Graph Layout

Task: visualize various interaction data:

e.g. **protein interaction data** (undirected):
- nodes – proteins
- edges – interactions

metabolic pathways (directed)
- nodes – substances
- edges – reactions

regulatory networks (directed):
- nodes – transcription factors + regulated proteins
- edges – regulatory interaction

co-localization (undirected)
- nodes – proteins
- edges – co-localization information

homology (undirected/directed)
- nodes – proteins
- edges – sequence similarity (BLAST score)

Graph Layout Algorithms

Graphs encapsulate relationship between objects
→ drawing gives **visual impression** of these relations

Good Graph Layout: **aesthetic**
- minimal edge crossing
- highlight symmetry (when present in the data)
- even spacing between the nodes

Many approaches in literature (and in software tools),
most useful ones usually NP-complete (exponential runtime)

Most popular for **straight-edge-drawing**:
→ **force-directed**: spring model or spring-electrical model
→ **embedding** algorithms like H3 or LGL

Force-Directed Layout

Peter Eades (1984): graph layout heuristic

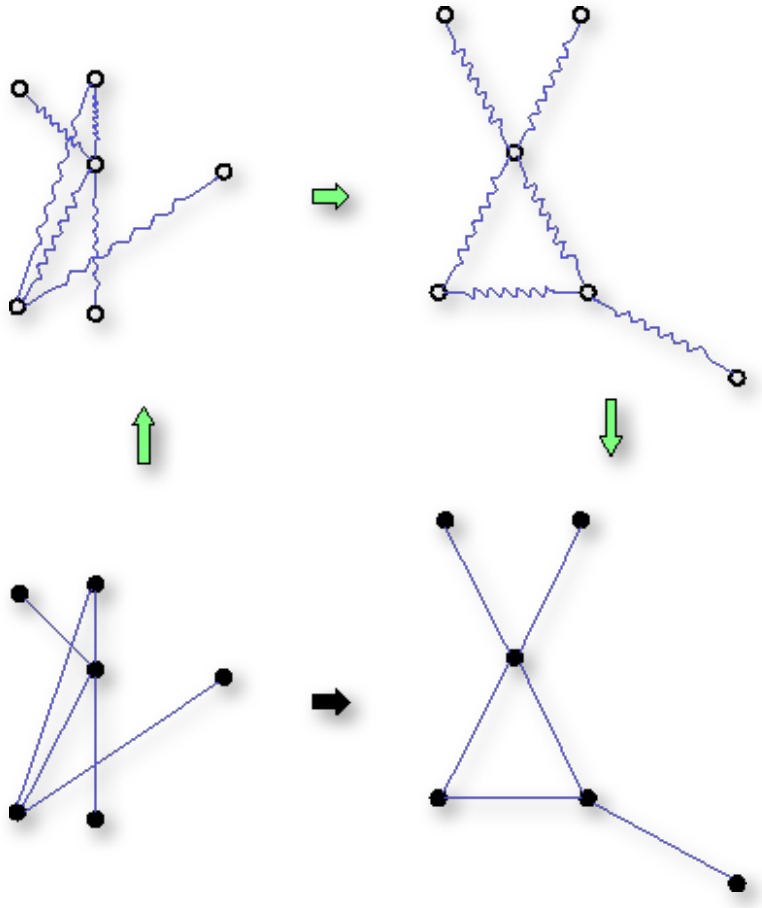
→ "**Spring Embedder**" algorithm.

- edges → springs
  vertices → rings that connect the springs

- Layout by dynamic relaxation

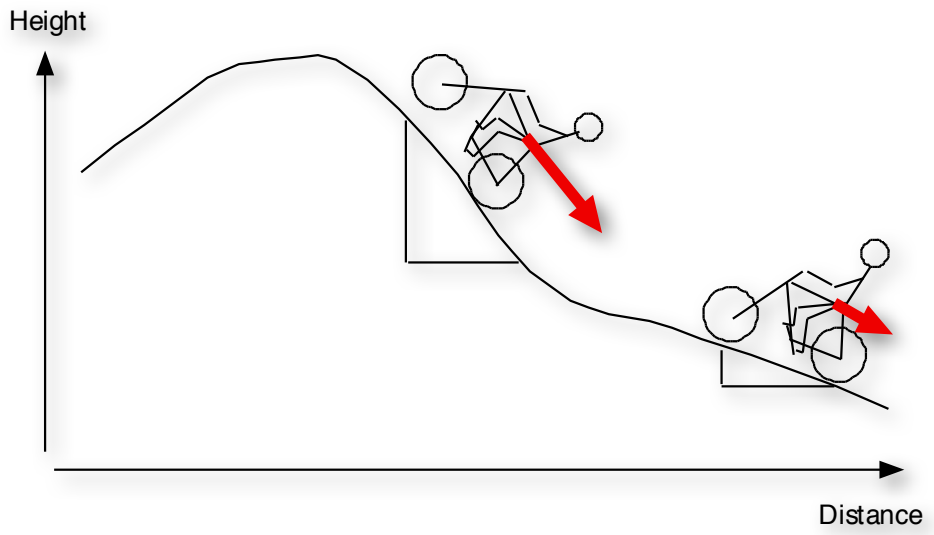
→ lowest-energy conformation

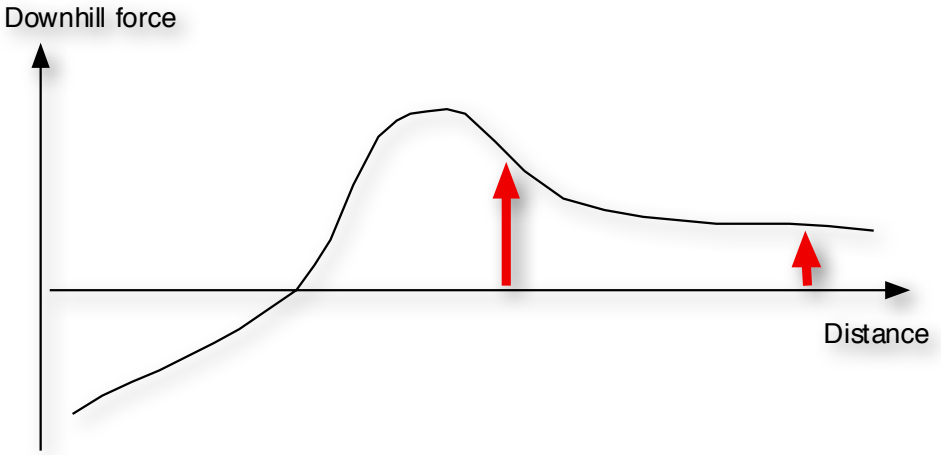
→ "**Force Directed**" algorithm



http://www.hpc.unm.edu/~sunls/research/treelayout/node1.html

Energy and Force





Energy: describes the altitude of the landscape

$$E(x) = mgh(x)$$

Energy increases when you go up the hill

You need more force for a steeper ascent

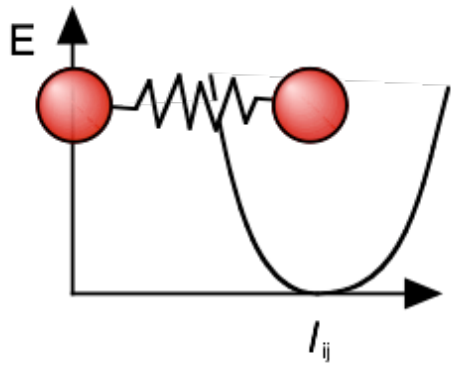
$$F(x) = -\frac{dE(x)}{dx}$$

Force: describes the change of the altitude, points downwards.

