4th February 2021
(`NetworkNotes` LaTeX-ed on 4th February 2021)

Network Notes

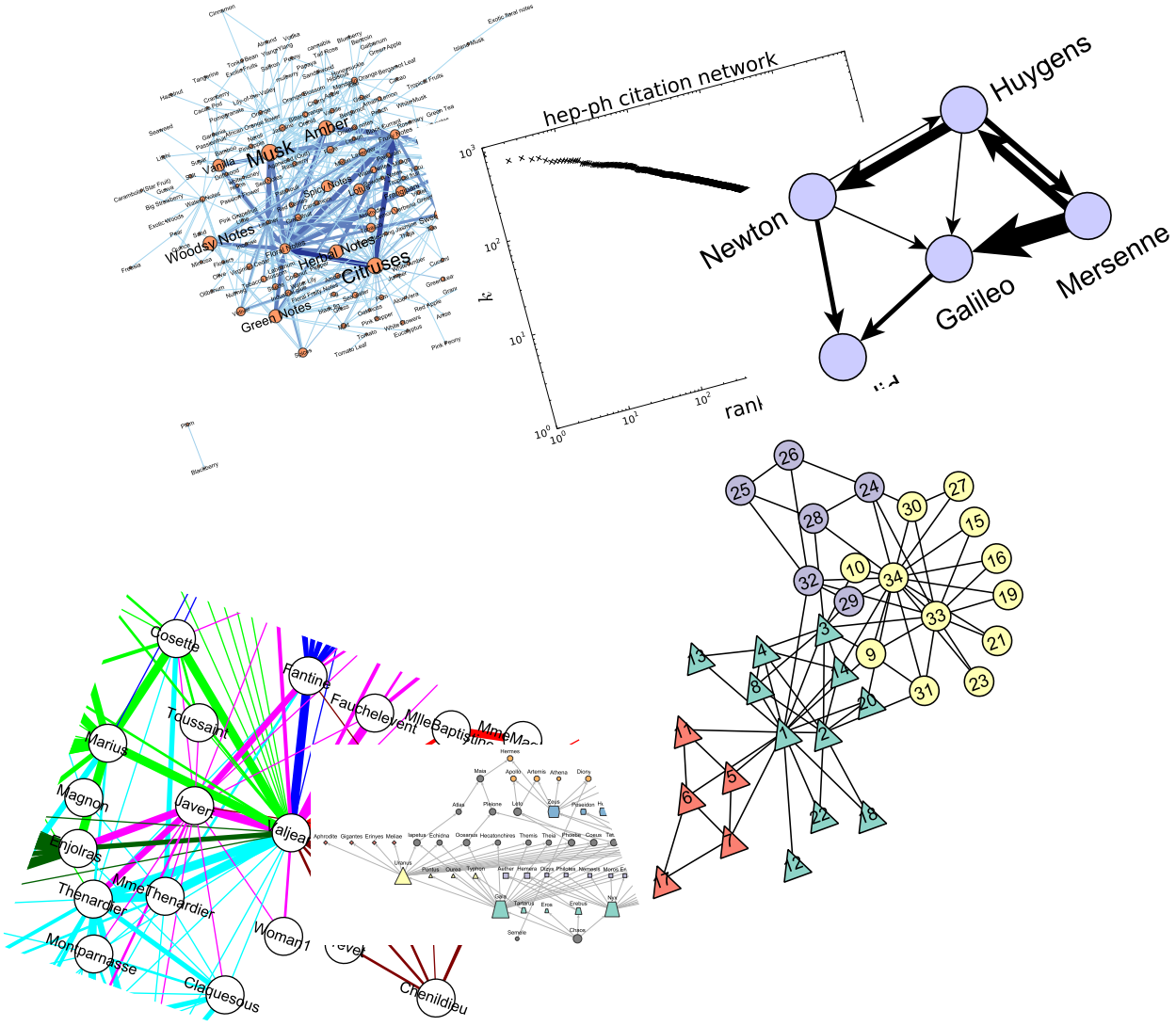
Complexity and Networks Course, Level 3 course
(`PHYS96008`)
Physics Department, Imperial College London.

T.S. Evans

Important Note. These are some notes derived from the slides used in the networks section of the Complexity and Networks course in the Physics department of Imperial College London. These notes are still in development. They are likely to contain typos and mistakes so they are not to be used as a primary source. Please use these notes with care. The slides from the course are more likely to be correct.

Also, in some places the content in these notes has been expanded over the lectures so these notes contain more material than is required for the course. In some cases, but not always, I will mark material not in the Complexity and Networks course with a # symbol. The slides for the course are much closer to the required syllabus. For a precise list of topics to be covered by the course, see the syllabus/aims and objectives for the course which can be obtained elsewhere.

Any feedback on these notes is very welcome.



Contents

Chapter 1

Basics

Aims and Objectives: To give define a network and its key components. To explain why it can be useful. To define various types of networks. To give examples of these. To give standard representations of networks. To discuss the role of visualisation.

1.1 What is a Network?

A loose definition of a NETWORK is that is is a set of objects, called NODES or VERTICES, which are connected in pairs by EDGES, also called LINKS. A simple example is shown in Figure 1.1. Since this is such a general idea used in so many different areas, it should not be too much of a surprise that we find that many different names are used for the same concept. Later, we will give a table of some of the typical variations seen.

For a start, in mathematics, a NETWORK is also known as a GRAPH, denoted $\mathcal{G}$, and there the subject is known as GRAPH THEORY. The formal mathematical definition of a graph or network $\mathcal{G}$ is[1]:-

- A set of VERTICES $\mathcal{V}$.
- A set of EDGES $\mathcal{E}$ which are *pairs* of vertices taken from $\mathcal{V}$, the set of possible vertices of the graph. We will denote an edge between vertices i and j as (i, j).

A network or graph $\mathcal{G}$ is often written as the pair of vertex and edge sets $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

C&N Notation In these notes we will use $N = |\mathcal{V}|$ for the number of vertices, and $E = |\mathcal{E}|$ for the number of edges. We will denote an edge between vertices i and j as (i, j) so more formally, if $(i, j) \in \mathcal{E}$ then $i, j \in \mathcal{V}$.

The definition of a graph is very general and there are several distinct types of edge and graph as we will mention below in section 1.2. An relatively simple example is shown in 1.1 Fig. 1.1. As you can see from Fig. 1.1 there is very natural way to visualise a network, we write the nodes as points and represent the network edges as lines between the points in the plot. However, there are many ways to layout the nodes and edges as points and lines. In general the nodes do not have any information about their location in some space. The only information in a graph is the relations, the edges, between pairs of nodes. Of course a good visualisation of a graph can lead to useful insights but equally, they can be very misleading. For any graph with more than around fifty nodes, they tend to be two complicated to show on the two-dimensional page. They often look likes some sort of 'spaghetti monster' as shown in Fig. 1.2 or see Fig. 1.8 below. Graph visualisation is a big subject in its own right, see section 1.3.1 below.

It is also useful to define a SUBGRAPH. A subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a *graph* made from a subsets of the vertex and edge sets of original graph $\mathcal{G}$. However note that these subsets $\mathcal{V}'$ and $\mathcal{E}'$ *must* form valid graph themselves where the critical issue is that the edges in the subgraph must be made from the vertices in the subgraph. That is if $(i, j) \in \mathcal{E}'$ then $i, j \in \mathcal{V}'$.

[1] See section C for a summary of set notation

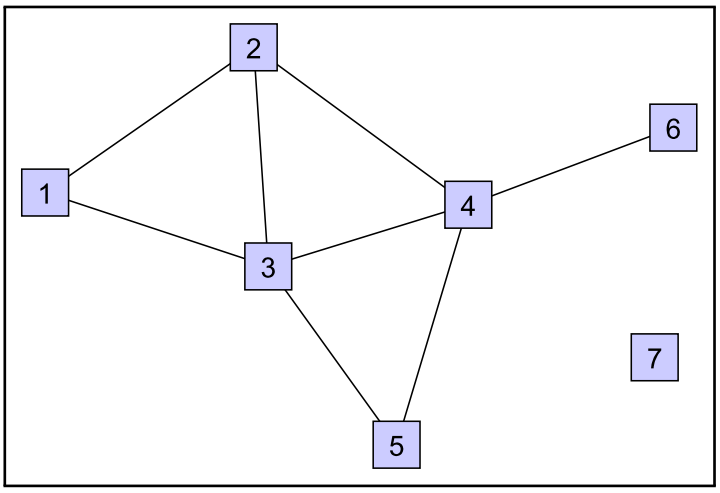


Figure 1.1: An example of a graph of 7 vertices and 8 edges. We may denote this as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where vertex set is $\mathcal{V} = \{1, 2, 3, 4, 5, 6, 7\}$ and the edge set is $\mathcal{E} = \{(1, 2), (1, 3), (2, 3), (2, 4), (3, 4), (3, 5), (4, 5), (4, 6)\}$. Note that not all vertices need have an edge, and the whole graph can have several disconnected parts called components. In this case there are two components: vertices 1 to 6 and their edges form one component, while vertex 7 is a component by itself. Components are special cases of subgraphs.

`graph with communities, spaghetti monster`

Figure 1.2: An example of a visualisation of a large graph. These can sometimes provide some useful information, perhaps revealing some large closely knit regions of the graph, known as communities or clusters. However often they convey very little information.

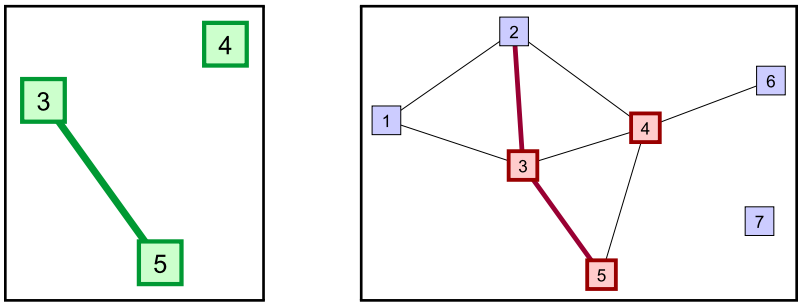


Figure 1.3: On the left one subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of the network in Fig. 1.1is shown. The subgraph has vertex set $\mathcal{V}' = \{3, 4, 5\}$ and edge set $\mathcal{E}' = \{(3, 5)\}$. A subgraph need not be connected nor need they include all the edges present in the original graph given the vertices in the subgraph. For instance the edge between vertices 3 and 4 is in $\mathcal{E}$ but has not been included in $\mathcal{E}'$. However a subgraph must be a valid graph in its own right. Suppose we add the edge between vertices 2 and 3 to the subgraph edge set $\mathcal{E}'$ used in the left hand subgraph example. This is illustrated in the righthand picture. This example is now no longer a subgraph as vertex 2 is in not in the vertex set $\mathcal{V}' = \{3, 4, 5\}$ of the righthand picture.

ASIDE Further Types of Subgraph

There are a few more terms associated with certain types of subgraph.
A CLIQUE is a simple graph (or subgraph) where all the nodes are connected. It is densest graph possible for the given set of vertices. One exercise is to try and find the CLIQUE COVER of a graph, that is to find a set of clique subgraphs, usually overlapping, such that every vertex and edge is at least one clique. Finding the smallest set of such cliques is a well studied problem in computer science.
An INDUCED SUBGRAPH is the subgraph defined by a set of vertices along with *all* the edges between these vertices present in the graph. That is it is the largest possible given subgraph given the set of vertices. If a subgraph is not an induced subgraph then it is a PARTIAL SUBGRAPH.
A MOTIF [Shen-Orr et al., 2002] is just another name for one type of subgraph of interest in a larger graph. A motif is invariably small, and a typical example is that we are counting how many subgraphs are the same as (isomorphic to) the motif of interest. We would typically then compare the number found against the same calculation performed on a null model. The original definition [Shen-Orr et al., 2002] stated a motif was always over represented though this does not always seem to be imposed when the term is used. Likewise, the original definition of motif [Shen-Orr et al., 2002] did not make it clear if the motifs were maximal subgraphs, i.e. induced subgraphs. So it maybe that the term is used with slightly different meanings. Note that when we find an example of the motif we are looking for in a larger graph, the vertices may have extra edges not included in the motif, that is it may be a partial subgraph. Thus we would find a motif of three vertices connected in a line in a triangle where three vertices are all connected directly to one another.
A GRAPHLET [Pržulj, 2007] is then the induced subgraph version of a motif. So if we are looking for a graphlet of three vertices connected in a line, a subgraph of a triangle of vertices would not count as being an example of this graphlet.

1.1.1 Terminology

There is no fixed terminology for networks and different terms are often used in different fields, see table 1.1.

The term VERTEX is typically used in mathematics for the "points" in our visualisations such as Fig. 1.1 but we will also use NODES (typical in engineering and computer science). Other terms encountered are SITE (physics) or ACTOR (in social science).

The "lines" in Fig. 1.1 are typically called EDGES in mathematics are also known as LINKS (engineering and computer science), BONDS (physics) or TIES (in social science).

"point"	"line"	"combination"	typical field
vertex	edge	graph	Mathematics
node	link	network	Engineering, Computer Science
site	bond	network	Physics
actor	tie	network	Social Science

Table 1.1: Table of terms frequently encountered

It is also useful sometimes to refer to the 'ends' of an edge and these are sometimes called STUBS.

1.1.2 Why use a network?

A network captures **bilateral relationships** between objects. A Vertex can be any object you like but will typically be one person, one web page etc. The edges then record some type of relationship between pairs of the objects; it could be a genetic relationship between people or a link from one web page to another.

network	vertex	edge
social	person	friendship
genetic	person	at least 50% shared DNA
organisation	person	line of command
world wide web	web pages	hyperlink
internet	network router	communication link
citation	paper	entry in bibliography
protein network	proteins	participate in same reaction
food web	species	predator-prey
travel	location	trips made between locations
streets	intersections	part of a street

Example 1.1.1. Networks have been used in social science for a long time, starting in the 30's but really active from the 70's onwards [Freeman, 2004]. Many ideas were first developed in this context though this is not always realised and acknowledged in the literature of other fields. Not surprisingly the graphs are called SOCIAL NETWORKS in this context and the study of such networks is often called SOCIAL NETWORK ANALYSIS [Borgatti et al., 2009; de Nooy et al., 2005; Freeman, 2004; Wasserman and Faust, 1994].

- You know where people live but you might use a network to represent who they actually know. A network where the people are vertices and links indicate friendship.
- In an organisation people usually have a position in an official hierarchy at work, so you may link people to their bosses. At Imperial we have
  [Rector] - [Head of Faculty] - [Head of Department] - [Head of Group] - [Staff member] - [Student]
  where here '[A]-[B]' indicates a edge between two vertices A and B.
- At work a different network might record actual working relationships which may cross the official hierarchical structure e.g. I work with staff from different Physics groups and from different departments.
- More complicated relationships might be important. Social science often stresses the role of triads e.g. an enemy of a friend ought to be my enemy (STRUCTURAL BALANCE THEORY). HYPERGRAPHS record general relationships between two or more nodes.

In general there is no one network representation of a data set, and often a network does not capture all the information we have about the objects. Sometimes we can add some of this information to nodes or edges (or both) as ATTRIBUTES but we will generally not do that here to keep everything as simple as possible. In any case, a network representation and approach may not always be the best way of tackling a problem. Sometime there is an 'obvious' network representation e.g. web pages and their links, some useful ones are less obvious, e.g. web pages linked if they have similar key words. The aim is to find network representation that is simple enough to work with yet which captures the essential information about the problem you are studying.

Example 1.1.2. Web Pages Take web pages as the vertices, then the edges could be

- The obvious network:
  edges = hyperlinks between web pages
- Semantic connection:
  edges = pages whose most common words are similar
- Co-citation:
  Link two pages if they point to the same third page.

Example 1.1.3. A Network of Streets

- The obvious network is
  vertex = intersections
  edge = portions of streets between intersections
- Another common representation is
  vertex = whole straight sections
  edges = intersection between straight sections

1.2 Classification of Networks by Edge Type

There are many types of network. Here we classify them by the type of edge, the type of information recorded for their bilateral relationships. Later we will introduce complementary ways of classifying networks. In terms of edge types we have:-

Simple graph: A graph where there is at most one edge between any two pairs of vertices *and* there are no self-loops — an edge where both ends are connected to the same vertex. See Fig. 1.1 for an example.

Directed network: Each edge has a direction, see Fig. 1.4a. Sometimes known as a DIGRAPH

Typically each DIRECTED EDGE is shown as an arrow on the edge pointing away from the SOURCE vertex and towards TARGET vertex.

For directed edges we can specify an edge as the ordered pair (s, t) but you must make clear if first entry is the source or target (conventions can differ)[2].

Example: Network of links between web sites.

Weighted network: Each edge has a numerical value, the EDGE WEIGHT, associated with it, see Fig. 1.4b. This is a simple example of an edge attribute.

The value associated with an edge is the WEIGHT of the edge and these are almost invariably non-negative numbers (for non-negative see signed networks in Section 1.2.1), sometimes integer,sometimes not.

Example: Network of people where edges are the calls between people, weight could be number of calls or could be total time spent on calls between people over certain interval.

Bipartite network: There are two types of vertex and edges **always** run between vertices of opposite type[3], see Fig. 1.4d. These are also known as TWO-MODE NETWORKSTWO-MODE NETWORKS—SEE BIPARTITE NETWORK in social network analysis.

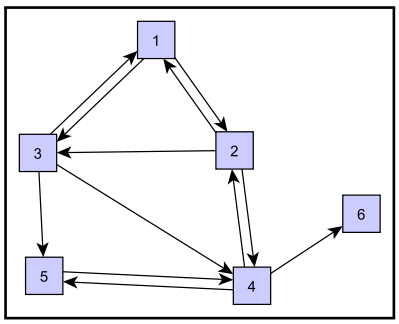
These are known as TWO-MODE networks in the social network literature. Note that these are often the projected down onto a network of just one of type of vertex, joined if they shared a common neighbour

Example: One type of vertex represents researchers and the other represents the papers they publish. Link each researcher to each paper they publish. An obvious projection here is onto a *coauthorship network* where vertices are authors of papers (not the papers themselves) and they are linked if they have published a paper together.

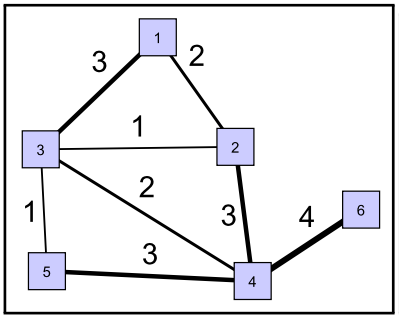
SELF-LOOPS are edges whose ends are connected to the same vertex. Sometimes they have no meaning e.g. you cannot make a Facebook account a friend with itself. Sometimes there is information in the data which is best represented by a self-loop, but you choose to explicitly exclude these examples because they do not carry the type of information we are examining. For instance, if you cc'd yourself

[2]Unless otherwise explicitly stated in a text, you can usually assume that there is no more than one edge going from vertex s to vertex t. If more than one edge is allowed between each pair of vertices then such an edge is known as a MULTIEDGE.

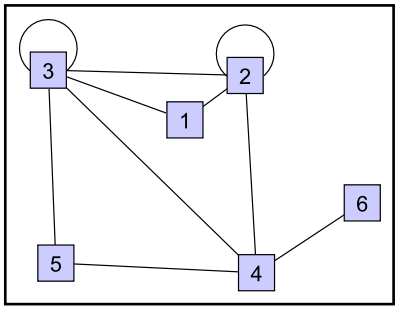
[3]So while the label giving the type of vertex is an example of a vertex attribute, the restriction on the edges allowed means these networks are not just about a binary attribute for the vertices.



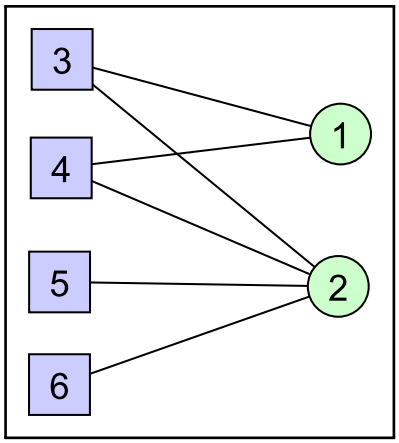
(a) An example of a directed network (or digraph) where the arrow on the edge represents a direction. Note some pairs of vertices are connected in both directions (the relationship is reciprocated), some are just one-way (unreciprocated). The hyperlinks between websites (the vertices) show this type of structure.



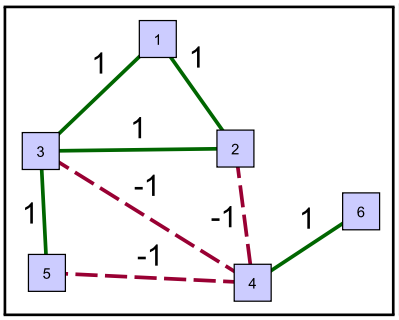
(b) An example of weighted network. The values associated with the edges, the WEIGHTS are shown beside each edge. For example it could represent the number of calls made in a week between different telephone numbers.



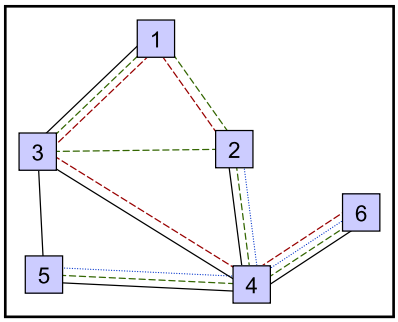
(c) Example of Network with self-loops. Vertices numbered 2 and 3 have a SELF-LOOP, an edge starting and ending at the same vertex.



(d) A bipartite network (a two-mode network). This two types of vertex are indicated by different shape and colour and all edges run between vertices of different types have example has four different types of edge, indicated by the different styles and colours of the edges. This could be used to represent four different transport options between six locations.



(e) An example of a signed network, one where the edge weights can be positive or negative. Can be used to represent friendship and enmity in a social context for example.



(f) A multiplex network (a multigraph). This example has four different types of edge, indicated by the different styles and colours of the edges. This could be used to represent four different transport options between six locations.

Figure 1.4: Examples of different types of network.

when sending an email, we might not count this email from one address to itself when examining email relationships. Sometimes there is some meaning to a self-edge and for operational reasons it hasn't been included in a network representation. For instance, when looking at commuting patterns, many analyses ignore the possibility that you commute from your home region to work in the same region e.g. you work from home, in part because many models do not capture this behaviour.

Sometimes self-loops are very useful, even to the extent you might add them to your network representation even if not naturally present in the data. Later we will look at simple diffusion processes (see Chapter 7.82) where at each tick of a clock, a random walker moves from one vertex to another falling the edges leaving that vertex. If we want to add the option that the walker is delayed, we could capture that behaviour with a self edge. This trick is sometime used in community detection

algorithms, see Example 1.3.3.

These are the types of network most often encountered but it is not exhaustive. The conditions mentioned here can be applied in different combinations. So we could imagine weighted directed networks, networks with graphs with negative weights of many values, any of the above types where self-loops are allowed. Sometimes it may be useful to allow more than one edge between the same pair of vertices (e.g. representing different phone calls between people occurring at different times).

Some further types of graph or descriptions of graphs often encountered are as follows:

- a COMPLETE GRAPH is one where all possible edges are present.
- a REGULAR GRAPH is one where all vertices have the same number of edges attached.
- a SPARSE NETWORK is one where the number of edges and the number of vertices are of the same order.
- a DENSE NETWORK is one where the number of edges is close to the maximum, and the graph is close to being a complete graph.

The definition of sparse and dense is rather vague here but this reflects the rather vague way they are used for any real data. The terms come from a more formal context when dealing with mathematical models where they can be given a precise definition. In this case, we consider taking the number of vertices N to infinity and we say that a network is sparse if the number of edges E scales as $O(N)$ or slower, while it is dense if the number of edges scales as $O(N^2)$. A common quantify how 'dense' a network is the DENSITY of a network, which is simply the fraction of possible vertex pairs which are linked by an edge,

$$\rho = \frac{2E}{N(N-1)}\,. \tag{1.1}$$

1.2.1 (#) Further Network Types

Given that a graph captures relationships between pairs of objects, and given that there can be many different types of relationship, it should not be surprising that there are many more types of network in use. In this course we will not consider such cases.

A SIGNED GRAPH is one in which each edge has a sign associated with it, usually an edge with weight $+1$ and -1. A positive (negative) sign could represent a positive or negative relationships, see Fig. 1.4e. For example this is a useful network representation of friendships and enmities between players in a massive online game. This happens where there is an advantage in marking some players as enemies rather than those with which they have no specific relationship, for example see [Szell et al., 2010; Szell and Thurner, 2010].

Another example of a signed graph comes from descriptions of complex systems such as gene regulatory networks. These are networks describing complex systems controlling some sort of dynamical process. They involve feedback and feedforward loops which require directed flows and so a directed network. However, some of the feedback is positive, allowing or amplifying the flow, and some is negative used to shut down flows. This is where we would need signs on the edges to indicate the nature of the process, for example see Roli et al. [2017].

In an MULTIGRAPH each pair of vertices can have several edges connecting them, see Fig. 1.4f. For example we have data on individual phone calls made between people at different times, we might use a separate edge for each call, characterised by the time of the call for instance. In that way the same two people, the same two vertices, can have more than one edge.

However if edges come in a limited number of different types, each type forming a LAYER, then these networks are often called a MULTILAYER NETWORK or a MULTIPLEX NETWORK. For example, in a network of interactions between people, each type of interaction is represented by a different type of edge, e.g. different edges for email communication, face-to-face meetings, and phone calls. However, the naming conventions for any of these 'multi-something' networks is particularly confusing and inconsistent, for example see [Kivela et al., 2014] for a survey.

In a SPATIAL NETWORK each vertex is linked to a location in some space, for example the geographical location of the vertex. If the space has low dimensions, such as two-dimensional geography, the constraint of the space alters the way we should look at the network and we probably need to adapt or use alternative measure to probe the nature of such networks. These will be considered further in Section **??**.

A PLANAR GRAPH is one where one can embed the vertices in a two-dimensional Euclidean plane such that any edge, which are drawn as a straight line between the vertices it connects, does not intersect with any other edge[4]. There are several exact results for planar graphs which can be exploited e.g. [Tumminello et al., 2005]. All the examples in Figure 1.4 are planar. For example you can see this for the the bipartite graph Figure 1.4d if you move vertex two to lie on the opposite side of the line of vertices 3, 4, 5 and 6.

A TREE is a connected network with no loops so it looks like the roots and branches of a tree, for example see Figure 1.5. The lack of any loops means you can always lay them out such that no edge need cross another edge, i.e. trees are always planar graphs. Of particular interest are SPANNING TREES of a network, which are subgraphs containing every node of a connected network. This means that spanning trees have one less edge than the associated connected network, the minimum number of edges needed to remain a connected subgraph. Of particular interest are MINIMAL SPANNING TREES (**MST**) which in weighted graph are spanning trees which have the lowest total weight edge weight in the tree.

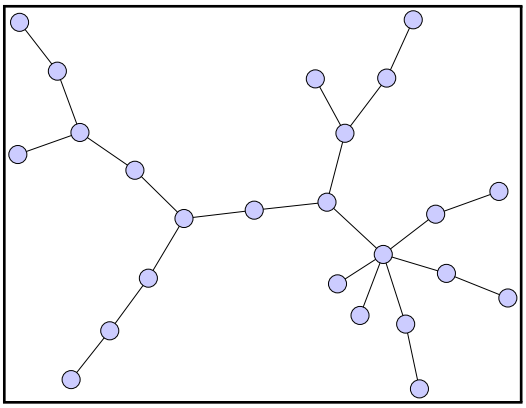


Figure 1.5: An example of a tree, a single component network with no loops. Trees are always examples of planar graphs.

1.3 Representation of Networks

In this course we will assume that there is at most one edge for every pair of edges if the network is undirected. If edges have a direction, we will allow at most one edge in each direction for every pair of vertices. This is true in most work and is sufficient for many actual problems.

The formal notation for a network typically used in graph theory is as follows. This means that we can represent each edge by the pair of vertices at its two ends, written formally as $(i, j) \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ where $i, j \in \mathcal{V}$. For undirected networks $(i, j) = (j, i)$, that is the order of the pair in the notation is irrelevant and refers to the same edge. For directed networks (i, j) and (j, i) represent two separate edges running in opposite directions between the same two vertices.

Most calculations, algebraic or numerical are carried out using one of three different representations of a network: an edge list, an adjacency List, or an Adjacency Matrix.

[4] As an aside, it is interesting to note that there are some powerful theorems in graph theory, Kuratowski's theorem and Wagner's theorem, about planar graphs. These state that a network is planar if and only if it does not contain subgraphs which are "related" to either the complete graph of five elements (denoted K_5) or the bipartite graph where three nodes of one type are connected to all three nodes of the second type (denoted $K_{3,3}$) two sets of three nodes. The complication is that the phrase "related" includes the case where subgraphs are equal to K_5 or $K_{3,3}$ but also includes more complicated relationships so for that reason we will not consider this in more detail here.

Adjacency Matrix

The ADJACENCY MATRIX A captures all the information in a network (if no multiple edges) and is defined such that A_{ij} is the weight of the edge running from vertex j to vertex i.

- DIRECTED EDGES: Adjacency matrix is not symmetric $\mathsf{A} \neq \mathsf{A}^{\mathrm{T}}$.

  C&N Notation A_{ji} represents the weight of an edge running from source vertex i to target vertex j. see Fig. 1.6.

- WEIGHTED EDGES: Entries of adjacency matrix entries not restricted to 0 or 1

- SELF-LOOPS: These are the diagonal entries of the adjacency matrix $A_{ii} \neq 0$ (no sum over repeated index i)[5].

- BIPARTITE NETWORK: Adjacency matrix can be arranged to be in an off-diagonal block form.

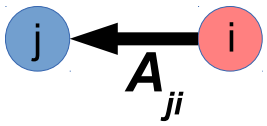


Figure 1.6: The convention used for adjacency matrices in this course.

Thus a SIMPLE GRAPH has an adjacency matrix which is symmetric (no directions), takes values 0 or 1 (no weights), and has no diagonal elements (no self-loops)

$$\text{Simple graph:} \qquad \mathsf{A} = \mathsf{A}^{\mathrm{T}}, \qquad A_{ij} \in \{0,1\}, \qquad A_{ii} = 0 \text{ (no sum over i).} \tag{1.2}$$

Example 1.3.1. The adjacency matrices for the networks in Figure 1.4 and Figure 1.7 are as follows:-

Simple network of Fig. 1.7

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

Directed Network of Fig. 1.4a

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

Weighted Network of Fig. 1.4b

$$\begin{pmatrix} 0 & 2 & 3 & 0 & 0 & 0 \\ 2 & 0 & 1 & 3 & 0 & 0 \\ 3 & 1 & 0 & 2 & 1 & 0 \\ 0 & 3 & 2 & 0 & 3 & 4 \\ 0 & 0 & 1 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \end{pmatrix}$$

Network with self-loops of Fig. 1.4c

$$\begin{pmatrix} \mathbf{0} & 1 & 1 & 0 & 0 & 0 \\ 1 & \mathbf{2} & 1 & 1 & 0 & 0 \\ 1 & 1 & \mathbf{2} & 1 & 1 & 0 \\ 0 & 1 & 1 & \mathbf{0} & 1 & 1 \\ 0 & 0 & 1 & 1 & \mathbf{0} & 0 \\ 0 & 0 & 0 & 1 & 0 & \mathbf{0} \end{pmatrix}$$

Bipartite network of Fig. 1.4d

$$\begin{pmatrix} \mathbf{0} & \mathbf{0} & 1 & 1 & 0 & 0 \\ \mathbf{0} & \mathbf{0} & 1 & 1 & 1 & 1 \\ 1 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 1 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

Signed network of Fig. 1.4e

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & -1 & 1 & 0 \\ 0 & -1 & -1 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

(1.3)

[5] In this course we will not use summation convention That is repeated indices do not indicate a sum, and sums over indices will be given explicitly.

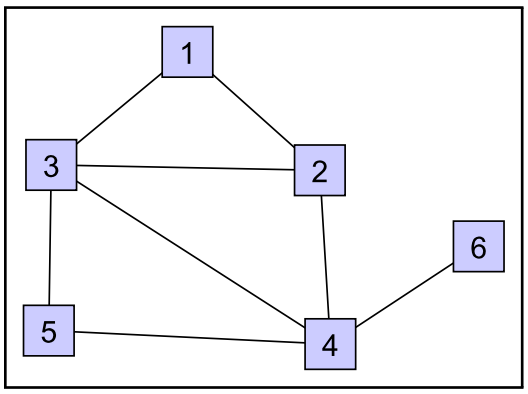


Figure 1.7: An example of a simple network. At most one edge between each pair of distinct vertices with no value or direction assigned to the edges. Most networks may be simplified, into a simple graph e.g. by ignoring edge directions and weights, removing self-loops. In some cases this is a useful simplification, and sometimes valuable information is lost.

Adjacency List

For all but the densest graphs, the most efficient way to store a network in a computer or in a file is an ADJACENCY LIST. That is for each vertex i you give a list of vertices which are adjacent to i, the set often referred to as the neighbours of vertex i.

ASIDE `simplegraph`

An example of the use of an ADJACENCY LIST can be found in `simplegraph`, a simple implementation of a graph structure in `C++` available on the Blackboard site for this course. In `simplegraph` the vertices are numbered consecutively from 0 to $(N-1)$ where N is the current number of vertices. The adjacency list is a vector of vectors, `v2v`, such that `v2v[i]` is a list of the neighbours of vertex i. One could do the same in python using a list of lists and the notation would be very similar. That means `v2v[i][n]` is the index of the n-th neighbour of vertex i. Each time you add a vertex to the graph, `simplegraph` has to add a new list of neighbours for that new vertex but that list is initially empty as the new node has no edges as yet. That is we need to use `v2v.push_back(vector<int>() );` You have to 'remember' that the vertex we have just added has index equal to `(N-1) = (v2v.size()-1)` *after* this `push_back` has happened.
Every time you add an edge, the vertex must already exist (so it has a list of neighbours ready to be filled) and then if you are adding an edge between vertices s and t you need to add s to be a neighbour of t, `v2v[t].push_back(s)`, and add t to be a neighbour of s using `v2v[s].push_back(t)`.
So for the standard five mathematician graph, we will end up with a vector of vectors with entries

```
v2v[0]= [1,2,3]
v2v[1]= [0,2]
v2v[2]= [0,1,3,4]
v2v[3]= [0,2,4]
v2v[4]= [2,3]
```

where $[1,2,3]$ is a vector. Thus `v2v[3][1]=2` represents the fact that the second neighbour (counting from 0 so we need `v2v[.][1]`) of vertex index 3 is the vertex with index 2. It is irrelevant that this is the second neighbour in terms of graphs so the order in each `v2v[i]` is irrelevant. If we added the edges in a different order we would get a different order to the numbers. What is important is that each it records the connection from vertex 2 to vertex 3. Note that this edge $(2,3)$ will also need to be recorded as 3 being a neighbour of 2, and indeed we see it appears here as the third neighbour of 3, the entry `v2v[2][2]=3`.
Also note the size of the $[\ldots]$ vectors is what is called the degree of that vertex `k_i = v2v[i].size()`, it is the number of neighbours of that vertex (see Section 2.1).
Interestingly for this small example we use 14 entries (plus some overhead for 6 vectors) while an array would use 25 locations so memory usage is close but even here the adjacency arrangement wins.

For example the example in Fig. 1.7 has an edge list of

source	neighbours
1	2,3
2	1,3,4
3	1,2,4,5
4	2,3,5,6
5	3,4
6	4

(1.4)

This is a good data structure for a network in a computer programme as you can write it as a list of lists and then it is

- Sparse, only records edges which are present.
- Easy to find properties of each vertex.

The adjacency list is not ideal for algebraic manipulations

Edge List Representation

An Edge List is a simple list of the source and target vertices of each edge, with weight of edge if needed. For the example in Figure 1.7 we would have

source	target
1	2
1	3
2	3
2	4
3	4
3	5
4	5
4	6

(1.5)

An Edge Representation has the following features.

- Good for storing as a data file.
  - Simple
  - Sparse, only records the edges which are present.
- Poor for programmes. It would be slow to find the edges of one vertex.
- Misses vertices with no edges (though this is fixable by strong this in some extra data structure)
- Not good for algebraic manipulations

(#) Self-Loops

Self-loops are not often encountered but when they are present it is worth noting how they should be represented. Consider an undirected unweighted network. A self-loop is an edge with both ends connected to the same vertex. This means that the entry in the diagonal entry A_{ii} for one self-edge at vertex i is 2 as that is the number of stubs connecting vertex i to vertex i. Essentially we are saying each entry A_{ij} counts the number of edges leaving vertex j and connecting to vertex i but if i and j are distinct then one edge gives two contributions — one in A_{ij} and one in A_{ji}. We need to mimic these two contributions for the self-edge case if some of the tricks we use when counting edges from the adjacency matrix are to work. Likewise, the adjacency list needs to put two entries for vertex i in the list of neighbours for vertex i. Essentially, a self-edge in this view point is adding two to the degree of a vertex because a self-edge adds two more stubs to the vertex it is attached to.

By following these rules, we will find we can count the number of edges in the usual way, e.g. the sum of entries in an adjacency matrix for a undirected unweighted network is twice the number of edges $\sum_{i,j} A_{ij} = 2E$, with or without self-edges, provided we follow the rules outlined here.

By following the same logic however, a single unweighted but directed self-loop gives a diagonal entry of just one. Every directed edge gives just a single contribution of one to the adjacency matrix entries. This ensures that when counting the outgoing edges from vertex j, $\sum_i A_{ij}$, the diagonal entry is included for any self-loop leaving j, while the same diagonal entry is the only term which also contributes when counting the in-coming edges to the same vertex j, $\sum_i A_{ji}$. This is as it should be, the self-loop is the only edge which is both incoming and outgoing from the same vertex.

1.3.1 Visualisation

Vertices do not in general have any coordinates in space. When we make a two-dimensional representation of a network in a figure, we may place the vertices anywhere provided we always connect vertices as specified by the edges of the network. Visualisation of networks is both an art and a science. The presentation of any data can make or break a piece of work. While pictures can help or simply be aesthetically pleasing, there is a danger, especially with large networks, to produce what are sometimes called "Spaghetti Monsters", figures that so not convey any useful information.

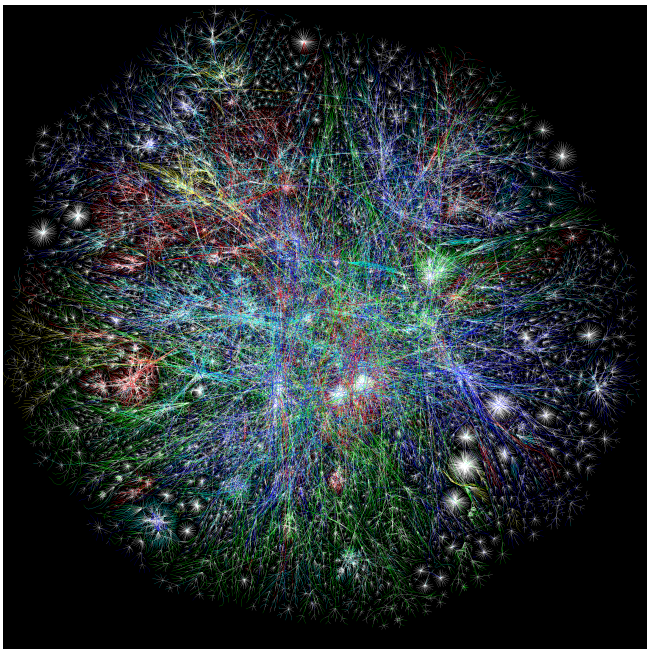


Figure 1.8: Partial map of the Internet based on data from November 24rd 2003, taken from `opte.org/maps`. Each line is drawn between two nodes, representing two IP addresses. The length of the line is indicative of the delay between the two connected nodes. Lines are colour-coded based on Class A allocation of IP space (Asia Pacific - Red, Europe/Middle East/Central Asia/Africa - Green, North America - Blue, Latin American and Caribbean - Yellow, RFC1918 IP Addresses - Cyan, Unknown - White).

