Lecture 9

Network structure and properties
Networks can be generated in infinite many ways. The goal of the existing and (new) models is to achieve:

- a realistic mechanism;
- local interactions and moves;
- power law tails in the degree distributions with exponents $2 \leq \alpha \leq 3$;
- realistic hierarchies in the community structure;
- simple and light numerical implementations;
- achievable analytical solutions.
Fitness Model

It is reasonable to think that two vertices become connected when the edge creates a mutual benefit.

\[ f(x_i, x_j) \]

For every vertex \( i \) assign a “fitness”: a real number \( x_i \), drawn from a distribution \( \rho(x) \).
For every couple of vertices, \( i, j \), we can draw an edge with a probability given by the “fitness” linking function \( f(x_i, x_j) \).
If the network is not oriented the function \( f \) is symmetric i.e. \( f(x, y) = f(y, x) \).

In this model “the good get richer”.

A special case is the Erdős-Rényi random network model which corresponds to \( f(x,y) = const = p \).
The number of vertices $V$ can be either fixed (static model) or it can vary with time (dynamic model).

Vertex degree expectation value

$$k(x) = V \int \rho(y) f(x, y) dy$$

Vertex degree probability distribution

$$P(k) dk = \rho(x) dx$$

$$P(k(x)) = \frac{1}{dk(x)} \rho(x) dx$$

Vertex clustering coefficient

$$C(x) = V^2 \frac{\iiint f(x, z) f(y, z) f(x, y) \rho(y) \rho(z) dy dz}{k(x)^2}$$
Fitness Model III

A simple case is for:

\[ f(x, y) = \frac{x y}{x_M^2} \]

The degree of a vertex with fitness ‘x’ is

\[ k(x) = k_0 x \]

with

\[ k_0 = V \frac{\langle x \rangle}{x_M} \]

the degree distribution is

\[ P(k) = \frac{1}{k_0} \rho \left( \frac{k}{k_0} \right) \]

and the clustering coefficient becomes

\[ C = \frac{\langle x^2 \rangle^2}{x_M \langle x \rangle^2} \]

Clearly a fitness distribution with Zipf’s behavior:

\[ \rho(x) = c \ x^{-\alpha} \]

generates power laws degrees distributions

\[ P(k) = \frac{c}{k_0^{\alpha+1}} \ k^{-\alpha} \]
Thermodynamics of networks

Another possibility to generate graphs is to shape them according to some “cost” function $E$.

A starting configuration $A$ is chosen and its cost $E_A$ is computed.

Then we do a “move” in the graph, this results in a new graph $B$ with cost $E_B$.

If $E_B$ is smaller than $E_A$ we accept the change.

If $E_B$ is larger than $E_A$ we could accept the change with a probability (rate) $w(B \leftrightarrow A) = \exp[-\beta(E_B - E_A)]$.

After a while, this recursive procedure selects the configurations (ergodic set) with a given average cost fixed by $\beta$. 
Thermodynamics of networks II  -  Ensembles

This a very general thermodynamics method. It can be applied to any kind of ‘move’ from a graph to another.

However realistic moves must use only ‘local moves’ (readjustments that do not require the knowledge about the whole graph) and the changes in the cost must also be associated only to local information.

Let us for instance assume that these ‘moves’ concern the rewiring of edges.

We have two possibilities:
1) constant number of edges  ⇒  canonical ensemble
2) variable number of edges  ⇒  macro canonical ensemble
At each step of the graph evolution one end of an edge chosen at random is rewired toward another vertex with degree $k$ chosen with a ‘preference function’ $f(k)$

The equilibrium degree distribution must satisfy

$$ V \frac{dP(k,t)}{dt} = \frac{1}{\langle f(k) \rangle} \left[ f(k-1)P(k-1,t) - f(k)P(k,t) \right] + $$

$$ + \frac{1}{\langle k \rangle} \left[ (k+1)P(k+1,t) - kP(k,t) \right] $$

Statistical equilibrium

$$ f(k) = (k + 1) \frac{P(k + 1)}{P(k)} $$
Thermodynamics of networks IV - Statistical Equilibrium

\[ f(k) = (k + 1) \frac{P(k + 1)}{P(k)} \]

Scale free networks:

\[ f(k) \sim (k + 1) \frac{k^\alpha}{(k + 1)^\alpha} \sim k + 1 - \alpha + O(1/k) \]
We can image a method that assigns an edge between vertex ‘i’ and vertex ‘j’ according to a given ‘cost’ $e_{i,j}$. From the fitness model we can take

$$e_{i,j} = \frac{x_i x_j}{x_M^2}$$

The transition probability associated to adding or removing an edge is $w \propto \exp\left( \mp \beta e_{i,j} \right)$ and the average number of edges between two vertices is:

$$n_{i,j} = \frac{1}{\exp\left( \beta (e_{i,j} - \mu) \right) - 1}$$

However we might want to avoid multiple edges, constraining the occupation number to 0,1 only, obtaining:

$$n_{i,j} = \frac{1}{\exp\left( \beta (e_{i,j} - \mu) \right) + 1}$$
Optimized graphs

In some cases we aim to build a graph with a structure suitably designed in order to realize in an optimal way some specific functions. These classes of graphs are called “e\textit{conomic graphs}” in the language of graph theory. Trees are often good candidates as economic graphs, because in a tree we connect $V$ vertices with the minimum possible number of edges ($V - 1$). But sometimes there are other properties such as resilience to attacks, robustness to cascading failures or stability for random fluctuations that make the choice of more complex graphs best suited.
In the early 1930s Max Kleiber's observed that an animal's metabolic rate (B - number of calories an organism needs to survive) scales to the 3/4 power of the animal's mass (M).

\[ B \sim \text{Body Volume} \quad M \sim \text{Quantity of Blood} \]
Banavar et al. shows that the most efficient transportation network is a directed network and it has $M \sim L^{d+1}$.