Spring Embedder Layout

Springs regulate the mutual distance between the nodes

- too close → repulsive force
- too far → attractive force



Spring embedder algorithm:

- add springs for all edges
- add loose springs to all non-adjacent vertex pairs

Total energy of the system:

$$E = \sum_{i=1}^{|V|-1} \sum_{j=i+1}^{|V|} \frac{R}{l_{ij}^2} \left(|x_i - x_j| - l_{ij}\right)^2$$

x_i, x_j = position vectors for nodes i and j
l_{ij} = rest length of the spring between i and j
R = spring constant (stiffness)

Problem: l_{ij} have to be determined a priori, e.g., from network distance

Spring Model Layout

Task: find configuration of **minimal energy**

In 2D/3D: force = negative gradient of the energy

$$\vec{F}(\vec{x}) = -\nabla E(\vec{x}) = -\begin{pmatrix} \frac{\partial E}{\partial x} \\ \frac{\partial E}{\partial y} \\ \frac{\partial E}{\partial z} \end{pmatrix}$$

→ Iteratively **move** nodes "**downhill**" along the gradient of the energy

→ displace nodes **proportional** to the **force** acting on them

Problems:

- local minima
- a priori knowledge of all spring lengths

→ works best for regular grids

The Spring-Electrical-Model

More general model than spring embedder model: use two types of forces

1) **attractive harmonic** force between connected nodes (springs)

$$F_{ij}^{h} = -k\,|r_i - r_j|$$

one uses usually the same spring constant *k* for all edges

2) **repulsive Coulomb**-like force between all nodes

"all nodes have like charges" → repulsion

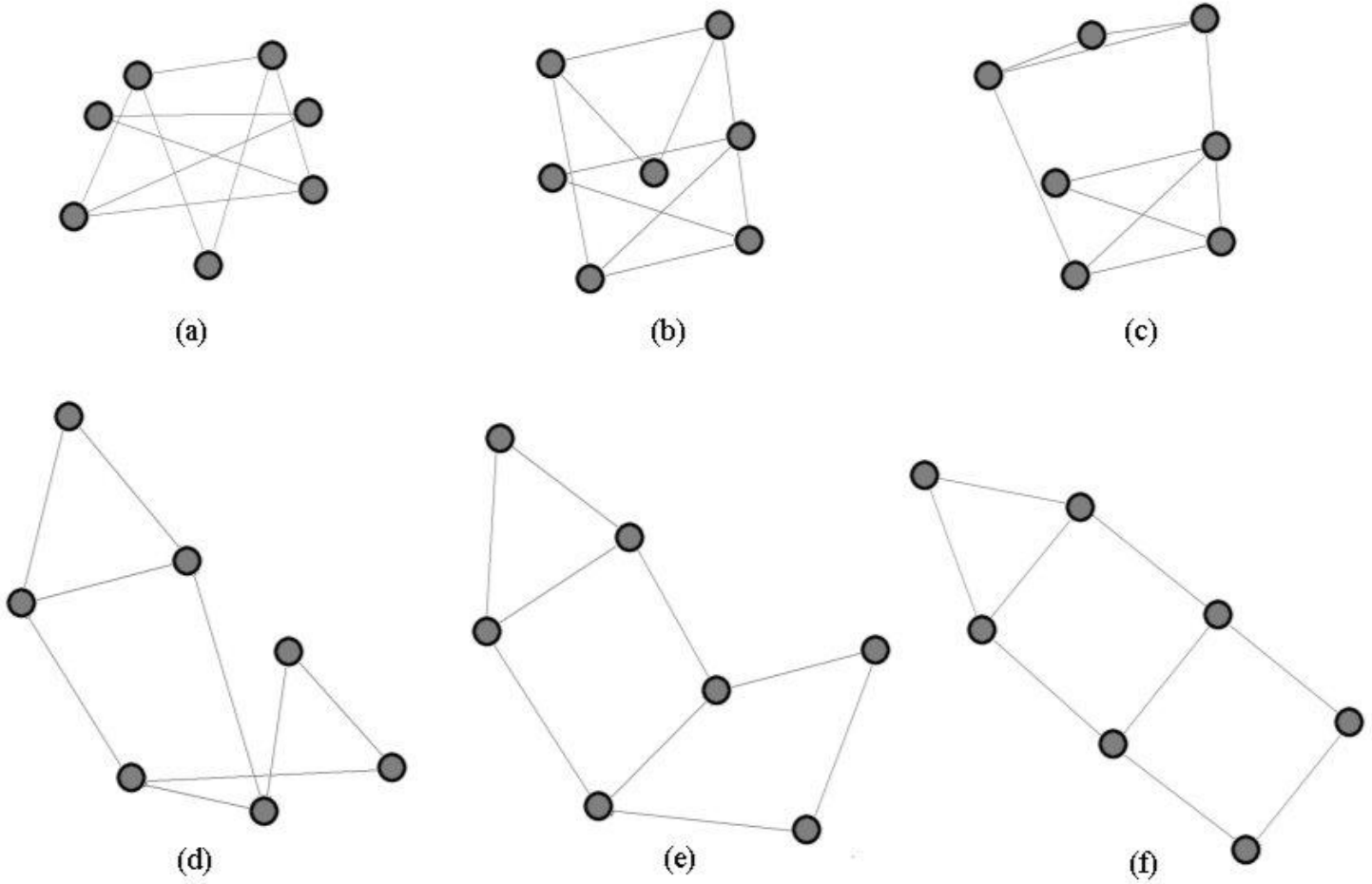
$$F_{ij}^{c} = \frac{Q_{ij}}{|r_i - r_j|^2}$$

either $Q_{ij} = Q$ or, e.g., $Q_{ij} = k_i\, k_j$

Repulsion pushes all nodes apart, springs pull connected nodes together
→ **workhorse method** for small to medium sized graphs

→ Do-it-yourself in Assignment 2 <=

Spring-Electrical Example



http://www.it.usyd.edu.au/~aquigley/3dfade/

Force-Directed Layout: Summary

Analogy to a **physical** system
=> force directed layout methods tend to meet various **aesthetic** standards:

- efficient **space filling**,
- **uniform** edge length (with equal weights and repulsions)
- **symmetry**
- smooth **animation** of the layout process (visual continuity)

Force directed graph layout $\rightarrow$ the "**work horse**" of layout algorithms.

Not so nice: the **initial random placement** of nodes and even very small changes of layout parameters will lead to **different representations**.
(no unique solution)

Side-effect: vertices at the periphery tend to be closer to each other than those in the center…

Runtime Scaling

Force directed layout:

```
loop until convergence:
   calculate forces:
      L springs
      N(N-1)/2 charge pairs
   move vertices
   output positions
```

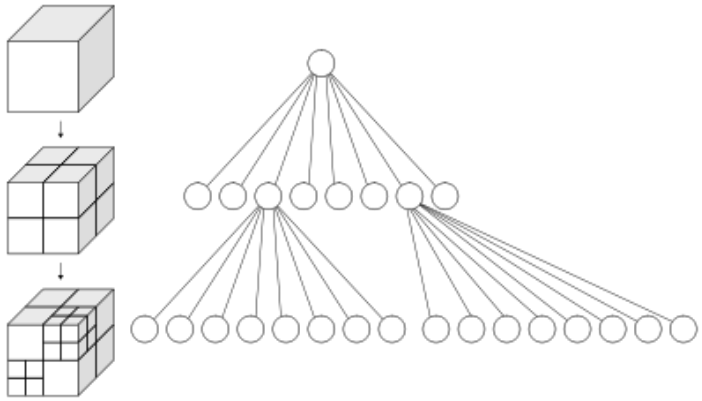
Several possible arrangements!!! (local minima)

$O(N^2)$!!!