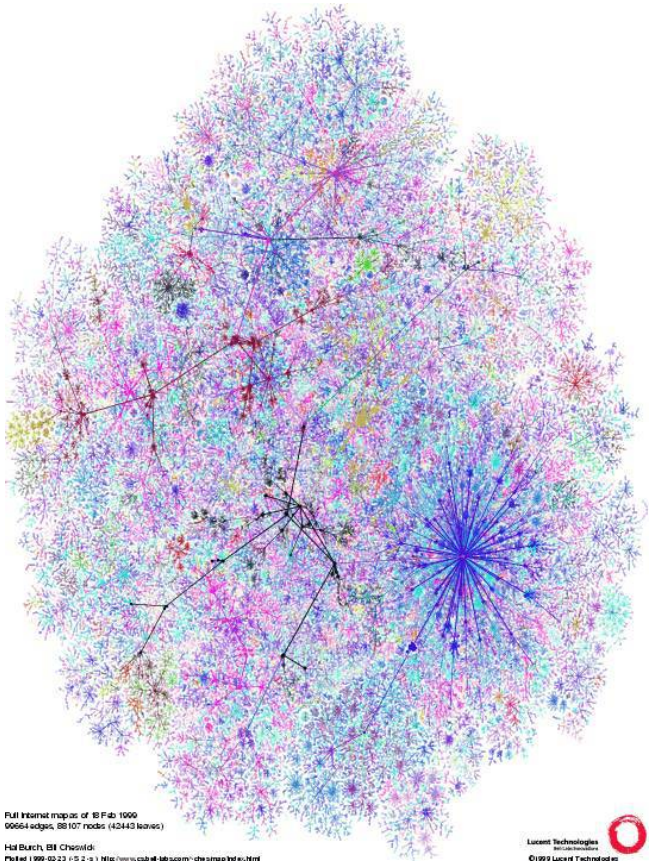


Figure 1.9: The router level connectivity of the Internet as measured by Hal Burch and Bill Cheswick's Internet Mapping Project. The work is being commercially developed by Lumeta `http://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/lumeta_large.jpg`. This appears to be a typical example of a visually attractive but scientifically uninformative visualisation of a network — a "Spaghetti Monster". However the colour coding and placement, if they are well defined, may well make this picture useful to experts.

Example 1.3.2. Visualisations of the same network Different visualisation of lattice and equivalent random regular graph. Very hard to spot which are equal, the hard problem of Graph Isometry (= are two graphs equal).

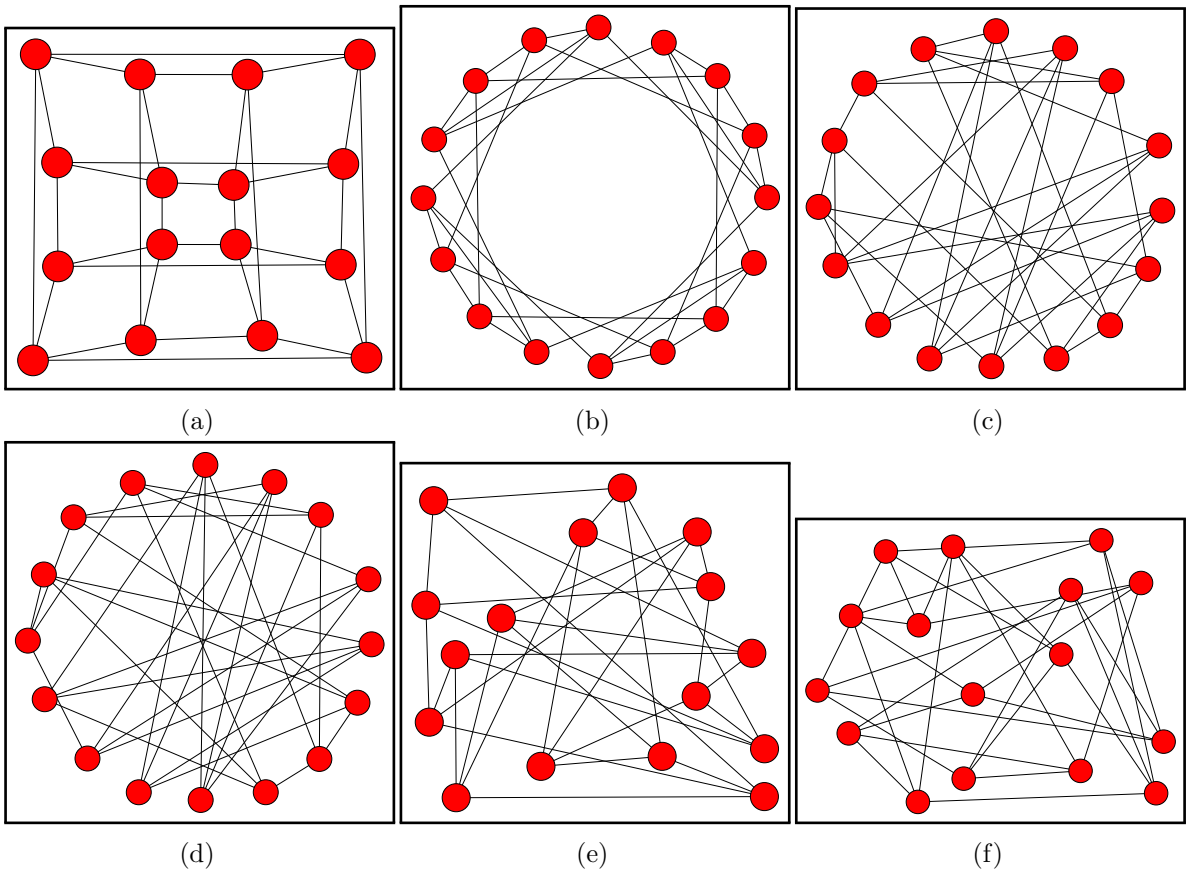


Figure 1.10: Two regular graphs are shown here with different layouts. The first is defined as a regular square lattice, with each vertex connected to nearest its four neighbours. The boundaries are periodic (i.e. working on a torus) so that each vertex has degree 4. The example in Fig. 1.10a is arranged on the defining lattice points then lightly distorted to show all the edges clearly. The second graph is a random graph with all vertices of degree 4 (so another a regular graph). It is not isomorphic to the regular lattice meaning that there is no way to rearrange the positions of the vertices to reproduce Fig. 1.10a without changing some of the edges. The challenge to the reader is to find which of these graphs is isomorphic to each other.

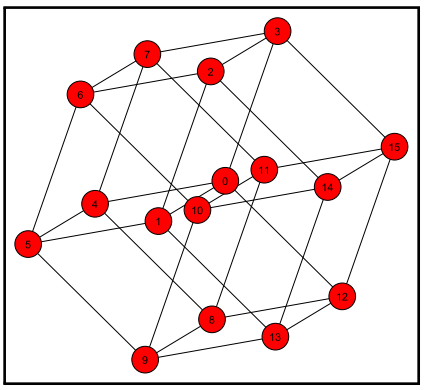


Figure 1.11: The regular square lattice of Fig. 1.10a displayed using metric MDS methods of the Visone package [Brandes and Wagner, 2004]. While not laid out as a square it does show the underlying pattern very clearly.

Example 1.3.3. Communities COMMUNITIES (or CLUSTERS in computer science) of vertices are ones with strong connections within each community. Scientific methods can be used identify such groups. It usually makes sense to place vertices in the same community close to each other in a visualisation, perhaps identified by different coloured vertices. So many visualisation techniques are implicitly finding these clusters in the network data when they place certain nodes close together. There however many approaches which find these communities directly [Fortunato, 2010] without any reference to visualisation.

Chapter 2

Properties of Networks

2.1 Degree and Degree Distribution

The DEGREE k_i of a vertex i is the number of edges attached to that vertex[1], see Figure 2.1. For simple networks

$$k_i = \sum_j A_{ij} = \sum_j A_{ji} \,. \tag{2.1}$$

The degree counts the number of stubs, the ends of an edge, attached to a vertex[2].

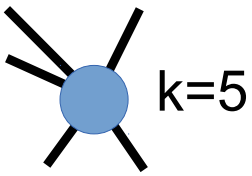


Figure 2.1: Example of degree for an undirected network.

For directed graphs, it is necessary to define the following, see Figure 2.2:
The IN-DEGREE $k_i^{(\text{in})}$ is the number of edges arriving

$$k_i^{(\text{in})} = \sum_j A_{ij} \,. \tag{2.2}$$

The OUT-DEGREE $k_i^{(\text{out})}$ is the number of edges leaving vertex i;

$$k_i^{(\text{in})} = \sum_j A_{ji} \tag{2.3}$$

where you should note that the order of indices is different in (2.2) and (2.3).

The total degree would be $k_i = k_i^{(\text{in})} + k_i^{(\text{out})}$[3].

Example 2.1.1. Consider a network representing web pages where each page is a vertex and there is a directed edge from page i to page j if there is a hyperlink on page i pointing to page j. Then the arrow points in the direction a web user will go when surfing the net[4] The number of links on your web page is your out-degree $k_{\text{me}}^{(\text{out})}$ while the number of pages with a link to your page is your $k_{\text{me}}^{(\text{in})}$. The web is such that the out-degree is trivial to calculate, but you don't know your web page's

[1] See L6 and L7 of Clauset [Clauset, 2013]. L3 Gastner [Gastner, 2011].

[2] If self-loops are present you may want to be careful about how you count them in degree, depending on the context. If you want them to be visualised as an edge with both ends connected to the same vertex, then for consistency each self-loop edge contributes a factor of two to the degree of the vertex to which they are attached.

[3] A directed graph where $k_i^{(\text{in})} = k_i^{(\text{out})}$ is known as a REGULAR GRAPH.

[4] Of course there are many other ways to traverse the web using information not on a web page, e.g. just the back button on a browser makes use of that user's own recent history.

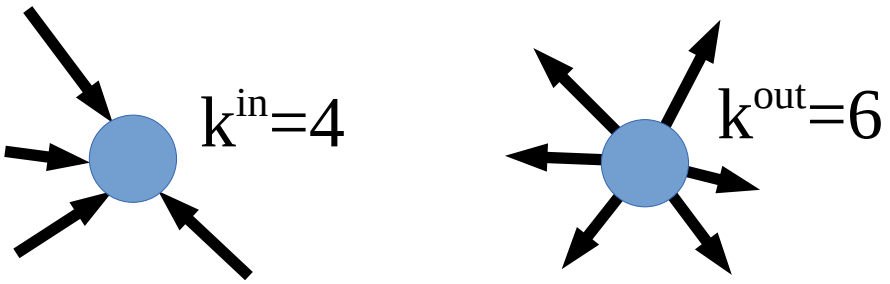


Figure 2.2: Examples of in- and out-degree in a directed network.

in-degree. The in-degree is one simple measure of how useful your page is. In social science measures of the importance of a vertex in a network are known as CENTRALITY measures. For example see Figure 2.3.

Note that there can be hyperlinks from a page to another part of that same page. In many studies these are ignored. However such internal links are naturally represented by a self-loop.

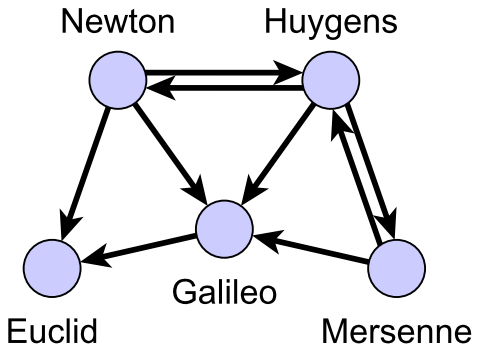


Figure 2.3: Example of the structure of a web site, where web pages are vertices and edges represent hyperlinks between pages. This network is derived from five biographies of mathematicians taken from the MacTutor website [O'Connor and Robertson, 2017]. Galileo has the highest in-degree of $k^{(\text{in})}_{\text{Galileo}} = 3$ making him the most important by this measure of CENTRALITY in this network. His out-degree is $k^{(\text{out})}_{\text{Galileo}} = 1$.

Example 2.1.2. Consider a twitter network where each account is a vertex and if account i is following account j then we put a directed edge from j to i so that the arrow is in the direction of information flow. The number of followers you have is your out-degree $k^{(\text{out})}_{\text{you}}$ while the number of people you are following is your $k^{(\text{in})}_{\text{you}}$.

One of the most basic measures of a network is the DEGREE DISTRIBUTION, $n(k)$ which is the number of nodes with degree k. This is often phrased as a probability distribution $p(k)$ where

$$p(k) = \frac{1}{N} n(k)\,, \qquad N = \sum_{k=0}^{\infty} n(k) = \sum_{i \in \mathcal{V}} 1 \tag{2.4}$$

and N is the number of nodes in the network. Another way to say this is that the value of $p(k)$ is the probability of finding a node with degree k when we choose a vertex at random, or more precisely when we choose a vertex uniformly at random from the set of vertices[5].

[5]For this interpretation to work we are formally treating all vertices equally when drawing one of them. However it is common in much of the literature that an unqualified use of the word 'random' really means something is being chosen

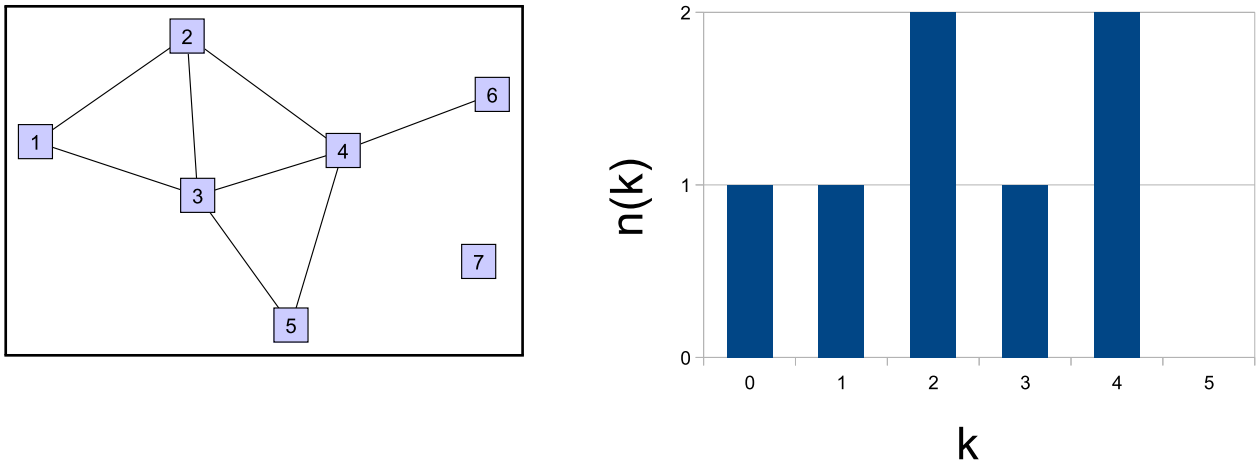


Figure 2.4: The degree distribution of the simple graph.

2.2 Walks, Paths, and Cycles

The degree is a very LOCAL property of a vertex, it only counts the number of nearest neighbours and typically does not probe the majority of the network containing that vertex. The main use of network analysis is to enable us to see how the structure on larger scales, beyond nearest neighbours, is important for the system we are studying. One set of ideas which probe this structure come with the concepts of walks and paths.

A WALK $\mathcal{W}$ in a graph is a sequence of vertices, v_n with $n \in \{0, 1, 2, \ldots, L\}$, where there is an edge from each vertex v_n to the next vertex in the sequence v_{n+1}, so that $(v_n, v_{n+1}) \in \mathcal{E}$. More formally

$$\mathcal{W} = \{v_n | v_n \in \mathcal{V}, n \in \{0, 1, 2, \ldots, L\}, \text{ and } (v_n, v_{n+1}) \in \mathcal{E}, n =\in \{1, 2, \ldots, L\}\}\,. \tag{2.5}$$

Note that if the graph is directed, you must follow the direction of the arrows along your walk. That is in the definition above each pair of neighbouring vertices in the sequence, (v_n, v_{n+1}), must be present in the edge set $\mathcal{E}$. If the edges are directed, the presence of an edge in the other direction, (v_{n+1}, v_n), is not sufficient for the walk $\mathcal{W}$ above to exist. For instance in Figure 1.4a there the sequence of vertices $\{3, 2, 4\}$ is not a walk as there is no edge $(3, 2)$, no edge from vertex 3 to vertex 2 even though there is an edge in the other direction between this pair of vertices.

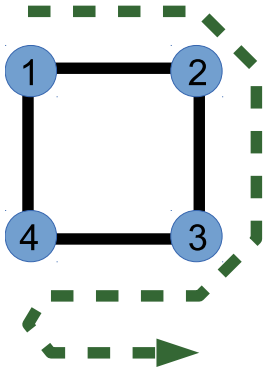


Figure 2.5: Example of a walk $\mathcal{W} = \{1, 2, 3, 4, 3\}$ with length $L = 4$.

The LENGTH of a walk is the number vertices in the sequence *minus one*, here L (note we index of nodes in the sequence starting from zero). So it is the number of edges traversed as you moved from

"uniformly at random" from the relevant set. Usually it is pretty obvious what is meant but we will see later that there are situations where we choose vertices with some bias, in particular choosing high degree nodes more often than low degree nodes.

the first vertex v_0 to the last v_n. See Figure 2.5 for an example. It is simple to show that if the graph is unweighted with only 0 or 1 entries in its adjacency matrix $A_{ij} \in \{0, 1\}$, then number of walks of length L from vertex i to vertex j is simply $[\mathsf{A}^L]_{ji}$.

Note that a vertex may appear more than once in a walk[6]. In particular a CYCLE is a walk which starts and ends at the same vertex $v_1 = v_L$, see Figure 2.6. It is also useful to define a PATH as the special type of walk with NO cycles, so each vertex in a path is unique, so $v_n = v_m$ iff $n = m$. An example is shown in Figure 2.6. An object without cycles is called ACYCLIC and, as a network without cycles is necessarily directed, such networks are called DIRECTED ACYCLIC GRAPHS.

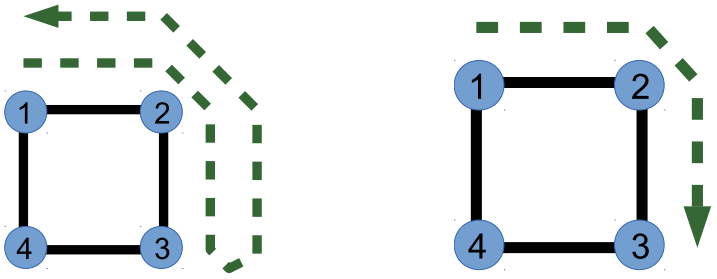


Figure 2.6: On the left a CYCLE is shown in a simple graph, the walk $\{1, 2, 3, 2, 1\}$ passes twice through both vertex 1 and vertex 2. A PATH $\{1, 2, 3\}$ of length 2 is shown on the right.

Terminology warning. It is not uncommon for people to use the term 'path' when in principle they are including walks in their definition. What is usually happening is in that context only the paths are going to be used so really it is just a loose (incorrect) definition has been used but it is usually straightforward to see what was intended.

In many situations we are interested in the SHORTEST PATH between two vertices, see Figure 2.8 for an example. For instance, the quickest way to send information between two computers is likely to be via the shortest path in the network of routers linking the two computers if the transfer of information at routers (the nodes of the network) is the bottleneck in that system[7].

We will indicate the shortest path from vertex i to j as ℓ_{ji}, following the same convention for direction as used here for the adjacency matrix. To find ℓ_{ji} we look at all possible paths (walks with cycles are always longer) and find the length of the shortest — there is often more than one path with the same length.

This immediately gives us a very useful characteristic length scale for a network — the AVERAGE SHORTEST PATH LENGTH ℓ. This is the average over all pairs of connected vertices of the shortest path lengths between those pairs. More formally[8] the shortest path between vertices i and j is

$$\ell(i,j) = \frac{1}{|\mathcal{P}|} \sum_{i,j \in \mathcal{P}} l_{ji} \text{ if } |\mathcal{P}| > 0\,, \text{ while } \ell = 0 \text{ if } |\mathcal{P}| = 0\,, \tag{2.6}$$

where $\mathcal{P}$ is the set of all distinct vertex pairs (i, j) $(i \neq j$, directed if needed) where there is at least one path from vertex i to vertex j. The size of this set, $|\mathcal{P}|$, is used to normalise the sum. It is this measure which is linked to the famous idea about the "six-degrees of separation".

Example 2.2.1. Milgram's Experiment

[6]# Aside. In fact an edge may also appear more than once in a walk. So there is another concept, the TRAIL, which is a walk where no edge appears more than once though a vertex can appear more than once. We will not use this in these notes.

[7]We can be a lot more formal and show that if we define the distance between two vertices to be the length of the shortest path between those vertices then this satisfies all the formal mathematical properties required of a DISTANCE and of a METRIC. For instance the TRIANGLE INEQUALITY is satisfied. We can then show that the shortest path is the GEODESIC, again satisfying the formal definition of a geodesic.

[8]Note that if there is a single strongly connected component then every vertex is connected to every other vertex so $\mathcal{P}$ is simply the set of all distinct vertex pairs. However, this definition has been carefully constructed to give a finite answer in situations where there is more than one strongly connected component. The precise definition used elsewhere in such situations varies.

Milgram (1967) (see Figure 2.7) asked people in Omaha (Nebraska) and Wichita (Kansas) to send packets to people in Cambridge MA specified by name, profession and rough location only. Packets were only swapped between people who knew each other by first name. If the packets arrived at the correct person, they had been through about five intermediaries.

This is much smaller than we might have guessed given the physical distance between the original senders and the final recipients, who also presumably had no direct social or other contacts. So it is intriguing to learn that these small world ideas championed by Milgram (see Figure 2.7) and certainly of great significance may not actually be well supported by all the data he collected, both that reported in the scientific literature and unreported results in the Yale archives of his work. Watts [Watts, 2003] gives an excellent overview of the work of Kleinfeld who reviewed the results Milgram actually obtained in his various small world studies.

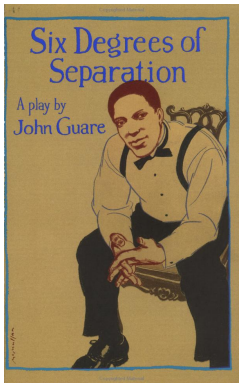


Figure 2.7: On the left, Stanley Milgram (1933–1984), a social psychologist who performed several well known if controversial experiments including the one on six degrees of separation. Milgram's experiment inspired the play "Six Degrees of Separation" by John Guare (1938–).

Aside John Guare

John Guare was born in New York, NY on February 5, 1938. The work of Milgram inspired John Guare's play "Six Degrees of Separation" [Guare, 1990] which includes the following passage.

> I read somewhere that everybody on this planet is separated by only six other people. Six degrees of separation. Between us and everybody else on this planet. The president of the United States. A gondolier in Venice. It's not just the big names. It's anyone. A native in a rain forest. A Tierra del Fuegan. An Eskimo. I am bound to everyone on this planet by a trail of six people. It's a profound thought. ... How every person is a new door, opening to other worlds. Six degrees of separation between me and everyone else on this planet.

The play received the New York Drama Critics Circle Award in 1990 and an Olivier Best Play Award in 1993. It was also made into a film of the same name in 1993, for example see Six Degrees of Separation for a YouTube clip.

Another equally good measure of the "size" of a network is to quote the DIAMETER D which is the longest shortest path[9], $D = \max\{l_{ji} | i, j \in \mathcal{P}\}$. That is you look at all connected pairs of vertices and find their shortest paths, and then you take the largest value from these. See Figure 2.8.

[9] Another related concept is the longest shortest path from one vertex, i, to any other, known as the ECCENTRICITY e_i of vertex i, so that $e_i = \max\{l_{ji} | j \in \mathcal{P}\}$. The diameter is then the largest eccentricity value $D = \max\{e_i | i \in \mathcal{P}\}$.

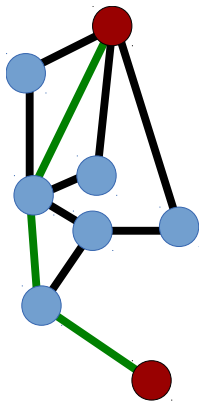


Figure 2.8: The shortest path between the highest and the lowest nodes, the two red nodes, is shown using the green edges. In this case there is a single path which has the shortest length, something which is not always true. If we look at all possible pairs of distinct nodes in this network, we will also see that 4 is the largest value of the shortest path length between such pairs. This length, the largest of all the shortest path lengths, is the definition of the DIAMETER of a network.

ASIDE

The Shortest Path Length is a Metric.
There is a more formal way to discuss the role of the shortest paths in network analysis. The concept of 'distance' is so useful for us in everyday life that we not only use it use it for geographical descriptions but apply it when discussing social systems, we talk about distant cousins or the six-degrees of separation. Not surprisingly, mathematics has often found distance a useful and productive concept in many other situations. As a result mathematics has the following formal set of definitions for a DISTANCE. A DISTANCE [Deza and Deza, 2009] on a set $\mathcal{X}$ is a map $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which satisfies for all $x, y \in \mathcal{X}$

$$d(x,y) \geq 0\,, \qquad \text{non-negativity}, \tag{2.7a}$$
$$d(x,y) = d(y,x)\,, \qquad \text{symmetry}, \tag{2.7b}$$
$$d(x,x) = 0\,, \qquad \text{reflexivity}\,. \tag{2.7c}$$

You will notice that this definition of distance works on any set of objects, it does not require a continuous space, such as the three-dimensional space we live in. For us, the vertices of our network form the natural set of objects. Suppose we adopt the following definition for a formal distance function in a network[a]

$$d(i,j) = \begin{cases} \ell(i,j) & \text{if a path exists from } i \text{ to } j \\ +\infty & \text{otherwise} \end{cases} \tag{2.8}$$

A quick check will show that if we have a simple graph then this distance function satisfies the formal requirements of a distance function (2.7). In fact it obeys a slightly stronger condition that the distance between two vertices is only zero when the two vertices are the same, $d(i,j) = 0$ if and only if $i = j$, the IDENTITY OF INDISCERNIBLES.
If the network is weighted, this distance function ignores the weights but it is straightforward to see this till works if the weights are non-negative and we work with the length of a path equal the sum of weights. Self-loops do not cause problems. A directed network does unless we have a very special case where symmetry is always true.
More interestingly, for an undirected graph, this shortest path based definition of distance between any two points also satisfies the TRIANGLE INEQUALITY

$$d(u,w) \leq d(u,v) + d(v,w)\,. \tag{2.9}$$

The use of $+\infty$ as the distance between disconnected vertices is important for the triangle identity to hold for all vertices[b]. This is trivial in our network context in the sense that by definition, the shortest path between any two vertices u and w in our network can not be longer than a path via any intermediate node v. However, in terms of formal mathematics this extra condition means our shortest distance function becomes what is called a METRIC. These properties for our shortest path distance mean that for undirected networks with non-negative edge weights we can connect up with the the formal mathematical topic of geometry and topology.
For us, the main message is that setting the distance between vertices to be equal to the length of the shortest path is both intuitively obvious and mathematically coherent.

[a] If you want to be even more formal, this definition states that the distance between two vertices is the supremum of the lengths of all paths (or even walks) between the two vertices.

[b] For instance using 0 for the distance between disconnected vertices would break the condition (2.9) needed for a metric.

2.3 Components

A COMPONENT of an undirected graph is a subgraph in which there is a path between all vertices in the component and *no* path between the vertices of the component and the remaining vertices of the graph. Visually components of an undirected graph are separate from other components as shown in Figure 2.9. In terms of set theory, the components define a partition of the set of vertices as every vertex is in one component and only one component.

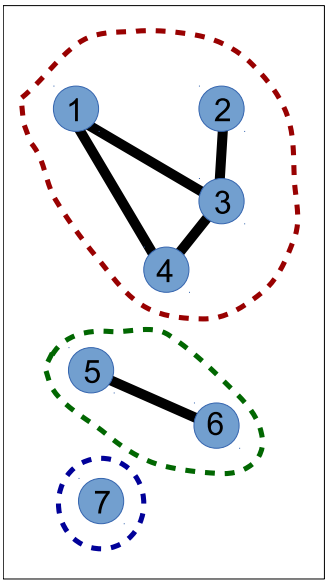


Figure 2.9: A simple graph with three components. The graph here consists of everything in the box indicated by the black solid line and contains all seven vertices and the edges between them. There are three components: the subgraph of vertices $1, 2, 3, 4$ and their edges, the subgraph of edges 5 and 6 plus the edge between them, and finally the subgraph of the single vertex 7 and no edges.

However for a directed graph we have a more complicated structure. If we take account of the direction then we say that a component is STRONGLY CONNECTED if there is a path between every pair of vertices in the component. These paths have to take account of the direction of the edges. That means that the path from some vertex i in a strong component to a vertex j in the same strong component need not be the same as the path from j to i. If we ignore the direction of the edges we are considering the undirected version of the directed graph. The components in this undirected version are referred to as the WEAKLY CONNECTED COMPONENTS of the directed graph.

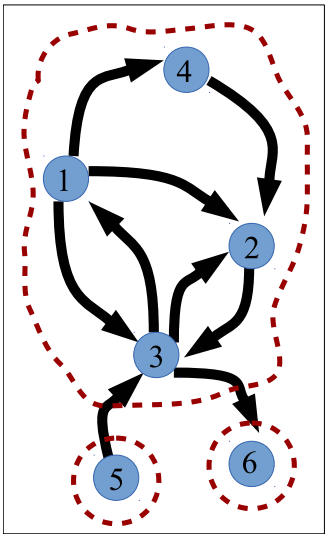
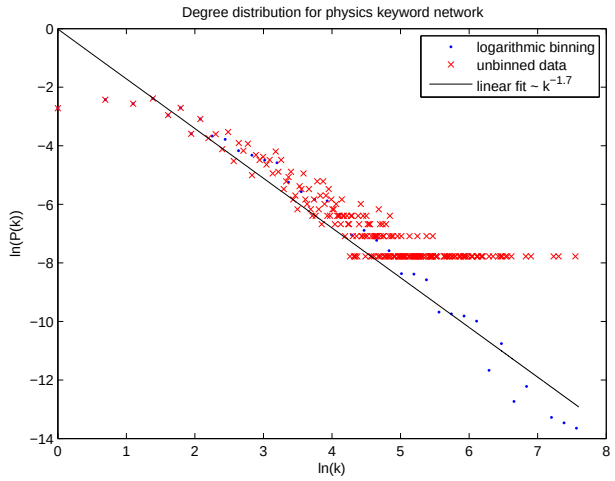


Figure 2.10: The strongly connected components of a directed graph. The graph here consists of everything in the box indicated by the black solid line and contains all seven vertices and the directed edges between them. Note for instance that any path from vertex 4 to vertex 2 can not be the same as any path from 4 to 2 because of the direction of the arrows coming in and out of vertex 4. However these two vertices are linked in both directions so are in the same strong component. The three strongly connected components are indicated by the red dashed lines and are the vertices $\{1, 2, 3, 4\}$ plus edges between them, plus the two single vertex subgraphs based on vertex 5 and 6 respectively. The example graph has a single weakly connected component as when we replace directed edges by undirected ones, the resulting simple graph has a single component.

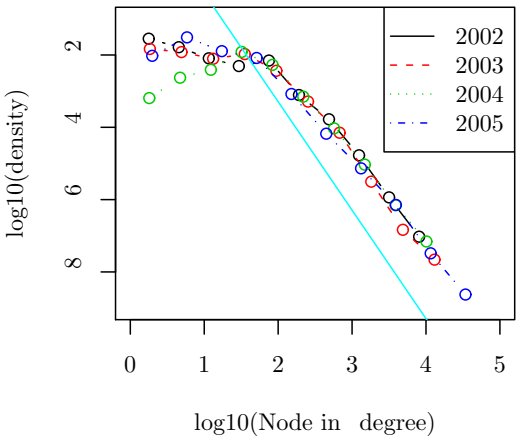
Chapter 3

Fat-Tailed Distributions

In this chapter we will look at the degree distribution of networks, $n(k) = Np(k)$. Our focus will be on those with a FAT-TAIL and apply to discussions of distributions in other contexts with similar mathematical properties, be they distributions of other network measures or distributions on non-network contexts e.g. wealth distributions in economics, word frequency, sizes of earthquakes, and see Fig. 3.1a and Fig. 3.1b.



(a) The degree distribution of a network of words taken from titles of Imperial Physics publications. Words are linked if they appear in same title (unpublished, Weir & Evans 2006).



(b) The degree distribution from a network describing the way users navigate a photo web site. Nodes are pictures stored on the site and these are linked if a user downloaded these two pictures consecutively (from [Argent-Katwala et al., 2007]).

Figure 3.1: Examples of fat-tailed degree distributions from real datasets.

3.1 Definitions of Fat Tails

Loose Definition of a Fat Tail

A loose definition of a fat-tailed distribution $p(k)$ is one where there is a "significant" probability of finding "large" values k.

The trouble is what do we mean here by "significant" or "large"? One way to do this is by making a comparison with another distribution.

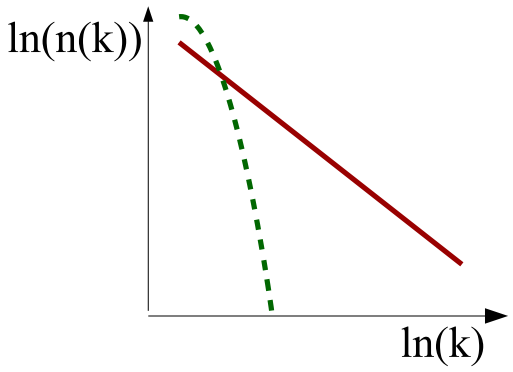


Figure 3.2: A sketch to illustrate power law degree distributions. The red solid straight line is the characteristic shape of a fat-tailed distribution on a log-log scale. The dashed line shows a Poisson distribution with its exponential tail leading to a sharp rapid fall on this log-log scale.

Precise Definition of a Fat Tail

A fat-tailed distribution $p(k)$ is one which decays **slower** than an exponential

$$\lim_{k\to\infty} \frac{p(k)}{\exp\{-k/k_0\}} > O(1)\,. \tag{3.1}$$

In practice what this means is that a fat-tailed distribution $p(k)$ is one where the probability of finding large measurements is a power-law[1] as shown in Fig. **??**. However this precise definition is only useful mathematically as in practice a real network will have a finite number of vertices so the degree k is always finite.

Practical Definition of a Fat Tail

In practice we can say that distribution $p(k)$ is fat-tailed if, at least for the larger values of k, we see a roughly **roughly** linear fall off on a **log-log plot**. We can then say that a power-law, $p(k) \propto k^{-\gamma}$, gives a reasonable description of the tail of the data

$$\ln(p(k)) \sim -\gamma \ln(k) + c\,. \tag{3.2}$$

This is rather a wooly definition but it is often about a precise as the data allows.

3.2 Issues with Fat Tails in Practice

A fat tailed degree distribution, such as the example in Fig. 3.3, means the following:-

- There are always a few vertices (data points) with large degrees (values).
  Large in this context means that they have degrees which scale with the size of the network as a power $k_{\text{large}} \sim N^{-\alpha}$ for some $\alpha > 0$ rather than these being exponentially suppressed $k_{\text{large}} \sim \exp -N/N_0$. Again this is a formal definition which has no precise interpretation when dealing with real data.

- There are many large values of k which are not the degree of any vertex, $n(k) = 0$ for $k \sim k_{\text{large}}$. These zeros cause problems form many simple statistical tests or even when plotting on a log-log axes. These value should not be ignored as they represent the important information that there is a low (if not zero) probability that such nodes exist. In many stochastic theoretical models, running the model many times you will eventually find at least one example run where a node has pone of these missing values.

The values of $n(k)$ for large values of k are crucial for a precise understanding of the shape and generally you need several orders of magnitudes to identify a power-law over other fat-tailed shapes. So how can we improve the situation?

[1] We are ignoring the possibility of having logarithmic corrections $\ln(k)$. With this simplification, all distributions can be split into categories based on the behaviour of their tails, the large k behaviour. The exponential and power law are two important categories.

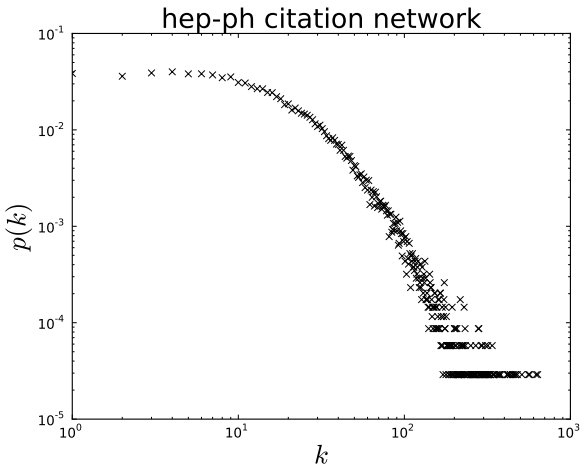


Figure 3.3: Data from the citation graph formed by the papers in the `hep-ph` section of`arXiv.org` repository till 2003 using the data provided for the KDD cup. This has 34546 nodes (papers) and 420921 edges (citations) so here $\langle k \rangle = 12.2$.

Three Approaches to a Fat Tailed Distribution

There are three main ways to deal with lack of non-zero data points for large degree values: log binning, using a cumulative distribution function, and a Zipf plot of rank vs. degree.

