V 6 – Network analysis

- Dijkstra algorithm: compute shortest pathways
- Graph layout
- Network robustness
- Graph modularity

Fri, May 4, 2018

The Shortest Path Problem

Edsger Dijkstra (1930-2002):

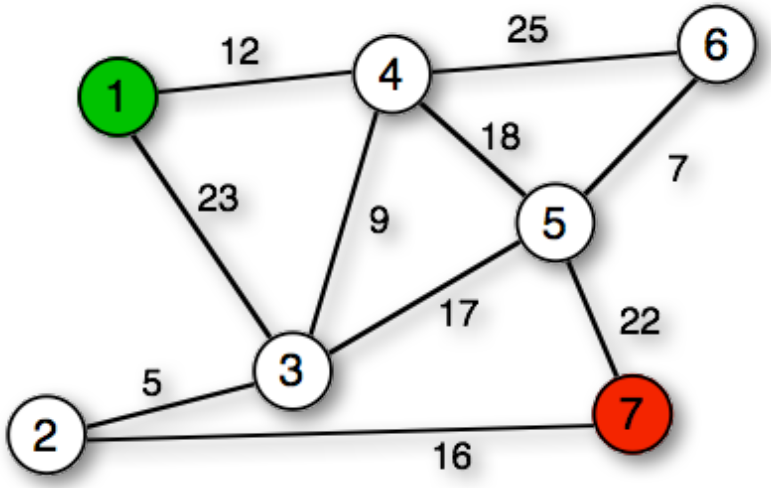
Problem:
Find the shortest path from a given vertex to the other vertices of the graph (Dijkstra 1959).

We need (input):
- weighted graph *G(V, E)*
- start (source) vertex *s* in *G*

We get (output):
- shortest distances *d[v]* between *s* and *v*
- shortest paths from *s* to *v*

Idea: Always proceed with the closest node
→ greedy algorithm

Real world application:
→ GPS navigation devices



Dijkstra Algorithm 0

Initialization:

```
for all nodes v in G:
    d[v] = oo
    pred[v] = nil
d[s] = 0
```

distance and path to all other nodes is still unknown

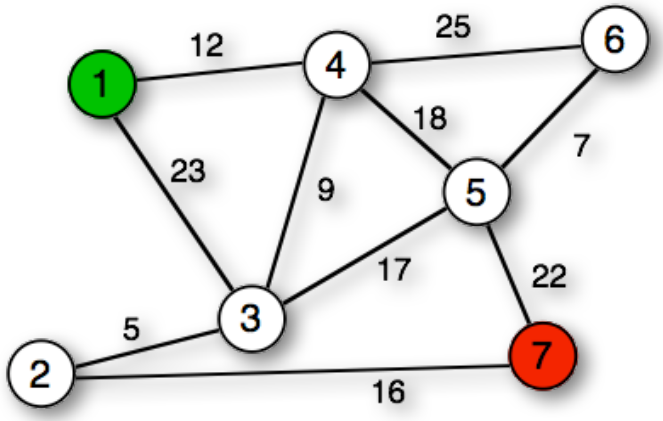
distance from source to source = 0

d[v] = length of path from *s* to *v*

pred[v] = predecessor node on the shortest path

In the example: *s* = 1

node	1	2	3	4	5	6	7
d	0	oo	oo	oo	oo	oo	oo
pred	–	–	–	–	–	–	–



Dijkstra I

Iteration:

```
Q = V
while Q is not empty:
   u = node with minimal d

   if d[u] = oo:
      break

   delete u from Q

   for each neighbor v of u:
      d_temp = d[u] + d(u,v)

      if d_temp < d[v]:
         d[v] = d_temp
         pred[v] = u

return pred[]C
```

Save {*V*} into working copy *Q*

choose node closest to *s*

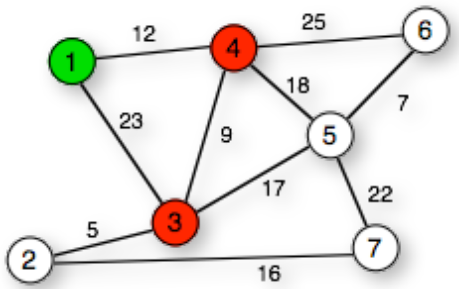
exit if all remaining nodes are inaccessible

calculate distance to *u*'s neighbors

if new path is shorter => update

Dijkstra-Example

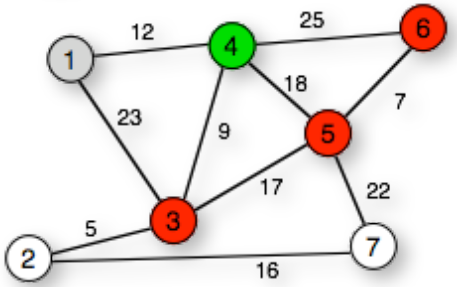
1)



Q = (**1**, 2, 3, 4, 5, 6, 7)

node	**1**	2	3	4	5	6	7
d	**0**	oo	23	12	oo	oo	oo
pred	–	–	1	1	–	–	–

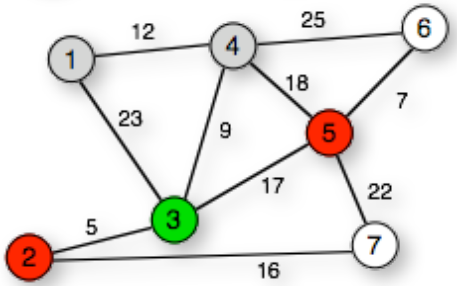
2)



Q = (2, 3, **4**, 5, 6, 7)

node	1	2	3	**4**	5	6	7
d	0	oo	21	**12**	30	37	oo
pred	–	–	4	**1**	4	4	–

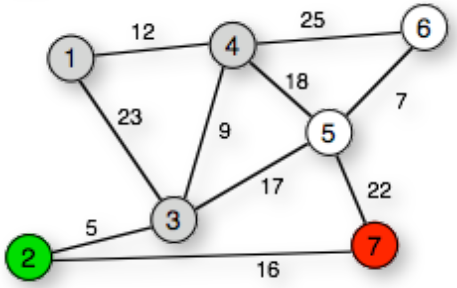
3)



Q = (2, **3**, 5, 6, 7)

node	1	2	**3**	4	5	6	7
d	0	26	**21**	12	30	37	oo
pred	–	3	**4**	1	4	4	–

4)



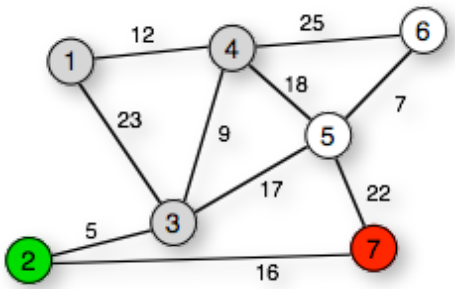
Q = (**2**, 5, 6, 7)

node	1	**2**	3	4	5	6	7
d	0	**26**	21	12	30	37	42
pred	–	**3**	4	1	4	4	2

```
Q = V
while Q is not empty:
   u = node with minimal d

   if d[u] = oo:
      break

   delete u from Q

   for each neighbor v of u:
      d_temp = d[u] + d(u,v)

      if d_temp < d[v]:
         d[v] = d_temp
         pred[v] = u

return pred[]C
```

Example contd.

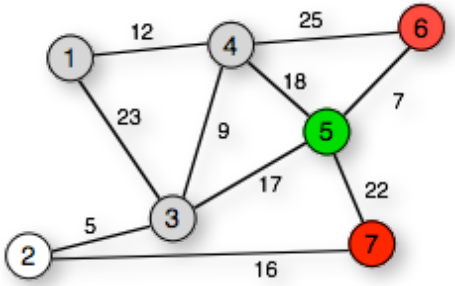
4)



Q = (**2**, 5, 6, 7)

node	1	**2**	3	4	5	6	7
d	0	**26**	21	12	30	37	42
pred	–	**3**	4	1	4	4	2

5)



Q = (**5**, 6, 7)

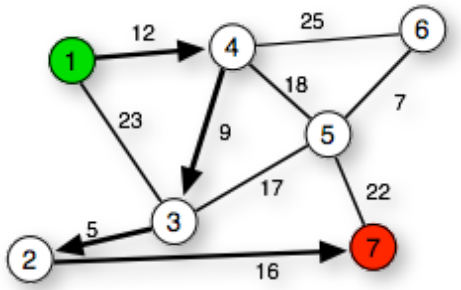
node	1	2	3	4	**5**	6	7
d	0	26	21	12	**30**	37	42
pred	–	3	4	1	**4**	4	2

Q = (**6**, 7)

Q = (**7**)

node	1	2	3	4	5	6	7
d	0	26	21	12	30	37	42
pred	–	3	4	1	4	4	2

Final result:



$d(1, 7) = 42$ path = (1, 4, 3, 2, 7)

$d(1, 6) = 37$ path = (1, 4, 6) or (1,4,5,6)

Beyond Dijkstra

Dijkstra works for directed and undirected graphs with **non-negative** weights.

