Department of Computer Science &
Engineering, IIT Kharagpur

Lecture Notes

Complex Network Theory
CS60078

Rishabh Poddar
08CS1044
Mayank Shrivastava
08CS3029

Lecturer:
Prof. Animesh Mukherjee

Contents

Chapter 1

Introduction

This material contains lecture notes for the course on **Complex Network Theory**, Spring 2012, Indian Institute of Technology, Kharagpur. The course was taught by Prof. Animesh Mukherjee (IIT Kharagpur), and Prof. Sitabhra Sinha (IMS Chennai).

Chapter 2

Basics of Graph Theory

In this section, we will revise some elementary concepts from graph theory, which will prove useful in our study of this course.

2.1 The Seven Bridges of Königsberg

The Seven Bridges of Königsberg is a historically notable problem in mathematics. The city of Königsberg in Prussia (now Kaliningrad, Russia) was set on both sides of the Pregel River, and included two large islands which were connected to each other and the mainland by seven bridges. The problem was to find a walk through the city that would cross each bridge once and only once. The islands could not be reached by any route other than the bridges, and every bridge must have been crossed completely every time; one could not walk halfway onto the bridge and then turn around and later cross the other half from the other side. Euler proved that the problem has no solution. There could be no non-retracing continuous curve that passed through all seven of the bridges. The difficulty was the development of a technique of analysis and of subsequent tests that established this assertion with mathematical rigour.

Some practical applications of his problem include - cost of wiring electrical components, shortest route between cities, matching/resource allocation, task scheduling, and floor planning and routing (VLSI).

2.2 Set-Theoretic Definition of Graphs

A graph G consists of an ordered tuple $G = (V, E)$, where V is a set of nodes, points, or vertices; E is a set whose elements are known as edges or lines. $E \subseteq V \times V$. Note that if E equals $V \times V$, then the graph is complete.

2.3 Representation of Graphs

A graph is generally represented in the following two ways: **adjacency matrix** and **adjacency list**.

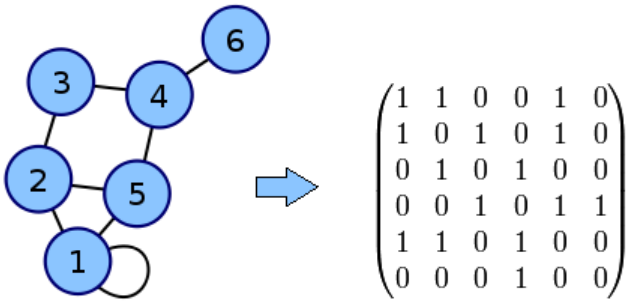


Figure 2.1: Adjacency matrix representation

Some properties of an adjacency matrix A are as follows. $A = \{a_{ij}\}$, where i and j are nodes, and $a_{ij} = 1$ if there is an edge between i and j, else it is 0. The entries of the matrix A^2 denote the number of paths of length 2 between nodes in the graph. Similarly, entries of A^n denotes the number of paths of length n. Note that the trace (sum of the diagonal elements) of the matrix A^3 is equal to six times the number of triangles in the graph.

2.4 Paths, Walks and Trails

A **path** in a graph is a single vertex or an ordered list of distinct vertices $v_1, \cdots, v_k$ such that $v_{i-1}v_i$ is an edge for all $2 \leq i \leq k$. No vertex may be repeated.

A **walk** of length k is a sequence $v_0, v_1, \cdots, v_k$ of vertices and edges such that (v_{i-1}, v_i) is an edge for $1 \leq i \leq k$.

A **trail** is a walk with no repeated edge.

2.5 Components of a Graph

Let $G = (V, E)$ be an undirected graph. We call G connected if there exists a path between any two distinct vertices of G.

A **strongly connected** directed graph is one where each node belonging to the graph can be reached from every other node via directed paths. A **weakly connected** directed graph is one where each node belonging to the graph can be reached from every other node, disregarding edge directions.

2.6 Complete and Complement Graphs

A complete graph is one in which an edge exists between any 2 vertices, that is, all the entries in the adjancency matrix are 1. A complete graph with n vertices is denoted as

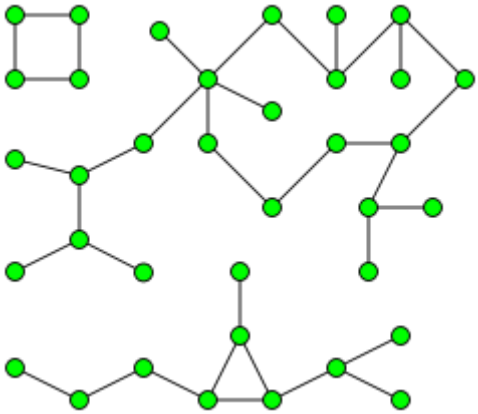


Figure 2.2: A graph with 3 components

K_n.

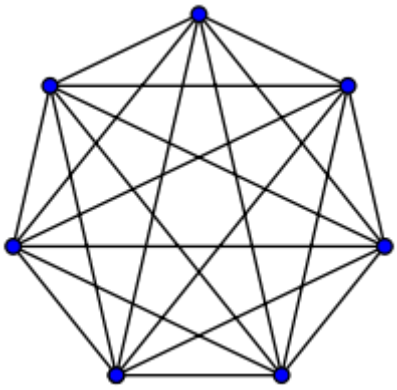


Figure 2.3: K_7, a complete graph with 7 vertices

The complement graph G' of a graph G is a graph such that $V(G') = V(G)$, and an edge exists between 2 nodes v_i, v_j in G' if there exists no edge between them in G.

2.7 Sparse and Dense Graphs

A graph $G(V, E)$ is called **sparse** if $|E| \approx |V|$; and it is called **dense** if $|E| \approx |V|^2$.

2.8 Regular Graphs and Lattices

A **regular** graph is one in which all the nodes have the same **degree**, that is, the same number of edges emanating from the node.
A **lattice** is a regular graph with vertices coupled to their k nearest neighbours.

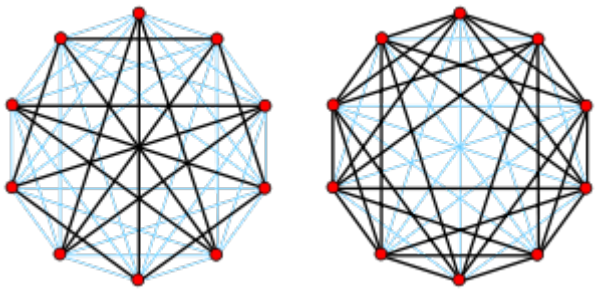


Figure 2.4: The Petersen graph on the left, and its complement graph on the right

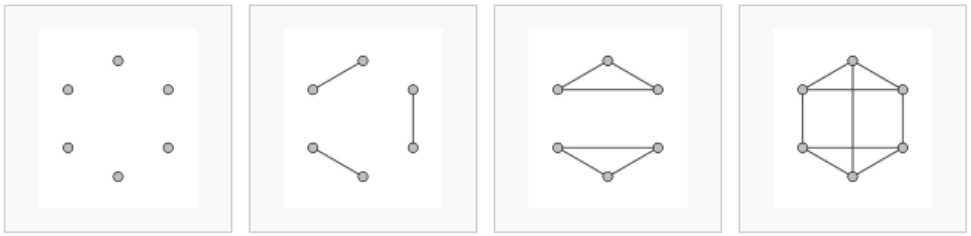


Figure 2.5: 0-regular, 1-regular, 2-regular and 3-regular graphs

2.9 Planar Graphs

A graph G is called planar if G can be drawn in the plane with its edges intersecting only at vertices of G. Such a drawing of G is called an **embedding** of G in the plane. Note that of all complete graphs K_n, only K_1, K_2, K_3 and K_4 are planar.

2.10 Geodesics

A **geodesic** from vertex a to vertex b is a path of minimum length between the nodes. The length of this path is called the **geodesic distance** between a and b.

The eccentricity of a vertex v is the greatest geodesic distance between v and any other vertex. The largest eccentricity of any vertex in the graph is called the **diameter** (d) of the graph. The **radius** (r) of a graph is the minimum eccentricity of any vertex.

A **central** vertex in a graph of radius r is one whose distance from every other vertex in the graph is at most r.

A **peripheral** vertex in a graph of diameter d is one that is at a distance d from some other vertex in the graph.

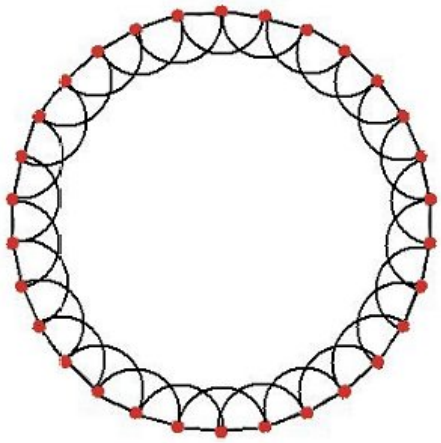


Figure 2.6: Regular lattice - each node is linked to its four immediate neighbours

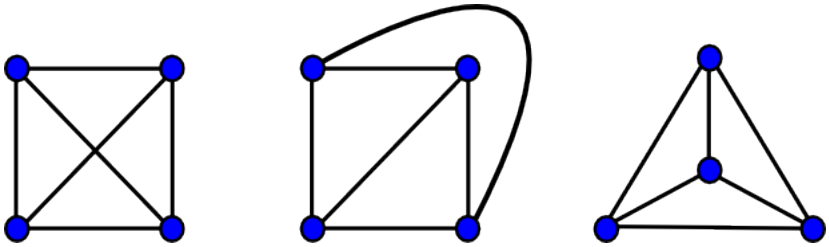


Figure 2.7: The graph K_4 (extreme left) and its planar embeddings

2.11 Average Path Length

The average path length l is defined as the average of the shortest paths between all nodes in the network, i.e.,

$$l = \langle d_{ij} \rangle = \frac{1}{N(N-1)} \sum_{i \neq j} d_{ij}$$

If the graph is disconnected, it makes sense to consider the reciprocal of the harmonic mean; this is because the distance between two nodes belonging to separate components is infinite, the reciprocal being 0.

$$l = \langle \frac{1}{d_{ij}^{-1}} \rangle = \left(\frac{1}{N(N-1)} \sum_{i \neq j} \frac{1}{d_{ij}} \right)^{-1}$$

2.12 Articulation Points

An **articulation point** or **cut point** is a vertex whose removal increases the number of components in the graph. Such points are called **brokers** in social networks. Removal of brokers creates communities that are totally isolated from each other.

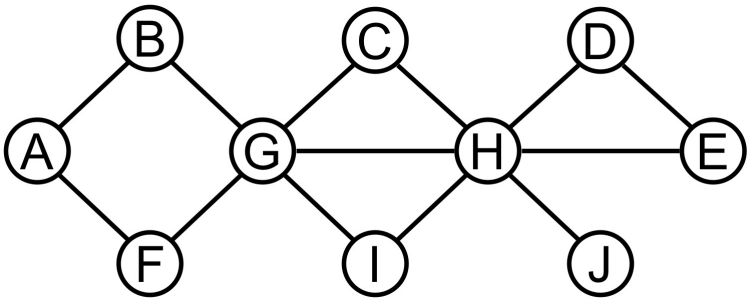


Figure 2.8: Nodes G and H are articulation points

2.13 Bridges

An edge is called a **bridge** if its removal increases the number of components in the graph.

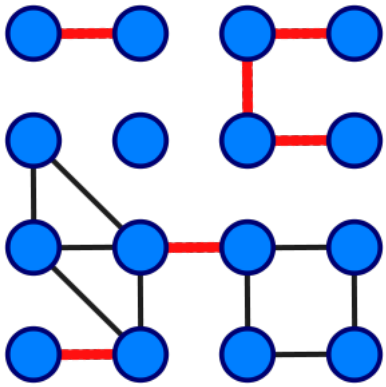


Figure 2.9: A graph with 6 bridges (highlighted in red)

2.14 Connection Density

The **connection density** in a graph is a metric used to estimate the number of connections in the graph. It is defined as the ratio of the number of edges actually present in the graph and the maximum number of edges possible.

$$cd = \frac{|E|}{\binom{n}{2}} = \frac{2|E|}{N(N-1)}$$

An alternate interpretation of this metric can be the probability that an edge exists between a randomly chosen pair of vertices.

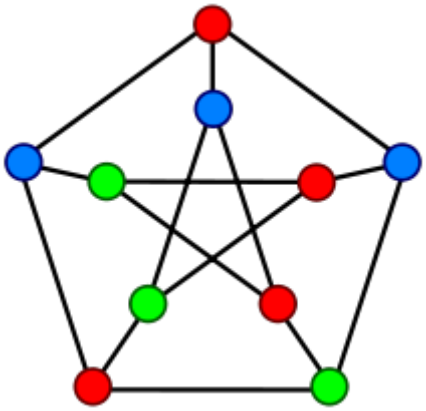


Figure 2.10: A 3-coloring for the Petersen graph

2.15 Chromatic Number

A **proper colouring** of a graph is an assignment of labels to each vertex of the graph such that no two adjacent vertices receive the same label. The **chromatic number** of a graph is the minimum number of colours required to achieve a proper colouring. This concept finds applications in problems such as job scheduling and register allocation among others.

2.16 Chordal Graphs

A graph is **chordal** if each of its cycles of four or more nodes has a chord, which is an edge joining two nodes that are not adjacent in the cycle.

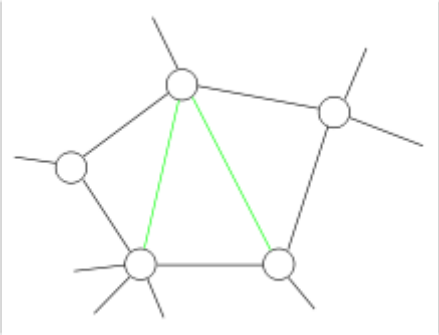


Figure 2.11: A cycle (black) with two chords (green)

Chapter 3

Basic Metrics for Network Analysis

In this section, we describe some metrics that are useful for statistical analysis of complex networks.

3.1 Degree Distribution

We shall describe the concept of degree distribution with the help of an example. Consider the case of **citation networks**. Scientific papers refer to works done earlier on related topics via **citations**. In a citation network each node represents a scientific paper and a directed edge from node A to B indicates that A has cited B. An important thing to note is that citation networks are acyclic in nature.

Alfred Lotka analysed such networks in 1926. **Lotka's Law** describes the frequency of publication by authors in any given field. It states that the number of authors making n contributions to that field is approximately $n^{-\alpha}$ of those making 1 contribution, where $\alpha \approx 2$. This distribution is nothing but the distribution of the degrees of the nodes in the network.

A famous outcome of Lotka's study was the $80 - 20$ Rule, which states that 80% of the people in such a network are 20% popular, and vice versa.

3.1.1 Definition

Let p_k be the fraction of vertices in a network that have degree k. p_k can also be interpreted as the probability that a vertex chosen uniformly at random has degree k. Then, the p_k versus k plot is defined as the **degree distribution** of the network. For most real world networks, p_k varies as $k^{-\alpha}$, which is the case for citation networks.

3.1.2 Cumulative Degree Distribution

Due to noise and insufficient data the definition of degree distribution is slightly modified at times. Instead, we use the **cumulative degree distribution**, which is plotted as P_k versus k, where

$$P_k = \sum_{k'=k}^{\infty} p_{k'}$$

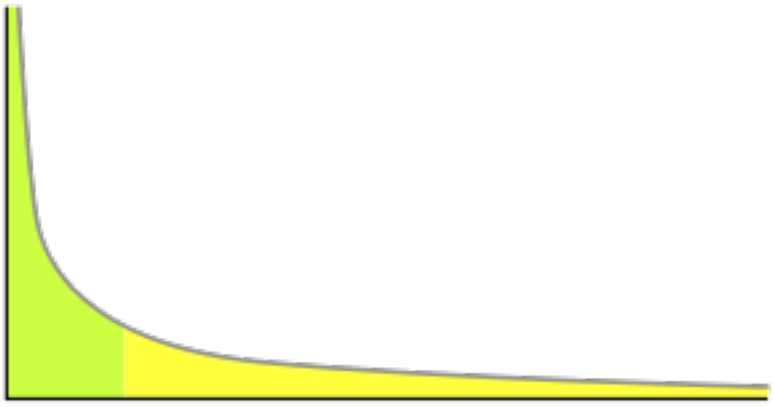


Figure 3.1: An example power-law graph, being used to demonstrate ranking of popularity. To the right is the long tail, and to the left are the few that dominate (also known as the 80-20 rule)

for discrete distributions, while for continuous distributions we have

$$P_k = \int_{k'=k}^{\infty} p_{k'} dk'$$

So, P_k can also be interpreted as the probability that the degree of a node selected uniformly at random is greater than or equal to k.

3.1.3 Scale-Free Functions

A scale-free function $f(x)$is one in which the independent variable x when rescaled does not affect the functional form of the original function. Mathematically,

$$f(ax) = bf(x)$$

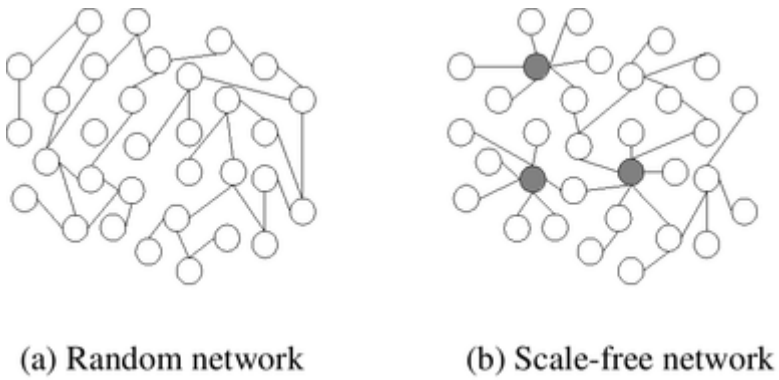


Figure 3.2: Random versus Scale-free network

Power Laws are scale-free functions, that is, at any scale, they still show power law behaviour. Other examples where such behaviour is manifested include fractals.