Log Binning

When binning data such as from an $n(k)$ distribution, we take values from several values of the ordinand k, and put these into a single bin. So we might consider a value n_b for bin b which is the mean of the values of $n(k)$ for degrees k between k_1 to $(k_2 - 1)$ inclusive. That is

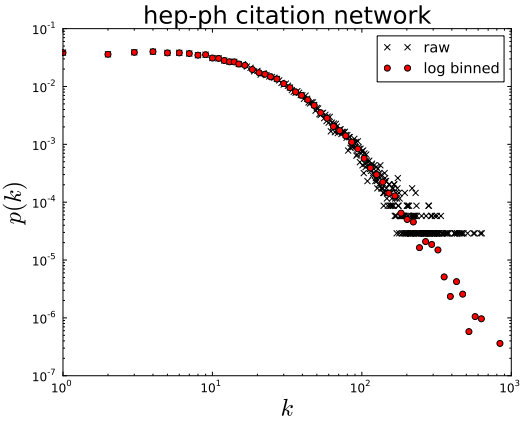
$$n_b = \frac{1}{k_2 - k_1} \sum_{k=k_1}^{k_2-1} n(k) \tag{3.3}$$

The standard practice would be to think that this is a good representation of the value of the distribution at the mid-point $k_b = (k_2 - 1 - k_1)/2$. For instance it would be common to plot a point at k_b with value n_b using the standard deviation of this set of values inn_b to give an estimate of the uncertainty in this point. However such averaging procedures are most appropriate for data without fat tails. For instance in Figure 3.3 we are plotting on log-log scale. We are really looking at points that represent the value of the function $f(x)$ against x where $f(x) = \log(n(k))$ and $x = \ln(k)$. So it is better to scale the bins so the size of each bin goes up by a constant multiple, something known as LOG BINNING, To do this we do the following:

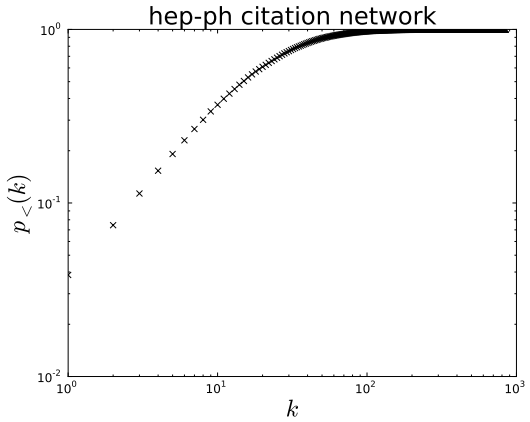
1. Define ith bin to include the values from degree $k = b_i$ to $k = (b_{i+1} - 1)$ where
$$\frac{b_{i+1}}{b_i} = \exp\{\Delta\} \text{ where } \Delta > 0\,. \tag{3.4}$$
2. Count number of vertices with degree in each bin — the sum in (3.3).
3. Plot *density*=(number in bins)/(bin width), n_b of (3.3) at the geometric mid point of bin i.e. at $k_b = \sqrt{b_i(b_{i+1} - 1)}$. Note that on a log scale $x = \ln(k)$ this is the mean of the $x = \ln(k)$ values included in the bin $x_b = \ln(k_b) = (\ln(b_i) + \ln(b_{i+1} - 1))/2$.
4. On log(p(k)) vs log(k) values are equally spaced at intervals of Δ

If we think in terms of a set of values in log space, for $f(x)$ against x, we are just using the usual procedure for taking simple averages to represent this function.

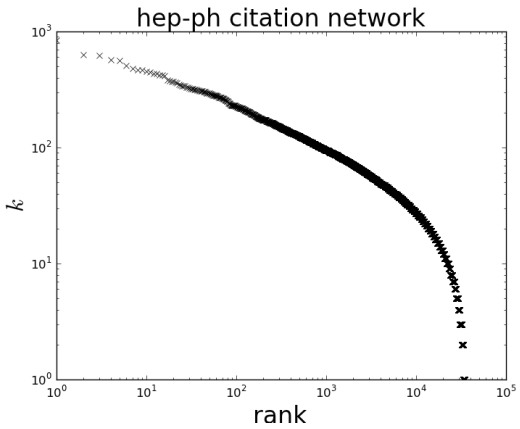
One of the key choices is the scaling Δ, as $\ln(\Delta)$ is the distance between points in the log space. The idea is to ensure no bin is empty. In particular at the tail of the distribution, there are many values of k with zero counts, $n(k) = 0$, between the few values of non-zero counts, typically $n(k) = 1$. However by looking at the average over the bin we are including the information from both these zero values and the non-zero values, the former are missed when using log-log scales. See the example in Figure 3.4a.



(a) The number of papers with a given number of citations, shown without binning (black crosses) and with log bins (red circles) defined using a width $\exp\{\Delta\} = 1.1$, $\Delta = 0.095$.



(b) The CDF (cumulative distribution function) on the same citation counts as Figure 3.4a.



(c) The same citation counts as Figure 3.4a shown as a citation count vs. rank plot, i.e. a Zipf plot. The rank of a paper is defined in defined in (3.6).

Figure 3.4: Different ways to visualise and analyse a fat-tailed distribution. The data in all figures are the citation counts for papers in the hep-ph section of the arXiv repository from 1992-2003.

CDF (Cumulative Distribution Function)

The **CDF** () $p_<(k)$ of a distribution $p(k)$ is defined as

$$p_<(k) = \sum_{k'=0}^{k} p(k') \tag{3.5}$$

This is much smoother than $p(k)$ but is closely related in shape, e.g. CDF is fat tailed if $p(k)$ is fat-tailed. However the CDF is non-zero for large values of k making it easy to plot on log-scales. See the example in Figure 3.4a.

Zipf Plots (1935, 1949)

Zipf plots are named after American linguist George Kingsley Zipf. A ZIPF PLOT is a plot of degree of each vertex against the RANK r of that degree where the vertex of rank r has the r-th largest value. If two vertices have the same rank one can assign the rank for such ties in a variety of ways but in this context, studying the long tail, it is usually best to assign a unique rank to each vertex, breaking any ties at random.

It is simple to see that a fat-tailed degree distribution implies a fat-tailed Zipf plot and vice versa as we can relate the two as follows. If a vertex has rank r and degree k then we may say that

$$r_i = \sum_{k=k_i}^{\infty} n(k) \tag{3.6}$$

at least in the region for $k > k_b$ where there are no ties, $n(k) = 0$ or 1 if $k > k_b$ for some[2] k_b. This means that if the degree distribution has a tail that is well approximated by a powerlaw probability function $p(k) = \frac{A}{k^\gamma}$ then the vertex of rank r will have degree k_r where

$$r = \frac{NA}{(\gamma - 1)} \frac{1}{k_r^{\gamma-1}} \tag{3.7}$$

assuming we have N vertices in total. So if the tail in a log-log Zipf plot is a straight line then this Zipf plot has a slope which is exactly 1.0 less than the slope in the corresponding log-log plot of the degree distribution. The advantage is that the Zipf plot has effectively smoothed out the missing zero value in the fat tail where there is no vertex of that high degree value, $n(k) = 0$, which cause a straight log-log plot of a real $n(k)$ data set so many problems. You may have noticed that the mathematical expressions given here were only for the highest tail regions where at most only one vertex had a degree of a given value, for $k > k_b$ where $n(k) = 0, 1$. Dealing with the data which is ranked is far more complicated mathematically so this approach is not always the simplest in terms of algebra. The cumulative distribution is likely to be more straightforward. However the use of ranked data is a relatively simple way to deal with a single set of real data with a fat-tail distribution using no more than paper, pencil and logarithmic tables. For this reason it plays an important part historically and remains a useful alternative view point when you have a single unique set of data. Zipf was an American linguist and he highlighted the fact that the frequency of words in a text was typically fat-tailed (though others had commented on this before Zipf). Moreover, his claim was that the frequency of a given word was inversely proportional to the rank of that word whatever the document — a relation now often referred to as Zipf's law. That is the slope γ in his log-log plots of the number of times a word appears (playing the role of our degree of a node) vs. rank was often roughly 1 meaning that $\gamma \sim 2$.

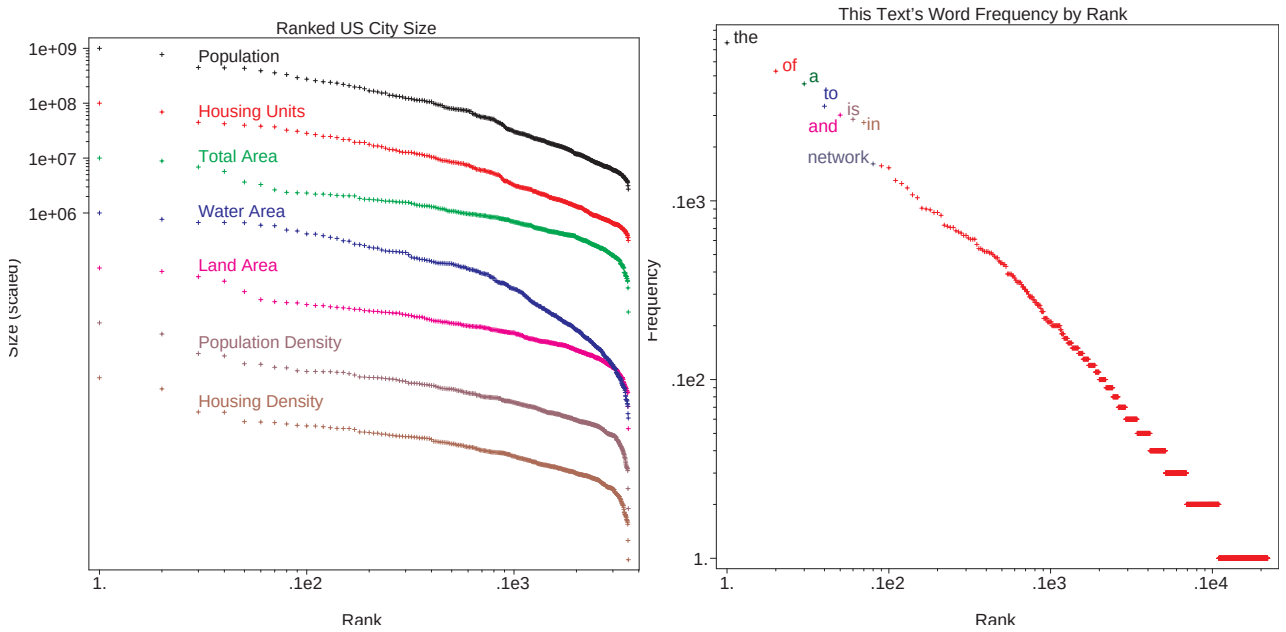
A More Precise Numerical Approach

So far we have highlighted several ways to work with the data to minimise the inherent noise in the tail of a distribution. One could use any of these and then just try simple linear fits to the tail of the tidier form of the data to look for possible power laws.

In practice, these basic ideas are used when we apply more sophisticated tools to become more precise. There are many statistical tools available for comparing data against a possible form. For instance the Kolmorgorov-Smirinov test is based on a comparison of the distance between the CDF of the data and function form. Such methods are nowadays readily available in numerical packages so it need not be difficult to apply these tools. Any good measure will have the added advantage of coming with ready estimates of the goodness of fit. To find the best fit, you would need to use a non-liner optimisation method (again readily available these days) to search through possible values of the powerlaw parameters γ and c of (3.2) along with the tail parameter $k_{\min}$.

The real problem lies in that you could perform a similar fit of the tail of data to many other forms. When faced with real data, we often find that even if data looks as if a power law is a reasonable description of the tail, its roughly linear on a log-log plot, other forms, the log-normal in particular, are often better. The moral of this story is that you should be very wary of assigning a precise form to fat-tailed data. In general real data sets are too small to give any conclusive proof of one functional form over another, one generally needs data over several decades of k values to become confident.

[2]Subscript b for *binary* as for each degree value in this region the number of vertices for each degree value takes one of these two values.



(a) US City size by rank (figure 10 in [Evans, 2004]).

(b) The word frequency distribution in a network review (figure 10 in [Evans, 2004]).

Figure 3.5: These plots illustrate the use of a Zipf plot, size of an object against rank by size of that object.

3.3 Origin of Fat Tails

3.3.1 Classic Distributions without Fat Tails

Let us start by reviewing some of the classic distributions, those which are commonly encountered. If k is a random variable then we can consider the probability $p(k)$ that we measure k to be one of the following standard distributions

- the BINOMIAL DISTRIBUTION $B(k, p, n)$.
  Here k is a non-negative integer and with maximum value n and a real parameter p where $0 \leq p \leq 1$ so that
  $$p(k) = B(k, p, n) = \frac{n!}{(n-k)!k!} p^k (1-p)^{n-k} \tag{3.8}$$
  This has mean $\langle k \rangle = pn$ and variance $\sigma^2 = pn(1-n)$.

- the POISSON DISTRIBUTION $P(k, \lambda)$
  Again k is a non-negative integer but now this has no upper limit and the distribution has a non-negative real parameter λ so that
  $$p(k) = P(k, \lambda) = \frac{\lambda^{-k} e^{-\lambda}}{k!} \tag{3.9}$$
  This has mean and variance $\langle k \rangle = \sigma^2 = \lambda$.

- the NORMAL DISTRIBUTION $P(k, \mu, \sigma)$
  Now k is a continuous real value with no upper or lower limit so it can be negative. The distribution has a non-negative parameters μ. which is equal to the mean $\langle k \rangle = \mu$, and σ which is equal to the standard deviation
  $$p(k) = P(k, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{ -\frac{(k-\mu)^2}{2\sigma^2} \right\} \tag{3.10}$$

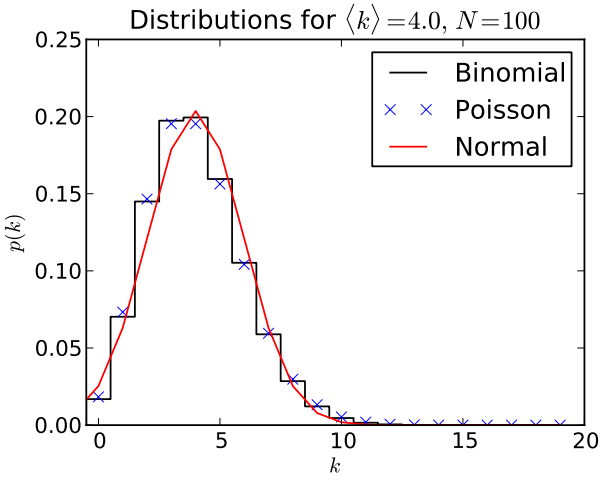


Figure 3.6: The Binomial distribution.

These classic distributions are related. The Binomial becomes the Poisson as $n \to \infty$ with $\lambda = pn$ finite. If $p \gg 1/n$ then the Poisson is approximately a Normal distribution with $\mu = \lambda = np \gg 1$ and $\sigma^2 = \lambda = np(1-p)$.

The Binomial/Poisson/Normal distributions are common because we often **ADD** many fluctuating values together, and the sizes of each value is fairly independent of the other values. Calculating the average height of humans within one interrelated population will show this sort of variation to a good approximation. The mathematical explanation of why many measurements are well approximated by these distributions is encoded by the CENTRAL LIMIT THEOREM. This states that the difference between the average of n measurements of some random variable k and a normal distribution tends to zero as $n \to \infty$ for *any* $p(k)$ *provided* $\langle k \rangle$ and $\langle k^2 \rangle$ are *finite*. That is "data which are influenced by many small and unrelated random effects are approximately normally distributed" (MathWorld)

However one characteristic of this family of distributions and so of many measurements in the real world is that they fall off exponentially fast. That is the chance of measuring a number much bigger than the mean becomes vanishing small very rapidly.

3.3.2 Classic Distributions without Fat Tails

Suppose we **MULTIPLY** random positive numbers $y_i > 0$ $(i = 1, 2, \ldots, n)$ created independently with probability $p(y)$. So the product of these numbers is $Y = \prod_{i=1}^{n} y_i$ and the geometric mean of these numbers is $\tilde{Y} = Y^{1/n}$. The distribution of Y or it's geometric mean $\tilde{Y}$ can be found by taking the logarithm of this function,

$$X = \ln(Y^{1/n}) = \frac{1}{n} \sum_{i=1}^{n} \ln(y_i)\,. \tag{3.11}$$

We are now working with the average, X, and variance of the random variables $x_i = \ln(y_i)$. As each y_i is independent of the other measurements, so are their logarithms. Provided the average and variance of the random variables $x_i = \ln(y_i)$ is finite (remember that $\ln(\langle k \rangle)$ is different from $\langle \ln(k) \rangle$ so this not completely trivial) we can then apply the central limit theorem to find that the sum or average of a large number of $x_i = \ln(k_i)$ is normally distributed whatever the form of $p(y)$, the distribution of $\ln(y)$. That is we have that

$$p(X) \approx \frac{1}{s\sqrt{2\pi}} \exp\left\{ -\frac{(X-m)^2}{2s^2} \right\}. \tag{3.12}$$

This means that the distribution of the geometric mean is a LOG-NORMAL DISTRIBUTION

$$p(Y) \approx \frac{1}{Y s\sqrt{2\pi}} \exp\left\{ -\frac{(\ln(Y)-m)^2}{2s^2} \right\} \tag{3.13}$$

as shown in Fig. 3.7. Care is needed as the two parameters of the log normal distribution, here m and s, are *not* the mean and standard deviation of Y but the (arithmetic) mean and standard deviation of $X = \ln(Y)$ as $m = \langle \ln(y_i) \rangle$, $s = \langle (\ln(y_i) - m)^2 \rangle$. The log normal is a **fat-tailed distribution** in

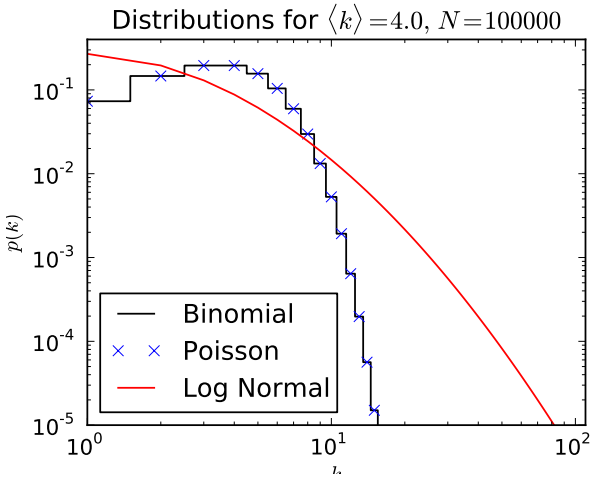


Figure 3.7: Log normal distribution (red line) against the Binomial and Poisson distributions for the same mean k.

Y, e.g. compare against plot of random variable with same mean ($\langle Y \rangle = \exp\{\mu + s^2/2\}$) and variance ($\sigma_Y^2 = \exp\{s^2 - 1\}.\exp\{2\mu + s^2\}$).

3.3.3 Yule/Power Law as a Fat Tail

The simplest continuous distribution decaying slower than exponential is a POWER-LAW DISTRIBUTION

$$p(k) = \frac{A}{k^\gamma} \qquad k \in \mathbb{R} \tag{3.14}$$

but this is defined for any real valued k while in our network context the degree k is discrete. The discrete equivalent of the power-law is the YULE DISTRIBUTION named after work by Yule in 1925. However the distribution has been used extensively in many contexts such as in the classic work by Simon [Simon, 1955].

$$p(k) = \frac{(r+1)!(k-1)!}{(k+r)!}, \qquad k \in \mathbb{N} \tag{3.15}$$

This is the natural discrete k generalisation of the continuous k power-law of (3.14). For instance, for large k the Yule distribution of (3.15) becomes a power law of the form (3.14) $\lim_{k\to\infty} p(k) = k^{-(r+1)}$ with $\gamma = (r+1)$. So again we find that moments of the Yule distribution, $\langle k^\alpha \rangle$, do not exist for $\alpha \geq (\gamma - 1) = r$.

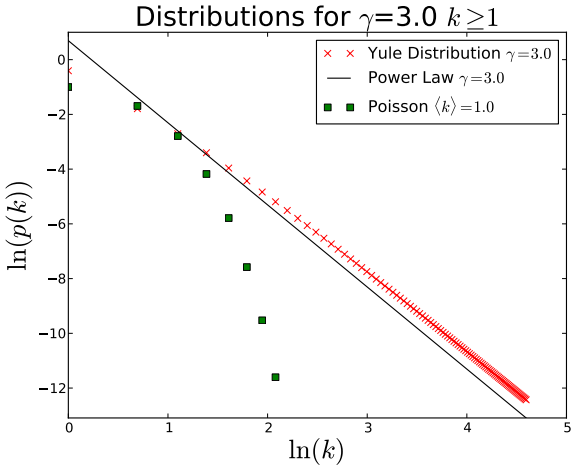


Figure 3.8: Yule distribution which has a power law tail

An interesting question is to ask what sort of process leads to a Yule/Power-Law distribution? That is a question we will ask in the next chapter.

Generic Fat Tails: Summary

We have

- seen that the Configuration Model can be used as a random graph null model for networks with fat-tailed degree distributions,
- defined fat-tailed degree distributions,
- seen how to organise fat-tailed data in useful ways e.g. for plotting,
- shown that multiplicative processes lead to a log-normal distribution as an example of a process leading to a fat-tailed distribution,
- defined the Power-Law and Yule distributions as the standard examples of fat-tailed distributions for continuous and discrete variables respectively.

Chapter 4

Random Networks

Suppose we have a network created from a data set or a theoretical model and we make some measurements on this network: the degree distribution, the number of components or whatever. How do we assess the meaning of the values we obtain? How do we know if a degree of a vertex is unusually large? How do we know if the average path length is short? The way to make these statements is to make a comparison with some appropriate standard, what are known as NULL MODELS.[1] That is we make measurements on the network of real interest, one coming from data or a sophisticated model, and compare this against measurements made on our null model.

4.1 Erdős-Réyni Random graph

Around 1959, Erdős[2] and Réyni produced a number of ground breaking results [Erdős and Réyni, 1959] for what is the simplest model of a COMPLEX NETWORK, i.e. something which is not a regular lattice. However some of these results had also been obtained earlier, by Solomonoff and Rapoport in 1951 [Solomonoff and Rapoport, 1951], but the model is usually linked to the names of Erdős and Réyni. Such complex networks are usually much more important when a network has been created by many small parts interacting without access to detailed global information about the structure they are creating. For instance in the wiring of the brain and other biological systems, physical systems created by many interacting groups of humans such as the hardware underlying the internet, and social networks such as friendship networks.

The model creates simple graphs in one of two ways:

A N vertices in which each pair of vertices is connected with probability p, independent of all other pairs,

B N vertices with E vertices connected chosen at random with uniform probability from $N(N-1)/2$ pairs.

As both definitions are stochastic, it means each defines an *ensemble* of graphs, that is a set $\mathcal{G} = \{G_1, G_2, \ldots, \}$ of individual simple graphs G_i, not a single graph. The difference is that in the set created with definition (A) the number of edges varies between different graphs while the graphs created using definition (B) all have the same number of edges. So we have $\mathcal{G}_B \subseteq \mathcal{G}_A$. The difference between the two definitions is just the same as the difference between the canonical and microcanonical ensembles in statistical mechanics. Just as with these statistical physics ensembles, there is no difference between the two random graph models in the limit of infinite size $N \to \infty$.

[1] See L10 of [Clauset, 2013].

[2] Paul Erdős was a famous itinerant mathematician who produced a prodigious number of papers with a vast number of different coauthors, at least in terms of the norms of mathematics. As a result mathematicians like to play a game where they calculate their Erdős number. This is the shortest path from an author to Erdős in the COAUTHORSHIP NETWORK, that is a network with researchers as the vertices linked if they have coauthored a paper. My Erdős number is at most 5 with three different short paths via my collaborators Umezawa, H., Kibble, Tom W.B. and Rivers, R.J. Note the family name of Erdős' has a double acute accent on the o, ő (unicode $U + 0151$, &#337 in HTML, $\backslash H\{o\}$ in LaTeX.), and is not a simple umlaut, ö.

Most precise mathematical results are derived in this infinite graph limit and so mathematicians work with the definition which is easiest mathematically, definition A. Computationally, method B is better because only E random numbers are considered (choosing one of N vertices) whereas method A requires a random number for all T possible vertex pairs, so $T = N(N-1)/2$. The differences between the two models are likely to be small and of little practical interest unless N is very small. Which ever way we approach our ensembles, an ensemble of Erdős-Réyni random graphs make the the simplest possible null models. A typical use of these models is to construct the Erdős-Réyni random graphs to have the same number of vertices and edges as the network of interest to us, and we compare the measurements we make. We will work with definition A in this text.

Since these models create an ensemble of graphs, the measurements we make should be defined to be averages over these random graphs. So, if $\mathcal{G}_A$ is the set of graphs created under definition A, we are interested in

$$\langle Q \rangle_A = \frac{\sum_{g \in \mathcal{G}_A} Q(g)}{\sum_{g \in \mathcal{G}_A} 1} \tag{4.1}$$

where $Q(g)$ is the value of some quantity A measured on graph g.

For definition A the degree distribution is simply a binomial distribution

$$p(k) = \binom{N-1}{k} p^k (1-p)^{N-1-k}\,, \tag{4.2}$$

since we have $N-1$ as the model does not allow self-loops. That is because every edge is created independently of any other edge. So if we look at one vertex it has $(N-1)$ potential edges. Each one is present with probability p and not present with probability $(1-p)$. If k particular edges are present that occurs with probability p^k while the remaining $(N-1-k)$ edges are not present with probability $(1-p)^{N-1-k}$. As it does not matter which k edges are present there are $\binom{N-1}{k}$ different ways we can choose the k edges which are present. The final point is that we could do this for any vertex, there would be no difference in the argument. So this distribution is also what we would measure if we asked what is the degree of a vertex chosen uniformly at random from the set of vertices, the probability distribution $p(k)$.

From the properties of a binomial distribution we deduce that that the average degree is $\langle k \rangle = p(N-1)$. If we work from our definition in terms of an average over the ensemble of graphs as have that

$$\begin{aligned} \langle k \rangle_A &= \frac{1}{\sum_{g \in \mathcal{G}_A} 1} \sum_{g \in \mathcal{G}_A} \frac{1}{N} \sum_{i \in \mathcal{V}k} k_i \\ &= \sum_k p(k) k = p(N-1) \end{aligned} \tag{4.3}$$

Note that we use the fact that there is no difference between vertices in this model, so that on average they will all have the same properties so each term in teh sum over individual vertices gives the same answer. The binomial degree distribution encode the result of measuring degree over the ensemble A for any N so that is how we deal with the sum over graphs.

As there are N vertices the average degree tells us that in this ensemble A of random networks the number of edges is simply $\langle E \rangle = pN(N-1)/2$. Thus when want to use this model to make a comparison with a second network of N_2 vertices and everage degree $\langle k \rangle_2$ edges, we would chose the parameters of our random graph ensemble A to be choose $N = N_2$ and $p = \langle k \rangle_2/(N_2 - 1)$.

It is useful to remember that in the limit $N \to \infty$ with $\lambda = pN$ held constant, the binomial distributions becomes the Poisson distribution

$$p(k) = \frac{\lambda^k e^{-\lambda}}{k!}\,, \text{ with } \lambda = pN \tag{4.4}$$

So for this large graph limit we have that the standard deviation for the average degree is just $\langle k \rangle / \sqrt{N}$. That is the ensemble A of random graphs becomes increasingly like a REGULAR GRAPH, a graph where all vertices have the same degree. For the same reason the total number of edges becomes almost always

equal to $E = \langle k \rangle N/2 = pN^2/2$. Thus when these random graph models become infinite ($N \to \infty$) the two ensembles A and B become identical[3].

4.2 Configuration Model

4.2.1 Motivation: Fat Tails and Null Models

We have seen FAT-TAILED DEGREE DISTRIBUTIONS are common, e.g. many network have HUBS — vertices with large numbers of neighbours. However, Erdős-Réyni random graphs with binomial degree distribution which has an exponential cutoff and so this is not fat-tailed. For networks with a fat-tailed distribution we need a better NULL MODEL, that is a model against which we can compare when investigating network problems. The CONFIGURATION MODEL allows us to specify any degree distribution we want. Typically we would take the same degree distribution as the network we are investigating.

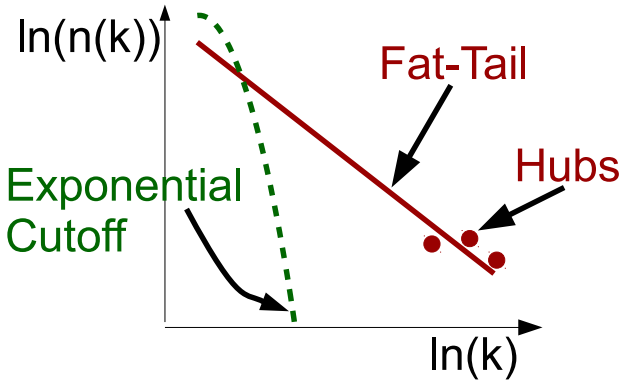


Figure 4.1: Sketch of two possible degree distributions for same total number of vertices and edges. The green dashed line shows an distribution with an exponential tail, such as an ER graph. The red solid line shows a distribution with a power-law tail. The lines represent a typical result averaged over many realisations for large N in an appropriate model. In practice for finite N, each realisation of a model with a power law distribution has a few nodes with very large degree k which are known as hubs, as indicated on the diagram.

4.2.2 Definition

The CONFIGURATION MODEL is a random graph where the degree of each vertex $\{\tilde{k}_i\}$ is given. This was discussed by Molloy and Reed [Molloy and Reed, 1995] but the model has been discussed at least as far back as 1972 see discussion and references[4] in [Dorogovtsev et al., 2003]. There are two variations, each of which leads to a slightly different ensemble of networks.

A - $\mathcal{G}(N, p(k))$ N vertices in which each pair of vertices (i, j) is connected with probability $p(i, j) = \tilde{k}_i \tilde{k}_j/(2E)^2$, independent of all other pairs,

B - $\mathcal{G}(N, \{k_i\})$ N vertices, vertex i has degree $k_i = \tilde{k}_i$ exactly.

These are stochastic definitions defining ensembles of graphs $\mathcal{G} = \{G_1, G_2, \ldots\}$. Definition **A**, $\mathcal{G}(N, p(k))$, is typically best for mathematical work. Definition **B**, $\mathcal{G}(N, \{k_i\})$ is easier to implement numerically. The definitions equivalent in the large graph limit $N \to \infty$ c.f. canonical and microcanonical ensembles. If degree distribution is Binomial then get we get an ER random graph.

[3]This is true for exactly the same reasons that the canonical and microcanonical ensembles in statistical mechanics become identical in the thermodynamic limit.

[4]A. Bekessy, P. Bekessy, J. Komlos, Studia Sci. Math. Hungar. 7 (1972) 343; E.A. Bender, E.R. Canfield, J. Combin. Theory A 24 (1978) 296; B. Bollobás, Eur. J. Combin. 1 (1980) 311; N.C. Wormald, J. Combin. Theory B 31 (1981) 156,168.

In our definition we stated that the configuration model was a 'random graph'. What this means is that these random graphs have **no statistical correlations** between neighbouring vertices. That is in our ensemble of graphs, the only thing which remains constant over all ensebles is the degree of each vertex. The relationship between vertices, the edges, is randomised.

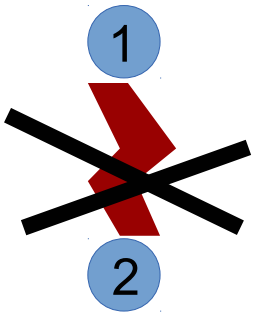
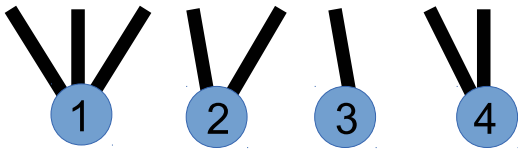


Figure 4.2: A typical assumption for the simplest analytical solutions is that there are no vertex correlations. That is the properties of one node are not related to tyhe properties of any of its neighbours.

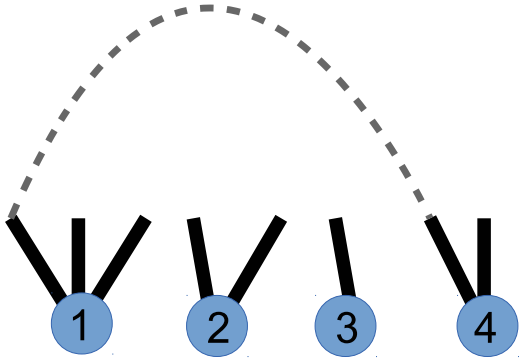
Numerical Implementation

One numerical implementation starts from a given degree distribution $\{k_i\}$ and the set of nodes [Newman et al., 2001]. Definition **B**, $\mathcal{G}(N, \{k_i\})$, with list of $\{k_i\}$) given.

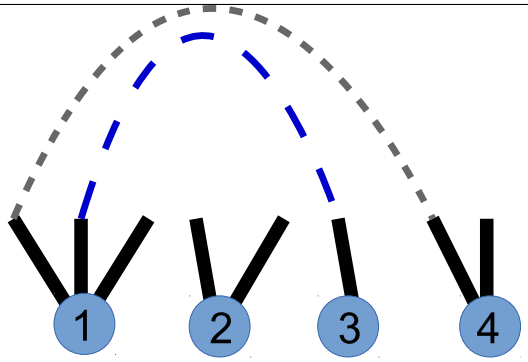
1. Assign k_i stubs (half an edge) to each vertex i.
2. Choose two stubs at random and join them into an edge.
3. Repeat until all stubs joined.



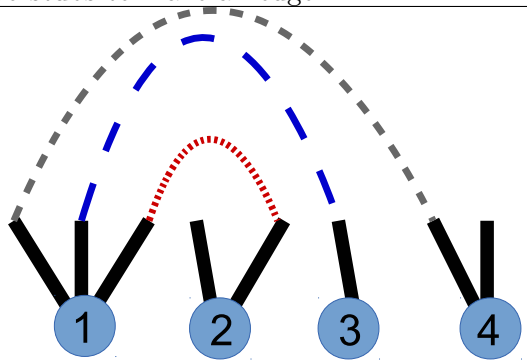
Stage 1: Specify the degree k_i for each vertex i and give each k_i stubs (half an edge)



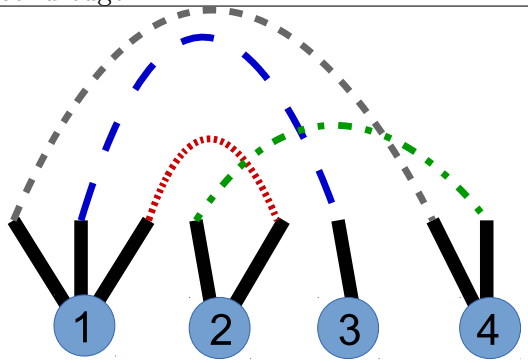
Stage 2: Pick two distinct stubs uniformly at random from the set of stubs. Connect these two stubs to make an edge.



Stage 3: Pick more two distinct stubs uniformly at random from the set of remaining stubs. Connect these two stubs to make an second edge.



Stage 4: Repeat this process for remaining stubs.



Stage 5: Finish when all stubs are paired off as edges.

Figure 4.3: Numerical implementation of configuration model from a list of degrees

In an alternative numerical implementation [Maslov and Sneppen, 2002] we start from an actual network. Definition **B**, $\mathcal{G}(N, \{k_i\})$, with existing network given e.g a data set

1. Choose two edges at random.
2. Swap one end of each edge with each other.
3. Repeat until every edge visited once.

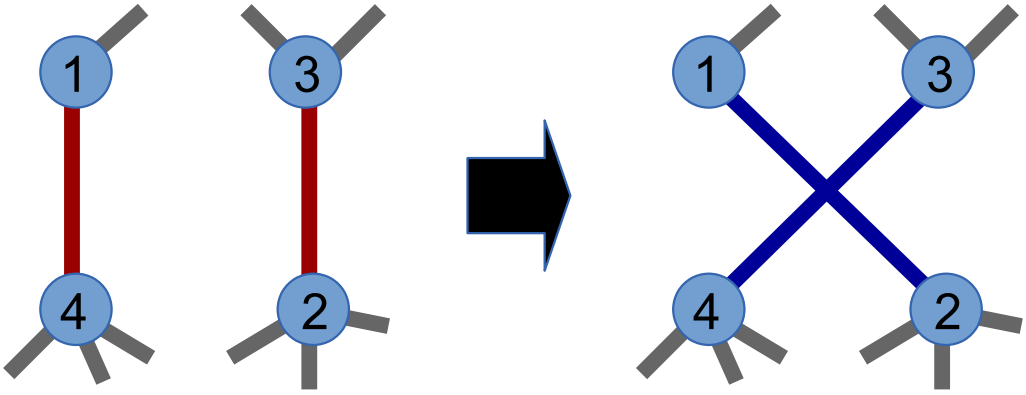


Figure 4.4: Illustration of how to rewire a pair of edges which preserves the degree of vertices.

4.2.3 Your Neighbour's Degree

We can exploit lack of statistical correlations between neighbouring vertices in the configuration model. To illustrate this we will ask what is the degree of your neighbour in a random graph. The answer seems obvious that the degree of my neighbours will be on average $\langle k \rangle$ but in fact this is not true. So consider the following process. Start by choosing a vertex at random. All are equivalent statistically, so we are choosing a given vertex i with probability $1/N$. Now choose one neighbouring vertex at random, say j. The probability of arriving at a neighbouring vertex with degree is **not** $1/N$. What we have actually done is first choose an edge attached to our source vertex i at random in order to leave vertex i. Since the configuration model randomises the edges, destroys the vertex correlation, the target end of this edge will, on average across the ensemble, have nothing to do with the source vertex i we started from. That is we are picking at edge uniformity at random from the set of all edges in the graph. Now the probability vertex j is at end of this random edge is equal to[5] $k_j/(2E)$. This is because this is the fraction of edges attached to j and we are on a random edge. Put another way k_j is the number of ways of arriving at a vertex.

This means that the probability distribution for the degree of your neighbour in a random graph is proportional to $kp(k)$. Normalisingmeans that we have

$$\boxed{p_{\rm nn}(k) = \frac{k}{\langle k \rangle} p(k)} \tag{4.5}$$

1. It is *not* just $p(k)$, we are not simply choosing a random vertex
2. The average degree of your neighbour $\langle k_{\rm nn} \rangle$ is

$$\langle k_{\rm nn} \rangle = \sum_k k\, p_{\rm nn}(k) = \sum_k k \frac{k}{\langle k \rangle} p(k) = \frac{\langle k^2 \rangle}{\langle k \rangle}$$

Your friend is more popular than you! (as $\langle k^2 \rangle \geq (\langle k \rangle)^2$) though *only* if friendship is random.

[5] This means we could be returning to vertex i in principle. We could exclude this but for any reasonably large graph it is going to be a small correction so we ignore the issue of self-loops here.

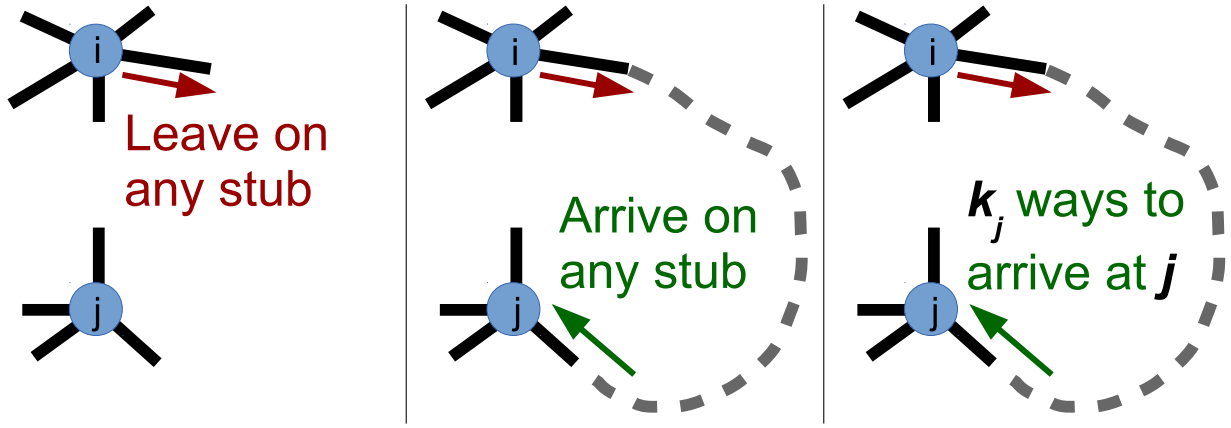


Figure 4.5: Illustration of the how we arrive at a neighbouring vertex j in a random graph when starting from vertex i.

4.3 Price Model

The model put forward by Derek de Solla Price de Solla Price, Derek Price—see de Solla Price (see Figure 4.6) in 1976 [Price, 1976] and reworked by Barabási and Albert [Barabási and Albert, 1999] is a simple model which generates a network whose degree distributions has a fat-tail. The most natural context for this model is that of citations between documents, the original context of Price's work Price [1965a,b, 1976].

