Network Analysis and Modeling — CSCI 5352
Lectures 1 and 2 — 27 and 29 August 2013
Prof. Aaron Clauset

1 Network basics

A *network* is a collection of vertices (or nodes or sites or actors) joined by edges (or links or bonds or ties). In mathematical jargon, networks are also called *graphs* and have been studied as mathematical objects for hundreds of years. Until the second half of the 20th century, most representations of empirical networks were small and largely confined to maps of social ties, constructed painstakingly by social scientists. Modern computers have now made it much easier to measure, store, draw and analyze the structure of extremely large networks. Arguably, the largest network currently studied is the World Wide Web, which contains tens of billions of nodes and likely trillions of links. ("Arguably" and "likely" because the WWW is so large that its structure is not known exactly.)

Because a network is logically equivalent to a graph, anything that can be represented as a set of discrete entities with pairwise[1] interactions can put in a network representation. For instance:

network	vertex	edge
Internet	computer	network protocol interaction
World Wide Web	web page	hyperlink
power grid	generating station or substation	transmission line
friendship network	person	friendship
metabolic network	metabolite	metabolic reaction
gene regulatory network	gene	regulatory effect
neural network	neuron	synapse
food web	species	predation or resource transfer

In some cases, the network representation is a close approximation of the underlying system's structure. In others, however, it's a stretch. For instance, in molecular signaling networks, some signals are conglomerations of several proteins, each of which can have its own independent signaling role. A network representation here would be a poor model because proteins can interact with other proteins either individually or in groups, and it's difficult to represent these different behaviors within a simple network.[2] In general, it's important to think carefully about how well a network representation captures the important underlying structure of a particular system, and how we might be misled if that representation is not very good.

[1]Higher-order interactions can also be defined, and networks of these are called *hypergraphs*. Examples include collaboration networks like actors appearing in a film, scientists coauthoring a paper, etc.

[2]Such a network could be represented using a mixed hypergraph, in which some edges are defined pairwise, while others are hyperedges of different orders, defined as interactions among sets of nodes.

1.1 Types of networks

Networks come in several flavors, largely based on the type of information the represent or the types of structures allowed/disallowed. Here, we will cover some of the most common types.

Simple graphs, multigraphs and self-loops.
A network is called *simple* if (i) there can be at most one edge (i, j) between any pair of vertices, i.e., edges are binary, (ii) there are no edges connecting a vertex to itself (a feature we call a *self-loop*), and (iii) a connection (i, j) implies a connection (j, i), i.e., edges are undirected. Figure 1a shows an example of a simple graph.

A *multigraph* relaxes the constraint on multiple connections (and generally also the constraint on self-loops). For instance, if vertices represent cities, and edges represent driving paths between a pair of cities, then a multigraph will be a reasonable representation because there can be several distinct such paths between a pair of cities. Similarly, in a network of neuron cells, two neurons can have multiple synapses and we might wish to represent each such connection as a distinct edge.

Weighted networks.
There are two types of weight information commonly used to annotate a network: edge weights and vertex attributes. A weight for an edge (i, j) is often denoted by a weight function $w(i, j)$ or by a scalar value w_{ij}. Edge weights are useful for representing attributes associated with the connection, e.g., interaction strength, frequency of interaction, capacity of interaction, etc., and are generally either a non-negative real number or natural number. Vertex attributes are simply values attached to the vertices, e.g., size, capacity, memory, etc. In social networks, common attributes are demographic values, e.g., age, sex, location, etc.