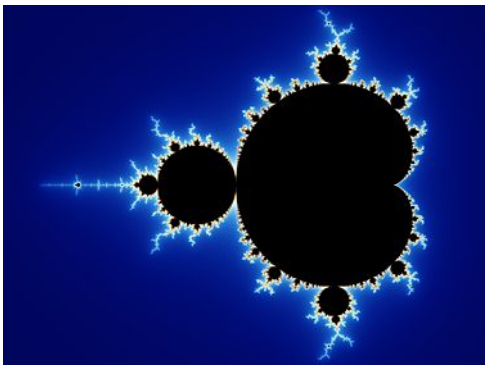


Figure 3.3: The Mandelbrot set is a famous example of a fractal

3.2 Transitivity

A **transitive** network is one in which for any 3 nodes a, b and c, if there exists an edge between a and b, and between b and c, then there exists an edge between a and c as well.

3.2.1 Measuring Transitivity: Clustering Coefficient

The **clustering coefficient** for a vertex v in a network is defined as the ratio between the total number of connections among the neighbors of v to the total number of possible connections between the neighbours. Mathematically,

$$C_v = \frac{L}{\binom{n}{2}}$$

where $L =$ the number of actual links between the neighbours of v, and $n =$ the number of neighbours of v.

The **clustering index** of the whole network is the average of the clustering coefficients of all the vertices. That is,

$$C = \frac{1}{N} \sum C_v$$

Note that higher the clustering index, larger the number of triangles in the network.

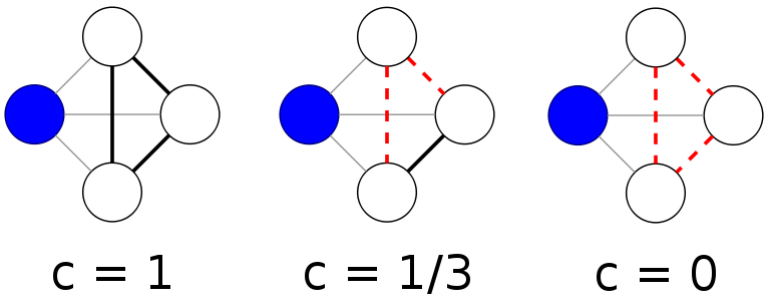


Figure 3.4: Local clustering coefficient values

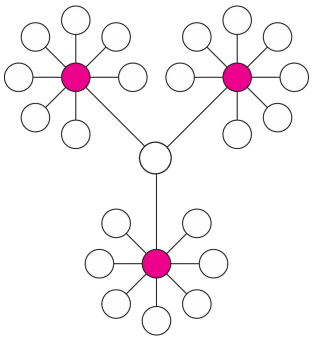


Figure 3.5: The most important vertices according to degree-centrality (red)

3.3 Centrality

Centrality is a measure indicating the *importance* of node in the network. Commonly, it measures the 4 P's - prestige, prominence, (im)portance and power. We wll get a better idea of what is meant by importance as the section progresses.

3.3.1 Degree Centrality

Degree centrality is defined as the ratio of the number of neighbours of a vertex with the total number of neighbours possible. Mathematically,

$$\text{Degree Centrality} = \frac{k}{N-1}$$

where k is the degree of the vertex, and N is the total number of nodes in the network.

The variance of the distribution of degree centrality in a network gives us the **centralization** of the network. One can see that a *star network* is an ideal centralized network, whereas a *line network* is less centralized.

Figure 3.6: Star Network and Line Network

3.3.2 Betweenness Centrality

The degree of a node is not the only measure of the importance of a node in the network, and this centrality measure addresses this fact. This concept was introduced by Linton Freeman. In his conception, vertices that have a high probability of occuring on a randomly chosen shortest path between two nodes are said to have high **betweenness centrality**.

Formally, centrality of a vertex v is defined as the summation of the geodesic path between

any two nodes s and t via v, expressed as a fraction of the total number of geodesic paths between s and t. Mathematically,

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

If v is an articulation point, then we can further simplify this as follows. Let the two components that the removal of v divides the graph into be C_1 and C_2 with N_1 and N_2 nodes respectively. Then,

$$\begin{aligned} g(v) &= 2 \sum_{s \in C_1, t \in C_2} \frac{\sigma_{st}(v)}{\sigma_{st}} \\ &= 2 \sum_{s \in C_1, t \in C_2} 1 \\ &= 2N_1N_2 \end{aligned}$$

Removal of a node with high betweenness centrality can lead to increase in the geodesic path lengths, and in the extreme case, the network might even get disconnected as exhibited in the case above. In real world networks, this can be important; for example, to prevent the spread of a disease in an epidemic network.

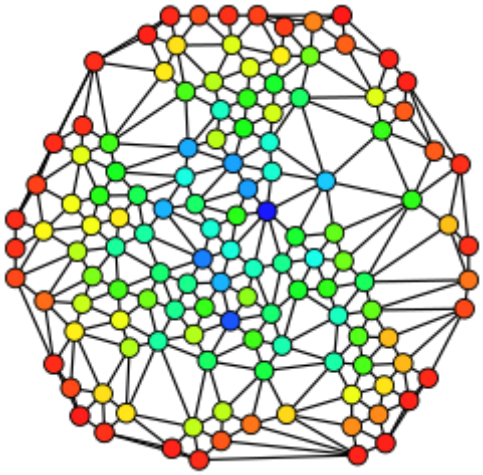


Figure 3.7: Hue (from red=0 to blue=max) shows the node betweenness.

3.3.3 Flow Betweenness

Suppose two nodes are connected by a **reluctant broker** (cut vertex), that is, the shortest path between them is blocked. Then, the nodes should use another pathway which is connecting them, rather than simply using the geodesic path.

The **flow betweenness** measure thus expands the notion of betweenness centrality. It assumes that any two nodes would use all the paths connecting them, instead of only using shortest path. However, it is to be noted that calculating flow betweenness is computationally intractable.

3.3.4 Eigenvector Centrality

This metric assigns relative scores to all nodes in the network based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. For example, consider the context of HIV transmission. A person x with one sexual partner is seemingly less prone to the disease than a person y with multiple partners. However, we must also take into account the number of partners that the sexual partner of x has. That is, it is not enough to merely gauge the popularity of a particular node on the basis of its degree; we must also take into account the popularity of its neighbours as well. This is the basic idea of **eigenvector centrality**.

We now proceed to define the centrality value of a vertex as a sum of centralities of its neighbours. To begin with, we initially guess that a vertex i has centrality $x_i(0)$. We gradually improve this estimate by employing a Markov model, and continue in this manner until no more improvement is observed. The improvement made at step t is defined as,

$$\begin{aligned} x_i(t) &= \sum_j A_{ij} x_j(t-1) \\ \Rightarrow x(t) &= \mathbf{A}x(t-1) \\ &= \mathbf{A}^t x(0) \end{aligned}$$

This is known as the Power Iteration method proposed by Hotelling.
Now, express $x(0)$ as a linear combination of eigenvectors v_i of the adjacency matrix $\mathbf{A}$

$$\begin{aligned} x(0) &= \sum_i c_i v_i \\ \Rightarrow x(t) &= \mathbf{A}^t \sum_i c_i v_i \end{aligned}$$

We know from our knowledge of eigenvectors that $\mathbf{A}^t x = \lambda^t x$ holds, where λ is an eigenvalue. Using this with the equation above, we have

$$\begin{aligned} x(t) &= \sum_i \lambda_i^t c_i v_i \\ &= \lambda_1^t \sum_i \left(\frac{\lambda_i}{\lambda_1}\right)^t c_i v_i \\ \Rightarrow \frac{x(t)}{\lambda_1^t} &= \sum_i \left(\frac{\lambda_i}{\lambda_1}\right)^t c_i v_i \end{aligned}$$

In the limit $t \to \infty$, $\left(\frac{\lambda_i}{\lambda_1}\right)^t$ remains only for $i = 1$. Thus,

$$\lim_{t\to\infty} \frac{x(t)}{\lambda_1^t} = c_1 v_1$$

Thus, we get that the limiting centrality is proportional to the principal eigenvector v_1.

Note that directed acyclic networks suffer from the problem of **zero centrality**. If there exists a node A with no incoming edges, then this node has zero centrality (the assumption seems reasonable for a web page). Consider another node B that has one incoming edge from A. Then the eigenvector centrality of B is 0 because A the centrality of A is 0. Hence, in a similar fashion, all the centralities in an acyclic network become 0. We will see how this problem is remedied by the Katz Centrality metric.

3.3.5 Katz Centrality

As discussed previously, eigenvector centrality metric suffers from the problem of zero centrality. **Katz Centrality** resolves this issue by assigning to each node *a priori* some positive centrality value. This is done according to the following equation

$$x_i = \alpha \sum_j A_{ij} x_j + \beta$$

Note that $\alpha, \beta > 0$. In matrix terms, the above equation is equivalent to

$$\mathbf{x} = \alpha \mathbf{A}\mathbf{x} + \beta \mathbf{1}$$

where $\mathbf{1} = (1, 1, \cdots, 1)^T$. On simplifying, we obtain

$$\mathbf{x} = \beta(\mathbf{I} - \alpha \mathbf{A})^{-1}\mathbf{1}$$

Instead of inverting the matrix as above, we can alternatively iterate over the following equation until convergence

$$\mathbf{x}(t) = \alpha \mathbf{A}\mathbf{x}(t-1) + \beta \mathbf{1}$$

3.3.6 PageRank

PageRank is a link analysis algorithm, named after Larry Page and used by the Google Internet search engine, that assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web. The PageRank of a page is defined recursively and depends on the number and PageRank metric of all pages that link to it. A page that is linked to by many pages with high PageRank receives a high rank itself. If there are no links to a web page there is no support for that page.

Simply put, the algorithm can be described as follows. PageRank can be thought of as a probability distribution representing the likelihood that a person randomly clicking on links will arrive at any particular page. The PageRank computations require several passes through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.

Essentially, PageRank is nothing but a variant of Katz Centrality. It can be mathematically expressed as follows.

$$x_i = \alpha \sum_j A_{ij} \frac{x_j}{k_j^{out}} + \beta$$

where k_j^{out} is the out-degree of node j. This normalization is done to obtain a stochastic matrix (a matrix where either all the rows or all the columns sum to one). Note that

the above definition does not take into account the possibility of $k_j^{out} = 0$. To solve this problem, set $k_j^{out} = 1$ in the above calculation, since a vertex with zero out-degree contributes zero to centralities of other vertices. In matrix terms, we have

$$\mathbf{x} = \alpha \mathbf{A}\mathbf{D}^{-\mathbf{1}}\mathbf{x} + \beta \mathbf{1}$$
$$\Rightarrow \mathbf{x} = \beta(\mathbf{I} - \alpha \mathbf{A}\mathbf{D}^{-\mathbf{1}})^{-1}\mathbf{1}$$

where $\mathbf{D}$ is a diagonal matrix such that

$$D_{ii} = \max\{k_i^{out}, 1\}$$

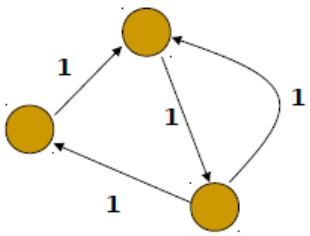
3.3.6.1 Random Walks

A **random walk** is a mathematical formalisation of a trajectory that consists of taking successive random steps. It was introduced by Karl Pearson in 1905.

Random walks are useful to analyze web surfing and to calculate PageRank values. Consider web surfing, initially, every page is chosen uniformly at random. With probability α, the surfer performs random walk by randomly choosing the hyperlinks in that page, and with probability $1 - \alpha$, the surfer stops the random walk. We already know that the steady state probabiliity that a web page is visited during web surfing represents its PageRank.

The transition matrix for web surfing is obtained from the adjacency matrix representing the underlying graph structure. The transition matrix is a stochastic matrix, all rows sum to 1, and is thus obtained by dividing each number in each row by the sum of the elements in that row in the adjacency matrix. Essentially, an entry in the transition matrix represents the probability with which that link is chosen.

As an example, consider the following graph and its equivalent adjacency matrix



$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

For the above graph, the transition matrix is given as,

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

Here, we pictorially show a random walk on this network.

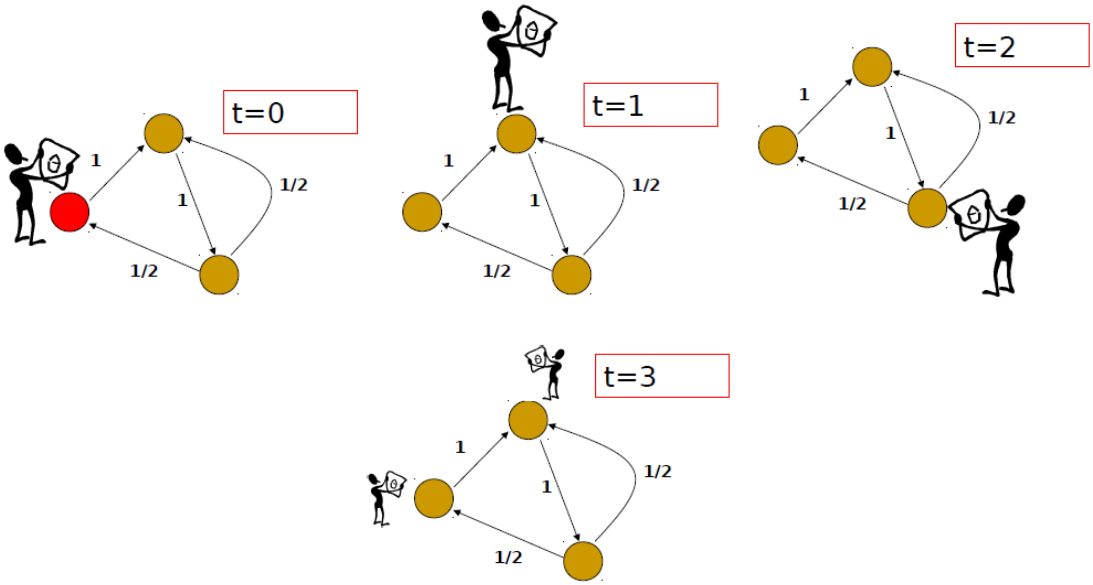


Figure 3.8: Random walk on the graph

3.3.7 Hubs and Authorities

Hyperlink-Induced Topic Search (HITS) (also known as **hubs and authorities**) is a link analysis algorithm that rates Web pages, developed by Jon Kleinberg. It was a precursor to PageRank. The idea behind Hubs and Authorities stemmed from a particular insight into the creation of web pages when the Internet was originally forming; that is, certain web pages, known as hubs, served as large directories that were not actually authoritative in the information that it held, but were used as compilations of a broad catalog of information that led users directly to other authoritative pages. In other words, a good hub represented a page that pointed to many other pages, and a good authority represented a page that was linked by many different hubs.

The scheme therefore assigns two scores for each page: its **authority**, which estimates the value of the content of the page, and its **hub value**, which estimates the value of its links to other pages. Mathematically, these two centrality values are expressed as follows. The **authority centrality** of a node (x_i) is proportional to the sum of hub centralities of nodes (y_j) pointing to it, and is defined as

$$x_i = \alpha \sum_j A_{ji} y_j$$

The **hub centrality** of a node is proportional to the sum of authority centralities of nodes it points to, and is defined as

$$y_i = \beta \sum_j A_{ij} x_j$$

In matrix terms, $\mathbf{x} = \alpha \mathbf{A}^T \mathbf{y}$, and $\mathbf{y} = \beta \mathbf{A} \mathbf{x}$. Solving these two equations gives us

$$\mathbf{x} = \alpha\beta \mathbf{A}^T \mathbf{A} \mathbf{x}$$

$$\mathbf{y} = \alpha\beta \mathbf{A} \mathbf{A}^T \mathbf{y}$$

where $\mathbf{x}$ converges to the prinicipal eigenvector of $\mathbf{A}^T\mathbf{A}$, and $\mathbf{y}$ converges to the principal eigenvector of $\mathbf{A}\mathbf{A}^T$.

3.3.8 Co-citation Index & Bibliographic Coupling

Bibliographic coupling or co-citation occurs when two works reference a common works in their bibliographies. It is an indication that the two works treat related subject matter.

For example, consider works A & B that both cite works C & D. A & B have never cited each other, nor have C & D. However, intuitively, there seems to be a latent relationship between A & B, as well as between C & D. The relationship between A & B is captured by **bibliographic coupling** and is equal to $\mathbf{A}\mathbf{A}^T$. Similarly, the relationship between C & D is captured by **co-citation index** and is equal to $\mathbf{A}^T\mathbf{A}$.

3.3.9 Closeness Centrality

The **Closeness Centrality** measure uses not only the neighbors of a node to determine its centrality, but also the neighbors of the neighbors. Therefore, nodes that are not directly connected to the given node are also considered. Nodes that are not directly connected with the given node receive a lower weight because the intensity of their relation or their influence is lower.

Formally, closeness centrality is a measure of the *mean distance* from the given node i to all other nodes. Let d_{ij} be the length of the geodesic path from node i to node j. Then, the mean geodesic distance from vertex i to the other nodes can be expressed as

$$l_i = (N)^{-1} \sum_j d_{ij}$$

where N = total number of nodes. Note that when $j = i$, $d_{ij} = d_{ii} = 0$; so, it is better to use

$$l_i = (N-1)^{-1} \sum_{i \neq j} d_{ij}$$

The mean geodesic distance gives low values for more central vertices. Therefore, we consider the reciprocal as the value of the centrality, and

$$C_i = l_i^{-1} = \frac{N}{\sum_j d_{ij}}$$

However, this expression suffers from sparsely placed values, and does not account for disconnected network components. The measure therefore can be further refined by considering the harmonic mean. The centrality value then becomes

$$C_i' = \frac{\sum_j d_{ij}^{-1}}{N-1}$$

3.4 Reciprocity

The concept of reciprocity can be described as follows. If there is a directed edge from node i to node j in a directed network and there is also an edge from node j to i, then

the edge from i to j is said to be **reciprocated**. Pairs of reciprocated edges are called **co-links**.

Formally, the **reciprocity** r is defined as the fraction of edges that are reciprocated. Thus, it can be expressed as

$$r = \frac{\sum_{ij} A_{ij}A_{ji}}{m}$$

where m is the total number of edges.