Aside Derek de Solla Price (1922–1983)

Derek de Solla Price was a physicist who became a historian of science. He was one of the first to measure and analyse scientific literature, the field of SCIENTOMETRICS (part of bibliometrics), so is sometimes seen as the "father of scientometrics". He put conjectured that if in any one field N authors produced P publications, then $\sqrt{N}$ of those authors would have produced $P/2$ of those publications. He suggested that the number of scientific papers produced per year were increasing exponentially, and looked at the half-life of a papers — the time it took to gather the first half of their total citations. Price's 1976 model, which we discuss here, gives the growth of citation networks using a principle of CUMULATIVE ADVANTAGE [Price, 1976], — now usually referred to as a PREFERENTIAL ATTACHMENT process.

Figure 4.6: Derek de Solla Price shown here with a reconstruction of the Antikythera mechanism.

4.3.1 Citation Networks

In a CITATION NETWORK a vertex represents one document while the edges represent references, CITATIONS, made from one document to another document so there these edges have a natural direction, see Figure 4.7. Examples include the citations found in academic papers and books, patents (which must refer to relevant "prior art", that is other recent relevant patents), or court judgements which build their case on previous legal judgements.

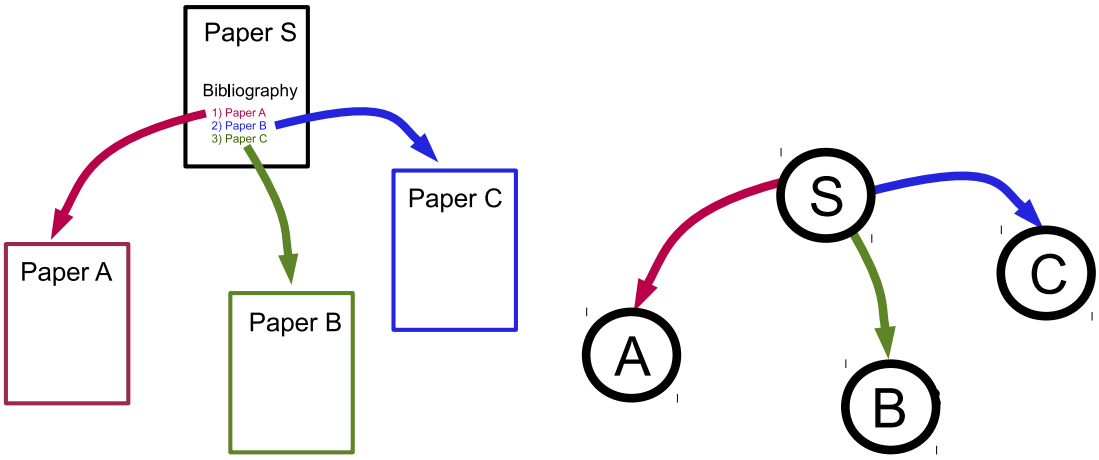


Figure 4.7: Sketch of how a citation network is defined. On the left we illustrate citations made in the source document S to older documents A, B, and C. On the right is the corresponding network with nodes representing the corresponding documents. The edges are directed where we have chosen the direction to point away from the source document making the citation and so the edges point towards the older documents being cited.

One of the key observations about citation networks is that the tail of the degree distribution is generally found to be fat-tailed, often well summarised by a power law description, see Figure 4.8.

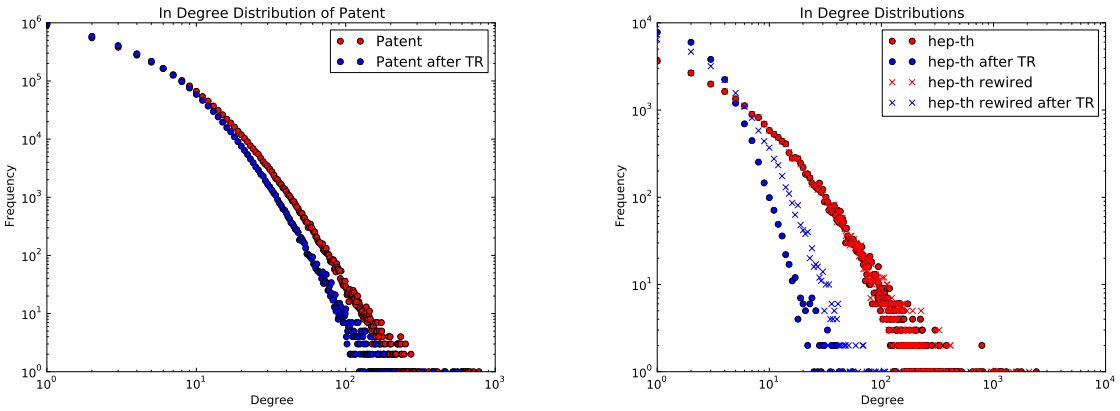


Figure 4.8: On the left the number of citations (the in-degree) for patents has a fat-tailed distribution ($N = 3.810^6$, $\langle k^{(\text{in})} \rangle = 12.8$, figures: Clough et al. 2014 [Clough et al., 2015]). On the right the Zipf plot for the number of citations received by each paper (in-degree) against the rank of the paper (ranked by in-degree) for papers in the hep-th arXiv section ($N = 2.710^4$, $\langle k^{(\text{in})} \rangle = 8.8$). The corollary is that many academic papers (10% - 30%) have zero citations. In both cases we see that a few papers/patents get most of the citations.

To explain these fat-tails Price [Price, 1976] proposed a model in which a newly published paper cites older papers with a probability proportional to the number of existing citations each older paper

has at that time. This is his principle of CUMULATIVE ADVANTAGE — the more citations you have the more likely you are to gain more, though Price was inspired [Price, 1965a] by the application by Merton [Merton, 1968] of Matthew's principle to the bibliometric context. This Cumulative Advantage principle is also used in many other contexts and appears under different names:

- The Pareto principle or the 80–20 rule, 19th cent.
- The Matthew effect, Matthew's gospel "For everyone who has will be given more"
- The Yule distribution [Yule, 1925] (1925) emerging from a description of biological taxa and subtaxa
- Simon's model for the frequency of words in text (**??**)
- The BARABÁSI-ALBERT MODEL (**BA**) used for web sites [Barabási and Albert, 1999].

Barabási and Albert used the term PREFERENTIAL ATTACHMENT to describe this idea and it is this term which has become the most popular in recent years in the context of networks.

4.3.2 The Price/Barabási-Albert network model

We can define the Price and BA (Barabási-Albert) network models as follows

1. Set up an initial seed (citation) network.
2. Increment time $t \to t + 1$
3. Add one new vertex — represents the t-th oldest document in the model.
4. Add m edges to the new document — the number of distinct documents referred to in the bibliography of the new document
5. Connect the other end of each new edge to an existing documents with probability proportional to the in-degree $k^{(\text{in})}$ (Price) or total degree k (BA) of the existing vertices
6. Repeat from (2).

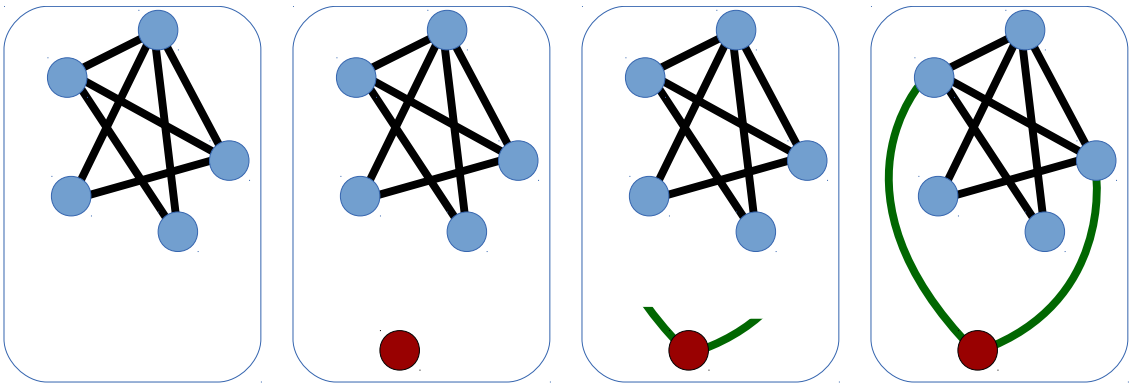
These rules are illustrated in Figure 4.9. Note that the original versions of the Price [Price, 1976] and Barabási-Albert [Barabási and Albert, 1999] models were both rather more limited. The rules given here cover generalisations that lead to a family of closely related models with the characteristic that they all have fat tailed degree distributions (with the exception of some limiting cases).

The only real difference between the Price [Price, 1976] and Barabási-Albert [Barabási and Albert, 1999] versions of the model is whether we make the edges directed or not, whether or not we use the in-degree or total degree to control the probability of attaching edges to existing documents. The mathematical analysis also works in a similar way for both cases. However for simplicity, in what follows we will use undirected edges and assume the probability of attachment is proportional to the degree — the BA form of the model.

Also note that in this model the 'time' parameter t is really a ranked time and is not necessarily an actual physical time. That is the paper published at time t is the t-th oldest paper in the model. Of course this model is a gross simplification of the process of citation. For instance in reality the length of the bibliography of each paper is not constant[6] so we should not have a single fixed value of the parameter m. Nevertheless, the purpose is to show that such a simple process is sufficient to create a fat-tiled distribution of the type shown in Figure 4.8.

The final comment is that this model is *not* the same as the configuration model with the same degree. The growth of the Price/BA model in time means that while the model is stochastic, it is random in a very particular way. The best way to see this is to use the edge swapping mechanism to produce graphs of the same degree distribution but with the edges otherwise randomised. Most network measurements will show a difference between this randomised network and the original Price/BA network.

[6]The number of papers in a bibliography, the out-degree, is often a fat-tailed distribution in its own right.



(a) (b) (c) (d)

Figure 4.9: Illustration of how the network grows in the BA (Barabási-Albert model). The initial network is shown in (a). Then in (b) a new vertex is added, the 6-th oldest so $t = 6$. In (c) we illustrate the idea that m new edges are attached to the new vertex but then we have to choose which vertices in the existing graph are attached to the other end of these m new edges. Once this has been done, using preferential attachment in the BA model, we arrive at the final network shown in (d). The cycle is then repeated.

4.3.3 BA Model Mean Field Master Equation

Master equations usually describe the evolution of the state of a system and enforce probability balance requirements. Here we will build the master equation for the degree distribution in the BA model, i.e. for the undirected version of the model. There are many other features of the networks created in this model but our equation will not capture anything beyond the degree distribution.

Even with this limited aim, to write down an equation we make further simplifying assumptions. The model is stochastic, that is each time we run the model a slightly different set of edges are added. For instance an edge will rarely be added to a low degree vertex but it does happen sometimes. The outcome of one update at time t could be any one of an extremely large number of different networks, $\{\mathcal{G}(t+1)\}$. The probability of reaching any one of these networks is easily found in principle from the rules of the model but the expressions are far too complicated and there are far too many possible networks for these results to be useful. So we do not ask questions about the exact network we arrive at, the $\mathcal{G}(t+1)$, but rather we ask about typical general properties of such a network. In terms of mathematics, we are going to take an ensemble average and ask what is the average value of some quantity Q. To be clear we ought to adopt a notation such as

$$\langle Q(t)\rangle = \sum_{G\in\mathcal{G}(t)} p(G)\,Q(G)\,. \tag{4.6}$$

Here $\mathcal{G}(t)$ is the set of possible networks at time t and $p(G)$ is the probability that we arrived at network G. The angular brackets $\langle\ldots\rangle$ indicate that we are taking this ensemble average.

Finding the average value $\langle Q(t)\rangle$ is only useful if $\langle Q(t)\rangle$ is typical of the result we get when we measure this quantity $Q(t)$ in any *one* run of the model. If the model produces a wide range of different results for $Q(t)$ every time we run the model then the average $\langle Q(t)\rangle$ is not going to be a very useful indicator. Fortunately in many cases we find that individual results are usually close to the ensemble average with little variation for large systems (e.g. the central limit theorem applies). This use of average values in our equation is known as a "mean field approximation". So we will build an approximate equation which states the average value for the number of nodes with degree k at time t, depends only on the *average* value of the degree distribution at earlier time steps.

In our case we are interested in the degree distribution so we want to find $\langle n(k,t)\rangle$, the number of nodes with degree k (undirected edges considered here) at time t averaged over all ensembles. In fact the normal practice is to drop the angular brackets $\langle\ldots\rangle$ so here we will use the standard notation of $n(k,t)$ for $\langle n(k,t)\rangle$.

The master equation will be an expression for the number of nodes of degree k at the next time step $(t+1)$, that is $n(k,t+1)$, given the distribution at previous times. Here the simple rules for the model mean that we only need to know the degree distribution at the current time t, i.e. $n(k,t)$, to predict what the results of adding m edges will be[7]. Suppose that $\Pi(k,t)$ is the probability that one of the new edges at time t is connected to an existing node with degree k. For our BA case we have $\Pi(k,t) = k/(2E(t))$ where $E(t)$ is the number of edges at time t. The MASTER EQUATION for the degree distribution in the BA model is then

$$\begin{array}{lll} n(k,t+1) = & n(k,t) & \text{(Start with this number.)} \\ & +m\Pi(k-1,t)n(k-1,t) & \text{(Effect of edges added to vertices with degree } (k-1)) \\ & -m\Pi(k,t)n(k,t) & \text{(Effect of edges added to vertices with degree } k) \\ & +\delta_{k,m} & \text{(The one new node has } m \text{ edges)} \end{array} \tag{4.7}$$

It is important to note that average number of edges added to nodes of degree $(k-1)$ at time t in the model as defined above is only roughly equal to the average number of $m\Pi(k-1,t)n(k-1,t)$. Again this turns out to be reasonable assumption, one we can check numerically posthoc.

This equation is an example of a difference equation which is the discrete variable (here k) equivalent of a differential equation in a continuous variable. The tricks used to solve such difference equations mimic those used in differential equations. For instance a differential equation requires boundary conditions to be specified and the same is true here. In our case we note immediately that physically (if not mathematically) we have at least that no vertex can have negative degree[8] so $n(k,t) = 0$ if $k < 0$. Equally, as we phrased this in terms of an evolution, is is natural to specify a boundary condition in time, that is the shape of the degree distribution at some initial time.

To solve this difference equation we first note that the number of edges grows as

$$E(t) \;=\; mt + N(t=0)\,. \tag{4.8}$$

For simplicity we will assume that[9] $N(t=0) = 0$. With this simplification we have that

$$n(k,t+1) \;=\; n(k,t) + \frac{m\,(k-1)\,n(k-1,t)}{2mN(t)} - \frac{m\,k\,n(k,t)}{2mN(t)} + \delta_{k,m}\,. \tag{4.9}$$

If we write the master equation in terms of the probability distribution $p(k,t) = n(k,t)/N(t)$ we find that

$$N(t+1)p(k,t+1) - N(t)p(k,t) \;=\; +\frac{1}{2}(k-1)\,p(k-1,t) - \frac{1}{2}k\,p(k,t) + \delta_{k,m}\,. \tag{4.10}$$

Here we will search for a solution for large times, equivalently for large networks since the number of vertices always growing as[10] $N(t) = t$. The ideas is that $n(k,t)$ must evolve too but there will be a trivial overall growth factor which reflects the growth in the number of vertices $\sum_k n(k,t) = N(t) = t$. So we factor this out by working with the probability distribution $p(k,t) = n(k,t)/N(t)$ and we search for a stable form for the probability distribution

$$\lim_{t\to\infty} p(k,t) = p_\infty(k)\,. \tag{4.11}$$

Assuming such a form exists in the long time, large N limit, we then have that

$$p_\infty(k) = \frac{1}{2}\left[(k-1)\,p_\infty(k-1) - k\,p_\infty(k)\right] + \delta_{k,m}\,. \tag{4.12}$$

[7]This means we have a MARKOV PROCESS, i.e. one where you do not need the whole history of the evolution of the system to predict its future evolution.

[8]In fact every vertex added always has at least m edges as these are added immediately. So only vertices in the initial network can ever have degree between 0 and m so one could always assume an initial network where all vertices have m or more edges. In that case $\Pi(k,t)n(k,t) = 0$ for all $k < m$.

[9]This assumption can be relaxed but this is left as an exercise for the reader. With a suitable definition of time coordinates and a suitable initial condition, this may also be implemented numerically.

[10]We may always choose to count time steps t such that this is true from some initial time t_0 equal to the number of nodes in the initial network G_0.

The factor of N in (4.10) has cancelled and the form of (4.12) does indeed suggest that there is a time and N independent $p_\infty(k)$ solution as we assumed.

It is possible to find an exact solution for the difference equation (4.12) but this is unusual, typically such master equations are too complicated for exact solutions. We will use an approach which can be used in many situations where the there is no exact solution for the difference equation and which is sufficient to reveal tail of the degree distribution in this model. So let us consider k large in (4.12). That means we can ignore the low k boundary term $\delta_{k,m}$. We also see that the remaining terms on the right hand side of (4.12) form an approximation of a derivative in k as follows

$$p(k) \approx -\frac{1}{2}\frac{k\,p_\infty(k) - (k-\Delta k)p_\infty(k-\Delta k)}{\Delta k} \tag{4.13}$$

$$\Rightarrow \quad p(k) \approx -\frac{1}{2}\frac{\partial(k\,p_\infty(k))}{\partial k} \tag{4.14}$$

where $\Delta k = 1$. A PDE such as this is much more familiar than the discrete difference equation form of them aster equation and we can solve this in a variety of ways. As we are looking for a power law, the simplest way is to make the ansatz that $p(k) = Ak^{-\gamma}$ and to substitute this into (4.14) to see if we can make this form satisfy the equation. If we do this we find a solution to (4.14) of the form

$$\lim_{t\to\infty} p(k,t) = p_\infty(k) \propto \frac{1}{k^3} \tag{4.15}$$

i.e. the tail of the degree distribution is a power law with power $\gamma = 3$.

We can check our solution numerically. In this case we simulate the ensemble average by making a number of runs for the same parameter values, stopping at the same time but each time changing the random numbers used to make the probabilistic decisions. The ensemble averages are approximated by making averages over the networks created at the end of the runs. Of course we can never access all possible networks but for many quantities the numerical simulations will work quickly to enough to produce examples to provide a good approximation, though this ought to be checked numerically. In the same way we can never run the simulation for infinite time but the hope is a 'long' time will be practical and again checking what time is long enough is a trickier part of the numerical simulation of this and many other stochastic models. An example of is shown in Figure 4.10.

4.3.4 (#) Generating Functions

Generating functions are a powerful mathematical technique and are useful in many situations. The master equation of the Price/BA model can provide an illustration of their use with difference equations for functions of a discrete variable.

A useful analogy is with the use of integral transforms when facing partial differential equations for functions of a continuous variable, such as time t. Examples of these include

$$\text{Fourier} \qquad f(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} d\omega\ \exp\{i\omega t\}\ \tilde{f}(\omega)\,, \tag{4.16}$$

$$\text{Laplace} \qquad f(t) = \frac{1}{2\pi i}\int_{\gamma-iT}^{\gamma+iT} ds\ \exp\{st\}\ \tilde{f}(s)\,, \tag{4.17}$$

$$\text{Mellin} \qquad f(t) = \frac{1}{2\pi i}\int_{\gamma-i\infty}^{\gamma+i\infty} dz\ t^{-z}\ \tilde{f}(z)\,. \tag{4.18}$$

In each case we replace a function of interest, in terms of one continuous variable, with new function in terms of a new continuous variable. Usually the original function is terms of a physical variable we can interpret easily (illustrated here by time t). The transformed function is expressed in terms of a variable may be much harder to interpret. Here the new variables are (angular) frequency ω, often with a natural interpretation, but s and z typically harder to interpret physically. Even if the variable is hard to interpret, the advantage is that if you choose the right transform then a solution should become apparent.

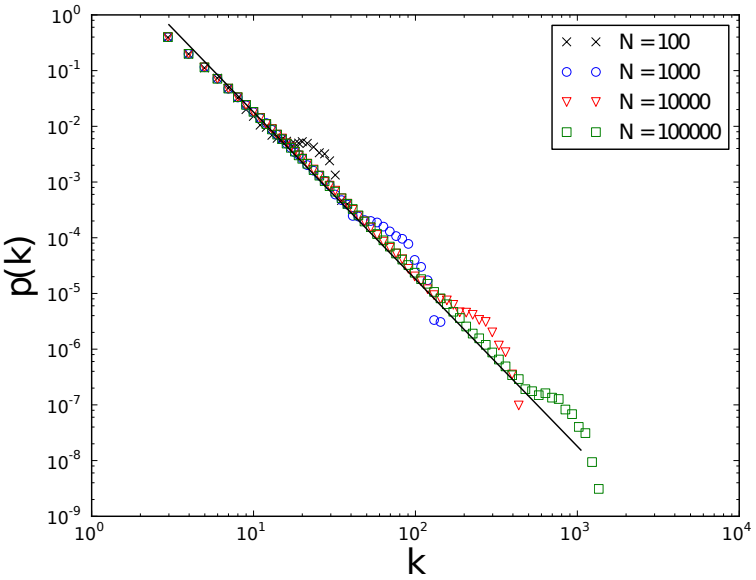


Figure 4.10: Numerical simulations of the BA model for $m = 3$ and several different values of N. Each point represents the density found in one logarithmic bin (where each bin is roughly 1.2 times larger than the previous bin) for one run of a model at a given N value. The line represents a power law with form k^{-3}.

Generating functions are ubiquitous in physics. PARTITION FUNCTIONS in statistical mechanics are good examples. For instance, if the macrostates of the system are characterised by an energy and the number of particles in the system, then the partition function is defined as the sum over all possible states of the exponential of the energy and particle number, each term scaled by a parameter, the inverse temperature β and the chemical potential μ respectively. The result is a partition function, a function of the continuous variables — $Z(\beta, \mu) = \sum_i \exp\{-\beta(E_i - \mu N_i)\}$. We have switched from a description in terms of energy and particle number to a description in terms of the inverse temperature β and the chemical potential μ.

In quantum field theory, sums over different possible quantum field configurations ϕ (the different states) of $\exp\{iS/\hbar\}$ gives generating functions of the Green functions used in QFT which is a function of unphysical sources j — $Z[j] = \sum_\phi \exp\{-(i/\hbar)(S[\phi] - j\phi)\}$ for the action $S[\phi]$ of the theory.

Let us illustrate the use of generating functions for in the context of degree distributions $p(k)$. The aim is to package all the information recorded in the $p(k)$ in a GENERATING FUNCTION $G(z)$ by defining

$$G(z) = \sum_{k=0}^{\infty} z^k p(k) = p(0) + zp(1) + z^2 p(2) + \dots . \tag{4.19}$$

Here the continuous variable z has no simple interpretation, and it is replacing the discrete variable k which is easily understood as the physical degree of nodes. The generating $G(z)$ still has all the information of $p(k)$ but it is wrapped up in a different mathematical form. Looking at (4.19) we see it is just a discrete version of an integral transform. There are several advantages to having a generating function

- We can invert by taking k-th derivative at $z = 0$

$$p(k) = \frac{1}{k!} \left. \frac{d^k G}{dz^k} \right|_{z=0} . \tag{4.20}$$

  Note that the analogy with the Mellin integral transform (4.18) shows another way to write the inverse

$$p(k) = \frac{1}{2\pi i} \oint dz \; z^{-k-1} G(z) . \tag{4.21}$$

- The MOMENTS of distribution are easy to obtain

$$\langle k^n \rangle = \left(z\frac{d}{dz} \right)^k G(z,t)\bigg|_{z=1} = \frac{d^k}{d(\ln(z))^k} G(z,t)\bigg|_{z=1} . \tag{4.22}$$

- the FACTORIAL MOMENTS of distribution are even easier to obtain

$$\langle k(k-1)\dots(k-n+1) \rangle = \left\langle \frac{\Gamma(k+1)}{\Gamma(k-n)} \right\rangle = \frac{d^n G}{dz^n}\bigg|_{z=1} . \tag{4.23}$$

We can use our (approximate) master equation for the BA model to illustrate another use of a generating function. Just as we can use integral transforms to rewrite PDE for functions of a continuous variable in a different, hopefully simpler form, we can use generating functions to replace a difference equation, such as (4.12), by a PDE for the generating function. Typically we are more experienced in finding solutions to a PDE.

So we start with the difference form of our BA master equation

$$p_\infty(k) = \frac{1}{2}(k-1)\, p_\infty(k-1) - \frac{1}{2} k\, p_\infty(k) + \delta_{k,m} . \tag{4.24}$$

Multiply this equation by z^k and then sum over k, using $G(z) = \sum_{k=0}^{\infty} z^k p(k)$. We find that[11]

$$G_\infty(z) = \frac{1}{2} z(z-1)\frac{\partial G_\infty(z)}{\partial z} + z^m . \tag{4.25}$$

Now we have a PDE encoding the master equation, a much more familiar beast.

Price/BA model: Summary

We have

- defined the Price/BA model in terms of a growing network model and the principle of cumulative advantage or preferential attachment,
- given citations as an example where such a mechanism might be relevant
- set up the master equation for the model
- shown the long-time large-degree solution to be a power law with power $\gamma = 3$
- stated the properties of Generating Functions
- given the Price/BA mater equation as a PDE for the generating function

[11] Try starting with the generating function form (4.25) and substitute in the definition of G to recover the original form of the master equation in terms of $p(k)$ (4.12).

Chapter 5

Phase Transitions

5.1 Phase transition in an ER Random graph

For the ER random graphs, when $p = 0$ we get a collection of N disconnected vertices while for $p = 1$ we get the complete graph of N nodes. Clearly there is a point in between where on average we get some kind of network which is largely connected, paths exist between most vertices. It is useful to look at how this transition occurs.

Suppose we calculate the average path length ℓ between vertices. Consider a very low p where there are a finite number of edges present, so that we must have $p \sim 1/N^2$. This means $\langle k \rangle \approx pN \sim 1/N \ll 1$ so most vertices have degree zero and are therefore disconnected. As we defined our shortest path ℓ (2.6) in terms of connected pairs only, these disconnected vertices contribute nothing to our measure ℓ. The few edges which do exist are almost certain to connect pairs of vertices of degree 1. Each of these vertex pairs will contribute 1 to our definition of the average path length. There are no larger components as at this point there will be essentially no vertices of degree two. That is since $p(2) \approx p(1)(pN)/2$ (using the Poisson approximation of the degree distribution (4.4)) we deduce that $p(2) \sim 1/N^2$. We only expect a degree k vertex in each graph if $p(k) \gtrsim 1/N$ as this is the probability that one vertex in N will have this property, and this is roughly the minimum number of vertices needed for the property to be observed. So with for this current situation, there is essentially no vertex of degree two or higher. The only non-trivial paths are of length 1 and so our measure of the average path length, ℓ of (2.6), will be almost exactly equal to 1. We can also talk about the average size of components. Most vertices have no edges so each forms a component of size 1. The few connected pairs represent a component of size 2 but the average will be dominated by the smaller components.

Now imagine increasing p until $p(2) \sim 1/N$. Now we are beginning to get a few degree two vertices the the vast majority are degree 1 or degree zero as before. Most components will still be a pair of vertices connected by a single edge, contributing 1 to the ℓ measure. However the few rare degree two vertices are almost certain to be part of a line of three vertices with the degree two vertex in the middle since there is little chance of these rare degree two vertices connecting to each other. So larger components are highly unlikely at this point[1] For these rare components with three vertices, the two vertices at the end are separated by a shortest path of length 2 so each of these components contribute $(1/3)(2+1+1) = 4/3$ to the ℓ measure. Thus we see thatℓ is now a bit bigger than 1 and we can see ℓ is rising as p increases. We also see that the average size of a component is rising, as these connected vertex pairs represent a component of size 2 are becoming more dominant.

As we continue to raise p we get more and more vertices of degree 2. If the number of degree 3 vertices remains insignificant, all that can happen is that it becomes more and more likely that we get longer chains[2] as well as more connected pairs. Longer chains mean that more pairs are separated by longer paths, ℓ rises. We can also see that the sizes of the components is growing.

At some point as we continue to add edges, raise p, these extra edges join together what would have been the small components found at lower p values and discussed above. The largest components,

[1] **Q** Can you explain why?

[2] **Q** Estimate this behaviour for the case where $p(2) \gtrsim 1/N$ but $p(3) \ll 1/N$ for large N.

with many vertices, have more chances that such extra edges will connect them to other components, and so these are likely to grow quicker than smaller components. We find that the LCC, the LARGEST CONNECTED COMPONENT, starts to dominate in size over smaller components. It will start to have a significant fraction of all vertices. Consider the point at which the average degree of the network is around one, $pN \sim 1$. We know from the Poisson form of the degree distribution (4.4) that about $e^{-1} \approx 37\%$ of vertices are disconnected (degree 0), another 37% have degree 1, 18% have degree two, 6% have degree 3 and the remaining 2% have degree 4 or higher. Clearly any connected component of three or more vertices must have some vertices of degree two or higher. Its conceivable that the components, and the LCC in particular are tree like, chains of degree two vertices ending at a degree three branching point or a terminating degree one vertex. Such trees have some very long shortest path lengths[3]. Thus it suggests that the average shortest path length ℓ can become very large. One can imagine other possibilities, but we shall support this picture with detailed calculations below.

Continuing to add more edges (rising p) means that more and more small components are connected to the LCC. The size of this largest component continues to grow, the smaller components therefore have to shrink in size and number. It also turns out that average path length starts to reduce. What is happening is adding more edges does not connect pairs which were not connected before. Most pairs are in the LCC and are already connected. What these extra edges do is provide alternative paths between pairs. If these additional edges create longer routes between pairs, then the average shortest path length measure ℓ will be unaffected. However sometimes they will create shorter routes in which case the *spl* measure will come down.

We can make this picture more precise mathematically if we take the limit $N \to \infty$, pN fixed i.e. a finite non-zero average degree. Formally we find that there is a *phase transition* at some critical value of the average degree. For low average degree all components, even the largest (the LCC) contain a finite number of vertices. Average path lengths measured by ℓ are also finite. As the average degree rises there is a critical value for the average degree above which the LCC suddenly has an infinite number of vertices. This is a true critical point in the system marking a phase transition in the usual sense of statistical physics. The largest connected component is now the GCC, the GIANT CONNECTED COMPONENT. A GCC is defined to be a component with containing a finite fraction of vertices of the graph. One consequence is that the GCC is only strictly defined in the $N \to infty$ limit so it always has an infinite number of vertices, just not necessarily all of them. So for average degrees above this critical value, the largest connected component is now also a GCC. With an infinite number of vertices in the GCC it is conceivable that the distance scales could also get large[4] In fact we find that the length scales above the critical point are infinite, scaling as $\ell \sim \ln(N)$.

Clearly in any numerical experiment, N is never infinite. In such cases the LCC is an approximation to the GCC seen in the infinite graph limit. So we should see that there is a region where small increases in average degree lead to a dramatic increases in the size of the LCC.

Let us now take a formal look at this infinite graph limit. Let u be the average fraction of vertices in an ER random graph $G(np)$ which are **not** in the GCC. Put another way, u is the probability that a randomly chosen vertex does not lie in the GCC. Consider a vertex i which is not in the GCC and pick one of the $N-1$ other vertices as a possible neighbour, say j. The probability that there is no edge from i to j is $(1-p)$. On the other hand with probability p there is an edge to j and now we must insist that j is not in the GCC given that i is not. The vertex j is not in the GCC with probability u. Put these together and we see that pu is the probability that i is connected to j but j is not in the GCC. If these two cases are satisfied then we know that vertex i is not connected to the GCC via vertex j and this is the situation with probability u_j where

$$u_j = \underbrace{(1-p)}_{\text{not connected to } j} + \underbrace{up}_{\text{connected to a } j \text{ not in GCC}} . \tag{5.1}$$

The factor is the same for every possible vertex in the graph (being an ER random all the vertices are statistically equivalent) so we need to multiply these probabilities together for the $(N-1)$ vertices of

[3]For a tree there is no difference between the longest path between two points and the shortest path between those two points.

[4]However, we can also imagine some infinite graphs which have a finite ℓ. A star configuration for instance.

the rest of the graph to find the probability that i is not part of the GCC. Since this is exactly what we defined u to be we have then that

$$u = ((1-p) + up)^{N-1} \tag{5.2}$$

Taking logs we can rearrange this equation as follows

$$\begin{aligned} \ln(u) &= (N-1)\ln(1-p+up) && (5.3)\\ &\approx N.p(u-1) \qquad \text{for } N\to\infty\,, \quad p\to 0 && (5.4)\\ \Rightarrow \quad u &= \exp\{-\langle k\rangle(1-u)\} && (5.5) \end{aligned}$$

Note that this is relation *exact* in the large sparse limit $N \to \infty$, $\langle k\rangle \sim O(1)$.

It is normal to think in terms of those vertices in the GCC. The fraction of vertices in the GCC is just its size $S = (1-u)$ so that we find

$$S = 1 - \exp\{-\langle k\rangle S\} \qquad \text{for } N\to\infty\,, \langle k\rangle \sim O(1)\,. \tag{5.6}$$

The ER Random graph phase transition is revealed if we solve our equation (5.6) for the size S of the GCC. However equation (5.6) is a transcendental equation with no simple analytical solution. It is, though, straightforward to solve numerically. As you can see in figure 5.1 the transition happens at

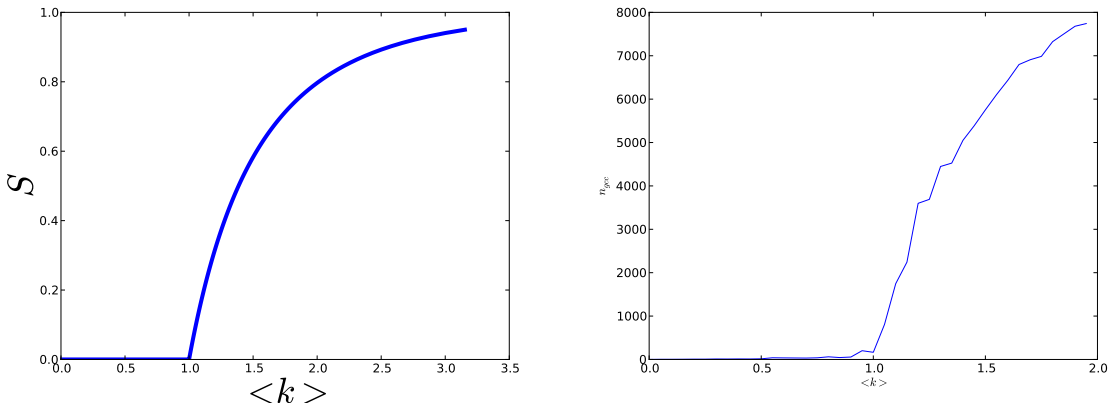


Figure 5.1: On the left the analytic equation for the size S of the GCC (Giant Connected Component) against the average degree $\langle k\rangle$ in an ER graph. On the right the size of LCC (Largest Connected Component) for single realisations of ER graphs with $N = 10^4$ vertices but different $\langle k\rangle$ (lines interpolating between values to aid visulaisation).

an average degree of about 1.00. A careful analysis of the equation near $\langle k\rangle = 1$ confirms the existence of a positive real solution for S only for $\langle k\rangle \geq 1$. We can also see that $\langle k\rangle = 1$ is the critical value when we look at the behaviour of path lengths in the ER random graph.

5.1.1 Shortest Path Length in ER Random graph

Consider our usual infinite sparse graph regime, $N \to \infty$, $\langle k\rangle \sim O(1)$. Suppose we find a *core* of vertices forming a small but connected subgraph $\mathcal{G}_0$ with a finite number of vertices $n_0 = |\mathcal{V}_0|$. The *average* number of edges from the core to the other $(N - n_0)$ vertices is just

$$p(N-n_0)n_0 \approx \langle k\rangle n_0\,. \tag{5.7}$$

Now we note that these edges, from a core vertex to a vertex outside the core, always pick out a unique new vertex lying outside the core. This follows since there are an infinite number of vertices to choose from outside the core as $(N - n_0) \approx N$ but we are only adding a finite number of edges $n_0\langle k\rangle$. So add these edges and the neighbouring vertices to the core subgraph to get a new connected subgraph which we will call $\mathcal{G}_1$. This has $n_1 = \langle k\rangle n_0$ extra vertices giving us $N_1 = \langle k\rangle n_0 + n_0$ in total.

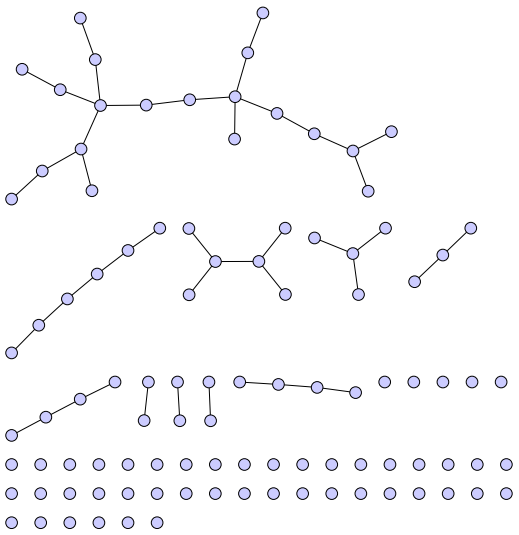


Figure 5.2: One realisation of an ER random graph for $\langle k\rangle = 1$ and$N = 100$.

Now we repeat this argument ℓ times. We find that the number of vertices we add at each step to our growing subgraph, those vertices at distance ℓ from the core, is $\langle k\rangle$ times the numbers we added at the last step, so that $n_\ell = \langle k\rangle n_{\ell-1} = \langle k\rangle^\ell n_0$. At each stage the number of edges is $p(N-n_\ell)n_\ell \approx pN \approx \langle k\rangle n_0$ since n_ℓ is finite and N is formally infinite. Again, for small ℓ in an infinite random graph we can be sure that every time we add another n_ℓ edges to connect to neighbouring vertices of the connected subgraph $\mathcal{G}_\ell$ there is zero chance these edges connect to the same vertex, so we always get $\langle k\rangle n_\ell$ vertices to add at each stage. Put another way, we are saying that the ER random graph looks like a tree and has no loops. This is also the structure formed when we are looking at a BRANCHING PROCESS.

The subtlety comes in that we want to look for the appearance of an infinite path length from the core to other vertices, that is to see when the core becomes part of a true giant connected component. As long as we are at values of $\langle k\rangle$ which are below the critical value (so we always have a finite sized component) and as long as $N \to \infty$ we are fine. We can even be arbitrarily close to the critical point as long as we stay strictly at $\langle k\rangle < \langle k\rangle_c$. Proving that ER random graphs are true trees in the sparse graph limit $N \to \infty$, $\langle k\rangle \to \langle k\rangle_c$ requires some care but it can be done rigorously. Here we will just accept this intuitive result.

