auf welche man bei der untersuchung der linearen verteilung galvanischer strome gef§hrt wird*, Ann Phys. Chem **72** (1847), 497–508.

[18] Los Alamos National Laboratory, *Networkx*.

[19] Linyuan Lu and László Székely, *A new asymptotic enumeration technique: The lovász local lemma*, submitted to J. Comb. Theory, 2011.

[20] Mei Lu, Lian-zhu Zhang, and Feng Tian, *Lower bounds of the Laplacian spectrum of graphs based on diameter*, Linear Algebra Appl. **420** (2007), no. 2-3, 400–406. 2278217 (2008c:05111)

[21] Brendan D. McKay and Nicholas C. Wormald, *Asymptotic enumeration by degree sequence of graphs with degrees*, Combintorica **11 no. 4** (1991), 369–382.

[22] Bojan Mohar, *Some applications of Laplace eigenvalues of graphs*, Graph symmetry (Montreal, PQ, 1996), NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci., vol. 497, Kluwer Acad. Publ., Dordrecht, 1997, pp. 225–275. 1468791 (98g:05096)

[23] Vladimir Nikiforov, *Bounds of graph eigenvalues. I*, Linear Algebra Appl. **420** (2007), no. 2-3, 667–671. 2278241 (2007j:05144)

[24] Derek de Solla Price, *Networks of scientific papers*, Science **149** (1965), 510–515.

[25] _____, *A general theory of bibliometric and other cumulative advantage processes*, J. Amer. Soc. Inform. Sci. **27** (1976), 292–306.

[26] Sidney Redner, *How popular is your paper?*, Phys. J. B **4** (1998), 131–134.

[27] Sidney I. Resnick, *A probability path*, 5th ed., Birkh auser, 2005.

[28] Jin-Long Shu, Yuan Hong, and Wen-Ren Kai, *A sharp upper bound on the largest eigenvalue of the Laplacian matrix of a graph*, Linear Algebra Appl. **347** (2002), 123–129. 1899886 (2003c:05147)

[29] Ray Solomonoff and Anatol Rapoport, *Connectivity of random nets*, Bull. Math. Biophys. **13** (1951), 107–117. 0041409 (12,843d)

[30] TheMathWorks, *Matlab*, 2007a, Natick, Massachussets.

[31] Wikipedia user: Booyabazooka, *Diagram of seven bridges of konisberg.*

[32] Wikipedia user: Riojajar, *Abstract graph corresponding to bridges of konigsburg.*

[33] Duncan J. Watts and Steven H. Strogatz, *Collective dynamics of 'small-world' networks*, Nature **393** (1998), 440–442.

[34] Xiao-Dong Zhang, *The laplacian eigenvalues of graphs: a survey*, Department of Mathematics, Shanghai Jiao Tong University.