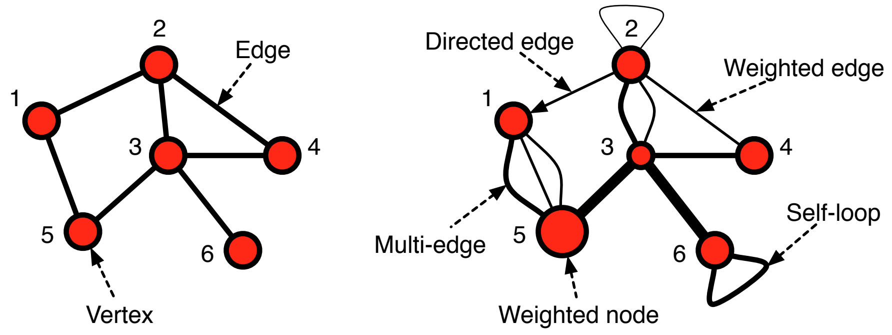

Figure 1: Examples of different types of edge and node structures. The left-hand network is an unweighted, undirected simple graph. The right-hand network is more exotic.

Directed networks.
A *directed network* allows asymmetric connections, i.e., connection (i, j) may exist independently of whether the connection (j, i) exists. We sometimes use the word "arc" to identify such a directed connection, rather than the more ambiguous term "edge." The World Wide Web is an example of a directed network, in which webpages are vertices, and hyperlinks are arcs or directed edges.

An *acyclic network* or graph is a special kind of directed network that contains no cycles, i.e., for all choices of i, j, if there exists a path $i \to \cdots \to j$ then there does not exist a path in the reverse direction $j \to \cdots \to i$. All trees are acyclic undirected networks. When we allow directionality in the edges, non-trees can be acyclic. For instance, a citation network, in which published papers are vertices and paper i connects to paper j if i cites j in its bibliography, is a kind of acyclic directed network (at least in theory; in practice, some cycles exist).

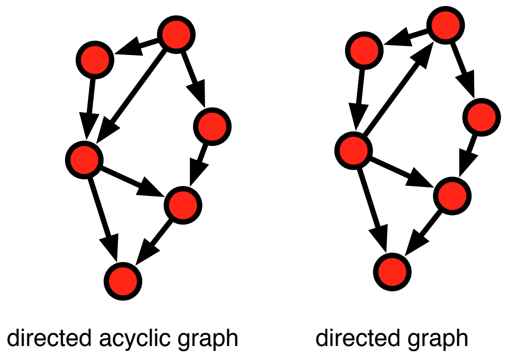

Figure 2: Examples of directed graphs.

Bipartite networks and one-mode projections.
If vertices can represent distinct classes of objects and only objects of different classes interact, i.e., edges cross between classes but not within classes, then we have a k-partite graph, where k is the number of classes. The simplest and most common form of such graph is the *bipartite* graph, with $k = 2$. One popular type of bipartite graph is the actor-film network, in which actors and films represent the two classes, and actors connect to the films in which they play a part.

Often, we wish to convert such a heterogeneous graph into a simple graph in which every vertex is of the same class, i.e., we want a *one-mode projection* of the bipartite graph. There are k one-mode projections for every k-partite graph. In a one-mode projection, two vertices are connected if they share a neighbor in the original graph. For instance, in the actor-film network, two actors would be connected if they ever acted in the same film together; or, two films would be connected if they have an actor in common.

One consequence of a one-mode projection is the construction of cliques, i.e., a subgraph of size ℓ in which every pair of nodes is connected. For instance, all actors in a particular film will be joined in a clique in the one-mode actor projection.

Temporal and dynamic networks.
The networks described so far are static, meaning that the vertices and edges do not change. Graphs that change over time are an important class of networks. For example, citation networks are dynamic networks, because new vertices join the network continuously and each time a new vertex joins, it creates new edges representing citations to papers in its bibliography.

There are two types of temporal networks, although these are not as different as they might sound.