3.5 Rich-Club Coefficient

In a network, when influential people (nodes) come together to collaborate on something, they form what is called a **Rich club**. As an example, *hubs* in a network are generally densely connected, and form a rich-club.

Formally, the rich-club of degree k of a network $\mathbf{G} = (V, E)$ is the set of vertices with degree greater than k. This can be mathematically expressed as,

$$R(k) = \{v \in V | k_v > k\}$$

The rich-club coefficient of degree k is given by,

$$\frac{\#\text{edge}(i,j)}{|R(k)||R(k)-1|}, \text{ where } (i,j) \in R(k)$$

3.6 Entropy of Degree Distribution

The **entropy of the degree distribution** of a network provides an average measure of its heterogeneity. Mathematically, it can be expressed as,

$$H = \sum_k p_k \log(p_k)$$

Intuitively, we can see that the entropy of the degree distribution of a regular graph is 0, and is maximum for a graph having a degree distribution that is distributed uniformly.

3.7 Matching Index

A **matching index** can be assigned to each edge in a network in order to quantify the similarity between the connectivity pattern of the two vertices adjacent to that edge. A low value of matching index would indicate dissimilar regions of the network, with the edge serving as a shortcut between distant regions in the network.

Formally, the matching index of edge (i, j) is defined as

$$\mu_{ij} = \frac{\sum_{k \neq i,j} A_{ik}A_{kj}}{\sum_{k \neq j} A_{ik} + \sum_{k \neq i} A_{jk}}$$

Chapter 4

Social Networks

In this section, we study properties and metrics that are useful to describe and analyze social networks.

4.1 Assortativity

Assortativity (also known as **homophily**) can be described as the preference for a network's nodes to attach to others that are similar or different in some way. Nodes that are similar are **assortative**, and nodes that are different are termed **disassortative**. Though the specific measure of similarity may vary, network theorists often examine assortativity in terms of a node's degree.

4.1.1 Measuring Assortativity

One means of capturing the degree correlation is by examining the properties of $\langle k_{nn} \rangle$, or the average degree of neighbors of a node with degree k. This term is formally defined as

$$\langle k_{nn} \rangle = \sum_{k'} k' P(k'|k)$$

where $P(k'|k)$ is the the conditional probability that an edge of node degree k points to a node with degree k'. If this function is increasing, the network is assortative, since it shows that nodes of high degree connect, on average, to nodes of high degree. Alternatively, if the function is decreasing, the network is dissortative, since nodes of high degree tend to connect to nodes of lower degree.

4.1.2 Mixing Patterns

Mixing patterns refer to systematic tendencies of one type of nodes in a network to connect to another type. Nodes might tend to link to others that are very similar or very different. Mixing, therefore, can be classified broadly as *assortative* or *disassortative*. Assortative mixing is the tendency for nodes to connect to like nodes, and disassortative mixing captures the opposite case in which very different nodes are connected.

Node characteristics involved in the process of creating a link between a pair shape a network's mixing patterns. Real-world node characteristics are virtually unlimited, but

fall under two headings: *discrete* and *topological*. We explore mixing based on discrete characteristics.

Discrete characteristics of a node are categorical, nominal and qualitative. Commonly examined characteristics of this type include race, gender and sexual orientation. To measure the mixing of a network, define e_{ij} to be the fraction of edges in a network that connect nodes of type i to type j. On an undirected network, this quantity is symmetric, i.e. $e_{ij} = e_{ji}$. It satisfies the following sum rules

$$\sum_{ij} e_{ij} = 1, \sum_{j} e_{ij} = a_i, \sum_{i} e_{ij} = b_j$$

where a_i and b_i are the fractions of each type of an edge's end that is attached to nodes of type i (see Figure). Then, an *assortativity coefficient*, a measure of the strength of similarity or dissimilarity between two nodes on a set of discrete characteristics can be defined as

$$r = \frac{\sum_i e_{ii} - \sum_i a_i b_i}{1 - \sum_i a_i b_i}$$

This formula yields $r = 0$ when there is no assortative mixing, and $r = 1$ when the network is perfectly assortative. If the network is perfectly disassortative, the formula yields

$$r_{min} = -\frac{\sum_i a_i b_i}{1 - \sum_i a_i b_i}$$

Women

Men	black	hispanic	white	other
black	506	32	69	26
hispanic	23	308	114	38
white	26	46	599	68
other	10	14	47	32

Women

e_{ij}	black	hispanic	white	other	a_i
black	0.258	0.016	0.035	0.013	0.323
hispanic	0.012	0.157	0.058	0.019	0.247
white	0.013	0.023	0.306	0.035	0.377
other	0.005	0.007	0.024	0.016	0.053
b_j	0.289	0.204	0.423	0.084	$r = 0.621$

Men

Figure 4.1: Mixing Patterns

4.2 Signed Graphs

A **signed graph** is a graph in which each edge has a positive or negative sign. Such graphs have been used to model social situations, with positive edges representing friendships and negative edges representing enmities between nodes, which represent people. The sign of a cycle in the graph is defined to be the product of the signs of its edges; in other words, a cycle is **positive** if it contains an even number of negative edges and **negative** if it contains an odd number of negative edges. A signed graph, or a subgraph or edge set, is called **balanced** if every cycle in it is positive.

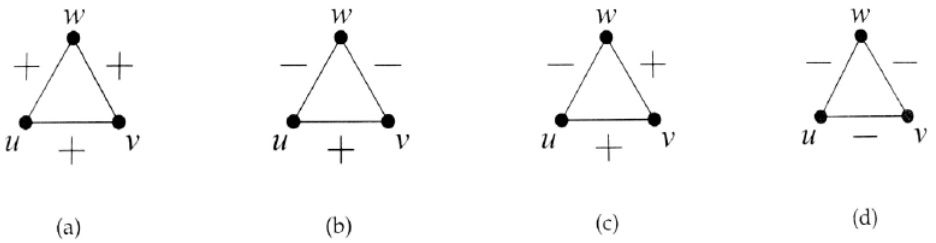


Figure 4.2: Triads (a) & (b) are stable configurations, while (c) & (d) are unstable

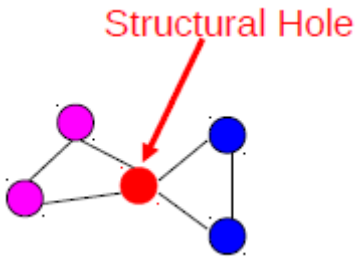
4.2.1 Stability of Cycles

Positive cycles are supposed to be **stable** social situations, whereas negative cycles are supposed to be **unstable**. For example, consider the case of triads, or possible 3-cycles. Then, a stable triad is either three mutual friends, or two friends with a common enemy; while an unstable 3-cycle is either three mutual enemies, or two enemies who share a mutual friend. According to the theory, in the case of three mutual enemies, this is because sharing a common enemy is likely to cause two of the enemies to become friends. In the case of two enemies sharing a friend, the shared friend is likely to choose one over the other and turn one of his or her friendships into an enmity.

4.3 Structural Holes

In a social network, **structural holes** are nodes that separate non-redundant sources of information, that is sources that are additive rather than overlapping.

Contacs that are strongly connected to each other are likely to have similar information and therefore provide redundant information benefits. On the other hand, contacts that link a *manager* to the same third parties have same sources of information and therefore provide redundant information benefit.



4.4 Social Cohesiveness

Social cohesiveness refers to the closeness of the members in the social network. In graph theoretic terms, it refers to the "cliquishness" of a graph. However, a complete clique is too strict to be practical and is rarely observed in social networks. In most real world groups, there are bound to exist at least a few members who are not connected to

each other. We define some other relaxed and pratical measures of social cohesiveness.

A k**-clique** is a *maximal* set S of nodes in which the geodesic path between every pair of nodes $\{u, v\} \in S$ is less than or equal to k. As an examples, consider the network in Figure 4.3.

$\{a, b, c, f, e\}$ forms a 2-clique, as the node d causes the distance between the nodes c

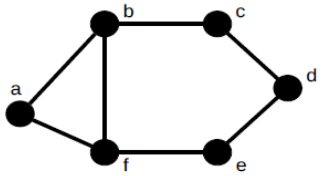


Figure 4.3: $\{a, b, c, f, e\}$ forms a 2-clique

and e to be 2, even though it is not a part of the 2-clique. Thus, k-cliques might not be as cohesive as they look. To resolve this issue, we consider k-clans.

A k**-clan** is a k-clique in which the subgraph induced by S has diameter less than or equal to k. In the previous figure, $\{b, c, d, e, f\}$ forms a 2-clan. Note than $\{b, e, f\}$ also induces a subgraph that has diameter 2, but it does not form a 2-clan, as it is not a maximal set. If we relax the maximality condition on k-clans, we get a k**-club**. $\{a, b, f, e\}$ forms a k-club in the network. It is easy to see that any k-clan is both a k-clique and a k-club.

A k**-plex** is a maximal subset S of nodes such that every member of the set is connected to at least $n - k$ other members, where n is the size of S. In Figure 4.4, we see that $\{a, b, e, d\}$ forms a 2-plex.

A k**-core** of a graph is a maximal subgraph such that each node in the subgraph

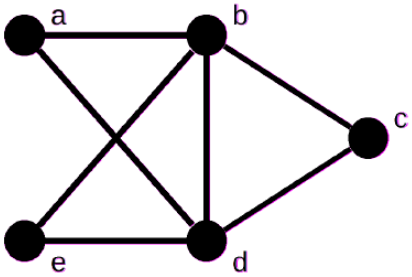


Figure 4.4: $\{a, b, e, d\}$ forms a 2-plex

has at least degree k, as shown in Figure 4.5.

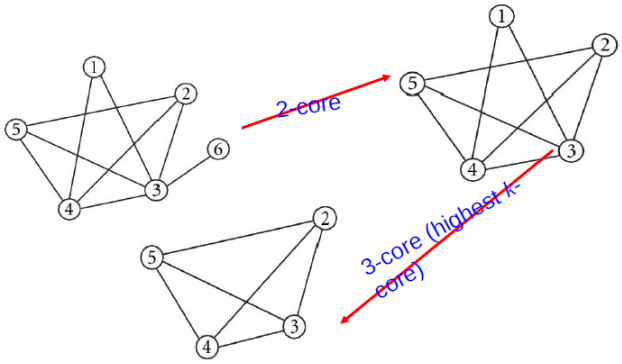


Figure 4.5: k-cores

4.5 Equivalence

In social networks, the different *roles*, or *positions*, or *social categories* are defined by the *relations* among the actors, represented as nodes. Two actors have the same position or role to the extent that their pattern of relationships with the other actors is the same. But how does one define such a similarity? To do this, we need to define some measure of **equivalence** between actors.

There are many ways in which actors could be defined as equivalent based on their relations with others. Three particular definitions of equivalence have been particularly useful in applying graph theory to the understanding of social roles and structural positions, namely, structural equivalence, automorphic equivalence, and regular equivalence.

4.5.1 Structural Equivalence

Two nodes are said to be exactly **structurally equivalent** if they have the same relationships with all other nodes, that is, one should be perfectly substitutable by the other. Simply put, the two must be connected to exactly the same set of neighbors. However, exact structural equivalence is likely to be rare, particularly in large networks. Therefore, there is a need to examine the **degree of structural equivalence**, rather than the simple presence or absence of exact equivalence.

The degree of equivalence between two nodes i and j can be measured by examining the number of common neighbors between the two nodes. Formally, it can be expressed as,

$$n_{ij} = \sum_k A_{ik} A_{jk}$$

which, incidentally, is nothing but the ij^{th} element of the matrix $\mathbf{A}^2$ for undirected graphs. Note that this is closely related to the cocitation measure in directed networks. Moreover, since we are measuring the extent of similarity, the above quantity must be **appropriately normalized**. Therefore, the measure can be refined by some alternate considerations which we enumerate below.

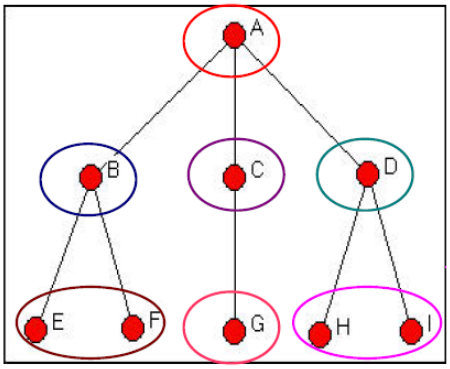


Figure 4.6: Different structural equivalence classes

4.5.1.1 Cosine Similarity

This similarity measure is defined as the inner product of two vectors. That is,

$$similarity(x, y) = \cos\theta = \frac{x \cdot y}{||x|| * ||y||}$$

Consider the i^{th} and j^{th} rows of the adjacency matrix $\mathbf{A}$ as vectors. Then the cosine similarity between vertices i and j is

$$\begin{aligned}\sigma_{ij} &= \frac{\sum_k A_{ik}A_{jk}}{\sqrt{\sum_k A_{ik}^2}\sqrt{\sum_k A_{jk}^2}} \\ &= \frac{n_{ij}}{\sqrt{k_i k_j}}\end{aligned}$$

4.5.1.2 Pearson Correlation Coefficient

The correlation coefficient between rows i and j is defined as

$$\begin{aligned}r_{ij} &= \frac{\text{Cov}(X_i, X_j)}{\sqrt{\text{Var}(X_i)\text{Var}(X_j)}} \\ &= \frac{\sum_k A_{ik}A_{jk} - \frac{k_i k_j}{n}}{\sqrt{k_i - \frac{k_i^2}{n}}\sqrt{k_j - \frac{k_j^2}{n}}}\end{aligned}$$

4.5.1.3 Euclidean Distance

The Euclidean distance is defined as

$$d_{ij} = \sum_k (A_{ik} - A_{jk})^2$$

For a binary graph, this is nothing but the Hamming distance. Moreover, to get the required similarity value, we need to normalize d_{ij} by the maximum possible distance between the nodes. This is achieved none of i's neighbors (k_i) match with the j's neighbors (k_j). Therefore, the similarity value is

$$similarity = \frac{d_{ij}}{k_i + k_j}$$

4.5.2 Automorphic Equivalence

Automorphic equivalence is not as demanding a definition of similarity as structural equivalence, but is more demanding than regular equivalence. There is a hierarchy of the three equivalence concepts: any set of structural equivalences are also automorphic and regular equivalences. Any set of automorphic equivalences are also regular equivalences. Not all regular equivalences are necessarily automorphic or structural; and not all automorphic equivalences are necessarily structural.

Formally, two vertices u and v of a labeled graph G are **automorphically equivalent** if all the vertices can be re-labeled to form an isomorphic graph with the labels of u and v interchanged. Two automorphically equivalent vertices share exactly the same label-independent properties.

More intuitively, actors are automorphically equivalent if we can permute the graph in such a way that exchanging the two actors has no effect on the distances among all actors in the graph. If we want to assess whether two actors are automorphically equivalent, we first imagine exchanging their positions in the network. Then, we look and see if, by changing some other actors as well, we can create a graph in which all of the actors are the same distance that they were from one another in the original graph.

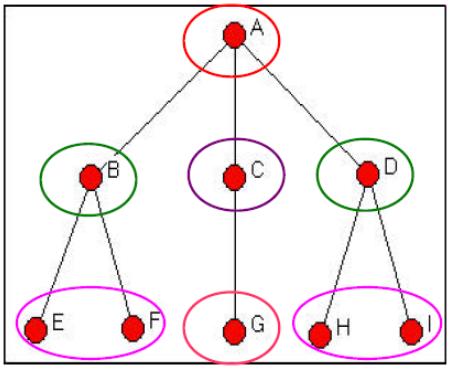


Figure 4.7: Different automorphic equivalence classes (color-coded)

4.5.3 Regular Equivalence

Regular equivalence is the least restrictive of the three most commonly used definitions of equivalence. It is, however, probably the most important for the sociologist. This is because the concept of regular equivalence, and the methods used to identify and describe regular equivalence sets correspond quite closely to the sociological concept of a role.

Formally, two actors are **regularly equivalent** if they are equally related to equivalent others. That is, regular equivalence sets are composed of actors who have similar relations to members of other regular equivalence sets. The concept does not refer to ties to specific other actors, or to presence in similar sub-graphs; actors are regularly equivalent if they have similar ties to any members of other sets.

The concept is actually more easy to grasp intuitively than formally. Susan is the daughter

of Inga. Deborah is the daughter of Sally. Susan and Deborah form a regular equivalence set because each has a tie to a member of the other set. Inga and Sally form a set because each has a tie to a member of the other set. In regular equivalence, we don't care which daughter goes with which mother; what is identified by regular equivalence is the presence of two sets (which we might label "mothers" and "daughters"), each defined by its relation to the other set. Mothers are mothers because they have daughters; daughters are daughters because they have mothers.

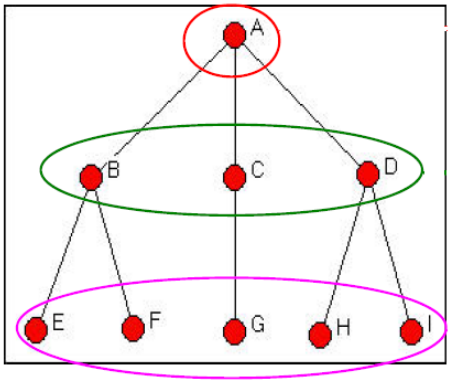


Figure 4.8: Different regular equivalence classes

4.5.3.1 Computing Regular Equivalence

To compute a measure of regular equivalence, we capture the following notion - vertices i and j are similar if i has a neighbor k that is itself similar to j. Mathematically,

$$\sigma_{ij} = \alpha \sum_k A_{ik}\sigma_{kj} + \delta_{ij}$$

This can be represented in matrix form as

$$\sigma = \alpha \mathbf{A}\sigma + \mathbf{I}$$
$$\Rightarrow \sigma = (\mathbf{I} - \alpha \mathbf{A})^{-1}$$

4.6 Ego-centric Networks

Social scientists often talk about people's egocentric networks - the cloud of friends and acquaintances that one has, which if diagrammed would have one at the center with edges connecting him/her to other people in his/her life. Such analyses, however, are most commonly used in the fields of psychology or social pyschology, ethnographic kinship analysis or other genealogical studies of relationships between individuals.