→ force directed layout suitable for small to medium graphs (≤ O(1000) nodes?)

Speed up layout by:

- **multi-level** techniques to overcome local minima
- **clustering** (octree) methods for distant groups of nodes → *O(N log N)*



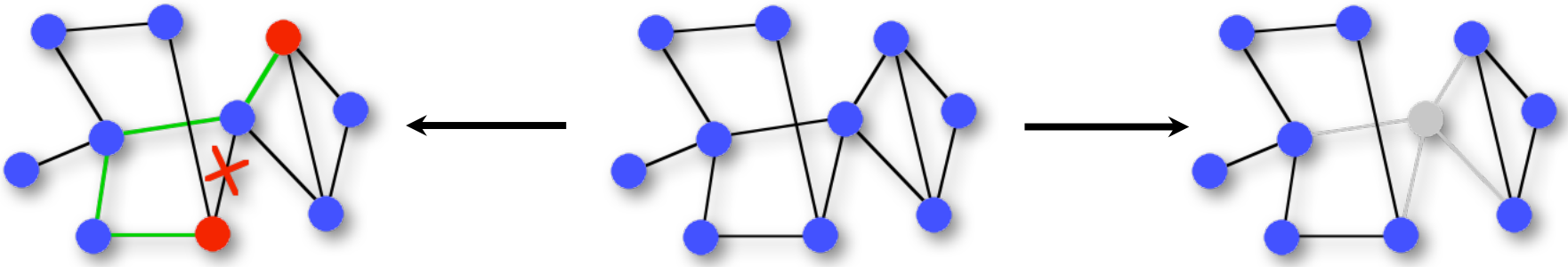
Network Robustness

Network = set of connections

Failure events:
- loss of edges
- loss of nodes (together with their edges)

→ loss of connectivity
- paths become longer (detours required)
- connected components break apart

→ network characteristics change



→ **Robustness** = how much does the network (not) change when edges/nodes are removed

Error and attack tolerance of complex networks

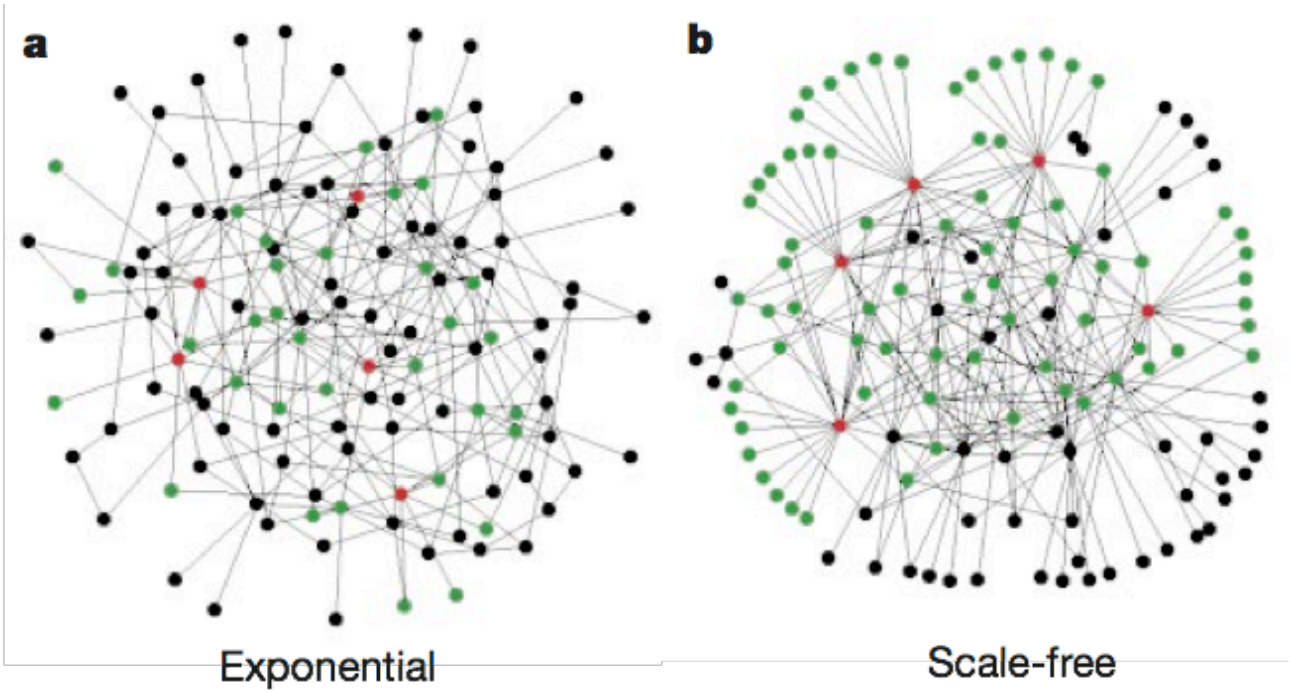
Réka Albert, Hawoong Jeong & Albert-László Barabási

Department of Physics, 225 Nieuwland Science Hall, University of Notre Dame, Notre Dame, Indiana 46556, USA

Many complex systems display a surprising degree of tolerance against errors. For example, relatively simple organisms grow, persist and reproduce despite drastic pharmaceutical or environmental interventions, an error tolerance attributed to the robustness of the underlying metabolic network[1]. Complex communication networks[2] display a surprising degree of robustness: although key components regularly malfunction, local failures rarely lead to the loss of the global information-carrying ability of the network. The stability of these and other complex systems is often attributed to the redundant wiring of the functional web defined by the systems' components. Here we demonstrate that error tolerance is not shared by all redundant systems: it is displayed only by a class of inhomogeneously wired networks,