Appendix A

Code

```
class Graph:
    def __init__(self):
        self.order=0
        self.size=0
        self.graph={}
        self.vertexNbhd=[]
        self.checksize=0
        self.edges=[]

    def addVertex(self,vertex,neighbor):
        self.vertexNbhd=[]
        self.vertexNbhd.append(neighbor)
        self.graph[vertex]=self.vertexNbhd
        self.order=self.order+1

    def IncidentVertex(self,index):
        if index==0:
            return 1
        if index==1:
            return 0
    def addEdge(self,edge):
```

```
        for i in range(2):
            vertex=edge[i]
            nbhd=self.graph.get(vertex,False)
            if nbhd != False:
                j=edge[self.IncidentVertex(i)]
                self.graph[vertex].append(j)
            else:
                j=edge[self.IncidentVertex(i)])
                self.addVertex(vertex,j)

        self.size=self.size+1
        self.edges.append(edge)

    def deleteEdge(self,edge):
        for i in range(2):
            vertex=edge[i]
            nbhd=self.graph.get(vertex,False)
            nbhd.remove(edge[self.IncidentVertex(i)])
        self.edges.remove(edge)

    def CheckSize(self):
        self.checksize=0
        nbhds=self.graph.values()
        for i in nbhds:
            self.checksize=self.checksize+len(i)
        self.checksize=self.checksize/2
```

```
    def Vertices(self):
        return self.graph.keys()

    def NbhdList(self,vertex):
        nbhd=self.graph.get(vertex,False)
        return nbhd

def main(data):
    graph1=Graph()
    for edge in data:
        graph1.addEdge(edge)
    graph1.CheckSize()
   return graph1

from GraphClass import Graph
from random import randrange
import networkx as nx
#1.  opens the date file of edges
#2.  reads the file as text strings but
#    then converts them to integers
#3.  saves the vertices in an edge as a list
#4.  creates an empirical network by repeatedly
#    adding the edges to DataGraph

def CreateDataGraph():
    DataGraph=Graph()
    infile=open("DataCopy.txt","r")
    for line in infile:
```

```
            Found=False
            length=len(line)-1
            j=0
            while Found==False:
                if line[j]==" ":
                    breakindex=j
                    Found=True
                else:
                    j=j+1
            V1=line[:j]
            V2=line[j+1:]
            V1=int(V1)
            V2=int(V2)
            edge=[V1,V2]

            DataGraph.addEdge(edge)
        return DataGraph

#gives a random integer to be used as the
#index for our random vertex
#hence essentially produces a random vertex
def RandomVertex():
    randVtx=randrange(1,49277)
    return randVtx
#takes a graph object, a random vertex, and
#a given number of vertices to
#create a vertex sample of the network
```

```
#centered around the random vertex
#and its d-neighborhood with the smallest
#d such that the subgraph will have
#at least n vertices

def VertexSample(graphObject,V_0,n):
    SampleVertexSet=[]
    PrimaryVertex=graphObject.graph.keys()[V_0]
    SampleVertexSet.append(PrimaryVertex)
    proxNBHD=graphObject.graph.get(PrimaryVertex,False)
    SampleVertexSet2=SampleVertexSet
    while len(SampleVertexSet)<n:
        NBHD=proxNBHD
        proxNBHD=[]
        for neighbor in NBHD:
            SampleVertexSet2.append(neighbor)
        for neighbor in NBHD:
            temp=graphObject.graph.get(neighbor,False)
            for vertex in temp:
                if vertex not in SampleVertexSet2:
                    if vertex not in proxNBHD:
                        proxNBHD.append(vertex)
        SampleVertexSet=SampleVertexSet2
    return SampleVertexSet

#returns the induced subgraph of the network
#on the vertex sample produced from
```

```
#VertexSample()

def InduceSample(graphObject,VertexList):
    SampleGraph=Graph()
    for vertex in VertexList:
        nbhd=graphObject.graph.get(vertex,False)
        for neighbor in nbhd:
            if neighbor in VertexList:
                j=VertexList.index(vertex)
                if VertexList.index(neighbor)>j:
                    SampleGraph.addEdge([vertex,neighbor])
    return SampleGraph

#returns the degree sequence of a graph
def DegreeSequence(graphObject):
    dsequ=[]
    nbhds=graphObject.graph.values()
    for item in nbhds:
        length=len(item)
        dsequ.append(length)
    dsequ.sort()
    dsequ.reverse()
    return dsequ

#given a graphical degree sequence, returns
#a realization of this sequence
def HavelHakimi(sequ):
```

```
    ResVer=[]
    HHGraph=Graph()
    length=len(sequ)
    for i in range(length):
        ResVer.append([i,sequ[i]])
    ResVer2=ResVer
    while ResVer2[0][1] !=0:
        deg=ResVer[0][1]
        for j in range(1,deg+1):
            edge=[ResVer[0][0],ResVer[j][0]]
            HHGraph.addEdge(edge)