Straight-forward implementation: $O(N^2)$

Graphs with positive and negative weights
→ **Bellman-Ford**-algorithm

If there is a heuristic to estimate weights:
→ improve efficiency of Dijkstra
→ **A***-algorithm

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

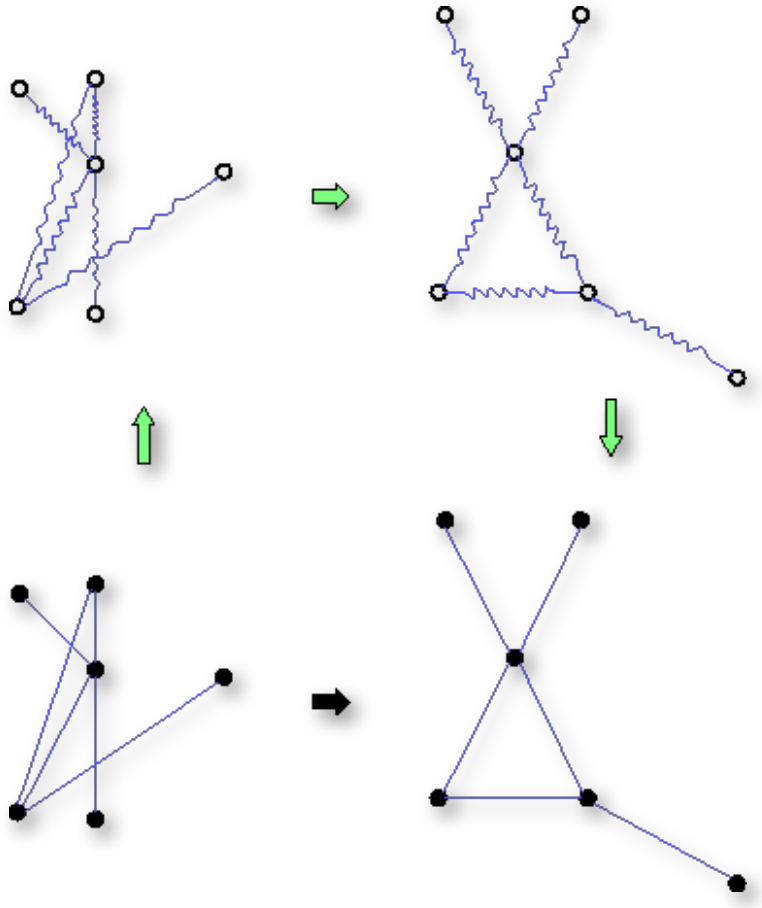
→ "**Spring Embedder**" algorithm.

- edges → springs
  vertices → rings that connect the springs

- Layout by dynamic relaxation

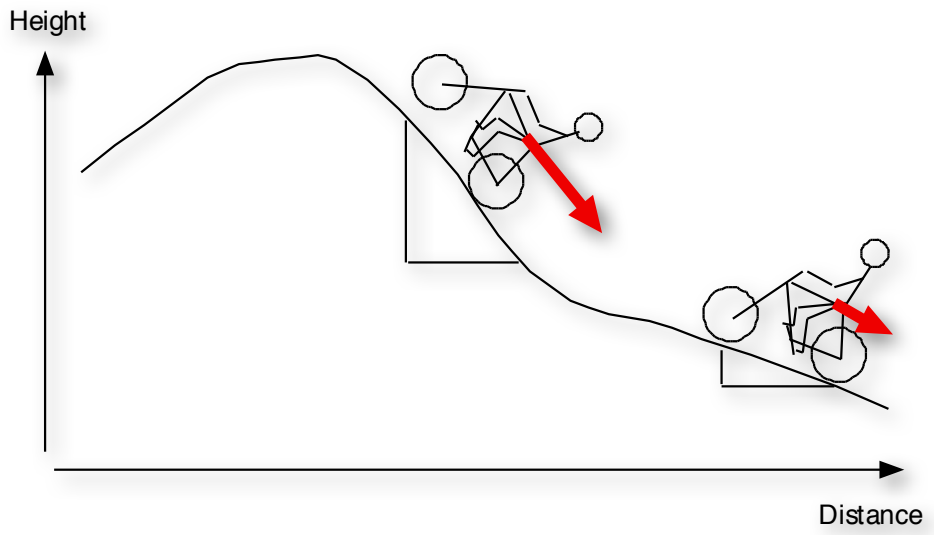
→ lowest-energy conformation

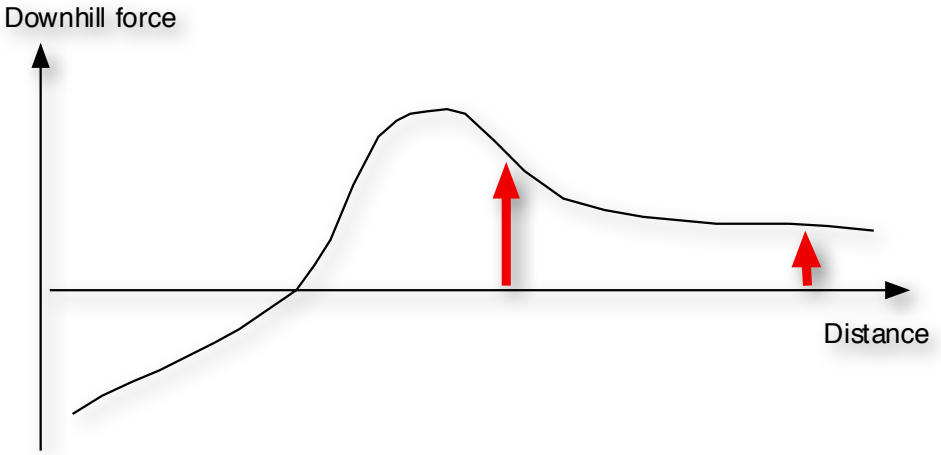
→ "**Force Directed**" algorithm



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

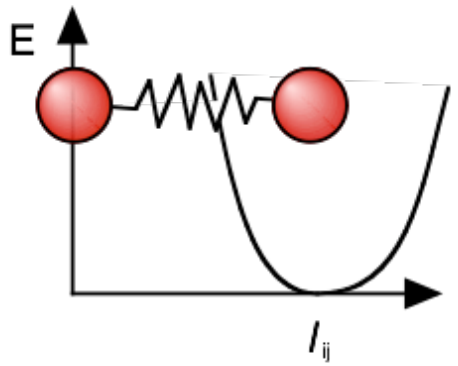
$$F(x) = -\frac{dE(x)}{dx}$$

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

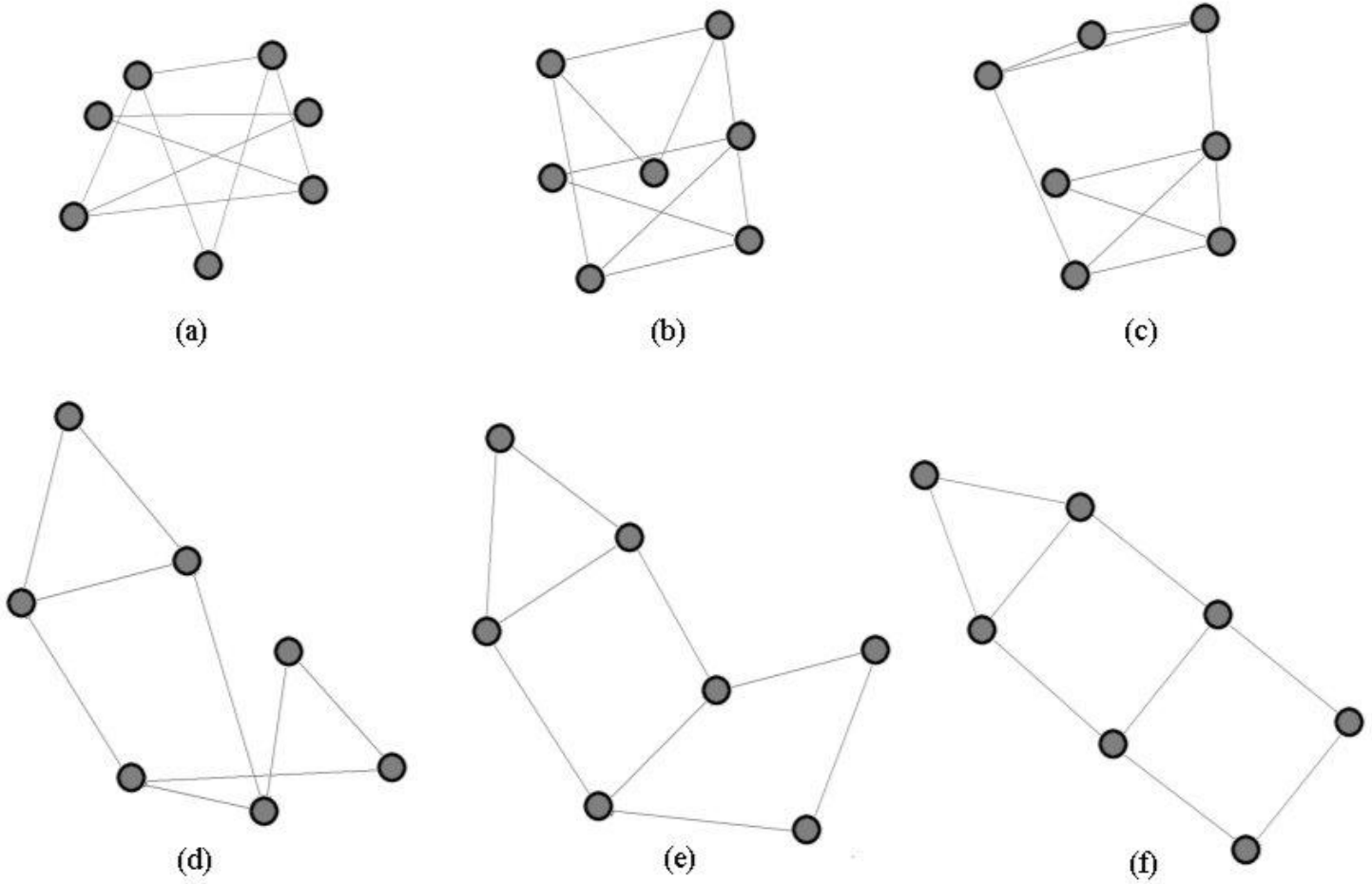
$$F_{ij}^{c} = \frac{Q_{ij}}{|r_i - r_j|^2}$$