millan Magazines Ltd NATURE | VOL 406 | 27 JULY 2000 | www.nature.com

Random vs. Scale-Free



130 nodes, 215 edges

The **top 5** nodes with the highest *k* **connect** to…

… 27% of the network

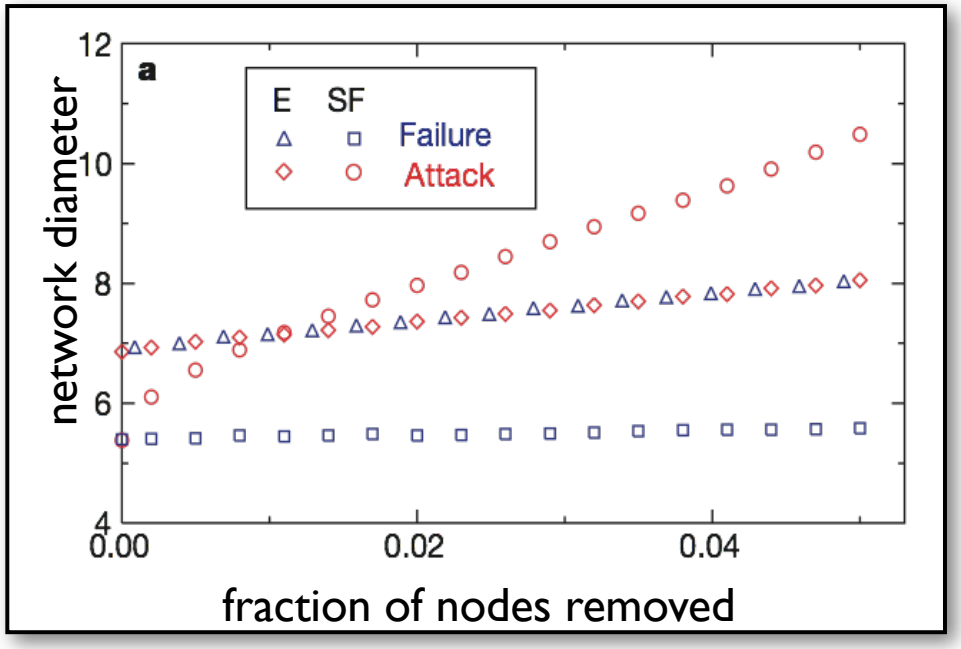
… 60% of the network

Albert, Jeong, Barabási, *Nature* **406** (2000) 378

Failure vs. Attack

Failure: remove **randomly** selected nodes

Attack: remove nodes with highest **degrees**



SF: scale-free network -> attack

E: exponential (random) network -> failure / attack

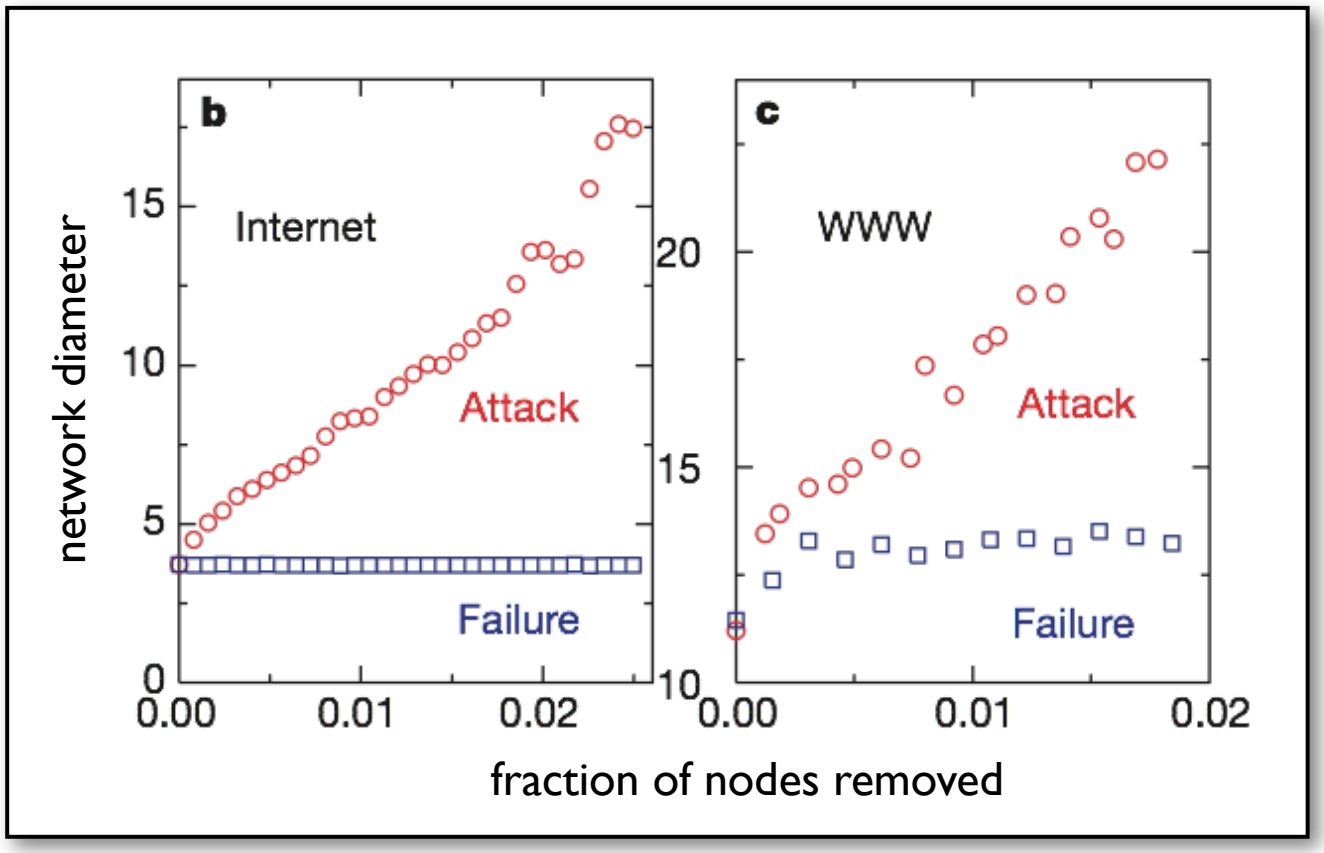
SF: failure

N = 10000, L = 20000, but effect is size-independent;

Interpretation:
SF network diameter increases strongly when network is attacked but not when nodes fail randomly

Two real-world networks

Scale-free:
- very **stable** against random **failure** ("packet re-rooting")
- very **vulnerable** against dedicated **attacks** ("9/11")



http://moat.nlanr.net/Routing/rawdata/ : 6209 nodes and 12200 links (2000)

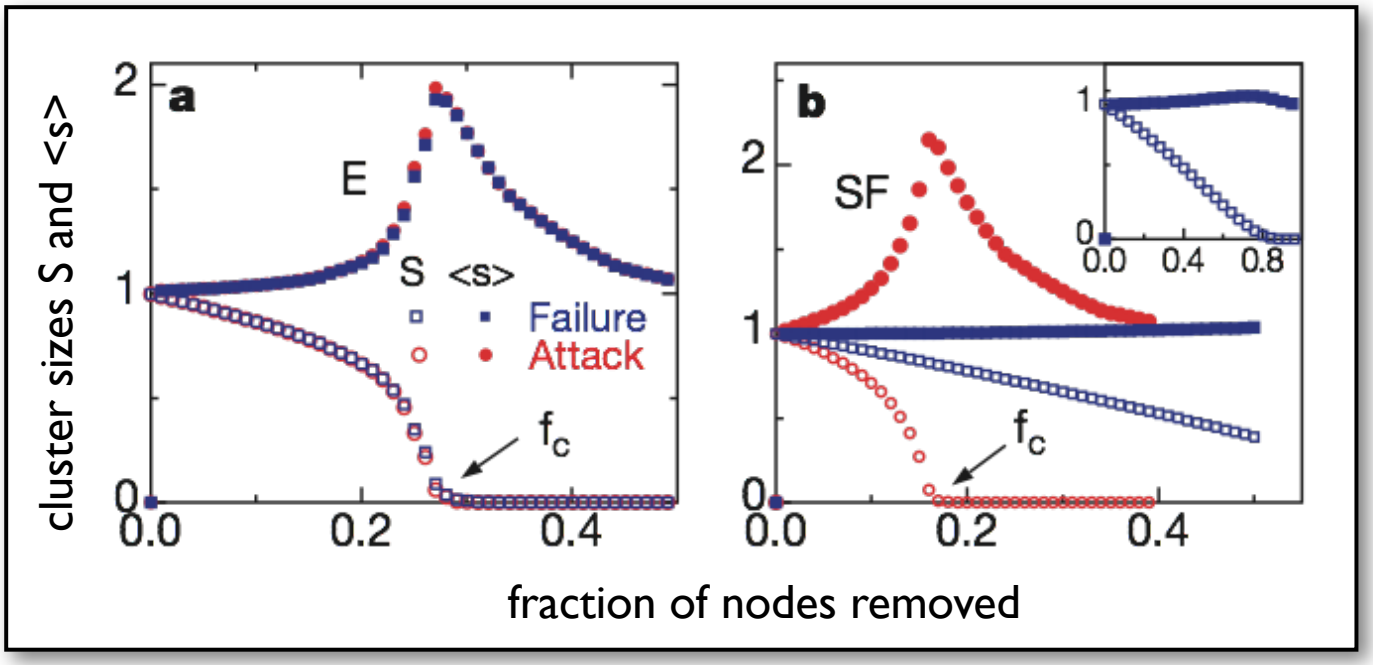
WWW-sample containing 325729 nodes and 1498353 links