            ResVer[j][1]=ResVer[j][1]-1
        ResVer.pop(0)
        ResVer.sort(key=lambda x:x[1])
        ResVer.reverse()
        ResVer2=ResVer
    return HHGraph


#randomly picks two edges.  if the edges permit a
#HH switch, it performs the switch and returns
#True and the new graph.  if it does not permit the
#switch,then it returns the same graph and False
def MCMCStep(GraphObject):
    GoodStep=True
    vtcs=[]
    edgeset=GraphObject.edges
    length=len(edgeset)
```

```
R1=randrange(length)
R2=randrange(length)
E1=edgeset[R1]
E2=edgeset[R2]
V10=E1[0]
V11=E1[1]
V20=E2[0]
V21=E2[1]
vtcs.append(V10)
vtcs.append(V11)
vtcs.append(V20)
vtcs.append(V21)
for item in vtcs:
    if vtcs.count(item)>1:
        GoodStep=False
nbhd10=GraphObject.graph.get(V10,False)
nbhd11=GraphObject.graph.get(V11,False)
nbhd20=GraphObject.graph.get(V20,False)
nbhd21=GraphObject.graph.get(V21,False)
newgraph=GraphObject
if GoodStep==True:
    if V20 not in nbhd10 and V21 not in nbhd11:
        if V21 not in nbhd10 and V20 not in nbhd11:
            select=randrange(2)
            if select==0:
                T=E1
                P=E2
```

```
                    j=[V20,V10]
                    k=[V21,V11]
                    newgraph=MCMCSwitch(GraphObject,T,P,j,k)
            if select==1:
                    T=E1
                    P=E2
                    j=[V20,V11]
                    k=[V21,V10]
                    newgraph=MCMCSwitch(GraphObject,T,P,j,k)

            else:
                j=[V20,V10]
                k=[V21,V11]
                newgraph=MCMCSwitch(GraphObject,E1,E2,j,k)
        elif V21 not in nbhd10 and V20 not in nbhd11:
            j=[V20,V11]
            k=[V21,V10]
            newgraph=MCMCSwitch(GraphObject,E1,E2,j,k)
        else:
            GoodStep=False
    return newgraph,GoodStep


#function that performs the physical switch.
#called in MCMCStep as a function
def MCMCSwitch(GraphObject,E1,E2,newE1,newE2):
    GraphObject.deleteEdge(E1)
```

```
    GraphObject.deleteEdge(E2)
    GraphObject.addEdge(newE1)
    GraphObject.addEdge(newE2)
    return GraphObject

#calls MCMCStep n^2 "good" times to perform
#the walk.  we mean "good" in that
#an actual switch is performed.  if a
#switch is not performed, then the index
#does not iterate and thus does not contribute
#to the walk at all.Begins by creating a new
#graph instance in newgraph that duplicates the given
#graphObject.  it then walks along this copy.
#this is done so that the original graphObject
#is not altered and thus we are left with the original and
#final step in the walk
def MCMCWalk(GraphObject,textfile):
    N=GraphObject.order
    i=0
    newgraph=Graph()
    for edge in GraphObject.edges:
        newgraph.addEdge(edge)
    while i<N*N:
        switch=MCMCStep(newgraph)
        if switch[1] == True:
            newgraph=switch[0]
            i=i+1
```

```
    Laplace(newgraph,textfile)
    return newgraph


#First creates a graph according to the Networkx format.
#Computes the Laplacian Matrix of the graph
#Saves the Laplacian to a text file that will then be used
#by MatLab to compute the eigenvalues
def Laplace(GraphObject,textfile):
    G=nx.Graph()
    order=GraphObject.order
    for edge in GraphObject.edges:
        V1=edge[0]
        V2=edge[1]
        G.add_edge(V1,V2)
    laplace=nx.linalg.laplacianmatrix.laplacian_matrix(G)
    outfile=open(textfile,"a")
    #outfile.write("M\n")
    #outfile.write(str(order)+"\n")
    outfile.write(str(order)+'\n')
    for row in laplace:
        for entry in row:
            string=str(entry)
            outfile.write(string)
            outfile.write("\n")
    #outfile.write("M\n")
    outfile.close()
    return laplace
```

```
#The main control of the program
#Performs the complete program as described in the paper
def main():
    #data=CreateDataGraph()
    #V=RandomVertex()
    #vertexsample= VertexSample(data,V,100)
    #inducedgraph=InduceSample(data,vertexsample)
    #sequ=DegreeSequence(inducedgraph)
    infile=open("DegreeSequenceC.txt","r")
    #infile.write(str(len(sequ))+'\n')
    #for degree in sequ:
    #    infile.write(str(degree)+'\n')
    sequ=[]
    for line in infile:
        deg=int(line)
        sequ.append(deg)
    infile.close()
    HH=HavelHakimi(sequ)
    #Laplace(inducedgraph,"E:\Matrices\SampleMatrixC.txt")
    for j in range(100000):
        MCMCWalk(HH,"E:\Matrices\LapMatrices1C.txt")
        if j%1000==0:
            print j
    for j in range(100000):
        MCMCWalk(HH,"E:\Matrices\LapMatrices2C.txt")
        if j%1000==0:
```

```
            print j
    for j in range(100000):
        MCMCWalk(HH,"E:\Matrices\LapMatrices3C.txt")
        if j%1000==0:
            print j
    for j in range(100000):
        MCMCWalk(HH,"E:\Matrices\LapMatrices4C.txt")
        if j%1000==0:
            print j


main()
```