Indeed, $M$ must be proportional to the number of edges ($E \sim L^d$) times the average distance from the nutrient source (longer the distance, larger the amount of blood to transport). Such a distance scales at least as $L$ (for all directed networks) and at most as $L^d$. 
The Kleiber relation $B \sim M^{d/d+1}$ must therefore hold for any optimized transport networks and should be valid for rivers as well as for living organisms.

\begin{align*}
W. J. Van Bloomestein Lake & \quad \text{Mississippi} & \quad \text{Rivers on Mars (?)}
\end{align*}

A 1655 nautical chart showing part of the Delaware River
Network Search

Algorithms for mining network crawl data.

Search engines make a local catalog of both the contents of web pages, and their hyper links. Then, when a query is made, the typical strategy would be to select a subset of pages from the database by matching for that query, and then to rank the results using the edge (link) information.

To make the database one ranks the pages in terms of “importance”: a page is important if it is pointed to by other “important” pages. Such “importance” can be assigned as:

$$x_i = \lambda^{-1} \sum_j A_{i,j} x_j$$

The “importance” \( (x_i) \) is the component of an eigenvector of the adjacency matrix. Provided the network is connected, there is only one eigenvector with all non-negative weights, which corresponds to the largest eigenvalue (Perron-Frobenius theorem) and it can be easily extracted by starting from a set of arbitrary weights and iterating the product:

\[
A^t \vec{X}_0 \propto \lambda^t \vec{X} \propto \vec{X}
\]

\( t \gg 1 \)

The graph’s eigenvector corresponding to the largest eigenvalue is the site ranking!
Can we build “maximally searchable graphs”?

We want to be able to navigate the network efficiently with local information only (greedy algorithm).

Let us put the vertices on a lattice and choose shortcuts between lattice vertices with probability

\[ p_s(i, j) \propto d_{i,j}^{-\alpha} \]

with \( d_{i,j} \) the distance between the vertices on the lattice.

Then, to find any target vertex from any random starting vertex will take a time bounded by \( \sim (\log V)^2 \) only when:

\[ \alpha = \text{space dimension} \]

Information structured graphs

Can we build graphs whose structure represents information?

Example: graphs from correlation matrices.

\[ c_{i,j} = \frac{\langle (x_i - \langle x_i \rangle)(x_j - \langle x_j \rangle) \rangle}{\sqrt{\langle (x_i - \langle x_i \rangle)^2 \rangle \langle (x_j - \langle x_j \rangle)^2 \rangle}} \]

16 Eurodollars interest rates

\[ \frac{N(N-1)}{2} \] different elements with a lot of redundant information

Information structured graphs - Minimum Spanning Tree

Given a connected, undirected graph, a spanning tree of that graph is a subgraph which is a tree and connects all the vertices.

In a weighted graph (a scalar weight $w_{i,j}$ assigned to each edge), a minimum spanning tree is a spanning tree ($\Gamma$) with total weight (sum of the weight of the edges belonging to the tree) $W(\Gamma) = \sum_{(i,j) \in \Gamma} w_{i,j}$ less than or equal to the weight of every other spanning tree.

$$W(\Gamma_{MST}) \leq W(\Gamma) \quad \text{for any } \Gamma$$
Information structured graphs - Minimum Spanning Tree

Correlation matrices: weights

\[ w_{i,j} = \sqrt{2(1 - c_{i,j})} \]

\[ c_{i,j} = \frac{\langle (x_i - \langle x_i \rangle)(x_j - \langle x_j \rangle) \rangle}{\sqrt{\langle (x_i - \langle x_i \rangle)^2 \rangle \langle (x_j - \langle x_j \rangle)^2 \rangle}} \]

1) Make an ordered list of edges ranking by increasing weights.
2) Take the first element in the list and add the edge.
3) Take the next element and add the edge if the resulting graph is still a tree.
4) Iterate the process from (3).

Additional material and questions
Question: is the network made by joining the triangles in contact in the Sierpinski gasket below a scale-free network? Which is the exponent?

Image a very dangerous viral outbreak.

It is known that the virus is propagating through social contacts.

We want to stop the epidemic BUT we have limited resources and we can immunize only a small fraction of the population.

Clearly, the good strategy is to immunize the most connected people (large betweenness).

Unfortunately we do not know the structure of the social network and we have no time to study it.

How can we efficiently reach the goal of vaccinating all the relevant social players without knowing a priori who are they?