Albert, Jeong, Barabási, *Nature* **406** (2000) 378

Network Fragmentation

<s>: average size of the isolated clusters (except the largest one)

S: relative size of the largest cluster S; this is defined as the fraction of nodes contained in the largest cluster (that is, *S = 1 for f = 0)*

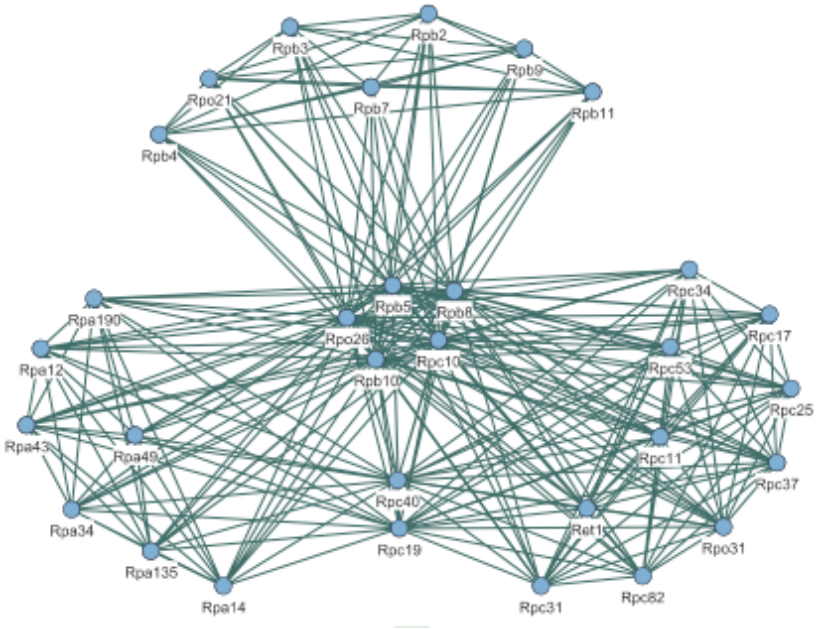


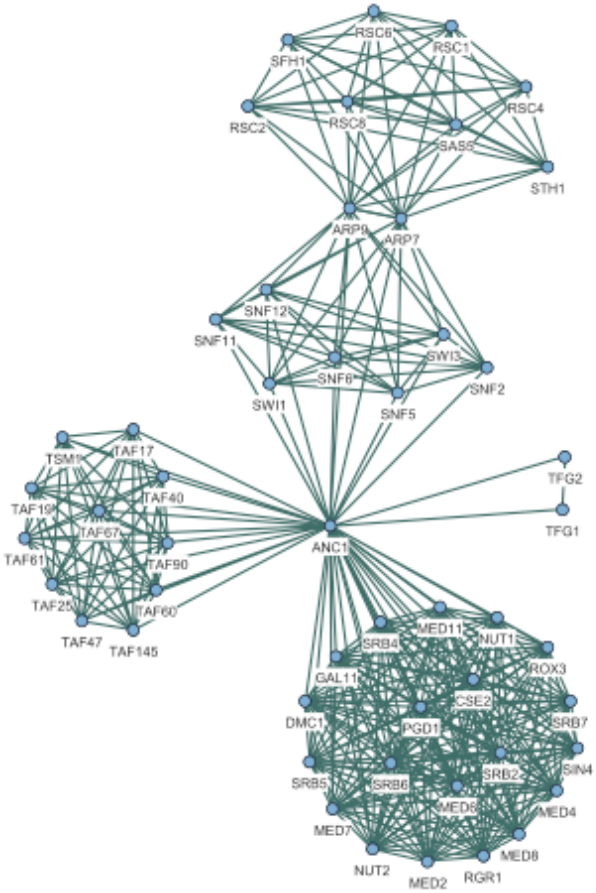
Random network:
- **no difference** between attack and failure (homogeneity)
- fragmentation threshold at $f_c \gtrsim 0.28$ ($S \approx 0$)

Scale-free network:
- **delayed fragmentation** and isolated nodes for failure
- critical breakdown under attack at $f_c \approx 0.18$

Albert, Jeong, Barabási, *Nature* **406** (2000) 378

Reducing Network Complexity?





Is there a **representation** that highlights the **structure** of these networks???

- Modular Decomposition (Gagneur, …, Casari, 2004)
- Network Compression (Royer, …, Schröder, 2008)

Method

Open Access

Modular decomposition of protein-protein interaction networks

Julien Gagneur*†, Roland Krause*, Tewis Bouwmeester* and Georg Casari*

Addresses: *Cellzome AG, Meyerhofstrasse 1, 69117 Heidelberg, Germany. †Laboratoire de Mathématiques Appliquées aux Systèmes, Ecole Centrale Paris, Grande Voie des Vignes, 92295 Châtenay-Malabry cedex, France.

Abstract

We introduce an algorithmic method, termed modular decomposition, that defines the organization of protein-interaction networks as a hierarchy of nested modules. Modular decomposition derives the logical rules of how to combine proteins into the actual functional complexes by identifying groups of proteins acting as a single unit (sub-complexes) and those that can be alternatively exchanged in a set of similar complexes. The method is applied to experimental data on the pro-inflammatory tumor necrosis factor-α (TNF-α)/NFκB transcription factor pathway.

Genome Biology **5** (2004) R57

Shared Components

Shared components = proteins or groups of proteins occurring in different complexes are fairly common. A shared component may be a small part of many complexes, acting as a **unit** that is constantly **reused** for its function.

Also, it may be the **main part** of the complex e.g. in a family of variant complexes that differ from each other by distinct proteins that provide functional specificity.

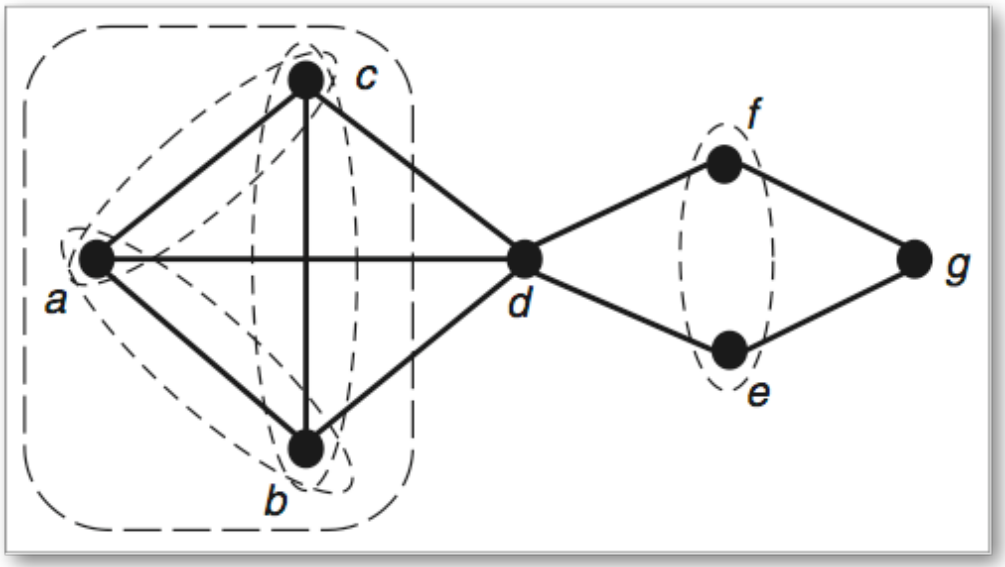
Aim: **identify** and properly **represent** the modularity of protein-protein interaction networks by identifying the **shared components** and the way they are arranged to generate **complexes**.