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
```

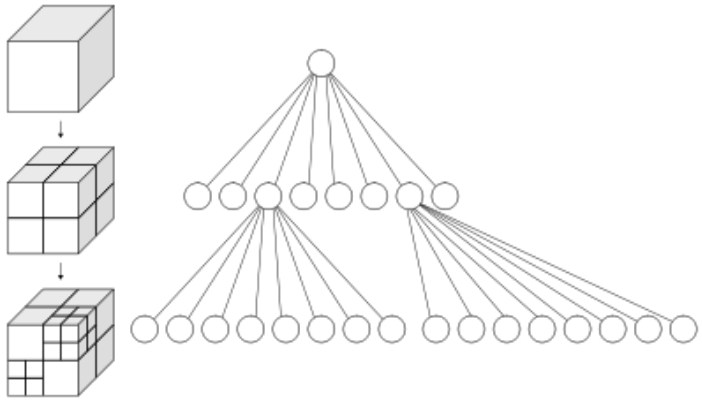
Several possible arrangements!!! (local minima)

$O(N^2)$!!!

→ 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)*



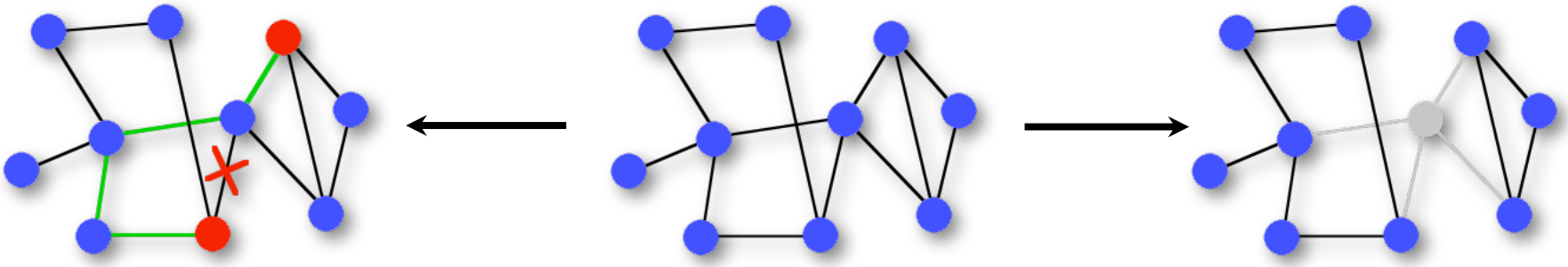
Network Robustness

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

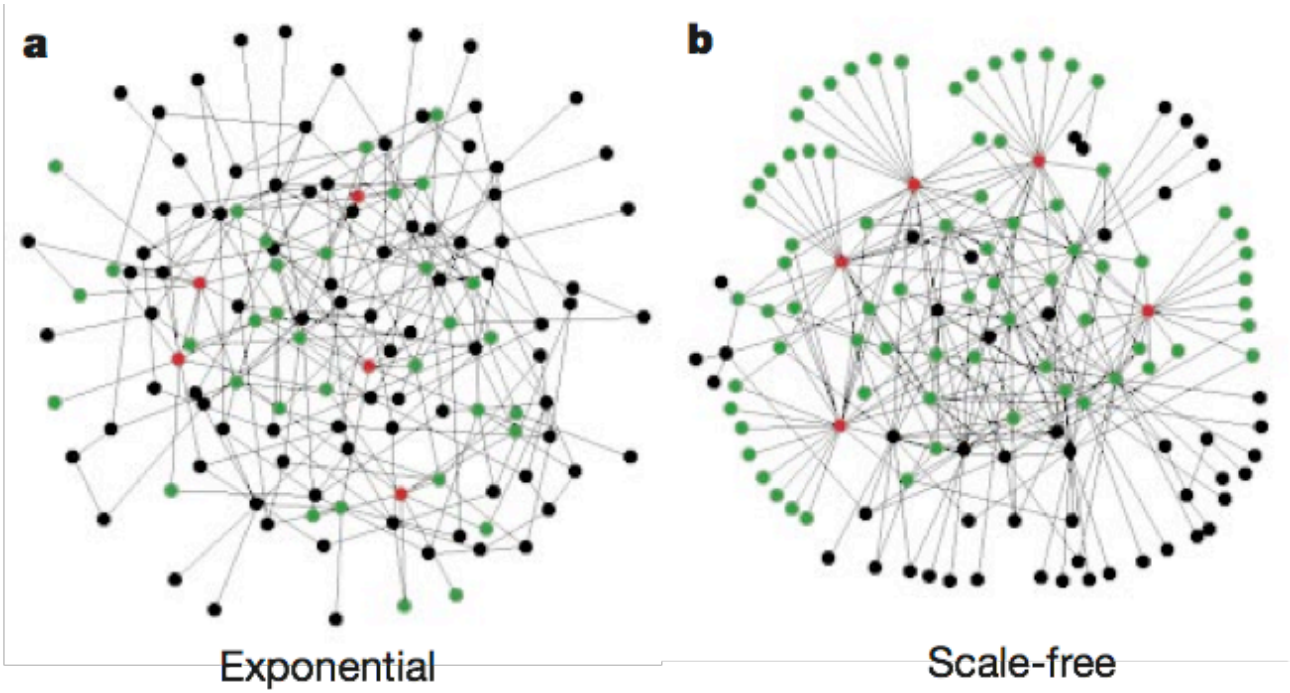
Réka Albert, Hawoong Jeong & Albert-László Barabási

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

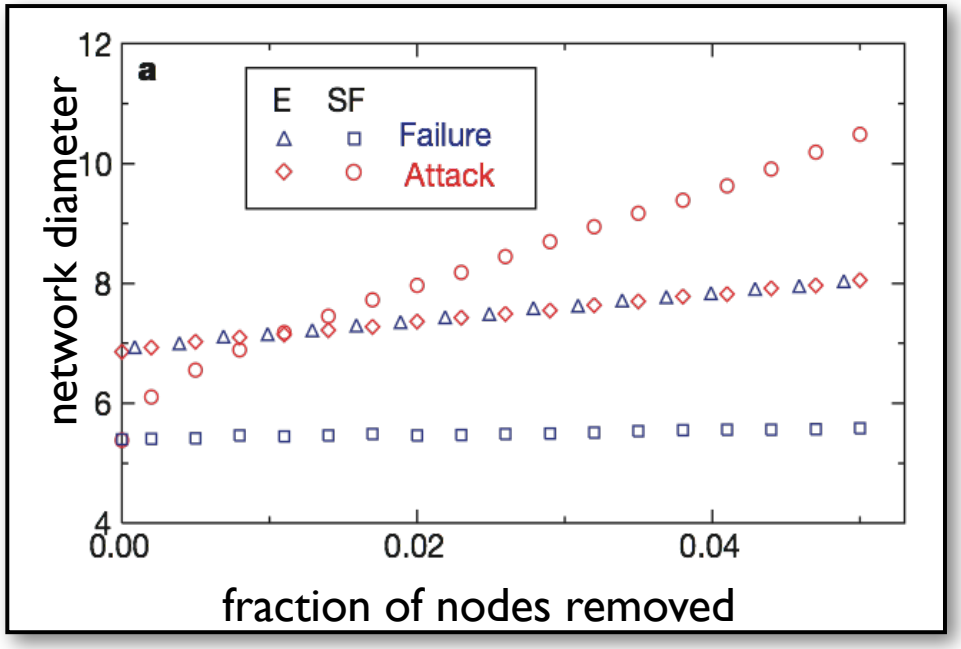
… 60% of the network

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

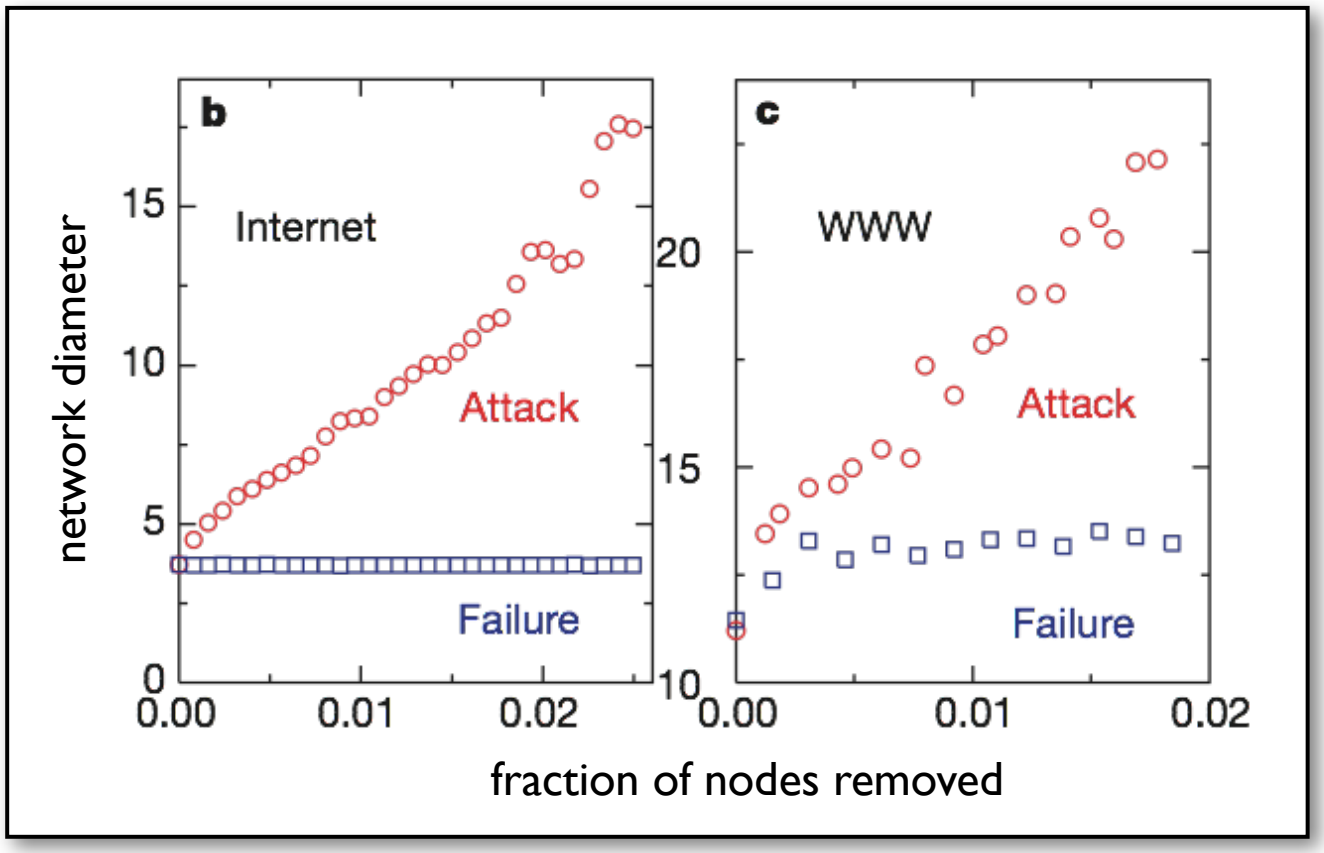
SF: failure

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)

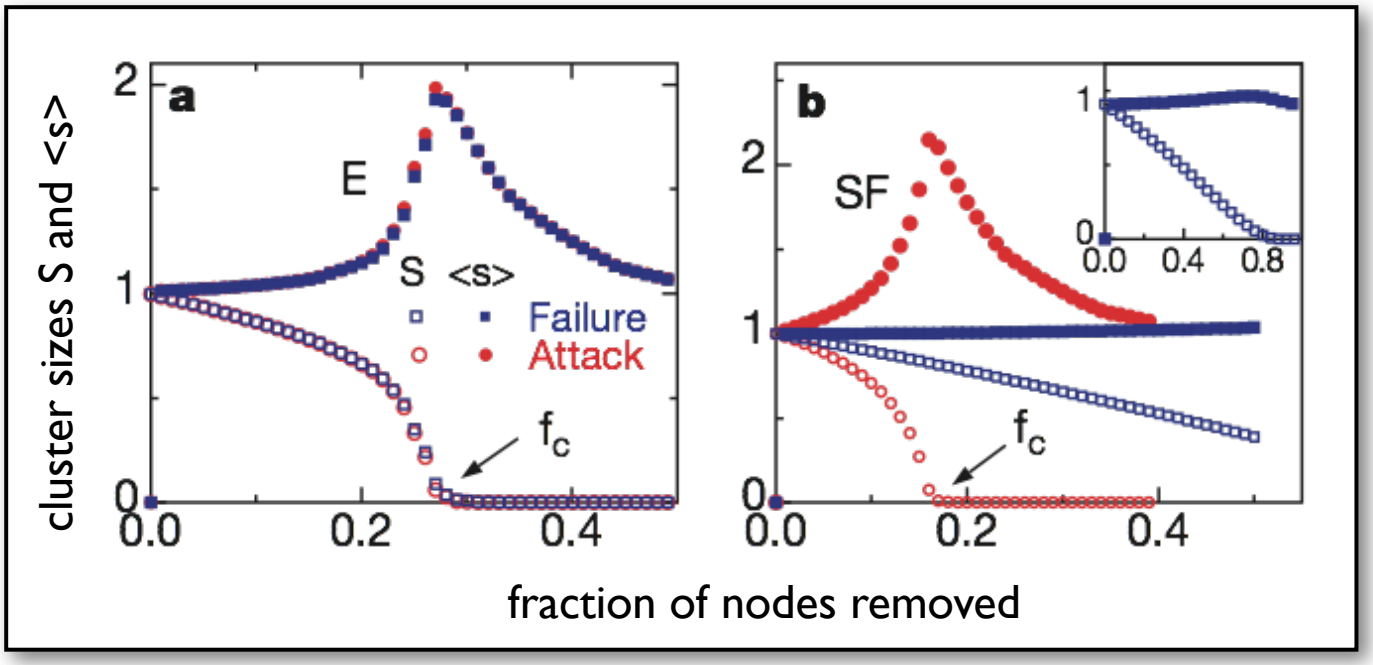
