Complex Networks

Introduction

Brief historical overview:

1736	Graph theory (Euler)
1937	Journal *Sociometry* founded
1959	Random graphs (Erdős-Rényi)
1967	Small-world (Milgram)
late 1990s	"Complex networks"

Complex Networks Research

Rapidly increasing interest over the last decade, since much more network data available now:

- Internet
- Biological networks, e.g.
  - Genetic networks
  - Food webs
- Social networks (e.g. Facebook, Twitter)
- Transport networks
- Mobile phone networks

Many show very similar features!

Describing a network formally

N nodes and E edges,

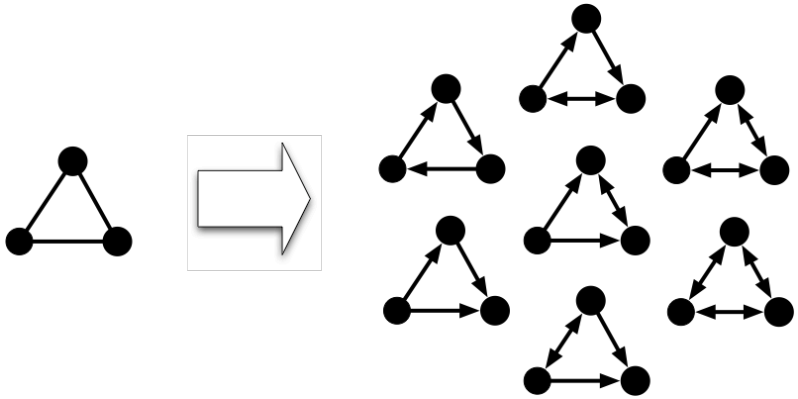
where $E \leq N(N\text{-}1)/2$

$N = 7, E = 9$

Note: In graph theory language this graph is of *order* 7 and *size* 9.

Directed networks

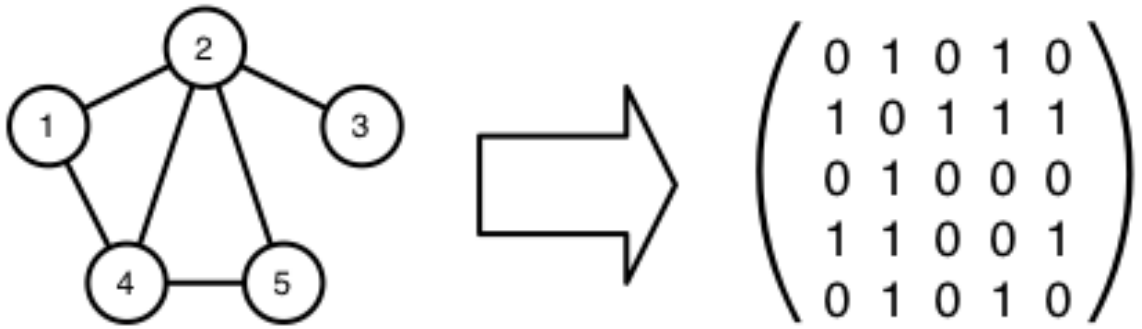
More edges: $E \leq N(N\text{-}1)$



Much more complex topology.

Adjacency matrix

The most convenient way of describing a network is the *adjacency matrix* a_{ij}.

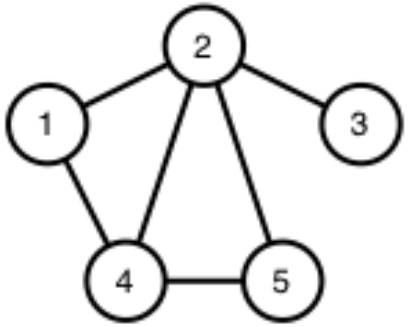


A link between node *i* to node *j* is recorded by a '1' in the *i*th row and the *j*th column.

Adjacency matrix

Undirected networks have a *symmetric* adjacency matrix

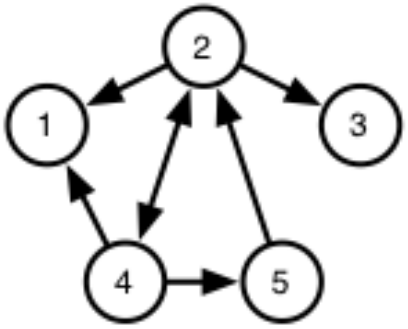
$a_{ij} = a_{ji}$.



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

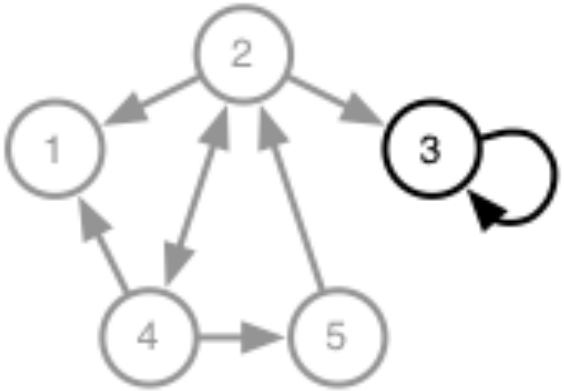
Directed networks in general have

asymmetric a_{ij}.



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

Self-interactions



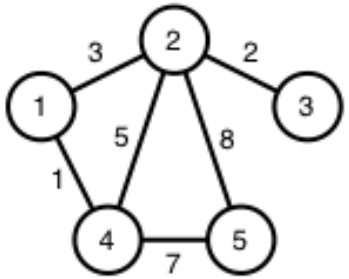
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Networks also can have *self-interactions*, which correspond to the diagonal entries a_{ii}.

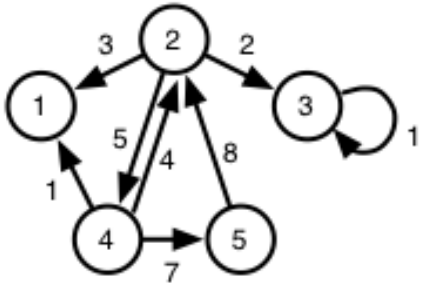
If we allow self-interactions, we can have up to $E = N^2$ edges.

Weighted networks

In a *weighted* network a real number is attached to each edge, so that we obtain a real adjacency matrix, usually denoted as w_{ij}.



$$\begin{pmatrix} 0 & 3 & 0 & 1 & 0 \\ 3 & 0 & 2 & 5 & 8 \\ 0 & 2 & 0 & 0 & 0 \\ 1 & 5 & 0 & 0 & 7 \\ 0 & 8 & 0 & 7 & 0 \end{pmatrix}$$



$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 2 & 5 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 4 & 0 & 0 & 7 \\ 0 & 8 & 0 & 0 & 0 \end{pmatrix}$$

Distance matrices

Something worth noting:

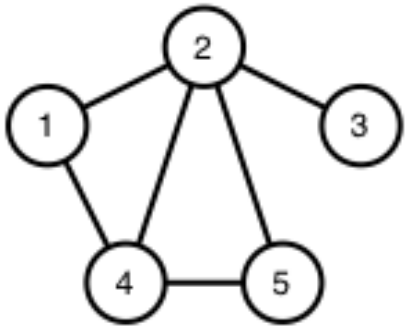
Define any distance measure on a set of objects.

This leads to a *distance matrix,* which is just the adjacency matrix of a fully connected weighted network.

Degree

In an undirected network the *degree* k_i of a node i is the number of nodes i is connected to:

$$k_i = \Sigma_j \, a_{ij} = \Sigma_j \, a_{ji}$$



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

Here $k_1 = 2$, $k_2 = 4$, $k_3 = 1$, $k_4 = 3$ and $k_5 = 2$.

In-degree and out-degree

In a directed network the *in-degree* $k_i^{(in)}$ of a node i is the number of directed edges pointing to node i:

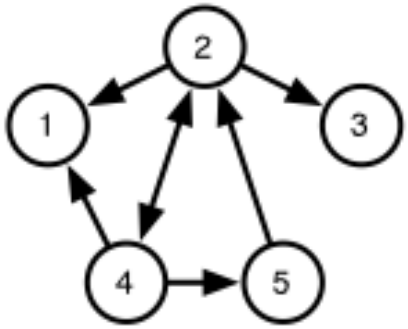
$$k_i^{(in)} = \Sigma_j \, a_{ji}$$

while the *out-degree* $k_i^{(out)}$ of a node i is the number of directed edges pointing from node i:

$$k_i^{(out)} = \Sigma_j \, a_{ij}$$

In-degree and out-degree

Thus, in a directed network, nodes can be highly connected, yet also isolated (e.g. in terms of sending or receiving information.)



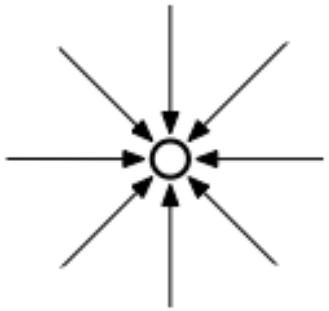
					out-degree:
0	0	0	0	0	**0**
1	0	1	1	0	**3**
0	0	0	0	0	**0**
1	1	0	0	1	**3**
0	1	0	0	0	**1**
in-degree:	**2**	**2**	**1**	**1**	**1**

Citations

The network of scientific citations provide examples illustrating two extremes:

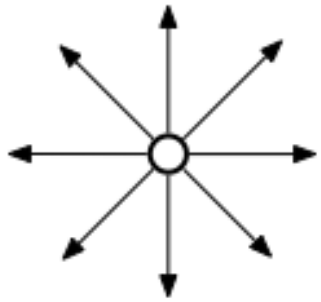
High in-degree and *low* out-degree:

much-cited research article



Low in-degree and *high* out-degree:

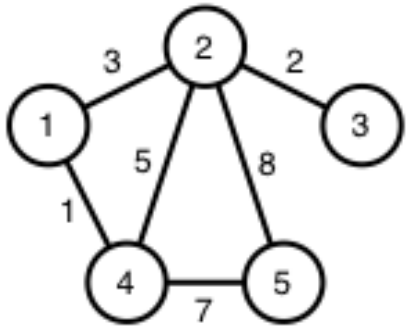
Book or review article



Strength

In a weighted, undirected network the *strength* is the sum of the weights for the edges connecting to a node:

$$s_i = \Sigma_j \, w_{ij} = \Sigma_j \, w_{ji}$$



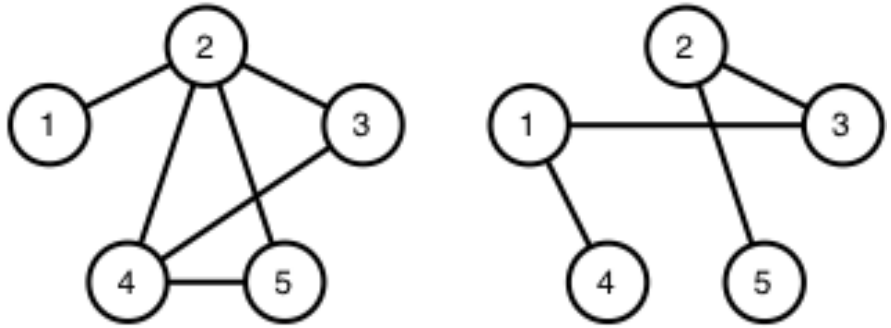
$$\begin{pmatrix} 0 & 3 & 0 & 1 & 0 \\ 3 & 0 & 2 & 5 & 8 \\ 0 & 2 & 0 & 0 & 0 \\ 1 & 5 & 0 & 0 & 7 \\ 0 & 8 & 0 & 7 & 0 \end{pmatrix}$$

Hence $s_1 = 4$, $s_2 = 18$, $s_3 = 2$, $s_4 = 13$ and $s_5 = 15$.

Erdős-Rényi networks

Random graphs studied by Paul Erdős and Alfred Rényi (1959):

Uniform probability p of two nodes i,j being connected.



Two different realizations for $N = 5$ and $p = 0.5$.

Erdős-Rényi networks

Some properties of E-R networks:

Average number of edges (= size of graph):

$$E = p\, N\, (N - 1)\, /\, 2$$

Average degree:

$$\langle k \rangle = 2\, E / N = p\, (N - 1) \simeq p\, N$$

Erdős-Rényi networks

The *degree distribution* P_k is a quantity of great interest in many networks, as we shall see later.

For E-R networks, in the limit of large N, it is given by:

$$P_k = \binom{N-1}{k} p^k (1-p)^{N-1-k}$$

Scale-Free networks

In a *scale-free* network

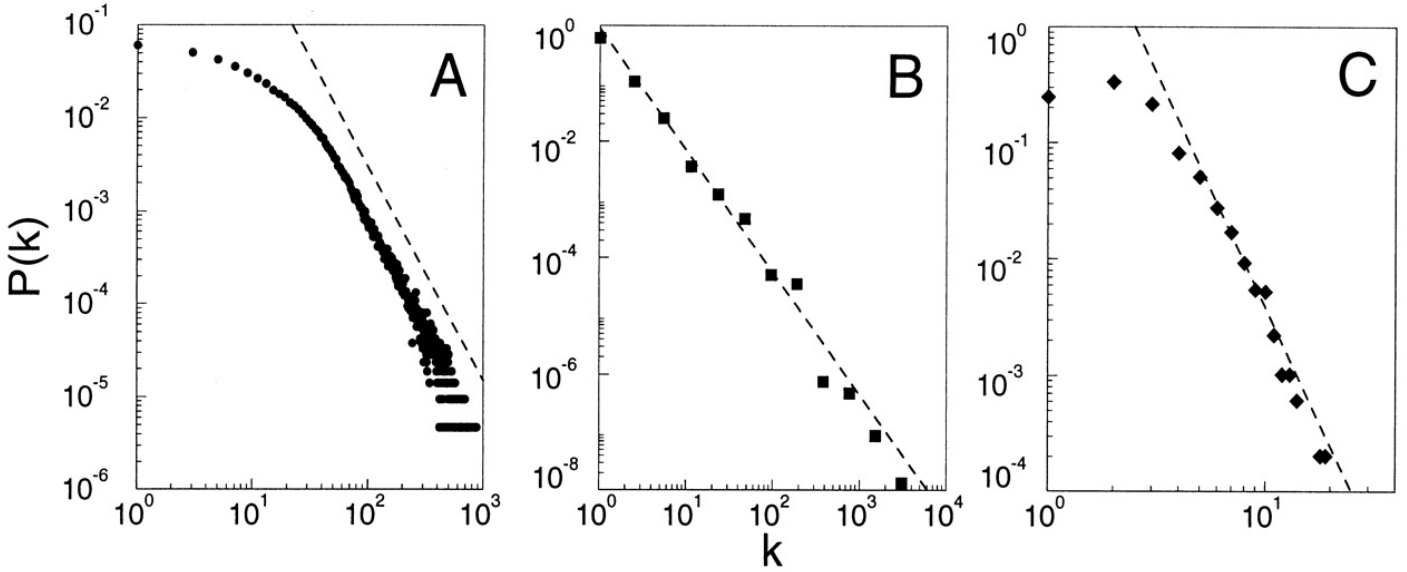
a) Many nodes have few connections and a few nodes have many connections.
b) This observation holds on the local and global scale of the network.

In other words, there is no inherent *scale*.

Scale-Free networks

Formally this translates into a *power-law* degree distribution:

$$P(k) = k^{-\gamma}$$



Examples: Actors, WWW, power grid

Image: Barabási and Albert, Science ***286****, 510 (1999)*

Scale-Free networks

Typical values of exponent γ observed:

Network	γ
Co-authorship	1.2
Internet	2.1
Yeast protein-protein	2.4
Word co-occurrence	2.7

Preferential attachment

Presented by Barabási & Albert [Science **286**, 509 (1999)]:

Probabilistic network growth model which produces scale-free networks.

Add new node and attach it to m existing nodes, where the probability of attaching it to a particular node i is:

$$p_i = k_i / \Sigma_j k_j$$

Preferential attachment

Nodes: $N = N_0 + t$

Edges: $E = m\ t$

Since one node and m edges are added per timestep.

What is the degree distribution for the B-A model?

Can get an answer by considering k as a continuous variable.

Preferential attachment

The variation of degree with time is given by:

$$\frac{\partial k_i}{\partial t} = mp_i = m\frac{k_i}{\sum_j k_j} = \frac{mk_i}{2mt} = \frac{k_i}{2t}$$

which for a node i joining at time t_i has the solution:

$$k_i(t) = m\sqrt{\frac{t}{t_i}}$$

Preferential attachment

By considering the probabilities:

$$P(k_i < k) = P(t_i > \frac{m^2 t}{k^2})$$

and given that at time t:

$$P(t_i > \frac{m^2 t}{k^2}) = 1 - P(t_i \leq \frac{m^2 t}{k^2})$$

$$= 1 - \int_0^{m^2 t / k^2} \frac{1}{N_0 + t} \, dt' = 1 - \frac{m^2 t}{k^2} \frac{1}{N_0 + t}$$

Preferential attachment

Hence we arrive at:

$$P(k) = \frac{\partial P(k_i < k)}{\partial k} = \frac{2m^2 t}{(N_0 + t)k^3}$$

which gives us a scale-free degree distribution with a power-law exponent of -3, in other words $\gamma = 3$.

Modified preferential attachment models lead to other γ values.

Arbitrary degree distributions

Newman et al. proposed a model to obtain random graphs with arbitrary degree distributions, by using a generating function approach.

$$G_0(x) = \Sigma_k \, p_k \, x^k$$

Phys. Rev. E 64, 026118 (2001)

Generating function approach

The generating function

$$G_0(x) = \Sigma_k \, p_k \, x^k$$

contains all information about the distribution of p_k, since

$$p_k = (1/k!) \, \mathrm{d}^k G_0 / \mathrm{d}x^k \, |_{x=0}$$

Generating function approach

Many properties of the network can be derived from this generating function, such as

- Average degree: $\langle k \rangle = \Sigma_k \, k \, p_k = G_0'(1)$

- Average number of second-nearest neighbours:

$$\langle k_{2\text{nd}} \rangle = G_0''(1)$$

(But this doesn't generalize simply)

- Clustering coefficient (we will come to this later)

Assortativity

Assortativity describes the correlation between the degree of a node and the degree of its neighbours.

Networks in which highly connected nodes are linked to other nodes with a high degree are termed *assortative*. Such networks include social networks.

Networks in which highly connected nodes are only linked to nodes with a low degree are termed *disassortative*. Such networks include the World Wide Web and biological networks.

Assortativity Coefficient

One way of measuring assortativity is to determine the Pearson correlation coefficient between the degrees of pairs of connected nodes. This is termed the assortativity coefficient r:

$$r = (1 / \sigma_q) \, \Sigma_{jk} \, jk \, (e_{jk} - q_j q_k)$$

where q_j and q_k are the distributions of the remaining degrees* j and k, when following a randomly selected edge. This can be defined in terms of the degree distribution p_k as:

$$q_k = \frac{(k+1) p_{k+1}}{\sum_j j p_j}$$

and where e_{jk} is the joint probability distribution of pairs of nodes with remaining degrees j and k at either end.

*remaining degree = degree minus one (the edge connecting the nodes)

Assortativity Coefficient

The value of r lies between:

-1 (disassortative) and 1 (assortative).

Some values for real networks:

Physics coauthorship:	0.363
Company directors:	0.276
Internet:	-0.189
Marine food web:	-0.247

Nearest-neighbour degree

The nearest neighbour degree k_{nn} of a node i is the average degree of the neighbours of i.

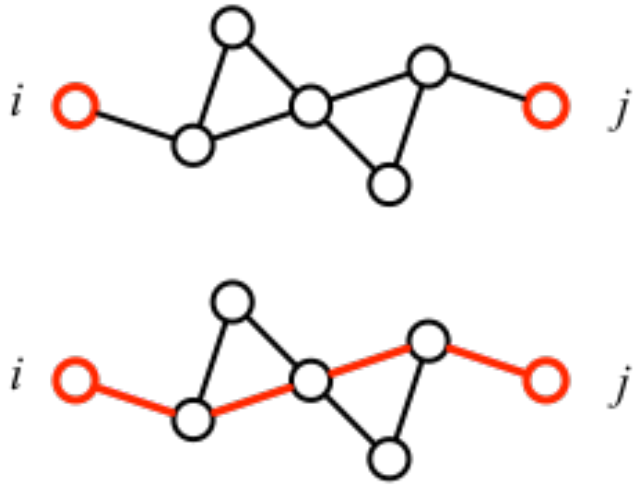
The average nearest neighbour degree $\langle k_{nn} \rangle$ is k_{nn} averaged over all nodes of the same degree k.

Assortativity can also be measured by plotting the average nearest neighbour degree $\langle k_{nn} \rangle$ as a function of the degree k.

An increasing slope indicates assortativity while a decreasing one signals disassortativity.

Distance

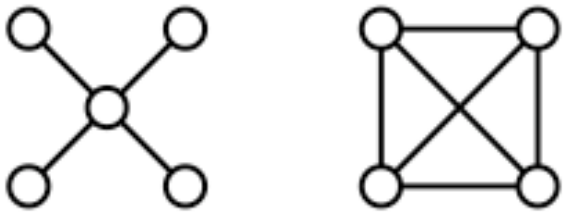
The *distance* between two nodes *i* and *j* is the shortest path connecting the two nodes.



$d_{ij} = 4$

Diameter

The *diameter* of a network is the largest distance in the network - in other words it is the maximum shortest path connecting any two nodes.



$D = 2$ $D = 1$

Note: Fully connected networks (like the one on the right) have diameter $D = 1$.

Clustering coefficient

The *clustering coefficient* measures how densely connected the neighbourhood of a node is.

It does this by counting the number of triangles of which a given node i is a part of, and dividing this value by the number of edge pairs.

$$c_i = [2 / k_i (k_i - 1)] \sum_{jk} a_{ij} a_{jk} a_{ik}$$

Often the clustering coefficient is averaged over the entire network:

$$C = (1 / N) \sum_{ijk} [2 / k_i (k_i - 1)] a_{ij} a_{jk} a_{ik}$$

Where N is the number of nodes.

Small-world networks

Watts and Strogatz (1998) consider a locally connected network and randomly rewire a small number of edges.

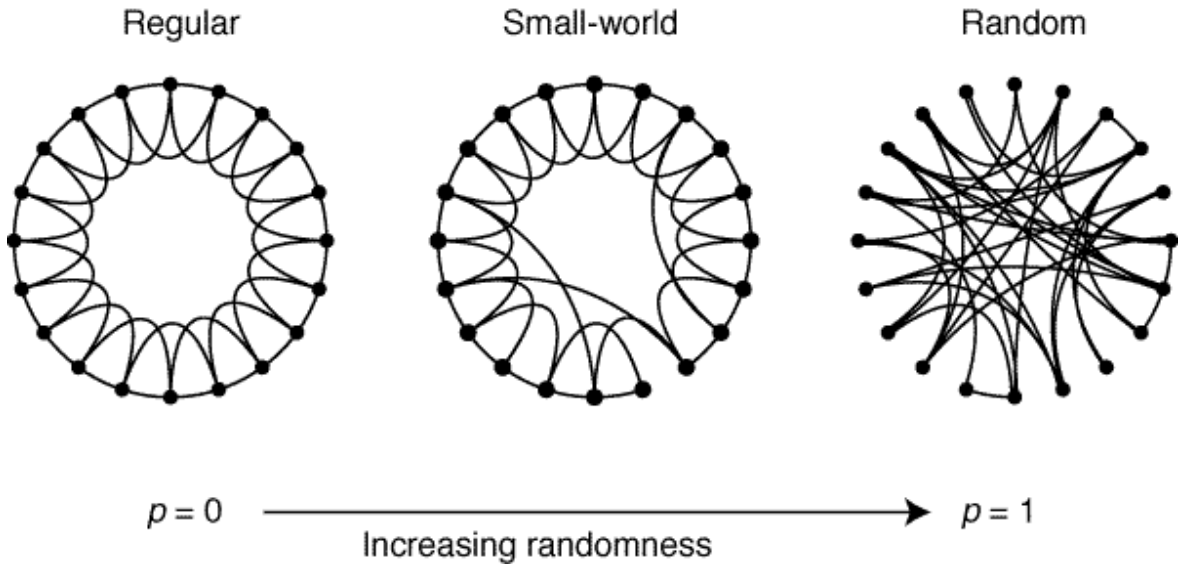


Image: Watts and Strogatz, Nature **393**, 440 (1998).

The probability of rewiring p 'tunes' the network between a regular lattice ($p = 0$) and a random (Erdős-Renyi) graph ($p = 1$).

Small-world networks

As the rewiring probability p increases, the average distance between two nodes falls drastically, while the clustering remains largely unchanged until p gets a lot larger.

$$L \geq L_{random}$$
$$C >> C_{random}$$

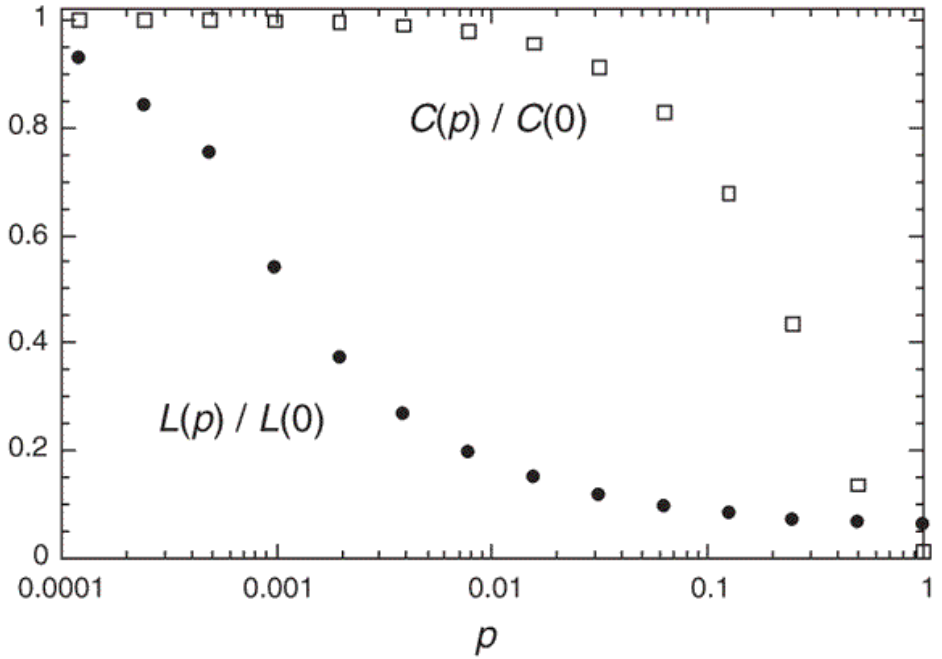


*Image: Watts and Strogatz, Nature **393**, 440 (1998)*

Small-world networks

Thus small-world networks are signified by small average distances, similar to random graphs, but much higher clustering coefficients than random graphs.

Such networks are termed *small-world*, in analogy to the "small-world phenomenon" which proposes that, roughly speaking, every person is connected to every other person by at most six connections.

The small-world property cannot be detected at the local level, as the random rewiring does not change the clustering coefficient.

Betweenness

The rather awkward word *betweenness* is a measure of the importance of a node or edge.

The most widely used is *shortest-path betweenness*:

Consider a pair of nodes, and all shortest paths between them. For any given edge or node in the network, we can determine the fraction of these shortest paths which pass through it. The shortest-path-betweenness is the sum of these fractions over all pairs.

Other forms include *random-walk betweenness* and *current-flow betweenness*.

Betweenness: an example

While betweenness of a given node or edge is calculated over *all* pairs of nodes, consider the contribution associated with one particular node (*s* below):

(a) In a tree, the betweenness is rather straightforward.

(b) In a network with loops, the betweenness becomes more complicated, e.g. $25/6 = 1 + 1 + 1 + 1/2 + 1/3 + 1/3$

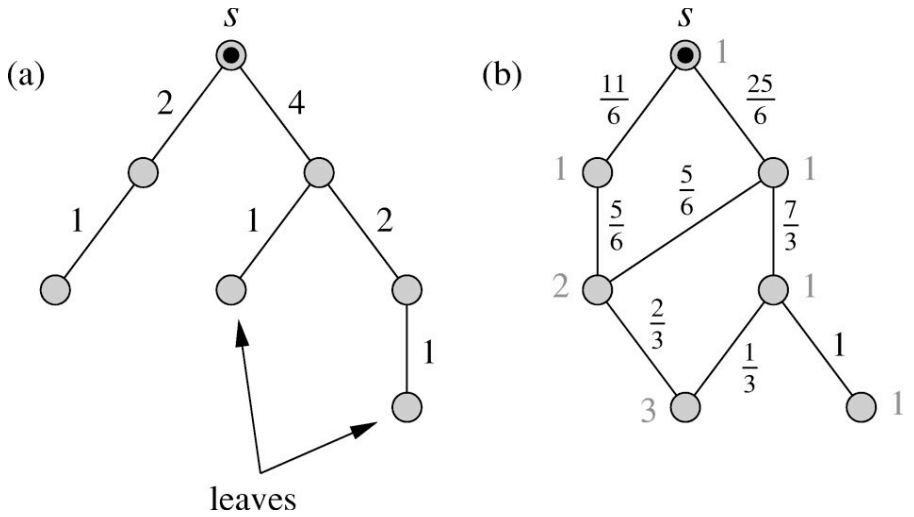
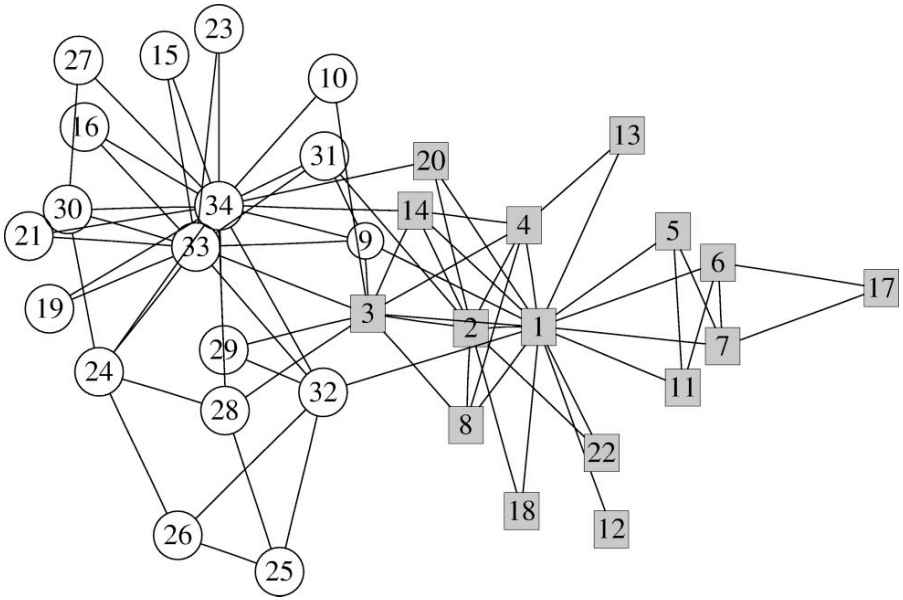


*Image: Newman and Girvan, PRE **69**, 026113 (2004)*

Community detection

Betweenness can help us to detect *communities* in networks.