Appendix B

Spectral Data

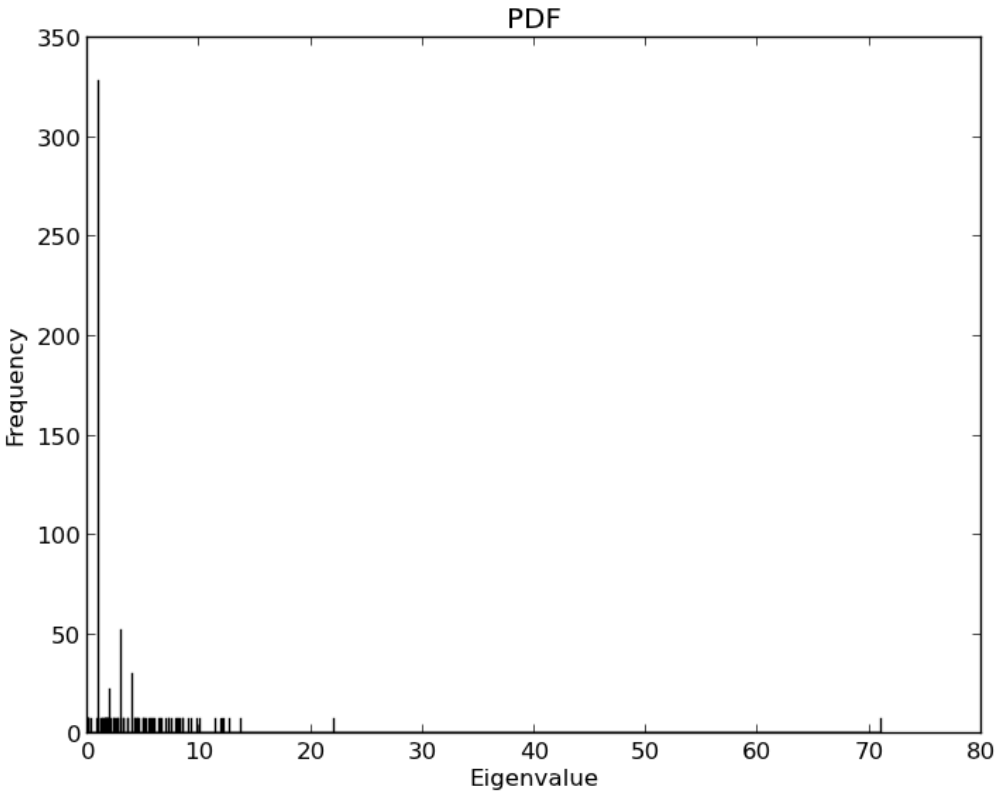


Figure B.1 Spectrum of Network Sample 1

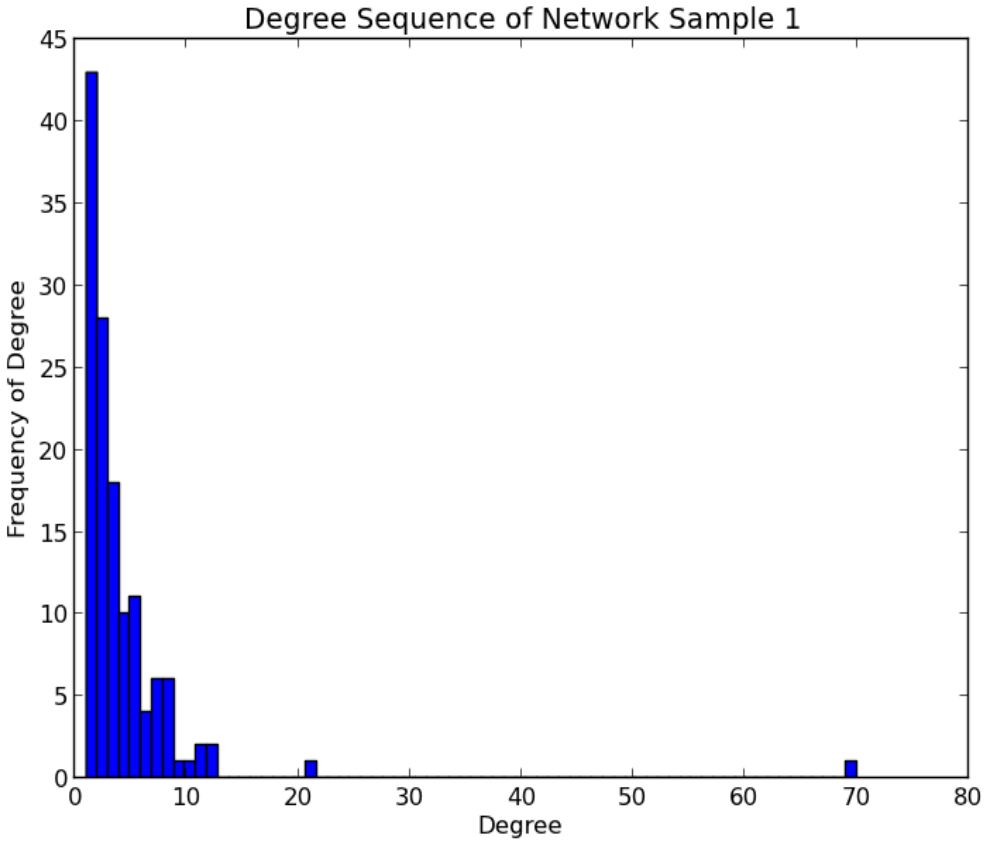


Figure B.2 Degree Multiplicity of Degree Sequence of Network Sample 1

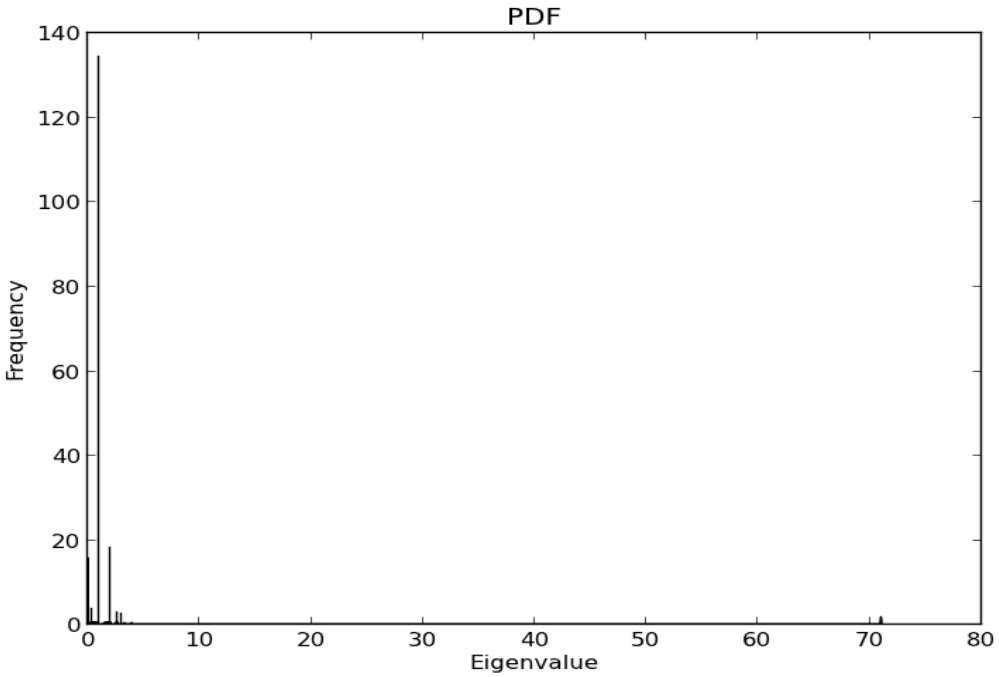


Figure B.3 Spectrum of MCMC Generated Graphs with Degree Sequence of Network Sample 1