WWW-sample containing 325729 nodes and 1498353 links

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)*

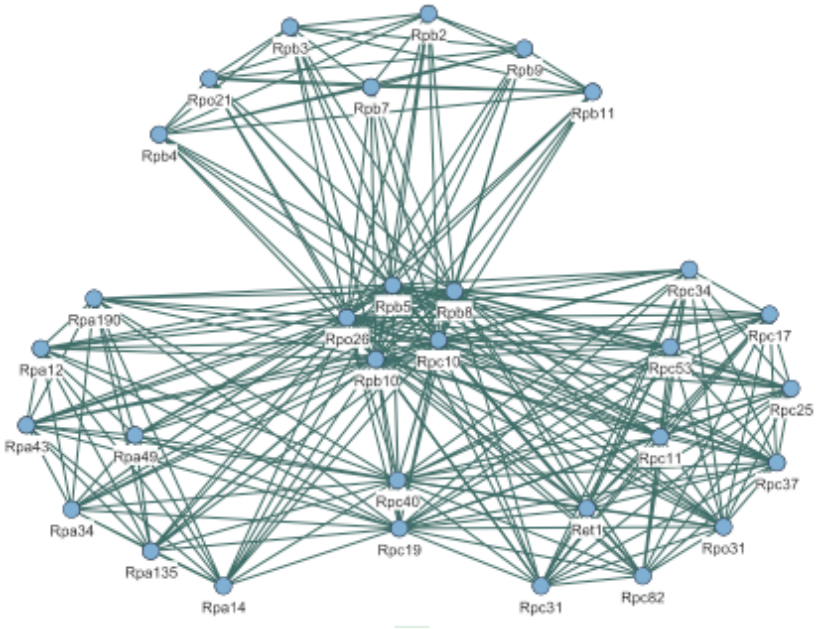


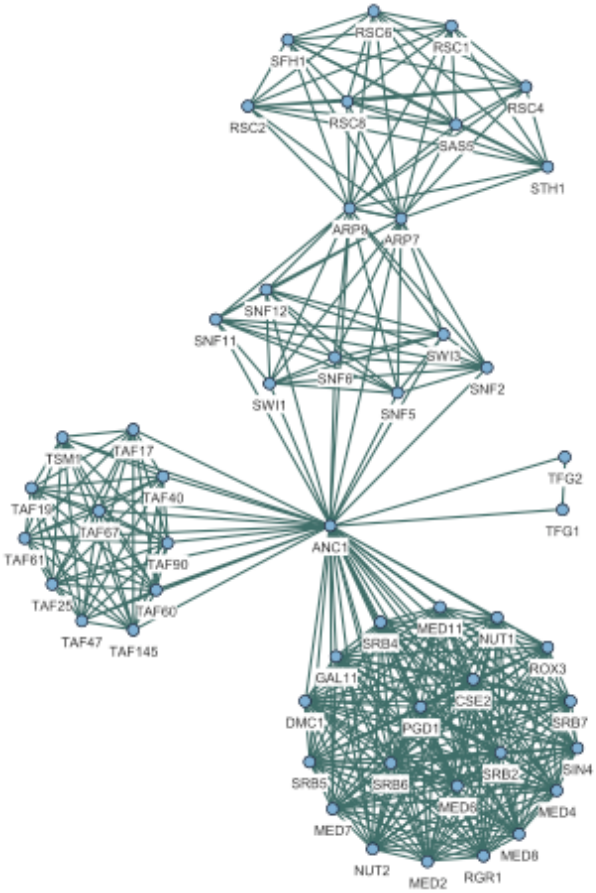
Random network:
- **no difference** between attack and failure (homogeneity)
- fragmentation threshold at $f_c \gtrsim 0.28$ ($S \approx 0$)

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.

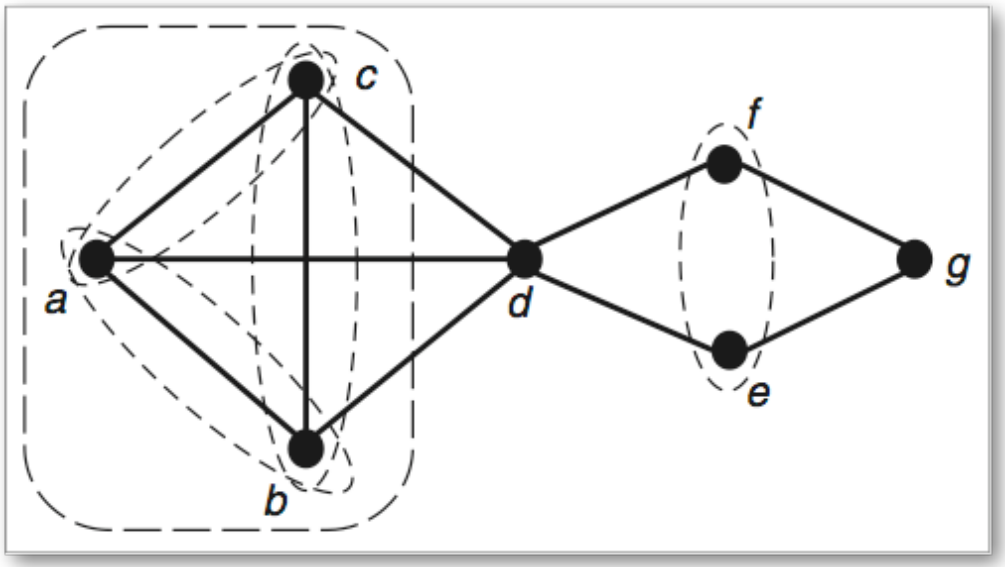
Aim: **identify** and properly **represent** the modularity of protein-protein interaction networks by identifying the **shared components** and the way they are arranged to generate **complexes**.

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}

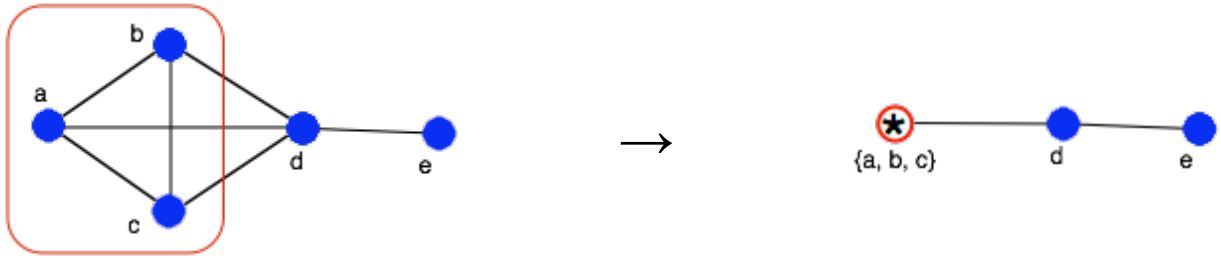