What this means is that the critical point is at exactly $\langle k\rangle = 1$ as for $\langle k\rangle < 1$ the growth of our connected component is always limited. In the sub-critical region $\langle k\rangle < \langle k\rangle_c$, the number of vertices added gets less and less at each stage and we expect the average component will be roughly of the size C where, recognising a geometric series, we find

$$C = \sum_{\ell=0}^{\infty} \langle k\rangle^\ell n_0 = \frac{n_0}{1-\langle k\rangle}\,, \qquad \langle k\rangle < 1\,. \tag{5.8}$$

The average path length from a vertex in the core to any other vertex it is connected to is also easy to find as it is given by

$$\ell = \frac{1}{C}\sum_{\ell=0}^{\infty} \ell\,\langle k\rangle^\ell\, n_0 \quad = \quad \frac{\langle k\rangle}{(1-\langle k\rangle)}\,, \qquad \langle k\rangle < 1\,. \tag{5.9}$$

In fact all path lengths, ℓ will be finite even as $N \to \infty$ provided we stay in the sub-critical region $\langle k\rangle < 1$. On the other hand, this length scale is clearly diverging as we approach $\langle k\rangle = 1$ showing us that $\langle k\rangle_c = 1$ is indeed the critical value.

In the supercritical region where $\langle k\rangle > \langle k\rangle_c = 1$ the number of vertices being added is growing at each stage. In fact it is growing exponentially fast as $n_\ell = \langle k\rangle^\ell n_0 = \exp\{\beta\ell\} n_0$ where $\beta = \ln\langle k\rangle > 0$. Think of N very large but finite, leaving the limit $N \to \infty$ to be taken at end of our calculation. In

the supercritical region, our process of adding $n_0\langle k\rangle^\ell$ vertices at each step will only stop when we run out of vertices to add. That is when $n_0\langle k\rangle^\ell \approx N$. Thus the largest distance from core is[5]

$$n_0\langle k\rangle_{\max} \approx N \qquad \Rightarrow \qquad \ell_{\max} \approx \frac{1}{\ln(\langle k\rangle)}\ln(N)\,, \qquad \langle k\rangle > 1\,. \tag{5.10}$$

Many people are connected in the supercritical as $S > 0$ yet the distance between people in the GCC scales as $\ln(N)$. Had we been in a finite d dimensional space we would expect something like a $N^{1/d}$ scaling as $N \to \infty$. This slow rise of the length scale is indicating that we have what is called a Small World network.

The critical point $\langle k\rangle = \langle k\rangle_c = 1$ is where we have neither growth nor decay in the number added at each stage. Rather we have, on average, a constant number being added at each stage. This suggests we could end up with an infinite sized component, its just that it grows far more slowly when compared to the exponential growth rate of the component in the supercritical region. Technically we have to be rather more careful with our limits in this case, and take into account fluctuations in the number of edges present at each step rather than the average number we have used in our calculations. However our argument here highlights the key features of the behaviour and also highlights the correct answer.

5.2 Phase transition in Configuration model

Path Lengths in Random Graphs

Working as usual in large sparse graph regime, $N \to \infty$, $\langle k\rangle \sim O(1)$, consider one initial vertex i with degree k_i. This has We $n_1 = k_i$ vertices (the nearest neighbours of i) one step away. However the the average degree of these neighbours in a random graph is not $\langle k\rangle$ but z where

$$z = \langle k^2\rangle/\langle k\rangle\,, \tag{5.11}$$

just as explained above. There is no correlation between the ends of an edge in an ensemble of random graphs[6] Of the z edges attached to each neighbour, one edge is back to vertex i while the remaining $(z-1)$ edges may be to new neighbours. Since we are assuming a large sparse graph, these new neighbours are unlikely to be the same vertex[7]. Thus the number of vertices two steps away, n_2, is

$$n_2 = (z-1)n_1 \qquad \text{with } z = \frac{\langle k^2\rangle}{\langle k\rangle} \tag{5.12}$$

Repeating we find the number of vertices ℓ steps away from vertex i is

$$n_\ell = (z-1)^{\ell-1}n_1\,. \tag{5.13}$$

Once again, at each stage we can consider adding all the vertices distance distance ℓ or less from i to form a subgraph $\mathcal{G}_\ell$ provided we also include any edges between these vertices. Again at each stage these subgraphs look like a tree — they contain no loops. That is every time we add new vertices and their edges, these edges are all pointing to new neighbours (apart from the edge which led us from the one vertex which is closer to i) since we have a large and sparse graph.

This process is also known as a BREADTH FIRST SEARCH in terms of computer algorithms. That is if we wanted to investigate all vertices in the same connected component as our starting vertex, one way is to list all the neighbours, layer by layer, processing all the vertices found ℓ steps out from the starting vertex before moving on to look at all vertices found at the next level $(\ell+1)$ and so forth[8].

[5] You may use a geometric progression to produce a better estimate from the total number number of nodes in this component but it will still have the same basic $\ln(N)$ scaling feature.

[6] For one very large example, averaging over all edges will also show the same effect with no correlations on average over all edges in a large graph.

[7] More formally in the $N \to \infty$ limit we will find the probability that any of these edges has one end at a verte we have already considered tends to zero.

[8] The natural alternative is to use a DEPTH FIRST SEARCH where you create a tree by going as deep as possible at each stage,so each new vertex is one step further away than the previous, until you can go no deeper. You then back up the tree until you find first vertex (so with the highest value of ℓ) which still has uninvestigated alternative roots

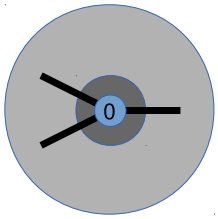
The trees we are finding can also be viewed as those formed by a BRANCHING PROCESS, one where the average number of branches in the tree is z — one back up the tree *and* $(z-1)$ branches reaching deeper into the tree, see Figure 5.3.

So we find that in the random graph in a configuration model starting from vertex i, at each stage we are adding n_ℓ vertices which are ℓ steps from vertex i where

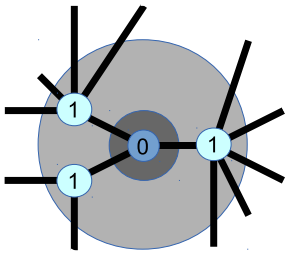
$$n_\ell = (z-1)^{\ell-1} n_1 \,, \qquad \ell > 0 \,, \tag{5.14}$$

$$z = \frac{\langle k^2 \rangle}{\langle k \rangle} \,. \tag{5.15}$$

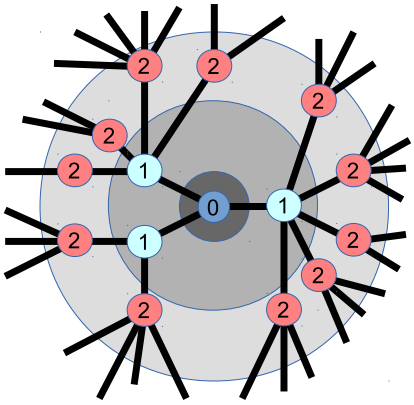
Note that the first number added is $n_1 = k_i$ whose average (over different randomly chosen starting vertices) is $\langle k \rangle$ not $z = \frac{\langle k^2 \rangle}{\langle k \rangle}$ however we are interested in the bahviour of this expression for large ℓ.



Pick first vertex, label with zero.



Visit all neighbours of first vertex and label these with 1.



Visit all neighbours of vertices labelled 1 provided these neighbours have no label. Label these neighbours with 2.

Figure 5.3: An example of the 'breadth first' creation of a connected component starting from vertex labelled 0. The label of a vertex gives the length of the shortest path from vertex 0 to that vertex. This is DIJSTRA'S ALGORITHM [**?**] for finding the shortest paths in a network.

Now we can look at the generalisation to general random graphs of the phase transition found earlier for the ER random graph. From the analysis above we can see that the number of vertices added to our sequence of subgraphs is growing as $\sim (z-1)^\ell$. Thus the critical point is where $(z-1) = 1$ i.e. $z_c = 2$. For lower z, the number added is always getting smaller. For larger z, the number added at each stage is growing exponentially (on average). For $z = 2$ we get constant growth, no acceleration or deceleration in the rate on average. It is easy to understand why $z = 2$ is the critical value. Since random graphs are trees in the large sparse graph limit are trees where the average degree of nodes in this tree are $z = \langle k^2 \rangle / \langle k \rangle$. This is because they were chosen to be added to the tree because they were nearest neighbours of an existing vertex in the tree and z is the average degree of nearest neighbours. If we arrive on one edge (from a vertex closer to the starting vertex i) then we will only continue the process if there is at least one more edge to leave on, to find vertices ever further away from i. So in the configuration model the critical point is at $z_c = 2$ where nearest neighbours have degree two: **one edge in, one edge out**.

The ER random graph fits perfectly into this pattern as it is straightforward to check that the critical point is at $z_c = 2$ in that case.

A couple of questions may come to mind with this analysis.

First we remember that we say the average degree of a node in this network is actually $\langle k \rangle$. This is almost always smaller, and certainly it is never bigger, than z. So clearly the way we are selecting vertices to add to our component is biassed, we tend to select the the vertices with more edges than those with few. Of course if we were to choose a vertex uniformly at random from the whole network

we would find its degree was $\langle k \rangle$ on average. So one thing to bear in mind here is that our growing tree, our breadth first search, is not choosing vertices uniformity at random from the set of all vertices. It is actually equivalent to first choosing uniformity at random from the set of edges and then choosing one end at random. So even in a random graph there are natural ways to choose vertices at random but with different biasses.

The second is to remember that not all vertices are in this large component, even if $z > 1$. Only a finite fraction are in the largest component. So you must remember the above analysis is what happens on average. It would be quite possible to choose a vertex i with no neighbours, its just $p(k = 0)$ which is often non-zero. Even if no vertex has zero neighbours it could have k_i neighbours but all of those neighbours by chance have just one edge, which would happen with probability $p(k_i)(p(k = 1)/\langle k \rangle)^{k_i}$. In general, there is a finite probability that we pick a node that lies in small finite size component, and not in the LCC.[9]

Let us revisit the subcritical region, where $z < 2$, but now for the general random graph in the large sparse graph limit. As ℓ increases the number of vertices added to collection at each step is now $n_1(z-1)^{\ell-1} \to 0$ which is decreasing as ℓ increases. This means the total number of vertices connected to vertex i, the size of the component containing vertex i, is on average a finite number, S_i where

$$S_i = 1 + \sum_{\ell=1}^{\infty} k_i (z-1)^{\ell-1} = 1 + \frac{k_i}{1-(z-1)} \tag{5.16}$$

$$= 1 + \frac{k_i}{2-z} \tag{5.17}$$

As this is a finite number if $z < 2$ we have that in this subcritical region, all components contain a zero fraction of the infinite total number of vertices N, that is $\lim_{N\to\infty}(S_i/N) = 0$. In the same way the average path length ℓ_i from a vertex i to any other vertex to which it is connected, i.e. to other vertices in the same component, are finite as we can see from

$$\ell_i = \frac{1}{(S_i - 1)} \sum_{\ell=1}^{\infty} k_i \ell (z-1)^{\ell-1} \tag{5.18}$$

$$= \frac{1}{(2-z)} \,. \tag{5.19}$$

Note that both the average size of the component started from vertex i, S_i, and the average distance within this component ℓ_i diverge as we approach $z = 2$, further confirmation that $z = 2$ is indeed a critical point in this model.

In the super critical region $z = \langle k^2 \rangle / \langle k \rangle > 2$ for our large sparse graph (i.e. where $N \to \infty$ $\langle k^2 \rangle, \langle k \rangle \sim O(1)$). If we think of keeping N very large but finite, taking $N \to \infty$ at end, we see that the number of vertices at distance ℓ from initial vertex i is growing exponentially but that this growth must stop when $n_1(z-1)^{\ell-1} \approx N$. This estimate shows that the largest distance from initial vertex core is going to scale as

$$\ell_{\max} \approx \frac{1}{\ln(z-1)} \ln(N) \tag{5.20}$$

Again this means that in random graphs a large fraction of the vertices are connected[10] and on relatively short length path scales, that is ℓ scales as $\ln(N)$ not as it would in a d-dimensional space(such as a regular lattice) where we would expect scaling as $N^{1/d} \gg \ln N$. So again, these generalised random graphs, here constructed via the configuration model, are Small World graphs.

[9] Of course if there are no nodes with degree zero or one then all nodes have degree two or more so we have $2 \leq \langle k \rangle \leq \frac{\langle k^2 \rangle}{\langle k \rangle} = z$. So a random graph of this type would expect to have a percolating cluster as $z \geq 2$.

[10] Technically the LCC is a good approximation to a GCC though the latter is only strictly defined in theoretical models in the large N limit.

A more sophisticated estimate is that if we start from a vertex with degree k then

$$N \approx \sum_{\ell=0}^{\ell_{\max}} n_\ell \tag{5.21}$$

$$\Rightarrow \quad N \approx 1 + k\frac{(z-1)^{\ell_{\max}} - 1}{z-2} \tag{5.22}$$

Rearranging this gives us that above the critical point we have that

$$\ell_{\max}(k) \approx \frac{\ln(1 + N(z-2)/k)}{\ln(z-1)}\,. \tag{5.23}$$

Chapter 6

Network Measures without Eigenvalue Calculations

We have encountered several useful measures of properties of nodes or of the network overall: the degree of each vertex and the overall degree distribution, the length of the shortest path between a pair of vertices, the average shortest path length of a network $\langle \ell \rangle$, the diameter D of a network , and the number and size of network components. In this chapter we will look at several more measures used to probe the nature of a network. The measures we consider in this chapter are not based on linear algebra, the use of eigenvectors to produce useful measures will be part of the following chapter.

6.1 Shortest Path Based Centrality Measures

Considered a network embedded in space, say the connections between different cities. A city in the middle of this space will never be a long way from any other city and because of this such a city could well become an important hub. In such a geographical context we know that it is cities in the centre of our space which have this property.

However we are interested in networks in general where there is no additional ruler and space telling us how to find a central node. However we can still find ways to use the network connections alone to find sites which are important to the network, and, by analogy, we can think of these are being more 'central' in the network. To do this we find measures of the importance of a node, and these are known in the network literature as a centrality measures[1].

Put another way, in our most basic definition of a network or graph, we have a set of vertices but there is nothing else to distinguish them. However, the connections between edges and then the wider structure of the network beyond leads to some vertices playing a more important role than others. It is this sense of importance of a node in a network that centrality measures try to capture.

We have already considered the simplest example of a pure network derived measure of the importance of a node, the degree k. The degree of a vertex is one of the simplest and yet one of the most useful centrality measures. A large degree in a social network can indicate someone who is very popular and this may well indicate that they are a very important part of the social structure. Someone with few friends is likely to be peripheral to social interactions.

An key aspect of degree measurements is that they are the most local measure of network properties, that is the degree is only sensitive to the existence of nearest neighbours of a vertex, it does not care about the large scale 'global' structure of the network. In this chapter we will look at some examples of centrality measures which probe the large scale structure. Here we will do this without using linear algebra. In the next chapter we will look at centrality based on eigenvectors but that discussion will also lead us beyond centrality measures.

We have already encountered the definition of a path between two vertices in Section 2.2. The concept of the length of the shortest path between a pair of vertices leads to several different network centrality measures and we will consider the most popular here.

[1] A very nice example of centrality measures is in Brandes & Hildenbrand [Brandes and Hildenbrand, 2014].

One important feature of all such shortest path based centrality measurements based on path is that, unlike the degree, these path based measures probe the whole structure of the network, they are sensitive to the global properties of a network.

6.1.1 (#) Closeness

A useful way to measure the distance from vertex i to vertex j is to use d_{ij}, the length of the shortest path from j to i. Given this, we can produce a measure of the similarity or CLOSENESS of two vertices i and j by taking the inverse of this distance, $c_{ij} = 1/d_{ij}$, since the closer two vertices are the larger the closeeness[2].

In particular we find the average distance from a node i to all others connected to it, the FARNESS. A node which is close to many vertices will have a low average distance, will be close to many. If we conjecture that an important node will be close to many vertices (low farness), an unimportant vertex far away (high farness) then for a centrality measure we want to reverse this order, produce a measure based on farness which is large (small) for important (unimportant) vertices. A simple way to do this is to take the inverse of the farness and this is the definition of the centrality measure known as CLOSENESS, first given by Bavelas [Bavelas, 1950]. Thus we define the CLOSENESS of a vertex i, c_i, to be

$$c_i = \frac{(n_i - 1)}{\sum_{j \in \mathcal{V}_i} d_{ij}} \tag{6.1}$$

where the sum is over all[3] n_i vertices j connected to i (the set denoted $\mathcal{V}_i$ so $n_i = |\mathcal{V}_i|$). The idea behind this centrality measure is that the vertex with the highest closeness is the one most similar on average to all other vertices. Since it uses distances between all vertices, closeness probes the large scale global structure of the network.

Note that if the network is directed the distances of shortest paths starting from j and finishing at i, d_{ij}, is not generally the same as found with shortest paths in the reverse direction, starting from i and finishing at j. So for directed networks there are two types of closeness: c_i^{in} and c_i^{out} with the latter using d_{ji} not d_{ij} in the denominator.

Another issue with this definition of closeness is when there is more than one component of a network. Then there are pairs of vertices for which there is no path linking these vertices. This means that d_{ij} is formally undefined for disconnected pairs, usually considered to be infinite, that is $d_{ij} = \infty$ if there is no path from j to i. Our version of closeness for vertex i in (6.1) has avoided this problem by limiting the sum in the formula for closeness to run over nodes in the same component as the vertex i under consideration. However in many cases the definition is given in terms of the connections to all nodes in a network with

$$c_i' = \frac{(N - 1)}{\sum_{j \in \mathcal{V}} d_{ij}} \tag{6.2}$$

with $N = \mathcal{V}$ as usual. For fully connected networks, there is no difference between (6.1) and (6.2) but when there are separate components, formally you have to assign a zero distance between to disconnected node pairs. This seems counter intuitive and can produce some strage effects. It is always worth checking the definition of closeness being used if your network has separate components.

One solution to the issue of disconnected components is known as HARMONIC CLOSENESS. In this case we stick with the logical assignment of an infinite distance between disconnected pairs of nodes, $d_{ij} = \infty$ if there is no path from j to i. This would force the classic closeness of (6.2) to be zero if there was more than one component. However rather than working with the inverse of the average distance between nodes, we work with the average of the inverse of individual node pair separations.

[2]There are many possible variations. There is no particular reason why we have to use $1/d_{ij}$ as a measure of closeness between two vertices, any monotonically decreasing function of distance other than a simple inverse might be just as effective but equally, there is unlikely to be much difference overall results of any analysis for any reasonable choice for the closeness function.

[3]We note that vertex i does not contribute to this sum as it is distance 0 away from itself, $d(i, i) = 0$. That is why one traditional normalisation is to use the number of vertices to which j is confected to other than itself, i.e. we use a factor of $(n_i - 1) = |\mathcal{V}_i| - 1$.

This is known as HARMONIC CLOSENESS $c_i^{(h)}$ of a vertex i and formally it is given by

$$c_i^{(h)} = \frac{1}{N-1} \sum_{j \in \mathcal{V} \setminus i} \frac{1}{d_{ij}} \,. \tag{6.3}$$

Note that we do include the vertex i in the sum in the denominator, but we can work over all other nodes, as indicated by the notation $\mathcal{V} \setminus i$. There are variations in the normalisation used in the expressed for harmonic closeness, with both $(N-1)^{-1}$ and $(n_i - 1)^{-1}$ used so again care needs to be taken. One drawback for harmonic closeness is that largest contribution comes from the closest nodes so closeness is often controlled by the size of the local neighbourhood of nodes. The more nearest neighbours and next-to-nearest neighbours a node has, the larger the closeness. As degree measures the number of the most immediate neighbours then a high degree node will be guaranteed a large contribution to its closeness values from it neighbours and the two measures are often correlated.

Closeness in a Random Network

Not surprisingly, if you start from a high degree node, a high k your first step will reveal far more of the network and so take you closer to the remaining parts. Thus the maximum distance in a random graph does depend on the initial degree. This idea of how close one vertex is to another vertex is the basis of a centrality measure, closeness (see Section 6.1.1). The closeness c_i of a vertex i is defined to be the inverse of farness, f_i, which in turn is the average distance from a node to all other nodes.

$$c_i = \frac{N-1}{\sum_j \ell_{ij}} \tag{6.4}$$

For the random graphs we can estimate this quantity as for a node i (writing $\ell_i = \ell_{\max}(k_i)$for simplicity)

$$f_i = \frac{1}{(N-1)} \sum_{\ell=1}^{\ell_i} \ell n_\ell \approx \frac{1}{(N-1)} \sum_{\ell=0}^{\ell_i} \ell k (z-1)^{\ell-1} \tag{6.5}$$

$$= \frac{1}{(N-1)} k \frac{d}{dz} \sum_{\ell=0}^{\ell_i} (z-1)^\ell = \frac{1}{(N-1)} k \frac{d}{dz} \frac{(z-1)^{\ell_i+1} - 1}{z-2} \tag{6.6}$$

After some further manipulations we find

$$f_i = \ell_{\max}(k_i) - \frac{1}{(z-2)} + \frac{k}{(N-1)} \frac{(\ell_{\max}(k_i) - 1)}{z-2} \tag{6.7}$$

where we have used (5.22). Not surprisingly this is dominated by the distance to the further nodes as in the tree they are the dominant contribution. From this we see that if $(z-2) \gg k/N$, i.e. if we are not close to the transition and we have a large N, then we might estimate farness as

$$f_i \approx \frac{\ln(N(z-2)/k)}{\ln(z-1)} \,. \tag{6.8}$$

While in this limit a random graph let alone a real graph is not a tree, it serves as a way of showing that we should expect the closeness centrality measure to be correlated with the degree of a node. Indeed the prediction is that the inverse closeness (farness) should show a linear dependence on the logarithm of the degree of a node, $\ln(k)$, with a slope that is the inverse of the log of the branching ration minus one, $1/\ln(z-1)$. That is

$$\frac{1}{c_i} = f_i = -\frac{1}{\ln(z-1)} \ln(k) + b \tag{6.9}$$

Since this expression is true where we do not have a tree, we do not expect the slope to match a value of $z = \langle k^2 \rangle / \langle k \rangle$. Rather, if we do find a linear relationship for the farness and logarithm of degree, then the slope is a way of defining an effective branching ratio.

6.1.2 Betweenness

Another standard centrality measure based on shortest path distances is the BETWEENNESS OF A VERTEX (VERTEX BETWEENNESS) which is the number of shortest paths passing through a vertex where we consider paths between all connected vertex pairs.[4] See Figure 6.1 and Figure 6.3. It is common to find that there is more than one shortest path between a pair of vertices, for instance between vertices E and H in Figure 6.1. In this case if the number of shortest paths between vertices i and j is $n_{\rm sp}(i,j)$ we weight each of these shortest paths by $1/n_{\rm sp}(i,j)$. More formally, if we write that the set of shortest paths between source node s and target node t is $\mathcal{P}_{\rm sp}(s,t)$, then we may write the vertex betweenness $b(i)$ of any node i as

$$b(i) \;=\; \frac{1}{Z_b}\sum_{s,t\in\mathcal{V}}\;\sum_{P\in\mathcal{P}_{\rm sp}(s,t)}\;\sum_{j\in P_b\subseteq P}\frac{1}{|\mathcal{P}_{\rm sp}(s,t)|}\delta_{i,j}\,, \tag{6.10}$$

where Z_b is some suitable constant normalisation. A common limitation is to exclude the end points of each path, so that each path P_b is defined to be the shortest path from s to t with vertices s and t excluded (a "subpath", possibly the empty set), so $P_b = P\backslash\{s,t\}$. If we did include the endpoints then for a fully connected graph we would get a constant factor of N^2 added to the score of each vertex from the cases where vertex i is equal to s or t in the sum over vertex pairs. Such a trivial constant is of little help hence we adapt the formula. Likewise, if the graph is undirected the normalisation factor $1/Z_b$ is often taken to be $1/2$ as the shortest paths from s to t will appear a second time with source and target reversed but contributing exactly the same factor.

It is clear that the mathematical expression of betweenness (6.11), with all the variations and caveats, is not particularly illuminating. This reflects its origin as a natural way to think about the flow of information in small social networks. It is much easier to understand when applying in practice or as a simple computational algorithm. The awkward nature of the formula (6.11) is also reflected in the difficulty in scaling up computational algorithms of betweenness to deal with large networks.

On top of computational issues, it is not always clear if information flow in a network is always along shortest paths. That might be the quickest route but don't we always know the quickest way to pass on information. In small networks where the concept was originally developed perhaps, in larger cases, it seems unlikable. Further, all pairs of nodes are treated equally in the standard form of betweenness (6.11). In many networks, not all nodes are equal in terms of producing or receiving information. If that is known, one could add some weighting to the sum of pairs $\sum_{s,t}$ in the formulas for betweenness (6.11). Still the standard form for betweenness represents at least one way to capture approximately, if not exactly, how nodes might be participating as information is transported around a network. It is a useful place to start and

We can also use the shortest paths as defined above for vertex betweenness to define EDGE BETWEENNESS. Simply we repeat the sum over all shortest between all pairs of vertices, weighting the paths as before, counting how often these shortest paths contain a given edge. More formally the vertex betweenness $b(e)$ of any edge e is

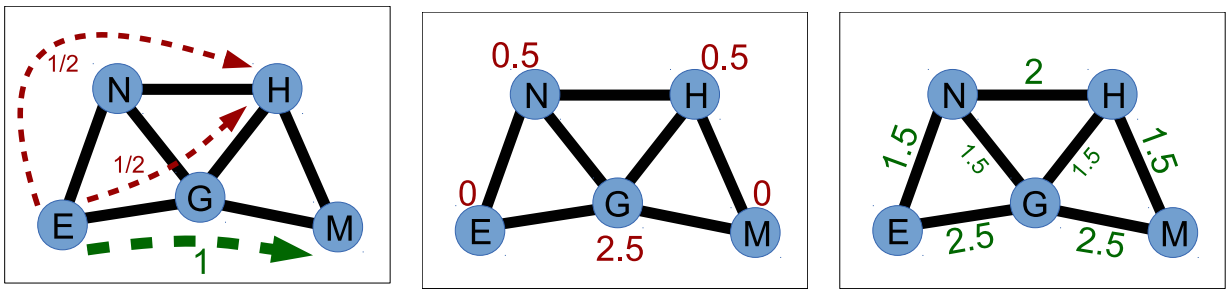
$$b(e) \;=\; \frac{1}{Z_b}\sum_{s,t\in\mathcal{V}}\;\sum_{P\in\mathcal{P}_{\rm sp}(s,t)}\;\sum_{e\in P}\frac{1}{|\mathcal{P}_{\rm sp}(s,t)|}\,. \tag{6.11}$$

Note we no longer need to exclude the endpoints of the shortest paths as we are only interested in the edges in the shortest paths.

6.2 Edge Percolation

One way to see why betweenness might tell us something useful is to draw on the work of Granovetter (see Figure 6.2) in his famous paper entitled "The Strength of Weak Ties" (1973) [Granovetter, 1973]. There Granovetter suggested that in some situations some of the most important links are those used only occasionally. For instance most people work alongside a relatively small number of colleagues

[4] In passing we note that any centrality measure for a vertex has an equivalent form for edges. It was first introduced by Freeman [Freeman, 1977]. For instance the BETWEENNESS OF AN EDGE, EDGE BETWEENNESS, is the number of shortest paths passing through an edge, where we consider paths between all connected vertex pairs.



(a) (b) (c)

Figure 6.1: These visualisations show the different betweenness calculations for a simple network of connections between biographies of five mathematicians, N=Newton, G=Galileo, M=Mersenne, H=Huygens, E=Euclid. In (a) we show some of the shortest paths used in the calculations. The heavy green dashed line shows that unique shortest path between vertices E and M. This will contribute $+1$ to the vertex betweenness of vertex G and $+1$ to the edge betweenness of edges (E, G) and (G, M). However there are two shortest paths between vertices E and H. So these paths each contribute $+1/2$ to the vertex betweenness of vertices lying on each path, here vertices G and N, and similarly adding $+1/2$ to the edge betweenness of edges lying on these paths: (E, G), (G, H), (E, N) and (N, H). Diagram (b) shows the vertex betweenness and diagram (c) shows the edge betweenness values for this network.

Figure 6.2: Mark Granovetter (1943–)

with whom they interact regularly. These will be their strong ties, relationships which would have a natural representation as a strongly weighted edge in a weighted graph. You would probably know these colleagues well and this group are likely to be thinking about similar ideas and tackling issues in a similar way. However there will also be people you meet occasionally, perhaps people for other parts of the same company you meet by chance, at the 'water cooler' perhaps. They are also people you know but who are working in similar areas but for different companies. Such people you meet much less often, perhaps at an unrelated a social event or you bump into them in a cafe. These are the 'weak ties' as it would be natural to repent these relationships as edges of low weights in a friendship network. However these interactions are far more likely to provide new information, novel ideas, as they represent interactions which can be very different from your regular contacts, they are 'outside the box' of everyday experience. The idea is that new information or inspiration which can be vital the long term health of a company, is more likely to come from these less frequent links, and these are the 'weak ties' of Granovetter's title because the frequency of these bilateral meeting for the two parties in the tie (edge) is a proxy for a weaker link.

We can interpret this idea in a more quantitative manner using EDGE PERCOLATION. Suppose we have a weighted graph[5]. This could represent a friendship structure where the weight of an edge

[5]If we had a simple unweighted graph, such as a friendship graph where there is no information on the strength of the tie, we can find network methods which will try to assign an appropriate edge weight. Essentially this would be trying to predict the strength of the friendship. An obvious candidate here (if a slow one computationally) is to use edge betweenness,edge!betweenness the number of shortest paths passing through each edge.

connecting two actors (the nodes) represents the amount of time those two actors interact. In an edge percolation process we start with no edges, just the nodes of the graph. We then add the edges one by one starting with the strongest edges, the strongest personal relationships in our friendship network example, first. We would then look to see when the GCC forms — a finite fraction of the nodes become connected (if this is a theoretical model with a formal $N \to \infty$ limit). For a finite sized graph, we would typically monitor the size of the LCC looking to see when or if it grows very quickly in this process. The phase transitions we considered for random graphs can be considered as special cases of edge percolation in which the weight of the edges are drawn uniformly from a the range 0 to 1.

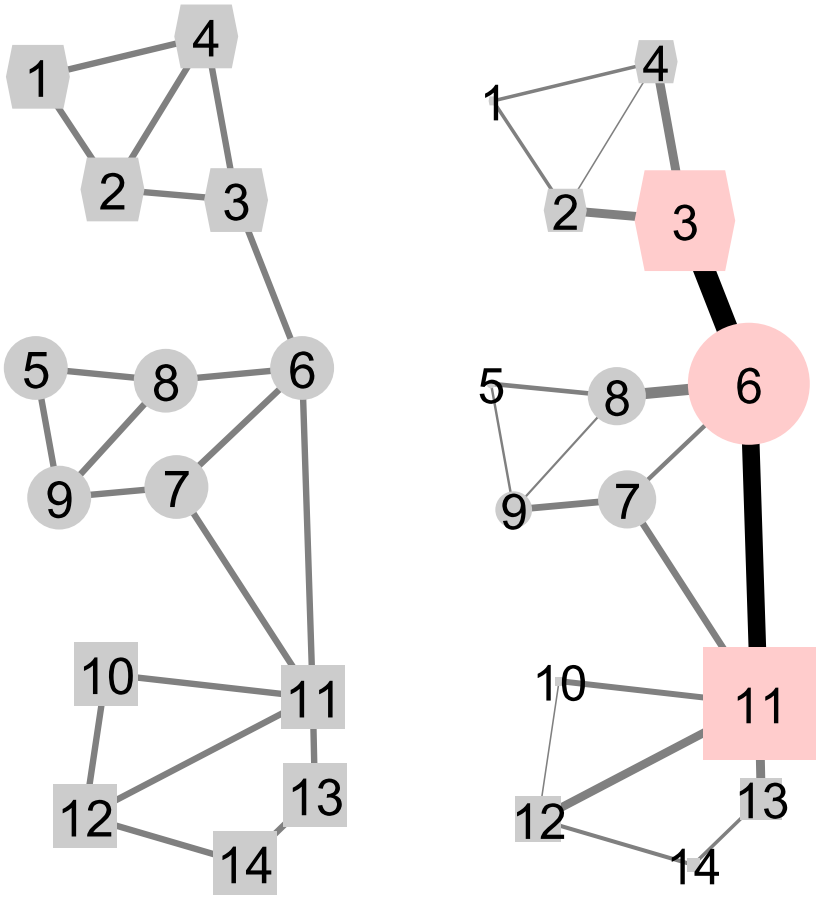


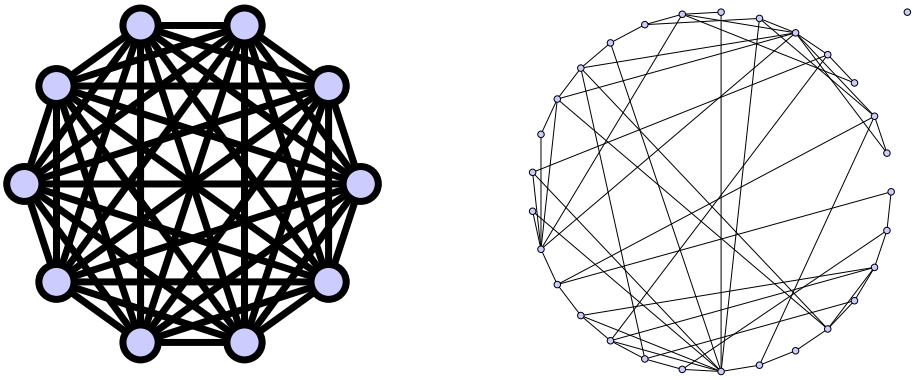
Figure 6.3: Which nodes and edges have the highest betweenness for the graph on the left? The answer is illustrated on the right where now the larger the node, the wider the edge, the higher the betweenness.

6.3 Clustering and the Small World model

A rather different type of measure is the CLUSTERING COEFFICIENT. For a friendship network the clustering coefficient tries to capture the idea that any friends of mine are likely to be friends of each other. To do this we need not to look at the edges between neighbours of a vertex so the clustering coefficient is requires us to look a step further away from a given vertex than the measurement of its degree requires.

The VERTEX CLUSTERING COEFFICIENT c_i of a vertex i is the fraction of complete triangles centred on i given number of possible triangles with nearest neighbours.

$$c_i = \frac{2}{k_i(k_i-1)} \sum_{j<k} A_{ij}A_{ik}A_{jk} \text{ (if simple)} \tag{6.12}$$



Complete graph, $N = 10$
"All my friends are friends"

ER random graph, $N = 30$, $\langle k \rangle = 4.0$
"My friends don't know each other"

Figure 6.4: A complete graph and an ER random graph.

The vertex clustering coefficient gives 1 if all possible triangles are present, 0 if there are no triangles However note that this definition (6.12) is undefined if degree less than 2 as then there are no triangles possible.

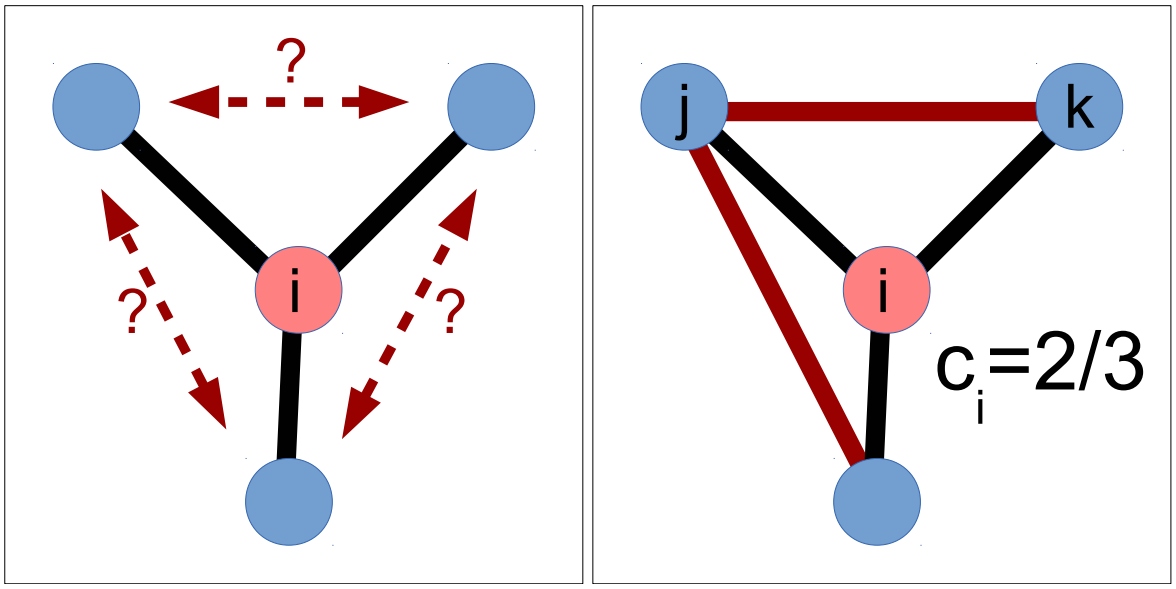


Figure 6.5: Example illustrating the clustering coefficient.

For a network, it is useful to get a feel for the overall clustering of a network. The NETWORK CLUSTERING COEFFICIENT[6] c of a network is the average is the number of shortest paths passing through an edge, where consider paths between all connected vertex pairs

$$c = \frac{1}{N} \sum_i c_i \tag{6.13}$$

This is not defined if c_i poorly defined.

However there is a second definition of a network clustering coefficient, $c^{(II)}$, as the fraction of

[6]This is also called the GLOBAL CLUSTERING COEFFICIENT or TRANSITIVITY.

'wedges' (two edges with a common vertex) which are part of triangles.

$$c^{(II)} = \frac{3(\text{number of triangles})}{(\text{number of wedges})} . \tag{6.14}$$

This is always well defined and typically fairly similar to the first definition of a network clustering coefficient.

Some examples of standard networks and their clustering coefficient values

- A complete graph $c = c_i = 1$
- A triangular lattice: each site has 6 triangles, but $15(= 6 \times 5/2)$ pairs of edges, so $c = c_i = 0.4$
- ER Random graph $c = \langle c \rangle = p = \langle k \rangle / (N-1)$
- Square lattice $c = c_i = 0$: want a square count measure here

The definition should give 1 if all possible triangles are present (e.g. in a complete graph) and 0 if there are no triangles (e.g. square lattice).

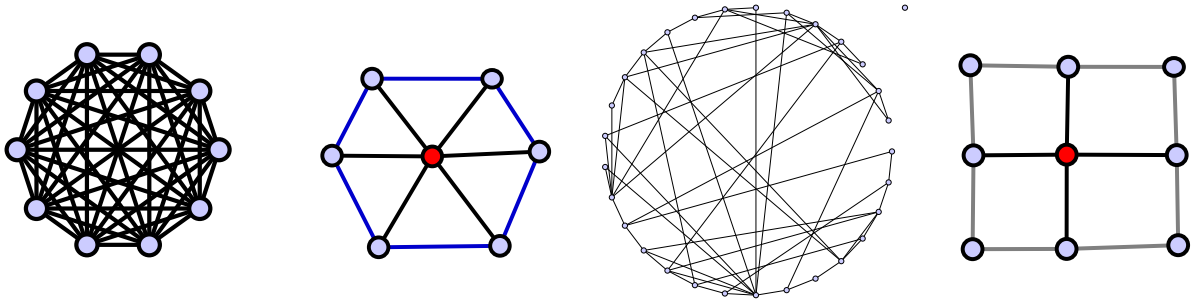


Figure 6.6: The first graph is a complete graph, the second is a triangular lattice, the third figure is an ER random graph, $N = 30$, $\langle k \rangle = 4.0$ and lastly we have a square lattice.

Clearly the square lattice of Fig. 6.6 is highly clustered in terms of squares. However the clustering measures we have defined, which are based on triangles, are zero for the square lattice showing that these cluster measures are not always the full story for a few networks with particular types of structure.