Georg Casari, Cellzome (Heidelberg)

Gagneur et al. Genome Biology 5, R57 (2004)

Modular Decomposition of a Graph

Module := set of **nodes** that have the
same neighbors outside of the module



trivial modules:

{a}, {b}, …, {g}
{a, b, …, g}

non-trivial modules:

{a, b}, {a, c}, {b, c}
{a, b, c}
{e, f}

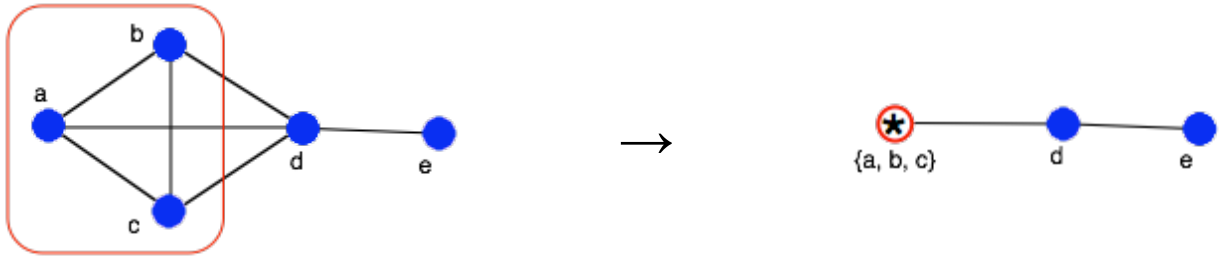
Quotient: representative node for a module

Iterated quotients → labeled tree representing the original network
→ "**modular decomposition**"

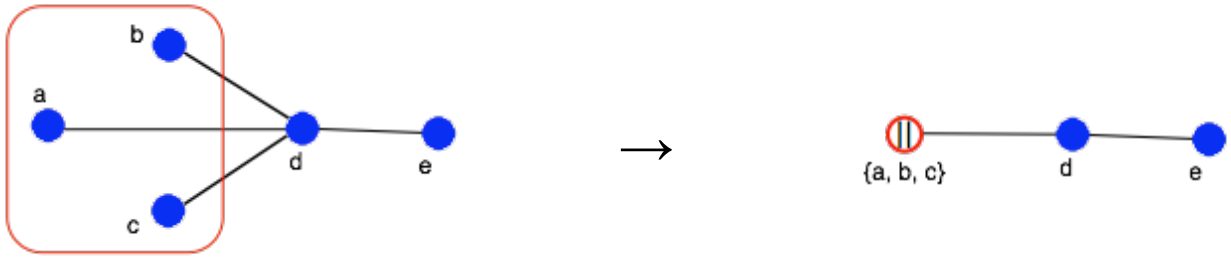
Gagneur et al, *Genome Biology* **5** (2004) R57

Quotients

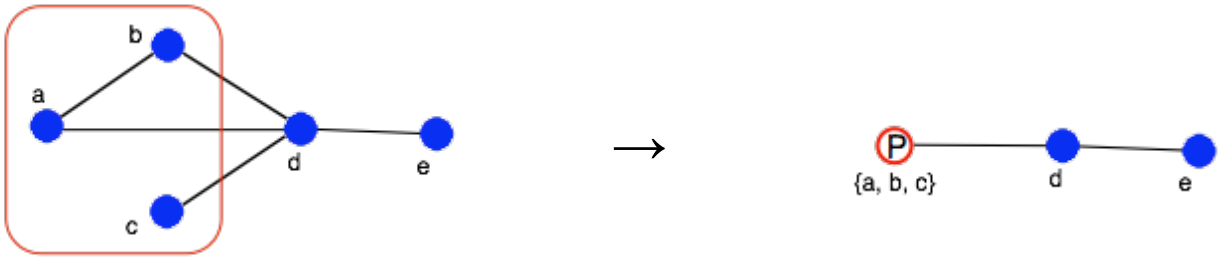
Series: all included nodes are direct **neighbors** (= **clique**)



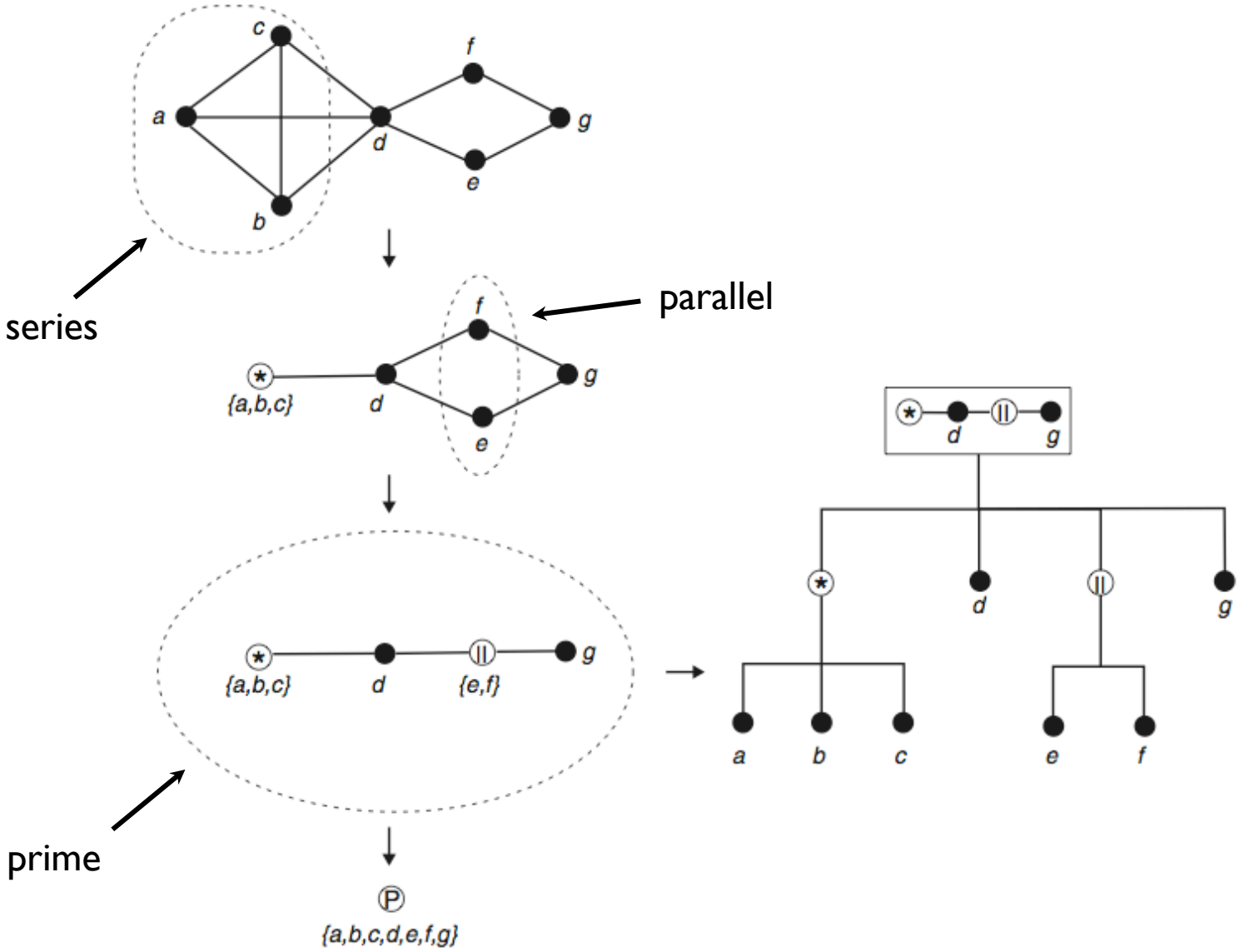
Parallel: all included nodes are **non-neighbors**