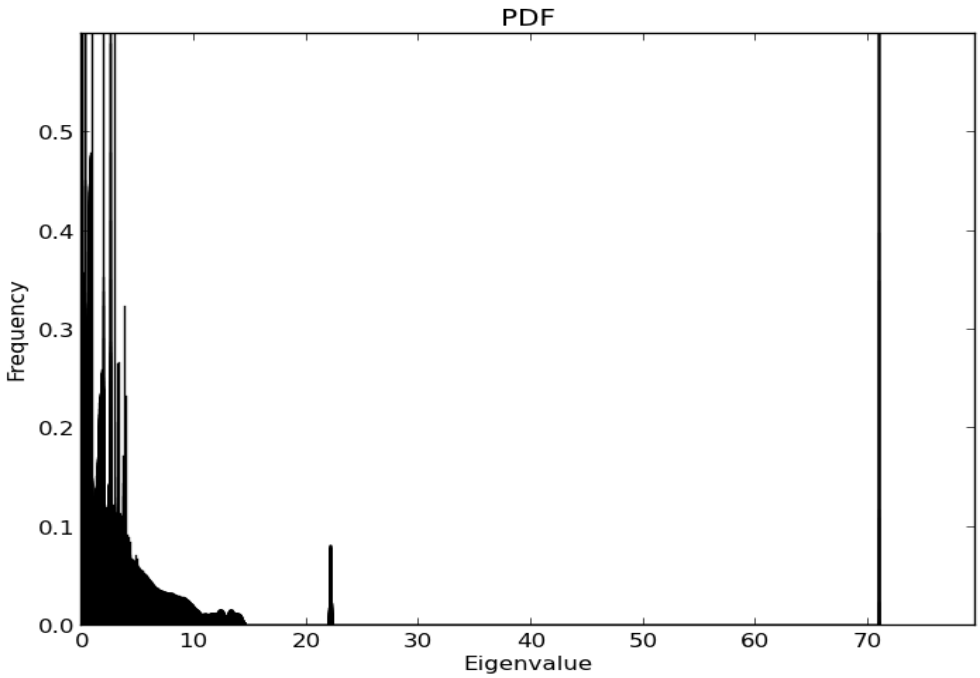


Figure B.4 Zoomed in view of Figure B.3

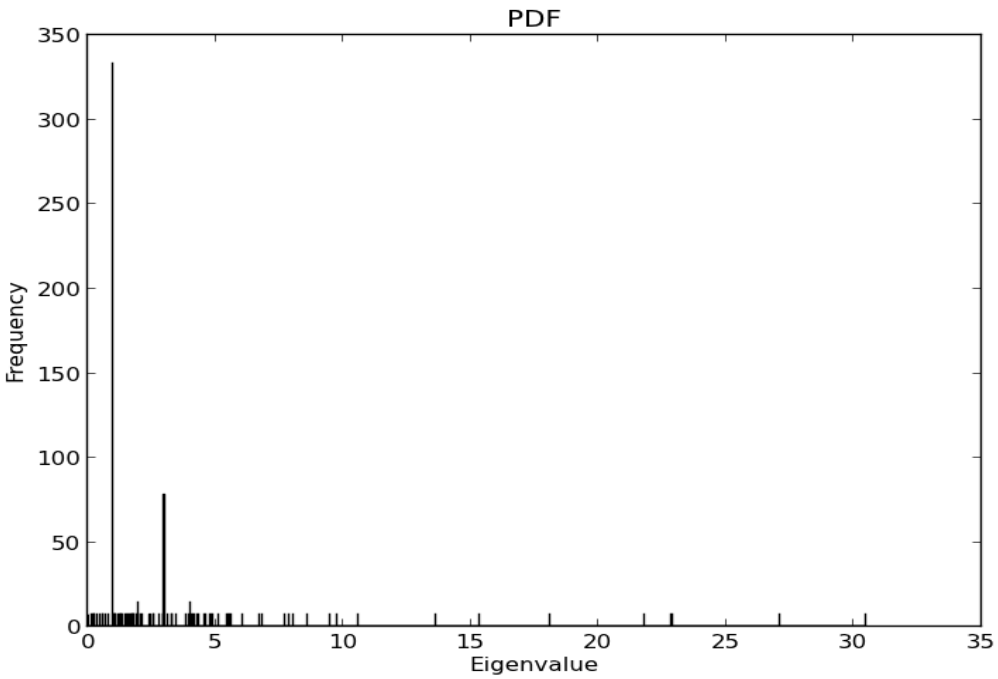


Figure B.5 Spectrum of Network Sample 2

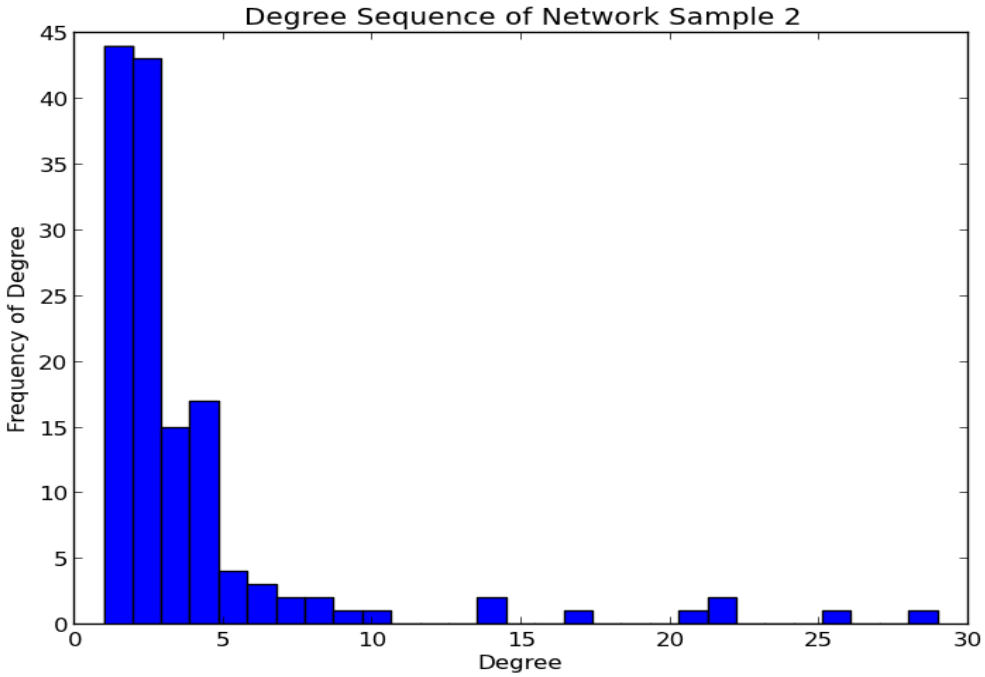


Figure B.6 Degree Multiplicity of Degree Sequence of Network Sample 2

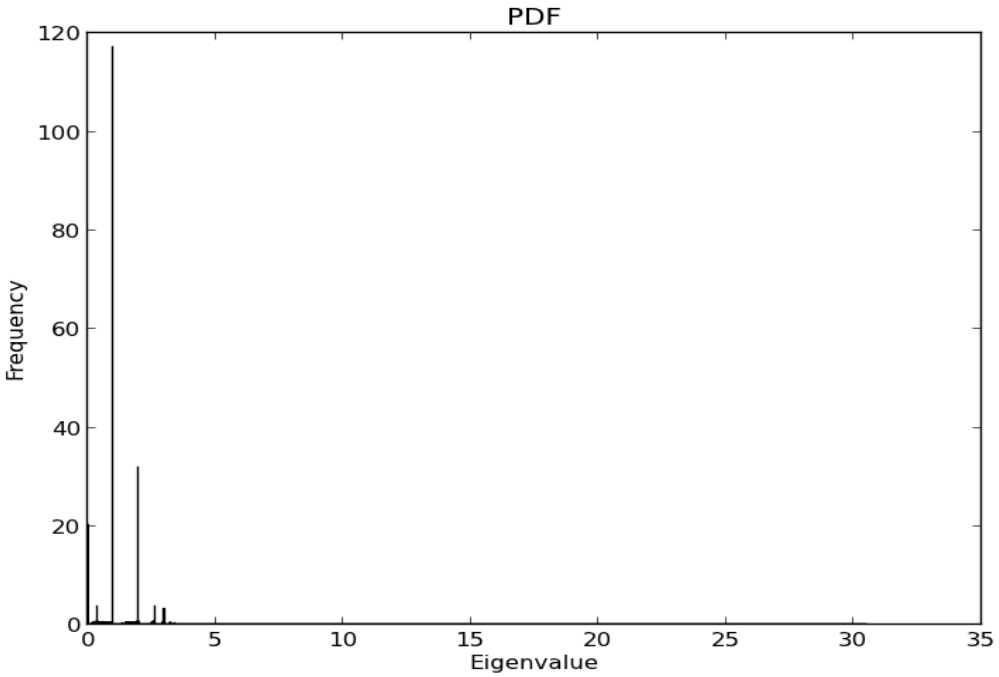


Figure B.7 Spectrum of MCMC Generated Graphs with Degree Sequence of Network Sample 2