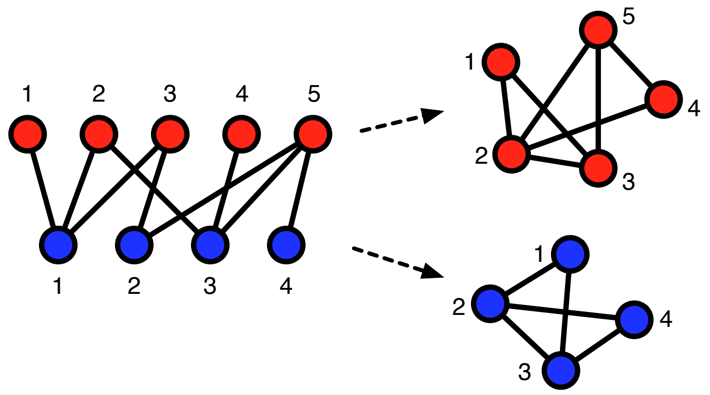

Figure 3: Example of a bipartite graph and its two one-mode projections.

In the first case, we convert some kind of underlying time-stamped interaction data into a sequence of network "snapshots" $A^{(t_1)}, A^{(t_2)}, \ldots$, where the sequence $t_1, t_2, \ldots$ represents a set of periods of time, e.g., hours in the day or days in a week. Within a particular snapshot, two vertices are connected if they ever interacted within the corresponding time period. An alternative representation of time-stamped interactions is to annotate each edge with the particular moment or span of time in which it occurred, e.g., (i, j, t_1, t_2) for an interaction starting at time t_1 and ending at time t_2.

Other types of networks.
There are, of course, many other types of networks. *Planar graphs* are those capable of being embedded in a 2d plane such that no edges cross. *Spatial networks* are empirical networks whose vertices have some position in space, commonly a position on the surface of the planet, e.g., road and city networks, airport transportation networks, oil and gas distribution networks, shipping networks, etc. These networks are often very close to, but not exactly, planar.

Hypergraphs are another form of network, in which edges denote the interaction of more than two vertices. *Multiplex networks* are a form of network in which multiple "layers," each of which may have a distinct edge set itself, represent different types of interactions between a common set of vertices. Crucially, there may be complicated dynamics on each vertex that govern which layer some kind of interaction occurs on, so multiplex networks are not merely a special kind of graph in which edges are annotated by different colors or layer numbers.

There are, of course, even more types of networks to be found in the literature, but these represent the most common types.

1.2 Representations of networks

There are three main ways to represent a network.

The first is an *adjacency matrix*, often denoted A, where

$$A_{ij} = \begin{cases} w_{ij} & \text{if } i \text{ and } j \text{ are connected} \\ 0 & \text{otherwise} \end{cases}$$

If A represents an *unweighted network*, then $w_{ij} = 1$ for all i, j.

An adjacency matrix representation of Fig. 1a is

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} .$$

Notice that the diagonal is all zeros and the non-zero entries are 0 or 1. This indicates that the corresponding network is a simple network. Also notice that the matrix is symmetric across the diagonal. Undirected networks have this structure because the upper triangle represents connections i, j while the lower triangle represents j, i.

Adjacency matrices are commonly used in mathematical expressions, e.g., when describing what a network algorithm does, but they are also sometimes used in algorithm's actual operation. The disadvantage of doing so, however, is a matrix always takes $O(n^2)$ memory to store, where n is the number of nodes in the network. Most empirical networks are *sparse*, meaning that the number of non-zero entries in the adjacency matrix is $O(n)$, and an adjacency matrix stores this small number of non-zero elements inefficiently. (Sparse-matrix data structures can circumvent this problem; these are essentially equivalent to our next representation.)

The second is an *adjacency list*, which stores only the non-zero elements of the adjacency matrix in a full list of all vertices. From a data-structures point of view, an adjacency list is like a hash table, in which each of the table entries corresponds to a vertex, and the elements we store in that table entry are a list of vertices reachable directly from that vertex, i.e., its out-going edges. Here is the adjacency list representation of Fig. 1a:

1	2	5		
2	3	1	4	
3	2	5	4	6
4	2	3		
5	1	3		
6	3			

Because our network is undirected, every edge (i, j) appears twice in the adjacency list, once as j in i's list, and once as i in j's list. The adjacencies for a given vertex can be stored as a linked list, or in a more efficient data structure, like a self-balancing binary tree (e.g., a red-black or a splay tree). This form of representation is essentially equivalent to a "sparse matrix," and is thus a common internal format for network data structures.