Quotient: representative node for a module

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

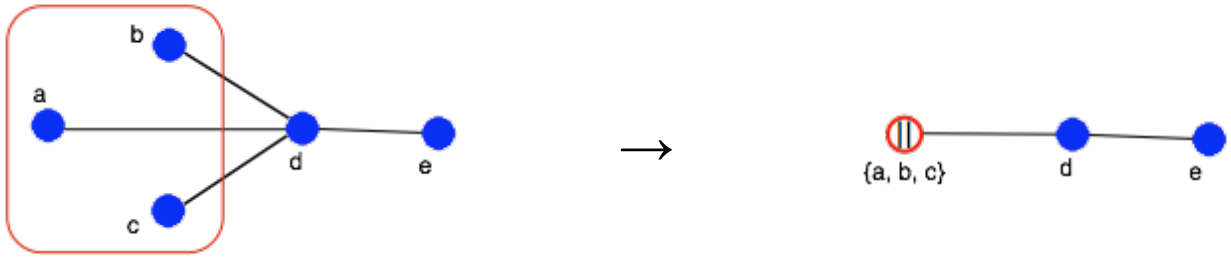
Gagneur et al, *Genome Biology* **5** (2004) R57

Quotients

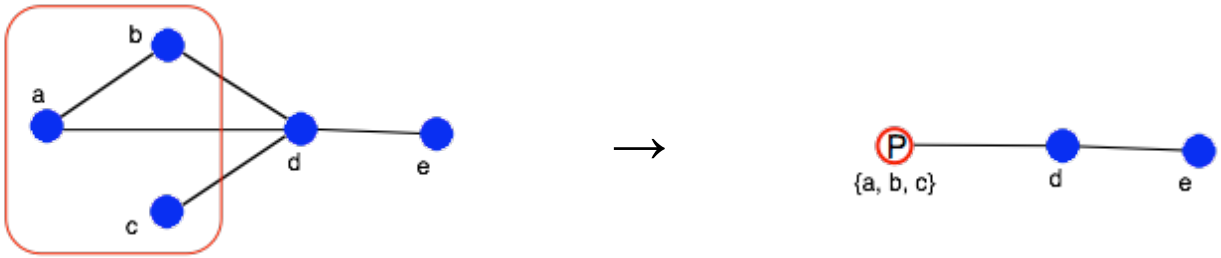
Series: all included nodes are direct **neighbors** (= **clique**)



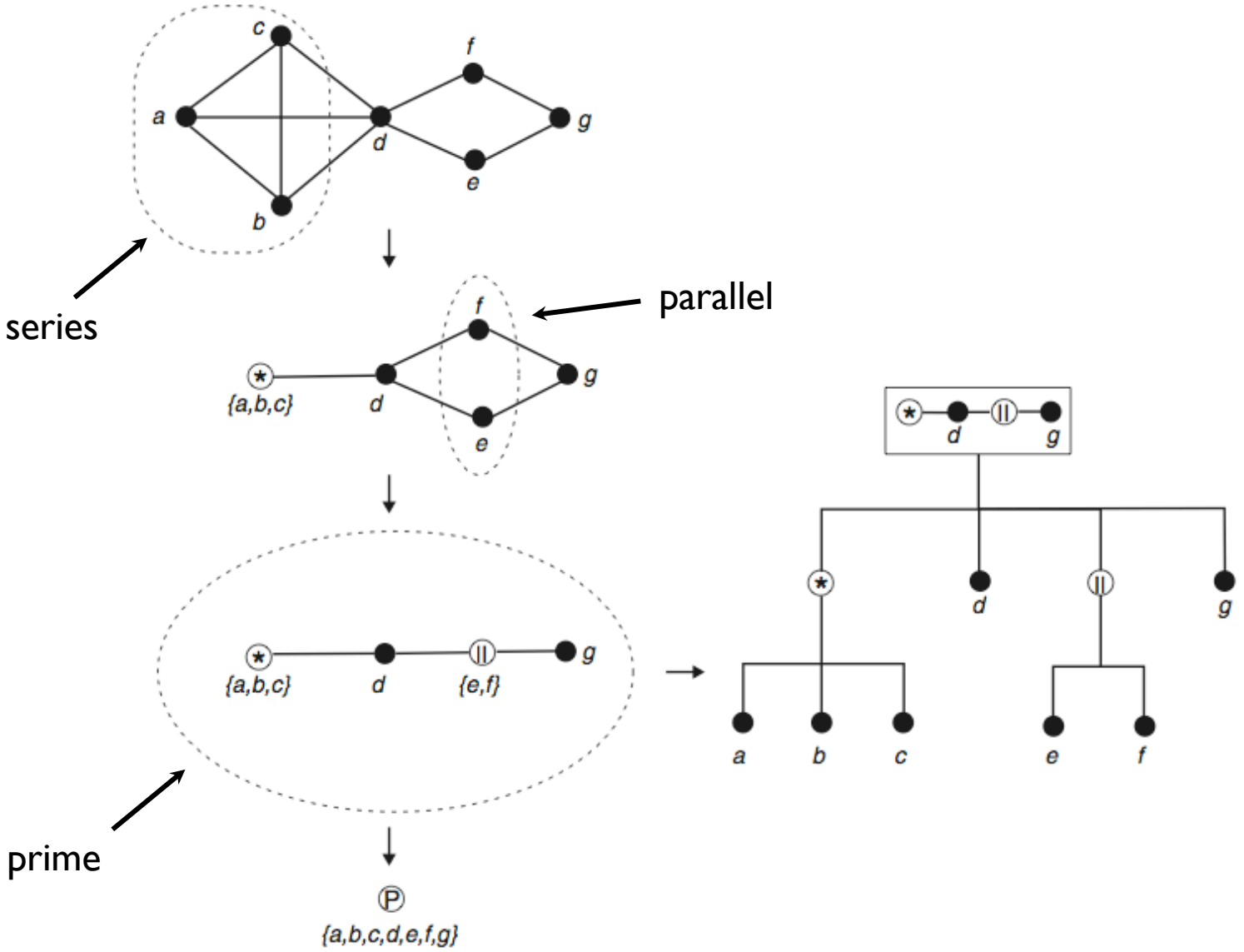
Parallel: all included nodes are **non-neighbors**



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

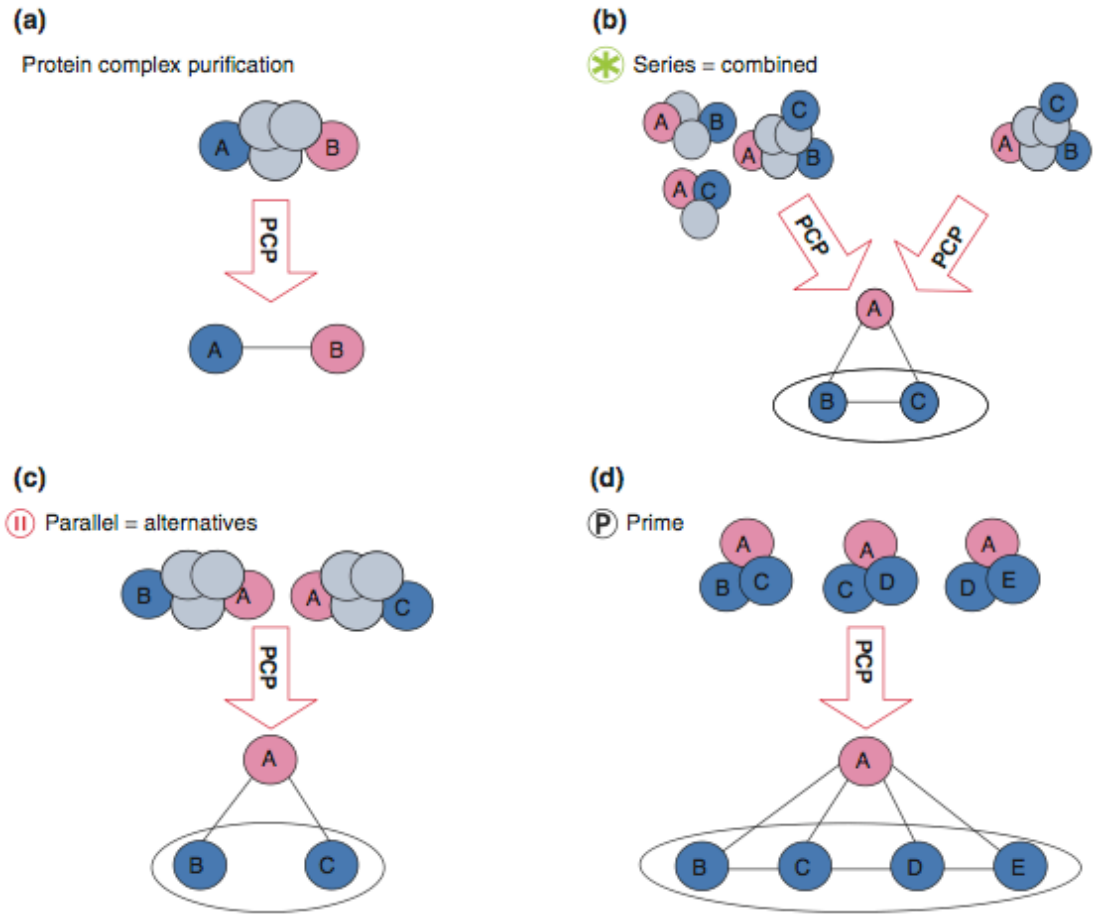
Graphs and module labels from systematic PCP experiments:

(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.

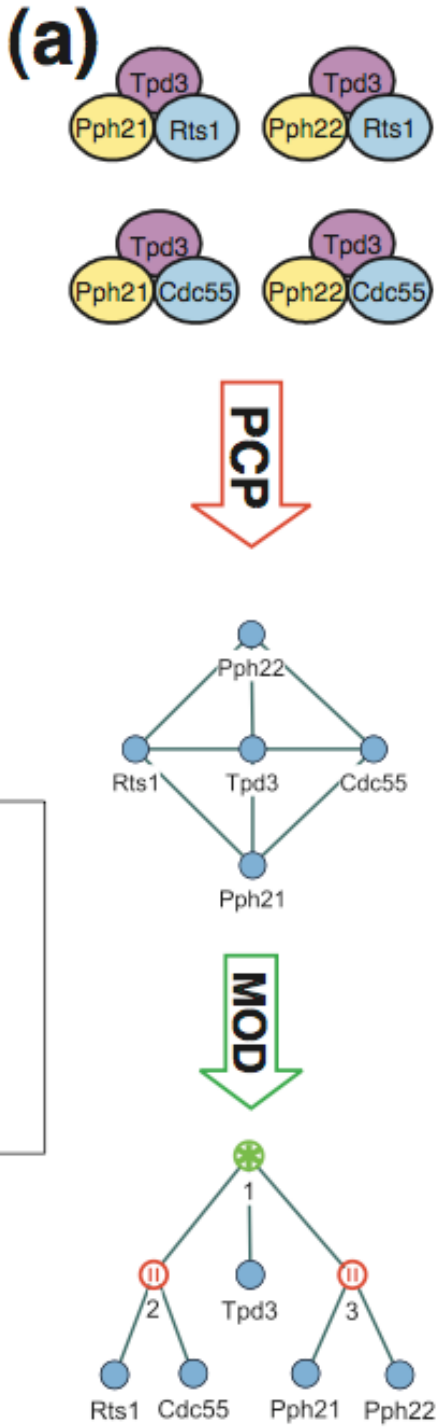
In each case from top to bottom: schemata of the complexes, the corresponding protein-protein interaction network as determined from PCP experiments, and its modular decomposition (MOD).