To better understand the idea, we need some definitions.

The **Ego** is an individual *focal* node. A network has as many egos as it has nodes. Egos can be persons, groups, organizations, or whole societies.

The nodes to whom the ego is connected are called the **Alters**.

The **Neighborhood** is the collection of the ego and all nodes to whom the ego has a connection at some path length. In social network analysis, the neighborhood is almost always one-step; that is, it includes only the ego and its alters. The neighborhood also includes all of the ties among all of the actors to whom ego has a direct connection. The boundaries of ego networks are defined in terms of neighborhoods.

The N**-step neighborhood** expands the definition of the size of the ego's neighborhood by including all nodes to whom ego has a connection at a path length of N, and all the connections among all of these actors. Neighborhoods of greater path length than 1 (i.e. egos adjacent nodes) are rarely used in social network analysis. When we use the term neighborhood here, we mean the one-step neighborhood.

Chapter 5

Community Structures

A network is said to have **community structures** if the nodes of the network can be easily grouped into (potentially overlapping) sets of nodes such that each set of nodes is densely connected internally, while interconnections between these sets are sparse. In this chapter we study different techniques for the identification of community structures, also known as clustering. Identification of such structures finds many practical applications, such as the construction of recommender systems, polarity detection, etc. An important point to note is that the definition of a community is subjective and largely depends on the application for which clustering is being performed.

Based on the approach employed, clustering techniques can be broadly categorized into the following computational methods, *agglomerative*, *divisive* and *spectral*.

Agglomerative techniques make use of a bottom-up approach for clustering. Starting with an empty graph G with N nodes and no edges, edges are iteratively added to the graph, while maximizing some quantity in the original network.

Divisive techniques make use of a top-down approach, removing certain edges from the original network so that separate community structures are obtained.

Spectral techniques split the graph into community structures based on eigenvalues / eigenvectors of the Graph Laplacian.

5.1 Similarity Measures

A crucial step in any algorithm to identify community structures is to select suitable metrics to measure similarity or dissimilarity between nodes. The goal remains to to group similar data together, which would constitute a community. However, there is no single method that works equally well in all applications; it depends on what we want to find or emphasize in the data. Therefore, correct choice of a similarity measure is often more important than the clustering algorithm. As discussed in previous chapters, similarity measures could be obtained as Cosine Similarity, Jaccard's Coefficient, etc.

5.2 Agglomerative Methods

In this section, we describe some agglomerative approaches towards clustering.

5.2.1 Hierarchical Clustering

The approach is as follows:

1. Start with every data point in a separate cluster
2. Merge the most *similar* pairs of data points / clusters together, until only a single cluster remains

The output of the above method is a binary tree, called a **dendogram**. The root of this tree is the final cluster, and each original data item is a leaf. Initially, the tree is empty, containing only the original data items as leaves. Whenever data items / clusters are merged together, a node is added to the tree (representing this new cluster) with edges between this new node and its constituent clusters.

As already mentioned, we could have used any of the previously defined measures of similarity to estimate the distance between data items. However, we need to define a **linkage** method that can estimate the distance between clusters. Since a data item can be thought of as a cluster with a single node, this linkage method will suffice for data items as well. Here we enumerate the different types of linkages that might be followed while merging any two clusters:

- **Single Linkage**: The minimum of all pairwise distances between points in the two clusters
- **Complete Linkage**: The maximum of all pairwise distances between points in the two clusters
- **Average Linkage**: The average of all pairwise distances between points in the two clusters

Despite its simplicity, this approach does not scale to large graphs, owing to its $O(n^3)$ time complexity in the worst case. Also, the method is not flexible; steps once taken cannot be undone. Another problem this approach suffers from is that arbitrary cut-offs need to be set to arrive at a community structure.

5.2.2 Local Algorithm based on Agglomeration

This algorithm, due to James P. Bagrow, agglomerates nodes one at a time, and maintains two groups - a *community* C and a *border* B consisting of the set of nodes adjacent to the community, i.e. each node in B has at least one neighbour in C. At each step, a node from B is chosen and agglomerated into C, then B is update to include any newly discovered nodes. This continues until an appropriate stopping criterion has been satisfied. Initially, a node is chosen as the source s, and $C = \{s\}$, and B contains the neighbours of s: $B = \{n(s)\}$.

Define the *outwardness* $\Omega_v(C)$ of a node $v \in B$ from community C as

$$\Omega_v(C) = \frac{\#\text{ of neighbours of } v \text{ outside } C - \#\text{ of neigbours of } v \text{ inside } C}{k_v}$$

Now, the algorithm moves that node from B to C whose outwardness value is minimum, breaking ties at random. B is now updated, and the procedure is repeated until the stopping criterion has been satisfied.

5.3 Divisive Methods

5.3.1 Girvan-Newman Algorithm

We studied the betweenness of a vertex previously, as a measure of centrality and influence of nodes in networks. The **Girvan-Newman** algorithm extends this definition to the case of edges, defining the **edge-betweenness** of an edge as the number of shortest paths between pairs of nodes that run along it. If there is more than one shortest path between a pair of nodes, each path is assigned equal weight, such that the total weight of all the paths is equal to unity. If a network contains communities or groups that are only loosely connected by a few intergroup edges, then all shortest paths between different communities must go along one of these few edges. Thus, the edges connecting communities will have high edge betweenness (at least one of them). By removing these edges, the groups are separated from one another and so the underlying community structure of the network is revealed.

The algorithm's steps for community detection are summarized below:

1. The betweenness of all existing edges in the network is calculated first.
2. The edge with the highest betweenness is removed.
3. The betweenness of all edges affected by the removal is recalculated.
4. Steps 2 and 3 are repeated until no edges remain.

The end result of the Girvan-Newman algorithm is a dendogram. As the Girvan-Newman algorithm runs, the dendogram is produced from the top down.

The crux of this method lies in the computation of the shortest paths. If we use simple BFS traversal for this computation, then, this can be done in $O(m)$ time for each source node, totalling to $O(mn)$ time for all the nodes, where m is the number of edges in the graph. In the worst case, $O(m)$ edges are removed, therefore, the total complexity of the algorithm is $O(m^2n)$, which is equivalent to $O(n^3)$ for sparse graphs, and $O(n^5)$ for dense graphs.

5.3.2 Radicchi's Algorithm

This algorithm is a divisive algorithm that is based on the notion that the number of triangles formed within communities is much higher than the number of triangles across communities. The algorithm tries to find the **edge clustering coefficient** of each edge; we remove the edge with the smallest value of the coefficient from the network. This

coefficient is a measure of the number of triangles a particular edge ij is a part of, and is defined as:

$$C_{ij} = \frac{Z_{ij} + 1}{\min{(k_i - 1), (k_j - 1)}}$$

where Z_{ij} = Number of triangles ij is a part of. Note that the denominator of the expression denotes the maximum number of triangles of which ij could possibly be a part of, but also 1 is added to the numerator to eliminate the possibility that $C_{ij} = 0$.

This algorithm runs in time $O(m^2)$ as each iteration of the algorithm requires $O(m)$ computations, and there can be $O(m)$ such iterations.

5.4 Modularity Optimization

Modularity is a metric that measures the strength of division of a network into communities. Networks with high modularity have dense connections between the nodes within communities, but sparse connections between nodes in different communitites. Modularity is often used in optimization methods for the detection of community structures. Intuitively, it can be measured as the total number of in community edges minus the expected number of edges in the absence of a community structure. Formally, it is defined as

$$Q = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

where m is the total number of edges, c_i is the community to which i is assigned, and $\delta(c_i, c_j)$ is 1, if $c_i = c_j$, and 0 otherwise.

Thus, modularity can be used as a stopping criterion in iterative clustering algorithms. The iterations are performed until modularity reaches a maximum; the point at which degradation starts is the point where further clustering is not performed.

5.4.1 Newman's Modularity Optimization Algorithm

The above discussion highlights the use of modularity for the evaluation of computed communities. However, it can itself be used for the purpose of community identification, as depicted by the following agglomerative algorithm.

1. Initially, all vertices are kept in separate clusters.

2. Join a pair of clusters, such that this results in the greatest increase or smallest decrease in the modularity, Q (optimizing ΔQ).

3. Repeat.

Note that this agglomerative algorithm is much faster than the Girvan-Newman edge-betweenness algorithm; there are $O(m)$ computations per step, and $O(n)$ steps, so the net complexity is $O(mn)$.

5.4.2 Modularity Optimization : Blondel et al.

A smarter way of applying modularity optimization due to Blondel et al., is given below. Each pass of the algorithm comprises two phases: one where modularity is optimized by allowing only local changes of communities; one where communities found are aggregated in order to build a new network of communities. The passes are repeated iteratively until no increase of modularity is possible.

Phase 1

1. All nodes are kept in separate clusters.
2. For each node i, consider all its neighbours j.
3. Check whether placing i in j's current community increases ΔQ (gain should always be positive).
4. Place i in the community for which ΔQ is maximum, breaking ties randomly.
5. Continue until ΔQ is 0.

Phase 2

1. Collapse all communities obtained from phase 1 to single nodes.
2. Multiple edges between the newly obtained collapsed communities are replaced by a single edge of weight equal to the sum of the weights of the edges connecting them previously.

An obvious question arises regarding the effect of ordering of vertices on the performance of the algorithm, however, choosing the order does not affect the algorithm much.

5.5 Infomap

Please refer to the PNAS paper titled Maps of random walks on complex networks reveal community structure for an in-depth analysis of Infomap. Also, refer to this article for Huffman Coding.

5.6 Spectral Bisection Methods

This method relies on eigenvector analysis. It considers the multiplicity of the eigenvalues of the matrix under consideration. All components of the principal eigenvector are positive, whereas for the second eigenvector, some are positive and some are negative, and this is what the spectral bisection methods use for classification.

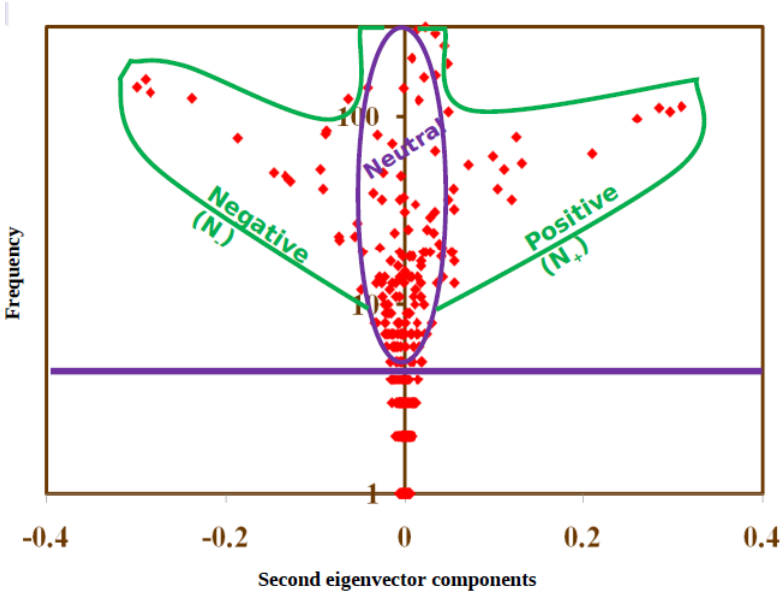
The **Laplace Matrix**, L, is given by $D - A$, where D is a diagonal matrix with the degrees of all the vertices on the principal diagonal, and A is the incidence matrix. It can be noted that an eigenvalue for the laplace matrix is the unit vector v, and the eigenvalue is 0, as the sum of any row is 0.

$$L.v = 0v$$

The incidence matrix, I_n, gives the relation between the vertices and the edges of a graph. We state the following results about the incidence matrix,

1. $L = I_n I_n^T$, where L is the Laplacian.
2. 0 is an eigenvalue.
3. All eigenvalues λ are positive, and given by $\lambda = \frac{\sum_{i,j}(v_i - v_j)^2}{\sum_i v_i^2}$.

As stated earlier, all components of the principal eigenvector are positive, but not so for the second eigenvector, which is exploited for classification, as shown. If some nodes fail to be classified, we can use the third eigenvector, or delete nodes that have already been classified, and repeat on the substrate graph till everything is classified.



Here we present an informal proof of point 3 above, and show how all eigenvalues are positive. Consider the Laplace matrix L, and an eigenvector v, with the corresponding eigenvalue λ.

$$\begin{aligned} Lv &= \lambda v \\ v^T Lv &= \lambda v^T v \\ \implies \lambda &= \frac{v^T Lv}{v^T v} \\ &= \frac{(v^T I_n)(I_n^T v)}{v^T v} \end{aligned}$$

The above equation is obtained using point 1 from above. Now, $v^T v = \sum_i v_i^2$, and $(v^T I_n)(I_n^T v) = Y^T Y = \sum_{i,j}(v_i - v_j)^2$. Hence we obtain,

$$\lambda = \frac{\sum_{i,j}(v_i - v_j)^2}{\sum_i v_i^2}$$

which shows all eigenvalues are positive.

Chapter 6

Random Graphs

In mathematics, a **random graph** is a graph that is generated by some random process. The theory of random graphs lies at the intersection between graph theory and probability theory, and studies the properties of typical random graphs.

6.1 Definition and Properties

Notationally, a random graph of n nodes where every pair of nodes is connected by an edge with a probability p is represented as $G(n, p)$. We have the following properties for a random graph $G(n, p)$.

1. The probability that $G(n, p)$ contains exactly m edges in any ensemble is proportional to $p^m(1-p)^{\binom{n}{2}-m}$.

2. The average degree $\langle k \rangle$ of the nodes is equal to $p(n-1)$.

3. The clustering coefficient of $G(n, p)$ is equal to p and is independent of any other parameter.

4. The probability p_k that a node has exactly k neighbours, which is nothing but the degree distribution of the graph, is equal to $\binom{n-1}{k}p^k(1-p)^{(n-1)-k}$. In the limit for large n,the degree distribution can be approximated by the Poisson law, with $p_k = e^{-z}z^k/k!$ where z is the average degree of the nodes . Such graphs are therefore called Poisson random graphs.

5. The average distance d is equal to $\log_z n$ where z is the average degree $\langle k \rangle$. This equation follows from the observation that $z^d = n$.

6.2 Giant Components

As the name suggests, a **giant component** is a connected component of a given random graph that contains a large fraction of the entire graph's vertices. Given a random graph $G(n, p)$, lower values of p imply low edge density, and the graph does not contain giant components. Higher values of p imply high edge density, and we thus observe the emergence of giant components in the graph.

We will now proceed to quantify the fraction of nodes that are part of a giant component GC in a random graph $G(n, p)$.

For any node v, let t be the probability with which v does not belong to GC. Then, the probability that none of v's neigbors are in GC either is given by t^k, where k is the degree of v.

Again, if no neighbor of v belongs to GC, then v does not belong to GC either. This observation leads to the following equation:

$$t = \sum_{k=0}^{\infty} t^k p_k$$

There should be no confusion about the presence of t on both sides of the equation. In the R.H.S., p_k gives the probability that the degree of v is k; t^k removes the neighbors of v from GC; the summation is then done over all possible degrees for v. Thus, the R.H.S. gives the probability with which no neighbor of v is in GC, which is nothing but t.

For large n, we can substitute p_k with the Poisson distribution. Thus,

$$\begin{aligned} t &= \sum_{k=0}^{\infty} e^{-z} \frac{t^k z^k}{k!} \\ &= e^{-z} \sum_{k=0}^{\infty} \frac{(tz)^k}{k!} \\ &= e^{-z} \cdot e^{tz} \\ &= e^{(t-1)z} \\ \Rightarrow 1 - t &= 1 - e^{(t-1)z} \end{aligned}$$

Let S be the fraction of nodes that belong to GC. Then $S = 1 - t$. Therefore, from the previous equation we have:

$$S = 1 - e^{-zS}$$

6.3 Generating Functions

In mathematics, a **generating function** is a formal power series in one indeterminate, whose coefficients encode information about a sequence of numbers a_n that is indexed by the natural numbers. Generating functions are not functions in the formal sense of a mapping from a domain to a codomain; the name is merely traditional, and they are sometimes more correctly called **generating series**. According to Herbert Wilf, "A generating function is a clothesline on which we hang up a sequence of numbers for display."

6.3.1 Definition

Generating functions are often expressed in closed form (rather than as a series), by some expression involving operations defined for formal power series. An **ordinary generating**

function is a power series defined as follows:

$$G(x) = \sum_{n=0}^{\infty} a_n x^n$$

For example, given a set of coefficients $\{a_i\} = \{1, 1, 1, 1, 0, 0, \cdots\}$, we can encode this as follows:

$$\begin{aligned} G(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 \\ &= 1 + x + x^2 + x^3 \\ &= \frac{1 - x^4}{1 - x} \end{aligned}$$

Thus, we can encode a sequence by a closed form equation (as opposed to a series), and very easily evaluate the required coefficients. As an example, consider the problem of finding the coefficient of x^{2012} in $\frac{1}{1+3x}$. By differentiating the expression 2012 times, and then substituting $x = 0$, we get the required coefficient as 3^{2012}.