Famous: The Zachary Karate Club network

*Image: Newman and Girvan, PRE **69**, 026113 (2004)*

Community detection

Newman and Girvan (2002) proposed a simple algorithm:

1) Calculate the betweenness of all edges in the network.

2) Remove the edge with the highest betweenness.

3) Recalculate the betweenness.

4) Continue at 2) until no edges are left.

The disconnected components of the network form a suggested partition into communities at each iteration.

Community detection

The network fragmentation achieved using this process suggests many possible partitions of the network into communities. Which one is the best one?

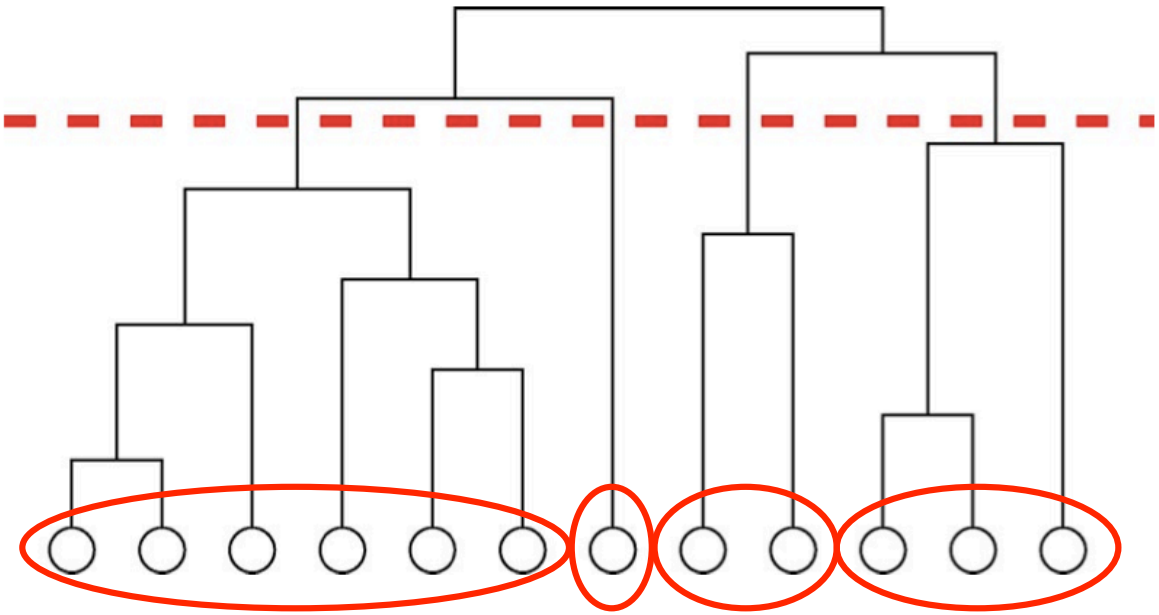


*Image: Newman and Girvan, PRE **69**, 026113 (2004)*

Community detection

The network fragmentation achieved using this process suggests many possible partitions of the network into communities. Which one is the best one?

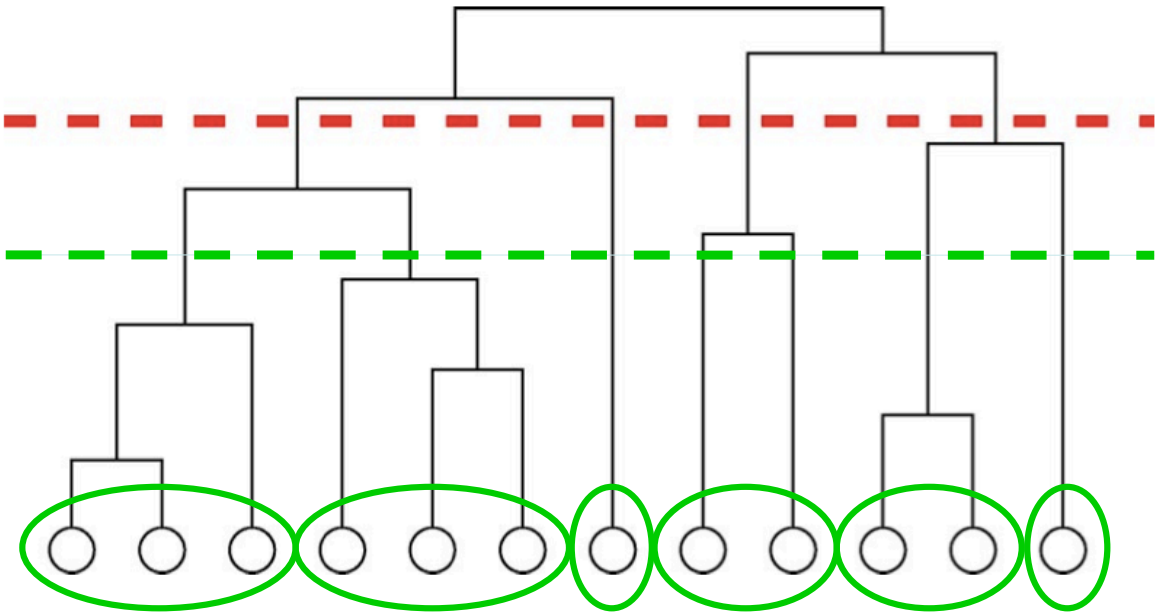


*Image: Newman and Girvan, PRE **69**, 026113 (2004)*

Modularity

The *modularity* of a network measures the *quality* of a given partition of the graph into sets S_i.

It does so by comparing the total number of connections *within* a set to the number of connections which would lie within this set *by chance*.

Given n_c sets, consider the $n_c \times n_c$ matrix e_{ij} which contains the *fraction of the total number of edges* which connect communities i and j.

Modularity

Thus the total fraction of edges connecting to nodes in set i is:

$$a_i = \Sigma_j \, e_{ij}$$

And if the edges were independent of the sets S_i, then the probability of an edge connecting two nodes within the same set would be

$$a_i^2 = (\Sigma_j \, e_{ij})^2$$

The actual fraction of edges internal to a set is e_{ii}, so that the summed difference of the two gives us a measure of *modularity*:

$$Q = \Sigma_i \, [e_{ii} - (\Sigma_j \, e_{ij})^2]$$

Using modularity

When using the betweenness-based Newman-Girvan algorithm to find communities, the modularity Q can be used to evaluate which partition is the most meaningful:

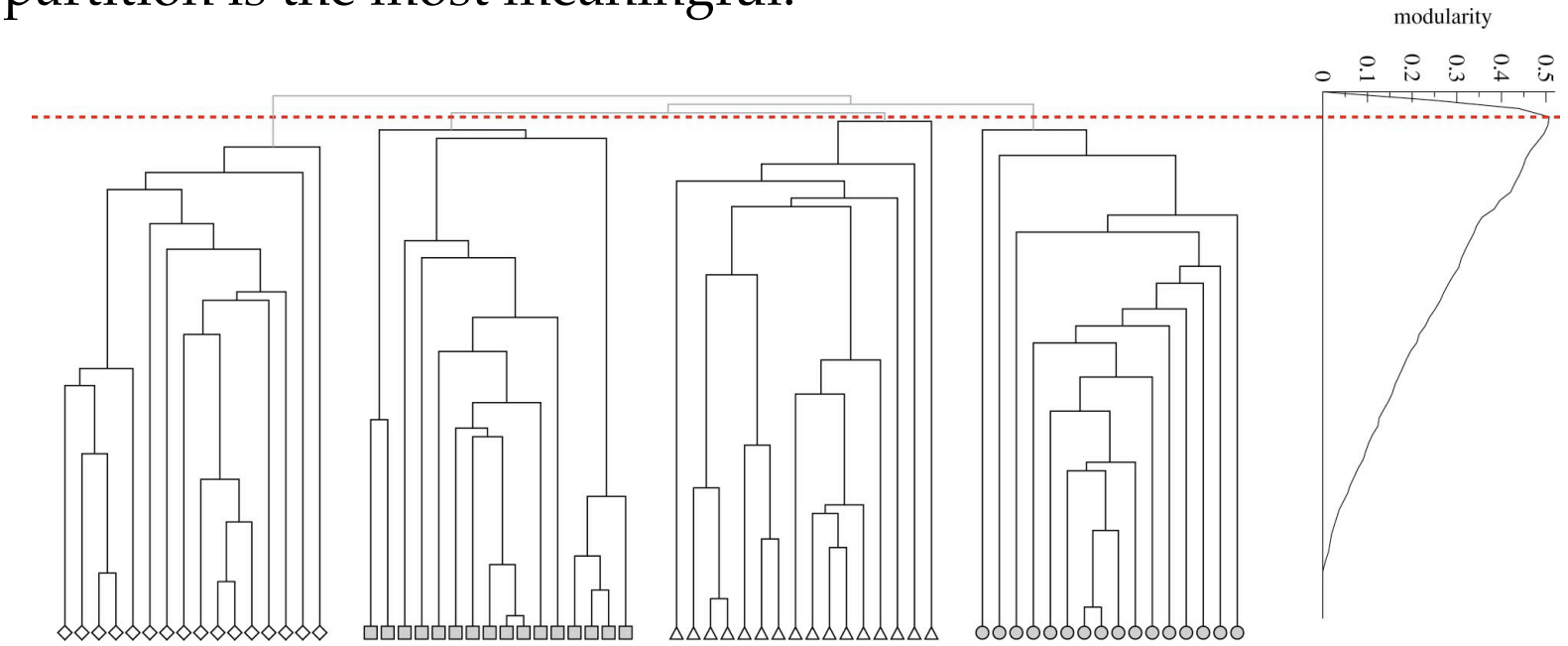


*Image: Newman and Girvan, PRE **69**, 026113 (2004)*

Network vulnerability

In many real-world networks it is of interest to measure the network's *vulnerability* to attacks or random failure.

One of the best-known results in this context is the observation that the degree of proteins in a protein-protein interaction networks is positively correlated with the *lethality* of the protein.

This implies that the highest-degree nodes are the ones whose removal would cause the most disruption.

Jeong et al., Nature **411**, 41 (2001).

Network vulnerability

Betweenness is also a useful measurement of the vulnerability of a network node or edge.

The removal of an edge or node with high betweenness is likely to disrupt the dynamics of flow across the network significantly.

In fact the strategy of removing nodes according to the Newman-Girvan algorithm is also one which damages the network very effectively (Holme *et al.*, 2002).

Network vulnerability

Scale-free networks are very *robust* against random removal of nodes, but very *vulnerable* to any targeted attacks.

Random graphs on the other hand are equally sensitive to both forms of disruption.

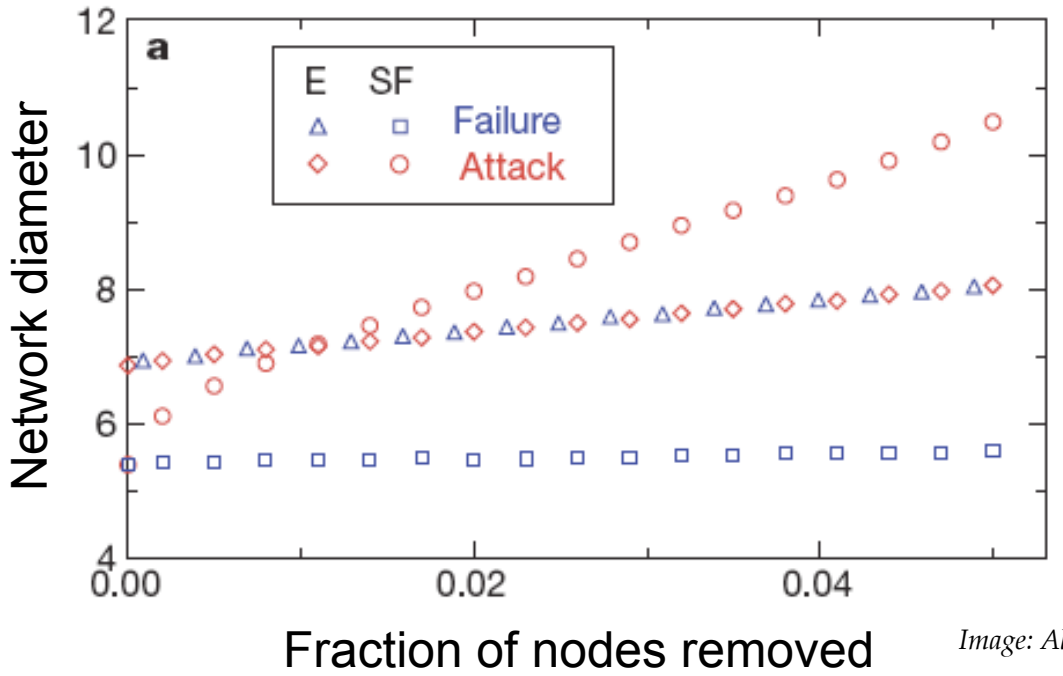


*Image: Albert et al., Nature **406**, 378 (2000)*

Hierarchical networks

Scale-free networks generated using preferential attachment have low clustering coefficients.

Some networks such as metabolic networks however have high clustering coefficients as well as scale-free topologies.

New category of networks: *Hierarchical networks,* characterized by a scale-free structure of densely connected modules.

Hierarchical networks

Hierarchical networks can be formed by simple algorithms such as the following:

1) Start with a module (small graph) with a central node and peripheral nodes.

2) Make *m* copies of the module.

3) Connect the central nodes of the copies to each other.

4) Connect the peripheral nodes of the copies to the central node of the original.

5) This is the new module, with the original central node as its central node.

6) Repeat from 2).

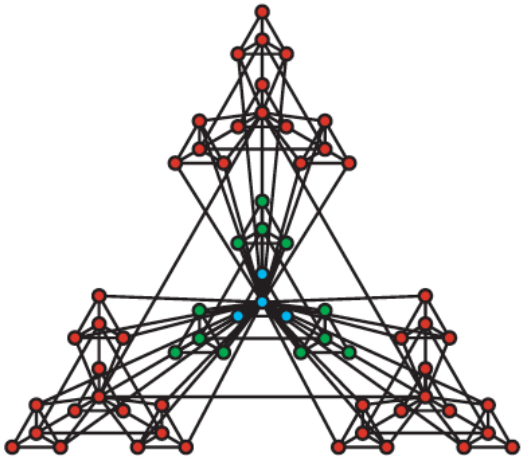


Image: Ravasz et al., Science ***297****, 1551 (2002)*

Hierarchical networks

In hierarchical networks we observe:

$$C(k) \sim k^{-1}$$

In other words we have small densely connected modules (small *k*, large *C*), connected through hubs (large *k*, small *C*).

Several metabolic networks show this behaviour (see right).

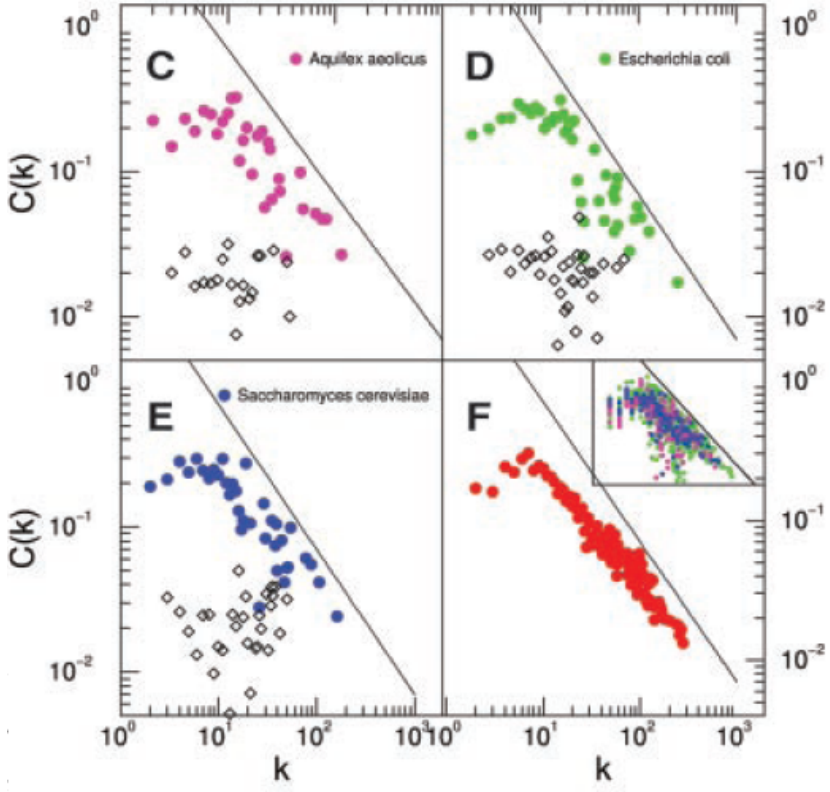


Image: Ravasz et al., Science **297**, *1551 (2002)*

Rich clubs

The so called *rich-club phenomenon* describes a network in which a small number of nodes with high degree are densely connected with each other.

The *rich-club coefficient* is calculated as follows:

1) Rank the nodes by degree, and normalize this rank by dividing by the total number of nodes N. This normalized rank is r.

2) The rich-club coefficient $\phi(r)$ is the fraction of *realized* edges among the nodes of rank r or lower (i.e. higher in the ranking).

$$\phi(r) = \frac{number\ of\ edges\ among\ nodes\ with\ rank < r}{rN(rN-1)/2}$$

Zhou & Mondragon. IEEE Comm. Lett. **8**, 180 (2004).

Rich clubs

But this is not enough. Since hubs are more likely to be connected to each other we need to compare the rich club to that of a randomized network with the same degree distribution.

We can achieve this e.g. using double-edge swaps.

Dividing by the randomized coefficient gives us a normalised coefficient.

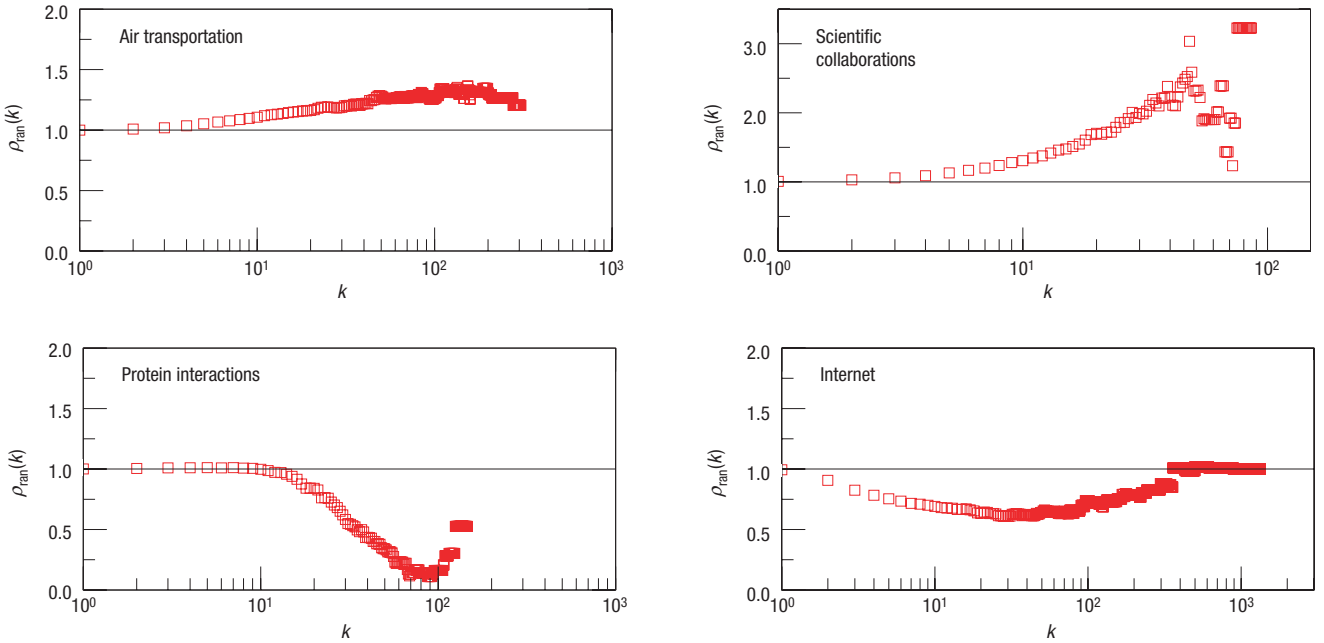
$$\phi_N(r) = \frac{\phi(r)}{\phi_{random}(r)}$$

If this is greater than 1 we have a rich club.

Colizza et al., Nat. Phys. 2, 110 (2006).

Rich clubs

Some real-world networks show rich clubs and some do not:



Colizza et al., Nat. Phys. 2, 110 (2006).

Network motifs

Network motifs are subgraphs of a few nodes which appear in directed networks more often than would be expected by chance.

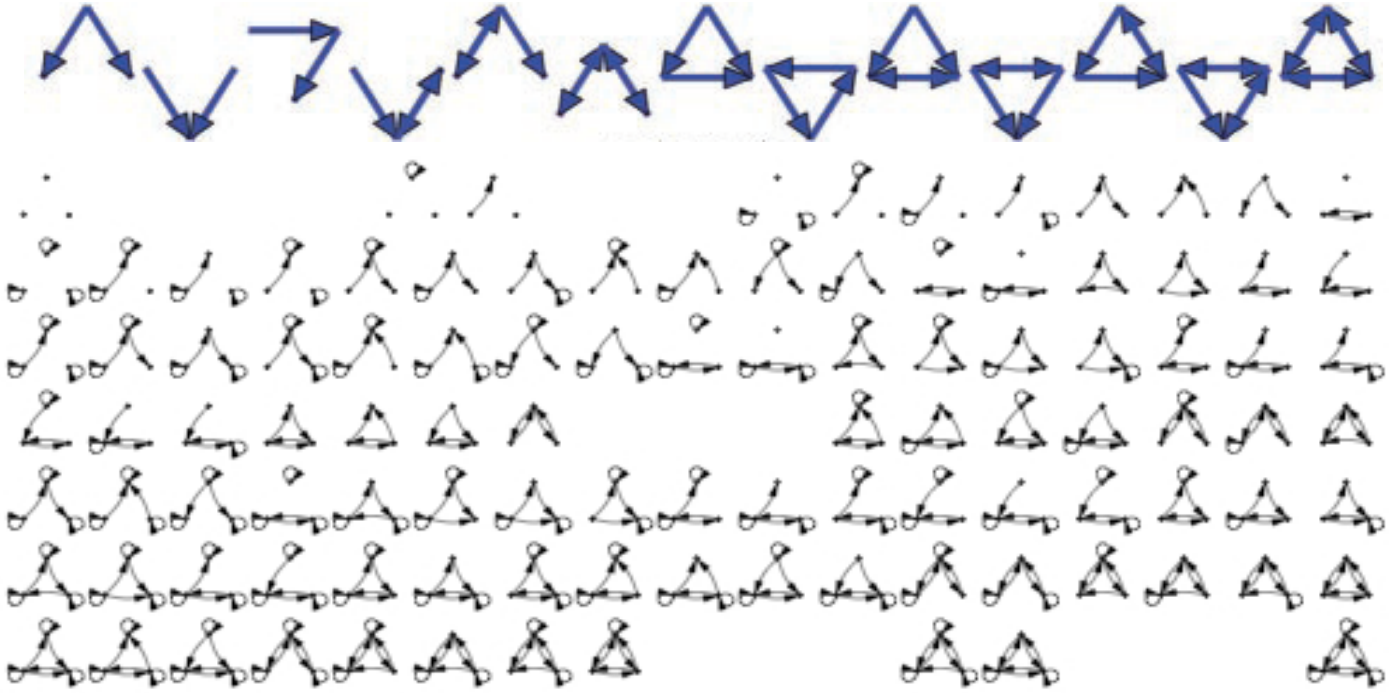


Image (top): Milo et al., Science ***303****, 1538 (2004)*

Network motifs

To evaluate whether their number is higher than would be expected by chance, the networks are randomized by swapping two inputs or two outputs.

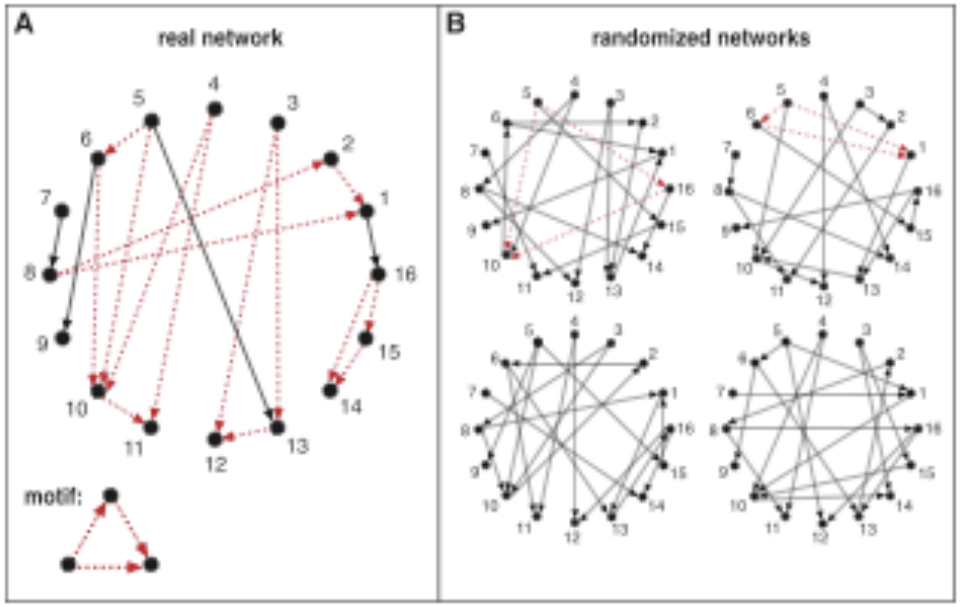


Image: Milo et al., Science ***298****, 824 (2002)*

This gives rise to a network with the same in- and out-degrees as the original network.

Superfamilies

Alon (2004) showed that the frequency signatures of network motifs classify networks into *superfamilies*.

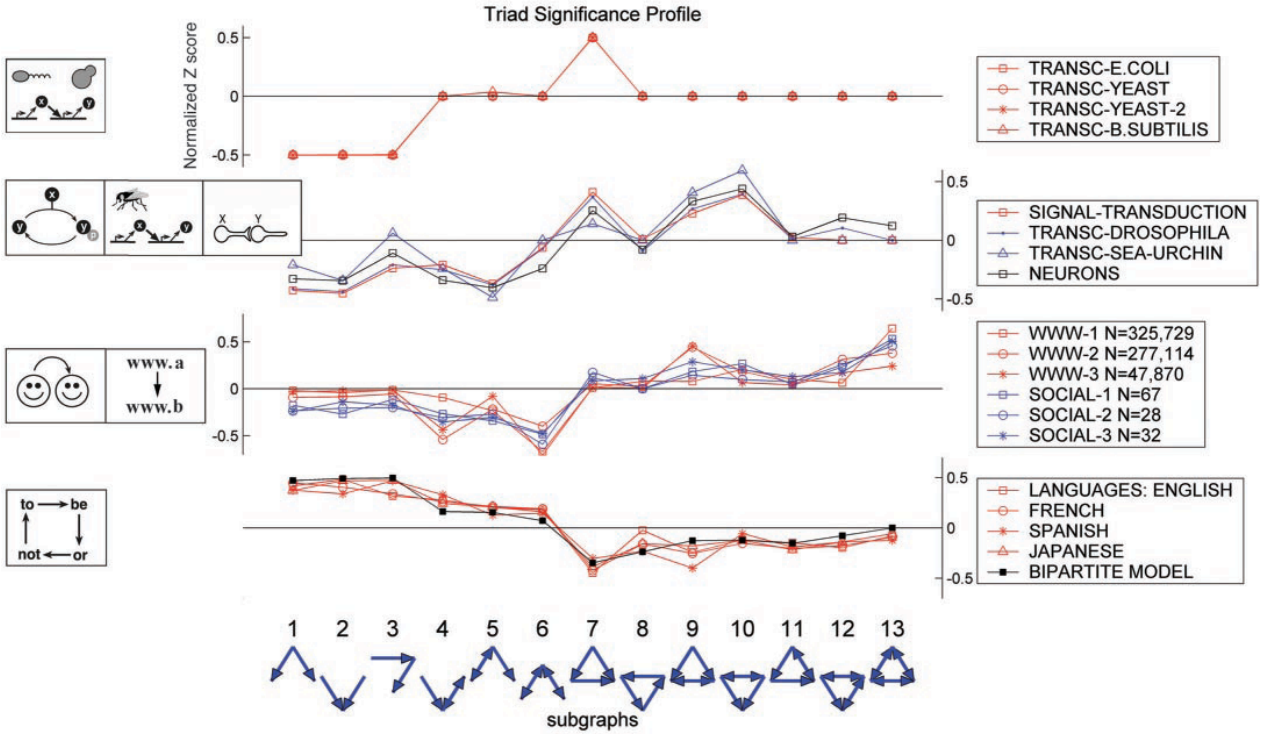


*Image: Milo et al., Science **303**, 1538 (2004)*

Superfamilies

What does this mean? Feed-forward loop is important in transcription networks and other regulatory networks.