Prime: "anything else" (best labeled with the actual structure)



A Simple Recursive Example



Gagneur et al, *Genome Biology* **5** (2004) R57

Using data from protein complex purifications e.g. by TAP

Different types of data:

• Y2H: detects direct physical interactions between proteins

• PCP by tandem affinity purification with mass-spectrometric identification of the protein components identifies multi-protein complexes

→ Molecular decomposition will have a **different meaning** due to different **semantics** of such graphs.

Here, we focus analysis on **PCP content** from TAP-MS data.

PCP experiment: select bait protein where TAP-label is attached → Co-purify protein with those proteins that co-occur in at least one complex with the bait protein.

Gagneur et al. Genome Biology 5, R57 (2004)

Data from Protein Complex Purification

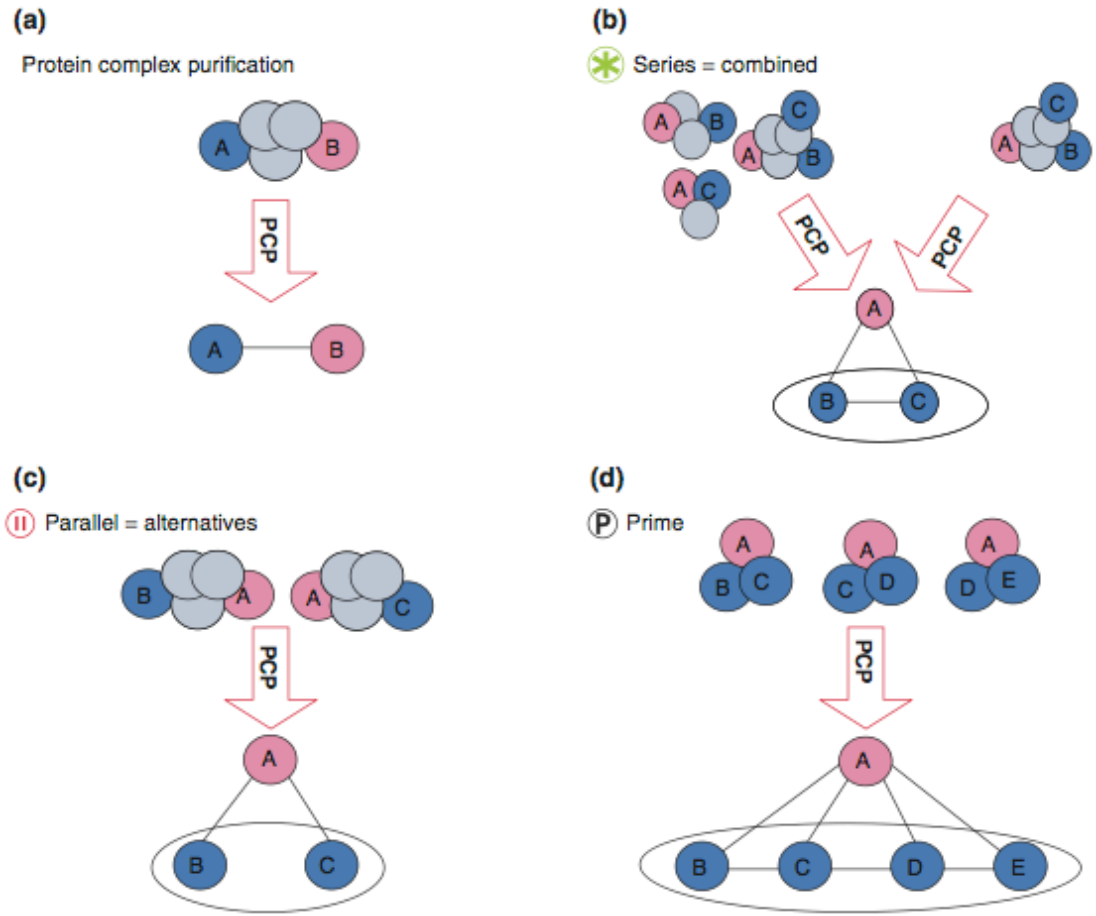
Graphs and module labels from systematic PCP experiments:

(a) Two neighbors in the network are proteins occurring in a same complex.

(b) Several potential sets of complexes can be the origin of the same observed network. Restricting interpretation to the simplest model (top right), the **series** module reads as a logical AND between its members.

(c) A module labeled ´**parallel**´ corresponds to proteins or modules working as strict alternatives with respect to their common neighbors.

(d) The ´**prime**´ case is a structure where none of the two previous cases occurs.



Gagneur et al. Genome Biology 5, R57 (2004)

Real World Examples

Two examples of modular decompositions of protein-protein interaction networks.

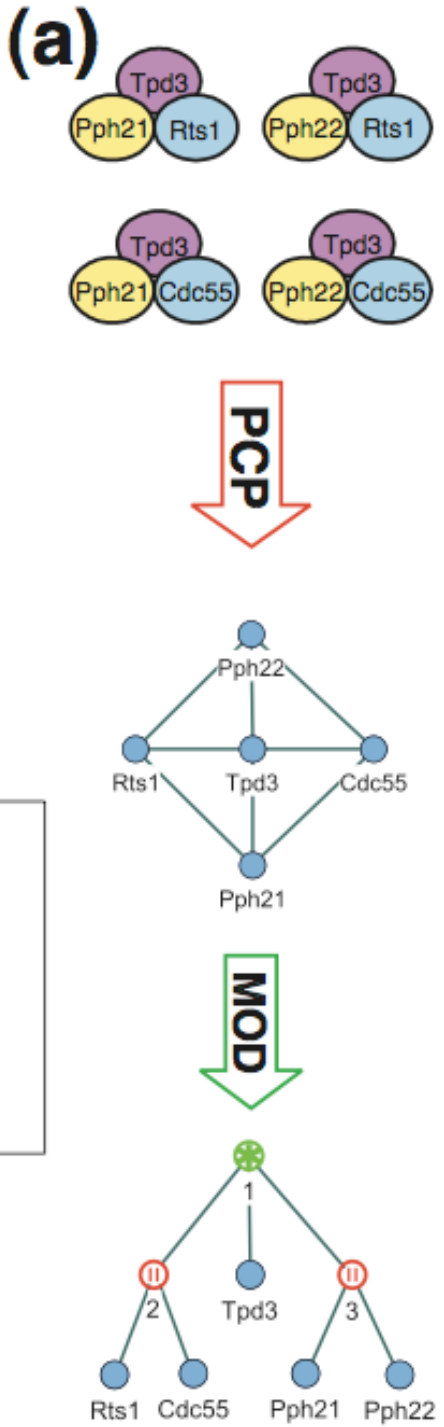
In each case from top to bottom: schemata of the complexes, the corresponding protein-protein interaction network as determined from PCP experiments, and its modular decomposition (MOD).