6.3.2 Generating Functions on Graphs

Consider the degree distribution of a graph p_k. Then, the generating function for the graph is given by

$$G(x) = p_0 + p_1 x + p_2 x^2 + \cdots$$

So, given the generating function of a graph, we can find the degree distribution p_k as

$$p_k = \frac{1}{k!} \left[\frac{d^k}{dx^k} (G(x)) \right]_{x=0}$$

Given a random graph G and its generating function $G(x)$, we can find the average degree $\langle k \rangle$ of its nodes as follows,

$$\begin{aligned} \langle k \rangle &= \sum_k k p_k \\ &= 0p_0 + 1p_1 + 2p_2 + \cdots \\ G_0(x) &= p_0 + p_1 x + p_2 x^2 + \cdots \\ \Rightarrow G_0'(x) &= p_1 + 2p_2 x + \cdots \\ \Rightarrow G_0'(1) &= 0p_0 + 1p_1 + 2p_2 + \cdots \\ &= \langle k \rangle \end{aligned}$$

6.3.3 Powers of Generating Functions

Consider the m^{th} power of the function $G_0(x)$, which is $[G_0(x)]^m$. If we choose m independent nodes from a graph, then the probability distribution of the sum of the degrees of these m nodes is given by $[G_0(x)]^m$. For example, if $m = 2$, then we have

$$\begin{aligned} [G_0(x)]^2 &= \left[\sum_{k=0}^{\infty} p_k x^k \right]^2 \\ &= \sum_{k,j} p_k p_j x^{k+j} \\ &= p_0 p_0 x^0 + (p_0 p_1 + p_1 p_0) x^1 + (p_0 p_2 + p_1 p_1 + p_2 p_0) x^2 + \cdots \end{aligned}$$

6.4 Degree Distribution

Random graphs have degree distributions that follow a *poisson* distribution, which can be seen from the following derivation. The probability distribution p_k is defined as

$$p_k = \binom{n}{k} p^k (1-p)^{n-k}$$

Now, using the theory of generating functions, we can write $G_0(x)$ for our graph G as

$$\begin{aligned} G_0(x) &= \sum_k p_k x^k \\ &= (1 - p + px)^n \end{aligned}$$

Now, in the $\lim_{n \to +\infty}$, we have that

$$\lim_{n \to +\infty} (1 - p + px)^n = \exp z(n-1)$$

where $z = np$. On differentiating the above equation k times, we get the poisson distribution for p_k.

Chapter 7

Network Growth Models

Many large real world networks are scale-free. If we can capture correctly the processes that assembled the networks that we see today, then we will obtain their topology correctly as well. In this chapter, we will explore the mechanism responsible for the emergence of scale-free networks.

7.1 Barabási-Albert's Model

Most real world networks decribe open systems which *grow* by the continuous addition of new nodes. Also, most real networks exhibit *preferential attachment*, such that the likelihood of connecting to a node depends on the node's degree. These two properties, growth and preferential attachment, inspired the introduction of the Barabási-Albert model that generates a scale-free network (has a power-law degree distribution). The algorithm is as follows:

1. *Growth*: Starting with a small number (m_0) of nodes, at every timestep t we add a new node with $m(\leq m_0)$ edges that link the new node to m different nodes already present in the system.

2. *Preferential Attachment*: When choosing the nodes to which the new node connects, we assume that the probability Π that a new node will be connected to node i depends on the degree k_i of node i, such that

$$\Pi(k_i) = \frac{k_i}{\sum_j k_j}$$

After t timesteps, this algorithm results in a network with $n = m_0 + t$ nodes, and mt edges. Also, note that

$$\begin{aligned}\Pi(k_i) &= \frac{k_i}{\sum_j k_j} \\ &= \frac{k_i}{2mt - m} \\ &\approx \frac{k_i}{2mt}\end{aligned}$$

The dynamic properties of the Barabási-Albert model can be addressed using various analytic approaches. We will describe the continuum theory, and the master/rate equation approach.

7.1.1 Continuum Theory Approach

The continuum approach calculates the time dependence of the degree k_i of a given node i. This degree will increase every time a new node enters the system and links to node i, the probability of this process being $\Pi(k_i)$. Assuming that k_i is a continuous real variable, the rate at which k_i changes is expected to be proportional to $\Pi(k_i)$. Consequently, k_i satisfies the following equation,

$$\begin{aligned}
\frac{\partial k_i}{\partial t} &= m\Pi(k_i) \\
&= \frac{mk_i}{2mt} \\
&= \frac{k_i}{2t} \\
\Rightarrow \frac{\partial k_i}{k_i} &= \frac{\partial t}{2t} \\
\Rightarrow \int_{k_i(t_i)}^{k_i(t)} \frac{\partial k_i}{k_i} &= \int_{t_i}^{t} \frac{\partial t}{2t} \\
\Rightarrow k_i(t) &= m\sqrt{\frac{t}{t_i}}
\end{aligned}$$

Assume a node of degree k. Consider the event that $k_i(t) < k$. Then,

$$\begin{aligned}
&k_i(t) < k \\
\Rightarrow &m\sqrt{\frac{t}{t_i}} < k \\
\Rightarrow &t_i > \left(\frac{m}{k}\right)^{\frac{1}{\beta}} t
\end{aligned}$$

where $\beta = \frac{1}{2}$. Now, the density function

$$P(t_i) = \frac{1}{n} = \frac{1}{m_0 + t}$$

. Thus, we have that,

$$\begin{aligned}
P\left(k_i(t) < k\right) &= P\left(t_i > \frac{m^{1/\beta}t}{k^{1/\beta}}\right) \\
&= 1 - P\left(t_i \leq \frac{m^{1/\beta}t}{k^{1/\beta}}\right) \\
&= 1 - \frac{1}{m_0 + t} \cdot \frac{m^{1/\beta}t}{k^{1/\beta}}
\end{aligned}$$

The degree distribution p_k can be obtained using

$$\begin{aligned}
p_k &= \frac{\partial P(k_i(t) < k)}{\partial k} \\
&= \frac{2m^{1/\beta}t}{k^{1/\beta+1}} \cdot \frac{1}{m_0 + t}
\end{aligned}$$

So, $p_k \propto 1/(k^{1/\beta+1})$. Putting $\beta = 1/2$, we have $p_k \propto k^{-3}$.

7.1.2 Master/Rate Equation Approach

This is approach can be summarized mathematically as follows. First, observe that $\sum kp_k$ denotes the mean degree of a network, and is given by $\sum kp_k = 2mt/t = 2m$. Therefore, we have that

$$\begin{aligned}\Pi(k) &= \frac{kp_k}{\sum kp_k} \\ &= \frac{kp_k}{2m}\end{aligned}$$

Recall that m denoted the number of new edges coming in. The number of vertices of degree k that gain at least one edge

$$\begin{aligned}&= m\frac{kp_k}{2m} \\ &= \frac{1}{2}kp_k\end{aligned}$$

This represents the concentration of edges flowing towards nodes of degree k. Additionally consider the following definitions:

- **Out-flux**: number of nodes of degree k gaining an edge due to the new node, becoming nodes of degree $k+1$
- **In-flux**: number of nodes of degree $k-1$ gaining an edge due to the new node, becoming nodes of degree k

Let n be the total number of nodes in the system. Then, the number of nodes of degree k is given by np_k. For the case of outflux, np_k decreases, while for the case of influx, it increases. Let $p_{k,n}$ denote value of p_k when there are n nodes in the system. Then, the net change in np_k is governed by the following *master equation*

$$(n+1)p_{k,n+1} - np_{k,n} = \frac{k-1}{2}p_{k-1,n} - \frac{k}{2}p_{k,n}$$

The first term on the right hand side denotes the change due to the influx, while the second term denotes the change that results due to the outflux. In the limiting case, as $n \to \infty$, we have

$$p_{k,n+1} \approx p_{k,n} \approx p_k$$

Therefore, at the stationary state, the recurrence relation reduces to

$$p_k = \frac{k-1}{2}p_{k-1} - \frac{k}{2}p_k$$

This implies that p_k is the solution of the following recursive equation:

$$p_k = \begin{cases} 0 & \text{for } k < m \\ 2/(m+2) & \text{for } k = m \\ \frac{k-1}{k+2}p_{k-1} & \text{for } k > m \end{cases}$$

This gives

$$\begin{aligned} p_k &= \frac{(k-1)(k-2)}{(k+2)(k+1)} p_{k-2} \\ &= \frac{(k-1)(k-2)\cdots m}{(k+2)(k+1)\cdots(m+3)} p_m \\ &= \frac{(k-1)(k-2)\cdots m}{(k+2)(k+1)\cdots(m+3)} \frac{2}{m+2} \\ &= \frac{2m(m+1)}{k(k+1)(k+2)} \end{aligned}$$

In the limit of large k, this gives us the power-law degree distribution as $p_k \propto k^{-3}$.

7.2 Price's Model

Derek de Solla Price described probably the first example, of what would now be called a scale-free network. He studied the citation network and discovered that it followed a power law degree distribution. The model can be described as follows.

- Consider a directed graph of n vertices.
- Let p_k be the fraction of vertices, with in-degree k.
- At each step, a new node is added to the system, with a certain out-degree. The out-degree may vary from one vertex to another, but the mean out-degree is kept at a constant m.
- The value of m is also the mean in-degree of the network: $\sum_k k p_k = m$
- The probability that a newly appearing node attaches to an old vertex, is proportional to the in-degree k of the old vertex.

Since each vertex starts with in-degree 0, it would forever have 0 probability of gaining new edges. This problem is circumvented by taking the probability of attachment to a vertex as $k + k_0$, where k_0 is a constant taken as 1. This value of k_0 is justified in the case of citation networks by considering the initial publication of a paper as a citation to itself. Thus if $\Pi(k)$ denotes the probability that a new edge attaches to any of the vertex of degree k, then $\Pi(k) \propto k + 1$. More specifically,

$$\Pi(k) = \frac{(k+1)p_k}{\sum_k (k+1)p_k} = \frac{(k+1)p_k}{m+1}$$

The mean number of new edges per vertex added is simply n, and hence the mean number of new edges to vertices with current in-degree k is given by $mp_k(k+1)/(m+1)$. Let $p_{k,n}$ denote the value of p_k when n nodes have been added. Then, the change in np_k, which is the number of vertices with in-degree k, is given by

$$\begin{aligned} (n+1)p_{k,n+1} - np_{k,n} &= \frac{kp_{k-1,n}}{m+1} m - \frac{(k+1)p_{k,n}}{m+1} m \\ &= \frac{m}{m+1} \left[kp_{k-1,n} - (k+1)p_{k,n} \right] \end{aligned}$$

Looking for stationary solutions as $n \to \infty$, we have $p_{k,n+1} = p_{k,n} = p_k$. Note that for vertices of degree $k = 0$, influx of exactly 1 is possible. We thus have the following recursive solution

$$p_k = \begin{cases} [kp_{k-1} - (k+1)p_k]m/(m+1) & \text{for } k > 0 \\ 1 - p_0 m/(m+1) & \text{for } k = 0 \end{cases}$$

Rearranging we have

$$p_0 = \frac{m+1}{2m+1}$$

and

$$\begin{aligned} p_k &= \frac{kp_{k-1}}{2+k+1/m} \\ &= \frac{k(k-1)\cdots 1}{(2+k+1/m)(1+k+1/m)\cdots(3+1/m)} p_0 \\ &= \frac{k(k-1)\cdots 1}{(2+k+1/m)(1+k+1/m)\cdots(3+1/m)} \frac{m+1}{2m+1} \\ &= \frac{k(k-1)\cdots 1 \times (1+1/m)}{(2+k+1/m)(1+k+1/m)\cdots(3+1/m)\times(2+1/m)} \\ &= (1+1/m)\left[\frac{\Gamma(k+1)\cdot(1+1/m)}{(2+k+1/m)(1+k+1/m)\cdots(1+1/m)}\right] \\ &= (1+1/m)\left[\frac{\Gamma(k+1)\Gamma(2+1/m)}{\Gamma(k+1+2+1/m)}\right] \\ &= (1+1/m)\beta(k+1, 2+1/m) \end{aligned}$$

where the Gamma function, $\Gamma(x+1) = x\Gamma(x)$, and the Beta function, $\beta(a,b) = \Gamma(a)\Gamma(b)/\Gamma(a+b)$. In the asymptotic limit, $\Gamma(a,b) \approx a^{-b}$. Hence we have the following power law,

$$p_k = (1 + \frac{1}{m})(k+1)^{-(2+1/m)}$$

7.3 Non Power Law Models

Every network in the real world does not follow the Power Law. The most striking example of such a real world network is the World Wide Web, or the internet.

Pennock et. al. mention that as a whole, the World Wide Web displays a striking "rich get richer" behavior, with a relatively small number of sites receiving a disproportionately large share of hyperlink references and traffic. However, hidden in this skewed global distribution, one can discover a qualitatively different and considerably less biased link distribution among subcategories of pages. For example, among all university homepages or all newspaper homepages.

While the connectivity distribution over the entire web is close to a pure power law, we find that the distribution within specific categories is typically unimodal on a log scale, with the location of the mode, and thus the extent of the "rich get richer" phenomenon, varying across different categories. Similar distributions occur in many other naturally-occurring networks, including research paper citations and movie actor collaborations.

Here we outline a model to explain the existence of such log-normal distributions for some real world networks.

- Start with a small number of nodes, say m_0
- A new node joins in the system in each step. Also, m edges are also introduced along with this new node.

It is important to note that the above procedure is not the same as the Barabási-Albert model. The difference between these two is illustrated by the means of the following figure.

Figure 7.1: Describes the working of the Barabási-Albert model, compared with the model to generate Log-Normal distributions.

The probability that any introduced edge sticks to the node i is given by

$$\Pi(k_i) = \frac{k_i \alpha}{\sum_i k_i} + (1-\alpha)\frac{1}{t+m_0}$$

where $\sum_i k_i = 2mt$ and t is the number of timesteps. The first term on the right hand side is the contribution due to preferential attachment, whereas the second term represents a random choice of any of the nodes. That is, α represents the probability for preferential attachment, whereas $1-\alpha$ represents the probability of uniform attachment. It can also be seen that $m_0 + t$ is the total number of vertices, and $2mt$ is the total connectivity at time t. Substituting for k_i, we get that:

$$\Pi(k_i) = \frac{k_i \alpha}{2mt} + (1-\alpha)\frac{1}{t+m_0}$$

This model however applies only to undirected graphs. To capture directionality, we define parameters α_{in} and α_{out}, such that

$$\Pi(k_j) = \frac{k_j \alpha_{in}}{mt} + (j-\alpha_{in})\frac{1}{t+m_0}$$
$$\Pi(k_i) = \frac{k_i \alpha_{out}}{mt} + (i-\alpha_{out})\frac{1}{t+m_0}$$

Using the above defined model, we can now explain the log-normal distribution obtained in the linked network between subcategories of pages. Applying the continuum theory equations to the above equations we get

$$p_k \propto k^{-(1+1/\alpha)}$$

7.4 Configuration Models

Random graphs can be extended to make them more realistic. Real graphs do not follow a poisson degree distribution, which leads us to the **configuration model**.

The model is defined in the following way. We specify a degree distribution p_k, such that p_k is the fraction of vertices in the network having degree k. We also choose a degree sequence, which is a set of n values of the degree k_i of vertices $i = 1, 2, \cdots, n$, from this distribution. This essentially gives us each vertex i in the graph k_i, with "spokes" sticking out of it, which are the ends of edges-to-be. Pairs of spokes are then chosen at random, and connected together to obtain the graph. The *configuration model* is defined as the ensemble of the graphs so produced, with each having equal weight.

Figure 7.2: Randomly pick up spokes and join together to get the graph.

There are a few important points to grasp about this model. Firstly, p_k, in the limit of the large graph size, is the distribution of degrees of vertices in our graph, but the degree of the vertex we reach by following a randomly chosen edge on the graph is not given by p_k. Since there are k edges that are incident to a vertex with degree k, we are k times as likely to be at a vertex with degree k, than to be at a vertex with degree 1. Thus the degree distribution at the end of a randomly chosen edge is kp_k.

We are generally interested in the *excess degree*, which measures how many edges leave the vertex other than the one we arrived along. So, the excess degree is one less than the total degree of the vertex. In the configuration model, the excess degree has a distribution q_k given by

$$q_k = \frac{(k+1)p_{k+1}}{\sum_k kp_k} = \frac{(k+1)p_{k+1}}{z}$$

where $z = \sum_k kp_k$ is the mean degree in the network.

Another important point about this model is that the chance of finding a loop in a small component of the graph goes as n^{-1}. The number of vertices in a non-giant compoenent is $O(n^{-1})$, and hence the probability of there being more than one path between any pair of vertices is also $O(n^{-1})$.

7.5 Small-World Model

A **Small-World network** is a type of mathematical graph in which most nodes are not neighbors of one another, but most nodes can be reached from every other node by a small number of hops or steps. Specifically, a small-world network is defined to bea network where the typical distance L between two randomly chosen nodes grows proportionally to the logarithm of the number of nodes N in the network,

$$L \propto \log N$$

7.5.1 Construction

This model was proposed by *Watts and Strogatz*. Small-world networks start by positioning a network built on a low-dimension regular lattice, and then adding or moving edges to create "shortcuts" that join remote parts of the lattices to each another.

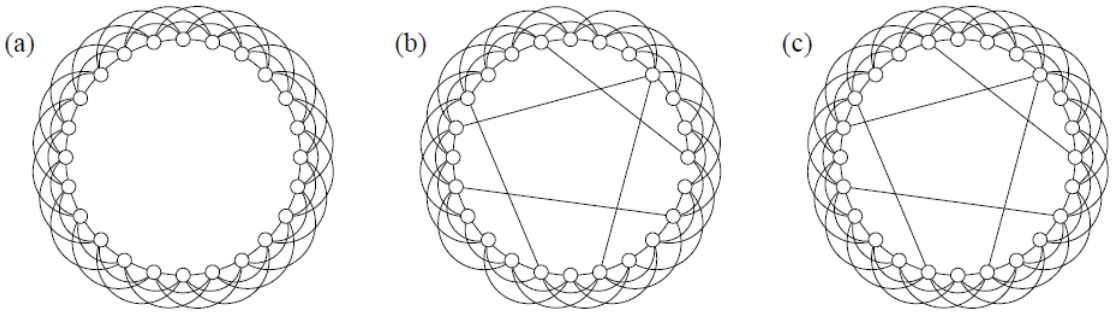


Figure 7.3: (a) A one-dimensional lattice with connections between all vertex pairs separated by k or fewer lattice spacing. (b) Choose at random a fraction p of the edges in the graph and move one end of each to a new location chosen randomly. (c) Shortcuts are added randomly between vertices, but no edges are removed from the underlying one-dimensional lattice.

Consider the case of a one-dimensional lattice of L vertices with a ring and each vertex joined to neighbours k or fewer lattice spacings away. Such a lattice is shown in Figure 7.2(a), with a total of Lk edges. The small-word model is then created by taking all edges in this graph, and with a probability p, moving one end of that edge to a new location chosen at random, such that no double edges or self-loops are created. This is shown in Figure 7.2(b).