The third representation is an *edge list*, which stores only the non-zero elements of the adjacency matrix. Here's the undirected edge list representation of Fig. 1a:

$$\{(1,2),(1,5),(2,3),(2,4),(3,5),(3,6)\},$$

where it is assumed that all edges have unit weight and that the presence of an edge (i,j) implies an undirected tie between i,j.

This kind of structure is more typically used to store a network in a file on disk, possibly with additional information representing annotations for weighted edges [e.g., (i,j,w_{ij})], node annotations (like weights or attributes; usually stored in a file header), etc.

2 Structural measures of networks

Given network G, there are many quantities we could potentially measure as a way to characterize its structure. For instance, we could measure localized structures and then compute the average value or its distribution over the entire graph. Alternatively, we could define more global measures of structure, and anything in between. We'll briefly cover some of the conventional measures of structure, starting with the more local measures.

2.1 Degrees

The *degree* of a node k_i is simply a count of the number of connections terminating (equivalently: originating) at that node. Using the adjacency matrix, the degree of vertex i is defined as

$$k_i = \sum_{j=1}^{n} A_{ij} \quad , \tag{1}$$

which is equivalent to summing the ith column (or row) of the adjacency matrix A. If A represents a weighted network, this sum is called the node *strength*; the term "degree" is reserved for unweighted counts.

Every edge in an undirected network contributes twice to some degree (once for each endpoint or "stub"), and so the sum of all degrees in a network must be equal to twice the total number of edges in a network m:

$$m = \frac{1}{2}\sum_{i=1}^{n} k_i = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} = \sum_{i=1}^{n}\sum_{j=i}^{n} A_{ij} \quad . \tag{2}$$

And, the mean degree of a node $\langle k\rangle$ in the network is

$$\langle k\rangle = \frac{1}{n}\sum_{i=1}^{n} k_i = \frac{2m}{n} \quad . \tag{3}$$

If we divide the mean degree by its maximum value

$$\rho = \frac{2m}{\binom{n}{2}} = \frac{2m}{n(n-1)} = \frac{\langle k \rangle}{n-1} \ , \tag{4}$$

we have a quantity that is sometimes called the network's *connectance* or *density*.[3]

One of the more common uses of the degree is to tabulate a network's *degree distribution*, denoted $\Pr(k)$, which gives the probability that a vertex selected uniformly at random will have k neighbors. (This is distinct but related to the *degree sequence*, which is simply a list of the degrees of every node in a network.) The degree distribution for Fig. 1a is

k	$\Pr(k)$
1	1/6
2	3/6
3	1/6
4	1/6

where $\Pr(k) = 0$ for all other values of k.

In studies of empirical networks, the degree distribution is often used as a clue to determine what kinds of generative models to consider as explanations of the observed structural patterns. Generally, empirical social, biological and technological networks all exhibit right-skewed degree distributions, with a few nodes having very large degrees, many nodes having intermediate degrees, and a large number having small degrees.

2.2 Geodesic paths

A *path* in a network is a sequence of vertices $x \to y \to \cdots \to z$ such that each consecutive pair of vertices $i \to j$ is connected by an edge (i, j) in the network. A *geodesic* or *shortest path* is the shortest of all possible paths between two vertices, and these serve as the basis for a number of important measures of network structure.

The first and simplest such measure is the network *diameter*, which is the length of the longest of these geodesics and is meant to evoke the notion of a volume in a metric space.[4] Like many measures based on paths, measuring the diameter is done by running an algorithm that solves either the *All Pairs Shortest Paths* (APSP) problem or the *Single Source Shortest Paths* (SSSP) problem (n times, in the case of the diameter).

If the network is unweighted and undirected, a simple Breadth-First Search (BFS) tree will solve the SSSP problem,[5] providing us with the length of the path from a given source vertex i to all other

[3]Sometimes, connectance is defined as c/n, which is asymptotically equivalent to $c/(n-1)$.