6.3.1 A Null Model for Social Networks

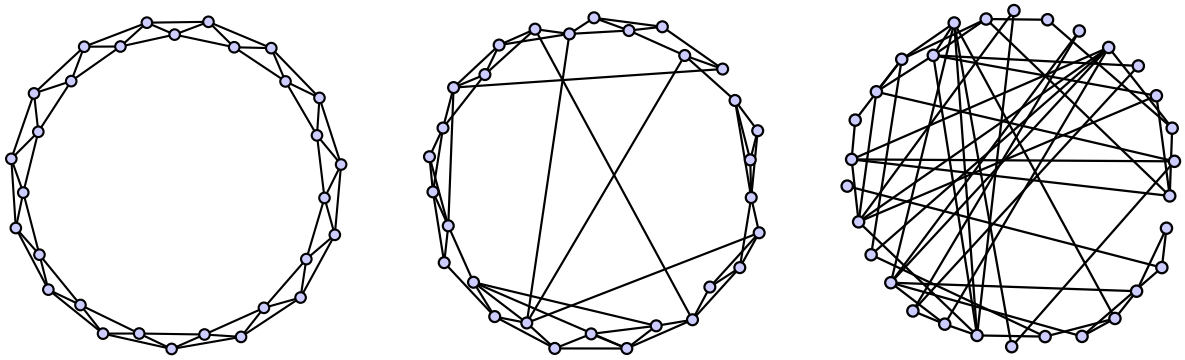
Social networks are highly clustered, perhaps $c \sim 0.1$ is a rough guide, and they have short path lengths as $\ell \sim \ln(N)$. For instance the hep-th citation network has $c = 0.25$. So what sort of model is a reasonable starting point for such networks?

ER Random networks have little clustering, $c \sim O(1/N)$, and short path lengths $\ell \sim \ln(N)$. At the other extreme regular lattices have lots of local structure and so high clustering, e.g. $c = 0.4$ for the triangular lattice, but long path lengths, $\ell \sim N^{1/d}$ for a d-dimensional lattice. So these examples capture one but not both of the clustering and path length features we often find in real social networks. This leads us to ask what would be a good null model for a social network? One answer was provided by Watts & Strogatz in 1998 [Watts and Strogatz, 1998] with their Small World model.

The process considered by Watts and Strogatz is as follows:

- Start with a regular lattice ($c \sim 0.1$, $\ell \sim N^{1/d}$)
- Rewire t edges, that is pick an edge and move one end from one vertex to another.

It does not matter how you pick the edges, choosing uniformly at random from the set of edges is an obvious choice, provided that after many rewirings we have rewired most of the edges. In that case we will approach a random graph ($c \sim O(1/N)$, $\ell \sim \ln(N)$) as the large t limit of this process. This is shown in Fig. 6.7.



$t = 0$
$c = 0.5$, $\ell = 4.1$

$t = 9$
$c = 0.34$, $\ell = 3.0$

$t = 60$
$c = 0.15$, $\ell = 2.7$

Figure 6.7: The rewiring process used to construct the Watts and Strogatz Small World model shown for a $N = 30$ graph. The initial graph, $t = 0$, is a one-dimensional ring where every node is connected to the next two nodes around the ring on both sides. That gives us $\langle k \rangle = 4.0$ so $E = 60$ edges.

The central idea in this model is that we start with a regular lattice. These have a lot of local structure, usually seen as a high clustering coefficient, which is a characteristic of most social networks. However the distance scales such as the diameter or average shortest path length ℓ are large, scaling as some power of the number of nodes for analytical models, e.g. for regular lattices in d dimensions we will have $\ell \sim O(N^\alpha)$ with $\alpha = 1/d > 0$. This is not the "six-degrees of separation" small world behaviour seen in social networks where distance measures scale as the logarithm of the number of vertices, so $\ell \sim O(\ln(N))$ etc. One the other hand after many rewirings, large t, we end up at a random graph which we know has the opposite properties. Random graphs have the small world $\ell \sim O(\ln(N))$ behaviour but no local structure. For instance the probability that two neighbours of a vertex are connected is random and completely independent of the common neighbour they share. So the clustering coefficient is $1/p \to 0$ for an ER random graph with fixed $\langle k \rangle = pN$ as $N \to \infty$. The interesting feature of the Watts & Strogatz model is that there is an intermediate stage for a small number of rewirings, $0 \ll t \ll O(E)$ where the resulting networks have high clustering and low path lengths as seen in Figure 6.8, precisely the sort of basic properties expected in realistic social networks. In this sense this is a far better model of many social networks than either random graphs or regular graphs.

6.4 Summary

Network Measures without Eigenvalue Calculations

We have

- recalled that
  - degree k_i and degree distributions $p(k)$ are useful measures
  - paths are used to define connectedness and components
  - shortest paths ℓ_{ij} are used to define average shortest path length $\langle \ell \rangle$, diameter
- defined vertex and edge betweenness using shortest paths
- discussed Granovetter's idea of the strength of weak ties
- defined the clustering coefficient for vertices, and (in two ways) for networks
- discussed Watts & Strogatz' Small World model.

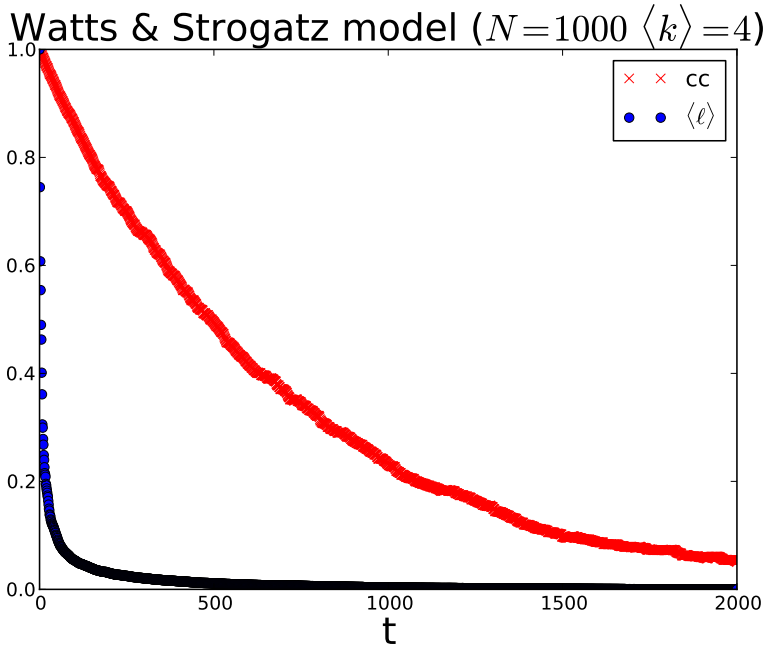


Figure 6.8: Plot of length scale ℓ (blue dots) and clustering coefficient c (red crosses) in the Watts and Strogatz model in terms of the number of rewirings t starting with a regular one-dimensional ring with $N = 1000$, $\langle k \rangle = 4.0$, $E = 2000$. Note that both ℓ and c have been rescaled to be 1.0 at $t = 0$, and 0.0 at $t = \infty$. Clearly in the region around 100 rewirings we have small world network characteristics, a random graph low path length but a high lattice type clustering.

Chapter 7

Processes on Networks

In this chapter we focus on processes happening *on* a network. After all networks are not an end in themselves, they are invariably just the substrate for other dynamical processes. For instance friends spreading rumours on a social network, people follow the network of hyperlinks when they navigate the web. The behaviour of a process on a network always reflects the properties of the network. This in turn means that we can also use these processes as to probe the network structure. We will motivate our work here by using different processes on a network to produce new measures of the importance or centrality of a node. However we should not lose sight that these ideas can be used for many other purposes, both in the study of other network fetaures and because processes on a network are of interest in their own right.

7.1 Local Processes and Eigenvalue Based Centrality Measures

We have come across some of the standard ways to assess the importance or, as it is called in social network analysis, the CENTRALITY of a vertex. Let us summarise those encountered earlier and highlight some of their drawbacks.

Degree. The more popular you, are the more important you are.
A reasonable starting point but this is a measure of the local environment of a vertex. The degree has no information about links beyond nearest neighbours, so it is not sensitive to the overall global structure of the network.
Example: perhaps you are popular in a small but remote community so the distance from you to most members of the network is so large so that the rest of the world barely knows you exist.

Closeness. The more people you have who are close to you, the more central you are. (Section 6.1.1)
This measure does probe the whole network structure to some extent but it is dominated by the vertices in the close neighbourhood of each vertex.

Betweenness. The more shortest paths passing through you, the more important you are. (Section 6.1.2)
This measure does probe the whole network structure but it is rare for shortest paths to be the only paths important for information transfer. Finding the shortest path is not trivial, it can take a long time to find them. So alternative good but still sub-optimal routes are often used in practice. Routes may not always be available as they become too popular and congested. In any case measuring distance in terms of the number of edges traversed is not always the right measure of the cost of a route.

It may be possible to know the optimal routes in a small social network where everyone knows everyone else. Indeed, small networks were the original context of social network analysis from which ideas such as betweenness emerged. Betweenness sounds very reasonable in that context.

However, today we can analyse much larger data sets. Real people in large networks often do not know the shortest paths between everyone in the network so we may well ask if betweenness the best measure of the importance of a person in a large social network. We need to find other measures of

importance, measures which tell us something about the wider structure of the network by probing all the links and routes in the network. One way to do this is by considering different types of processes happening on a networks. If these processes probe the network along paths which are more realistic than just the set of shortest paths, then maybe these processes can be used to produce more 'natural' network measures.

The processes we consider here are still very simple but they may in some circumstances be a better reflection of how we use networks, in cases where the processes on the network do not know the shortest paths. Even if our models of these processes are not very realistic, they may still provide a useful null model. The success of PageRank when rating web pages does though suggest the simple processes we discuss in this chapter can capture useful information about the way some networks are used.

In this chapter our focus is on LOCAL PROCESSES on a network, see Figure 7.1 for an illustration. Think of an agent at each vertex and all each agents knows is who their neighbours are but nothing beyond that. The agents do not see a bigger picture: they can not see what the shortest paths are, they do not know most of the nodes in the network, they do not even know how big the network is. This is often a more realistic picture especially when agents are part of a large network. At each time step in the process, the agent can decide to pass on some token, or not, to some or all of its neighbours according to some simple rules based but *only* on the *local* information available to each agent. These tokens can represent information or the presence of an actual information or just a way to probe the network. In the context of centrality we are using here, it is helpful to think of these tokens as votes for popularity. The more tokens or votes a node receives, the more important it is.

The opposite to our local processes are processes based on *global* knowledge. For instance choosing a vertex from the set of all vertices (in any way) requires knowledge of the whole network to normalise the probabilities. The simplest implementations of the various random graph models and the Price/BA model use global information. In practice global knowledge might represent information passed by a newspaper or website too all the actors in the network so it can be an important factor. However, the focus of this chapter is not on such global processes.

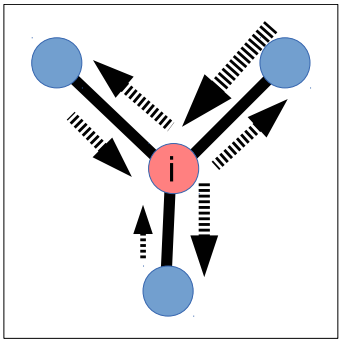


Figure 7.1: An illustration of a local Processes.

The processes we consider in this chapter are the simplest type of process. In each case we imagine that votes (or other token) are moving from vertex to vertex along edges of the network. An key assumption we make here is that each vote does not interact with any of the other votes. Each one moves around without noticing what other votes or tokens are doing. It follows that it does not matter if there is one token or millions. Mathematically we are saying that we may simply add the effects of very token or vote to find the overall picture, we have a *linear process*. We are therefore able to use the tools of *linear* algebra — vectors, matrices, eigenvalues and eigenvectors. The centrality measures derived from the processes discussed in this chapter are all based on eigenvalues and eigenvectors of various matrices, while the centrality measures of Section 6.1 are not.

Of course we can relax these linearity conditions and look at more complicated cases, non-linear versions of the processes below. In reality, the actual processes may well be more complicated than the linear processes considered here. However it does appear that in many cases the simplified processes examined in this chapter are good enough to capture the statistical properties on large scales in many

cases, suggesting that linear processes often capture key aspects of the system of interest.

Within these local linear processes there are two main types of process we can consider: BROADCAST or DIFFUSION. These represent two opposite ends of the spectrum. Before we study such processes, we first need to visit some basic properties of matrices.

7.2 Properties of Matrices

7.2.1 Eigenvector properties

It is useful to summarise some of the key properties of matrices which we will need. First we must remember that there are two types of eigenvector equation. The first defines the n-th RIGHT-EIGENVECTOR $\mathbf{v}^{(n)}$ with entries $v_i^{(n)}$ where $i = 1, 2, \ldots, N$ for an N-dimensional matrix M

$$\lambda_n v_i^{(n)} = \sum_j M_{ij} v_j^{(n)} \,, \qquad \lambda_n \mathbf{v}^{(n)} = \mathsf{M}\mathbf{v}^{(n)} \,. \tag{7.1}$$

In the same way we may define a LEFT-EIGENVECTOR equation

$$\lambda_n u_j^{(n)} = \sum_i u_i^{(n)} M_{ij} \,, \qquad \lambda_n (\mathbf{u}^{(n)})^\mathsf{T} = (\mathbf{u}^{(n)})^\mathsf{T} \mathsf{M} \,. \tag{7.2}$$

These left- and right-eigenvectors and their eigenvalues obey some useful properties

1. The set of left- and right-eigenvalues are identical.
   We might start with distinct eigenvalues, say $\lambda_n^{(R)}$ in (7.1) and $\lambda_n^{(L)}$ in (7.2). It is straightforward to show that these must be identical (consider $\mathbf{u}^{(n)}\mathsf{M}\mathbf{v}^{(m)}$) justifying our notation that $\lambda_n = \lambda_n^{(R)} = \lambda_n^{(L)}$

2. The left- and right-eigenvectors are not necessarily the same.
   If they can be chosen to be the same it indicates that the matrix M has additional properties, namely that it is hermitian and so, if the entries are real, M is then a symmetric matrix.

3. The left- and right-eigenvectors can be chosen to be orthogonal.

$$(\mathbf{u}^{(n)})^\mathsf{T}.\mathbf{v}^{(m)} = \delta_{nm} \,, \qquad \sum_i u_i^{(n)} u_j^{(m)} = \delta_{nm} \,. \tag{7.3}$$

   Here we will always assume this has been done.

Note that when the matrix is real and symmetric then we have additional properties but many of the matrices we encounter will not be symmetric even for simple networks (whose adjacency matrix is real and symmetric) so we will not use such special properties.

7.2.2 Diagonalisation

We will assume that our eigenvectors (both the left and right sets independently) span the vector space. That is a general N-dimensional vector can be represented as a linear sum of eigenvectors and each eigenvector is linearly independent, i.e. no eigenvector can be expressed as a sum of the other eigenvectors. This means that

$$\mathsf{U}^\mathsf{T}.\mathsf{M}\mathsf{V} = \mathbf{\Lambda} \,, \qquad \mathsf{M} = (\mathsf{U}^\mathsf{T})^{-1} \mathbf{\Lambda} \mathsf{V}^{-1} \,, \tag{7.4}$$

where

- The i-th row of U^T is the i-th left-eigenvector $(\mathbf{u}^{(i)})^\mathsf{T}$, $U_{ji} = u_j^{(i)}$.
- The j-th column of V is the j-th right-eigenvector $\mathbf{v}^{(j)}$, $V_{ij} = v_i^{(j)}$.
- $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalue λ_i as the i-the entry, $\Lambda_{ij} = \lambda_i \delta_{ij}$.

Orthogonality of the left-/right-eigenvectors means

$$\mathsf{U}^{\mathsf{T}}.\mathsf{V} = \mathbb{1}\,, \qquad (\mathsf{V})^{-1} = \mathsf{U}^{\mathsf{T}}\,, \qquad \mathsf{M} = \mathsf{V}\mathbf{\Lambda}\mathsf{V}^{-1}\,. \tag{7.5}$$

In some cases the eigenvectors (both the left and right sets independently) may not span the vector space, we may have less than N linearly independent eigenvectors. In these situations we can still follow the programme set out below but it takes a lot more effort[1]

7.2.3 Properties of Non-Negative Matrices

Many of the matrices we will consider, such as the adjacency matrix of most networks, are NON-NEGATIVE[2], that is all the entries of the matrix are positive or zero, $M_{ij} \geq 0$.

We will also assume that the adjacency matrix here is for a network that has a single strongly connected component, that is there is a path from every vertex to every other vertex. Should the network have several components, we can apply this analysis to each part in turn. This means that the matrix is IRREDUCIBLE.

With these reasonable constraints, we are able exploit a powerful theorem, the PERRON-FROBENIUS THEOREM. This gives us information on a real $N \times N$ non-negative irreducible matrix M. So first let us look at these requirements. A REAL MATRIX is one where all the entries are real $M_{ij} \in \mathbb{R}$. A NON-NEGATIVE MATRIX is one where all entries are zero or positive $M_{ij} \geq 0$.

Definition 7.2.1. Irreducible Matrix An IRREDUCIBLE MATRIX is one where you can not rearrange rows and columns of the matrix to produce a block upper-triangular[3] form.

A block upper-triangular form is one which has an off-diagonal block of just zeros, for instance

$$\begin{pmatrix} M_{11} & M_{12} & M_{13} & M_{14} & M_{15} \\ M_{21} & M_{22} & M_{23} & M_{24} & M_{25} \\ M_{31} & M_{32} & M_{33} & M_{34} & M_{35} \\ 0 & 0 & 0 & M_{44} & M_{45} \\ 0 & 0 & 0 & M_{54} & M_{55} \end{pmatrix} \text{ is reducible} \tag{7.6}$$

Note that the opposite of an irreducible matrix is a REDUCIBLE MATRIX which is a matrix which can be organised with square blocks along the diagonal and then the corresponding off-diagonal blocks are only non-zero in the upper triangular part[4]. That makes the off-diagonal blocks rectangular in general. For instance if we have split our matrix into two parts, the diagonal pieces will be $\mu \times \mu$ and $\nu \times \mu$ with $\mu + \nu = n$ the dimension of the total matrix. The off-diagonal blocks will then be $\mu \times \nu$ and $\nu \times \mu$ blocks. In the example (7.6) we have $\mu = 3$ and $\nu = 2$. An example of a matrix which looks similar to a reducible matrix, such as (7.6), but is in fact *irreducible* is

$$\begin{pmatrix} M_{11} & M_{12} & M_{13} & M_{14} & M_{15} \\ M_{21} & M_{22} & M_{23} & M_{24} & M_{25} \\ M_{31} & M_{32} & M_{33} & M_{34} & M_{35} \\ 0 & 0 & M_{43} & M_{44} & M_{45} \\ 0 & 0 & M_{534} & M_{54} & M_{55} \end{pmatrix} \text{ is irreducible.} \tag{7.7}$$

Remember a strongly connected network always has an irreducible adjacency matrix. This is because any power of a reducible matrix is also itself reducible.

[1] You need to use JORDAN NORMAL FORMS, see Michael Gastner's lecture notes, ch.4. Alternatively, you may imagine adding a small (positive) but different term to every entry which will create a better behaved matrix. For instance you could add ϵ to every entry where we keep $0 < \epsilon \ll 1/N$.

[2] In some circumstances it can be useful to assign negative weights to edges. For instance to indicate enmity rather than friendships a social context. Using ± 1 in entries of an adjacency matrix is certainly one way to describe this situation but many of the linear analysis tools we are describing here can not be applied or do not make any sense. For such SIGNED NETWORKS a different approach is needed.

[3] Alternatively, you may consider the lower block diagonal structure, replacing 'upper' by 'lower' in what we say here.

[4] It is often easier to define an irreducible matrix as one which is not a reducible matrix.

Definition 7.2.2. The PERRON-FROBENIUS THEOREM (1907,1912)
Consider a real $N \times N$ non-negative irreducible matrix M. This means the following
Let $\mathbf{v}^{(i)}$ be the eigenvector with eigenvector λ_i. Such a matrix M has the following properties:-

- The largest eigenvalue, λ_1, is unique and real, $\lambda_1 > |\lambda_j| \quad \forall j \neq 1$.
- The PERRON VECTOR $\mathbf{p} = \mathbf{v}^{(1)}$ is the right-eigenvector associated with this eigenvalue λ_1 and it is the **only** eigenvector with the same sign for all entries. So we may choose[5] $p_i = v_i^{(1)} > 0 \quad \forall\, i$, $\sum_i v_i^{(n)} = 0$ iff $n = 1$.
- The largest eigenvalue λ_1 is bounded by the sums of the rows of M

$$\min_i\{\sum_j M_{ij}\} \leq \lambda_1 \leq \max_i\{\sum_j M_{ij}\} \tag{7.8}$$

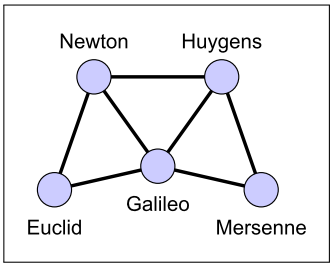
- The largest eigenvalue λ_1 is bounded by the sums of the columns of M

$$\min_j\{\sum_i M_{ij}\} \leq \lambda_1 \leq \max_j\{\sum_i M_{ij}\} \tag{7.9}$$

Let us look at some examples which illustrate the Perron-Frobenius theorem.

Example 7.2.1. Perron-Frobenius on 5 Mathematicians
Let us look at a simple network formed by the links between five mathematicians on the MacTutor website. The adjacency matrix is

$$\mathsf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \tag{7.10}$$



where the numerical values of indices on the adjacency matrix correpsond to mathematicians as follows:- 1=Newton, 2=Euclid, 3=Galileo, 4=Huygens, 5=Mersenne.

This is clearly a non-negative matrix. The adjacency matrix is also not reducible as, by inspection, there is a single connected component; there is a path from any one vertex to any other. That is the network is strongly connected. Since it is simple network, the row and column sums of the adjacency matrix are the same and so the row sums tell us exactly the same as the column sums. That is that we know that $2.0 \leq \lambda_1 \leq 4.0$ from the Perron-Frobenius theorem. The full set of eigenvalues and eigenvectors for this adjacency matrix are as follows (numerical results given to two-significant figures):

	n	Perron 1	2	3	4	5
	λ	2.94	0.62	−0.46	−1.62	−1.47
(N)	$\mathbf{v}_1^{(n)}$	-0.47	-0.37	0.51	0.60	0.14
(E)	$\mathbf{v}_2^{(n)}$	-0.35	-0.60	-0.44	-0.37	0.43
(G)	$\mathbf{v}_3^{(n)}$	-0.56	-0.00	-0.31	-0.00	-0.77
(H)	$\mathbf{v}_4^{(n)}$	-0.47	0.37	0.51	-0.60	0.14
(M)	$\mathbf{v}_5^{(n)}$	-0.35	0.60	-0.44	0.37	0.43

[5]The only alternative is to choose all entries negative but this is usually less intuitive.

We can clearly see that the largest eigenvalue is 2.94, within the bounds demanded by the Perron-Frobenius theorem. Likewise the Perron vector, labelled $n = 1$, has all the same sign (here the numerical routine chooses negative values) while all the other eigenvalues have mixed signs.

Example 7.2.2. Directed and weighted matrix example satisfying Perron-Frobenius

For our second example, let us look at a directed network with adjacency matrix

$$\mathsf{A} = \begin{pmatrix} 1 & 2 & 1 & 2 \\ 3 & 4 & 3 & 4 \\ 1 & 4 & 2 & 4 \\ 3 & 1 & 3 & 2 \end{pmatrix} \tag{7.11}$$

Again this is a non-negative matrix and every pair of vertices is connected by an edge. It is clearly also strongly connected and so irreducible. We may therefore apply Perron-Frobenius to this adjacency matrix. This gives bounds on λ_1 from the row and column sums of the form $8.0 \leq \sum_j A_{ji}, \sum_i A_{ji} \leq 12.0$. A numerical solution for the eigenvalues and eigenvector gives us

n	Perron 1	2	3	4
λ	10.16	-1.85	0.45	0.24
$\mathbf{v}_1^{(n)}$	0.29	0.26	-0.18	0.18
$\mathbf{v}_2^{(n)}$	0.67	0.02	-0.68	-0.56
$\mathbf{v}_3^{(n)}$	0.56	0.65	0.04	-0.40
$\mathbf{v}_4^{(n)}$	0.39	-0.71	0.71	0.70

The largest eigenvalue is 10.16, within the bounds demanded by the Perron-Frobenius theorem. The associated eigenvector, the Perron vector with label $n = 1$, has all the same sign while all the other eigenvalues have mixed signs, again as demanded by the theorem.

Example: Example of matrix not satisfying Perron-Frobenius Consider the matrix

$$\mathsf{A} = \begin{pmatrix} 1 & 2 & -1 & -2 \\ 3 & -4 & 3 & 4 \\ 1 & 4 & -2 & -4 \\ 3 & 1 & 3 & 2 \end{pmatrix} \qquad 0.0 \leq \sum_j A_{ji}, \sum_i A_{ji} \leq +8.0 \tag{7.12}$$

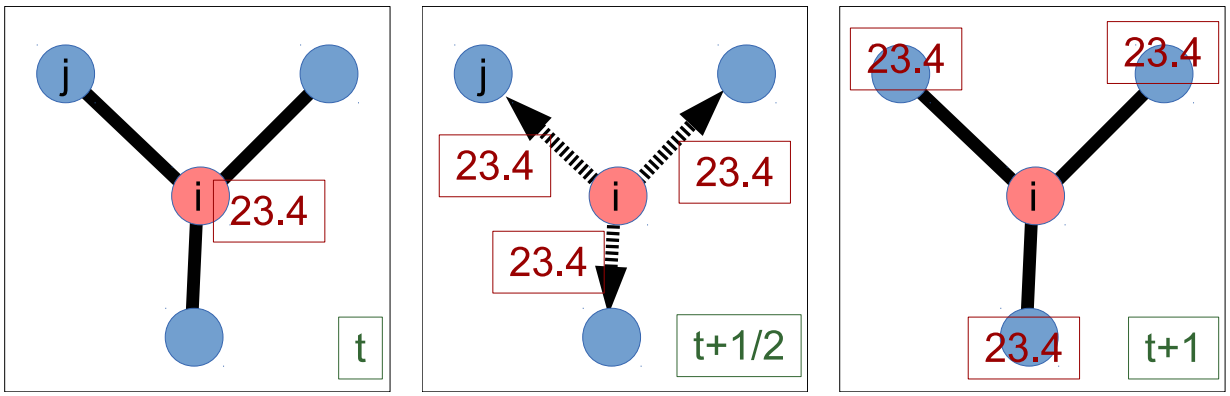
The negative entries means it does not satisfy the Perron-Frobenius criteria and we find that no largest real positive eigenvalue and the eigenvectors do not have any simple features

n	0	1	2	3
λ	$1.41 + 2.10i$	$1.41 - 2.10i$	0.16	-5.98
$\mathbf{v}_0^{(n)}$	-0.09+0.19i	-0.09-0.19i	-0.15	0.25
$\mathbf{v}_1^{(n)}$	$0.45 + 0.11i$	$0.45 - 0.11i$	-0.28	-0.79
$\mathbf{v}_2^{(n)}$	-0.21+0.31i	-0.21-0.31i	0.66	0.53
$\mathbf{v}_3^{(n)}$	0.77	0.77	-0.68	-0.19

7.3 Broadcasting and Centrality measures

As we noted above, it is interesting to find network measures, such as centrality measures, which are sensitive to the whole network structure which probe all the links and routes in the network. However it would make a lot of sense if we did this using a simple local process which might capture some aspect of the way we use a given network in practice. For instance is there a linear process which roughly imitate the way rumours spread on a social network?

The first such simple process we consider is a BROADCAST PROCESS as shown in Figure 7.2. In a broadcast process we imagine that at each time step t, every node i has some value $w_i(t)$ representing



(a) (b) (c)

Figure 7.2: Illustration of one iteration of a broadcast process showing how the value from vertex i alone is distributed. In (a) we show that vertex i starts with value 23.4. This value is then copied to form part of the value of each neighbouring vertex at the next time step. There is no amplification or reduction of this value as each edge has weight 1.0. So we can imagine 23.4 flowing along each edge, as shown in (b), arriving at the neighbouring vertices at time $(t+1)$. Note that (c) shows that the value of vertex i at time t does not contribute to the value at i at time $(t+1)$.

its importance. As we are considering the context of a centrality measure, this value will be the number of 'votes' that vertex currently has and we are hoping that these votes will settle down to some equilibrium which we can interpret. At every tick of the clock, each node i copies or **broadcasts** its current 'value' to **all** its neighbours j. Any edge weight for the link from i to j, A_{ji}, is used as a multiplier, amplifying the signal if the weight is greater than one, damping it if the weight is less than one. Thus the contribution from node i to the total value at node j at the next time step will be $A_{ij}w_i(t)$. Each node adds up all the values ('votes') it receives from its neighbours and this becomes the new value for that node. An important feature of this process is that the number of votes is not conserved as the number of votes sent out by each vertex is proportional to its strength. Unless the graph is regular, unless every node has the same out-strength, the number of votes sent is not just proportional to the w_i at value each vertex i. The higher degree nodes will have proportionality more influence than lower degree nodes so we are making high degree nodes more important in this broadcast process.

One view of this broadcast process is that the more neighbours you have, the more important you are. However this process also takes account that some nodes are more important than others. The more important a node is, as represented by its current value, the more important is its input, its vote, to the value of neighbouring vertices. Amplifying the value of each vote by the value of the edge means that a weak link will have less influence than a strong tie. All of these ideas seem reasonable in an abstract sense. If we had information on a particular real world process we could construct a more realistic process perhaps. The point is that this simple process balances realism with simplicity, the latter allowing us to exploit linear algebra techniques to perform its analysis efficiently.

From this basic definition of broadcasting we can derive two well known centrality measures: Eigenvalue centrality and Katz centrality.

7.3.1 Eigenvalue centrality

Definition 7.3.1. Eigenvalue Centrality

The EIGENVALUE CENTRALITY E_i of vertex i is the i-th entry in the Perron vector $E_i = p_i = v_i^{(1)}$ of the adjacency matrix.

We can justify that this definition gives a reasonable centrality by thinking of the following process.

1. Each vertex j starts with centrality value $w(0)_j$ at step $t = 0$

2. Each vertex **broadcasts** (copies) its value to all their neighbours, weighted (amplified or reduced) by any edge weight.

3. The neighbours sum up all these values, their 'votes', to get their new centrality value for next step $t+1$, namely $w(t+1)_j$.
   So a vertex with strong links to important vertices will get a large boost in their value. A vertex with few or weak links to unimportant neighbours will have a low score.
   The process can therefore be described by the equation

$$w(t+1)_i = \sum_j A_{ij} w(t)_j \,. \tag{7.13}$$

4. Repeat for long time, i.e. $t \to \infty$.

5. The long-time limit of this process gives the Eigenvalue centrality value E_i of the i-th vertex and this is is proportional to the Perron vector entry p_i

$$\boxed{E_i = \lim_{t\to\infty} w(t)_i \propto p_i \,, \qquad w(t)_i = \sum_j A_{ij}\; w(t-1)_j} \tag{7.14}$$

Note that this measure encodes some kind of feedback as the score or importance of a vertex changes as the importance of its neighbours evolves.

In the longtime limit of this process we reach a steady scaling of the values at each node, in which the value at every node changes by a factor given by the largest eigenvalue of the adjacency matrix. In this steady scaling regime, the score of each vertex is proportional to its entry in the Perron vector of the adjacency matrix. We can show this as follows.

Eigenvalue centrality proof

First decompose initial vector in terms of right-eigenvectors

$$w(t=0)_j = \sum_n c_n v_j^{(n)} \,. \tag{7.15}$$

Next you deduce (e.g. by induction) that after t steps, application of A^t, the scores will evolve into the form

$$\begin{aligned}
\mathbf{w}(t+1) &= \mathsf{A}\mathbf{w}(t) && (7.16)\\
\mathbf{w}(t+2) &= \mathsf{A}\mathsf{A}\mathbf{w}(t) && (7.17)\\
\Rightarrow \mathbf{w}(t) &= (\mathsf{A})^t \mathbf{w}(t=0)\,. && (7.18)\\
\Rightarrow \quad w(t)_i &= \sum_n c_n (\lambda_n)^t v_j^{(n)} \,. && (7.19)
\end{aligned}$$

As $t \to \infty$ the Perron vector term dominates since it has the largest eigenvalue,

$$\lim_{t\to\infty} (|\lambda_n|/\lambda_1)^t \to 0 \,, \tag{7.20}$$

so

$$\lim_{t\to\infty} w(t)_i = c_1 (\lambda_1)^t v_j^{(1)} \,. \tag{7.21}$$

This gives us

$$E_i = \lim_{t\to\infty} w(t)_i \propto p_i \,. \tag{7.22}$$

That is the Eigenvalue centrality of a vertex is given by its entry in the Perron vector. Note the normalisation of an eigenvector is arbitrary so the the Eigenvalue centrality of a vertex is only defined relative to one another, that is only ratios of centrality values E_i/E_j have absolute meaning here.

Path Interpretation of Eigenvalue centrality

We mentioned before that if $A_{ij} \in \{0, 1\}$, we have that $[\mathsf{A}^n]_{ij}$ is the total number of walks of length n from vertex j to vertex i. Thus, for such binary networks (such as a simple network) another interpretation of Eigenvalue centrality of a vertex is that it is proportional to the number of long walks passing through that vertex.

7.3.2 Katz centrality

There are a few problems with Eigenvalue Centrality. First not all $N \times N$ matrices have N linearly independent eigenvectors, e.g. matrices which are projections or nilpotent. In terms of networks, this corresponds to nodes that are unreachable from some starting points or can not reach all other vertices. e.g. a web site which is not linked from any other site or web sites with no hyperlinks on their pages.

One solution is to change our process so that we add a basic value for centrality, β, to each vertex at each time step in addition to the votes each vertex gets from its neighbours. The broadcast process is now described by the equation

$$w(t+1)_i = \beta + \alpha \sum_j A_{ij} w(t)_j \,. \tag{7.23}$$

Note we are also adding an amplification or damping factor α.

The Katz Centrality K_i of vertex i is proportional to the long-time limit of this process

$$\boxed{K_i = \lim_{t\to\infty} w(t)_i \,, \qquad w(t)_i = \beta + \alpha \sum_j A_{ij} w(t-1)_j} \tag{7.24}$$

The addition of β at each step guarantees a minimum value of β for the Katz centrality for each vertex. The β term also means that this centrality is well defined for *any* network if α is small enough. We can see this by finding a general solution for Katz Centrality as follows. Here $\mathbf{1}$ is a vector where every entry is one, $1_i = 1$.

$$\begin{aligned}
\mathbf{w}(t) &= \beta\mathbf{1} + \alpha\mathsf{A}\mathbf{w}(t-1) && (7.25)\\
&= \beta\mathbf{1} + \alpha\mathsf{A}\left(\beta\mathbf{1} + \alpha\mathsf{A}\mathbf{w}(t-2)\right) && (7.26)\\
&= \beta\mathbf{1} + \beta\alpha\mathsf{A}\mathbf{1} + (\alpha)^2\mathsf{A}\mathsf{A}\mathbf{w}(t-2) && (7.27)\\
&= \beta\mathbf{1} + \beta\alpha\mathsf{A}\mathbf{1} + \beta(\alpha\mathsf{A})^2\mathbf{1} + \ldots + \beta(\alpha\mathsf{A})^{t-1} + (\alpha\mathsf{A})^t\mathbf{w}(0) && (7.28)
\end{aligned}$$

So if α is small enough (clearly $|\alpha A_{ij}| < 1 \,\forall\, i, j$ is sufficient), the last term tends to zero as $t \to \infty$,

$$\lim_{t\to\infty} (\alpha\mathsf{A})^t\mathbf{w}(0) = 0 \,, \tag{7.29}$$

That is initial conditions $\mathbf{w}(t = 0)$, don't effect the long time result which is

$$\lim_{t\to\infty} \mathbf{w}(t) \;=\; \beta\sum_{t=0}^{\infty}(\alpha\mathsf{A})^t\mathbf{1} \,. \tag{7.30}$$

We now see that β acts as an overall scale and so as long as we compare ratios, β is also irrelevant. So we can define the Katz centrality K_i of a vertex i as the corresponding entry in the vector

$$\mathbf{K} = \sum_{t=0}^{\infty}(\alpha\mathsf{A})^t\mathbf{1} = (\mathbb{1} - \alpha\mathsf{A})^{-1}\,\mathbf{1} \,. \tag{7.31}$$

For graphs with adjacency matrix containing only ones and zeros $A_{ij} \in \{0, 1\}$ (e.g. simple graphs) Katz Centrality can be interpreted in terms of paths again. Since $(\mathsf{A}^t)_{ij}$ is number of walks (cycles allowed) from j to i of length t we have that the Katz centrality of vertex i is proportional to the number of walks which end at i from *any* starting vertex, where we **weight** walks by α^t for a walk of length t.

We can also be more precise about the value of α needed for this process to converge to give us a well defined Katz Centrality. If we use $\mathsf{A} = \mathsf{V}\mathbf{\Lambda}\mathsf{V}^{-1}$ then $(\mathsf{A})^n = \mathsf{V}(\mathbf{\Lambda})^n\mathsf{V}^{-1}$ so

$$\mathbf{K} \propto \mathsf{V}.\left(\sum_{t=0}^{\infty}(\alpha\mathbf{\Lambda})^t\right)\mathsf{V}^{-1}\mathbf{1} \tag{7.32}$$

$$= \mathsf{V}.\begin{pmatrix} (1-\alpha\lambda_1)^{-1} & 0 & 0 & \cdots \\ 0 & (1-\alpha\lambda_2)^{-1} & 0 & \cdots \\ 0 & 0 & (1-\alpha\lambda_3)^{-1} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}.\mathsf{V}^{-1}\mathbf{1} \tag{7.33}$$

provided $|\alpha\lambda_n| < 1$ for convergence of sums. The most stringent of these N constraints from the diagonal values comes from the largest eigenvalue so for a well defined Katz centrality we need

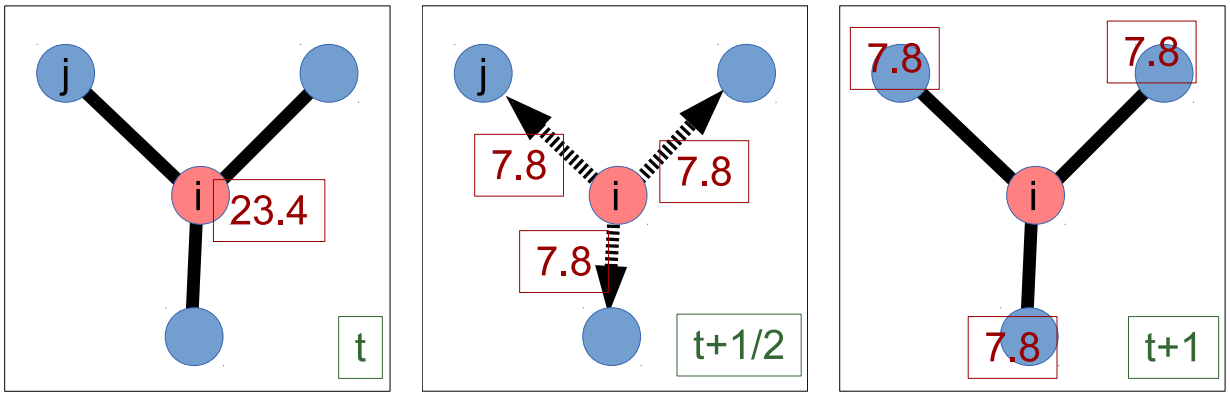
$$|\alpha| < \frac{1}{\lambda_1}\,. \tag{7.34}$$

To summarise for Katz centrality

- The allowed values for the α parameters of Katz centrality are bounded by (7.34).
- Only relative values of Katz centrality have any meaning, (K_i/K_j) is important.
- As long as β is non-zero, it's value is otherwise of no consequence.
- For simple graphs, Katz centrality counts the number of walks of length ℓ weighted by $(\alpha)^\ell$.