This "rewiring" process allows the small-world network to interpolate between a regular lattice and a random graph, based on the value of p. When $p = 0$, we have a regular lattice, with a clustering coefficient of $C = (3k - 3)/(4k - 2)$, which tends to 3/4 for large k. The mean geodesic distance tends to $L/4k$ for large L, and thus it does not show the small-world effect. For $p = 1$, each edge is rewired to a new location, and the resulting graph is almost a random graph, with typical geodesic distances of the order of $\log L/\log k$, but with low clustering coefficient of $C \approx 2k/L$. There exists a sizeable region between these two extremes for which the model has both low path lengths and high transitivity.

A modified version of the Watts-Strogatz model was proposed by Newman and Watts. In this variant, no edges are rewired, instead, "shortcut" edges between randomly chosen pairs of vertices are added to the low-dimension lattice. The mean total number of shortcuts is Lkp, and the mean degree is $2Lk(1 + p)$. This version of the model also has the property that no edge can ever become disconnected from the network, and hance the mean vertex-vertex distance is always finite. This version is shown in Figure 7.2(c).

7.5.2 Properties

In the following we will summarize the main results regarding the properties of small-world models.

7.5.2.1 Clustering Coefficient

The clustering coefficient for the Watts Strogatz model can be calculated easily, and is numerically given by

$$C = \frac{3(k-1)}{2(2k-1)}(1-p)^3$$

7.5.2.2 Degree Distribution

The degree distribution of small-world model does not match most real-world networks very well, but this is not surprising, as this was not a goal of the model in the first place. The expression for degree distribution is rather complicated and given by

$$p_j = \sum_{n=0}^{\min(j-k,k)} \binom{k}{n}(1-p)^n p^{k-n} \frac{(pk)^{j-k-n}}{(j-k-n)!} e^{-pk}$$

7.5.2.3 Average Path Length

There does not exist an exact solution to the value of the average path length l, however, small-world behaviour is exhibited when $l \propto \log L$.

7.6 Vertex Copying Models

Kleinberg *et al.* proposed that many real networks grow, at least in part, by the copying of vertices. We proceed to describe the basis of such a model next.

7.6.1 Trawling the Web for Cyber-Communities

Algorithms such as the HITS algorithm, essentially are search algorithms designed to find high-quality pages (authorities) about a fixed *topic*. How should one extend such algorithms if the need is to enumerate all topics (under a certain definition)?

7.6.1.1 Definition

We first need to carefully define what is meant by a topic in the web graph. Note that each topic present in the web graph structure can be detected by looking for complete bipartite graphs $K_{i,j}$ in the graph (see Figure 7.3 for an example).

Define a bipartite core $C_{i,j}$ to be a graph on $i+j$ nodes, such that it contains $K_{i,j}$ as a subgraph. This notion is motivated by the fact that for any well represented topic on the Web, for appropriate values of i and j, there will exist a bipartite core for it on the Web graph. Figure 7.3 shows $C_{4,3}$ in which the nodes on the left connect to the home pages of major aircraft manufacturers. Subgraphs like these represent cyber-communities. In the given case, the nodes on the left represent aficionados of commercial aircraft manufacturers. These nodes create hub-like pages, which co-cite the authority-like pages on the right.

It is important to note that the hub pages may not co-cite all the authority pages in such cyber-communities, but, every such community will contain a bipartite core $C_{i,j}$ for non-trivial values of i and j. This process of enumerating all possible bipartite cores in the web for some fixed value of i and j is called **trawling**.

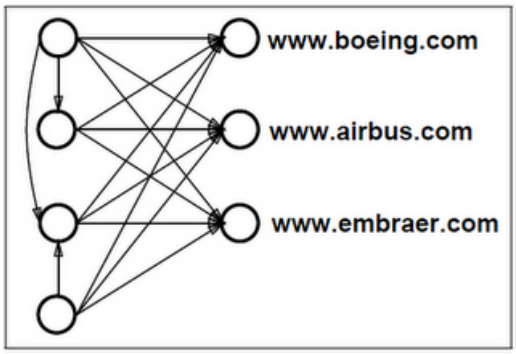


Figure 7.4: An example of $C_{4,3}$

7.6.1.2 The Elimination-Generation Algorithm

An algorithmic paradigm called the **elimination-generation** paradigm is used for the purpose of trawling. The algorithm makes a number of passes over the Web graph. The paradigm is explained below in the context of the core $C_{4,4}$.

- **Elimination:** For $C_{4,4}$, no nodes with indegree ¡ 4 are allowed in the right hand side of the core. Similarly, no nodes with outdegree ¡ 4 are allowed on the left hand side. All such nodes are pruned from the graph.
- **Generation:** For $C_{4,4}$, a node u is termed as *barely acceptable / qualified* if its indegree = 4. Find all such u's. If the 4 nodes pointing to u have a neighborhood intersection of size 4, then u belongs to $C_{4,4}$.

The above mentioned algorithm makes several passes through the web graph, eliminating nodes in each pass. In most of the experiments, the benefits of elimination / generation tail off as fewer and fewer nodes are eliminated at each phase, and the algorithm stops after some time.

7.6.2 The Model of Kleinberg *et al.*

The structures described in the previous section appear to be a fundamental by-product of the manner in which Web content is created. In this section, we describe a framework for a class of random graph models in which the presence of these structures is established.

7.6.2.1 Characterization of the Model

The model is characterized by four stochastic processes:

1. **Vertec creation V_c:** At each step, a node is created with probability $\alpha_c(t)$.
2. **Vertex deletion V_d:** At each step, delete a node with probability $\alpha_d(t)$ along with all the edges incident on the node.
3. **Edge creation E_c:** At each step, randomly select a node v and a constant m denoting the number of edges to be added. With probablility β add m edges from v to nodes chosen independently and at random. Alternatively, with probablity $1-\beta$ select any node u and *copy* its out-links to v. In this case, if outdegree$(u) < m$, then choose other connections for v randomly. If outdegree$(u) > m$, then randomly choose a subset of the edges of u.

4. **Edge deletion E_d:** At each step, delete a randomly chosen node with probability δ. Note that this is essentially a replica of the node deletion process.

7.6.2.2 Analysis

Let us analyze the cases under which a node i gains an edge. This can happen in the following two ways:

- i is the randomly chosen node. Let P_{uni} be the probability with which this event occurs.
- i is an indegree neighbor of u, the vertex selected vertex. Let P_{copy} denote the probability with which this occurs.

Assume that there are n nodes in the system at this step. Then we have the following observations:

$$P_{uni} = \frac{m(1-\beta)}{n}$$
$$P_{copy} = \frac{\beta q_i}{n}$$

where q_i is the indegree of i in the second event. Therefore, the total probability with which a node i gains an edge is given by

$$\begin{aligned} P_{total} &= P_{uni} + P_{copy} \\ &= \frac{m(1-\beta)}{n} + \frac{\beta q_i}{n} \end{aligned}$$

Let $p_{q,n}$ denote the fraction of nodes with indegree q when there are n nodes in the system. Then, the number of nodes of indegree $q = np_{q,n}$. The expected no. of nodes of indegree q gaining an edge is given by

$$\begin{aligned} E &= np_{q,n} \times P_{total} \\ &= np_{q,n} \times \frac{m(1-\beta) + \beta q_i}{n} \end{aligned}$$

Observe that for the case that $i = 0$ (i.e. it is the first node, and $n = 1$), we have that

$$\begin{aligned} q_0 &= m(\frac{1}{\beta} - 1) \\ \Rightarrow \beta &= \frac{m}{q_0 + m} \end{aligned}$$

Thus,

$$\begin{aligned} E &= np_{q,n} \times \frac{1}{n}\left[q_i \frac{m}{q_0+m} + m\left(1 - \frac{m}{q_0+m}\right)\right] \\ &= p_{q,n} \times \frac{m(q_i + q_0)}{q_0 + m} \end{aligned}$$

Chapter 8

Search on Networks - Distributed Hash Tables

A number of methods can be employed for performing search on networks, some examples being PageRank, HITS, and the Elimination-Generation model. However, such techniques are unsuitable for peer-to-peer networks because of the follwoing reasons - there is no synchronized source, and all nodes are equivalent in functionality (in the sense that any node can act as the client or server). Therefore, a fundamental problem that confronts peer-to-peer applications is to efficiently locate a node that stores a particular data item. The problem is solved by making use of distributed hash tables.

A **distributed hash table** (DHT) is a class of a decentralized distributed system that provides a lookup service similar to a hash table; (key, value) pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows a DHT to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures. An example of such a service is **Chord** - a scalable P2P lookup protocol for internet applications.

8.1 Overview on Hash Functions

Simply put, a **hash function** is any algorithm or subroutine that maps data sets of variable length, called *keys*, to data sets of a fixed length. The values returned by a hash function are called hash values, or simply hashes.

Hash functions are primarily used in hash tables, to quickly locate a data record given its search key. Specifically, the hash function is used to map the search key to the hash. The index gives the place where the corresponding record should be stored. In general, a hashing function may map several different keys to the same index. Additionally, a good hash function f must satisfy the following properties - computation of the inverse function f^{-1} should be difficult, and the function should be collision-free.

Like most hashing schemes, **consistent hashing** assigns a set of items to buckets so that each bucket receives almost the same number of items. But unlike standard hashing schemes, a small change in buckets does not induce a total remapping of items to bucket.

By using consistent hashing, only K/n keys need to be remapped on average, where K is the number of keys, and n is the number of buckets.

8.2 The Chord Protocol

A distributed hash table stores key-value pairs by assigning keys to different computers (or nodes); a node will store the values for all the keys for which it is responsible. Chord specifies how keys are assigned to nodes, and how a node can discover the value for a given key by first locating the node responsible for that key.

8.2.1 Construction

The Chord protocol assigns each node and key an m-bit *identifier* using a base hash function such as SHA-1. This is done as follows:

- Each node (or computer) is mapped to an identifier by hashing the IP address of the node. The mapping is done using the SHA-1 hashing algorithm.
- Each key (or data item) is also hashed using SHA-1 to get an identifier.

Note that both the above sets of identifiers belong to the same identifier space. Also, the identifier length m must be large enough so that the probability of two nodes or keys hashing to the same identifier is negligible.

Next, keys are assigned to nodes as follows.

- All possible identifiers are arranged in an *identifier circle* modulo 2^m. Because the identifiers are of length m-bits, there can be 2^m possible values, ranging from 0 to $2^m - 1$.
- Key k is assigned to the first node whose identifier is equal to or follows (the identifer of) k in the identifer circle. This node is denoted as $successor(k)$.

Note that $successor(k)$ is the first node clockwise from k in the identifier circle, including k. Figure 8.1 shows an identifier circle with $m = 3$. The circle has three nodes: 0, 1, and 3. The successor of identifier 1 is node 1, so key 1 would be located at node 1. Similarly, key 2 would be located at node 3, and key 6 at node 0.

Consistent hashing (as in SHA-1) is designed to let nodes enter and leave the network with minimal disruption. When a node n joins the network, certain keys previously assigned to n's successor now become assigned to n. When node n leaves the network, all of its assigned keys are reassigned to n's successor. No other changes in assignment of keys to nodes need occur. In the example above, if a node were to join with identifier 7, it would capture the key with identifier 6 from the node with identifier 0.

8.2.2 Search

The Chord protocol requires each node to keep a *finger table*, containing up to m entries. The i^{th} entry of node n will contain the address of $successor((n + 2^{i-1}) \mod 2^m)$, where $1 \leq i \leq m$. With such a finger table, the number of nodes that must be contacted to find a successor in an N-node network is $O(\log(N))$. As an example, figure 8.2 shows the finger tables for the nodes 0, 1 and 3.

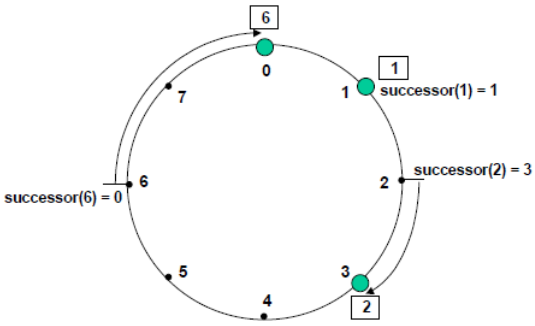


Figure 8.1: An identifier circle consisting of three nodes 0, 1, and 3. In this example, key 1 is located at node 1, key 2 at node 3, and key 6 at node 0.

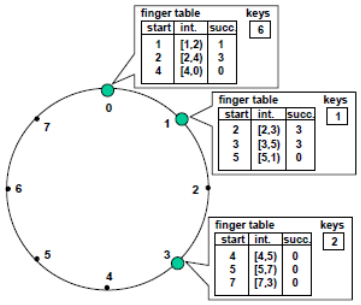


Figure 8.2: Finger tables and key locations for a network with nodes 0, 1, and 3, and keys 1, 2, and 6.

Chapter 9

Network Dynamics

9.1 Protein Networks

This section deals with the use of graph theory to investigate protein structure and dynamics. Networks appear at all scales in biology, from the microscopic networks in case of proteins, to food webs involving a variety of species. A protein in its unfolded form essentially contains only covalent interactions amongs atoms, but an active protein which has folded into a proper structure, forms a network of non-covalent interactions (*links*) between amino acids (*nodes*).

To model such networks, we can use a regular lattice or grid, having high clustering, but a large average path length, or random networks from graph theory, having low average path length, and low coefficient of clustering. However, most empirical networks are not random, and possess certain structural patterns, eg. Small-World networks.

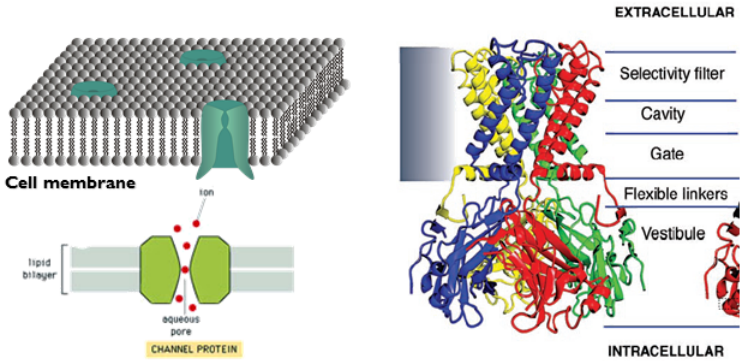


Figure 9.1: Kirbac 1.1 Potassium ion Channel Protein

We consider the example of **Kirbac1.1** Potassium ion channel protein. It comprises of 4 identical sub-units, which interact amongst each other to generate the protein network. The protein contact network is constructed using structural data from the Protein Database (PDB). Using this data, we can calculate the Euclidean distance between each pair of amino acids $P = (p_x, p_y, p_z)$ and $Q = (q_x, q_y, q_z)$ as $\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2}$. These distances can then be used to obtain the distance matrix, and the adjacency matrix.

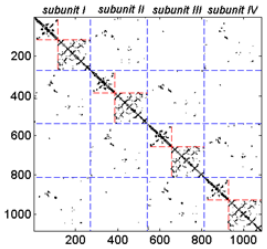


Figure 9.2: The adjacency matrix for the protein contact network

The magnification of the adjacency matrix for a sub-unit reveals an underlying modular structure present in the network. The degree distribution of this network is not scale-free, but high degree nodes (hubs) do exist.

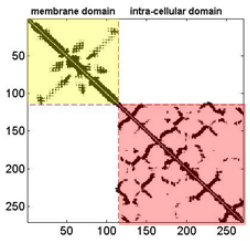
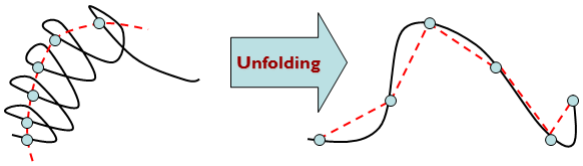


Figure 9.3: Magnification of a sub-unit reveals a modular structure

We observe an important feature in the protein network. The protein contact network **is a small-world network**, as it has low average path length and high clustering coefficient. The genesis of small-world nature is from the existence of cross-links as a result of the folding of the protein. These cross links are essentially van der Waals forces of attraction between nearby atoms. These cross links are also functionally important, as they provide structural stability to the protein.



9.1.1 Protein Dynamics from Network Analysis

We consider a protein as an elastic network of balls (C-α atoms), that are connected together by springs (chemical interactions). Now, under the harmonic potential approximations, we get

$$V(x) \approx V(x = x_0) + \frac{1}{2}(x - x_0)^2 \frac{\partial^2 V}{\partial x^2} + \cdots$$

Potential energy of the network, V is $\frac{k}{2}\sum_{i,j=1,2,\cdots N}(R_{ij} - R^0_{ij})^2$, k being the force constant. This can be written as $V = \frac{k}{2}\sum_{i,j=1,2,\cdots N}(\Delta R_i - \Delta R_j)^2$, where $R_{ij} = R_i - R_j = (R^0_{ij} + \Delta R_i - \Delta R_j)$.

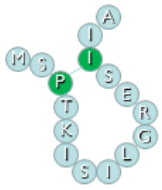


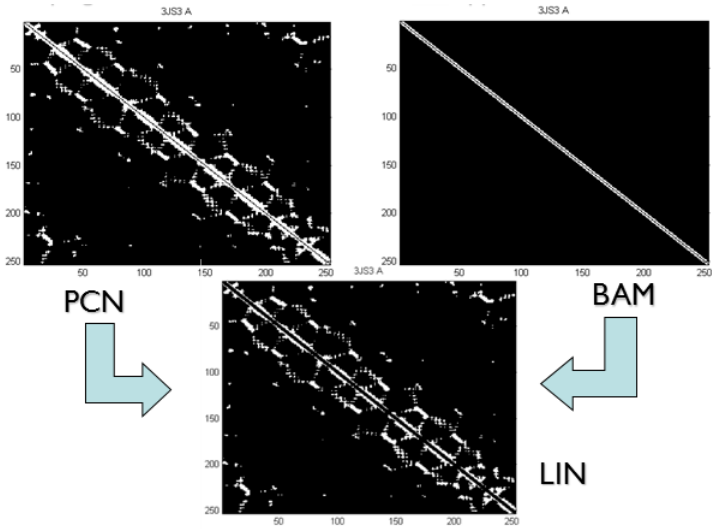
Figure 9.4: $d_{\text{cumulative}}(M, P) = d(M, S) + d(S, P)$

In vector notation, the potential energy of the network can be written as $V = \frac{k}{2}(d\mathbf{R})^{\mathbf{T}}\mathbf{L}(d\mathbf{R})$, where $d\mathbf{R}$ is the column vector of fluctuations, or displacements from the equilibrium, and $\mathbf{L}$ is the *Laplacian* or *Kirchoff* matrix. The entries of this matrix are

$$L_{ij} = \begin{cases} 0 & \text{if } d_{ij} \text{ ¿ cut-off} \\ -1 & \text{if } d_{ij} \text{ ¡ cut-off} \\ \text{degree(i)} & \text{for } i = j \end{cases}$$

The vibrational normal modes of the protein are governed by the eigenvalues of $\mathbf{L}$, and a small eigenvalue implies greater large-scale motion. This can be seen in case of the Kirbac 1.1 protein. The spectrum of eigenvalues for $\mathbf{L}$ shows 4 very small eigenvalues, which indicate the dominance of largest scale motion by these 4 sub-units. Other large scale motions are possible dominated by the modular structure of the network. Eigenvector components corresponding to the smallest eigenvalues indicate how the module motions are correlated.