(a) Protein phosphatase 2A.

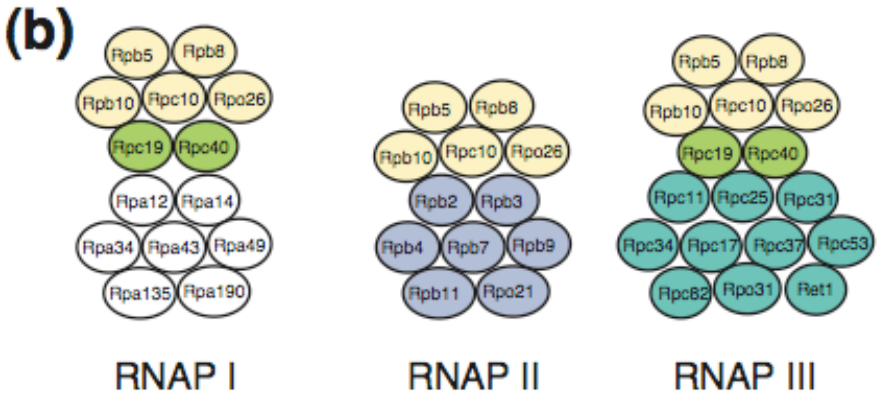
Parallel modules group proteins that do not interact but are functionally equivalent.

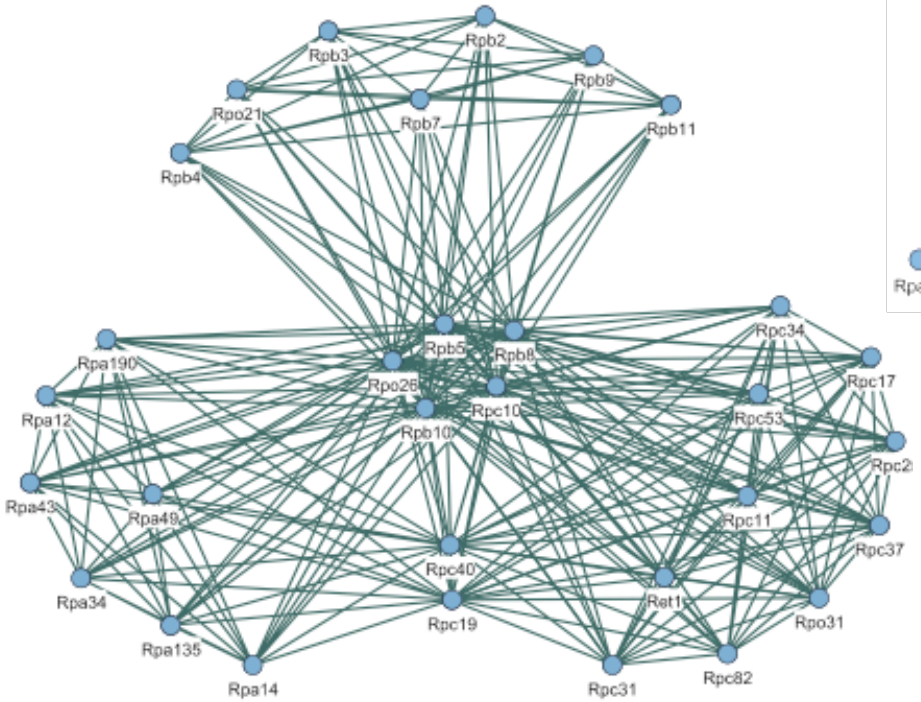
Here these are the catalytic proteins Pph21 and Pph22 (module 2) and the regulatory proteins Cdc55 and Rts1 (module 3), connected by the Tpd3 „backbone“.

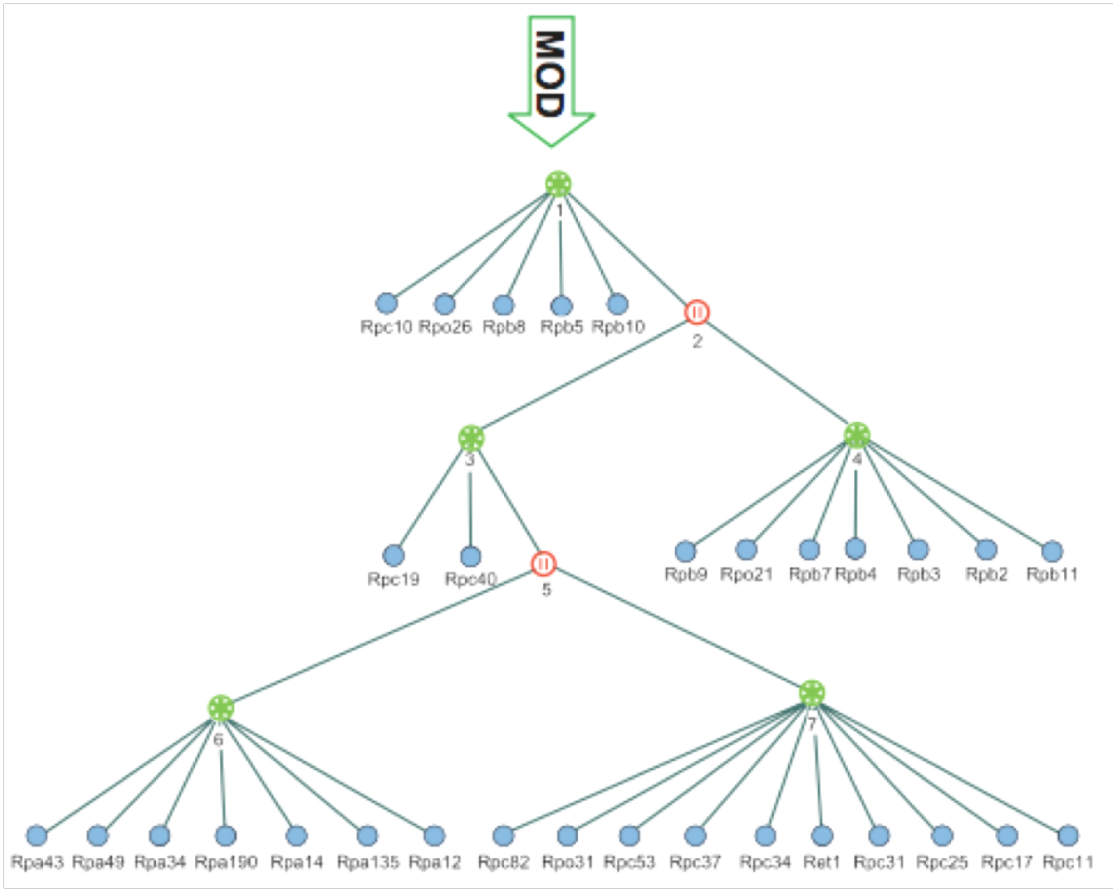
Notes: • Graph does not show functional alternatives!!!
• other decompositions also possible



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)

Summary

Modular decomposition of graphs is a **well-defined concept**.

• One can proof thoroughly for which graphs a modular decomposition exists.