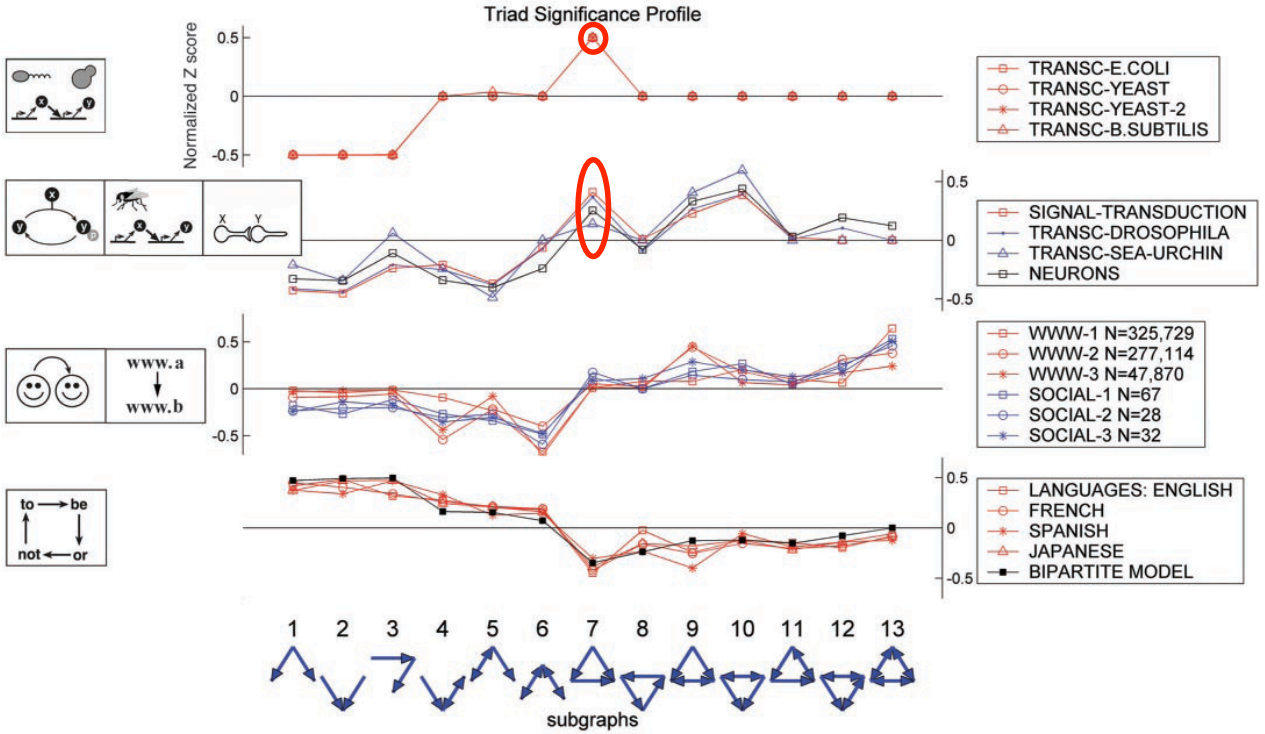


*Image: Milo et al., Science **303**, 1538 (2004)*

Superfamilies

…and feedback loops seem to be bad?

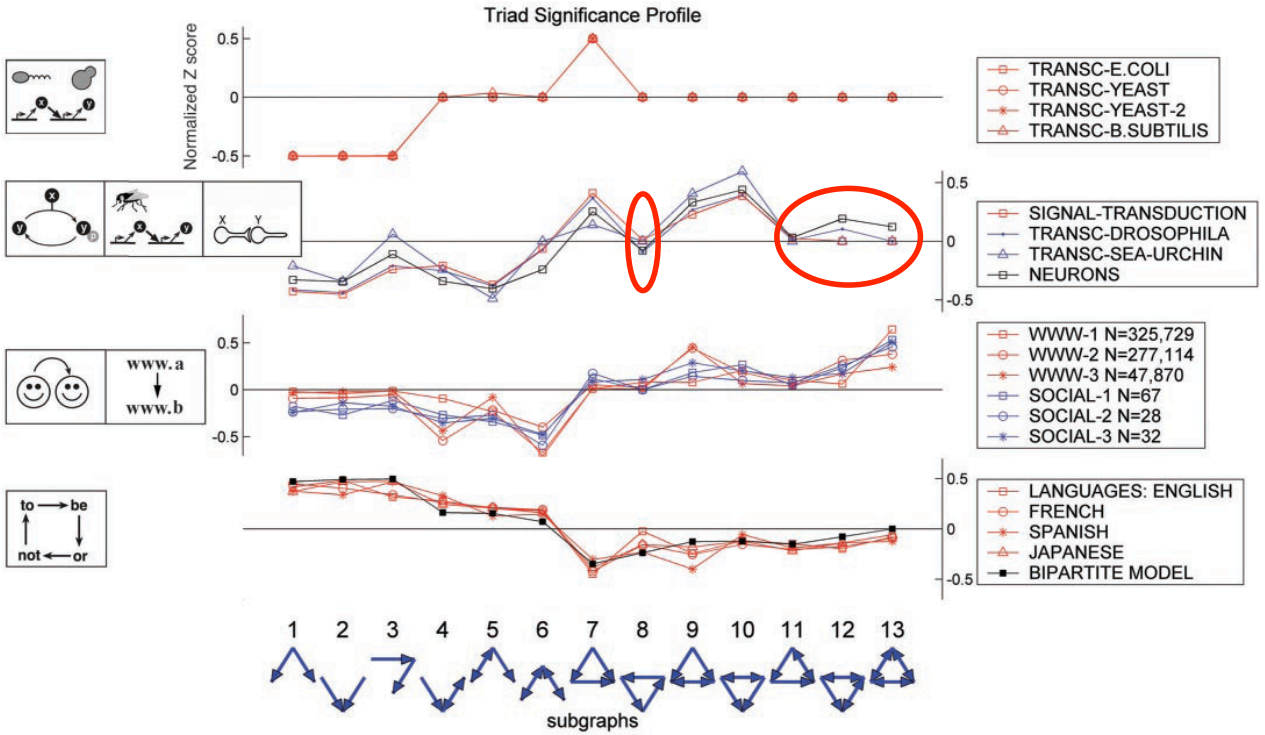


*Image: Milo et al., Science **303**, 1538 (2004)*

Transition matrix

A *transition matrix* can be defined by:

$$\mathbf{N} = \mathbf{K}^{-1}\,\mathbf{A}$$

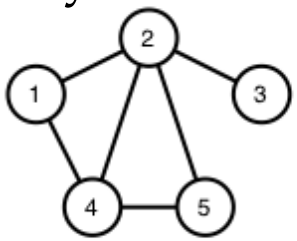
where **K** is the diagonal matrix with the degrees k_i on the diagonal:

$$k_{ij} = \delta_{ij}\,k_i = \delta_{ij}\,\Sigma_k\,a_{ik}$$

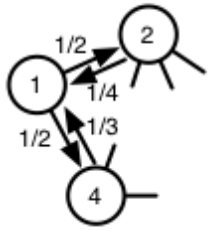
and where **A** is the adjacency matrix.

Transition matrix

In a transition matrix, all edges emanating from one node are divided by the degree, which corresponds to giving them a uniform probability.



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



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

The transition matrix thus describes the way a random walker would traverse the network, if we *left-multiply* a row vector.

More about this later.

Transition matrix

We can write **N** also as:

$$n_{ij} = a_{ij} / k_i$$

Because all the entries in a row of **N** add to one, any constant vector **b** given by:

$$b_i = c \quad \forall i$$

will be a (right-)eigenvector of **N** with eigenvalue 1:

$$\mathbf{(N\ b)}_i = \Sigma_j\, n_{ij}\, b_j = \Sigma_j\, a_{ij}\, b_i / k_i = c\, \Sigma_j\, a_{ij}\, / k_i = c = b_i$$

since $k_i = \Sigma_j\, a_{ij}$, so that $\mathbf{N\ b} = \mathbf{b}$.

Transition matrix

Although **N** is *not* symmetric, all the eigenvalues **x** of the transition matrix are real, since:

$$\mathbf{N}\,\mathbf{x} = \lambda\,\mathbf{x} \qquad \text{(eigenvector equation)}$$

Left-multiplying both sides by $\mathbf{K}^{1/2}$ gives:

$$\mathbf{K}^{1/2}\,\mathbf{N}\,\mathbf{x} = \mathbf{K}^{1/2}\,\lambda\,\mathbf{x}$$

Introducing $\mathbf{x'} = \mathbf{K}^{1/2}\,\mathbf{x}$ and thus $\mathbf{x} = \mathbf{K}^{-1/2}\,\mathbf{x'}$ we get:

$$\mathbf{K}^{1/2}\,\mathbf{N}\,\mathbf{K}^{-1/2}\,\mathbf{x'} = \mathbf{K}^{1/2}\,\lambda\,\mathbf{K}^{-1/2}\,\mathbf{x'}$$

Transition matrix

(cont'd)

We had:

$$\mathbf{K}^{1/2}\,\mathbf{N}\,\mathbf{K}^{-1/2}\,\mathbf{x}' = \mathbf{K}^{1/2}\,\lambda\,\mathbf{K}^{-1/2}\,\mathbf{x}'$$

And since $\mathbf{N} = \mathbf{K}^{-1}\,\mathbf{A}$ (RHS) and $\mathbf{K}^{1/2}\,\mathbf{K}^{-1/2} = \mathbf{I}$ (LHS) we get:

$$\mathbf{K}^{-1/2}\,\mathbf{A}\,\mathbf{K}^{-1/2}\,\mathbf{x}' = \lambda\,\mathbf{x}'$$

So that the eigenvalues λ of $\mathbf{N}$ are shared by the symmetric matrix $\mathbf{K}^{-1/2}\,\mathbf{A}\,\mathbf{K}^{-1/2}$ and hence must be real.

Transition matrix

But a constant vector is only an eigenvector of **N** if the network is connected.

If a network consists of n disjoint subgraphs (or n *connected components*), we get a degeneracy of n eigenvalues equals to 1.

The corresponding eigenvectors have constant entries

$$b_i = c$$

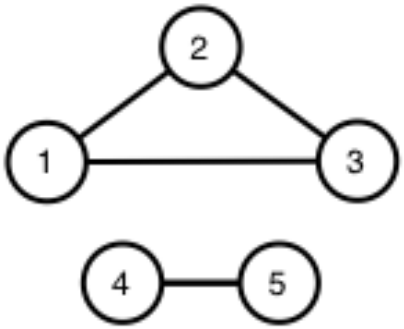
for nodes i that are part of the component, and

$$b_i = 0$$

for all other i.

Transition matrix

Divison of a network into two connected components:



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

These two eigenvectors of **N** correspond to the degenerate eigenvalues $\lambda = 1$.

Transition matrix

The largest eigenvalue **N** can have is $\lambda = 1$.

One way of obtaining the eigenvector **x** corresponding to the largest eigenvalue of a matrix **N** is to raise it to the power of m where $m \to \infty$ and apply it to (almost) *any* vector **y**.

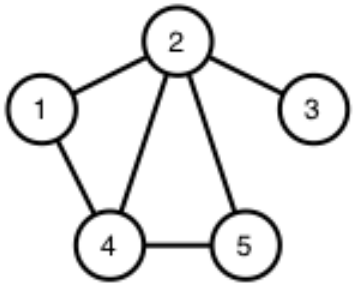
$$\mathbf{N}^m \mathbf{y} \to \mathbf{x} \qquad \text{as } m \to \infty$$

Since any vector can be expressed in terms of an eigenvector expansion, the eigenvector(s) with the largest eigenvalue eventually dominate. Which of the eigenvectors with eigenvalue 1 we end up with depends on our starting vector **y**.

Laplacian matrix

The *Laplacian* matrix is a similarly useful matrix defined by:

$$\mathbf{L} = \mathbf{K} - \mathbf{A}$$



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

$$\mathbf{L} = \mathbf{K} - \mathbf{A} = \begin{pmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ 0 & -1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 3 & -1 \\ 0 & -1 & 0 & -1 & 2 \end{pmatrix}$$

Laplacian matrix

The matrix **L** can also be written as:

$$l_{ij} = \delta_{ij} k_i - a_{ij}$$

from which we can quickly deduce that constant vectors **b** with $b_i = c$ are eigenvectors of **L** with eigenvalue 0:

$$(\mathbf{L}\ \mathbf{b})_i = \Sigma_j\ l_{ij}\ b_j = \Sigma_j\ \delta_{ij}\ k_i\ b_j - \Sigma_j\ a_{ij}\ b_j = c\ \Sigma_j\ \delta_{ij}\ k_i - c\ \Sigma_j\ a_{ij} = 0$$

since $k_i = \Sigma_j\ a_{ij}$.

Laplacian matrix

Hence the eigenvectors which identified connected components with $\lambda = 1$ in $\mathbf{N}$ correspond to $\lambda = 0$ eigenvectors of $\mathbf{L}$.

With $\mathbf{L}$ we can also identify communities - meaning subgraphs which to a good approximation form separate connected components and are only linked by a few connections.

The degeneracy of $\lambda = 0$ eigenvalues is broken and we get one trivial eigenvector which is entirely constant as well as the first non-trivial eigenvector with λ close to zero, which for m communities is split into m sets of equal or almost equal values.

Eigenvector centrality

Eigenvector centrality is another way of assessing the importance of a node in a network. It is constructed as follows:

Consider a measure of importance x_i of every node i, which fulfills the following condition:

We want the importance of each node to be proportional to the sum of the importance of its neighbours.

This is a recursive, and thus very elegant, definition.

Eigenvector centrality

One way of writing this is:

$$x_j \propto \Sigma_i \, x_i \, A_{ij}$$

With a constant of proportionality $1/\lambda$ this becomes the eigenvector equation:

$$\lambda \, \mathbf{x} = \mathbf{x} \, \mathbf{A}$$

Hence an eigenvector of the adjacency matrix gives us the importance values of each node.

But which eigenvector?

Eigenvector centrality

It is the eigenvector with the largest eigenvalue, since - according to the *Perron-Frobenius theorem* - this is the only one guaranteed to be entirely non-negative.

We know how to get this eigenvector: By raising a matrix to a power m where $m \to \infty$, this time the adjacency matrix.

Applying the adjacency matrix to a constant vector of ones will be equivalent to every node passing a 'vote' to every neighbour.

When applying the adjacency matrix again, let every node pass as many 'votes' as it has received to each neighbour.

While the total number of votes grows, the normalized distribution of votes will become more and more similar to the eigenvector of the largest eigenvalue, which gives us the *eigenvector centrality*.

The PageRank algorithm

Now consider our transition matrix we discussed earlier:

$$\mathbf{N} = \mathbf{K}^{-1}\,\mathbf{A}$$

What we did before was to *right*-multiply **N**, but if we *left*-multiply it by a row-vector **v**, then this gives us the average occupancy of a set of random walkers with initial configuration **v**, after one time step.

As we apply the matrix **N** to this vector repeatedly, we model the probability distribution of the walker, which eventually becomes the left-eigenvector of the largest left-eigenvalue and the equilibrium walker occupancy across the network.

The PageRank algorithm

The *PageRank* algorithm which powers the Google search engine is very similar to this:

The only difference is that the adjacency matrix **A** is now directed, and its entries are normalized by the out-degree $k_i^{(out)}$:

$$n_{ij}^{(\mathrm{PR})} = a_{ij} / k_i^{(out)}$$

or

$$\mathbf{N}^{(\mathrm{PR})} = \mathbf{K_{out}}^{-1}\mathbf{A}$$

The PageRank algorithm

Thus we can again consider a random walk on the network, governed by this time by the transfer matrix $\mathbf{N}^{(PR)}$, with the eigenvector solution

$$\mathbf{p} = \mathbf{p}\mathbf{N}^{(PR)}$$

Where the entries of eigenvector $\mathbf{p}$ are the PageRank values.

The PageRank values can be considered as the long-term distribution of random walkers across the network.

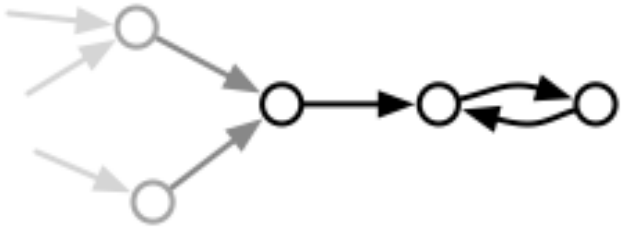
Note that we need to cut out any *dangling nodes* with zero out-degree (of which there are many in the WWW).

The PageRank algorithm

Solving an eigenvalue problem for a matrix with billions of rows and columns like the WWW would be, is impossible analytically.

What is done in practice, is to apply the power method which we have mentioned before - in other words to apply the matrix $\mathbf{N}^{(PR)}$ iteratively.

However, there is a danger of the evolution being trapped due to subgraphs such as this one:



The PageRank algorithm

The way to avoid these trapped states is to make random jumps to other nodes possible, with a small probability.

This corresponds to creating a new transfer matrix

$$\mathbf{N'}^{(\mathrm{PR})} = \alpha\mathbf{N}^{(\mathrm{PR})} + (1 - \alpha)\mathbf{E}$$

where $\mathbf{E}$ is a matrix with $e_{ij} = 1/N$ with N being the number of nodes and 1-α being the probability of a random jump.

The eigenvector of this matrix $\mathbf{N'}^{(\mathrm{PR})}$ corresponds to the original PageRank proposed by Sergey Brin and Larry Page in 1998.

The PageRank algorithm

A few things worth noting:

- The random jump capability is sometimes also interpreted as an *attenuation* or *damping factor*, representing the fact that a random surfer on the web will stop clicking at some point.

- The modified matrix $\mathbf{N'}^{(PR)}$ without trapped states is called *irreducible* and there exists a unique solution for the power method, which is the eigenvector corresponding to the largest eigenvalue.

- PageRank vectors are usually normalized to 1, which is why the PageRank equation is sometimes written as:

$$\mathrm{PR}(v_i) = (1 - d)/N + d \sum_j \mathrm{PR}(v_j)/\mathrm{L}(v_j)$$

where $\mathrm{PR}(v_j)$ and $\mathrm{L}(v_j)$ are the PageRank and out-degree of vertex j.

A new impact factor

The PageRank algorithm has been applied to other systems apart from the World Wide Web.

Most notably, a paper by Bollen, Rodriguez and Van de Sompel (BRV) applies it to the network of journal citations in order to create a new kind of *impact factor*.

Traditionally the impact factor as defined by the company ISI is simply the average number of citations per paper which a journal receives over the preceding two years.

This is quite a crude measure, since it does not reflect the quality of the citations.

A new impact factor

An important difference between the WWW and journal citations is that the network of journal citations is a *weighted* matrix w_{ij}. This leads to a definition of the weighted PageRank transfer matrix $\mathbf{N}^{(\mathrm{wPR})}$ as:

$$n_{ij}^{(\mathrm{wPR})} = w_{ij} / s_i^{(out)}$$

where

$$s_i^{(out)} = \sum_j w_{ij}$$

is the *out-strength* of node i.

What this means is simply that the random walker now is more likely to go to some journals than others, *proportional to their relative share of citations*. Other than that the algorithm is the same.

A new impact factor

The BRV paper distinguishes *popularity* of a journal, which is simply its number of citations, or in-degree, and the *prestige*.

The ISI impact factor is an indicator of the popularity of a journal, while the PageRank indicates its prestige.

BRV suggest a combined measure which is the product of the two:

$$\mathrm{Y}(v_i) = \mathrm{IF}(v_i) \times \mathrm{PR}_{\mathrm{w}}(v_i)$$

A new impact factor

Ranking journals by the Y-factor gives an intuitively sensible picture:

	ISI IF		PR_w		Y-factor	
rank	value	Journal	value (x 10^3)	Journal	value(x 10^2)	Journal
1	52.28	ANNU REV IMMUNOL	16.78	NATURE	51.97	NATURE
2	37.65	ANNU REV BIOCHEM	16.39	J BIOL CHEM	48.78	SCIENCE
3	36.83	PHYSIOL REV	16.38	SCIENCE	19.84	NEW ENGL J MED
4	35.04	NAT REV MOL CELL BIO	14.49	PNAS	15.34	CELL
5	34.83	NEW ENGL J MED	8.41	PHYS REV LETT	14.88	PNAS
6	30.98	NATURE	5.76	CELL	10.62	J BIOL CHEM
7	30.55	NAT MED	5.70	NEW ENGL J MED	8.49	JAMA
8	29.78	SCIENCE	4.67	J AM CHEM SOC	7.78	LANCET
9	28.18	NAT IMMUNOL	4.46	J IMMUNOL	7.56	NAT GENET
10	28.17	REV MOD PHYS	4.28	APPL PHYS LETT	6.53	NAT MED

from: Bollen et al., Scientometrics ***69*** *(3) (2006)*

A new impact factor

Popular and prestigious journals in physics*:

	Popular: ISI IF ↑, PR_w < 40%-tile				Prestigious: ISI IF ↓, PR_w > 90%-tile			
	Journal title	ISI IF	$PR_w \times 10^5$	IF_Δ	Journal title	ISI IF	$PR_w \times 10^3$	IF_Δ
1	ANNU REV NUCL PART S	8.67	6.35	7.11	PHYS REV LETT	7.04	8.41	−1.52
2	SOLID STATE PHYS	7.00	3.85	5.46	J APPL PHYS	2.17	2.59	−1.50
3	PROG NUCL MAG RES SP	5.97	6.53	4.41	PHYS REV E	2.20	2.34	−1.27
4	ATOM DATA NUCL DATA	4.63	5.94	3.08	APPL PHYS LETT	4.05	4.28	−1.05
5	CRIT REV SOLID STATE	4.44	3.11	2.91	JPN J APPL PHYS	1.17	0.83	−1.03
6	ADV ATOM MOL OPT PHY	4.11	6.16	2.55	NUCL INSTRUM METH A	1.17	0.57	−0.81
7	PROG SURF SCI	3.74	6.31	2.19	J PHYS A−MATH GEN	1.36	0.61	−0.65
8	CHEM VAPOR DEPOS	2.07	5.29	0.52	J PHYS−CONDENS MAT	1.76	0.93	−0.52
9	RIV NUOVO CIMENTO	1.70	3.21	0.17	J CHEM PHYS	2.95	2.32	−0.50
10	J NONLINEAR SCI	1.62	6.10	0.06	PHYS FLUIDS	1.57	0.62	−0.45

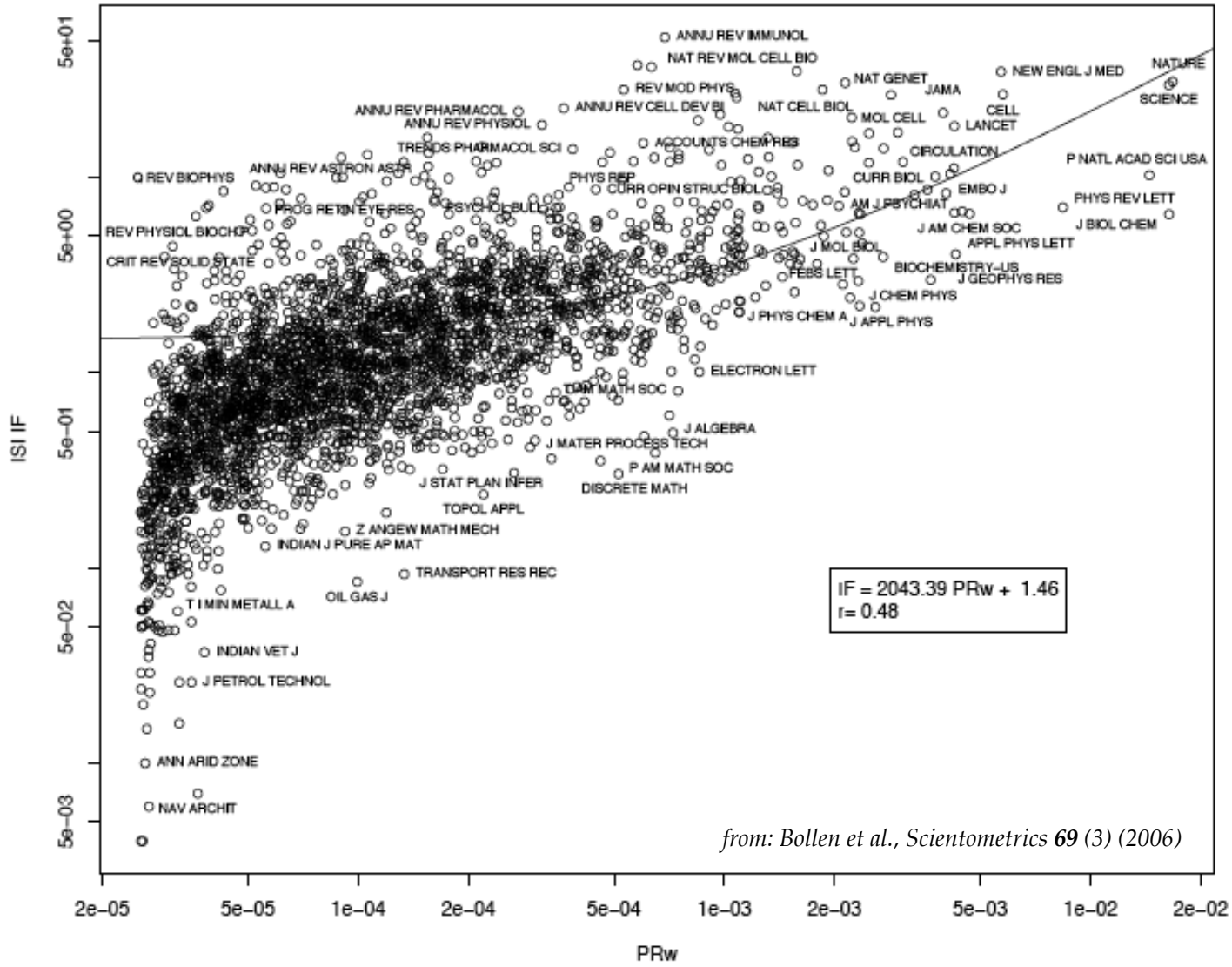
from: Bollen et al., Scientometrics ***69*** *(3) (2006)*

*ranked by IF_Δ , the deviation from the ISI IF linear regression shown as a solid line in the IF vs. PR_w plot.

A new impact factor

Also very interesting:

PR_w vs. IF



*from: Bollen et al., Scientometrics **69** (3) (2006)*

A new impact factor

While there is some correlation between the ISI IF and weighted PageRank, there are significant outliers which fall into two categories:

Popular Journals - cited frequently by journals with little prestige: *high* ISI IF, *low* weighted PageRank

Prestigious Journals - not frequently cited, but when they are, then by highly prestigious journals:
low ISI IF, *high* weighted PageRank

Boolean networks

Often we are not only interested in the topological properties of a network, but also in its *dynamical* properties.

Dynamic processes take place on many networks. The nodes interact and their state changes as a result of these interactions.

One of the simplest models of a dynamical network is the *Boolean* network.

Boolean networks

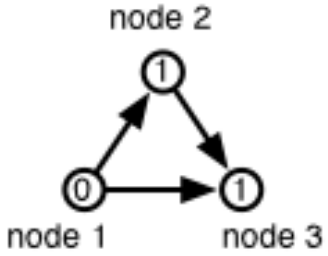
A Boolean network is directed, and each node is in one of two states, 0 or 1.

Furthermore, each node has a set of rules which tell it its state depending on the states of its neighbours in the network.

This set of rules is called a *Boolean function* and consists of a bit string of length 2^k where k is the number of inputs (i.e. the in-degree) of the node.

Boolean networks: Example

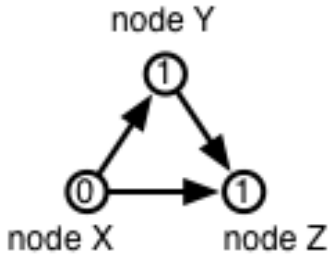
Consider a three node directed network where each node is in state 0 or 1, for example:



Now we need a dynamic rule for each node which tells it what state to be in, depending on the state of the nodes it gets inputs from.

Boolean networks: Example

Node Y has one input, coming from node 1.



Node X can be in state 0 or in state 1.

And node Y can respond accordingly, in four different ways:

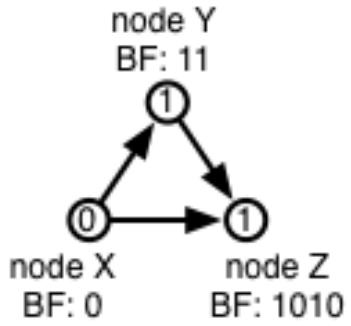
State of node X:	0	1	
Responses of node Y:			
	0	0	(independent of node X)
	0	1	(copy node X)
	1	0	(do the opposite of node X)
	1	1	(independent of node X)

Boolean networks: Example

Thus node Y has four possible rules of length two: 00, 01, 10 and 11.