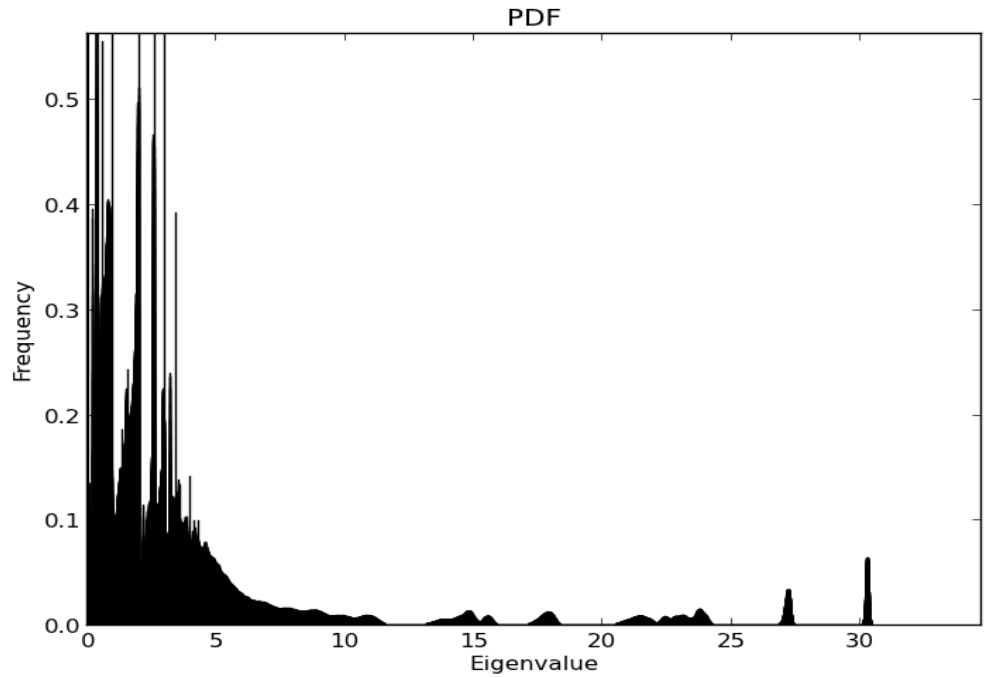


Figure B.8 Zoomed in view of Figure B.7

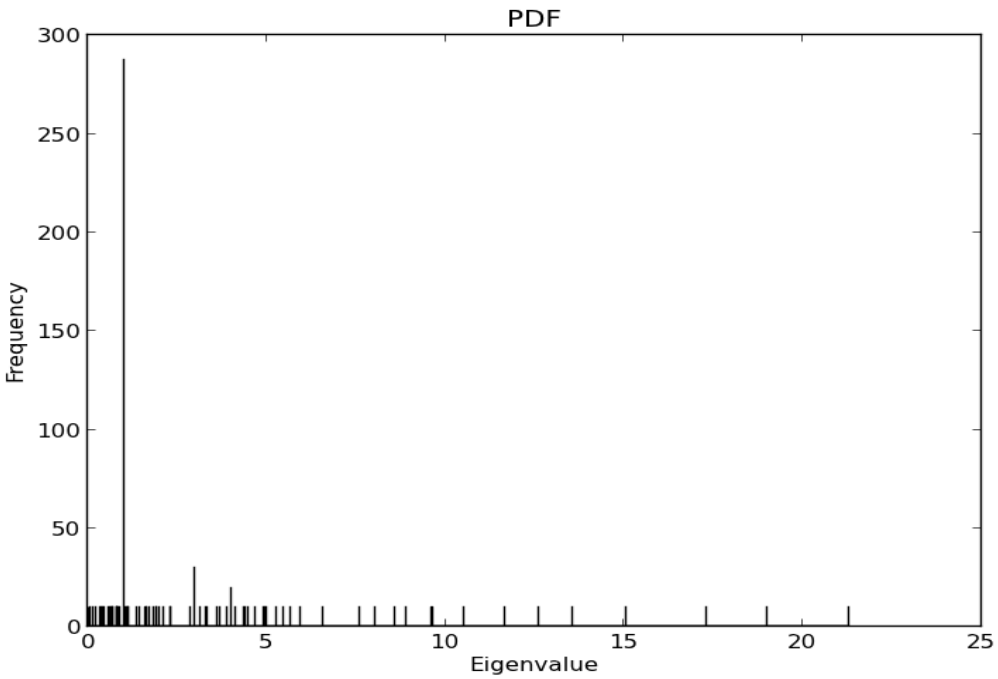


Figure B.9 Spectrum of Network Sample 3

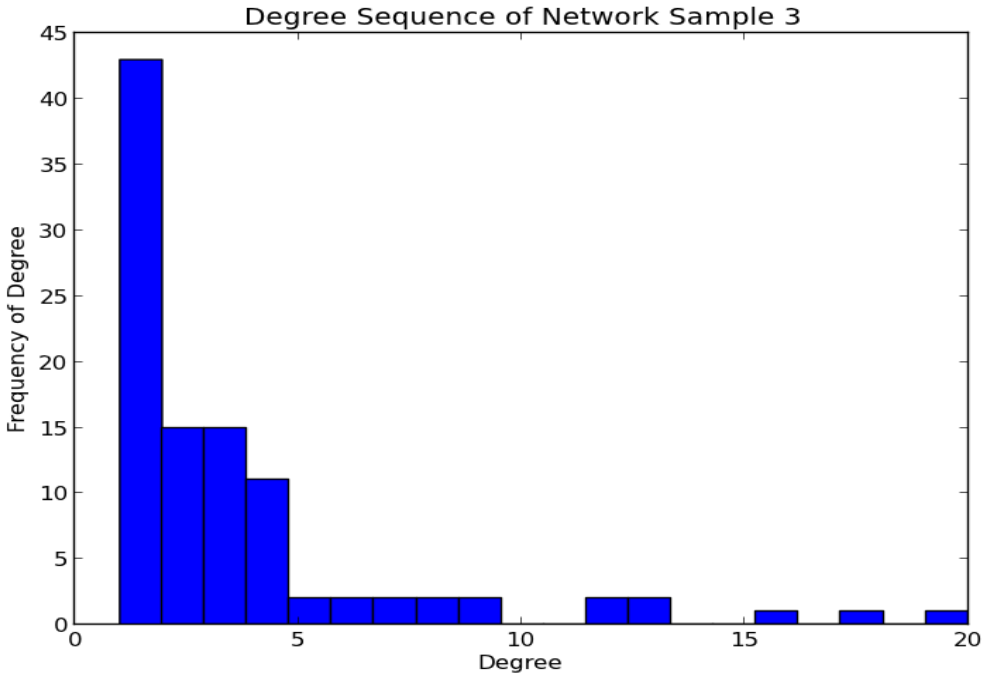


Figure B.10 Degree Multiplicity of Degree Sequence of Network Sample 3

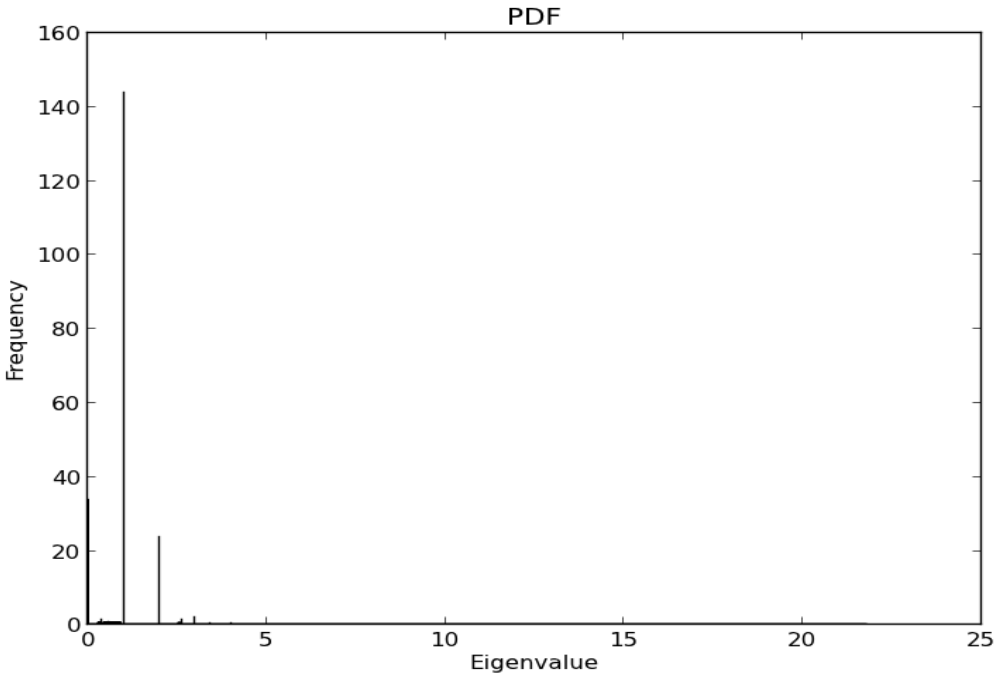


Figure B.11 Spectrum of MCMC Generated Graphs with Degree Sequence of Network Sample 3