7.4 Diffusion and PageRank Centrality

An alternative to the broadcast process is a DIFFUSION PROCESS on the network. In this case, each node i *divides* its current 'value' between its neighbours j in proportion to the weights of the edges leaving vertex i, i.e. in proportion to A_{ji}. Thus the flow of 'votes' away from a vertex is divided in proportion to edge weights leaving that vertex as illustrated in Fig. 7.3. This means the number of 'votes' on the network is conserved.



(a) (b) (c)

Figure 7.3: Illustration of one iteration of a diffusion process showing how the value from vertex i alone is distributed. In (a) we show that vertex i has value 23.4. This value is then split equally (all edges have equal weight) so a value of 7.8 is sent to to form part of the value of each neighbouring vertex at the next time step. We can imagine 7.8 flowing along each edge, as shown in (b), to arrive at the neighbouring vertices at time $(t+1)$. Note that in (c), the value at vertex i at time t does not contribute to its own value at time $(t+1)$.

The diffusion process is described by the TRANSFER MATRIX T, where T_{ji} describes the fraction of the value from source vertex i passed onto one neighbour j.

$$w(t+1)_j = \sum_i T_{ji} w(t)_i \tag{7.35}$$

where

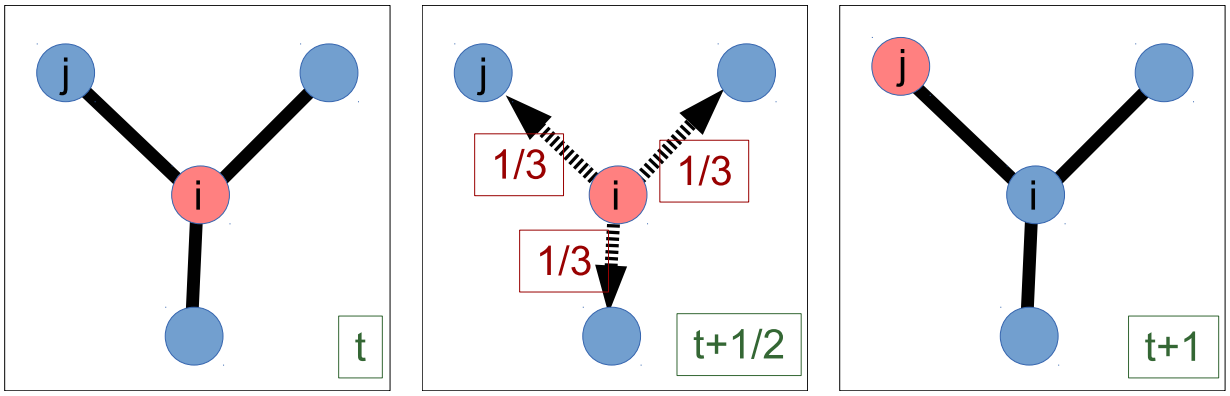
$$T_{ji} = \frac{1}{s_i^{(\text{out})}} A_{ji} \qquad s_i^{(\text{out})} = \sum_j A_{ji} \qquad (= k^{(\text{out})} \text{ if } A_{ij} \in \{0,1\}) \tag{7.36}$$

Example 7.4.1. Transfer Matrix for 5 Mathematicians Simple Network

$$\mathsf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}, \qquad \mathsf{T} = \begin{pmatrix} 0 & 1/2 & 1/4 & 1/3 & 0 \\ 1/3 & 0 & 1/4 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/3 & 1/2 \\ 1/3 & 0 & 1/4 & 0 & 1/2 \\ 0 & 0 & 1/4 & 1/3 & 0 \end{pmatrix} \tag{7.37}$$

n	1	2	3	4	5
λ	1.00	0.27	−0.17	−0.61	−0.50
$\mathbf{v}_1^{(n)}$	-0.46	-0.45	-0.53	-0.62	-0.00
$\mathbf{v}_2^{(n)}$	-0.31	-0.55	0.27	0.34	0.41
$\mathbf{v}_3^{(n)}$	-0.62	-0.00	0.53	0.00	-0.82
$\mathbf{v}_4^{(n)}$	-0.46	0.45	-0.53	0.62	0.00
$\mathbf{v}_5^{(n)}$	-0.31	0.55	0.27	-0.34	0.41

The transfer matrix for the diffusion process also describes the probability that a **random walker** at source vertex i moves to target vertex j at the next time step, as illustrated in Figure 7.4.



(a) (b) (c)

Figure 7.4: Illustration of the motion of one random walker from vertex i at time t. In (a) we show that vertex i has one random walker present. Figure (b) show the probability that this walker follows one of the edges. As all edges carry equal weight, the probability of following any edge is the same, simple 1/3. In a stochastic process, the walker has to end up at just one vertex and we show one possible outcome in (c).

Conservation and Transfer Matrix Properties

The random walker must go somewhere and we don't add any more walkers, so the total number of random walkers is conserved. Equivalently we noted before that in diffusion the number of votes on

the network is conserved. This conservation law is reflected in the property that the sum of each column of T must equal 1, e.g. the sum of the i-th column is

$$\sum_j T_{ji} = \sum_j \frac{1}{s_i^{(\text{out})}} A_{ji} = \frac{1}{s_i^{(\text{out})}} \left(\sum_j A_{ji} \right) = 1 \,. \tag{7.38}$$

The transfer matrix is non-negative as every entry T_{ji} is the probability for a random walker moving from i to j. If we also assume there is a path from every vertex to every other vertex then we also have an irreducible matrix and we can use the Perron-Frobenius theorem. So we can use the fact that the maximum and minimum of the sums of columns provides upper and lower bound on λ_1 to deduce that largest eigenvalue is always exactly one. This is related to the conservation of the number of random walkers (a conservation of the flow). Suppose the total value assigned to the vertices at time t in this process (total number of random walkers) is $W = \sum_i w(t)_i$. Then we find that

$$\begin{aligned} W(t+1) &= \sum_j w(t+1)_j & (7.39)\\ &= \sum_j \sum_i T_{ji} w(t)_i & (7.40)\\ &= \sum_i \left(\sum_j T_{ji} \right) w(t)_i & (7.41)\\ &= \sum_i w(t)_i = W(t) \,. & (7.42) \end{aligned}$$

So the total number of votes $W(t)$, the total number of random walkers on the network, never changes throughout this process and the conservation law is proven.

Special properties of Diffusion Matrices

For strongly connected networks, Perron-Frobenius gives us several properties shared by the transfer matrix of a diffusion process on a network

- The largest eigenvalue is 1, $\lambda_1 = 1$.
  Corresponds to the conservation of the number of random walkers.
  — Use the column sums of T to show this.

- The left-eigenvector associated with the largest eigenvalue is proportional to $\mathbf{1}$, i.e. we may choose
  $$u_i^{(1)} = 1 \,. \tag{7.43}$$
  This comes from the conservation of probability for the process at each vertex.
  — Direct substitution into the eigenvalue equation confirms this.

- All the other right eigenvectors of T, $\mathbf{v}^{(n)}$ for $n \geq 2$, have entries which sum to zero, i.e.
  $$\sum_i v_i^{(n)} = 0, \qquad (n \geq 2) \,. \tag{7.44}$$
  This again means that these non-leading eigenvectors have both positive and negative entries.
  — Use orthogonality of right and left eigenvectors with $\mathbf{u}^{(1)}$.

As usual there are some special properties that are only true for the case of an undirected network, i.e. where the adjacency matrix is symmetric $\mathsf{A}^\mathsf{T} = \mathsf{A}$.

- If the network is undirected, then the entry of Perron eigenvector of the corresponding diffusion process transfer matrix for a given vertex i is proportional to strength (degree if unweighted) of

that vertex
— Use direct substitution into the eigenvalue equation to show this.

$$p_i = v_i^{(1)} \propto s_i = \sum_j A_{ij} \text{ if } \mathsf{A}^\intercal = \mathsf{A}\,. \tag{7.45}$$

- If the network is undirected, the eigenvalues of the diffusion process transfer matrix T are all real.
  — You can consider the normalised adjacency matrix

$$\mathsf{A}' = \mathsf{S}^{-1/2}\mathsf{A}\mathsf{S}^{-1/2}\,, \qquad S_{ij} = s_i\delta_{ij}\,, \tag{7.46}$$

  where S is a diagonal matrix with the strength of the vertex i on the i-th diagonal entry. This is also a symmetric matrix $\mathsf{A}' = (\mathsf{A}')^\intercal$ and so has real eigenvalues. The transition matrix T is then conjugate to this A' matrix as $\mathsf{T} = \mathsf{S}^{-1/2}\mathsf{A}'\mathsf{S}^{+1/2}$ so it shares the same eigenvalues.

- If the network is undirected, the eigenvectors of the diffusion process transfer matrix T are all real.
  Again this is true for a real symmetric non-negative matrix such as the normalised adjacency matrix A' of (**??**). Since the T is conjugate to A' using a real matrix $\mathsf{S}^{1/2}$, the entries of the eigenvectors of T are only rescaled by real values from those of A so T.

Our first look at the PageRank centrality measure R_i of vertex i is to use a simplified definition that it is just proportional to the number of random walkers at vertex i at time t in the infinite time limit,

$$\boxed{R_i^{\text{simple}} = \lim_{t\to\infty} w(t)_i\,, \qquad w(t+1)_j = \sum_i T_{ji} w(t)_i\,.} \tag{7.47}$$

As before this is given by the eigenvector with the largest eigenvalue of the transfer matrix T, the Perron vector p_i of that matrix

$$R_i^{\text{simple}} = p_i\,. \tag{7.48}$$

Note that for this Markovian transfer matrix T here we know $\lambda_1 = 1$ so there is no scaling with each step, the process will naturally converge to this answer.

For undirected networks with a single component (so it is strongly connected) we can go further as we know the Perron vector in this case is just proportional to the degree of the network. So PageRank is trivial in this case, it is equal to degree.

For directed networks, the situation is not so simple. There is no known general solution for the Perron vector of T in this case. However, we often see that in practice the PageRank is correlated with in-degree. To see why consider approximating our network with adjacency matrix A with one where the adjacency matrix N has the form

$$\mathsf{N} = W^{-1} s_j^{(\text{out})} s_i^{(\text{in})}\,, \qquad W = \sum_{i,j} A_{ij} \tag{7.49}$$

The point here is that we have defined this new network so it has the same in- and out-strength for every vertex,

$$s_i^{(\text{out})} = \sum_j A_{ij} = \sum_j N_{ij}\,, \qquad s_j^{(\text{in})} = \sum_i A_{ij} = \sum_i N_{ij}\,. \tag{7.50}$$

The edges of our new network have a weight equal to what we would measure on average when performing the configuration model many times on our network. So it retains only the local information at each vertex, all the other structure is lost. It is a useful null model in many problems, hence the notation N for it's adjacency matrix. Now this null model does have a simple solution for the Perron eigenvector. It is easy to see by direct substitution that the entry in the Perron eigenvector of N for the i-the vertex is equal to the *in-strength* of that vertex, the *in-degree* for unweighted networks, $p_i = s_i^{(\text{in})}$.

Our average configuration model network with the edge weights 'smeared out' over all pairs of vertices appears to be a long way from a real network. Nevertheless, it appears in practice that PageRank values are often correlated with in-strength (in-degree), suggesting this can be a useful null model for directed networks.

Full PageRank centrality measure

However this simple definition of PageRank has the same issues in practice as we noted for Eigenvalue centrality. So just as some simple modifications to Eigenvalue centrality got round such problems, to give us Katz centrality, PageRank is normally defined in a more sophisticated way which will always give us a well defined PageRank. Consider the following issues.

- What if there were no exits from one vertex $k_i^{(\text{out})} = 0$?
  Then $T_{ji} = 0/0$ so the process is badly defined.
  One solution is to replace the entries in any column i where all entries are zero, i.e. where $T_{ji} = 0$ for all j and so $k_i^{(\text{out})} = 0$. A new standard value to choose is to have a new column where all entries are $1/N$, i.e. define $T_{ji} = 1/N$. This means that if random walkers reach a dead-end node with $k_i^{(\text{out})} = 0$ they now jump to a randomly chosen vertex.

- What if there were no entrances from a vertex i, so $k_i^{(\text{in})} = 0$?
  This will lead to $R_i = 0$ for that vertex i.
  You can just live with this as PageRank will always give lowest rank to vertices with no in-degree. If at least one dead-end has been removed using the replacement suggested above, then this will give a path into every vertex which solves this issue too.

So we now arrive at the general definition of PageRank centrality measure[6], as given first by Brin & Page [Brin and Page, 1998], famously the original basis for the success of the Google search engine. The PageRank R_i of vertex i is proportional to the long-time limit of the following process

$$R_i = \lim_{t\to\infty} w(t)_i \,, \tag{7.51}$$

$$w(t+1)_i = (1-\alpha)h_i + \alpha \sum_j T_{ij} w(t)_j = \sum_j R_{ij} w(t)_j \tag{7.52}$$

$$R_{ij} = (1-\alpha)h_i 1_j + \alpha \sum_j T_{ij} w(t)_j \,. \tag{7.53}$$

You may interpret the α parameter as the random walkers choosing to move along an edge to a neighbour a fraction α of the time, but the rest of the time the random walker makes a 'hyperjump' to a random vertex i chosen with probability h_i. Note this means we generally demand that $\sum_i h_i = 1$. That means that on average a random walker makes $\alpha/(1-\alpha)$ steps before making a hyperjump. The value used for α will vary with the context but a traditional value is $\alpha = 0.85$ for which the average number of steps taken by a random walker contributing to PageRank is about 5.67.

Normally the hyperjump is made with an equal probability for all vertices, $h_i = 1/N$ and in this case the PageRank matrix R is often called the Google matrix G which is of the form

$$G_{ij}(\alpha) = (1-\alpha)\frac{1}{N} 1_i 1_j + \alpha T_{ij} \,. \tag{7.54}$$

Here $1_i = 1$ for all i. Note that to maintain this probabilistic interpretation we need the entries of the hyperjump vector to sum to one, $\sum_i h_i = 1$.

PageRank analysis

The formal analysis works exactly as for Katz centrality. For convergence we need $|\alpha|\lambda_1 < 1$ so here we may deduce that $|\alpha| < 1$. This ensures that $\lim_{t\to\infty}(\alpha\mathsf{T})^t = 0$ giving us the PageRank centrality

[6] The name derives from one of the authors though conveniently it was also first applied to web pages.

measure R_i as of vertex i as the long time limit of this diffusion plus hyperjump process

$$\mathbf{R} = \lim_{t\to\infty} \mathbf{w}(t) = (1-\alpha)\left[\sum_{s=0}^{\infty}(\alpha\mathsf{T})^s\right]\mathbf{h} = (1-\alpha)\left[\mathbb{1}-\alpha\mathsf{T}\right]^{-1}\mathbf{h} \tag{7.55}$$

We may write this succinctly as

$$\mathbf{R} = \frac{(1-\alpha)}{(1-\alpha\mathsf{T})}\mathbf{h}\,, \tag{7.56}$$

This form (7.56) is really just a mnemonic for (7.55), the former is defined by the binomial expansion of the inverse in powers of α.

We noted above that the simplified PageRank (no hyperjumps so the $\alpha = 1$ limit of our full definition) is given by the entries of the the Perron vector. However for $\alpha < 1$ all eigenvectors contribute to the PageRank result (7.56) because in general, the hyperjump vector $\mathbf{h}$ is expressed as a sum of many eigenvectors. If we write

$$\mathbf{h} = \sum_n h^{(n)}\mathbf{v}^{(n)} \tag{7.57}$$

then the PageRank result (7.56) can be rewritten as

$$\mathbf{R} \;=\; (1-\alpha)\sum_n \frac{h^{(n)}}{(1-\alpha\lambda_n)}\mathbf{v}^{(n)}\,,\,. \tag{7.58}$$

Still the term $(1-\alpha\lambda_n)^{-1}$ is largest for the leading eigenvalue $\lambda_1 = 1$ so the Perron vector is likely to be the most important contribution. Thus we would expect to see a correlation between full PageRank centrality and vertex strength (degree for unweighted cases). For directed networks, we can use our analysis for the simple PageRank case in terms of the null model built from the average over all configuration models, $N_{ij} = s_j^{(\mathrm{out})}s_i^{(\mathrm{in})}/W$ of (7.49). That showed that the in-strength of a vertex is often a good first guess for the Perron vector entry and again, that that eigenvector may well dominate our PageRank solution (7.58) for reasonably large α. This correlation of in-strength and the full PageRank is often found in practice.

The solution also shows that as $\alpha \to 0$, PageRank becomes equal to the hyperjump vector, $\mathbf{R} = \mathbf{h}$. Thus it is clear that to get non-trivial information about the network we need to keep α away from small values. Finding an appropriate value for α is not trivial. We know that the random walkers make $\alpha/(1-\alpha)$ steps between hyperjumps on average so to probe the network structure we might want to make sure this is bigger that one, suggesting that $\alpha > 0.5$ is sensible. Too close to $\alpha = 1$ and the PageRank will be dominated by the Perron vector, which may or may not be desired. The traditional value of $\alpha = 0.85$ is therefore not a bad first guess.

PageRank, Web Pages and Google

Brin and Page (1998) defined a web search engine in terms of the PageRank algorithm. This is the basis for the ranking of web sites in a google search. The higher the PageRank (centrality) of a web page in a web page network, the higher it is placed when returned by a search engine. The basis of Google's success is that web pages form a directed network but they do not record which other pages link to them. This means that PageRank is not trivial to calculate yet it seems to be a better match statistically to the way real people view the web. So that would suggest that while individual behaviour is not random, viewed as a whole, coarse graining over many people, the behaviour is then statistically similar to that of a random walker who has a limited attention, around five clicks, before they start a completely new search. Another part of Google's success is that they found ways to store information about all web pages. PageRank probes the whole network so you need to know about the links on all to calculate PageRank. Google's success illustrates how a centrality measure can be used. By showing viewers pages listed in order of their centrality measure, here PageRank, they are using centrality to make recommendations to a user. So centrality measures of networks are part of a much wider discussion about , an important topic in computer science.

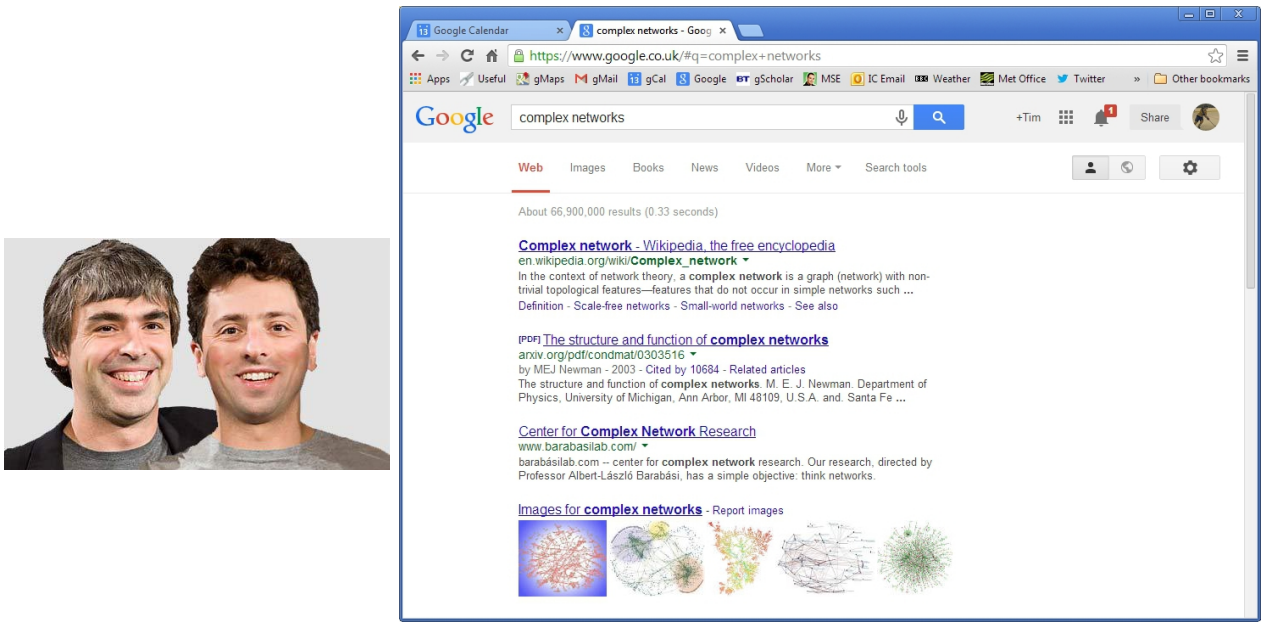


Figure 7.5: Brin and Page, the founders of Google. The success of the search engine was built on the use of PageRank to rank web pages so that results of searches were presented in order of PageRank values of the web pages using the directed but unweighted network defined by hyperlinks between web pages.

7.4.1 (#) Comparing Centrality Measures

So far we have given a number of centrality measures: degree, betweenness, closeness, eigenvalue centrality and PageRank. However, there are a vast number of centrality indices available — Schoch has a nice visualisation of many as a periodic table [Schoch, 2015, 2016]. This suggests that many of these measures, perhaps even the ones we have considered, are encoding similar information so that there is some (much) redundancy in this zoo of centrality measures. One way to look for this is to study the correlations between centrality indices on a variety of networks [Batool and Niazi, 2014; Bolland, 1988; Lee, 2006; Lozares et al., 2015; Oldham et al., 2019; Rothenberg et al., 1995; Schoch et al., 2017; Valente et al., 2008].

We will not look at this in detail but it is informative to look at a graph found by Brandes and Hildenbrand 2014. If we think of vertex centrality as measuring the centre of the network then, by analogy with geometric shapes where there is only centre, then we might imagine that there will be one node which has the largest centrality in a network. It follows that we expect this node to be the most important by any measure of centrality. Of course, a little more thought suggest this is not true for most social systems. There may be a central point of a city as measured by geographical means, but there could well be different central points in a large city for business, politics, entertainment and so forth.

So there has been a long standing challenge to find a network where the vertex which has the largest value for each given centrality measure is a DIFFERENT vertex. The idea is that this shows that the different centrality measures are picking up on different aspects of contractility or importance. As an example will help us to understand this it is best if these are as small as possible so the challenge was to find the SMALLEST such network for the 'classic' centrality measures BCDE: betweenness, closeness, degree, and eigenvalue centrality. By using an exhaustive search of all possibilities, Brandes and Hildenbrand 2014 gave several examples and we present one in Figure 7.6.

The example in Figure 7.6 is very instructive. Node C with highest closeness is in some sense at the geometric centre as it is the only node which is within two steps of every other node. On the other hand, node B with highest betweenness is the only one connected to two "leaves", subgraphs where the only connection to the rest of the graph is through node B. So node B acts as a unique bridge to these two leaves so all routes to and from nodes F and G are via B. If we look at other paths between nodes we there are often several similar routes so no one path dominates. The one exception might be

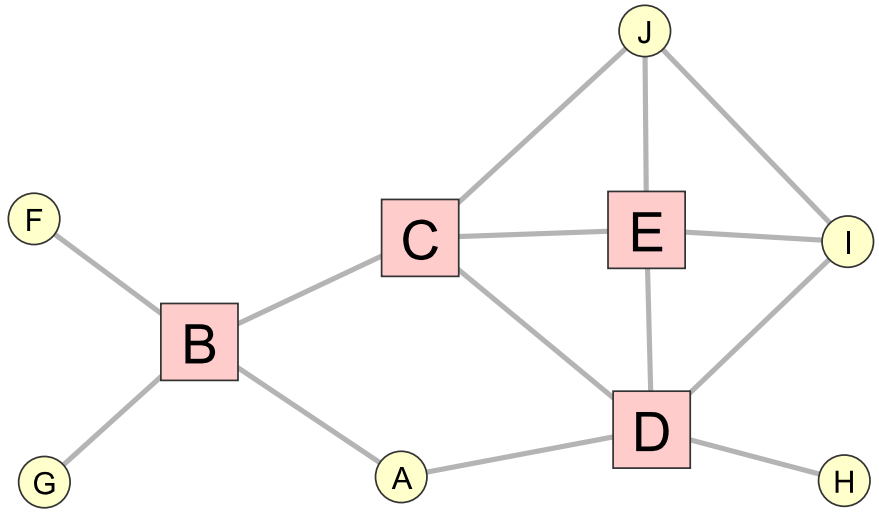


Figure 7.6: The smallest network where the vertices with largest betweenness (B), closeness (C), degree (D), and eigenvalue centrality (E) are all distinct [Brandes and Hildenbrand, 2014].

the only other leaf, node H and its edge, but as there is only one such leaf connected to the bridging node D, we can see why B would have a higher betweenness than D. Perhaps the most interesting is E. This node is at the centre of a dense clique, every neighbour of E is connected to at least two other neighbours. As we have seen, eigenvalue centrality is related to a broadcasting process, so we can see that this dense interconnectivity means that broadcast messages are more likely to stay within the clique $\{C, D, E, I, J\}$ than they would in other regions of our graph.

	Values				Values					
	b	*c*	*k*	*e*	*b*	*c*	*k*	*e*		
Vertex	**BTW**	**CL**	**DEG**	**EV**	**BTW**	**CL**	**DEG**	**EV**	**Mean Rank**	**Vertex**
D	26.17	0.067	5	0.46	2	2	1	2	1.8	D
C	24.83	0.071	4	0.43	3	1	2	3	2.3	C
E	4.00	0.063	4	0.47	5	3	2	1	2.8	E
B	31.50	0.063	4	0.22	1	3	2	6	3.0	B
J	2.17	0.053	3	0.37	6	6	5	5	5.5	J
A	7.50	0.059	2	0.19	4	5	7	7	5.8	A
I	1.83	0.050	3	0.37	7	7	5	4	5.8	I
H	0.00	0.043	1	0.13	8	8	8	8	8.0	H
G	0.00	0.042	1	0.06	8	9	8	9	8.5	G
F	0.00	0.042	1	0.06	8	9	8	9	8.5	F

Table 7.1: The centrality values for the vertices in the network of Figure 7.6. Here BTW is betweenness, CL is closeness, DEG is degree, and EV is eigenvector centrality. The values on the left are the raw values and the values on the right are the ranks of the values. The last column on the right gives the mean value of the four ranked values and the table is ordered by this measure. Values are produces by the built-in analysis tools of the `Visone` package [Brandes and Wagner, 2004].

While Figure 7.6 is a useful illustration of the differences between eigenvalue centralities, there is another message in this small example. Looking at the table of the centrality values confirms the assertion in Brandes and Hildenbrand [2014] that the vertices with the highest value of each of these four centrality measures are distinct. However, the results also show that these all four vertices have high centrality values by at least three of the four measures considered. The vertices with the top three values of these centrality measures are always from the set $\{B, C, D, E\}$. This observation then supports the idea that while the precise centrality value may vary depending on the measure, all these measures do tend to pick out the most important vertices which is what centrality measures are meant to do[7]. The analogy with a city is useful since the the geographical, business, political and entertainment centres of a city are all often close to one another even if they are not usually identical.

[7]We can push this example further. It is clear that the centrality measures are not only identifying the top nodes

7.5 (#) Network Laplacian

Network Laplacian

- Broadcasting with Conservation on a Network
- Properties of the Network Laplacian
- Resistor Networks

7.5.1 Particle Currents — Broadcasting with Conservation on a Network

Consider the of particles on a network but where we add a conservation rule. Again let the number of particles at vertex i at time t be $w(t)_i$. Then we let particles flow from vertex i to vertex j at a constant **rate** which is proportional to the number of particles at i and the 'capacity of the link from i to j which will be A_{ji}, the weight of the edge from i to j. So the particle current, the number particles passing along edge (i, j) (directed from i to j with weight represented by A_{ji} in the adjacency matrix) per unit time, is[8] $DA_{ji}w(t)_i$ where is some rate constant. So far this is as we had with our broadcast process except we scaled the flows by a factor D. The flow down each edge is proportional to the value at the vertex at time t, we are copying that value to it's neighbours $\{j\}$. The key difference here is that we are now demanding that the number of particles is fixed, the loss of particles from a source vertex i to flow along directed edge (i, j) is exactly compensated by the gain in particles at the target vertex j at the other end of the the edge. The number of particles at the vertices conserved. This means that we introduce a second term to represent the loss of particles at a vertex.

Given these dynamics the rate of change in the number of particles at any vertex i is

$$\frac{\partial w(t)_i}{\partial t} = -D \underbrace{\sum_j A_{ji}\mathbf{w}(t)_i}_{\text{flow out}} + D \underbrace{\sum_j A_{ij}\mathbf{w}(t)_j}_{\text{flow in}} \tag{7.59}$$

where D is a rate or diffusion constant which we must specify. The conservation can be seen here by noting that each term in (7.59) appears in the equations for the change in particle number of two different vertices but with opposite sign — once representing the flow into a vertex and the second time representing the flow out of a vertex. These two appearances represent the conservation of flow along on edge. Mathematically the conservation of particle number can be found by showing that the total number of particles

$$W(t) = \sum_i w(t)_i \tag{7.60}$$

is the same for all times t.

Equally important, as rate at which particles lost from a vertex is proportional to $w_i(t)$, the number of particles at a node never goes negative, maintaining our interpretation of this process in terms of real particles.

---

effectively in this network but they are also working well for the middle and low centrality nodes. The nodes A, J and I are consistently in the middle and the nodes F, G and H always have the lowest centrality values. The latter are the three peripheral 'leaf' nodes and this topological position explains their poor performance.

[8] Note that the *linear* form used for our currents, that is the flow along edges $A_{ji}w(t)_i$ is proportional to the particle number, is a very special choice. The linear properties are the basis for all the linear algebra tricks we use. Had we chosen forms such as $A_{ji}(w(t)_i)^2$ we would have complicated if interesting non-linear effects. However even if the fundamental dynamics was nonlinear, a linear analysis of small deviations from equilibrium would return us to the simple linear dynamics assumed here.

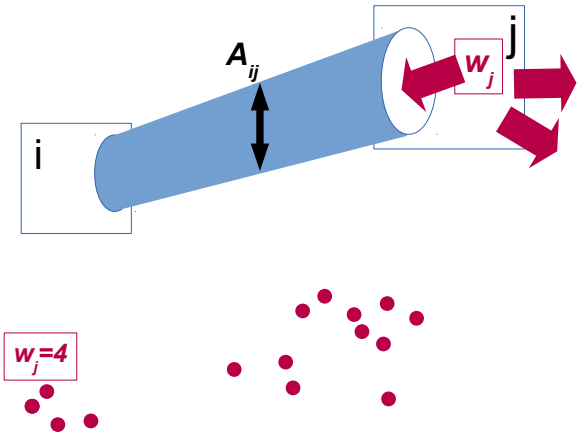


Figure 7.7: A not very useful attempt to illustrate broadcasting with conservation.

We may rewrite this as

$$\Rightarrow \quad \frac{\partial w(t)_i}{\partial t} = -D\underbrace{\sum_j A_{ji}\mathbf{w}(t)_i}_{\text{flow out}} + D\underbrace{\sum_j A_{ij}\mathbf{w}(t)_j}_{\text{flow in}} \tag{7.61}$$

$$= -D\left(\sum_j A_{ji}\right)\mathbf{w}(t)_i + D\left(\sum_j A_{ij}\mathbf{w}(t)_j\right) \tag{7.62}$$

$$= -Ds_i^{(\text{out})}\mathbf{w}(t)_i - D\left(\sum_j A_{ij}\mathbf{w}(t)_j\right) \tag{7.63}$$

$$\Rightarrow \quad \frac{\partial w(t)_i}{\partial t} = -Ds_i^{(\text{out})}\mathbf{w}(t)_i - D\left(\sum_j A_{ij}\mathbf{w}(t)_j\right) \tag{7.64}$$

$$= -D\left(\sum_j \delta_{ij}s_i^{(\text{out})}\mathbf{w}(t)_j\right) - D\left(\sum_j A_{ij}\mathbf{w}(t)_j\right) \tag{7.65}$$

$$= -D\sum_j \underbrace{\left(\delta_{ij}s_i^{(\text{out})} - A_{ij}\right)}_{L_{ij}}\mathbf{w}(t)_j = -D\sum_j \left(L_{ij}\right)\mathbf{w}(t)_j \tag{7.66}$$

$$\Rightarrow \quad \frac{\partial \mathbf{w}(t)}{\partial t} = -D\mathsf{L}\mathbf{w}(t). \tag{7.67}$$

The matrix L captures the dynamics of this broadcast with conservation process and is called the LAPLACIAN for this graph. The network Laplacian is the difference between a diagonal matrix $\mathsf{S}^{(\text{out})}$, containing the in-strength of each relevant vertex on its diagonal, and the adjacency matrix A

$$\boxed{\mathsf{L} = \mathsf{S}^{(\text{out})} - \mathsf{A}}, \qquad \boxed{S^{(\text{out})}_{ij} = \delta_{ij} s^{(\text{out})}_i} = \delta_{ij} \sum_k A_{kj} \tag{7.68}$$

Note that this definition works if there are self-loops in the network but these self-loops do not contribute to the Laplacian defined this way.

As noted above, the process conserves the total number of particles/votes/tokens, $W = \sum_i \mathbf{w}(_)i(t)$, being moved around the network. In fact this form of the evolution equation using a Laplacian defined in terms of *any* matrix A is always a conservative equation. If we sum over index i in (7.67) we have that

$$\sum_i \frac{\partial \mathbf{w}(t)_i}{\partial t} = -D \sum_{i,j} (L_{ij})\, \mathbf{w}(t)_j \tag{7.69}$$

$$\frac{\partial W t}{\partial t} = -D \sum_j \left(\sum_i L_{ij} \right) \mathbf{w}(t)_j = 0 \tag{7.70}$$

since from the definition of the Laplacian in (7.68) we have that

$$\sum_i L_{ij} = \sum_i \left(\delta_{ij} s^{(\text{out})}_i - A_{ij} \right) = s^{(\text{out})}_i - s^{(\text{out})}_i = 0\,. \tag{7.71}$$

This proof makes no assumptions about the properties of the matrix A so it is a universal property of the Laplacian. It corresponds to the existence of a constant left-eigenvector of the Laplacian matrix with eigenvalue zero, as we will note below.

While conservation works for all matrices A, we can see other often desirable properties do require us to limit ourselves to the familiar real non-negative matrices, the typical network adjacency matrix. For instance we often want our $\mathbf{w}(t)_i$ to be non-negative for all time. For instance it may represent the number of visits made to a site by a user, the number of rumours heard by an individual, both of which must be non-negative. To ensure $\mathbf{w}(t)_i$does not go negative, it is sufficient to demand that A_{ij} is real and non-negative.

7.5.2 The Laplacian for an undirected network

Many examples work with undirected networks, and many of these use just simple graphs. The Laplacian for these cases simpler and it also has some special properties. For those reasons it is this form that is most often seen.

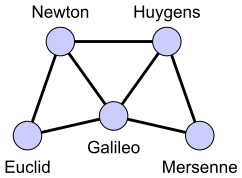
For unweighted graphs S has simply the in-degree $k^{(\text{in})}_i$ on the diagonal rather than the in-strength. In turn this means that for simple graphs S has just the degree k_i on the diagonal which gives the Laplacian in its most common form

$$\boxed{\mathsf{L} = \mathsf{D} - \mathsf{A}}, \qquad \boxed{D_{ij} = \delta_{ij} k_i} = \delta_{ij} \sum_k A_{kj} \text{ for simple graphs.} \tag{7.72}$$

Note that if there are self-loops in an undirected network then this definition again means that these self-loops do not contribute to the Laplacian.

Example 7.5.1. The Laplacian Matrix for 5 Mathematician Network

If we work with the adjacency matrix for the network of five Mathematicians we have for this simple network that



$$\mathsf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}, \quad \Rightarrow \quad \mathsf{L} = \begin{pmatrix} 3 & -1 & -1 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{pmatrix}. \tag{7.73}$$

The network is undirected so these matrices are both symmetric and their eigenvalues are real. The row sum of the Laplacian is always zero and the symmetric nature ensures the column sums are also zero. The smallest eigenvalue is zero as expected. Out-strength of this unweighted undirected network is just the degree and the largest degree is four. This, as we will see below, puts an upper limit on the modulus of eigenvalues of the Laplacian of 8.0.

n	1	2	3	4	5
λ	0.00	1.59	3.00	4.41	5.00
$\mathbf{v}_1^{(n)}$	-0.45	-0.27	0.50	-0.65	-0.22
$\mathbf{v}_2^{(n)}$	-0.45	-0.65	-0.50	0.27	-0.22
$\mathbf{v}_3^{(n)}$	-0.45	-0.00	0.00	0.00	0.89
$\mathbf{v}_4^{(n)}$	-0.45	0.27	0.50	0.65	-0.22
$\mathbf{v}_5^{(n)}$	-0.45	0.65	-0.50	-0.27	-0.22

(7.74)

Properties of the Network Laplacian

The Laplacian is not a non-negative matrix. However the eigenvalues and eigenvectors have a simple relationship to those of closely related matrix M which is non-negative. So consider the positive matrix $\mathsf{M} = \mu\mathbb{1} - \mathsf{L}$ where μ is bigger than the largest out-strength, $\mu > \max_i s_i^{(\text{out})}$. We can apply Perron-Frobenius to this M matrix. At the same time, the eigenvectors of M and A are identical, the only difference is that for a given eigenvector $\mathbf{v}$ we have that the eigenvalues satisfy a simple relationship (similarly for the left eigenvectors)

$$\mathsf{L}\mathbf{v} = \lambda\mathbf{v}\,, \qquad \Rightarrow \qquad \mathsf{M}\mathbf{v} = \lambda'\mathbf{v}\,, \qquad \lambda' = \mu - \lambda\,. \tag{7.75}$$

We know from the definition of the Laplacian that all its rows sum to zero. This means that the rows of M sum to μ so that the largest eigenvalue of M is $\lambda'_1 = \mu$. Thus the *smallest* eigenvalue of the Laplacian must be zero, $\lambda_1 = \mu - \lambda'_1 = 0$. It is also unique for a strongly connected network. It is important to note then that the Laplacian does not have an inverse, as the existence of a zero eigenvalue indicates. Note that we label the Laplacian eigenvalues from the smallest upwards, $0 = \lambda_1 < |\lambda_j|$ for $j => 1$.

In the same way, the lower bound on the modulus of the eigenvalues of M is zero, so the upper bound on the modulus of eigenvalues of L is[9] $2\max_i(s_i^{(\text{out})})$. We may also deduce that all the remaining eigenvectors of both M and L sum to zero.

By inspection, as shown in (7.71), you find that right eigenvector associated with the zero eigenvalue is proportional to a vectors of ones $\mathbf{1}$. It then follows from orthogonality of left- and right-eigenvectors, that the left eigenvectors for the non-zero eigenvalues, sum to zero.

$$\sum_i u_i^{(1)} v_i^{(n)} =\propto \sum_i v_i^{(n)} = 0 \text{ if } n \neq 0\,. \tag{7.76}$$

So for strongly connected networks (irreducible adjacency matrix), the Laplacian has the following properties.