Such rules which list a response for every possible input are called *Boolean functions*.

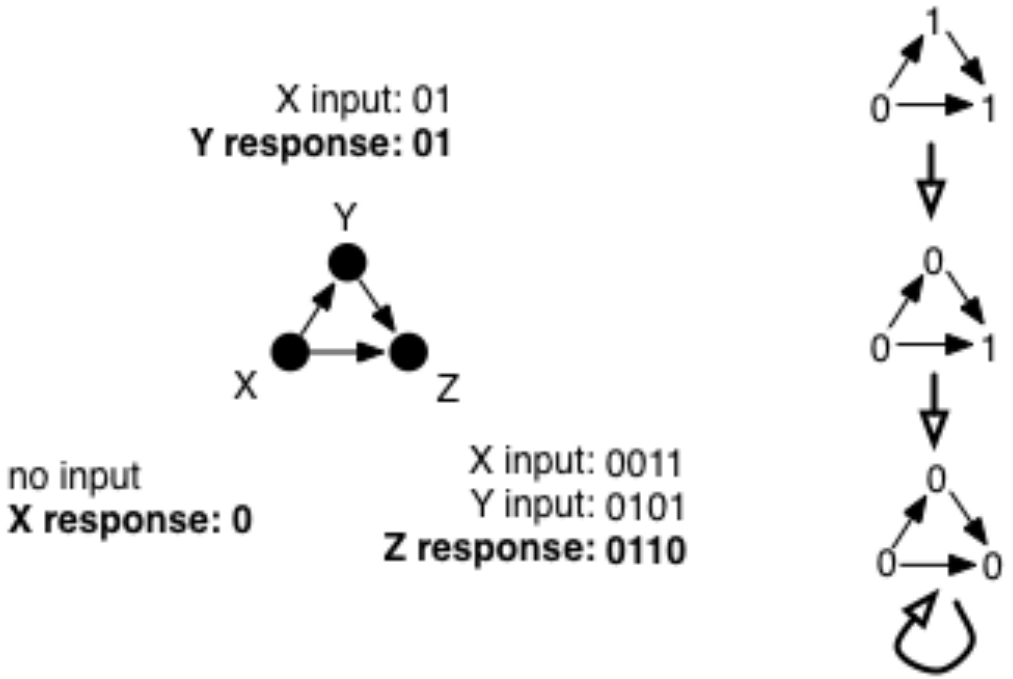
In general a node with k inputs (i.e. in-degree k) will have a Boolean function of length 2^k.



Hence our Boolean network is fully specified if we add three Boolean functions of length one, two and four to nodes X, Y and Z, respectively.

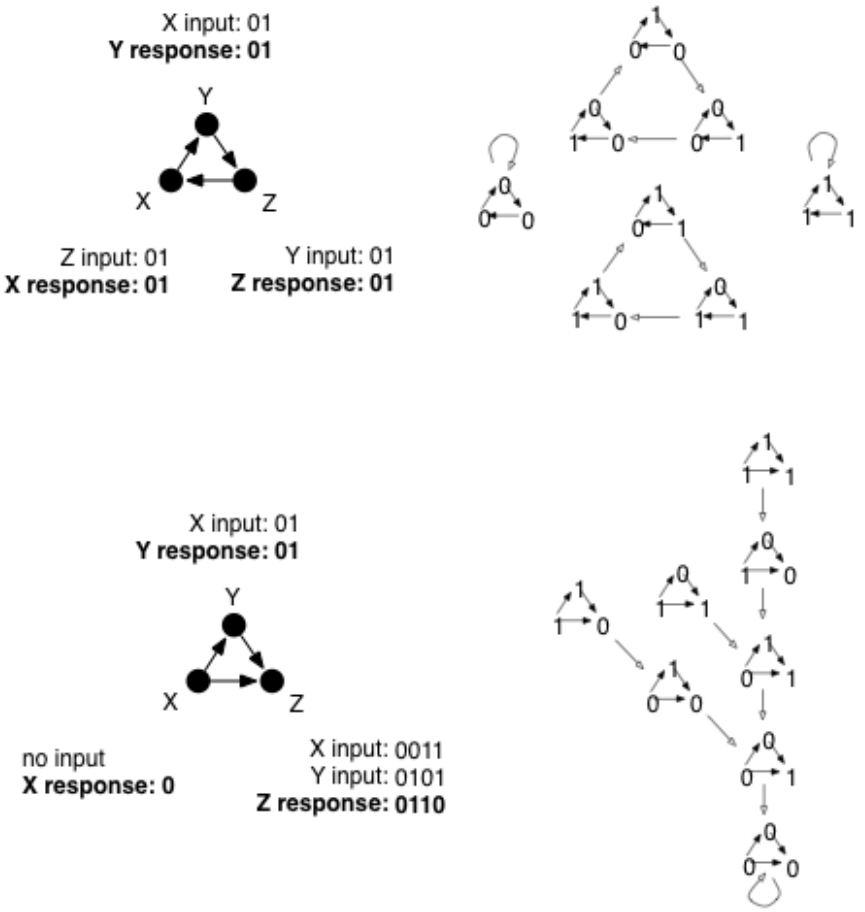
State space

A Boolean network of n nodes can be in one of 2^n states. As the rules are applied at each time step, the state of the network moves through *state space*.



Attractors and basins

The state space of a given Boolean network is partitioned into one or more *attraction basins*, each of which lead to an *attractor cycle*.



Basin entropy

An interesting measure of dynamical complexity which has been proposed by Shmulevitch & Krawitz (2007) is the *basin entropy* of a Boolean network.

This is simply the entropy S of the basin size distribution, so that for a N node network whose 2^N states are divided into M attraction basins of size b_i we have:

$$S = - \Sigma_M \, (b_i / 2^N) \ln (b_i / 2^N)$$

We have low entropy when there is only one basin, and high entropy when there are many similarly sized basins.

The authors suggest that the entropy S is a measure of the dynamical complexity of the Boolean network.

Kauffman networks

Kauffman networks (1969) are a particular class of Boolean network, in which:

1) N nodes are connected randomly such that each node has degree K.

2) The Boolean functions of length 2^K on each node are also random.

Random Boolean networks (RBNs) are sometimes termed *NK networks* (not to be confused with Kauffman's *NK model*).

Kauffman networks

The most interesting Kauffman networks have $K = 2$. In this case we have 16 possible Boolean functions, which we can divide into four categories:

Frozen:	0000, 1111
Canalyzing (C1):	0011, 1100, 0101, 1010
Canalyzing (C2):	0001, 0010, 0100, 1000, 1110, 1101, 1011, 0111
Reversible:	0110, 1001

The *frozen* functions ignore both inputs.

The *canalyzing* ones ignore one input completely (C1) or at least some of the time (C2).

The *reversible* ones never ignore any inputs, and are thus the only ones which do not lose information.

Kauffman networks

Kauffman networks as a whole can be in two phases, *frozen* and *chaotic*:

Frozen phase - Any perturbation travels on average to *less* than one node per time step.

Chaotic phase - Any perturbation travels on average to *more* than one node per time step.

In the chaotic phase the distance between two states increases exponentially with time, even if they are very close to start with.

Networks on the boundary between the frozen and chaotic phases are termed *critical*.

Critical networks

At $K = 2$, we need a perturbation to be passed on with probability $p = 1/2$ for the network to be critical, since we have two inputs and want to pass on a perturbation to one node on average.

- Frozen functions pass perturbations on with zero probability,

- Canalyzing functions pass a perturbation on with probability $p = 1/2$, and

- Reversible functions with unit probability.

Hence Kauffman networks with $K = 2$ are critical if frozen (0000, 1111) and reversible (1001, 0110) functions are selected with equal probability.

This is the case, for example, if the Boolean functions are drawn from a uniform random distribution.

Dynamical node classes

In terms of their dynamical behaviour, the nodes also fall into categories:

Frozen core - these nodes remain unchanged

Irrelevant nodes - these nodes have only frozen nodes as their outputs

Relevant nodes - all remaining nodes

The relevant nodes completely determine the number and size of attractors in the network.

Scaling laws

Much work has been done on the scaling of dynamical properties with network size, most notably the *number of attractors* and the *number of relevant nodes*.

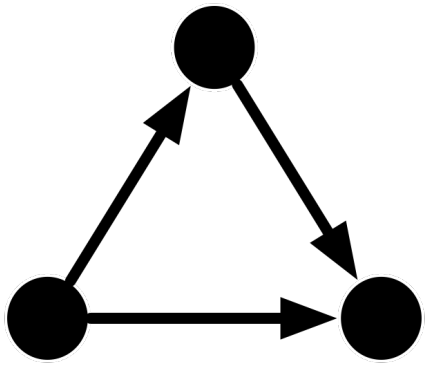
For many years it was believed that the number of attractors in an N-node Kauffman network scales as $N^{1/2}$, but recently it the scaling was shown to be superpolynomial (Samuelsson & Troein, 2003).

The number of relevant nodes has been shown to scale as $N^{2/3}$.

These scaling behaviours can only be detected in very large computer simulations, with $N > 10^9$.

The feed-forward loop

A fundamental building block of dynamical, regulatory networks is the *feed-forward* loop:

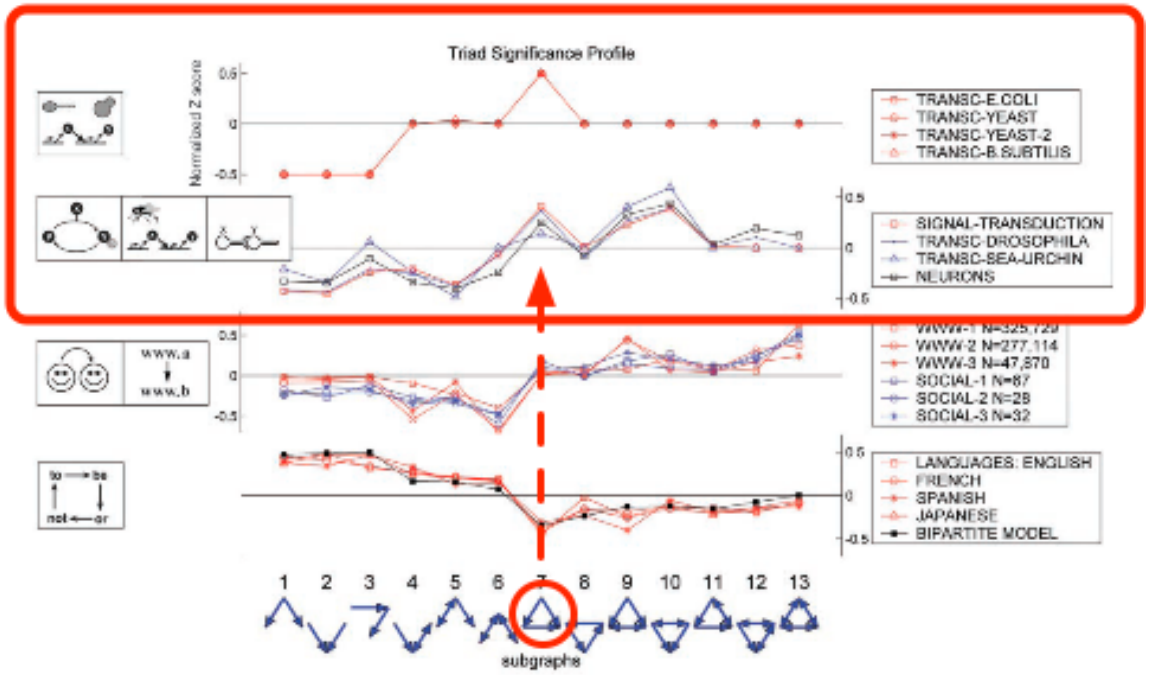


It has:

- one node with out-degree 2 and in-degree 0
- one node with out-degree 1 and in-degree 1
- one node with out-degree 0 and in-degree 2

The feed-forward loop

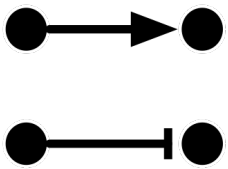
We have already seen evidence that the *topology* of the feed-forward loop occurs in transcription networks and neural networks. But what about the *dynamical* behaviour?



The feed-forward loop

In transcription networks each network edge can represent an *activation* or *inhibition*. In other words nodes can switch each other on and off.

Activation and inhibition are represented as…

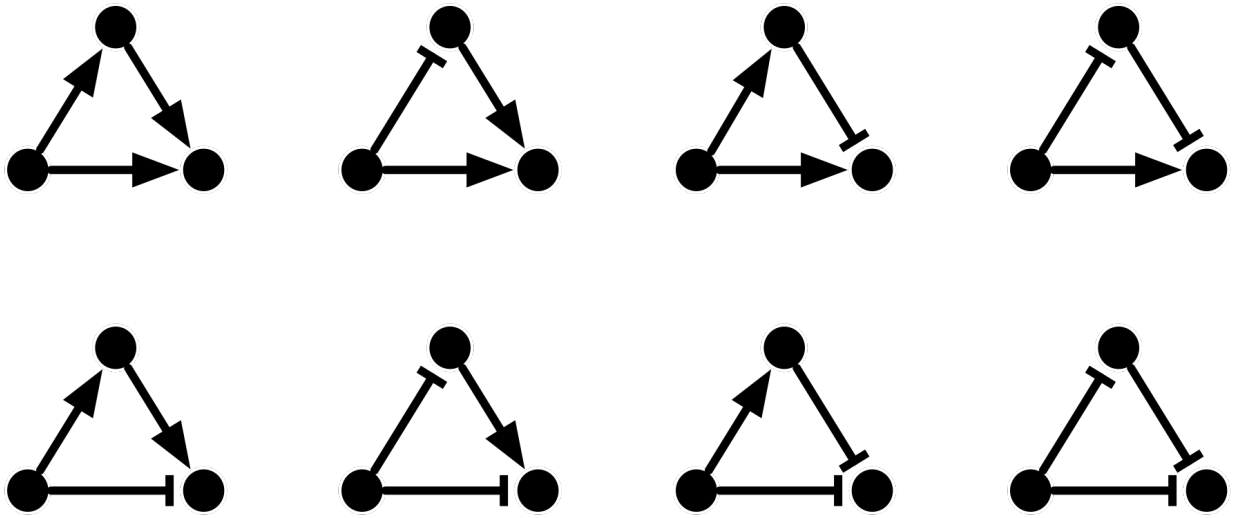


…respectively.

We will talk more about the biological details of this later.

The feed-forward loop

Hence we can draw eight different feed-forward loops:

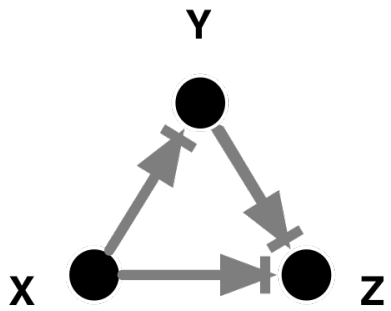


The feed-forward loop

In order to study the different dynamics this results in, we need to model the way in which these nodes switch each other off and on.

In the case of gene regulation, the following model is appropriate:

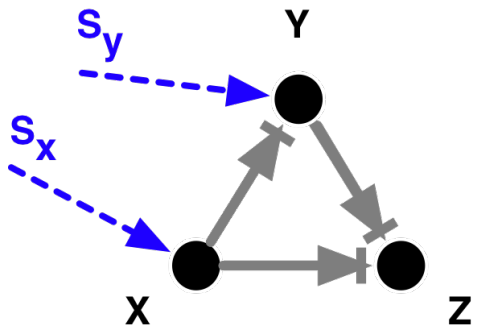
First, consider three genes, X, Y and Z, linked by activation or inhibition:



Each of these has a *concentration*, which is akin to an activation level.

The feed-forward loop

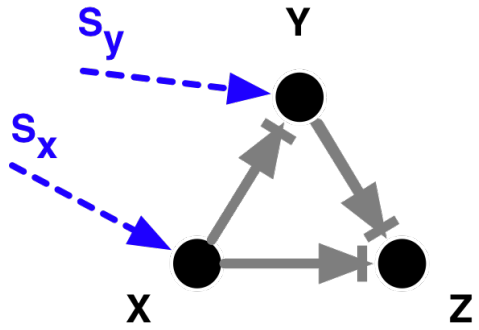
We have two external switches, S_x and S_y, which can be off (0) or on (1).



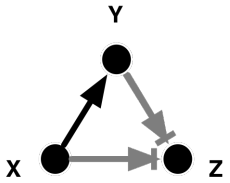
The concentration of **X**, denoted by x, is fully controlled by S_x:

$$x = S_x$$

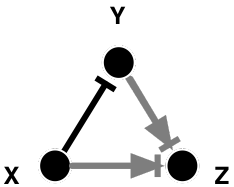
The feed-forward loop



The concentration of **Y**, denoted by y^*, is controlled by S_y via $y^* = y\, S_y$, but also by x. For the two cases of activation and repression we have:

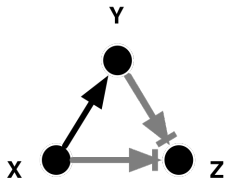


$$dy / dt = b_y + \beta_y \left[(x / k_{xy})^H / (1 + (x / k_{xy})^H) \right] - \alpha_y y$$

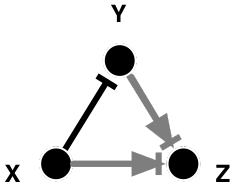


$$dy / dt = b_y + \beta_y \left[1 \; / \; (1 + (x / k_{xy})^H) \right] - \alpha_y y$$

The feed-forward loop



$$dy / dt = b_y + \beta_y \left[(x / k_{xy})^H / (1 + (x / k_{xy})^H) \right] - \alpha_y y$$



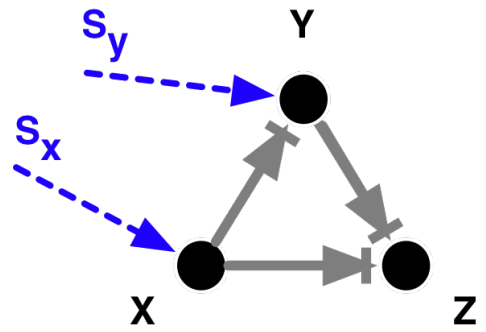
$$dy / dt = b_y + \beta_y \left[1 \ / \ (1 + (x / k_{xy})^H) \right] - \alpha_y y$$

where b_y is the *basal rate* of **Y** production, α_y is the decay rate, and the rest of the equation is based on the *Hill equation*, with H as the *Hill coefficient* and β_y and k_{xy} as constants.

For the dynamics we will consider, we set $H = 2$.

Note that the repressor case is the activator with $-H$.

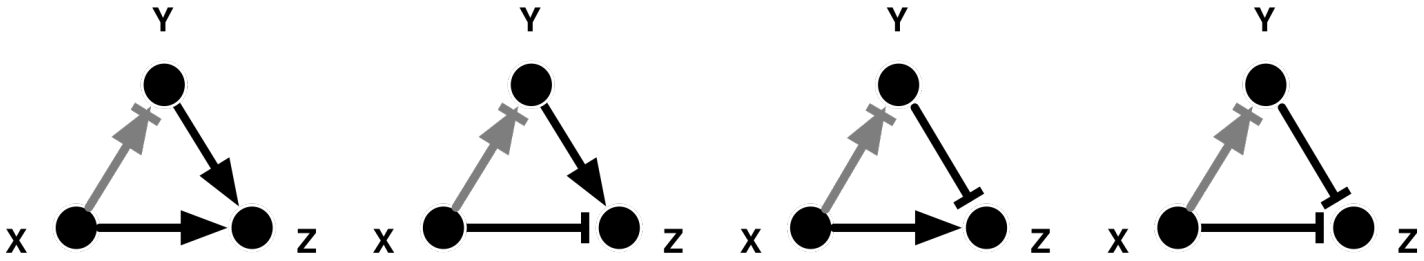
The feed-forward loop



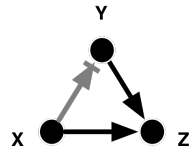
The concentration of **Z**, denoted by z, is dependent on x and y.

Here we consider an AND-type logic function (but others, such as OR are also possible).

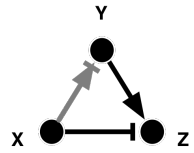
Altogether we have four possible responses, corresponding to the four scenarios:



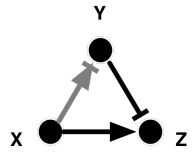
The feed-forward loop



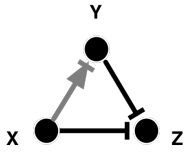
$$dz/dt = b_z + \beta_z \left[(x/k_{xz})^H/(1+(x/k_{xz})^H)\right]\left[(y^*/k_{yz})^H/(1+(y^*/k_{yz})^H)\right] - \alpha_z z$$



$$dz/dt = b_z + \beta_z \left[1/(1+(x/k_{xz})^H)\right]\left[(y^*/k_{yz})^H/(1+(y^*/k_{yz})^H)\right] - \alpha_z z$$



$$dz/dt = b_z + \beta_z \left[(x/k_{xz})^H/(1+(x/k_{xz})^H)\right]\left[1/(1+(y^*/k_{yz})^H)\right] - \alpha_z z$$

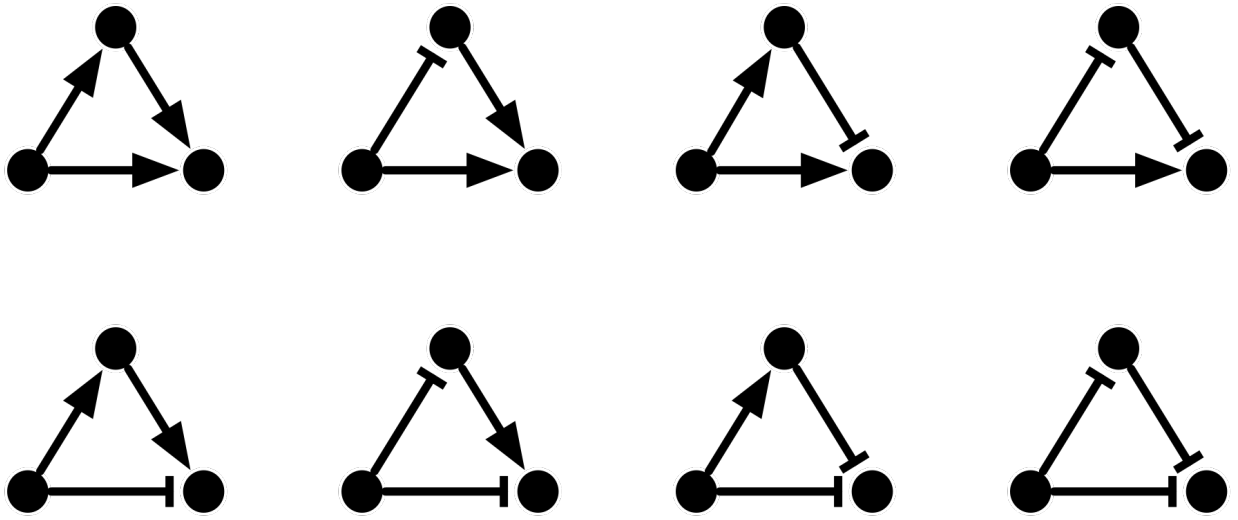


$$dz/dt = b_z + \beta_z \left[1/(1+(x/k_{xz})^H)\right]\left[1/(1+(y^*/k_{yz})^H)\right] - \alpha_z z$$

where $y^* = y\ S_y$.

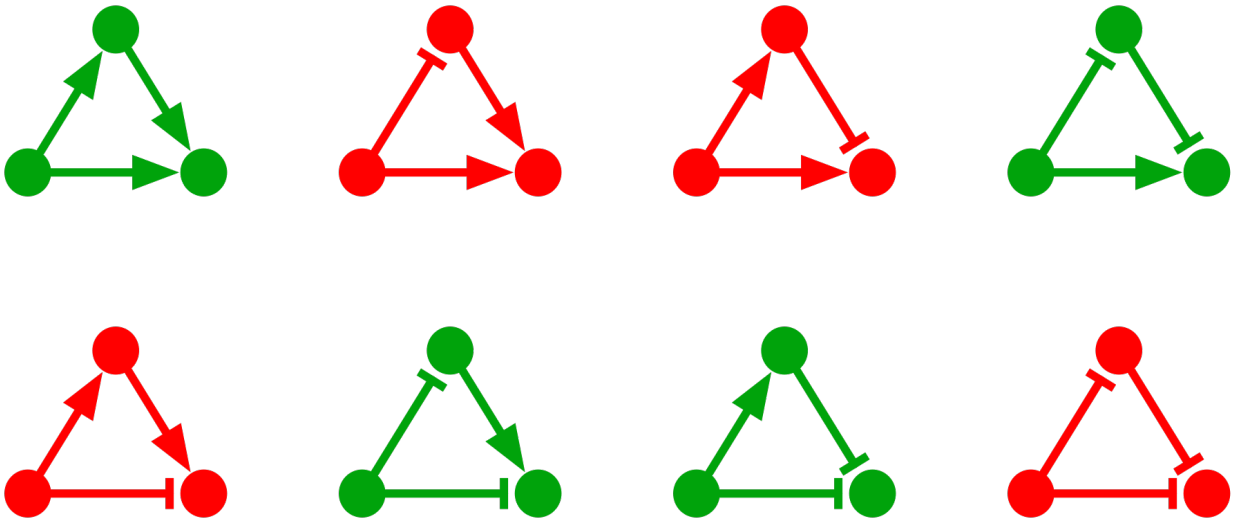
The feed-forward loop

Before we look at dynamics in detail, let us distinguish two basic types of feed-forward loop among the eight possible:



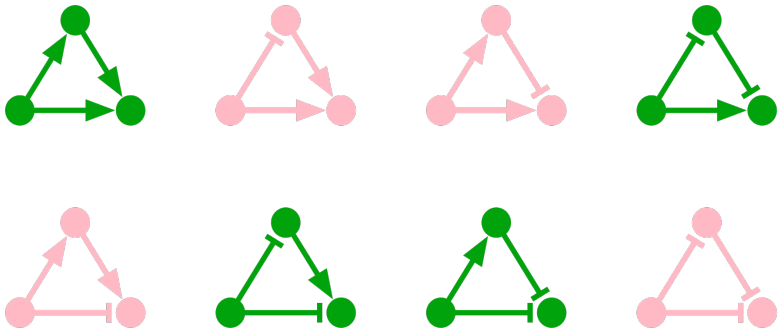
The feed-forward loop

Coherent and **incoherent**:

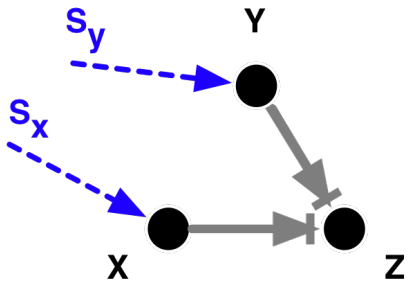


A loop is coherent if the *sign* of the XZ interaction is *the same as the combined sign* of the XY and YZ interactions.

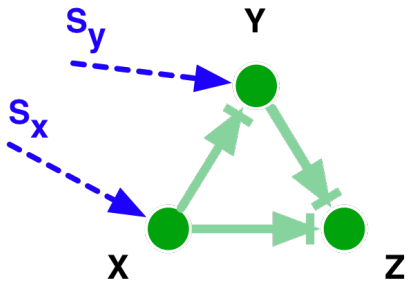
The feed-forward loop



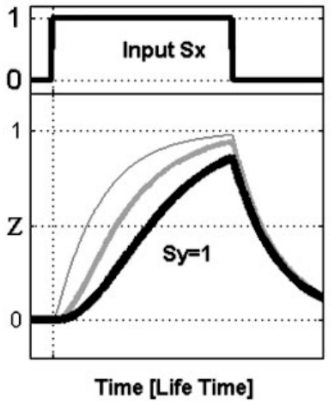
All coherent loops act as a *delay* circuit, relative to a simple XY and XZ co-regulation.



(thin line)

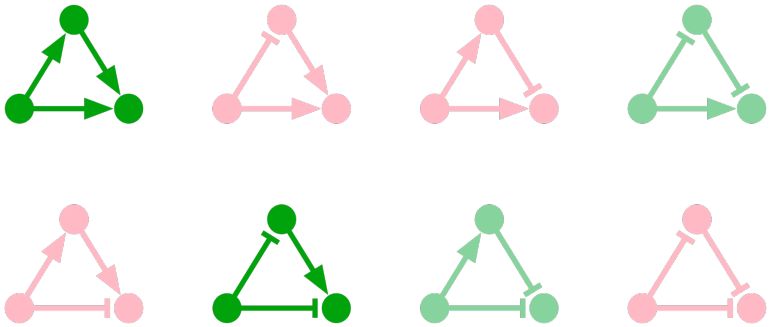


(thick lines)

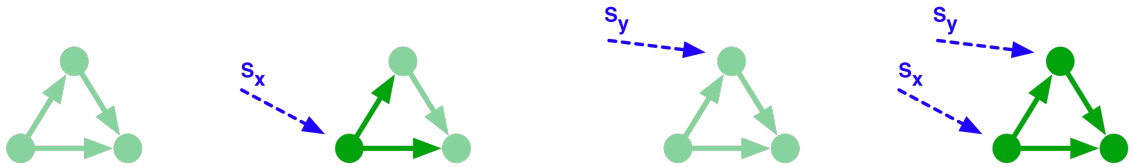


The delay is *sign-sensitive*, as it only affects the on-switching.