[4]*Eularian* and *Hamiltonian* paths, which traverse every edge and every node exactly once, respectively are examples of other special kinds of paths. These, however, appear relatively infrequently in the study of networks.

[5]Or, run Bellman-Ford or Dijkstra's algorithm. All SSSP algorithms return a $n \times 1$ vector of distances from a single input vertex i to each of the other n vertices. By repeating this procedure for each vertex, we may construct the $n \times n$ pairwise distance matrix, one column at a time. BFS on an unweighted, undirected network takes $O(n+m)$ time, so the total running time is $O(n^2 + mn)$. For sparse graphs, this is a relatively fast $O(n^2)$ but becomes a slow $O(n^3)$ for dense graphs.

vertices reachable from i. For directed or weighted networks, we would instead use an algorithm like Floyd-Warshall or Johnson's algorithm.[6] Note that *reachability* is a crucial detail: if some vertex j is unreachable from i, then there is no geodesic path between i and j and the distance between them is either infinite or undefined. The diameter of a network is thus the diameter of its largest component (see Section 2.3 below).

Small worlds and network diameter.
In mathematical models of networks, the diameter plays a special role and can often be shown to vary in a clean functional way with the size of the network. If the diameter grows very slowly as a function of network size, e.g., $O(\log n)$, a network is said to exhibit the "small world" property.

The basic idea of "small world" networks comes from a seminal study in social networks by the American sociology Stanley Milgram (1933–1984).[7] Milgram mailed letters to "randomly selected" individuals in Omaha, Nebraska and Wichita, Kansas with instructions asking them to please pass the letter (and instructions) to a close friend of theirs who either knew or might be likely to know a particular doctor in Boston. Before doing so, they should also write their name on a roster to record the chain of message passing. Of the 64 letters that eventually reached the doctor—a small fraction of those sent out—the average length was only 5.5, not hundreds, and a legend was born.

Duncan Watts and Steve Strogatz, in a 1998 *Science* paper, studied this phenomenon using a toy model, now called the "small world model," in which vertices are arranged on a 1-dimentional circular lattice (a "ring" network) and connected with their k nearest neighbors. Each edge is then rewired, with probability p, to connect a uniformly random pair of vertices. At $p = 0$, the network is fully ordered, where the density of local connections is largest and the diameter is $O(n)$. At $p = 1$, the network is fully disordered, local connections are absent and the diameter is $O(\log n)$. See Figure 2 below.

The interesting behavior emerges as we dial p between these two extremes: when only a small number of edges have been randomly rewired (small p), the diameter of the network collapses from $O(n)$ to $O(\log n)$ while the local structure is still largely preserved. That is, a highly-ordered "big world" can be transformed, by rewiring only a small number of connections, into a still mostly ordered small world, in which geodesic paths traverse a vanishing fraction of the network. This behavior reminded them of some properties of social networks, which have small diameter (as evidenced by Milgram's study) but also many triangles.[8]

The small-world result is interesting for several reasons. First, it exemplifies an early effort to understand, using a toy model, how social networks can have both substantial local structure (mainly triangles) but also a small diameter. Second, it shows that some measures of network structure

[6]Floyd-Warshall takes $O(n^3)$ time in the worst case, which doesn't scale up to large networks ($n > 10^5$ or so)

[7]The term "six degree of separation" is not due to Milgram, but comes from a play written by John Guare in 1990. The play was subsequently made into a movie of the same name starring Will Smith in 1993, and, ironically, not starring Kevin Bacon.

[8]The small worlds model was generalized to d-dimensions by Jon Kleinberg in 2000, who further showed that under a particular distribution of the long-range links, greedy routing is highly efficient and takes only $O(\log^2 n)$ steps in expectation. Some of my own early work showed that this result can be achieved dynamically and adaptively using a simple rewiring rule. There's a potential independent project here, if anyone is interested.