• Efficient *O(m + n)* algorithms exist to compute the decomposition.

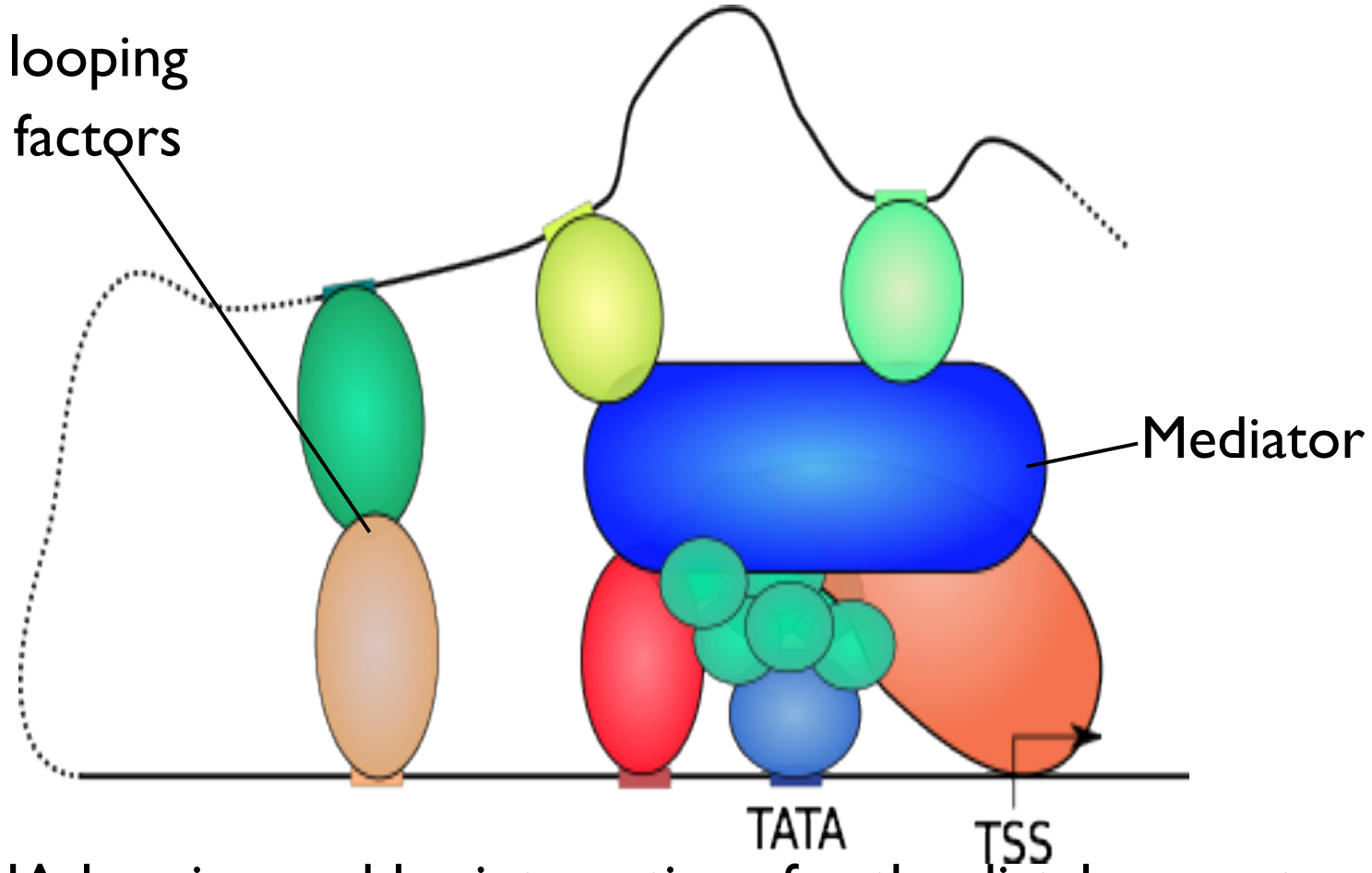
However, experiments have shown that **biological** complexes are **not strictly disjoint**. They often share components

→ separate complexes do not always fulfill the strict requirements of modular graph decomposition.

Also, there exists a „danger“ of false-positive or false-negative interactions.

→ **other methods**, e.g., for detecting communities (Girven & Newman) or densely connected clusters are **more suitable** for identification of **complexes** because they are more sensitive.

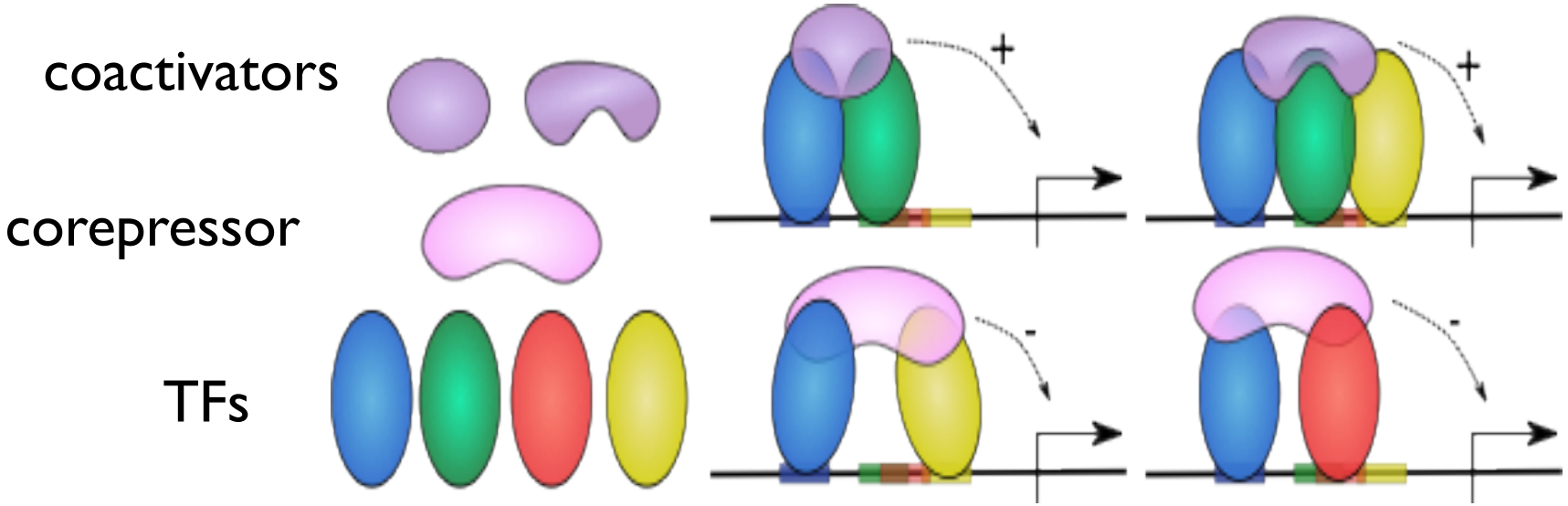
Transcriptional activation