The feed-forward loop



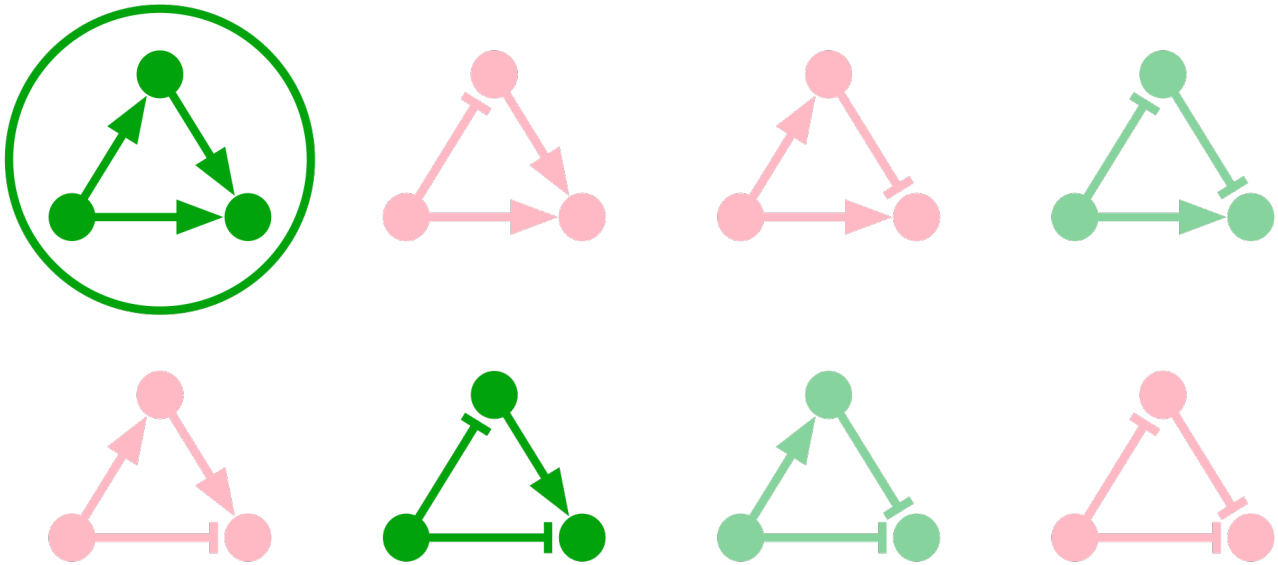
In only two of the four coherent loops does the steady-state of **Z** depend on both S_x and S_y. Contrast one of these two with one of the other two:



$\mathbf{Z} = S_x$ AND S_y

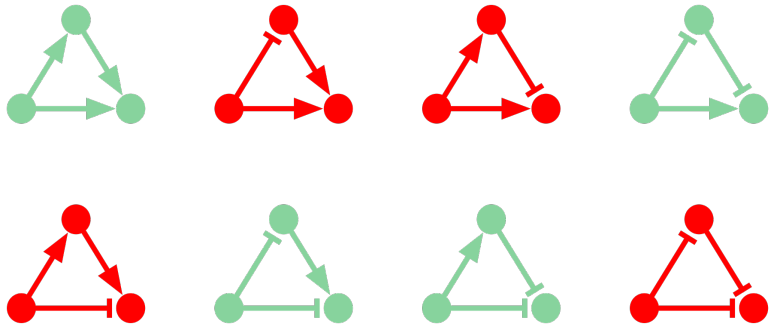
$\mathbf{Z} = S_x$

The feed-forward loop

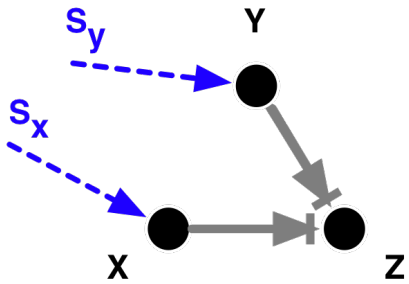


Interestingly, one of these two coherent loops with dependency on S_x and S_y is **much** more prevalent in real transcription networks than all the other coherent loops.

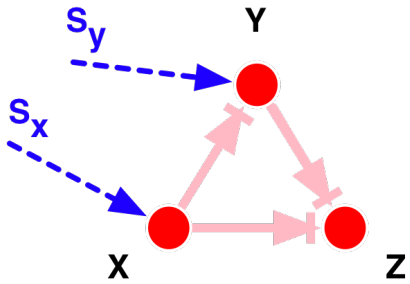
The feed-forward loop



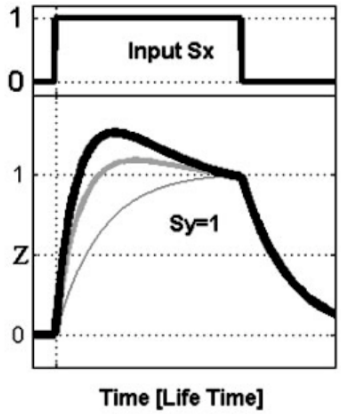
All incoherent loops act as a *accelerator* circuit, relative to a simple XY and XZ co-regulation.



(thin line)

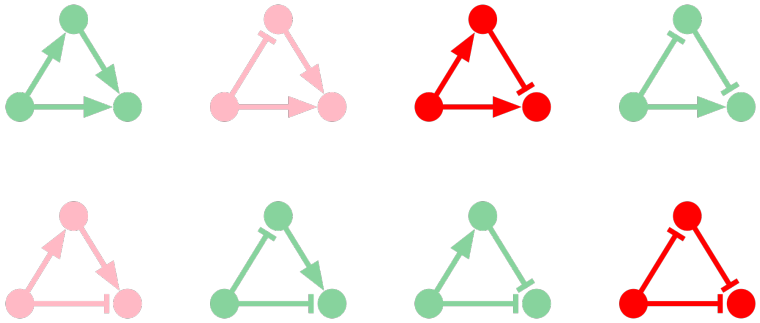


(thick lines)

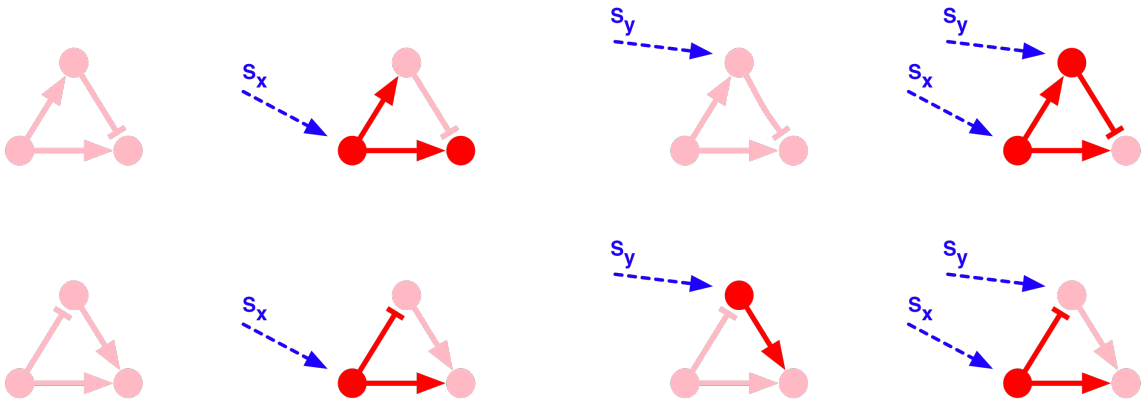


The acceleration is *sign-sensitive*, as it only affects the on-switching.

The feed-forward loop



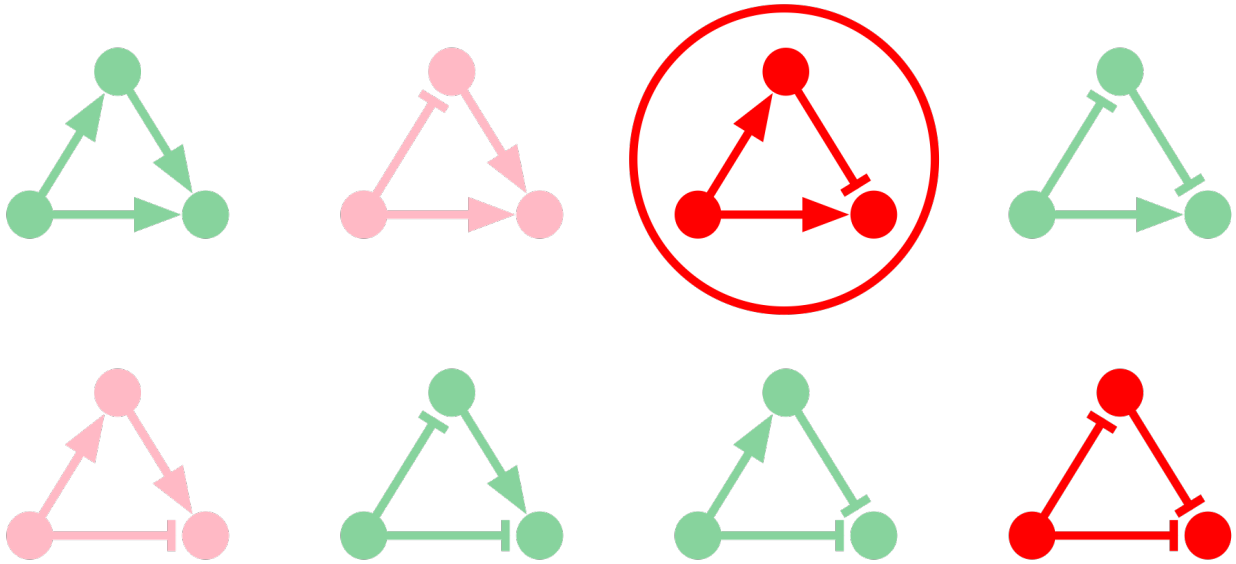
Again only two of the four loops have a steady-state of **Z** that depends on both S_x and S_y. Contrast one of these two with one of the other two:



$\mathbf{Z} = S_x$ AND NOT S_y

$\mathbf{Z} = 0$

The feed-forward loop



Here too we find that one of these two loops with dependency on S_x and S_y is much more prevalent in real transcription networks than all the other incoherent loops.

Evolution and modularity

Biological and engineered networks are very often *modular*, as modular organization makes it easy to construct complex network to solve a given task.

Modular networks are however most often *not optimal* for a given task.

When creating and changing a network through some evolutionary process, we would like modularity to emerge *spontaneously*, i.e. without top-down constraints.

Evolution and modularity

In order to study this question, Kashtan and Alon propose a particular model of evolving networks.

But first of all we need a measure of the modularity for our evolving networks. Recall the Newman-Girvan modularity measure:

$$Q = \Sigma_i \, [e_{ii} - (\Sigma_j \, e_{ij})^2]$$

where e_{ij} is the fraction of edges running between sets i and j. In order to allow for different null models and for the purpose of normalization, Kashtan & Alon define a modified version:

$$Q_m = (Q_{real} - Q_{rand}) \, / \, (Q_{max} - Q_{rand})$$

Kashtan & Alon, PNAS **102**, 13773 (2005).

Evolution and modularity

$$Q_m = (Q_{real} - Q_{rand}) / (Q_{max} - Q_{rand})$$

where

Q_{real} is the actual modularity,

Q_{rand} is the modularity of a randomized version of the network (e.g. with the same degree distribution as the original), and

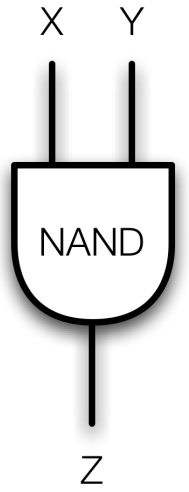
Q_{max} is the maximum modularity achieved with a modularity-maximizing fitness function, over a large number of runs.

Kashtan & Alon, PNAS **102**, 13773 (2005).

Evolution and modularity

The network Kashtan and Alon use is a network of logic gates, and more specifically, NAND gates.

The NAND gate performs the NOT AND operation, and therefore responds to two inputs X and Y with the following Z:

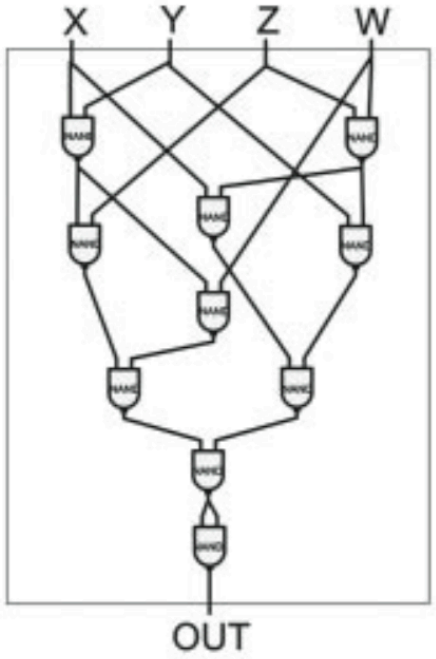


X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

Kashtan & Alon, PNAS **102**, 13773 (2005).

Evolution and modularity

We want this network to solve a task, performed on four inputs X, Y, Z, and W.



We can *evolve* the network by swapping connections and accepting or rejecting those swaps depending on the resulting *fitness* of the network.

The fitness is the fraction of correct output states obtained for all possible 16 input states.

Note that the NAND gate is universal, so that any logical operation can be performed by a network of NAND gates.

Kashtan & Alon, PNAS **102**, 13773 (2005).

Evolution and modularity

Real biological networks, such as the transcription network of the bacterium *E. coli* and the neural network of *C. elegans* show a high modularity Q_m of around 0.5.

However, when we evolve our NAND-network to find a network which implements the task

$$G1 = (X \text{ XOR } Y) \text{ AND } (Z \text{ XOR } W)$$

so that G1 = 0 unless XYZW is 0110, 0101, 1001, or 1010, we get a low modularity of about $Q_m = 0.12$.

Kashtan & Alon, PNAS **102**, 13773 (2005).

Evolution and modularity

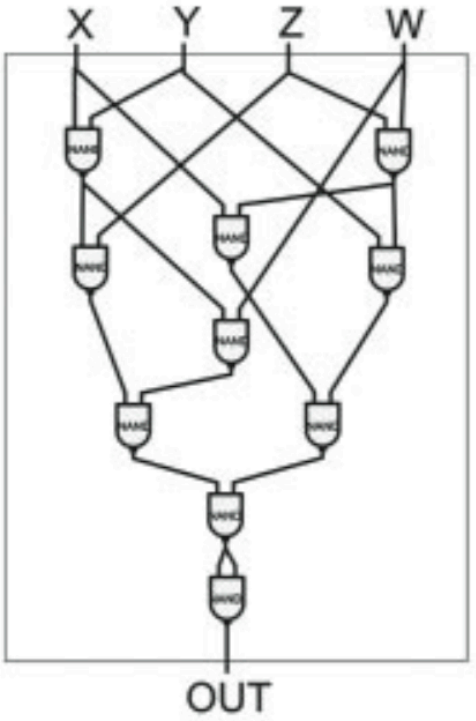
There are many solutions, i.e. many networks that solve this task, all with low Q_m.

One resulting network which solves the task is shown on the right.

Note that our task itself *was* modular:

G1 = (X XOR Y) AND (Z XOR W)

But the network is not.



Kashtan & Alon, PNAS **102**, 13773 (2005).

Evolution and modularity

Now consider the following two evolutionary targets:

$$G1 = (X \text{ XOR } Y) \text{ AND } (Z \text{ XOR } W)$$

$$G2 = (X \text{ XOR } Y) \text{ OR } (Z \text{ XOR } W)$$

which means that G2 = 1 unless XYZW = 0000, 0011, 1100, or 1111.

Let us evolve the network towards *both*…

Kashtan & Alon, PNAS **102**, 13773 (2005).

Evolution and modularity

Now consider the following two evolutionary targets:

G1 = (X XOR Y) AND (Z XOR W)

G2 = (X XOR Y) OR (Z XOR W)

which means that G2 = 1 unless XYZW = 0000, 0011, 1100, or 1111.

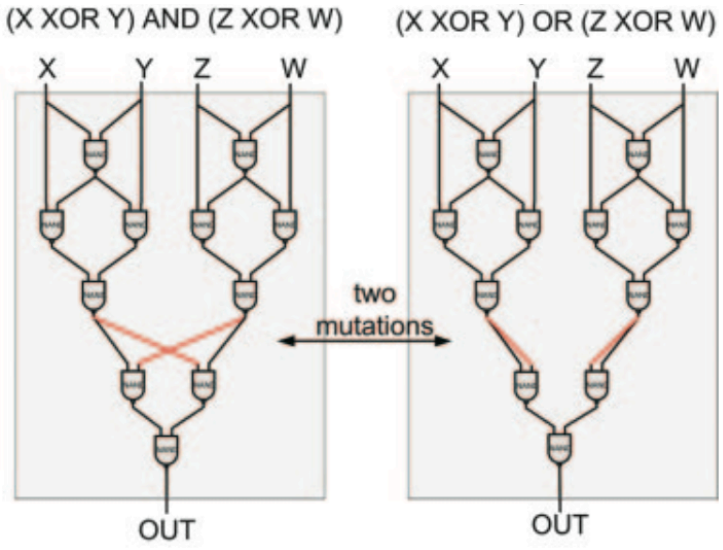
Let us evolve the network towards *both*…

…by *randomly flipping between them!*

Kashtan & Alon, PNAS **102**, 13773 (2005).

Evolution and modularity

The result are two *highly modular* ($Q_m = 0.54$) and *very similar* networks:



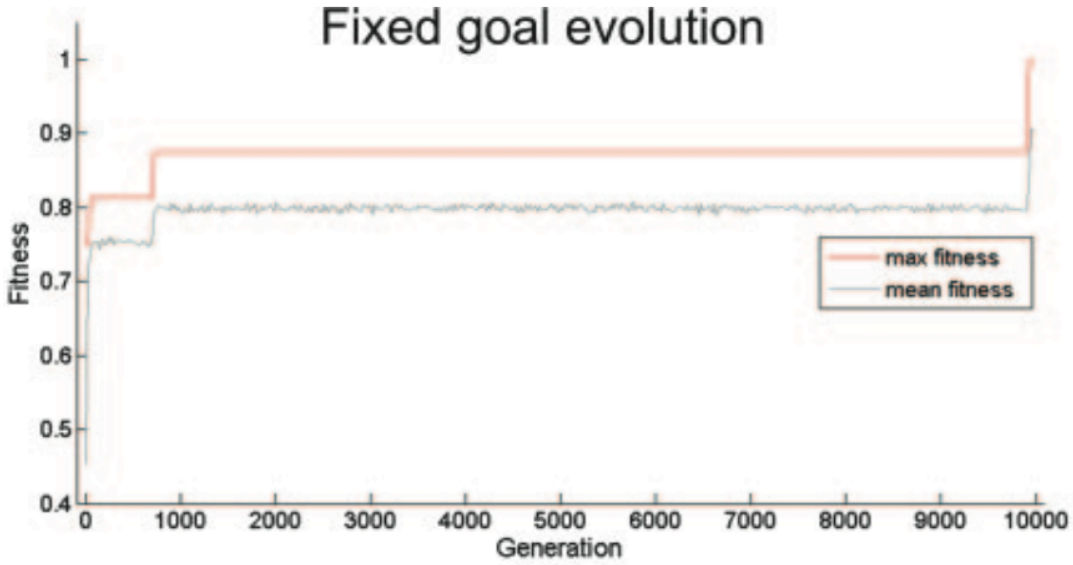
As a result, each time we flip the target, the network adapts within a few iterations.

Kashtan & Alon, PNAS **102**, 13773 (2005).

Evolution and modularity

Interestingly the target flipping also speeds up the overall discovery time of the solution:

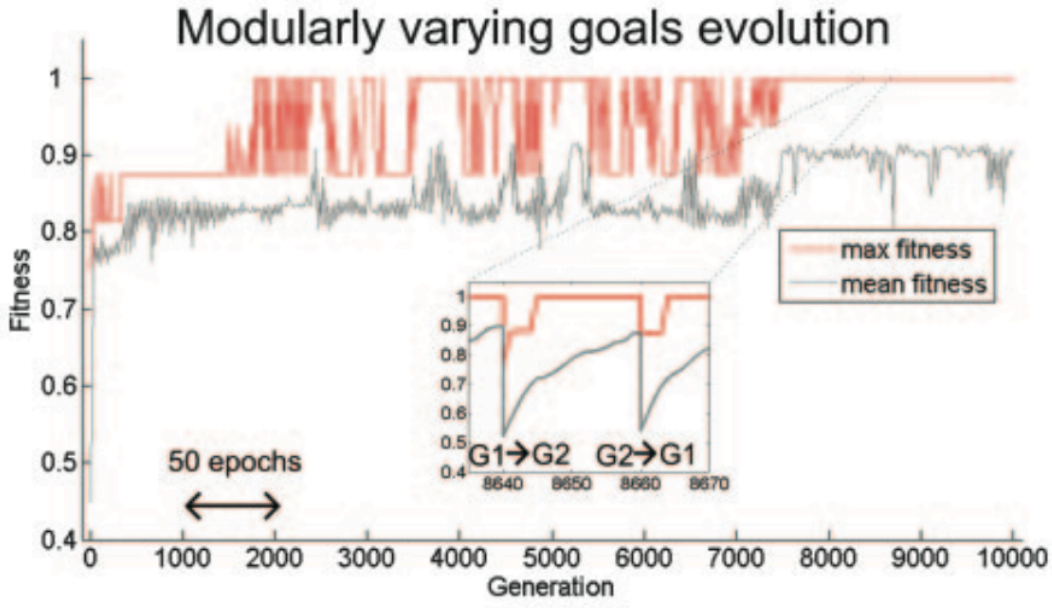
We take about 10000 generations to find a solution for G1 by itself…



Kashtan & Alon, PNAS **102**, 13773 (2005).

Evolution and modularity

…but only around 2000 generations to find *both* G1 and G2 when flipping between them!



Kashtan & Alon, PNAS **102**, 13773 (2005).

Evolution and modularity

Hence modularity appears to spontaneously evolve when a system is faced with a modular, changing environment.

The challenges which a biological organism is likely to encounter are likely to be modular in the sense that they involve similar subtasks and occur repeatedly, with minor variations.

This makes the target-flipping model a feasible representation of an effective response to a changing environment, and a plausible explanation of the modularity observed in biological networks.

Kashtan & Alon, PNAS **102**, 13773 (2005).

Social networks

From 1930s onwards, subject of *Sociometry* develops.

This involves measuring and analyzing social networks, and can be viewed in some way as the birth of network science.

Here we will look at some classic data sets and examine the properties which social networks share.

FIGURE 2

MATRIX OF RELATIONSHIPS IN THE CLUB: THE MATRIX *E*

Individual Number

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0
2	1	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0
3	1	1	0	1	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0
4	1	1	1	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
10	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
11	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
17	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
20	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
22	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
28	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
29	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
31	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
32	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
33	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	1	0	1	0	1	1	0	0	0	0	0	1	1	1	1
34	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	1	0	0	1	1	1	0	1	1	0	0	1	1	1	1	1	0

This is the matrix representation of the graph shown in Figure 1. The rows and columns represent individuals in the club. An entry, determined by a row/column pair, is valued at 1 if an edge was drawn in Figure 1 between the two individuals represented by the row and column. The entry is valued at 0 otherwise. For example, the entry in the fifth row, seventh column, is a 1, indicating there is an edge existing between individuals 5 and 7. Notice that the matrix is symmetrical--i.e., the seventh row fifth column is also valued at 1, since it identifies the same edge. Because, by definition, no individual can interact with himself, zeroes occupy the main diagonal (from row/column 1 to row/column 34). Later, this matrix will be referred to by the symbol *E*.

from: Zachary, J. Anthropol. Res. ***33****, 452 (1977)*

Social networks

Zachary Karate Club data set

Wayne W. Zachary published an article in 1977 describing a Karate Club whose members formed two factions.

This was because they disagreed whether their instructor should receive a pay rise.

After the instructor was fired, the club split as some joined him at a new club.

Social networks

Properties of the Zachary Karate Club data set:

34 nodes = people
78 *undirected* connections = friendships*

*defined as consistent social interactions outside the club.

Note that while there were about 60 members in the club, only 34 had friends within the club, leaving the other members as *disconnected nodes* in the graph (and therefore irrelevant).

Social networks

In the original paper, Zachary also produced a *weighted* version of the network, recording the strength of interactions between individuals.

He then used the *maximum-flow-minimum-cut* algorithm to (successfully) predict the two parts which the club would split into.

Newman and Girvan (2002) managed to predict the split for the unweighted version using their community detection algorithm.

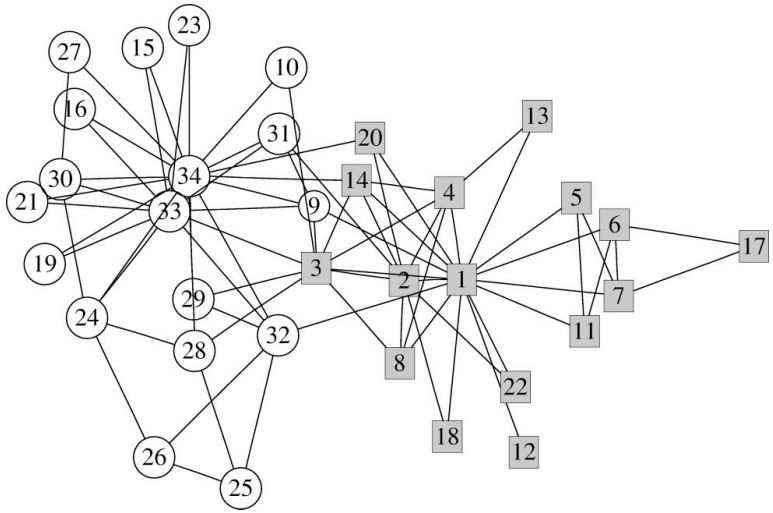
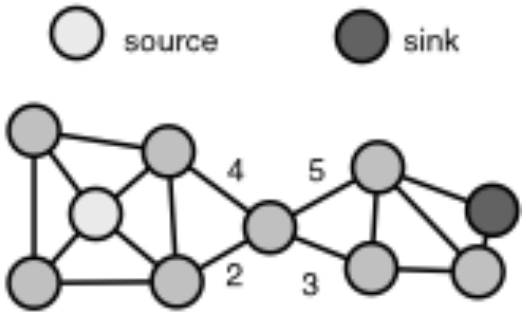


*Image: Newman and Girvan, PRL **69**, 026113 (2004)*

Max-flow-min-cut

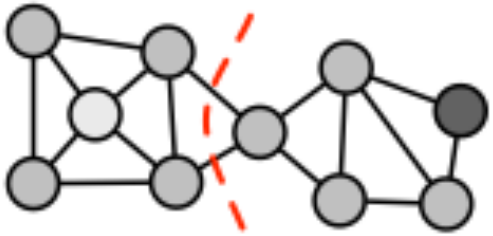
The *maximum-flow-minimum-cut* or *max-flow-min-cut* theorem simply states that the flow in a network is limited by the smallest bottleneck.

A *cut* is a set of edges which separates the nodes into two sets, one containing the source and one containing the sink.



The smallest bottleneck corresponds to the *minimum cut*.

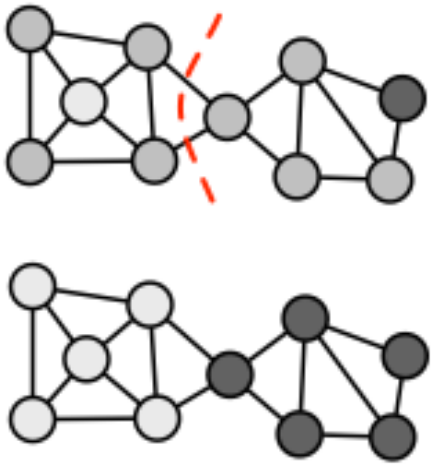
In unweighted networks the size of a cut is the number of edges.



In a weighted network the size of a cut is the sum of the edge weights.

Max-flow-min-cut

The *maximum flow* between source and sink across the whole network cannot exceed the capacity of the *minimum cut*.



The minimum cut is what Zachary used to predict the split of the Karate Club.

Social networks

In some cases, social networks are also *directed*, e.g.:

- Study by Bruce Kapferer of interactions in an African tailor shop with 39 nodes, where friendship interactions (undirected) and work-related interactions (directed) were studied.

- Study by McRae of 67 prison inmates, in which each inmate was asked to name other prisoners he was friends with. This matrix too is directed.

Generally speaking even directed social networks usually turn out to be fairly symmetric, which is not too surprising.

If people are free to choose whom they interact with they most likely will not bother with someone who does not reciprocate the interaction.

Bipartite graphs

Bipartite graphs have two types of nodes and there are no edges between the same type of node.

Bipartite real-world networks include collaboration networks between scientists (papers), actors (films), and company directors (boards).

Often these networks are converted using a *one-mode projection* with fully connected subgraphs.