(a) Protein phosphatase 2A.

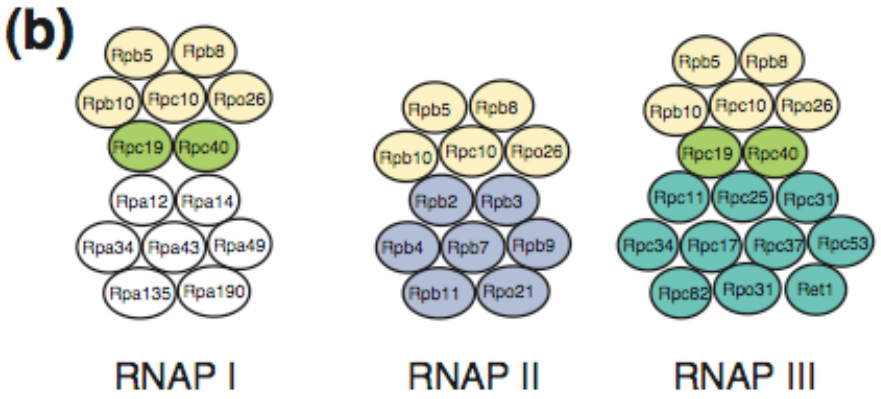
Parallel modules group proteins that do not interact but are functionally equivalent.

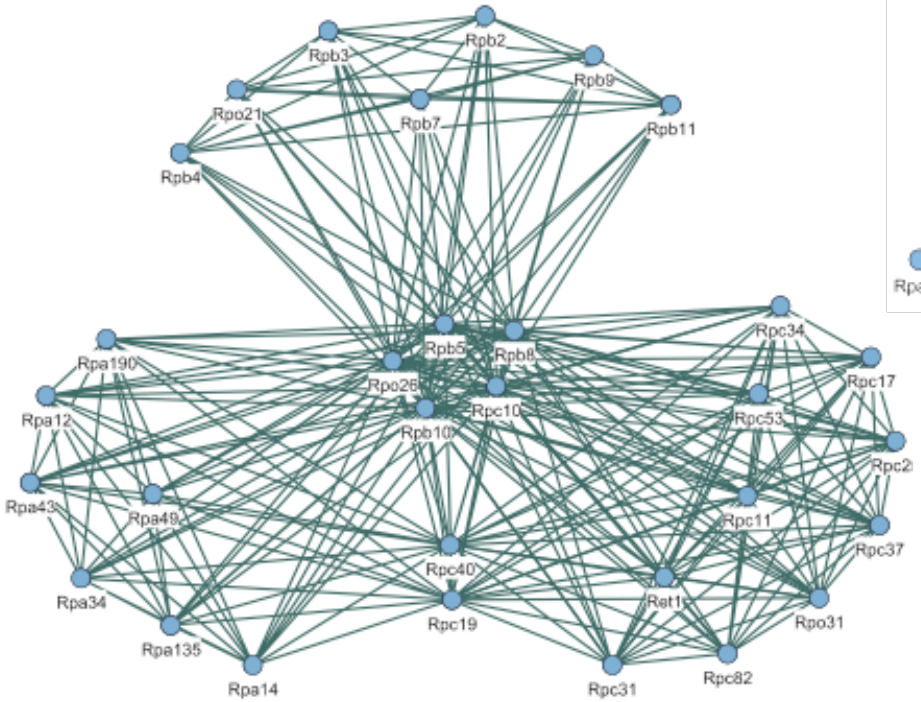
Here these are the catalytic proteins Pph21 and Pph22 (module 2) and the regulatory proteins Cdc55 and Rts1 (module 3), connected by the Tpd3 „backbone“.

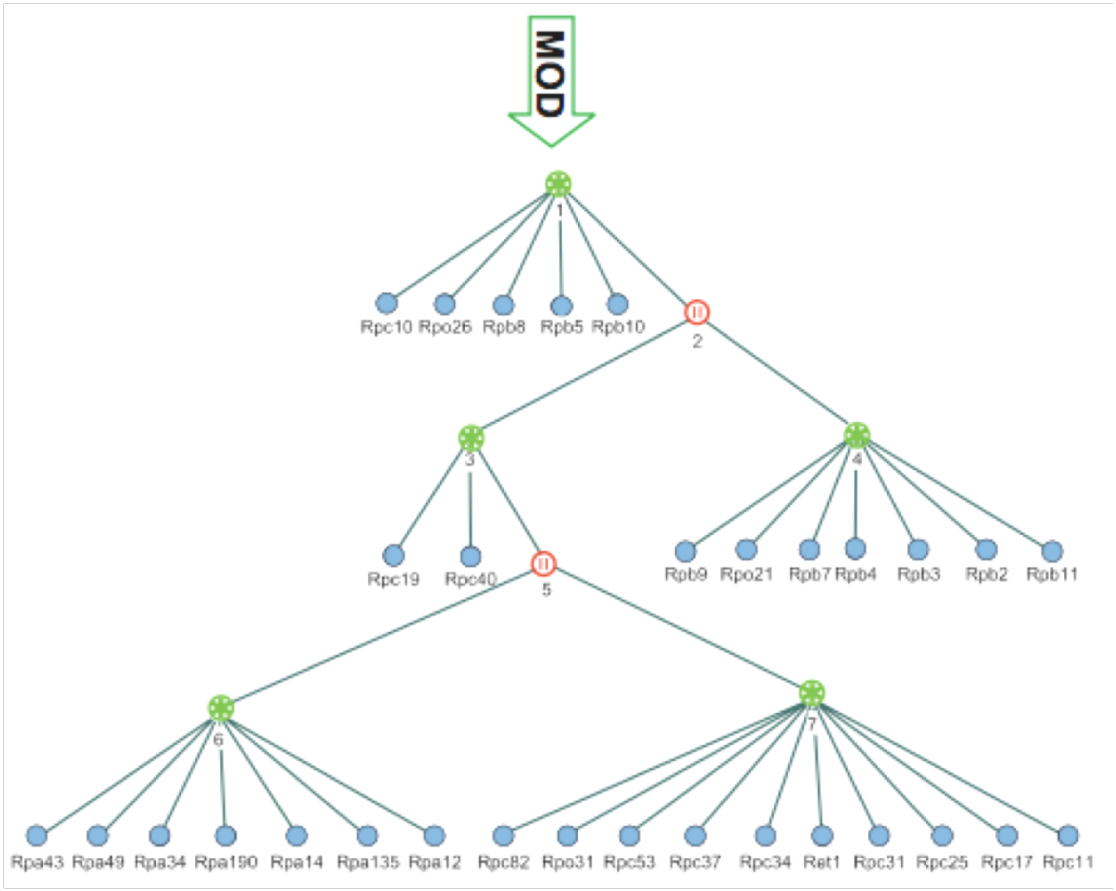
Notes: • Graph does not show functional alternatives!!!
• other decompositions also possible



RNA polymerases I, II and III







Again: modular decomposition is much easier to understand than the connectivity graph

Gagneur et al. Genome Biology 5, R57 (2004)