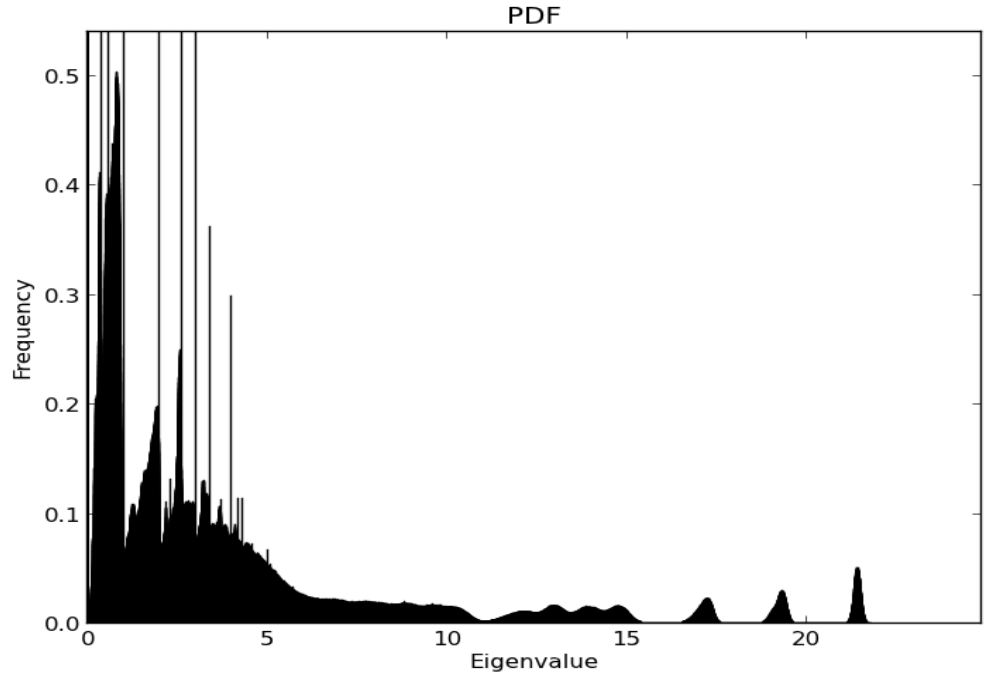


Figure B.12 Zoomed in view of Figure B.11

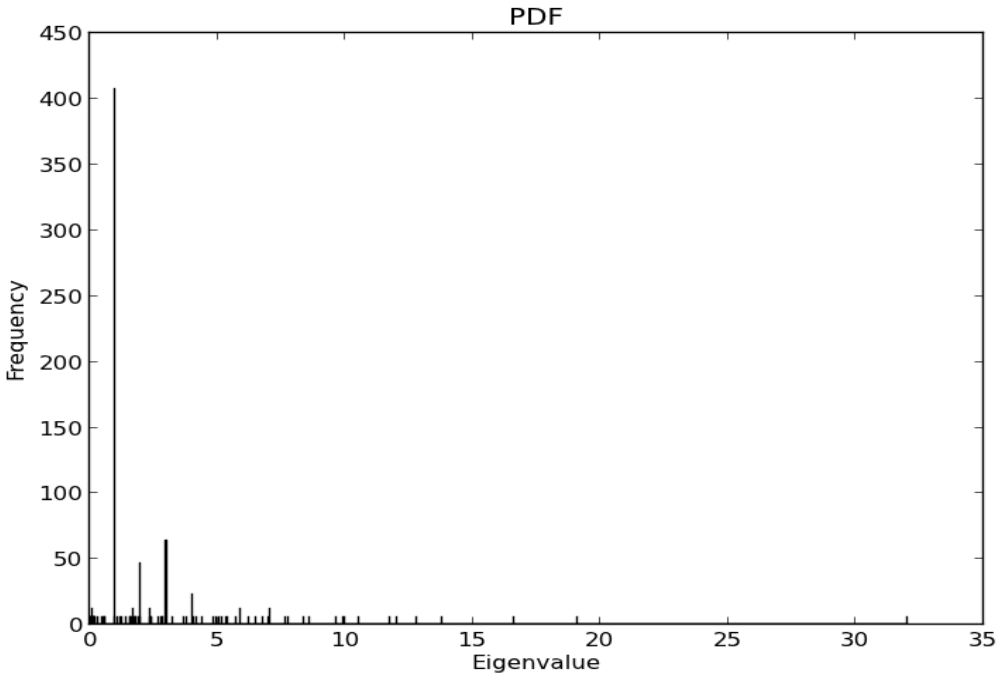


Figure B.13 Spectrum of Network Sample 4

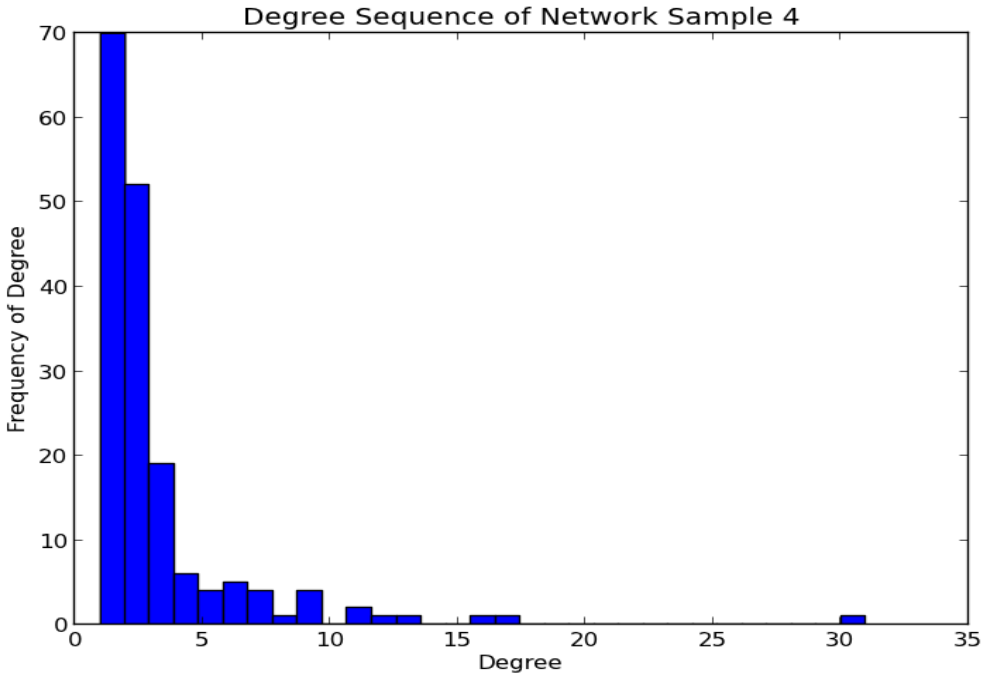


Figure B.14 Degree Multiplicity of Degree Sequence of Network Sample 4