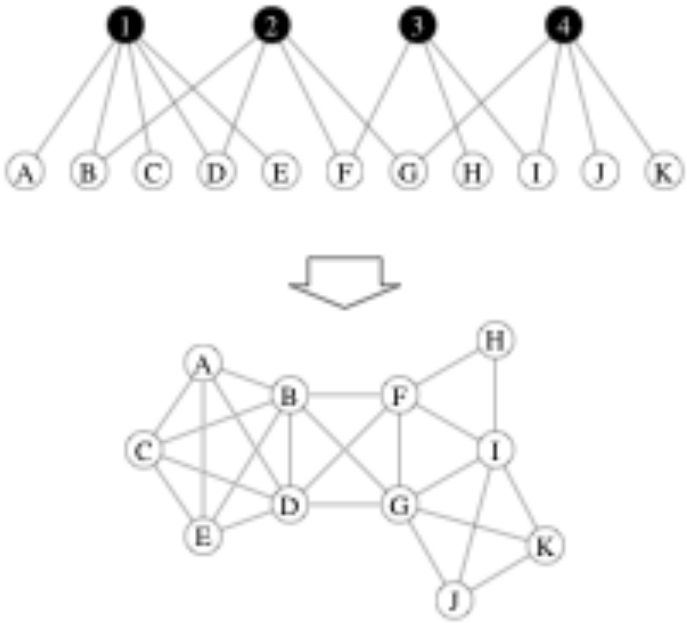


Image: Newman et al., PRE ***64****, 026118 (2001)*

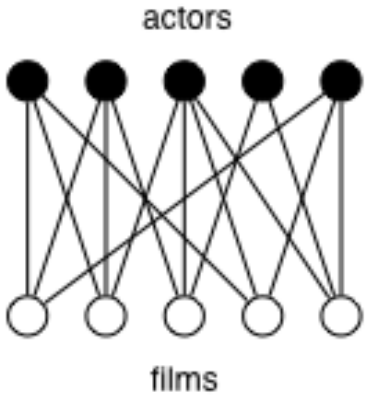
Collaboration networks

A particular class of social networks are *collaboration networks*.

These are bipartite graphs because we have:

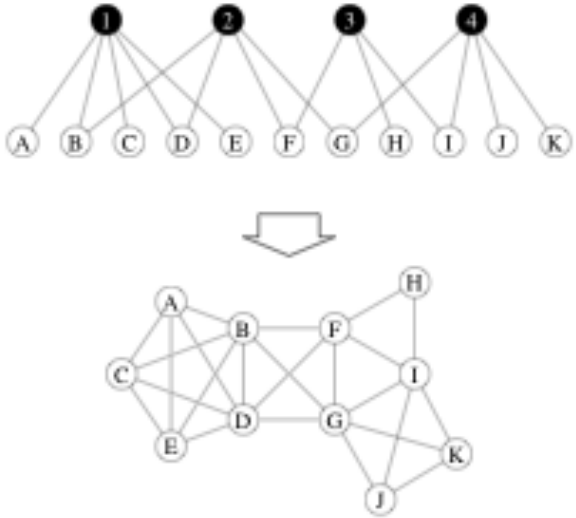
a) *People*, who belong to

b) *Collaborations*, such as films, scientific papers or company boards.



Collaboration networks

In order to analyze them we transform them into a simple network between people by *connecting all members of a collaboration to each other*.



This is why collaboration graphs have a *high clustering coefficient*.

Image: Newman et al., PRE ***64****, 026118 (2001)*

Collaboration networks

Collaboration networks however also show *short average path lengths*.

This, together with their high clustering coefficient makes them *small-world* networks.

They are *not* scale-free however, and seem to closely match models with a scale-free distribution with an expontial cutoff:

$$P(k) = k^{-\gamma} e^{-k/k_0}$$

The finite cutoff may reflect the finite size of the time window from which the data is collected.

Collaboration networks

Finally, recall that collaboration networks are *assortative*, meaning that highly connected nodes are connected to other highly connected nodes.

This is quite unusual - many real-world networks are *disassortative*, as high-degree nodes connect to low-degree ones.

Network	n	r
Physics coauthorship (a)	52 909	0.363
Biology coauthorship (a)	1 520 251	0.127
Mathematics coauthorship (b)	253 339	0.120
Film actor collaborations (c)	449 913	0.208
Company directors (d)	7 673	0.276
Internet (e)	10 697	−0.189
World-Wide Web (f)	269 504	−0.065
Protein interactions (g)	2 115	−0.156
Neural network (h)	307	−0.163
Marine food web (i)	134	−0.247
Freshwater food web (j)	92	−0.276
Random graph (u)		0
Callaway *et al.* (v)		$\delta/(1+2\delta)$
Barabási and Albert (w)		0

*Image: Newman, PRL **89**, 208701 (2002)*

Social networks: Summary

Social networks tend to be *undirected* even if the direction is actually recorded.

Collaboration networks form an important subset of social networks.

They are originally *bipartite,* and their one-mode projection is:

- *small-world*
- *assortative*
- *not scale-free*

Collaboration networks are studied much more than other social networks because it is easy to gather large data sets of this kind.

Biological networks

There are many different types of networks in biology:

- Transcription networks
- Protein-protein interaction networks
- Metabolic networks
- Neural networks
- Food webs

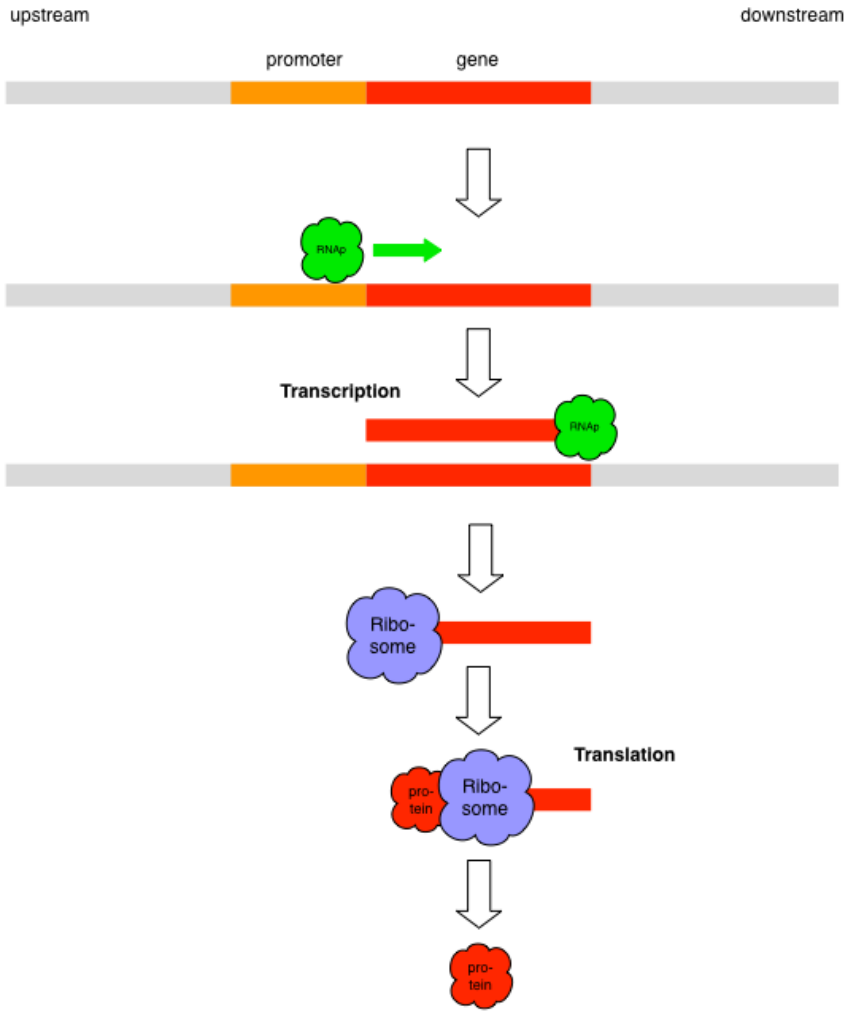
… among others.

Transcription networks

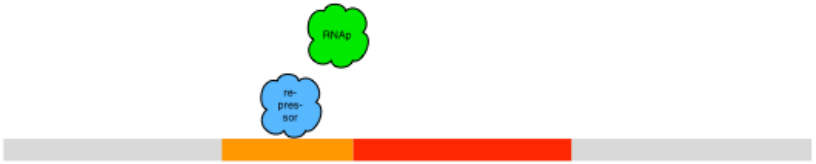
The DNA of every living organism is organized into *genes* which are *transcribed* and then *translated* into *proteins*.

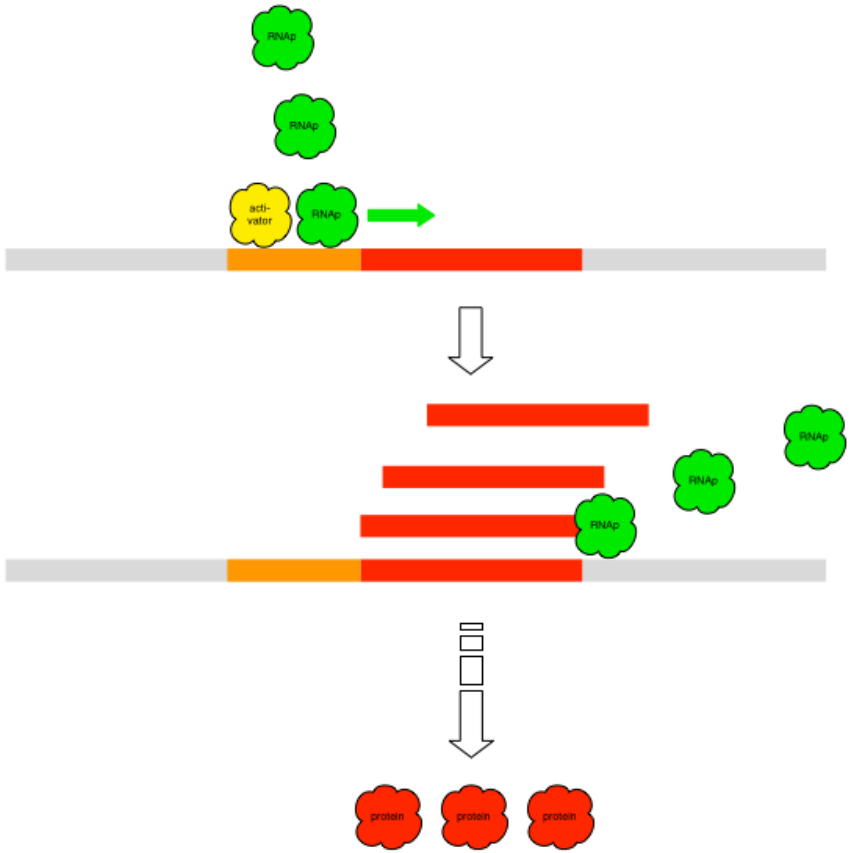
Transcription is performed by the *RNAp* molecule which binds to the *promoter* region and produces a copy of the gene, called *mRNA*.

The *ribosome* then translates the mRNA into a *protein*.



Transcription networks





Whether or not a gene is transcribed at a given point in time depends on proteins called *transcription factors,* which bind to the promoter region.

Activators enhance transcription while *repressors* inhibit it.

Transcription networks

Since transcription factors themselves are also proteins encoded by genes we can get a transcription factor which activates another transcription factor, etc.

Hence we can construct a network where the nodes are genes and a *directed* edge

$$X \to Y$$

means that the product of gene X is a transcription factor which binds to the promoter region of gene Y, or shorter, that

gene X controls the transcription of gene Y.

This is the kind of network we studied in the context of the feed-forward loop.

Transcription networks

The in-degree and out-degree distributions of transcription networks are very different.

Some transcription factors regulate large numbers of genes, and are called *global regulators*. This means we can get *high out-degrees*.

In fact, the out-degrees follow a scale-free distribution $P(k) \sim k^{-\gamma}$.

On the other hand, no gene is regulated by many other genes. Therefore there we only get *low in-degrees*.

Transcription networks

And as we know, feed-forward loops are particularly prevalent in transcription networks.

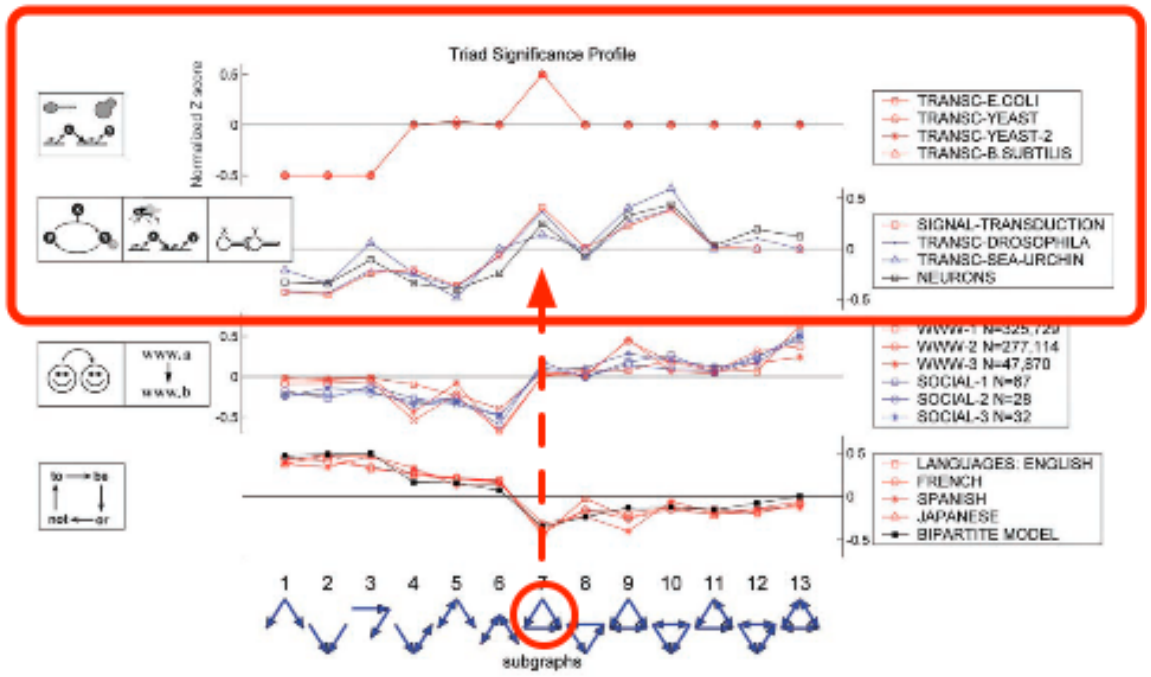


Image: Milo et al., Science ***330****, 1538 (2004)*

Accelerating networks

Many organizational and regulatory networks require *global integration*. What this means is that, as the network grows, we have to make sure that it remains highly connected.

Examples include gene regulatory networks, supercomputer wirings, and stock exchanges.

If the connectivity of a network, measured as the fraction of possible edges that are realized, is to remain *constant*, then this means that the number of connections between N nodes has to grow proportional to the number of possible edges, which is $N(N\text{-}1)/2$.

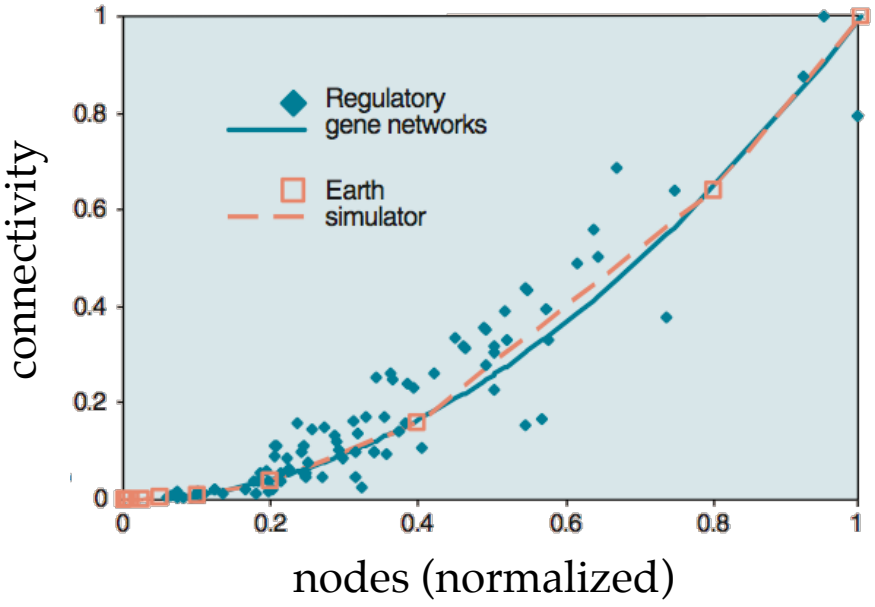
In other words, the number of edges has to grow *quadratically* with the number of nodes. Such networks are called *accelerating networks*.

Accelerating networks

This quadratic growth is indeed observed in:

a) supercomputers

b) gene regulatory networks (number of regulatory genes as a function of the total number of genes) in single-cell organisms



Mattick & Gagen, Science **307**, 856 (2005)

Accelerating networks

What this means however, is that such networks are likely to hit a *growth ceiling*.

This is because in any physical network the nodes have a *finite capacity to connect* to other nodes. In other words the network has a maximum degree k_{max}.

Hence, if the number of edges is given by $E = \alpha N^2$, the average degree is given by

$$k_{av} = 2E/N = 2\alpha N$$

and since $k_{av} \leq k_{max}$ the network cannot grow larger than

$$N_{max} = k_{max}/2\alpha$$

Accelerating networks

Mattick and Gagen suggest that this growth ceiling is overcome by technological or biological innovation.

Most biological organisms fall into two basic categories, *prokaryotes* and *eukaryotes*.

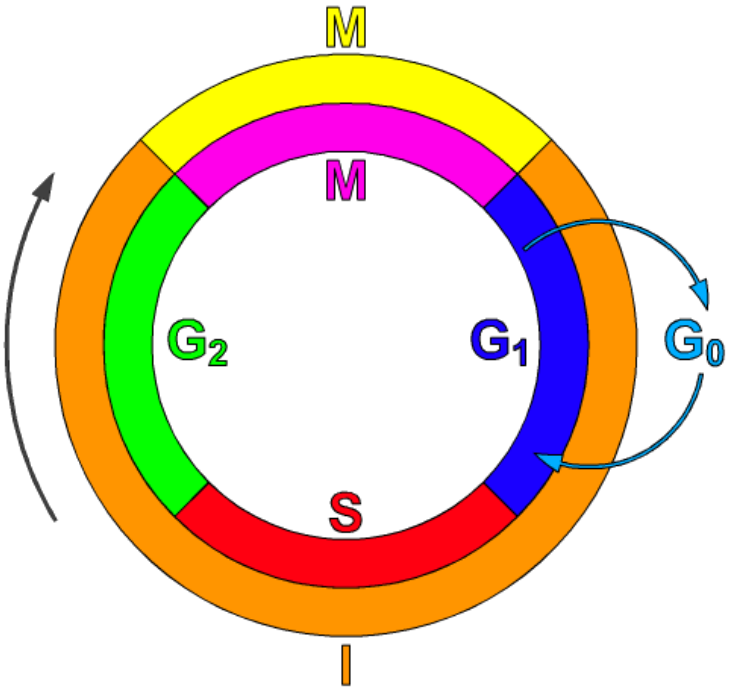
Prokaryotes are single-cell organisms such as bacteria.
Eukaryotes are (more or less) everything else, including us.

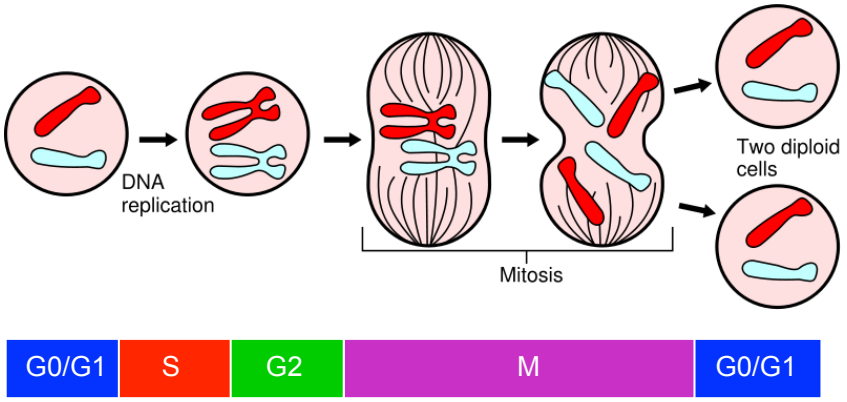
Prokaryotes show a quadratic growth of the number of regulatory genes. Mattick and Gagen claim that eukaryotes developed when prokaryotes hit the growth ceiling, forcing the development of new regulatory mechanisms, such as the use of non-coding DNA.

Mattick & Gagen, Science **307**, 856 (2005)

Boolean cell-cycle network

The fundamental cycle of cell division that occurs in all living matter is termed the *cell-cycle*.

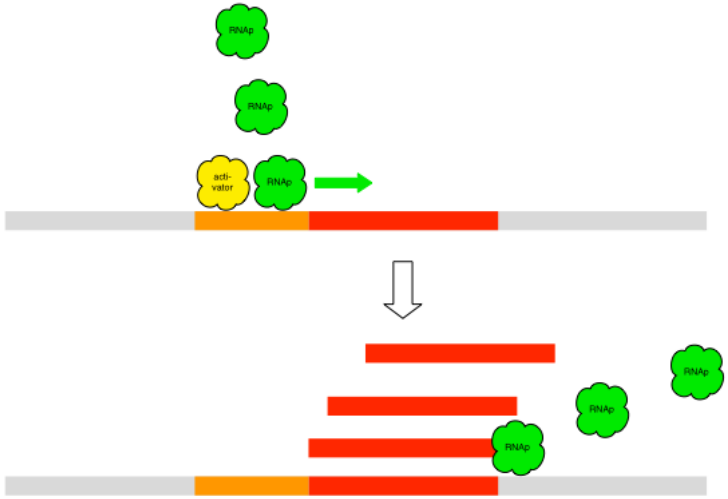


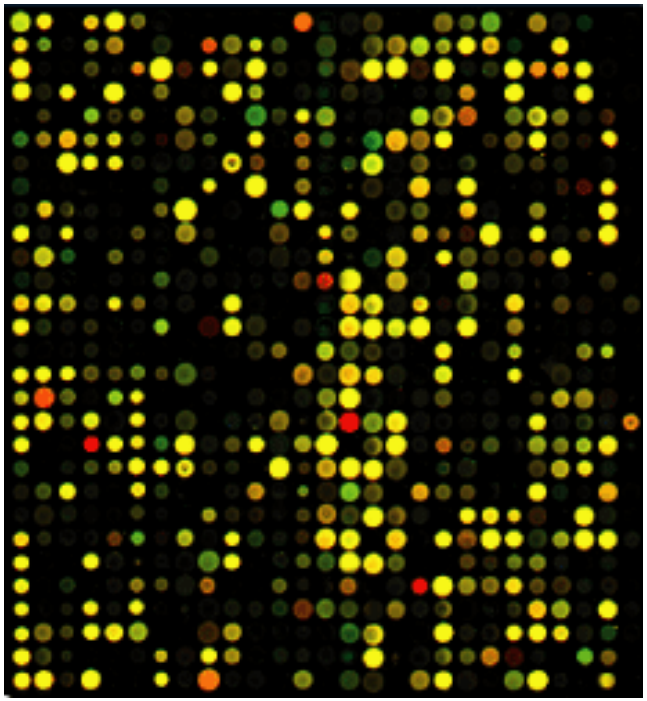


Images: Wikipedia

Boolean cell-cycle network

Underlying the cell-cycle is a regulatory network of genes, switching each other on and off. We can measure which genes are switched on at which times, using *microarrays*.

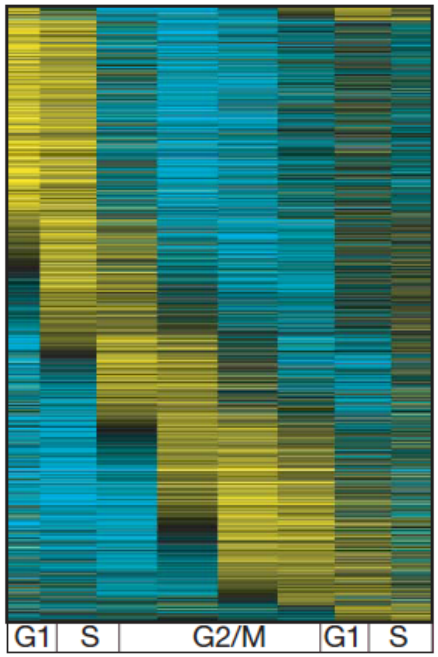




Boolean cell-cycle network

These measurements tell us that a number of genes are activated at each stage of the cell-cycle, and that these genes are activated in succession.

rows = genes

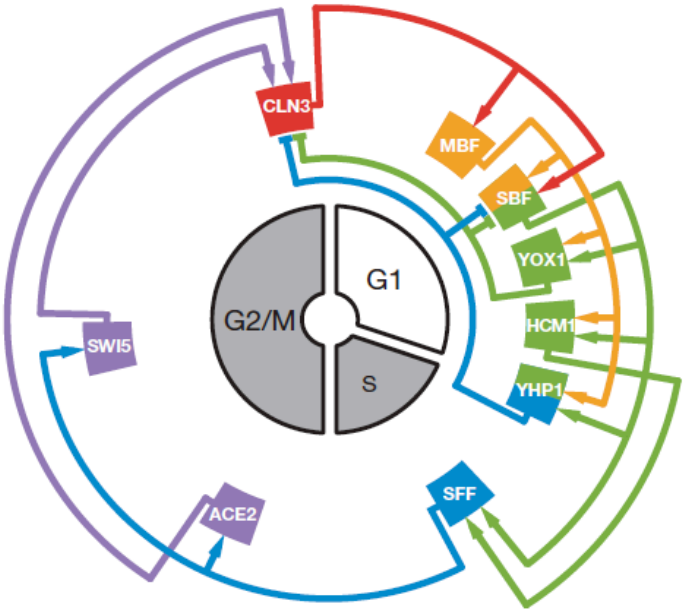


Boolean cell-cycle network

The protein cyclin-CDK has long been thought to play a crucial role in regulating the cell cycle oscillations.

However, if we knock out cyclin-CDK, we find that 70% of genes oscillating with the cell-cycle still oscillate.

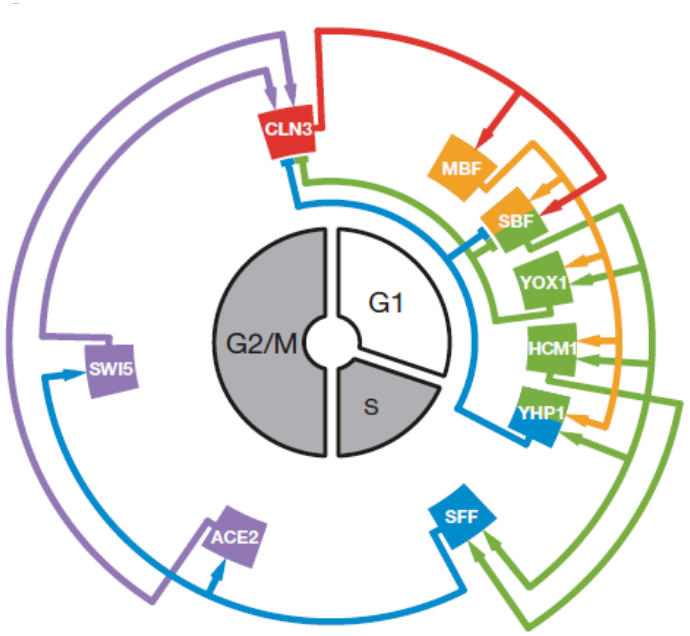
To model this cyclin-independent oscillation, we can isolate the transcription factors among these genes that oscillate independently of cyclin, and draw a directed network of regulation.



Boolean cell-cycle network

These nine genes switch on successively, in five distinct groups (different colours).

By casting this network as a Boolean network with (mostly) AND functions, one finds a robust attractor cycle of size five which dominates the state space with an attraction basin covering 80% of the 512 states.



Protein-protein networks

In protein-protein networks we are interested in the direct interactions between proteins.

Unlike transcription networks, protein-protein networks are *undirected*.

They have a *scale-free* degree distribution, and therefore a small number of highly connected nodes, or *hubs*.

These hubs have been shown experimentally to correspond to biologically essential proteins. Removing these is lethal for an organism.

This is often referred to as the equivalence of *lethality* and *centrality* in proteins, where centrality here is simply the degree.

Protein-protein networks

We can distinguish two types of hubs in protein-protein interaction networks:

Party hubs, which interact with several other proteins *simultaneously*.

Date hubs, which interact with several other proteins *sequentially*.

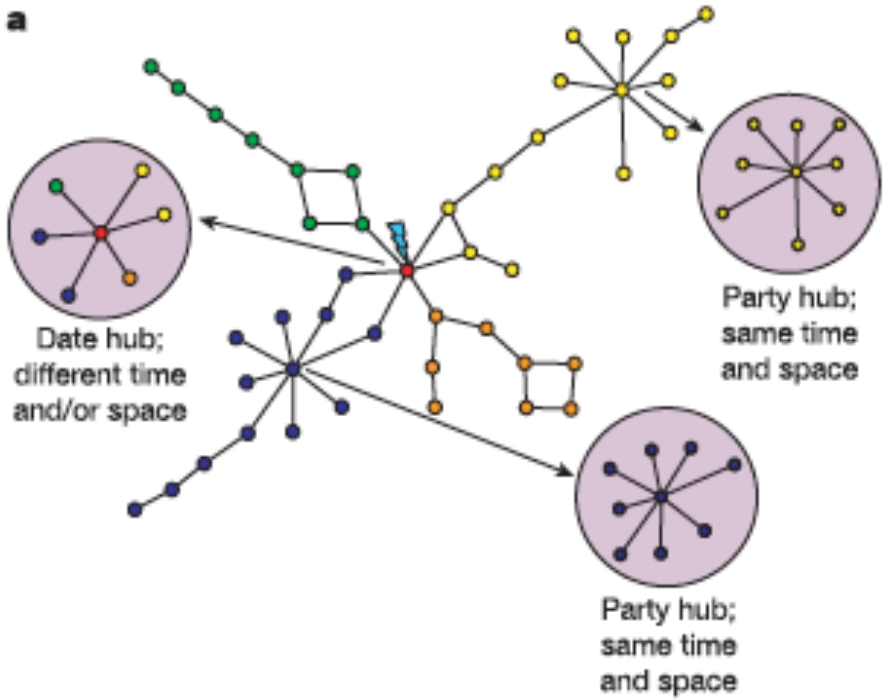


*Image: Han et al., Nature **430**, 88 (2004)*

Protein-protein networks

We can distinguish party hubs and date hubs by looking at a set of confirmed protein-protein interactions and observing which pairs of genes are expressed together.