The Protein Contact Network also contains links that correspond to the backbone structure of the protein, and does not give us much information about the folded tertiary structure of the protein. To focus on the cross-links, we need to compute the **Long-Range Interaction Network** (PCN), obtained from the PCN by excluding links among spatially neighbouring nodes along the backbone.



The first step in constructing the LIN is to obtain the **Cumulative Distance Matrix** (CDM), which carries the euclidean distances between all pairs of C-α atoms. An example for this is shown in Figure 9.4. The CDM is then used to obtain the **Backbone Adjacency Matrix** (BAM), from the CDM by retaining only those links which correspond to Euclidean distance $< 10\mathring{A}$. Finally, the LIN is obtained from the PCN by

keeping those links which do **not** appear in the BAM.

Using PCN and LIN, we can now identify the network core of the proteins, which can contain functionally critical residues. We notice that many networks possess a core-periphery organization, as shown. We can use a technique, called *k-Core Decomposition*, to obtain the fundamental structural organization of a complex network through a process of successive pruning. An iterative procedure to determine the k-core (subnetwork containing all nodes that have degree atleast equal to k) is as follows:

1. To remove all nods having degree less than k.

2. Check the resulting network to see if any of the remaining nodes now have degree less than k as a result of step 1, if so,

3. Repeat steps 1 and 2 untill all remaining nodes have degree at least equal to k.

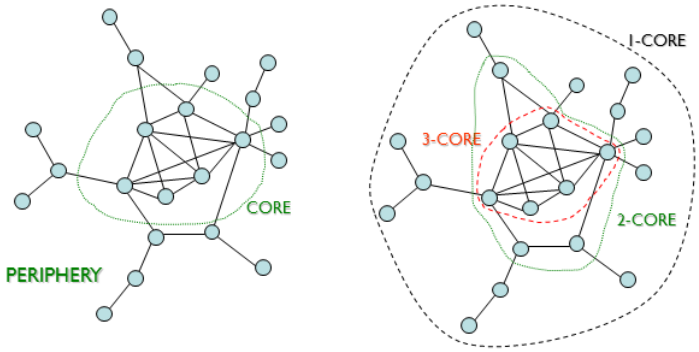


Figure 9.5: Network showing the core periphery organization, and it's k-core decomposition

Residues belonging to the core of a protein are functionally important, eg. as ligand binding sites or for imparting structural stability. These cores are also relevant for pharmaceutical treatment of infectious diseases.

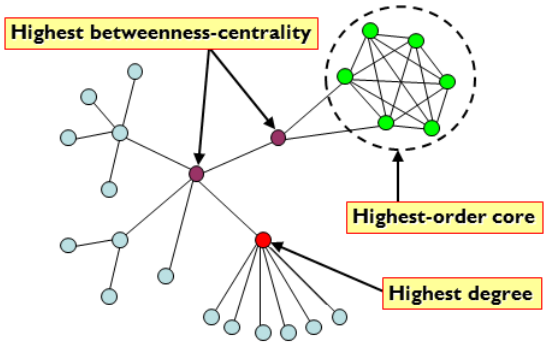


Figure 9.6: A comparison of different node specific measures of importance

Figure 9.7: k-Core Decomposition of a protein

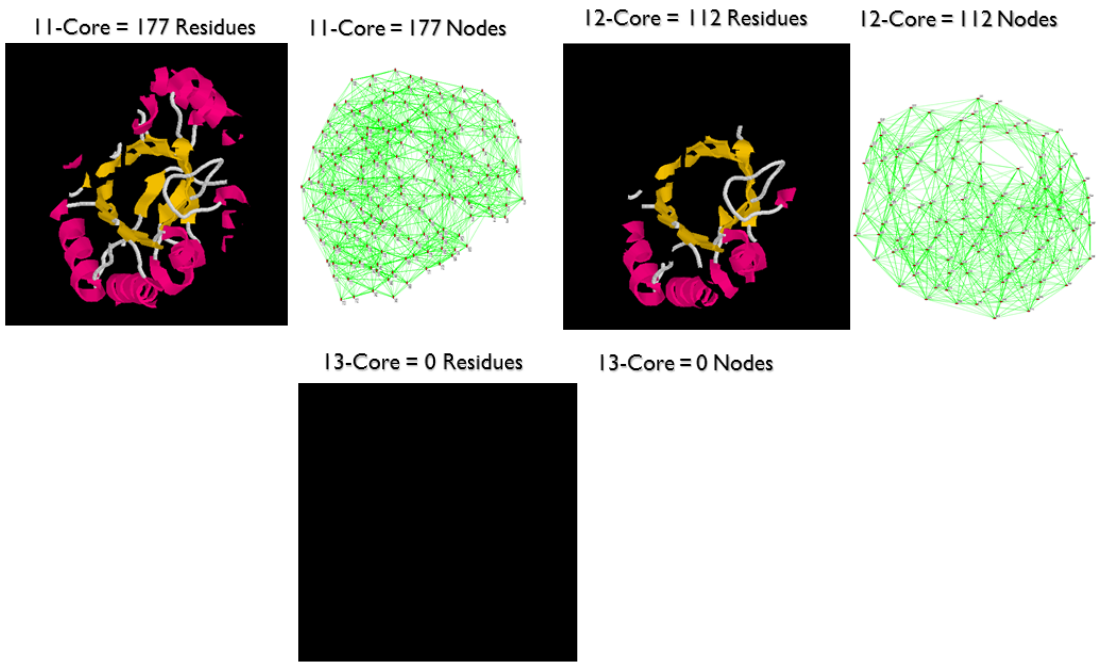


Figure 9.8: k-Core Decomposition of a protein (contd.)

9.2 Percolation Theory

A **percolation** process is one in which vertices or edges on a graph are randomly designated either "occupied" or "unoccupied" and one asks about various properties of the resulting patterns of vertices. One of the main motivations for the percolation model when it was first proposed in the 1950s was the modeling of the spread of disease. We will first discuss its simple application to the question of network resilience.

Real-world networks are found often to be highly resilient to the random deletion of their vertices. Resilience can be measured in different ways, but perhaps the simplest indicator of resilience in a network is the variation (or lack of variation) in the fraction of vertices in the largest component of the network, which we equate with the giant component in our models. If one is thinking of a communication network, for example, in which the existence of a connecting path between two vertices means that those two can communicate with one another, then the vertices in the giant component can communicate with an extensive fraction of the entire network, while those in the small components can communicate with only a few others at most. The problem of resilience to random failure of vertices in a network is equivalent to a **site percolation** process on the network, while that of failure of edges is equivalent to **bond percolation**. Vertices are randomly occupied (functioning) or unoccupied (failed), and the number of vertices remaining that can successfully communicate is precisely the giant component of the corresponding percolation model.

9.2.1 Uniform Random Removal of Nodes

Let ϕ denote the occupation probability, which is the probability that a node is present in the network (functioning). When $\phi = 0$ then none of the vertices are present (or functioning). When $\phi = 1$ then all the vertices are present.Also, let ϕ_c denote the *critical* value of ϕ at which a *percolation transition* occurs, and the network transforms from containing a

giant cluster to one without a giant cluster.

Consider the configuration model with degree distribution p_k. Let u denote the average probability that a node is not connected to the giant cluster via a neighbor.

- If a node has degree k, then the total probability that it is not connected to the giant cluster is u^k.
- The average probability of not being connected to the giant cluster for any node = $\sum p_k u^k = g_0(u)$, where g_0 is a generating function.
- The average probability that a vertex does not belong to a giant cluster for any node = $1 - g_0(u)$.
- Total fraction of the network belonging to the giant cluster is given by $S = \phi[1 - g_0(u)]$.

We can calculate u as follows. For a vertex, there are 2 ways of being connected to the giant cluster via a node x.

1. x has been reomoved (probability $1 - \phi$)
2. x has not been removed but x does not connect to the giant cluster through any of its k neighbors (probability ϕu^k)

The probability that the vertex is not connected to the giant cluster via node x = $1 - \phi + \phi u^k$.

The probability that the neighboring vertex is connected to k other nodes is denoted by the excess degree distribution q_k.

Let L be the total no. of links in the network. Then $2L = \sum k_i$. The probability that the vertex has an edge at a particular vertex of degree k is $k/(2L-1) \approx k/2L$ for large L. The total number of vertices of degree $k = Np_k$. Then, the probability of attaching to any vertex of degree k

$$\begin{aligned} &= Np_k \frac{k}{2L} \\ &= \frac{p_k k}{\langle k \rangle} \end{aligned}$$

The excess degree of a vertex reached by travelling along an edge is equivalent to the probability of arriving at a vertex with degree $k+1$. Thus,

$$q_k = \frac{p_{k+1}(k+1)}{\langle k \rangle}$$

Averaging over the degree distribution, we have that

$$\begin{aligned} u &= \sum_{k=0}^{\infty} q_k (1 - \phi + \phi u^k) \\ &= 1 - \phi + \phi \sum_{k=0}^{\infty} q_k u^k \\ &= 1 - \phi + \phi g_1(u) \end{aligned}$$

where $g_1(u) = 1 - \sum_{k=0}^{\infty} q_k u^k$. Note that $g_1(u)$ is non-negative because the coefficients are positive. Also, it is an increasing function as the derivatives are non-negative.

A trivial solution for u then is $u = 1$ (or $S = 0$) for which a giant cluster does not exist. For $u \neq 1$, a solution exists implies that a giant cluster exists.

We calculate the percolation threshold ϕ_c as follows:

$$\frac{d}{du}(1 - \phi + \phi g_1(u))\bigg|_{u=1} = 1$$
$$\Rightarrow \phi_c = \frac{1}{g_1'(1)}$$

Now,

$$\begin{aligned} g_1(u) &= \sum_{k=0}^{\infty} q_k u^k \\ &= \sum_{k=0}^{\infty} \frac{(k+1)p_{k+1}}{\langle k \rangle} u^k \\ \Rightarrow \frac{d}{du} g_1(u)\bigg|_{u=1} &= \frac{1}{\langle k \rangle} \sum_{k=0}^{\infty} k(k+1)p_{k+1} \\ &= \frac{1}{\langle k \rangle} \sum_{k=0}^{\infty} k(k-1)p_k \\ &= \frac{1}{\langle k \rangle} \left[\sum_{k=0}^{\infty} k^2 p_k - \sum_{k=0}^{\infty} k p_k \right] \end{aligned}$$

Therefore, we have that

$$\phi_c = \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle}$$

A network is **structurally robust** if it can tolerate the loss of a large fraction of its vertices without significant decline in function, i.e. if it has very low ϕ_c, which can be obtained by making $\langle k^2 \rangle$ large.

Therefore, for dynamical stability we need to minimize k_{max} and maximize $\langle k^2 \rangle$.

9.2.2 Non-Uniform Random Removal of Nodes

In non-uniform removal, the occupation probability of a vertex depends on its degree k, and is denoted as ϕ_k. Then the probability that a vertex of degree k is part of the giant cluster is $= \phi_k(1 - u^k)$. Thus, the total fraction of vertices belonging to the giant cluster is given by

$$\begin{aligned} S &= \sum_{k=0}^{\infty} p_k \phi_k (1 - u^k) \\ &= \sum_k p_k \phi_k - \sum_k p_k \phi_k u^k \\ &= f_0(1) - f_0(u) \end{aligned}$$

where $f_0(u) = \sum_k p_k \phi_k u^k$, and $f_0(1) = \sum_k p_k \phi_k = \phi'$, the average probability that a vertex is occupied.

The average probability u that a vertex won't belong to the giant cluster via a neighbor is

$$\begin{aligned} u &= \sum q_k (1 - \phi_{k+1} + \phi_{k+1} u^k) \\ &= 1 - f_1(1) + f_1(u) \end{aligned}$$

where $f_1(u) = \sum q_k \phi k + 1 u^k = f_0'(u)/g_0'(1)$.

9.3 Epidemic Networks

The study of epidemic disease has always been a topic where biological issues mix with social ones. An Epidemic model is a simplified means of describing the transmission of communicable disease through individuals. Epidemic models can be described on multiple scales:

- **Homogeneous Mixing**: No consideration of spatial or social structure in a community
- **Social Structure**: Individuals grouped according to demographic properties such as age, gender, etc.
- **Contact Networks Models**: Agents affect those who are in direct contact with them
- **Multi-scale Models**: Spreading between communities in different spatial locations
- **Agent-Based Models**: Detailed consideration of spatial movements and interpersonal contact

9.3.1 The Deterministic SIR Model

Consider modeling the spread of an epidemic by direct contact. The model assumes that an individual node in the network goes through three potential stages during the course of the epidemic.

1. **Susceptibility**: Let $S(t)$ denote the susceptible population at time t, and have not yet been infected
2. **Infection**: Let $I(t)$ denote the infected population at time t, who are capable of spreading the disease to the susceptible population
3. **Recovery**: Let $R(t)$ denote the recovered population at time t, who have recovered from the disease and will neither be infected again nor be able to transmit the disease to others.

If N be the total size of the population, the $S + I + R = N$. Under the assumption of homogeneous mixing, i.e. anyone is likely to infect anyone else, we have the following rate equations:

$$\frac{dS}{dt} = -\beta SI$$
$$\frac{dI}{dt} = \beta SI - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$

where β = rate of infection spreading, γ = recovery rate = τ^{-1} (τ is defined as the average infectious period). Note that at $t = 0$, we have $S = N$, the total population. Also, the epidemic spreads if $dI/dt > 0$, i.e. $S(t = 0) > \gamma/\beta$.

9.3.2 Basic Reproduction Number

There is a threshold quantity which determines whether an epidemic occurs or the disease simply dies out. This quantity is called the basic reproduction number. Let $R(t)$ be the **effective reproduction number**, denoting the average number of new infections per infected person. Then the **basic reproduction number** R_0 is defined as

$$R_0 = R(t = 0) = N\beta\tau$$

R_0 represents the mean number of new infections caused by a single infectious individual in a wholly susceptible population (as in the beginning of an epidemic). If each infected person on average infects more than one other individual, then the epidemic will spread. Thus, we can say that the epidemic spreads if $R_0 > 1$; and it will die out in the long run if $R_0 < 1$.

We can estimate the value of R_0 using the following frequently used approach:

- Fit an exponential function to the incidence data to obtain the exponential growth rate λ, and,
- Use the approximate relation $R_0 \approx \exp(\lambda\tau_g)$, where τ_g is the observed mean generation interval of the epidemic (defined as "the sum of the average latent and the average infectious period").

For small $\lambda\tau_g$ we can further approximate $R_0 \approx 1 + \lambda\tau_g$.

9.3.3 Stochastic SIR dynamics

Transmission of infection and recovery are essentially stochastic processes. The deterministic SIR model does not account for fluctuations, particularly important at the beginning of an epidemic when I is small. The stochastic infection dynamics can be modeled as:

$$\text{Infection: } S + I \xrightarrow{\alpha} 2I \qquad \text{Recovery: } I \xrightarrow{\beta} 0$$

where α: infection rate, β: recovery rate. We are interested in the value of $p(S, I; t)$ - the probability of finding S susceptibles and I infected in a population of size N at time t.

The master equation for the evolution of this probability id given by:

$$\partial_t p(S,I;t) = \frac{\alpha}{N}(S+1)(I-1)p(S+1,I-1;t) + \beta(I+1)p(S,I+1;t) \\ - \left(\frac{\alpha}{N}SIp - \beta I\right)p(S,I;t)$$

with the initial condition as $p(S,I;t=t_0) = \delta_{I,I_0}\delta_{S,N-I_0}$. In the limit for large N, this approaches the deterministic SIR model results because the fluctuations decay as $\sqrt{N^{-1}}$.

9.3.4 Epidemic Spreading on Small-World Networks

In reality, social relations and physical / geographical proximity make some people more likely to be infected that others. Therefore, we need to go beyond the assumptions of homogeneous mixing. It becomes Important to consider either or both of the following effects in any realistic model of epidemic spreading:

1. Role of contact structure (spreading through a network), and

2. Role of space (diffusion of an epidemic front)

This can be done be considering SIR dynamics on Watts and Strogatz networks (small-world networks).

Epidemics spread much faster on WS networks than an equivalent regular network, and are far more difficult to control by partial removal of nodes than on a random network. This is because the presence of *shortcut* links make the transport fast. Unlike a random network, where every node is more or less equivalent, so that removing a certain fraction of susceptible agents (making $R < 1$) ensures the epidemic dies, here the few nodes that are terminals of shortcut links are principally responsible for rapid transit of infection. For efficient control of epidemic on WS networks, it is necessary to identify the shortcuts and preferentially control those.

9.3.5 Epidemic Spreading on Scale-Free Networks

Networks of sexual relations (in the context of STDs) have been claimed to be scale-free. A few highly promiscuous individuals act as *hub* nodes, and may play a crucial role in spreading sexually transmitted diseases.

If the contact structure of a disease is a network with an inhomogeneous degree distribution, then the condition for occurrence of an epidemic is given by:

$$R_0 = \frac{\beta N}{\gamma} > \frac{\langle k\rangle}{\langle k^2\rangle}$$

where R_0: basic reproduction number representing the epidemic threshold. For a scale-free network having degree exponent $2 < \alpha \leq 3$, $\langle k^2\rangle \to \infty$. This implies that there is no epidemic threshold. This makes scale-free networks especially vulnerable because Even diseases with extremely low transmission probabilities are likely to cause a major outbreak involving a signficant fraction of population.