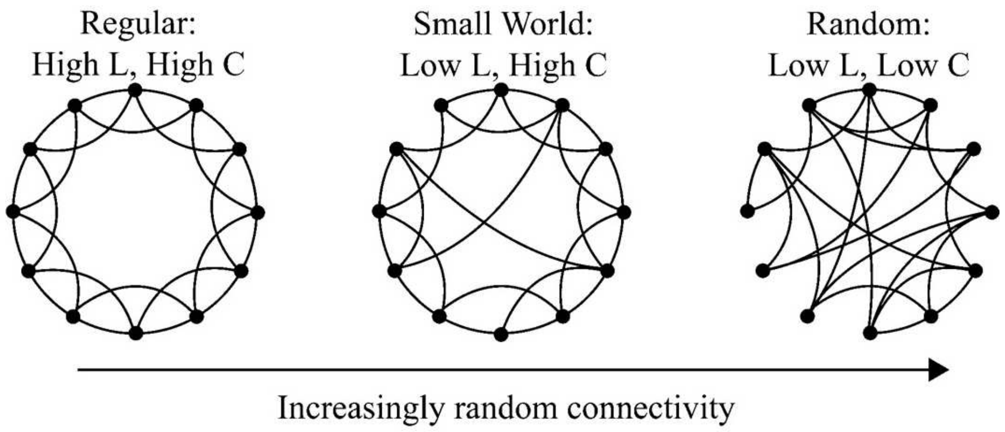

Figure 4: Watts and Strogatz's schematic illustrating the small worlds model.

can be extremely sensitive to uncertainty in the network structure. Suppose we obtained data generated by our toy model with an intermediate value of p, such that only a small number of edges had been rewired. If we observe each real edge with probability p, then it is possible that we will miss several of the long-range connections that cross the ring. The result would be a substantial overestimate of the diameter of the network, potentially confusing $O(n)$ with $O(\log n)$. Fortunately, not all measures of network structure are this sensitive to sampling errors, but it is well worth our time to consider the impact of the data generation and data observation processes when analyzing and modeling networks.

2.3 Components

If every pair of vertices is connected by some path, then the network is *connected.* If there is some pair of vertices between which no path exists, the network is said to be *disconnected*, i.e., it is composed of more than one *component.*

In an undirected graph, the set of vertices reachable from one vertex is called a *component* (and, for every vertex j reachable from i, i is also reachable from j). In a directed graph, reachability in one direction does not imply reachability in the other, and the notion of "connected" becomes more nuanced. A group of nodes that is pairwise reachable only if we ignore the direction of the edges is a *weakly connected component*, while a group of notes that is pairwise reachable if we obey the directions is a *strongly connected component.* Similarly, an *out component* is the set of vertices that can be reached from i, while an *in component* is the set of vertices that can reach i.

Many empirical networks are composed of multiple components, and among them, there is always a *largest component*, which is usually the component of greatest interest. In mathematical models, the *giant component* is also the largest component, but we add an additional property, which is to say that the size of the giant component must be $O(n)$.[9] We will revisit the notion of a giant component later in the semester, when we study models of random graphs.

Counting components.
To count and measure the size of the components within a network, we use any standard SSSP or APSP algorithm. For undirected networks, a breadth-first or depth-first search forest suffices, while for weighed or directed networks, Dijkstra's algorithm works well.

While the algorithm runs, we need only label all vertices that are pairwise reachable with the same vertex label, in an auxiliary array. When the algorithm terminates, we may make a single pass through the array to count the number of unique labels (the number of components) and count the number of times each label occurs (the sizes of each component). In a weighted graph, identifying strongly connected components is most easily done via a depth-first search forest, in $O(n + m)$ time.

3 At home

1. Peruse Chapters 1–5 (pages 15–104) in *Networks*
2. Read Chapter 6.1–6.12 (pages 109–149) in *Networks*
3. Next time: notions of centrality in networks

[9]The term "giant component" is meaningless in most empirical situations since it is only defined asymptotically. For empirical networks with multiple components, we instead focus on the largest component and reserve the term "giant component" for mathematical models.