The similarity of gene expression is measured using the Pearson correlation coefficient.

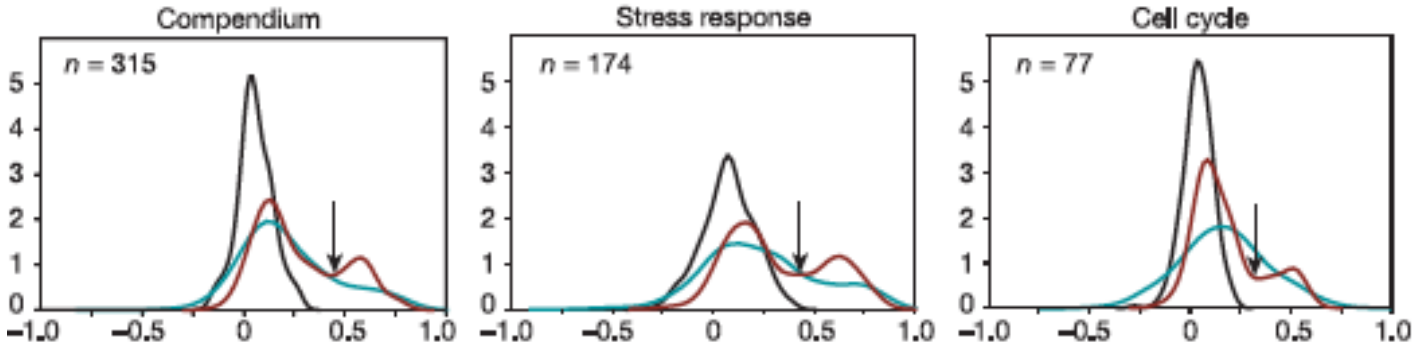


*Image: Han et al., Nature **430**, 88 (2004)*

We observe a bimodal distribution for proteins of degree $k > 5$ which indicated a separation of date hubs (low similarity) and party hubs (high similarity).

Metabolic networks

Metabolic networks are networks of *molecular interactions within the biological cell*, which makes them very general.

By comparing these networks for 43 organisms, Barabasi *et al.* established that they have

- a *scale-free degree-distribution*, but also
- a *high clustering coefficient* scaling as $C(k) \sim k^{-1}$, which suggests modularity.

In order to explain the discrepancy they came up with the model of *hierarchical networks*, which we discussed in lecture 2.

Neural networks

The complete neural network of the worm *C. elegans* has been mapped, giving valuable insights into the topology of real neural networks.

It is a *directed* network of 280 nodes and 2170 edges.

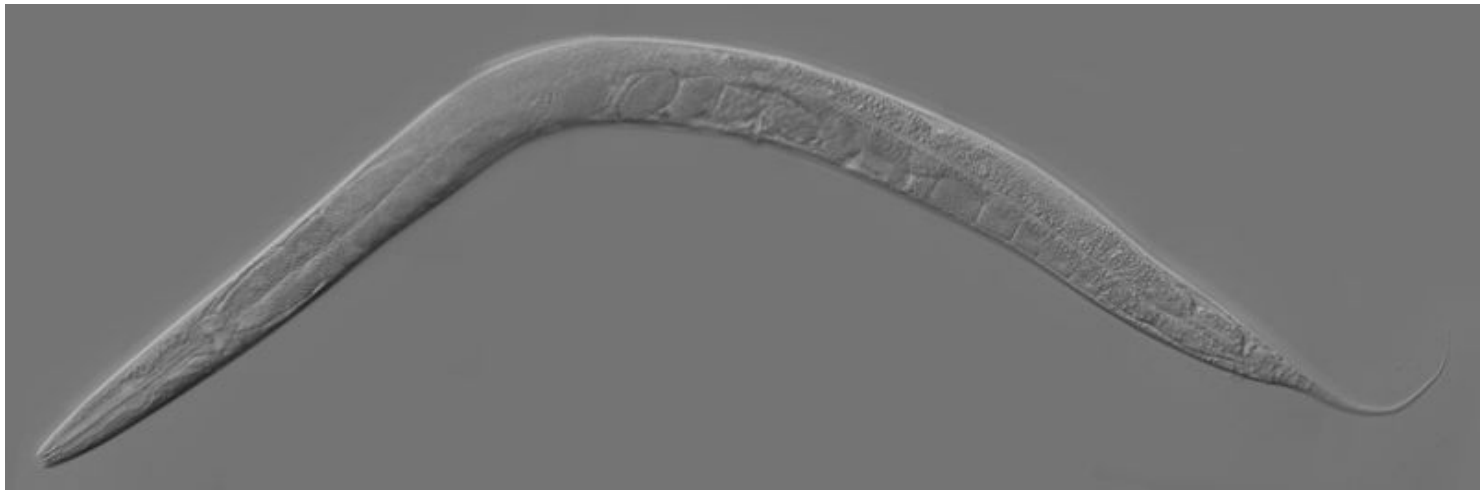


Image: Wikipedia

Neural networks

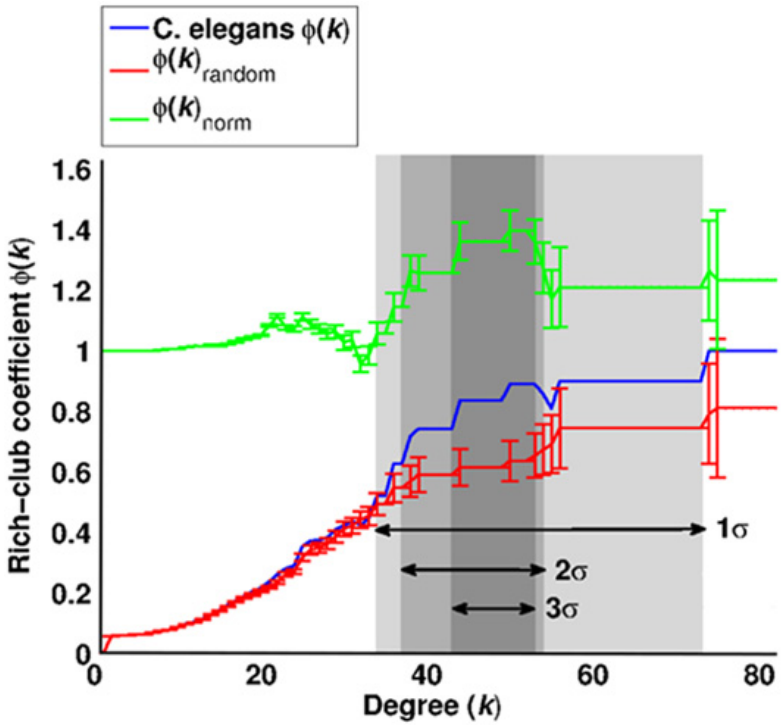
The network falls into the superfamily of transcription and signal transduction networks with a *high frequency of feed-forward loops*.

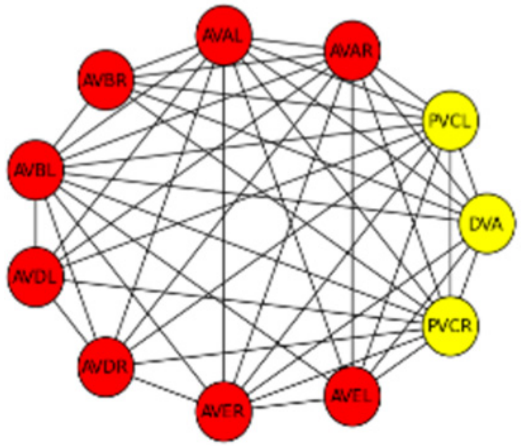
This makes sense as neural networks, like transcription networks, are also complex control circuits.

The neural network of *C. elegans* is also *small-world* as it has a *high clustering coefficient* and a *short average path length*.

Rich clubs

The neural network of the worm *C. elegans* also has a rich club. We can think of this as a control center of the neural network.

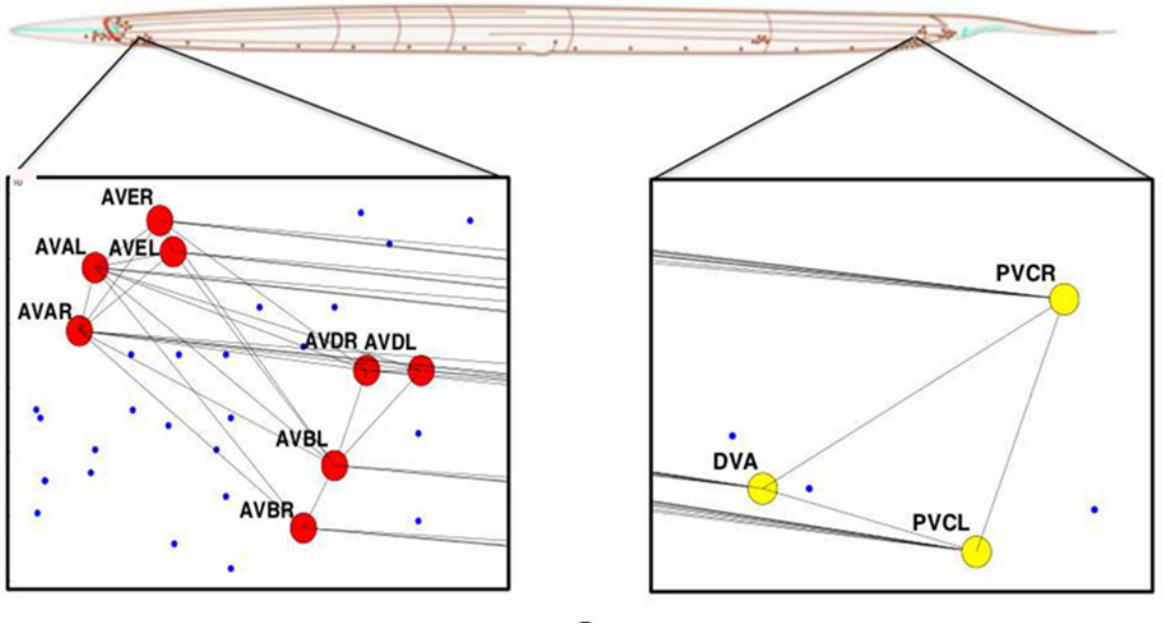




Towlson et al., J. Neurosci. 33, 6380 (2013).

Rich clubs

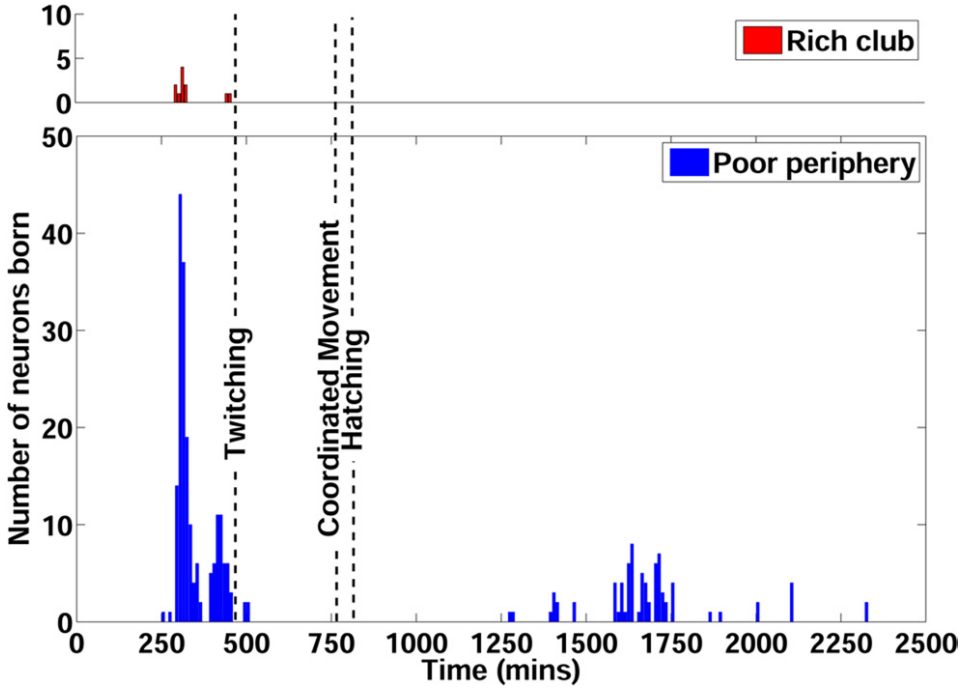
Interestingly the rich club is split between the head and the tail of the worm. At first sight this is odd as it requires many long-range connections, which are ‘expensive’ to build and maintain.



Towlson et al., J. Neurosci. 33, 6380 (2013).

Rich clubs

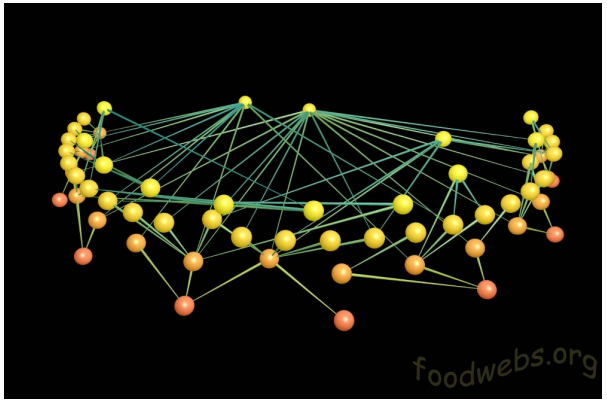
But it turns out that the rich club is fully formed before the worm hatches and grows. Furthermore the work starts moving once the rich club is complete, but long before all of the motor neurons are created.



Towlson et al., J. Neurosci. 33, 6380 (2013).

Food webs

Food webs are ecological networks in which the nodes are species and *directed* edges signify which species eats which other species.



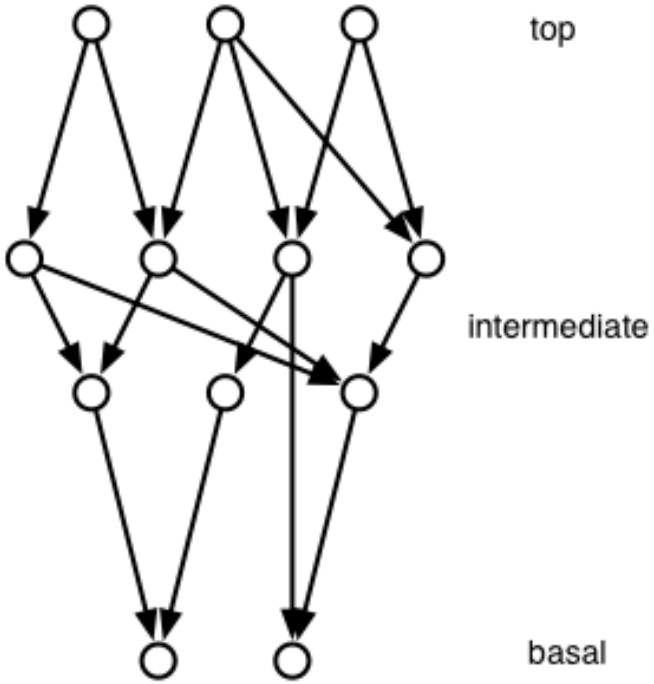
Typically these networks have tens or hundreds of nodes and hundreds or thousands of connections.

(Picture: UK Grassland Food Web, www.foodwebs.org)

Food webs

In food webs we have:

- **top level species** which are *purely predators* and thus have *in-degree zero*,
- **intermediate species** which are *both predator and prey*, and which have *non-zero in- and out-degree*, and
- **basal species** which are *only prey*, and which therefore have *out-degree zero*.



Food webs

Food webs are characterized by set of properties, such as:

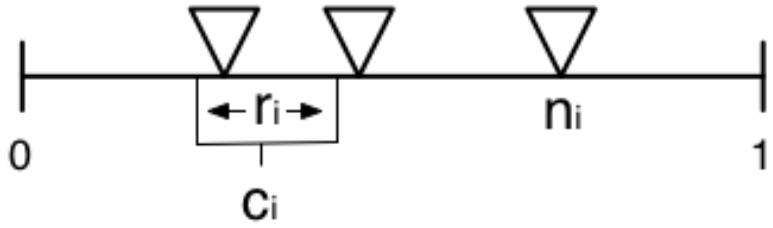
- the *fraction* of top, intermediate and basal species
- the standard deviation of *generality* and *vulnerability*, which are out-degree and in-degree, divided by average degree.
- the *number*, *mean length* and *standard deviation of the length* of food chains
- the fraction of species that are *cannibals* or *omnivores*.

All of these properties of networks can be reproduced using a simple model known as the *niche model*.

Food webs

The niche model maps the hierarchy of species to the unit interval and allows a model food web with N species and E edges to be constructed by drawing, for each species:

- a random number n_i uniformly between 0 and 1.
- a random number r_i between 0 and 1 from a beta distribution with mean E/N^2 (= overall connectivity).
- a random number c_i between $r_i/2$ and n_i.



The species i at n_i eats species in the range r_i, centred around c_i.

Biological networks: Summary

Transcription networks:
directed, low in-degree, scale-free out-degree, feed-forward loops

Protein-protein networks:
undirected, scale-free, 'party hubs' and 'date hubs'

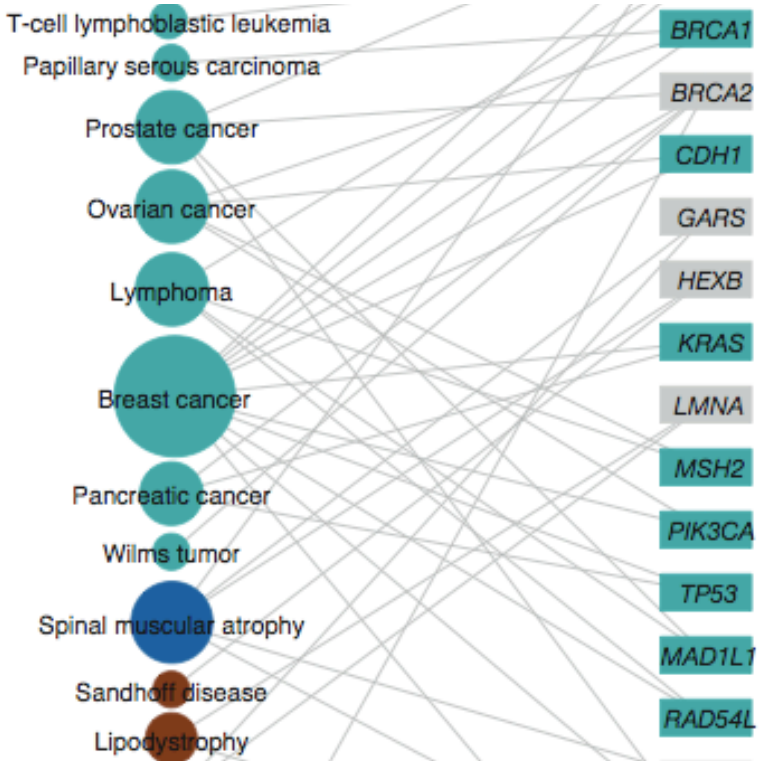
Metabolic networks:
undirected, scale-free, high clustering coefficient, modular, 'hierarchical'

Neural networks:
directed, small-world, feed-forward loops

Food webs:
directed, three-tier structure, predicted well by niche model

Human Disease Network

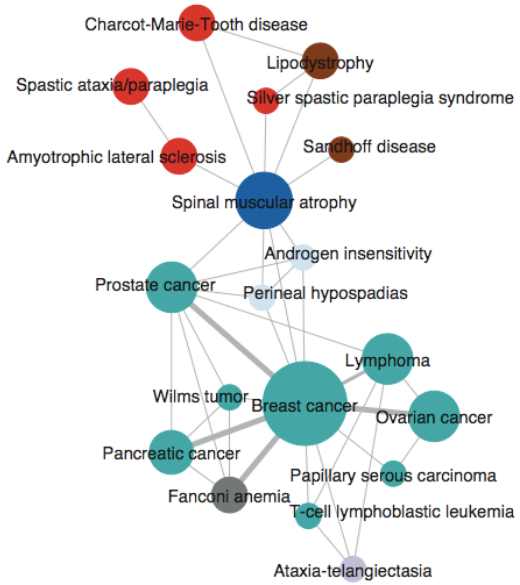
The bipartite network of *diseases* and *disease-related genes* is also known as the *diseasome*.



Goh et al. PNAS **104**, 8685 (2007).

Human Disease Network

The one-mode projection of the diseasome onto diseases gives us a *weighted* network of diseases, in which the weights indicate the number of shared disease genes.

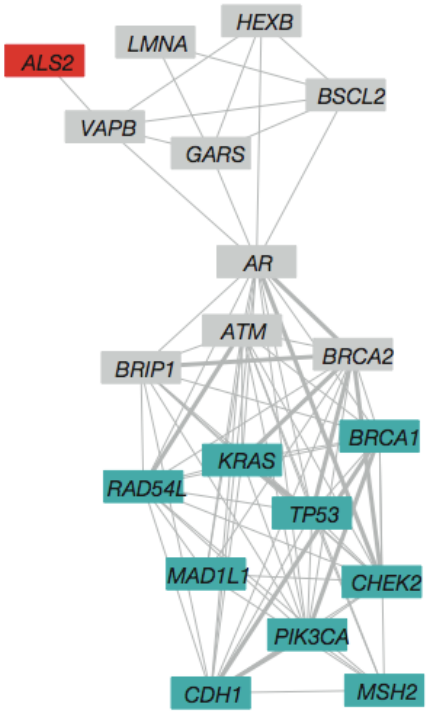


This is the *Human Disease Network* (HDN).

Goh et al. PNAS **104**, 8685 (2007).

Human Disease Network

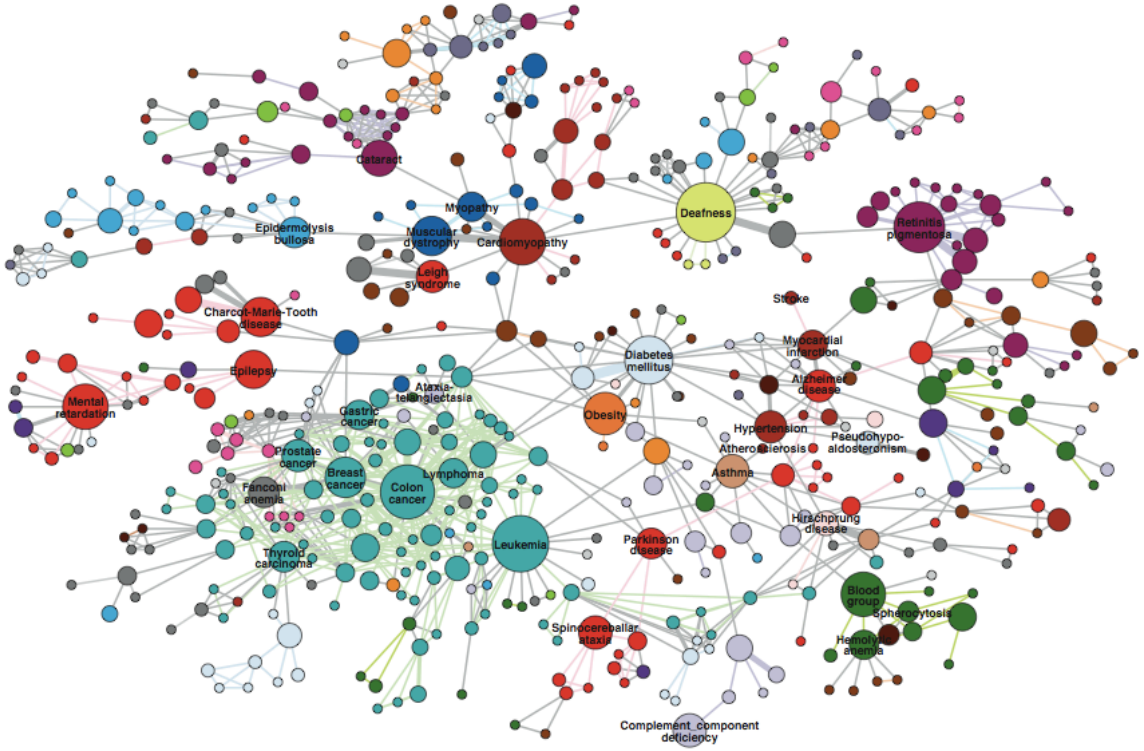
We can equally perform a one-mode projection onto disease genes, and create a *Disease Gene Network* (DGN).



Goh et al. PNAS **104**, 8685 (2007).

Human Disease Network

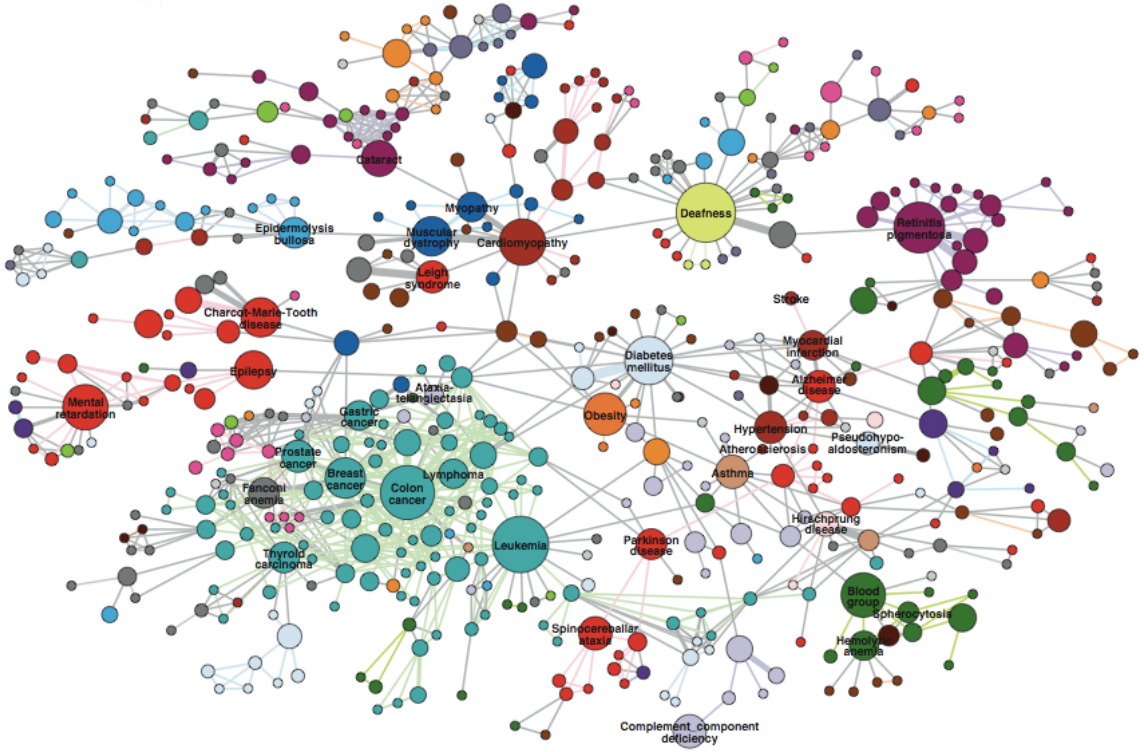
The Human Disease Network forms a giant component with 516 out of 1284 disorders.



Goh et al. PNAS **104**, 8685 (2007).

Human Disease Network

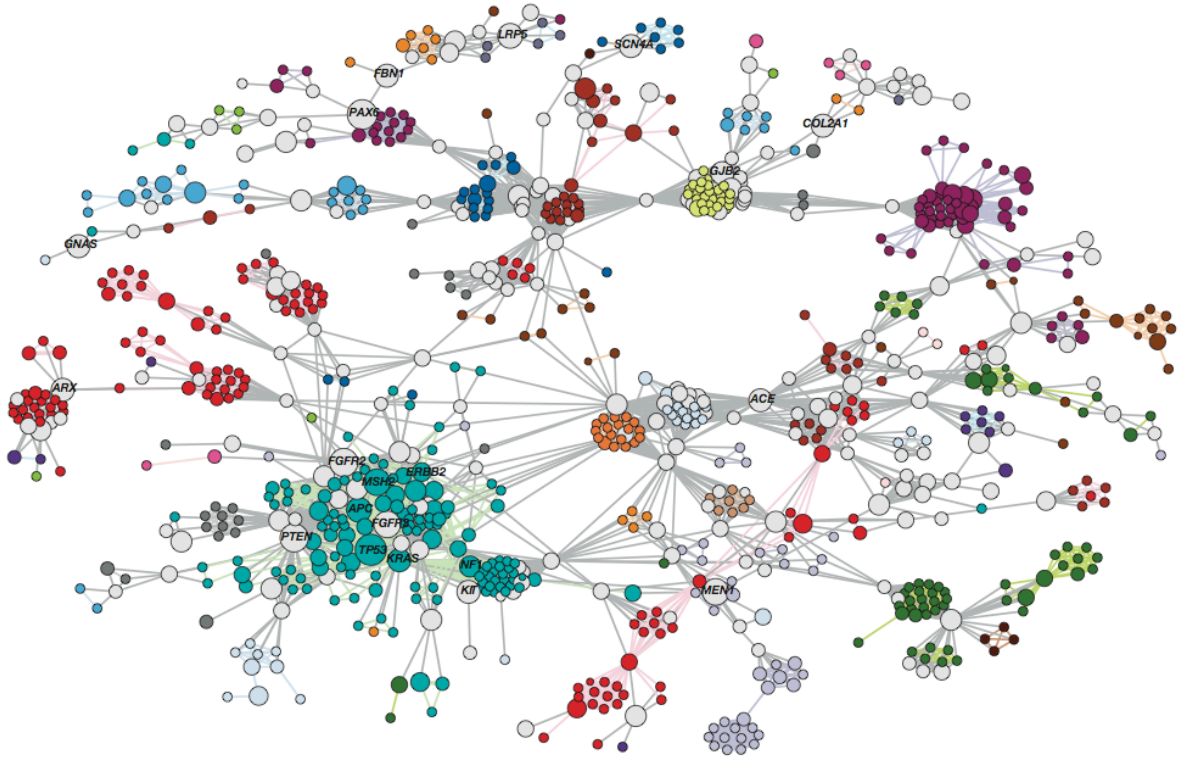
There are eight times as many connections between disorders of the same class than we would expect by chance.