9.4 Ecological Networks

An **ecological network** is a representation of the biotic interactions in an ecosystem, in which species (nodes) are connected by pairwise interactions (links). These interactions can be *trophic* or *symbiotic*. Ecological networks are used to describe and compare the structures of real ecosystems, while network models are used to investigate the effects of network structure on properties such as ecosystem stability.

9.4.1 Population Dynamics

Consider a system consisting of a single species. The population dynamics of this system taking into account intra-species competition can be represented as:

$$\frac{dx}{dt} = rx\left[1 - \frac{x}{K}\right]$$

where r is the Growth rate, and K is the carrying capacity, which is the maximum poputation that can be supported by the system. At equilibrium, for positive r, $x^* = K$, and for negative r, $x^* = 0$.

Consider mutual competition between the same species, using the same resources. This interaction can be modelled as

$$\frac{dx}{dt} = rx\left[1 - \frac{x}{K} - \frac{x'}{K}\right]$$

However, the interactions between two species can be of many kinds (and not just competition for the same resources). The trophic relations between two species can be of the following forms:

- Prey-Predator
- Host-Parasite
- Plant-Herbivore

9.4.1.1 Two-Species Resource Consumer Dynamics

The two species resource-consumer dynamics can in general be modeled as

$$\frac{dx}{dt} = \phi(x) - g(x,y)y$$
$$\frac{dy}{dt} = n(x,y)y - yd$$

Here, x and y refer to the number of prey and predators respectively, $\phi(x)$: growth of prey in absence of predators, $g(x,y)$: capture rate of prey per predator, $n(x,y)$: rate at which captured prey is converted into predator population increase – assumed to be $\epsilon g(x,y)$ where ϵ is efficiency, and d: rate at which predators die in absence of prey.

Different choices of the functional forms in the above equations give different models. We will study the Lotka-Volterra model in this context.

9.4.1.2 The Lotka-Volterra Model

The Lotka–Volterra equations, also known as the predator–prey equations, are used to describe the dynamics of biological systems in which two species interact, one a predator and one its prey. They evolve in time according to the pair of equations:

$$\frac{dx}{dt} = rx - k_{xy}xy$$
$$\frac{dy}{dt} = k_{xy}xy - yd$$

Consider the example of a two-species system of fishes, where x and y represent prey and predator fishes respectively. If the effect of fishing were to be additionally considered for modeling the population dynamics, then the equations would be as follows:

$$\frac{dx}{dt} = (r - a)x - k_{xy}xy$$
$$\frac{dy}{dt} = k_{xy}xy - (d + b)y$$

The equilibrium populations would then be $x^* = (d + b)/k_{xy}$, and $y^* = (r - a)/k_{xy}$. Note that in the absence of fishing, $a = 0$ and $b = 0$.

The Lotka-Volterra model suffers from chaos if it is used to model a system with more than 2 species.

9.4.1.3 Food Webs

So far, we have only considered interaction between 2 species. Going beyond 3 species opens up an entire world of dynamical possibilities, and we enter the realm of **ecological (interaction) networks**. Of these, **food webs** form a special case, comprising links representing trophic relations, i.e. only predator-prey interactions. While many different kinds of food webs are found in the real world (such as rainforests, lakes, deserts, etc.), it is possible to understand the general features of such networks using theoretical approaches.

9.4.2 Robustness of Complex Systems

Despite being complex systems, biological networks are robust, and stable against perturbations (such as infections, random fluctuations, targeted attacks). This is a vital property of complex networks. It is therefore important to understand how dynamics interacts with complex network structure to impart robustness to natural systems, and how such robust systems emerge.
If the complexity of a system were to be increased (by increasing the number of nodes, the density of connections, and the strenght of interaction between nodes), then theoretical results imply that stability of the system decreases. However, experimental observations sometimes show the opposite. It is important to note that most results were obtained assuming networks are random and at equilibrium (both at the level of the nodes as well as the network). There are two contrasting views in the literature regarding the relation between the diversity of a network and its stability.

9.4.2.1 The Empiricists' View

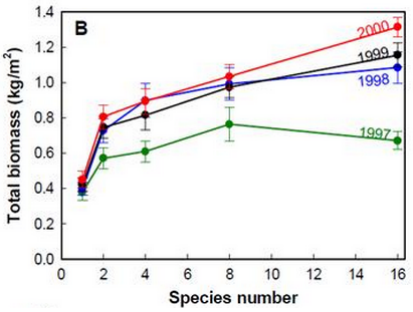
This view suggests that diversity is *essential* for maintaining network stability. *Charles Elton* proposed in 1958 that simple ecosystems are less stable than complex ones, on the basis of the field observations such as violent fluctuations in population density are more common in simpler communities, simple communities are more likely to experience species extinctions etc.

9.4.2.2 The Theorists' View

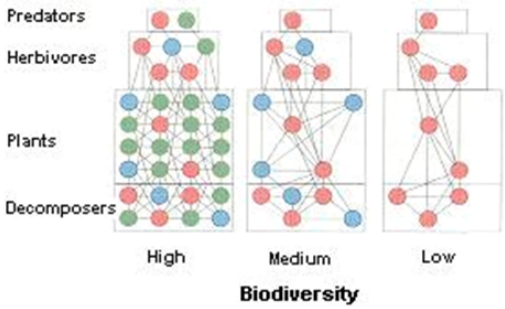
Despite the above field observations, the view was challenged by experiments on the stability of random networks by Garnder and Ashby (1970), and on the analysis of randomly constructed ecological networks by May (1972). These networks pointed to a contradictory view, stability decreases as network size, connectivity and interaction strength increases. In other words, increasing diversity leads to network instability.

The contradictory views laid the basis for the **Stability vs. Diversity** debate in ecology. Experiments conducted have challenged the theorists' views.

Common garden experiments examine the response of population and community level biomass to environmental perturbation. The outcome of the experiments is that diverse systems are more productive, as the total biomass produced increases with the number of species. With high diversity, the system is also more resistant, although there is no effect on population variability.



Bottle experiments examine network stability with varying biodiversity. Results suggest that with increasing biodiversity, the network is more stable, such as shown in the following figure.



Even though experiments suggest that high diversity communites are more productive than low diversity ones, it is still unclear how these results scale to real communities.

9.4.2.3 Mathematical Formulation - Theorists' View

Consider a simple community of one predator and one prey. We can then have the following rate equations

$$f_1 = \frac{dX}{dt} = X(a - bY)$$
$$f_2 = \frac{dY}{dt} = Y(-d + cX)$$

Taylor expansion around the equilibrium yields the following Jacobian matrix, also called the "community matrix".

$$J = \begin{bmatrix} a & -bd/c \\ ac/b & -d \end{bmatrix}$$

Such a system is stable if the largest real component of the eigenvalues, that is $\mathrm{Re}(\lambda_{max}) < 0$.

Robert May (1972) constructed randomly generated matrices representing interaction strenght in a network of N nodes whose isolated nodes are stable ($J_{ii} = -1$). He then obtained the eigenvalues λ, and using the criterion that if $\lambda_{max} > 0$, then the system is unstable, came to the observation that stability *decreases* as network size, connectivity and interaction strength *increases*.

The reasoning behind considering the network to be stable when the largest real component of the eigenvalues is less than 0 is as given here. Consider a N-dimensional vector $\mathbf{x}$, which represents the state of the network with N nodes. That is, $\mathbf{x} = (x_1, x_2, \cdots, x_N)$, where x_i is the state of the i^{th} node. The time evolution of $\mathbf{x}$ is given by a set of equations (eg. such as the Lotka-Volterra model),

$$\frac{dx_i}{dt} = f_i(\mathbf{x})$$

Now consider a fixed point equilibrium in the network state given by $\mathbf{x}^0 = (x_1^0, x_2^0, \cdots, x_N^0)$, such that $f(\mathbf{x}^0) = 0$. Now consider the effect of introducing a deviation δx at the equilibrium $\mathbf{x}^0$, to measure the local stability of the equilibrium point.

$$\begin{aligned} \frac{d(\delta x)}{dt} &= f(\mathbf{x}^0 + \delta x) \\ &= f(\mathbf{x}^0) + \delta x \left.\frac{df}{dt}\right|_{\mathbf{x}^0} + O(\delta x^2) \\ &= 0 + \delta x \left.\frac{df}{dt}\right|_{\mathbf{x}^0} \\ &= \lambda \delta x \\ \implies \delta x(t) &= \delta x(0) e^{\lambda t} \end{aligned}$$

Thus we see that the equilibrium point is locally stable if $\lambda < 0$, and unstable if $\lambda > 0$. But, what is the probability that for a network, $\lambda_{max} < 0$? Firstly, we need that each node is independently stable, that is, all the diagonal elements of $J < 0$ (choose $J_{ii} = -1$. Let $J = B - I$, where B is a matrix with diagonal elements 0 and I is the $N \times N$ identity

matrix. For matrix, what is the probability that $\lambda^B_{max} < 1$?

To answer this question, we apply random matrix theory. We assume that B has no particular structure, i.e. B is a random matrix. Also, we assume that B has connectance C, i.e $B_{ij} = 0$ with probability $1 - C$. The non-zero elements are independent random variables from a $(0, \sigma^2)$ Normal distribution. As the value of N grows larger, we can apply *Wigner's theorem* for random matrices.

Using this theorem, we have that the largest real part of the eigenvalues of B is $\lambda^B_{max} = \sqrt{NC\sigma^2}$. Thus we get that, for large N, the probability of stability is 0, when $\sqrt{NC\sigma^2} > 1$, while the system is *almost surely stable* if $\sqrt{NC\sigma^2} < 1$.

9.4.2.4 Other Measures of Stability

Apart from the measures as outlined above, we can also measure stability in terms of the following,

- **Global Stability**: A system is stable if it returns to equilibrium after any perturbation (large or small).
- **Resistance**: The ability of a community to resist change in the face of a potentially perturbing force.
- **Resilience**: The ability of a community to recover to normal levels of function after disturbance.
- **Variability**: The variation in population or community densitites over time. It is usually measure as the coefficient of variation (mean/variance).

9.4.3 Network Structure and Stability

In nature, networks are not random, and many have certain structural patterns eg. small world networks, scale free networks. Initial work done by Montoya and Sole (2001) regarding the network analysis of some food webs estimated that these networks have a high clustering coefficient, which is a characteristic of a small-world model. This view was however later challenged by Dunne et. al. (2002), who proposed that most food webs do not display typical small world topology. According to Sinha (2005), small-world topology **does not** affect the stability of a network.

Scale-free networks are generated using the Barabási-Albert Model. Montoya and Sole (2006) discovered that the Kyoto plant-pollinator web followed a power-law distribution. A simple Barabási-Albert network does not exhibit any correlations between the degrees of connected nodes, i.e. The probability a link connects nodes of degree k and k' is $p(k, k') = \frac{kp_k k' p_{k'}}{<k>^2}$, (degree-uncorrelated network). Most real-world networks though exhibit degree correlations measured using Assortativity,

$$Assortativity, r = \frac{\frac{1}{L}(\sum_{i=1}^{L} j_i k_i) - (\frac{1}{L}(\sum_{i=1}^{L} \frac{1}{2}(j_i + k_i)))^2}{\frac{1}{L}(\sum_{i=1}^{L} \frac{1}{2}(j_i^2 + k_i^2) - (\frac{1}{L}(\sum_{i=1}^{L} \frac{1}{2}(j_i + k_i))))^2}$$

Most biological networks are disassortiative in nature, that is, have $r < 0$, whereas social networks are assortative, and have $r > 0$. Brede and Sinha (2005) proposed that degree correlation infact **does** affect the stability of a network. Disassortative networks are relatively more stable than assortative networks. However, networks become more unstable as system size increases.

9.4.4 Modular Networks

Modular networks are characterized by dense connections within certain sub-networks (*modules*) and relatively few connections between different modules. Pimm proposed that such compartments or modules correspond to different habitats in the system. Later, May proposed that compartmentalization in ecosystems contributes towards their stability.

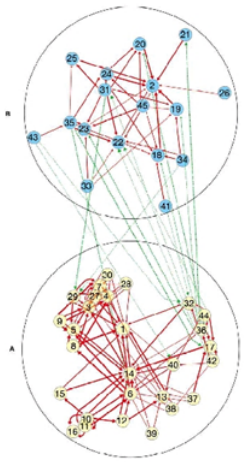


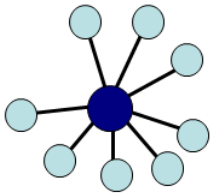
Figure 9.9: Chesapake Bay Foodweb

Consider the linear stability of a random network with a modular structure. As random networks are divided into more modules, they become more unstable. So, the suggestion that modularity imparts robustness is not entirely true. But still, most of the real-world networks that one observes have a modular structure. A clue to this puzzle is that many of these modular networks also possess multiple hubs, i.e. nodes having high degree relative to other nodes.

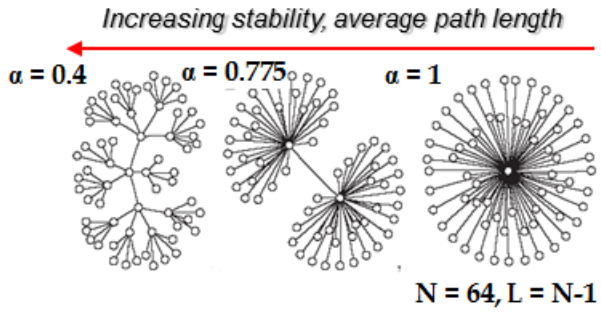
RK Pan (2007) noticed that real networks optimize between several constraints,

- Minimizing link cost, i.e Total number of links (L)
- Minimizing average path length, l
- Minimizing instability, λ_{max}

Minimizing link cost and average path length naturally yields a *star-shaped* network with a single hub. However, such networks are unstable, as, for the start network, λ_{max} $\sqrt{N}$. The simple star-shaped network thus obtained is unstable. In order to satisfy **all** three constraints, the solution lies in making the network more modular.



As star shaped networks are divided into more modules, they become more stable, as shown in the figure. As the number of modules increases, the instability decreases, as $\lambda_{max}\ \sqrt{N/m}$, where m stands for the number of modules in the network. This is also shown explicitly by network optimization. For this, we fix the link cost to a minimum value ($L = N - 1$), and then minimize the following energy function $E(\alpha) = \alpha l + (1 - \alpha)\lambda_{max}$, where α stands for the relative importance of path length constraint over stability constraint.



However, we note that perfect star-shaped networks are unrealistic. The link cost constraint is therefore relaxed, i.e. $L > N - 1$. This has the effect of introducing random links between non-hub nodes, and produces *clustered*-stars. As L increases, additional links occur between modules, and not within modules. In order to achieve an optimal network, we prefer inter-modular links between non-hub nodes, as this has the effect of decreasing l, while not increasing instability of the network.

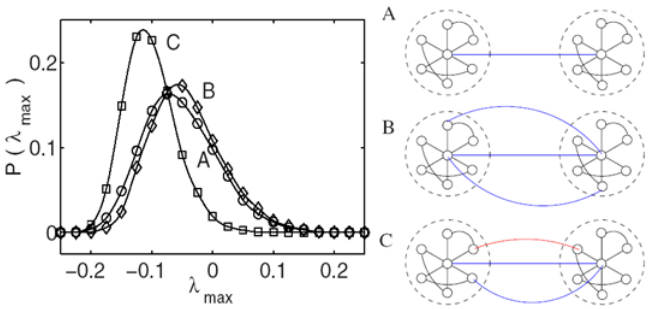


Figure 9.10: Relative stabilities of modular star-shaped networks with (A) no relaxed link cost constraint, (B) relaxed link cost with inter modular links between hub nodes, (C) relaxed link costs with intermodular links between non-hub nodes

Modular networks thus are stable with regard to the dynamical instability criterion for network robustness. In case of stability against structural perturbations, such as random or targeted removal of nodes, we know that scale-free networks are robust against random removal, and random networks are robust against targeted removal. In the limit

of extremely small L, we discover that optimal modular networks are actually equivalent to networks with bimodal degree distributions, and thus are robust with regard to **both** random as well as targeted removal of nodes.

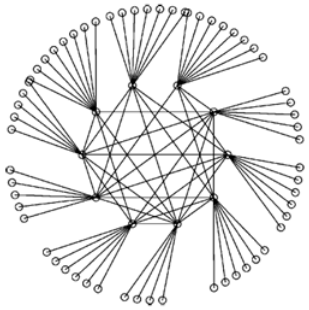


Figure 9.11: Robust modular star-networks in the limit of extremely small L

Introducing certain structures in the network topology does **not** change the relation between the complexity of a network and its resulting instablility.

9.4.5 Network Evolution

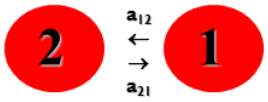
Networks in the real world do not occur fully formed, but gradually evolve over time. Examples of such networks are predator adaptation or prey switching, changing of trophic links between species due to changes in relative densities etc. The literature presented above did not yet consider the dynamics of networks. Now we present models which try to model such evolving networks, and then discuss their stability.

Consider the case of assembling ecological communities. Ecological networks are gradually organized over time by the community assembly rules, which decide which species and co-exist in a system, and the sequence in which species are able to colonize a habitat. The **WSB Network Assembly Model** gives an algorithm to construct such ecological networks.

9.4.5.1 Wilmers-Sinha-Brede (WSB) Network Assembly Model

The model outlines the following algorithm:

- Start with one node.
- Add another node with random number of links, and randomly chosen interaction strengths, a_{ij}.



- Check stability of the resultant network:
  - If unstable, remove a node at random, and analyze stability again.

– If stable, add another node.

We see that the networks generated using this model initially grow in size monotonically, but later settle into a pattern of growth spurts and collapses. A surprising outcome results, for the evolved networks, the more the complexity, the more the robust the networks. Larger networks thus obtained are less variable *more* robust), and more resilient. Also, such large networks have a smaller chance of a large magnitude collapse. May proposed that communities with overall weaker interactions support a larger mean number of species, in other words, *weak links are stabilizing.*