- The smallest eigenvalue is $\lambda_1 = 0$.
- The Laplacian does **not** have an inverse.
- There is a single left-eigenvector $\mathbf{u}^{(1)}$ for $\lambda_1 = 0$ which is constant $u_i^{(1)} \propto 1$ (from (7.71)).
- The right-eigenvectors associated with non-zero eigenvalues, $\mathbf{u}^{(n)}$ for $n > 1$, sum to zero $\sum_i v_i^{(n)} = 0$ (from (7.76)).

[9] More generally the eigenvalues of the Laplacian must lie in a circle in the complex plane, with centre at the real number $\max_i s_i^{(\text{out})}$ and radius also $\max_i s_i^{(\text{out})}$.

- All eigenvalues are in or on a circle in the complex plane centred at $\max_i s_i^{(\text{out})}$ and of radius $\max_i s_i^{(\text{out})}$.

Note that the conservation of the number of particles is equivalent to the equation for the left-eigenvector, $\mathbf{u}^{(1)}$, associated with eigenvalue 0 which is indeed a constant vector.

$$\mathbf{u}^{(1)}\mathsf{L} = \lambda_1 \mathbf{u}^{(1)} = 0\,, \tag{7.77}$$

Given the definition of the process in (7.67), summing both sides i.e. contracting from the left by a constant vector $\mathbf{1}$ and using (7.71), gives

$$\frac{\partial W(t)}{\partial t} = -D\mathbf{1}\mathsf{L}\mathbf{w}(t) = 0\,. \tag{7.78}$$

Properties of the Network Laplacian for Undirected Graphs

The Laplacian has a number of nice properties when the graph is undirected, even if the edges are weighted, i.e. when the adjacency matrix is symmetric. The simple graph is the obvious example of this

Also useful is to note that if the underlying network is undirected then L and M are symmetric real matrices then standard linear algebra properties tell us that the Laplacian now has real eigenvalues and we can choose the left- and right-eigenvectors to be identical and real.

This reality property leads to an interesting geometric representation of the Laplacian of a simple network. Consider $x_n(i) = \sqrt{\lambda_n} u_i^{(n)}/|\mathbf{u}^{(n)}|$ for $n > 1$ where $|\mathbf{u}^{(n)}|^2 = \mathbf{u}^{(n)}.\mathbf{u}^{(n)}$ is the usual normalisation of a vector. We may consider these $\mathbf{x}(i)$ to be vectors in an $(N-1)$ Euclidean space, each one giving a point in this space representing one of the N nodes i of the network. It turns out that these points define a SIMPLEX with some remarkable properties, all of which come from the special properties fo the eigenvectors of the Laplacian. For instance the centre of mass of these points (imagining them all to be of equal weight) is the origin in this space. That follows as the sum of entries of each eigenvector is zero (from (7.76)). For more information on this construction see [Devriendt and Mieghem, 2019].

Example 7.5.2. The Laplacian Matrix and Simplex Representation for a line of three.

Consider a line of three nodes connected to their nearest neighbours. The adjacency matrix and Laplacian are simply

$$\mathsf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}\,, \quad \Rightarrow \quad \mathsf{L} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \tag{7.79}$$

The eigenvalues and eigenvectors, labelled in the standard way, can be written as

n	3	2	1
λ	3	1	0
$\mathbf{v}_0^{(n)}$	-1	-1	1
$\mathbf{v}_1^{(n)}$	2	0	1
$\mathbf{v}_2^{(n)}$	-1	1	1

(7.80)

As noted, the eigenvectors and eigenvalues in this case are all real (or may be chosen to be so) and are orthogonal. The lowest eigenvalue is zero and has a constant eigenvector while the others have components which sum to zero.

Normalising the two non-trivial vectors and scaling by the square root of the corresponding eigenvector we we find the coordinates (x, y) for the vertices of the simplex equivalent to this graph, namely

node	x	y
0	$-1/\sqrt{2}$	$-1/\sqrt{2}$
1	$\sqrt{2}$	0
2	$-1/\sqrt{2}$	$1/\sqrt{2}$

(7.81)

The Laplacian matrix is in fact the outer product, the Gram matrix, of these two vectors $L_{ij} = x(i)x(j) + y(i)y(j)$.

This illustrates several of the general results linking the geometric properties of this simplex to graph properties. For instance the length squared of each vector, the square of the distance of the point to the centre of mass and the origin of the simplex, is the the degree of the corresponding node of the graph. That is since the length squared of the vector representing vertex i is $(x(i))^2 + (y(i))^2$, and this is the diagonal entry of the Gram matrix, i.e. L_{ii}, we know that that is by definition the degree of vertex i.

The Network Laplacian and Continuum Diffusion

Consider the standard diffusion equation for the density $\rho(t, \mathbf{x})$ of some particles at position $\mathbf{x} \in \mathbb{R}^d$ in some d-dimensional space at time t

$$\frac{\partial \rho(t, \mathbf{x})}{\partial t} = -D\nabla^2 \rho(t, \mathbf{x}) \tag{7.82}$$

where $\nabla^2 = \partial^2/\partial x^2 + \ldots$ is the usual (spatial) Laplacian.

Figure 7.8: Continuum diffusion picture

If we use the simplest approximation for a second order differential we may write

$$\frac{\partial^2 f(x)}{\partial x^2} \approx \frac{f(x+a) - 2f(x-a) + f(x-a)}{a^2} + O(a^2)\,, \tag{7.83}$$

where a is taken to be a small separation in space. Suppose now we approximate our continuous space by a d-dimensional square lattice. For any vertex i in this lattice, there are $2d$ edges between nearest neighbour sites and i and the positions of these neighbours relative to vertex i may be specified by $2d$ vectors $\{\mathbf{e}(n)\}$ $(n = 1, 2, , \ldots, 2d\})$. We may then write that the continuous Laplacian in our diffusion equation (7.82) is approximated by

$$\nabla^2 f(\mathbf{x}) \;\approx\; a^{-2}\left[\left(\sum_{\mathbf{e}(n)} f(\mathbf{x} + \mathbf{e}(n))\right) - 2d\, f(\mathbf{x})\right]. \tag{7.84}$$

Writing $f(\mathbf{x}) = f_i$ for the value of function f at the i-th lattice points, and setting $A_{ij} = 1$ $(= 0)$ if sites i and j are (not) neighbours, this action of the Laplacian on f is approximated by

$$\left[\nabla^2 f\right]_i \;\approx\; a^{-2}\left(\sum_j A_{ij} f_j - 2d\, f_i\right) \tag{7.85}$$

which we now see is just our network Laplacian on the network defined by our d-dimensional lattice

$$\left[\nabla^2 f\right]_i \;-\approx\; a^{-2}\sum_j L_{ij} f_j\,. \tag{7.86}$$

Here $L_{ij} = k\delta_{ij} - A_{ij}$ where we have a simple regular network, A_{ij} is one if i and j are connected vertices of the lattice and so the degree is a constant k for all vertices.

The simplest approximation for the diffusion equation obtained by placing it on a square spatial lattice is just

$$\frac{\partial \rho(t, \mathbf{x})}{\partial t} \;=\; -D\nabla^2 \rho(t, \mathbf{x}) \tag{7.87}$$

$$\frac{\partial w(t)_i}{\partial t} \;\approx\; -(Da^{-2})\sum_j L_{ij}\mathbf{w}(t)_j \tag{7.88}$$

where $\rho(t, \mathbf{x}) = w(t)_i$ when at the i-th lattice point.

So a broadcasting with conservation on a regular lattice is just the discrete network approximation for standard diffusion processes. The Network Laplacian for a regular lattice is just the discrete approximation of the continuum Laplacian ∇^2.

The is an extremely important caveat here. For a regular lattice, in the expressions for the diffusion process above, we can divide the Laplacian by the constant degree k, absorbing a compensating factor of k into the diffusion constant D. The Laplacian matrix has the same form as (7.68) except the that the adjacency matrix A in (7.68) becomes the transition matrix of the PageRank/diffusion matrix T of (7.36). The out-strength term is trivially one in this case. Then we see that we also have equivalence between diffusion on the lattice and a Laplacian-like matrix built from the transition matrix T. This equivalence between the two processes and their Laplacians, broadcasting with conservation and the diffusion/PageRank case, breaks down when we have inhomogeneous networks, i.e. where the degree is not the same for all vertices. The link to diffusion on a regular lattice is useful to see why we call this matrix L of (7.68) the network Laplacian. However there is a much richer structure here for general networks. This we will pursue in Chapter 8.

Conserved Currents and Undirected Networks

If we have an undirected network then there is a second way to view this process, now in terms of the flow of electrical or fluid currents. For simplicity we will use the language of electrical currents here. Consider the equation for conserved broadcasting as written down in the form in (7.62). If we now exploit the case that $A_{ij} = A_{ji}$ we can rewrite this as

$$\begin{aligned} \frac{\partial w(t)_i}{\partial t} &= -D\left(\sum_j A_{ji}\right)\mathbf{w}(t)_i + D\left(\sum_j A_{ij}\mathbf{w}(t)_j\right) && (7.89)\\ &= -D\left(\sum_j A_{ij}\right)\mathbf{w}(t)_i + D\left(\sum_j A_{ij}\mathbf{w}(t)_j\right) && (7.90)\\ &= -D\sum_j A_{ij}\left(\mathbf{w}(t)_i - \mathbf{w}(t)_j\right). && (7.91) \end{aligned}$$

If we now interpret $\mathbf{w}(t)_i$ as a 'potential' with the net flow of particles along an undirected edge (i, j), the current in the undirected network case of conserved broadcasting process is seen to be proportional to the *potential difference* $\left(\mathbf{w}(t)_i - \mathbf{w}(t)_j\right)$. The product DA_{ij} is then the 'conductivity' of the edge. So in an electrical case, $\mathbf{w}(t)_i$ is the electrical potential, the voltage of node i, while the resistance of the connection (i, j) is $R_{ij} = 1/(DA_{ij})$. Our equation (7.59) denotes the change in the electrical charge at vertex i.

Note that this picture translates missing links in a sensible manner because resistance is the inverse of the edge weight. If there is no edge between vertex i and j in this network, it corresponds to an edge of zero weight. This translates into a link of zero conductivity, and so infinite resistance. No current will flow down such a link reflecting fact that we want no processes, no direct flows, to be happening directly between i and j.

Of course changes in a real circuit of pure resistors (no capacitors etc) will be nearly instantaneous. A more realistic case is to add in an external current representing a direct current $\mathbf{J}$ and assume we have reached equilibrium. The entry J_i represents an external current flowing into vertex i, with negative values representing net current leaving a vertex. For equilibrium, a stable DC circuit, we must have a balanced flow in and out of the circuit so $J = \sum_i J_i = 0$ is required. Given this we may write

$$J_i = -D\sum_j A_{ij}\left(\mathbf{w}(t)_i - \mathbf{w}(t)_j\right). \quad (7.92)$$

Now we might interpret the current flowing through each edge as a measure of the importance of that edge. Current will tend to flow through the routes of lowest resistance, typically the shortest

paths. However some current will flow down each possible path. This would mean we might use this to give edges a centrality measure. If we find the total current flowing through a vertex, we could give vertices a centrality measure based on this current picture, see [Newman, 2005]. Interestingly, the currents can be expressed in terms of a sum over all eigenvectors of the Laplacian *except* the leading eigenvector, the one with eigenvalue zero. That is it is the non-trivial eigenvectors which are controlling this measure of importance of edges and vertices based on current when we imagine an undirected network as a DC electrical circuit.

Aside Kirchhoff's matrix tree theorem

The link between the Laplacian and electrical circuits means that we should not be surprised to find a link with theorems on Laplacians produced by a physicist whose name is linked to the analysis of circuits, namely Kirchhoff. In graph theory Kirchhoff's matrix tree theorem is that the number of spanning trees of a graph is equal to any cofactor[a], that is the determinant of the matrix obtained by deleting any one row and column. The number of spanning trees is therefore equal to the product of the non-zero eigenvalues divided by the number of vertices. Indeed the Lapalcian is sometimes called the Kirchoff matrix

[a] Cofactors of the Laplacian are all equal as indicated by this theorem

Aside The Incidence Matrix

The Laplacian can also be written in terms of the incidence matrix E. The incidence matrix for a simple graph is defined by considering each edge $\alpha = (i, j) \in \mathcal{E}$. Then we define $E_{i\alpha} = 1$ and $E_{j\alpha} = -1$. We then find that the Laplacian can be written as $\mathsf{L} = \mathsf{E}.\mathsf{E}^{\mathsf{T}}$.

7.5.3 Discrete vs. Continuous Time Processes

We have described our broadcasting with conservation process in terms of a continuous time process, a derivative d/dt in our equation (7.59) whereas in previous processes we have considered discrete time steps, $t \to t+1$ such as in the broadcast process used for Katz centrality (7.5.3) or the full PageRank diffusion process (7.52). Discrete or continuous time makes relatively little difference mathematically in any of these cases. For the case of broadcasting with conservation, the Laplacian based process considered in this chapter, we used it here because it means the number of particles currently at a vertex, the number of particles can never go negative. That is in a small time interval Δt, we will lose approximately $Ds_i^{(\text{out})} w_i(t)\,\Delta t$ particles from a vertex i. In principle, for fixed Δt, with large enough D we can make this change bigger than $w_i(t)$ so $w_i(\Delta t)$ can become negative. That is not a disaster mathematically, but it means our analogy and interpretation using conserved particles breaks down. If the particles are a simple representation of some real property of a network, negative numbers $w_i(t)$ could be unphysical and this is a real issue. However, if this is just some mathematically useful game being used to capture some network properties, we might not be concerned. In fact, just as we had with the scaling factor α in our discussion of broadcast processes for the Katz centrality measure, , we can avoid negative $w_i(t)$ by keeping D small. So exactly as we found with Katz centrality, we can have a discrete time step, say $\Delta t = 1$, a broadcasting process with a conservation law, and for small enough D we can keep our $w_i(t)$ positive to keep our particle interpretation.

In this case our broadcasting with conservation process of (7.67) can be written as

$$\mathbf{w}(t+1) - \mathbf{w}(t) \quad = \quad -D\mathsf{L}\mathbf{w}(t)\,. \tag{7.93}$$

We have simply replaced $D^{-1}.dw_i(t)/dt$ by $(w_i(t+1) - w_i(t))/(D.\Delta t)$ with units of time chosen such that $\Delta t = 1$. This is a typical simple approximation used when implementing continuous time processes numerically on a computer. The analysis works much as before and builds on the eigenvectors of the Laplacian L. If we write

$$w(_)i(t) = \sum_n c_n(t)\mathbf{v}_i^{(n)} \tag{7.94}$$

then we see that

$$\sum_n (c_n(t+1) - c_n(t))\mathbf{v}_i^{(n)} = -D\sum_n c_n\lambda_n c_n(t)\mathbf{v}_i^{(n)}. \tag{7.95}$$

Using orthogonality with the left-eigenvectors we can project out the contribution from each eigenvector to see that

$$(c_n(t+1) - c_n(t)) = -Dc_n\lambda_n c_n(t) \tag{7.96}$$

$$c_n(t) = (1 - D\lambda_n)^t c_n(t=0) \tag{7.97}$$

$$\Rightarrow \quad w(_)i(t) = \sum_n c_n(t)(1 - D\lambda_n)^t \mathbf{v}_i^{(n)} \tag{7.98}$$

This makes it clear that we we need $|D\lambda_n| < 1$ to get a convergent process and this will also ensure that if $w_i(t=0) > 0$ then $w_i(t) > 0$ for all t.

So in the end, moving from continuous to discrete time has still left us with a system where the eigenvalues and eigenvectors of the Laplacian control the general solution. The evolution is now in terms of $(1 - D\lambda_n)^t$ rather than the factors of $(\exp(-D\lambda_n t)$ but the two are clearly linked.

Finally we note the reverse is also true. That is we can write down continuous time versions of the other processes we have considered, broadcasting and diffusion. However, the message is the same. The properties of the network control these linear processes and this is expressed in terms of the eigenvectors and eigenvalues of the relevant network matrix, be it the adjacency matrix, the transition matrix or the Laplacian.

Network Laplacian Summary

- Interpretation as Broadcasting with Conservation on a Network
- Definition of the Network Laplacian
- Key Properties of the Network Laplacian
- Describing Resistor Networks in terms of the Laplacian

Bibliography

Argent-Katwala, A., Evans, T. and Harder, U. [2007]. Exploration of the network spun by website users, *Proceedings of the 23rd Annual UK Performance Engineering Workshop (UKPEW 2007), Edge Hill University, UK, July 9th-10th, 2007*.

Barabási, A.-L. and Albert, R. [1999]. Emergence of scaling in random networks, *Science* **286**: 173.

Batool, K. and Niazi, M. A. [2014]. Towards a methodology for validation of centrality measures in complex networks, *PloS one* **9**(4): e90283.

Bavelas, A. [1950]. Communication patterns in task-oriented groups, *The Journal of the Acoustical Society of America* **22**(6): 725–730.

Bolland, J. M. [1988]. Sorting out centrality: An analysis of the performance of four centrality models in real and simulated networks, *Social networks* **10**(3): 233–253.

Borgatti, S. P., Mehra, A., Brass, D. J. and Labianca, G. [2009]. Network analysis in the social sciences, *Science* **323**(5916): 892–895.
URL: `http://www.sciencemag.org/content/323/5916/892.abstract`

Brandes, U. and Hildenbrand, J. [2014]. Smallest graphs with distinct singleton centers, *Network Science* **2**(03): 416–418.
URL: `http://dx.doi.org/10.1017/nws.2014.25`

Brandes, U. and Wagner, D. [2004]. Visone – analysis and visualization of social networks, *in* M. Jünger and P. Mutzel (eds), *Graph Drawing Software*, Springer-Verlag, pp. 321–340.

Brin, S. and Page, L. [1998]. The anatomy of a large-scale hypertextual web search engine, *Computer networks and ISDN systems* **30**(1-7): 107–117.

Clauset, A. [2013]. Network analysis and modeling lecture notes, *Technical report*.
URL: `http://tuvalu.santafe.edu/~aaronc/courses/5352/`

Clough, J. R., Gollings, J., Loach, T. V. and Evans, T. S. [2015]. Transitive reduction of citation networks, *Journal of Complex Networks* **3**: 189–203.

de Nooy, W., Mrvar, A. and Batagelj, V. [2005]. *Exploratory Social Network Analysis with Pajek*, Structural Analysis in the Social Sciences (No. 27), Cambridge University Press.

Devriendt, K. and Mieghem, P. V. [2019]. The simplex geometry of graphs, *Journal of Complex Networks* **7**(4): 469–490.

Deza, M. M. and Deza, E. [2009]. Encyclopedia of distances, *Encyclopedia of distances*, Springer, pp. 1–583.

Dorogovtsev, S. N., Mendes, J. F. F. and Samukhin, A. [2003]. Principles of statistical mechanics of uncorrelated random networks, *Nuclear Physics B* **666**(3): 396–416.

Eck, D. [2007]. *Introduction to programming using Java*, 5.0.2 edn, Hobart and William Smith Colleges, Department of mathematics and computer science.
URL: `http://math.hws.edu/javanotes/`

Erdős, P. and Réyni, A. [1959]. On random graphs. i, *Publicationes Mathematicae* **6**: 290–297.
URL: http://www.renyi.hu/~p_erdos/1959-11.pdf

Evans, T. [2004]. Complex networks, *Contemporary Physics* **45**(6): 455–474.
URL: http://uk.arXiv.org/abs/cond-mat/0405123

Fortunato, S. [2010]. Community detection in graphs, *Physics Reports* **486**(arXiv.org:0906.0612): 75—174.
URL: http://arXiv.org/abs/0906.0612

Freeman, L. C. [1977]. A set of measures of centrality based on betweenness, *Sociometry* pp. 35–41.

Freeman, L. C. [2004]. *The Development Of Social Network Analysis: A Study In The Sociology Of Science*, ΣP Empirical Press, Vancouver.
URL: https://www.researchgate.net/publication/239228599

Gastner, M. [2011]. Networks: Theory and application, *Technical report*, Imperial College London.
URL: http://www2.imperial.ac.uk/%7Emgastner/teaching/networks_autumn11/networks_autumn11.html

Granovetter, M. S. [1973]. The strength of weak ties, *American Journal of Sociology* **78**(6): 1360.
URL: http://www.journals.uchicago.edu/doi/abs/10.1086/225469

Guare, J. [1990]. *Six Degrees of Separation: A Play*, New York: Random House.

Kemeny, J. G. and Snell, J. L. [1960]. *Finite Markov Chains*, VanNostrand, Princeton.

Kivela, M., Arenas, A., Barthelemy, M., Gleeson, J. P., Moreno, Y. and Porter, M. A. [2014]. Multilayer networks, *Journal of Complex Networks* **2**(3): 203–271.

Lee, C.-Y. [2006]. Correlations among centrality measures in complex networks, *arXiv preprint physics/0605220* .

Lozares, C., López-Roldán, P., Bolibar, M. and Muntanyola, D. [2015]. The structure of global centrality measures, *International Journal of Social Research Methodology* **18**(2): 209–226.

Maslov, S. and Sneppen, K. [2002]. Specificity and stability in topology of protein networks, *Science* **296**: 910.
URL: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0205380

Merton, R. K. [1968]. The matthew effect in science: The reward and communication systems of science are considered, *Science* **159**(3810): 56–63.

Molloy, M. and Reed, B. [1995]. A critical point for random graphs with a given degree sequence, *Random Structures and Algorithms* **6**: 161–180.
URL: citeseer.ist.psu.edu/molloy95critical.html

Newman, M. E. J. [2005]. A measure of betweenness centrality based on random walks, *Social Networks* **27**: 39–54.

Newman, M. E. J., Strogatz, S. H. and Watts, D. J. [2001]. Random graphs with arbitrary degree distributions and their applications, *Phys. Rev. E* **64**(2): 026118.

O'Connor, J. J. and Robertson, E. F. [2017]. Mactutor history of mathematics archive.
URL: http://www-history.mcs.st-and.ac.uk/

Oldham, S., Fulcher, B., Parkes, L., Arnatkevičiūtė, A., Suo, C. and Fornito, A. [2019]. Consistency and differences between centrality measures across distinct classes of networks, *PLOS ONE* **14**(7): e0220061.

Price, D. J. d. S. [1965a]. The scientific foundations of science policy., *Nature* **206**: 233–238.

Price, D. S. [1965b]. Networks of scientific papers, *Science* **149**: 510–515.
URL: `http://garfield.library.upenn.edu/papers/pricenetworks1965.pdf`

Price, D. S. [1976]. A general theory of bibliometric and other cumulative advantage processes, *J.Amer.Soc.Inform.Sci.* **27**: 292–306.

Pržulj, N. [2007]. Biological network comparison using graphlet degree distribution, *Bioinformatics* **23**(2): e177–e183.
URL: `http://bioinformatics.oxfordjournals.org/content/23/2/e177.abstract`

Roli, A., Villani, M., Filisetti, A. and Serra, R. [2017]. Dynamical criticality: Overview and open questions, *Journal of Systems Science and Complexity* **31**(3): 647–663.

Rothenberg, R. B., Potterat, J. J., Woodhouse, D. E., Darrow, W. W., Muth, S. Q. and Klovdahl, A. S. [1995]. Choosing a centrality measure: epidemiologic correlates in the colorado springs study of social networks, *Social Networks* **17**(3-4): 273–297.

Schoch, D. [2015]. *A Positional Approach for Network Centrality*, PhD thesis, Universität Konstanz.

Schoch, D. [2016]. Periodic table of network centrality.
URL: `http://schochastics.net/sna/periodic.html`

Schoch, D., Valente, T. W. and Brandes, U. [2017]. Correlations among centrality indices and a class of uniquely ranked graphs, *Social Networks* **50**: 46–54.

Shen-Orr, S. S., Milo, R., Mangan, S. and Alon, U. [2002]. Network motifs in the transcriptional regulation network of escherichia coli, *Nature Genetics* **31**.

Simon, H. [1955]. On a class of skew distribution functions, *Biometrica* **42**: 425.

Solomonoff and Rapoport [1951].

Szell, M., Lambiotte, R. and Thurner, S. [2010]. Multirelational organization of large-scale social networks in an online world, *Proceedings of the National Academy of Sciences* **107**: 13636–13641.
URL: `http://www.pnas.org/content/early/2010/07/13/1004008107.abstract`

Szell, M. and Thurner, S. [2010]. Measuring social dynamics in a massive multiplayer online game, *Social Networks* **32**: 313–329.
URL: `http://www.sciencedirect.com/science/article/B6VD1-50H1HFB-1/2/cb75683a0ceb73ff6c729b40a0d8972c`

Tumminello, M., Aste, T., Di Matteo, T. and Mantegna, R. N. [2005]. A tool for filtering information in complex systems, *Proceedings of the National Academy of Sciences of the United States of America* **102**(30): 10421–10426.
URL: `http://www.pnas.org/content/102/30/10421.short`

Valente, T. W., Coronges, K., Lakon, C. and Costenbader, E. [2008]. How correlated are network centrality measures?, *Connections (Toronto, Ont.)* **28**(1): 16.
URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2875682/`

Wasserman, S. and Faust, K. [1994]. *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*, Cambridge University Press.

Watts, D. [2003]. *Six Degrees: The Science of a Connected Age*, W. W. Norton & Company.

Watts, D. J. and Strogatz, S. H. [1998]. Collective dynamics of 'small-world' networks., *Nature* **393**(6684): 440–442.
URL: `http://dx.doi.org/10.1038/30918`

Yilmaz, S., Dudkina, E., Bin, M., Crisostomi, E., Ferraro, P., Murray-Smith, R., Parisini, T., Stone, L. and Shorten, R. [n.d.]. A note on community-detection (kemeny) based testing for covid-19.

Yule, G. U. [1925]. A mathematical theory of evolution based on the conclusions of dr. j.c. willis, *F.R.S. Phil. Trans. B* **21-87**: 21–87.

Appendix A

Reading List

Optional background material of a less technical type. The books put the ideas in the course into a much broader context.

Caldarelli and Catanzaro 2012 Guido Caldarelli and Michele Catanzaro "Networks: A Very Short Introduction" (2012) Oxford University Press ISBN: 978-0199588077
[Pleasingly short. I have not read this but colleagues have recommended it to me. £5]

Ball 2004 Philip Ball "Critical Mass" (2004) ISBN: 9780099457862.
[General popular introduction to ideas about Complexity in general, including networks. Readable background material. £7]

Watts 2004 Duncan Watts "Six Degrees: The Science of a Connected Age" (2004) Vintage Press, ISBN: 978-0099444961.
[A non-technical overview focussing on networks. I found this very balanced even though written by one of the first authors to bring field to the attention of physicists. £9]

Evans 2004 Tim Evans "Complex Networks", Contemporary Physics **45** (2004) 455–474 [cond-mat/0405123].
[My own Complex Network review aimed as an introduction to general physics researchers. Only has the very basic technical issues making it short. I use this as a starting point for my undergraduate project students. Download from arxiv.org as http://arxiv.org/abs/cond-mat/0405123. Free.]

NetLit "Network Literacy: Essential Concepts And Core Ideas"
[Free. It is a very short 'poster' overview to the main concepts.]
https://sites.google.com/a/binghamton.edu/netscied/teaching-learning/network-concepts
See also What are essential concepts about networks?.

Technical and more detailed sources, with the most useful for this course given first:-

Clauset 2013 Aaron Clauset, "Network Analysis and Modeling lecture notes" (2013).
[Longer course so greater depth. Similar topics covered in a different order. Download notes from http://tuvalu.santafe.edu/~aaronc/courses/5352/. Free. Clauset's blog, Structure and Strangeness, and twitter feed, (@aaronclauset), are also informative.]

Newman 2011 Mark Newman, "Complex Networks", (2011) Oxford University Press.
[Very large book and comprehensive. Quite a lot of discussion which makes it large and less easy to use as a reference, but not a bad place to learn from unless you have to carry it around with you. £42.]

Easley and Kleinberg 2010 David Easley and Jon Kleinberg, "Networks, Crowds, and Markets: Reasoning About a Highly Connected World" (2010) Cambridge University Press, ISBN: 978-0521195331.
[Well written book. Focus of later parts is rather different from this course but the first sections on networks in general should be very useful. Download free copy from http://www.cs.cornell.edu/home/kleinber/networks-book/ or £32.]

Gastner 2011 Michael Gastner, "Networks: Theory and Application" (2011).
[Longer and more mathematical course given in Imperial Maths, Autumn 2011. General approach very similar and the mathematics but with effort should accessible to all Physics theory students. Good place for more general proofs of topics in this course. Download from Complexity and Networks Blackboard site. Free.]

Hanneman and Riddle 2005 Robert Hanneman and Mark Riddle, "Introduction to social network methods", (2005).
[Has a social science focus making the later parts less relevant to this course. However since it is free you might find the early sections useful. Download from `http://faculty.ucr.edu/~hanneman/`. Free.]

Menczer, Fortunato and Davis 2020 Filippo Menczer, Santo Fortunato, and Clayton A. Davis, "A First Course in Network Science", CUP, (2020), ISBN: `9781108471138`,
[I saw a copy of this book on the publisher's stand and it looks good and at the right level for the course and beyond. £35]

Latora, Nicosia and Russo 2017 Vito Latora, Vincenzo Nicosia, Giovanni Russo "Complex Networks: Principles, Methods and Applications" Cambridge University Press, (2017) ISBN: `9781107103184`.
[Looks well written with examples and exercises. Covers pretty standard ground in enough detail. So nothing to make it stand out but a very solid text book worth a look. £55]

Lewis 2009 Ted G. Lewis, "Network Science: Theory and Applications", Wiley, (2009) ISBN: `0470331887`.
[Has exercises. Looks comprehensive. Table 1.1 p2 for time line of papers as history of network science. £70].

Boccara 2010 Nino Boccara, "Modeling Complex Systems", Springer, (2010) ISBN: `1441965610`.
[General coverage of complex systems including chaos, recurrence equations. In particular has chapter 6 on spatial models including cellular automata and sociophysics models, chapter 7 on networks and chapter 8 on power laws e.g. power law vs log normal distributions. £63]

Caldarelli and Chessa 2016 Guido Caldarelli and Alessandro Chessa, "Data Science and Complex Networks", (2016), Oxford University Press, ISBN: `9780199639601`,
[Yet to read this in detail. Remarkably thin for the price (144 pages in total, closer to 120 for actual text) but looks quite useful. £40]

Steen 2010 Maarten van Steen, "Graph Theory and Complex Networks", (2010).
[Free download from web site. Style is rather different from this course but again general introductory sections on networks could be useful. Download from `http://www.distributed-systems.net/index.php?id=graph-theory-and-complex-networks`. Free electronic version or £15.]

Cohen and Havlin 2010 Reuven Cohen and Shlomo Havlin, "Complex Networks: Structure, Robustness and Function", Cambridge University Press, (2010) ISBN: `0521841569`.
[The text often seems a bit too bit brief. Some chapters are very short and feel like the transcript of a lecture rather than more extensive notes expanded from an hour long presentation. Good basic mathematics. See table 3.1 for nice list of properties of standard networks. Exercises. Smaller and lighter than Newman's book but no cheaper and less comprehensive. £42.]

Barabási 2016 A.-L. Barabási, "Network Science", Cambridge University Press, (2016), ISBN: `9781107076266`.
[Very nice layout and great graphics. Very readable but not as comprehensive as it at first seems —- there appears to be no section on centrality measures, and an odd selection of existing literature in some places. However the text is (and will remain) free online along with additional resources, see `http://barabasi.com/networksciencebook/`.
`http://barabasilab.neu.edu/networksciencebook/downlPDF.html`. Free to download, £42 hardback.]

Wasserman and Faust 1994 Stanley Wasserman and Katherine Faust, "Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)", Cambridge University Press, (1994), ISBN: 978-0521387071.
[This is a standard text written from the view point of social science, hence the title. It is, however, quite mathematical in places. It is a standard reference book for SNA as well as a textbook. I've never found it very readable or useful. I don't recommend it for beginners, but always felt that my dislike of the text is my problem, not the book's. £47]

Freeman 2004 Freeman, L. C., "The Development Of Social Network Analysis: A Study In The Sociology Of Science", ΣP Empirical Press, Vancouver, (2004), ISBN: 1-59457-714-5.
[A history of SNA (social network analysis) from one of the leading authors in the field. useful background for those deeply interested in network science. Note a free copy is available on ResearchGate.
https://www.researchgate.net/publication/239228599]

Appendix B

Mathematics

A Gamma Function

The GAMMA FUNCTION $\Gamma(x)$ plays a central role in many problems, and is particularly important when dealing with discrete quantities and DIFFERENCE EQUATIONS such as the Price/BA master equation (4.7). Search for the Gamma function online and you will find many useful properties e.g. the Gamma function entry on Wikipedia isn't bad (as with many core maths concepts) and is quite readable but it is always important to check Wikipedia's claims in which case MathWorld is reliable and authoritative but often not quite so easy to learn from.

The central property for the Gamma function, one that can be used to define the Gamma function, is that

$$\Gamma(z+1) = z\Gamma(z)\,, \qquad \Gamma(1) = 1\,, \tag{A1}$$

and this is for a complex number z. This means that for positive integer n we have[1]

$$\Gamma(n) \equiv (n-1)! \qquad \text{for } n \in \mathbb{Z}^+\,. \tag{A2}$$

Put another way, the Γ function is the *unique* analytic continuation of the factorial function[2] so it is the unique way to define a factorial for values other than positive integers.

Another property we will use is Stirling's approximation which in this language is the asymptotic expansion

$$\lim_{|z|\to\infty} \ln(\Gamma(z)) = z(\ln(z)-1) - \frac{1}{2}\ln(z/(2\pi) + O(z^{-1})\,. \tag{A3}$$

In particular, from this you can show that

$$\lim_{|z|\to\infty} \frac{\Gamma(z+1+a)}{\Gamma(z+1+b)} = z^{a-b}\,. \tag{A4}$$

For numerical work, it is important to note that since $\Gamma(z)$ gets very large very quickly as $z \to \infty$ for real z (it rising like a factorial). This quickly overwhelms the numerical ability of any computer. So it is often not a good idea to work with the Γ function directly in code. Instead you should use the logarithm of the Gamma function which, as Stirling's formula shows (A3), is much more reasonably behaved. For instance, to calculate a ratio of Γ functions numerically you should work numerically with the log of the ratio taking the exponential only at the end of the calculation. The ratio of gamma functions rises much slower than factorial as (A4) shows. So then what we use is

$$\frac{\Gamma(z+1+a)}{\Gamma(z+1+b)} = \exp\left(\texttt{gammaln}(z+1+a) \;-\; \texttt{gammaln}(z+1+b)\right) \tag{A5}$$

where `gammaln` is the natural logarithm of the Γ function, i.e. $\ln(\Gamma)$. There is always a library routine for this, for instance for python it is in `scipy` as `scipy.special.gammaln`. These `gammaln` give

[1] I prefer to reserve the factorial notation $n!$ for positive integers n only, but I'm unclear if that is a universal convention.

[2] Technically you still need another condition, usually some condition on the behaviour of the function as $n \to \infty$. After all the function $\exp(2\pi i z)\Gamma(z)$ also has the same value as $\Gamma(z)$ at integer values of z but this first form oscillates and behaves badly at infinity.

reasonable numerical values, their difference is also sensible, and we know the final result for this ratio is not a factorially large number, so the final exponential will work too.

Appendix C

Sets

A Mathematical Sets

We will use set notation as a convenient shorthand to save writing out long sentences rather than exploiting its power. If you are uneasy with this, just write out the meaning of the terms in words alongside the formal notation until you get used to the notation. For quick overview (sufficient for this course) the set theory definitions and notation on MathWorld are definitive while searching Wikipedia for relevant terms usually produces reliable information with a bit more description — I tend to use both as a check when looking up a topic. For a longer discussion I'd recommend [**??**].

A SET is a collection of *distinct* objects with no implied order. For example suppose we have the integers 1, 2 and 3. Then a set of these three is denoted $\mathcal{S} = \{1, 2, 3\}$ using braces (curly brackets). Note that even though we know a natural order for the integers, the set does not care about any order so $\mathcal{S} = \{1, 2, 3\} = \{3, 1, 2\} = \ldots$ all represent the same set.

Note that the set contains distinct elements. Thus the set of numbers in the traditional Fibonacci sequence, $f_{n+2} = f_{n+1} + f_n$, starting with $f_1 = 0$ and $f_2 = 1$ is $\{0, 1, 2, 3, 5, 8, \ldots\}$. In this Fibonacci sequence the number one appears twice as the second and third numbers in the sequence, $f_2 = f_3 = 1$, but a *set* of these Fibonnaci numbers can not have 1 repeated.

The number of elements in a set is denoted as $|\mathcal{S}|$ so if $\mathcal{S} = \{1, 2, 3\}$ then $|\mathcal{S}| = 3$.

The ELEMENTS or MEMBERS of a set are the objects in the set. Here our example has three elements, 1, 2 and 3. We show this with the $\in$ symbol which means "is a member of". Thus $1 \in \mathcal{S}$ means "1 is a member of the set $\mathcal{S}$".

A SUBSET $\mathcal{T}$ of a set $\mathcal{S}$ is denoted $\mathcal{T} \subseteq \mathcal{S}$ and the subset is a set made only of elements of the first set $\mathcal{T}$. That is if $v \in \mathcal{T}$ then $v \in \mathcal{S}$. Note that valid subsets include the empty set $\emptyset$ and the whole of the original set $\mathcal{T}$, so $\emptyset \subseteq \mathcal{T}$ and $\mathcal{T} \subseteq \mathcal{T}$.

A PROPER SUBSET $\mathcal{T}$ of a set $\mathcal{S}$ is denoted $\mathcal{T} \subset \mathcal{S}$. A proper subset $\mathcal{T}$ is any subset of $\mathcal{S}$ *except* for the whole set $\mathcal{S}$. Note that the empty subset is a proper subset[1] $\emptyset \subset \mathcal{S}$.

B Sets and Computer Languages

Many modern computer languages have some very useful ways of storing large amounts of information other than an array: lists or sets, ordered or not. These are known as COLLECTIONS in JAVA (see Eck "Introduction to Programming Using Java" chapter 10 [Eck, 2007]), CONTAINERS in the STL of C++ (Standard template Library) and simply data structures in PYTHON. In fact in python there is no array in the main language[2] and the whole python language is rooted in these more sophisticated ways of storing data.

In terms of our discussion of sets it is important to know that these advanced types of data storage always have a type which looks a bit like a set and in many cases is explicitly called a `set`. The `set` in java, `set` in python and the STL `set` of C++ are the same as the mathematical set. Each of these

[1] This is different from group theory where a proper subgroup excludes both the whole original group *and* the smallest group, the trivial group of one element.

[2] Arrays are part of packages added onto python such as the `numpy` package.

computing structures will only ever hold one reference of an object. If you try to add object A to these computing `set`s and they already contain an A, then nothing changes. This is made clearer when you discover that these languages contain other types of data structure in which an element, technically a reference to a single value/object, can appear more than once in this type of collection. The list objects in list in python or java do this job, but there is a specific `multiset` in the STL of C++.

The behaviour of these computer `set` structure mimics what we actually want for most network problems, e.g. a network is a set of vertices where no vertex appears twice.

Do a web search with the language name and the concept of interest. Sites I find useful are:-

-
- C++ reference site `http://www.cplusplus.com/`.
- http://docs.python.org `http://docs.python.org`.
- David Eck, "Introduction to Programming Using Java", `http://math.hws.edu/javanotes/`, [Eck, 2007].

Index