Goh et al. PNAS **104**, 8685 (2007).

Human Disease Network

Looking at the Disease Gene Network, we find that ten times as many interactions as would be expected by chance are shared between this network and a network of protein-protein interactions.



Goh et al. PNAS **104**, 8685 (2007).

Human Disease Network

Recall that in protein-protein networks the biological *essentiality* of proteins is correlated with their *degree* in the network - we discussed this under 'Vulnerability'.

However, most disease genes are non-essential, and have a low degree, making them *peripheral* in the network.

A likely reason for this is that an essential disease gene with a central position in the protein-protein interaction network would be too disruptive, and prevent the carrier of the disease from surviving long enough to pass on the gene.

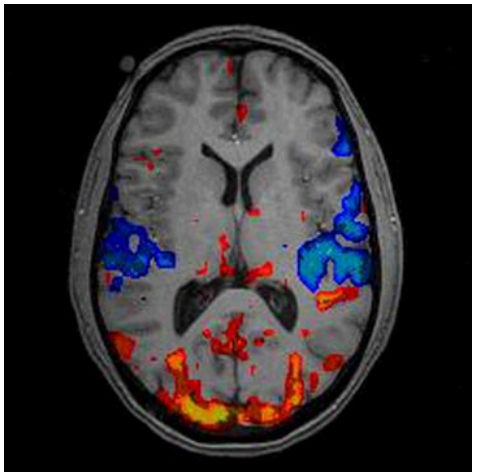
This hypothesis is confirmed by an exception to this rule: Disease genes activated by mutations *during* the life of the organism, such as some cancer-related genes, are more likely to be high-degree.

Goh et al. PNAS **104**, 8685 (2007).

Functional Brain Networks

We have already encountered *structural* neural networks in C. elegans. In recent years neurologists have started to investigate *functional* neural networks, and in particular, *functional brain networks*.

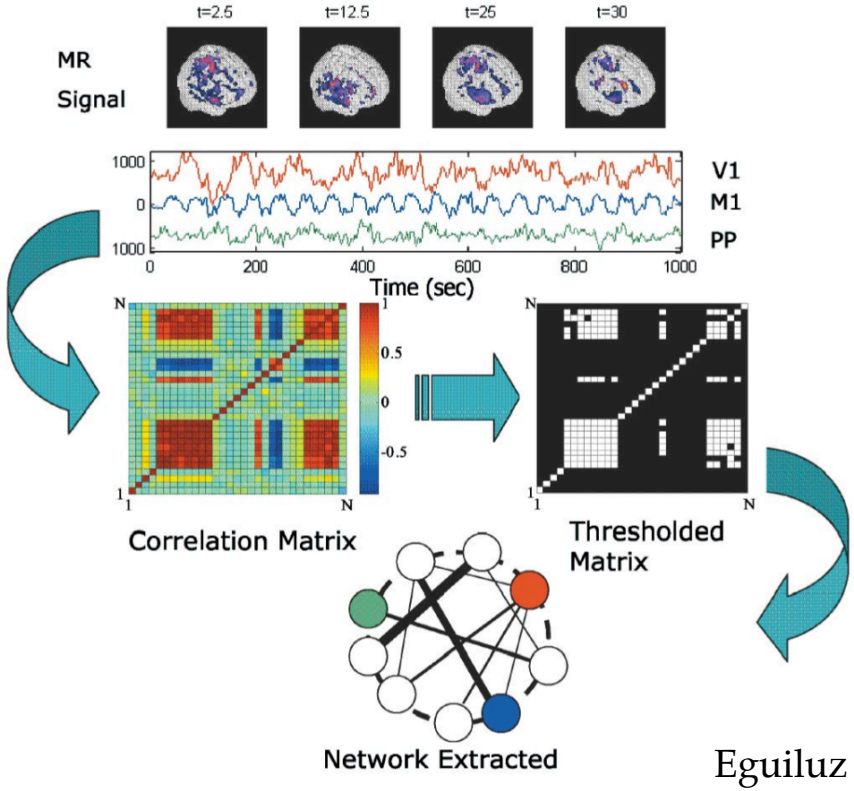
These are often constructed using *functional magnetic resonance imaging (fMRI)*, which is capable of providing a 3D snapshot of brain activity every one or two seconds by measuring blood flow in the brain. The spatial resolution is limited to voxels of a few cubic millimeters.



Science Museum / Oxford Centre for Functional Magnetic Resonance Imaging of the Brain

Functional Brain Networks

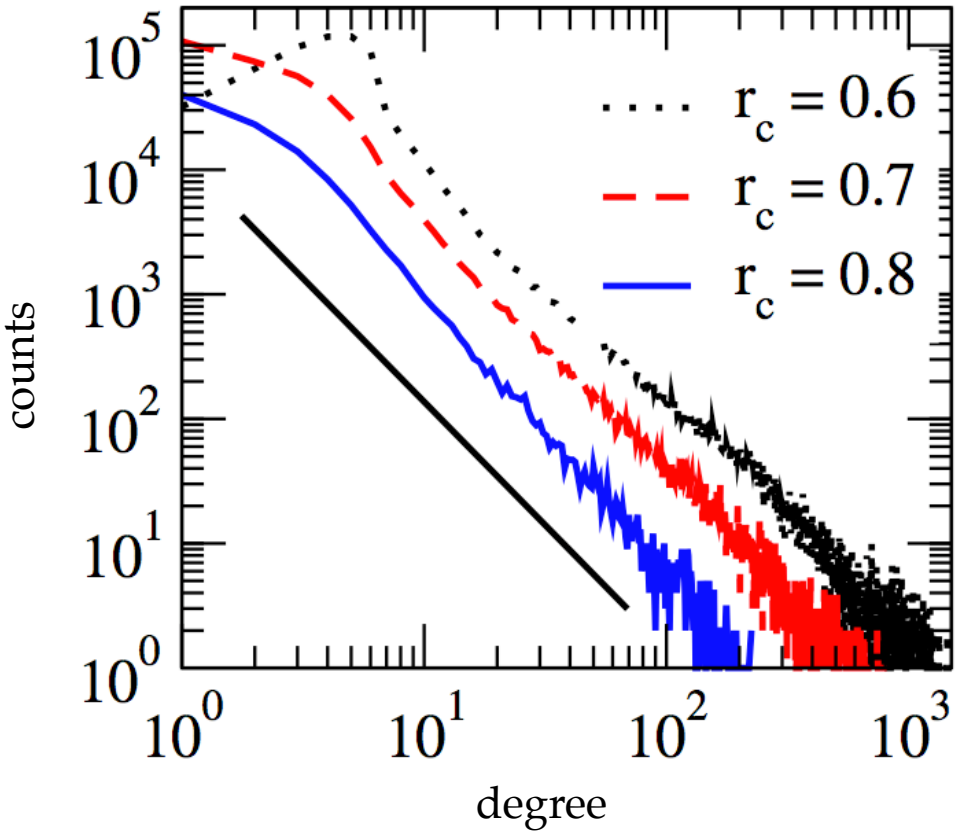
By measuring the activity across the brain over time (typically on the order of 10-20 minutes) and by measuring the temporal correlation of the brain activity between voxels, one can construct a weighted network between brain regions (which can then be thresholded).



Eguiluz et al. PRL **94**, 018102 (2005).

Functional Brain Networks

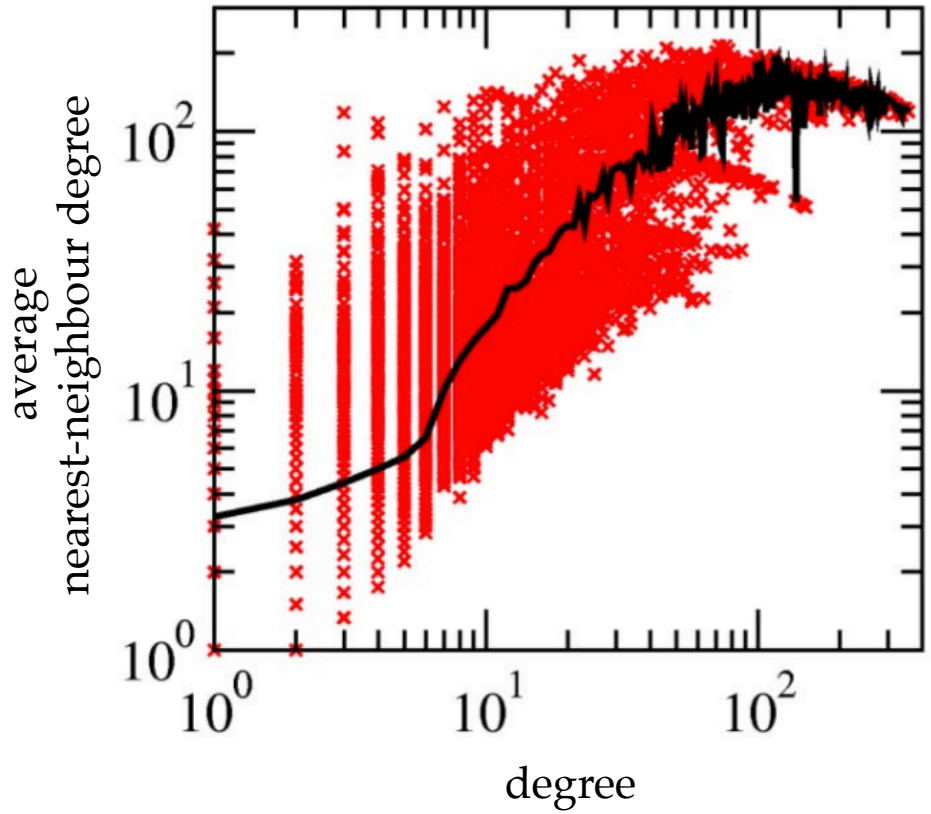
This network turns out to be *scale-free* – a feature not observed in small structural neural networks such as the C. elegans network.



Eguiluz et al. PRL **94**, 018102 (2005).

Functional Brain Networks

This network also turns out to be *assortative*, so highly connected nodes are connected to other highly connected nodes – like social networks.



Eguiluz et al. PRL **94**, 018102 (2005).

Functional Brain Networks

Lastly, this network is also a *small-world* network, just like structural neural networks such as the C. elegans neural network.

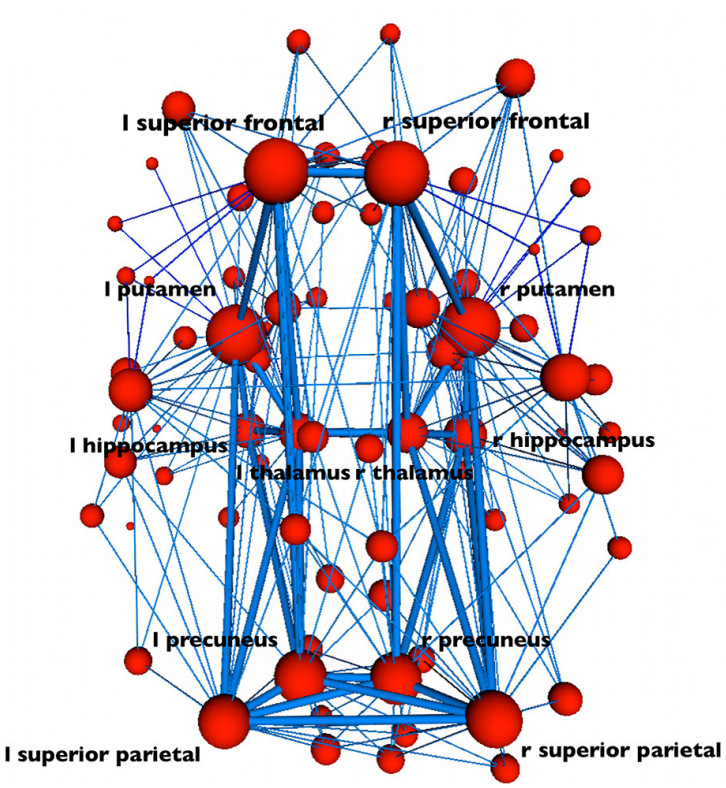
TABLE I. Average statistical properties of the brain functional networks.

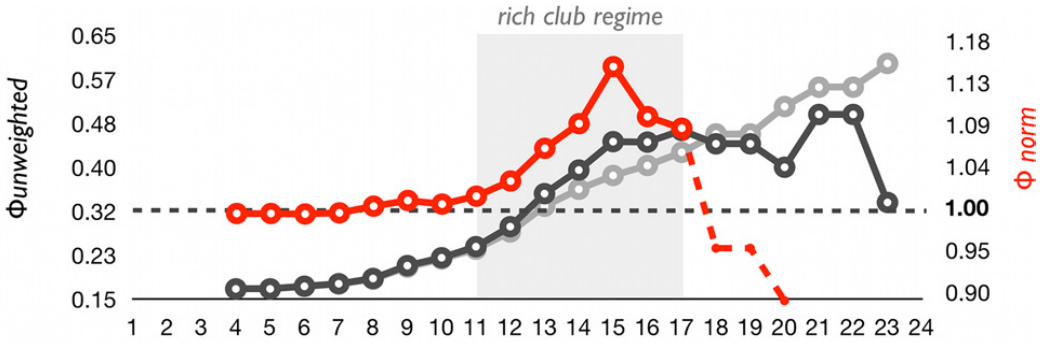
r_c	N	C	L	$\langle k \rangle$	γ	C_{rand}	L_{rand}
0.6	31 503	0.14	11.4	13.41	2.0	4.3×10^{-4}	3.9
0.7	17 174	0.13	12.9	6.29	2.1	3.7×10^{-4}	5.3
0.8	4891	0.15	6.0	4.12	2.2	8.9×10^{-4}	6.0

Eguiluz et al. PRL **94**, 018102 (2005).

Brain network rich clubs

Like the neural network of the primitive worm *C. elegans*, the fMRI network of the human brain has also been found to contain a rich club. This might point to a centralized organisation of brain activity.





van den Heuvel et al., J. Neurosci. 31, 15775 (2011).

American Football network

Networks can be found in many places, even in American football.

American football is organized into regional *conferences*.

Teams within the conferences all play each other, but any given team will play far fewer matches with teams in other conferences.

Therefore, overall rankings of all teams are often unsatisfactory as they rely on ‘expert opinions’ or very complex and somewhat arbitrary algorithms.

Even more confusingly, a couple of teams aren’t part of any conference!

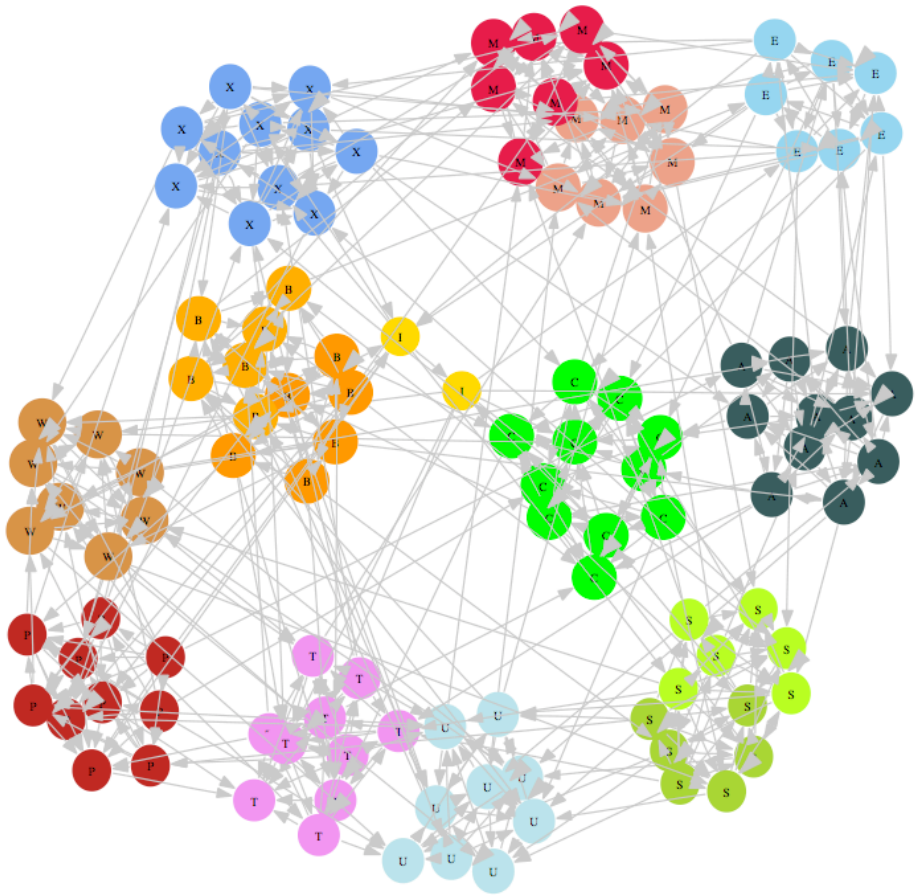
American Football network

A new approach:

We can draw a directed network between teams, where an edge signifies a win in a match, pointing from winner to loser.

Since a team can beat another one twice in a season, this network is weighted.

The adjacency matrix is **A**, with entries a_{ij}.



Park & Newman. J Stat Mech-Theory E, P10014 (2005).

American Football network

The number of *direct wins* is now simply the out-degree:

$$k_i^{out} = \sum_j a_{ij}$$

where $a_{ij} = 1$ signifies that i has beaten j once.*

We can now however also consider *indirect wins*, which follow the argument

"A has beaten B, and B has beaten C, therefore A must be stronger than C."

*In the original paper the notation convention is reverse.

Park & Newman. J Stat Mech-Theory E, P10014 (2005).

American Football network

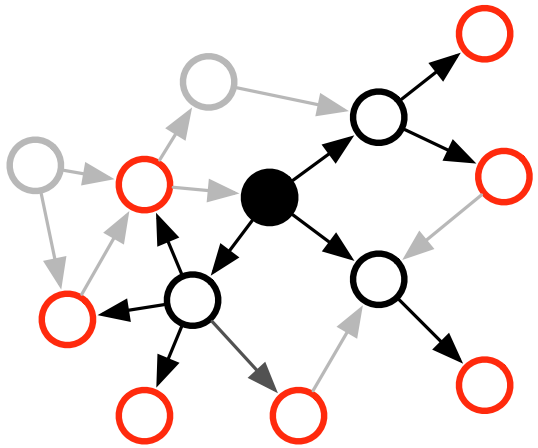
The number of these *indirect wins* is given by the number of second-nearest 'out-neighbours':

$$k_i^{out,\,2^{nd}} = \sum_{jk} a_{ij} a_{jk}$$

In this example,

$$k_i^{out} = 3$$

$$k_i^{out,\,2^{nd}} = 7$$



Park & Newman. J Stat Mech-Theory E, P10014 (2005).

American Football network

This can be generalized to a *total win score* w_i defined as:

$$w_i = \sum_j a_{ij} + \beta \sum_{jk} a_{ij} a_{jk} + \beta^2 \sum_{jkl} a_{ij} a_{jk} a_{kl} + \mathsf{K}$$

where β is a constant. This series will only converge if:

$$\beta < \lambda_{\max}^{-1}$$

Since the maximum eigenvalue $\lambda_{\max}$ is the maximum factor by which the entries of a vector can increase after multiplication by the matrix **A**.

Park & Newman. J Stat Mech-Theory E, P10014 (2005).

American Football network

We can rewrite the win score as:

$$w_i = \sum_j a_{ij} + \beta \sum_{jk} a_{ij} a_{jk} + \beta^2 \sum_{jkl} a_{ij} a_{jk} a_{kl} + \mathsf{K}$$

$$= \sum_j a_{ij} \left(1 + \beta \sum_k a_{jk} + \beta^2 \sum_{kl} a_{jk} a_{kl} + \mathsf{K} \right)$$

$$= \sum_j a_{ij} \left(1 + \beta w_j \right)$$

$$= k_i^{out} + \beta \sum_j a_{ij} w_j$$

Park & Newman. J Stat Mech-Theory E, P10014 (2005).

American Football network

Similarly, we can define a *loss score* as:

$$
\begin{aligned}
l_i &= \sum_j a_{ji} + \beta \sum_{jk} a_{kj} a_{ji} + \beta^2 \sum_{jkl} a_{lk} a_{kj} a_{ji} + \mathrm{K} \\
&= \sum_j \left(1 + \beta \sum_k a_{kj} + \beta^2 \sum_{kl} a_{lk} a_{kj} + \mathrm{K} \right) a_{ji} \\
&= \sum_j \left(1 + \beta w_j \right) a_{ji} \\
&= k_i^{in} + \beta \sum_j a_{ij}^T w_j
\end{aligned}
$$

Park & Newman. J Stat Mech-Theory E, P10014 (2005).

American Football network

We can then define a *total score* s_i as:

$$s_i = w_i - l_i$$

which we will use to rank the teams.

Our equations for the win and loss scores in vector form are:

$$\mathbf{w} = \mathbf{k}^{\mathbf{out}} + \beta \mathbf{A}\,\mathbf{w}$$

$$\mathbf{l} = \mathbf{k}^{\mathbf{in}} + \beta \mathbf{A}^{\mathbf{T}}\,\mathbf{l}$$

Park & Newman. J Stat Mech-Theory E, P10014 (2005).

American Football network

$$\mathbf{w} = \mathbf{k}^{\mathbf{out}} + \beta\, \mathbf{A}\, \mathbf{w}$$

$$\mathbf{l} = \mathbf{k}^{\mathbf{in}} + \beta\, \mathbf{A}^{\mathbf{T}}\, \mathbf{l}$$

Solving these equations for **w** and **l** we get:

$$\mathbf{w} = (I - \beta\, \mathbf{A})^{-1}\, \mathbf{k}^{\mathbf{out}}$$

$$\mathbf{l} = (I - \beta\, \mathbf{A}^{\mathbf{T}})^{-1}\, \mathbf{k}^{\mathbf{in}}$$

But which β should we choose?

Park & Newman. J Stat Mech-Theory E, P10014 (2005).

American Football network

We already know that β has to be smaller than λ_{max}^{-1}.

Importantly, λ_{max} is zero if the network has no directed loops.

This is because the adjacency matrix of a network without loops 'transports' all vector entries to nodes with out-degree zero, where these entries are set to zero (as there are no self-loops).

If the network does have loops we have a choice of

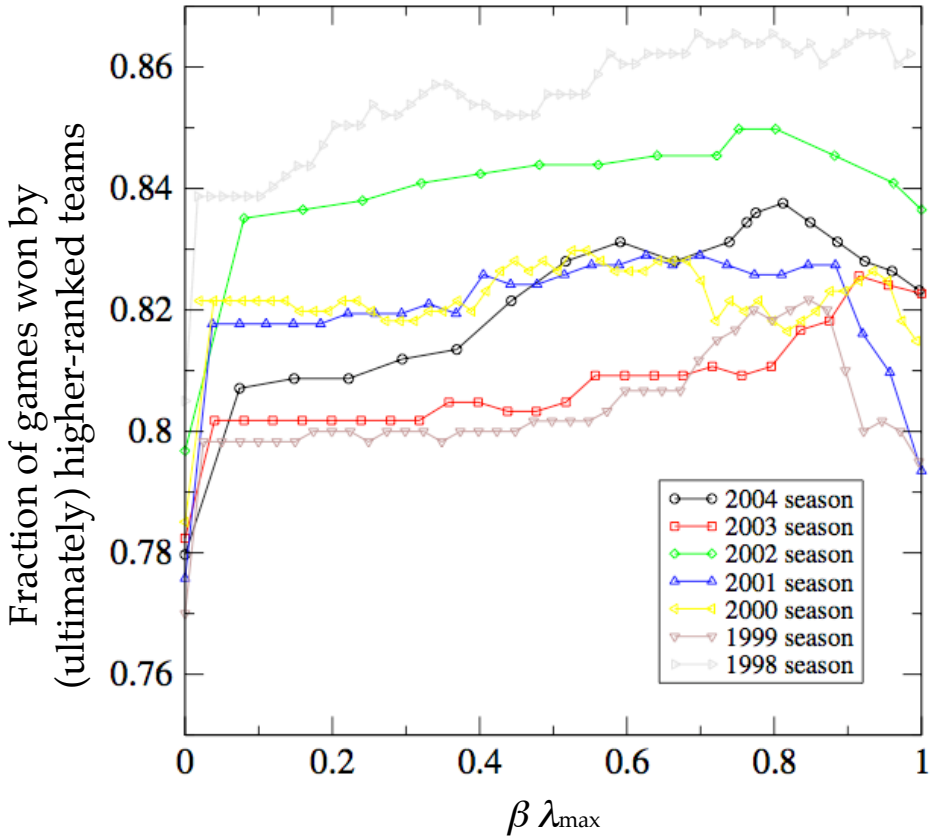
$$0 < \beta < \lambda_{max}^{-1}$$

Let us look across a range of β and see how the algorithm performs.

Park & Newman. J Stat Mech-Theory E, P10014 (2005).

American Football network

The algorithm performs well for a broad range of β, and best around $\beta = 0.8\ \lambda_{max}^{-1}$.



Park & Newman. J Stat Mech-Theory E, P10014 (2005).

American Football network

The only problem is we do not know λ_{max} until we have whole network, and we might want to use β to calculate results for the partial network.

But we can estimate λ_{max} using a random network with the correct total degree distribution.

Consider the number of paths $n(l)$ of length l, starting at a given node. It is important to realize that, in the limit of large l,

λ_{max} approaches $n(l+1) / n(l)$

i.e. it is the factor by which the number of paths increases if we increase the path length by one. This is equal to the

average out-degree of a node reached by following a random edge.

Park & Newman. J Stat Mech-Theory E, P10014 (2005).

American Football network

It is not simply the average out-degree, since out-degree and in-degree can be correlated, and by walking along the edges of the network we are much more likely to walk into a node with high in-degree.

Our probability of walking into a node with in-degree $k_{\text{in}} = i$ from a randomly selected edge is

$$P^{\text{walk}}(k_{\text{in}} = i) = i\, P(k_{\text{in}} = i)\, N \;/\; E = i\, P(k_{\text{in}} = i) / \langle k_{\text{in}} \rangle$$

Note that $\langle k_{\text{in}} \rangle = \langle k_{\text{out}} \rangle = E/N$ in directed networks.

Hence the probability of out-degree $k_{\text{out}} = \text{j}$ of a node we walk into is:

$$\sum_i P(k_{\text{out}} = j \mid k_{\text{in}} = i)\, P^{\text{walk}}(k_{\text{in}} = i)$$

Park & Newman. J Stat Mech-Theory E, P10014 (2005).

American Football network

And the average out-degree of such a node, which corresponds to $\lambda_{\max}$ is

$$\lambda_{\max} = \sum_{ij} j \, P(k_{\text{out}} = j \,|\, k_{\text{in}} = i) \, P^{\text{walk}}(k_{\text{in}} = i)$$

$$= \sum_{ij} j \, P(k_{\text{out}} = j \,|\, k_{\text{in}} = i) \, i \, P(k_{\text{in}} = i) \; / \; \langle k_{\text{in}} \rangle$$

$$= \sum_{ij} ij \, P(k_{\text{out}} = j \, , \, k_{\text{in}} = i) \; / \; \langle k_{\text{in}} \rangle$$

$$= \sum_{ij} ij \, P(k_{\text{out}} = j \, , \, k_{\text{in}} = i) \; / \; \sum_{ij} i \, P(k_{\text{out}} = j \, , \, k_{\text{in}} = i)$$

So we need the joint distribution $P(k_{\text{out}} = j \, , \, k_{\text{in}} = i)$ for a random directed network with a fixed total degree distribution p_k since the total number of games played by each team is fixed.

Park & Newman. J Stat Mech-Theory E, P10014 (2005).

American Football network

If the edge directions are chosen randomly, the joint in- and out-degree distribution takes the binomial form

$$P(k_{\text{out}} = j\ ,\ k_{\text{in}} = i) = 2^{-(i+j)}\ {}^{(i+j)}\mathrm{C}_i\ p_{i+j}$$

Plugging this into

$$\lambda_{\max} = \Sigma_{ij}\ ij\ P(k_{\text{out}} = j\ ,\ k_{\text{in}} = i)\ /\ \Sigma_{ij}\ i\ P(k_{\text{out}} = j\ ,\ k_{\text{in}} = i)$$

gives, after some algebra:

$$\lambda_{\max} = (\langle k^2 \rangle - \langle k \rangle)\ /\ 2\ \langle k \rangle$$

where $k = i + j$ is the total degree of a node. Hence we can now choose β, calculate the score vector and predict American football!

Park & Newman. J Stat Mech-Theory E, P10014 (2005).

THE END

Thank you for coming!