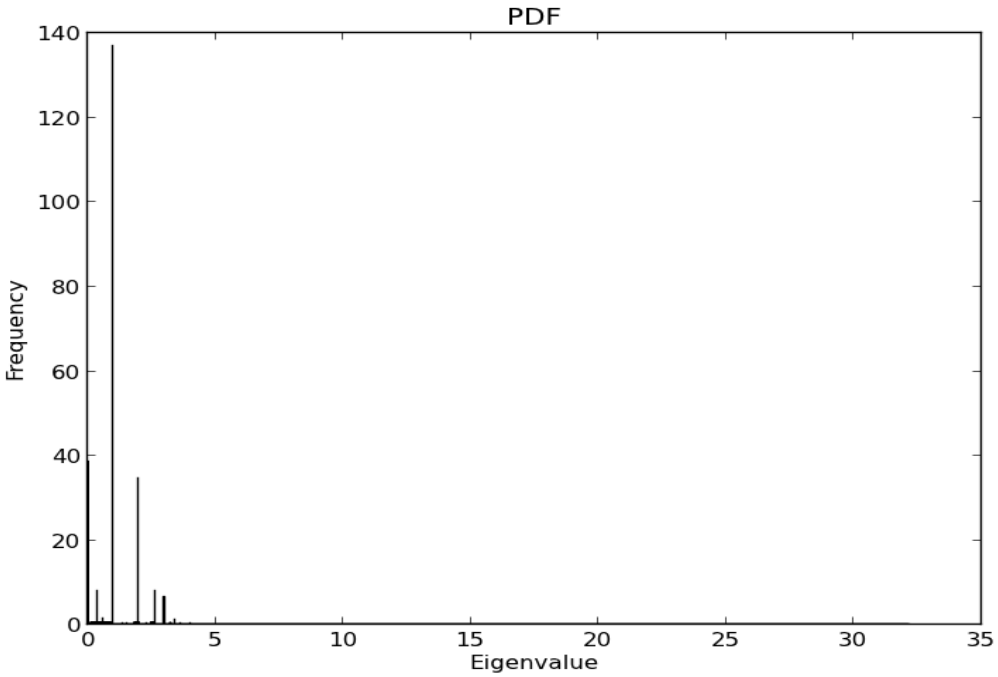


Figure B.15 Spectrum of MCMC Generated Graphs with Degree Sequence of Network Sample 4

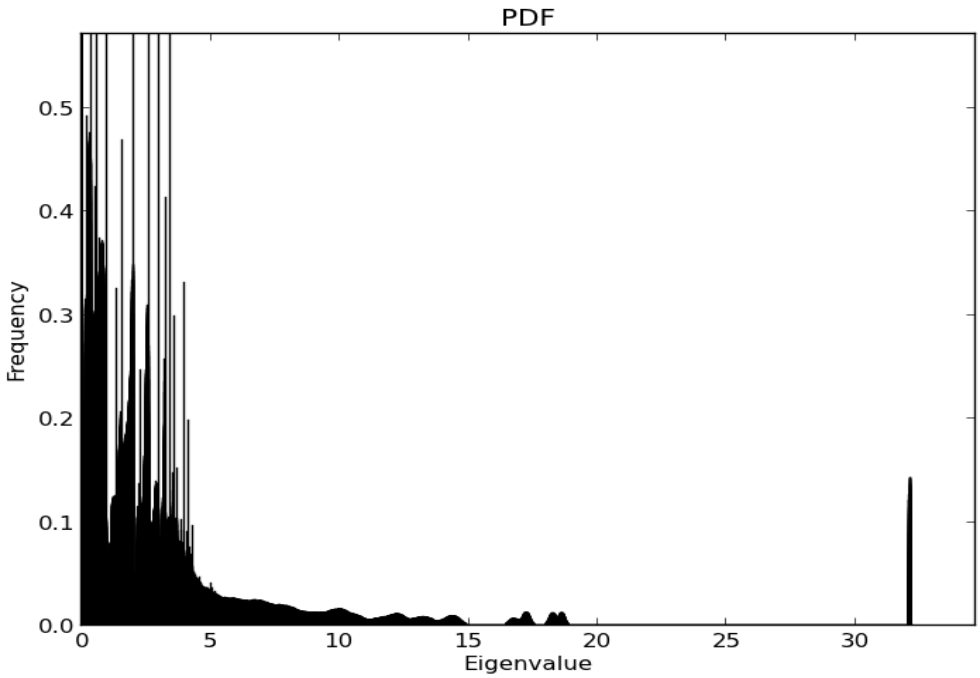


Figure B.16 Zoomed in view of Figure B.15