DNA-looping enables interactions for the distal promotor regions,

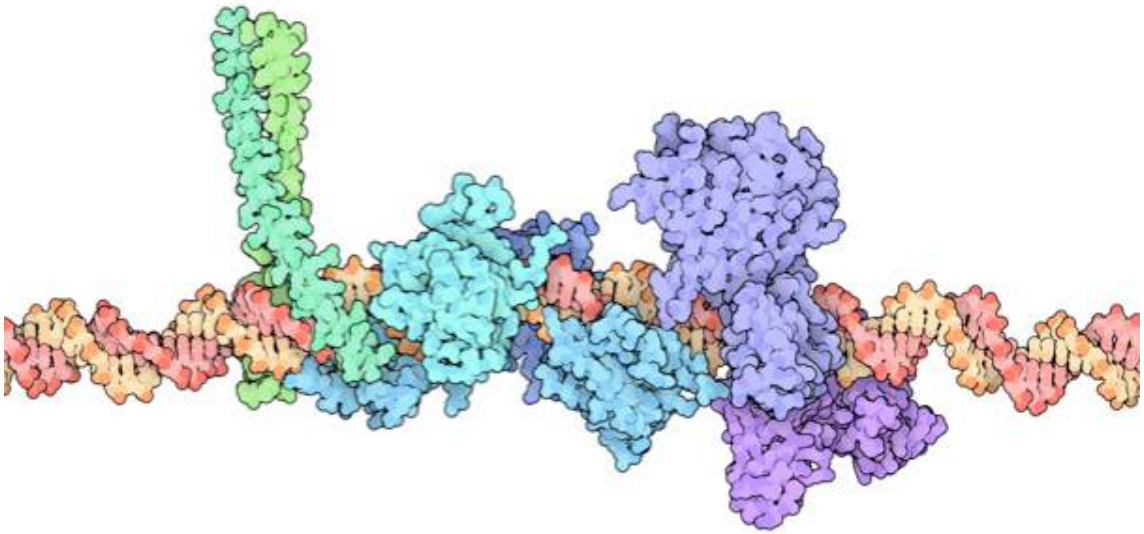
Mediator cofactor-complex serves as a huge linker

cis-regulatory modules



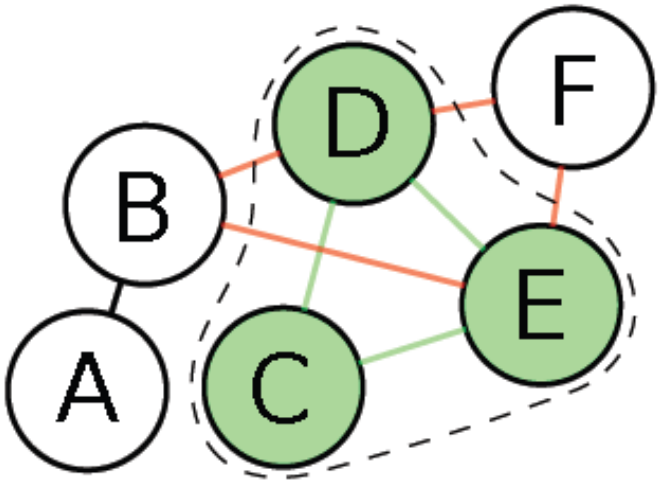
TFs are not dedicated activators or respressors!

It‘s the assembly that is crucial.



IFN-enhanceosome from RCSB Protein Data Bank, 2010

Protein complexes involving multiple transcription factors



Borrow idea from ClusterOne method:

Identify candidates of TF complexes

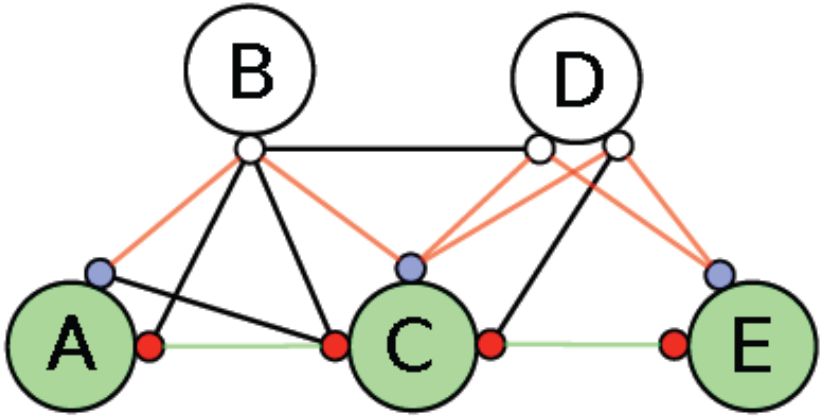
in protein-protein interaction graph

by **optimizing the cohesiveness**

$$f(V) = \frac{w^{in}(V)}{w^{in}(V) + w^{bound}(V)}$$

underlying domain-domain representation of PPIs

Assumption: every domain supports only one interaction.



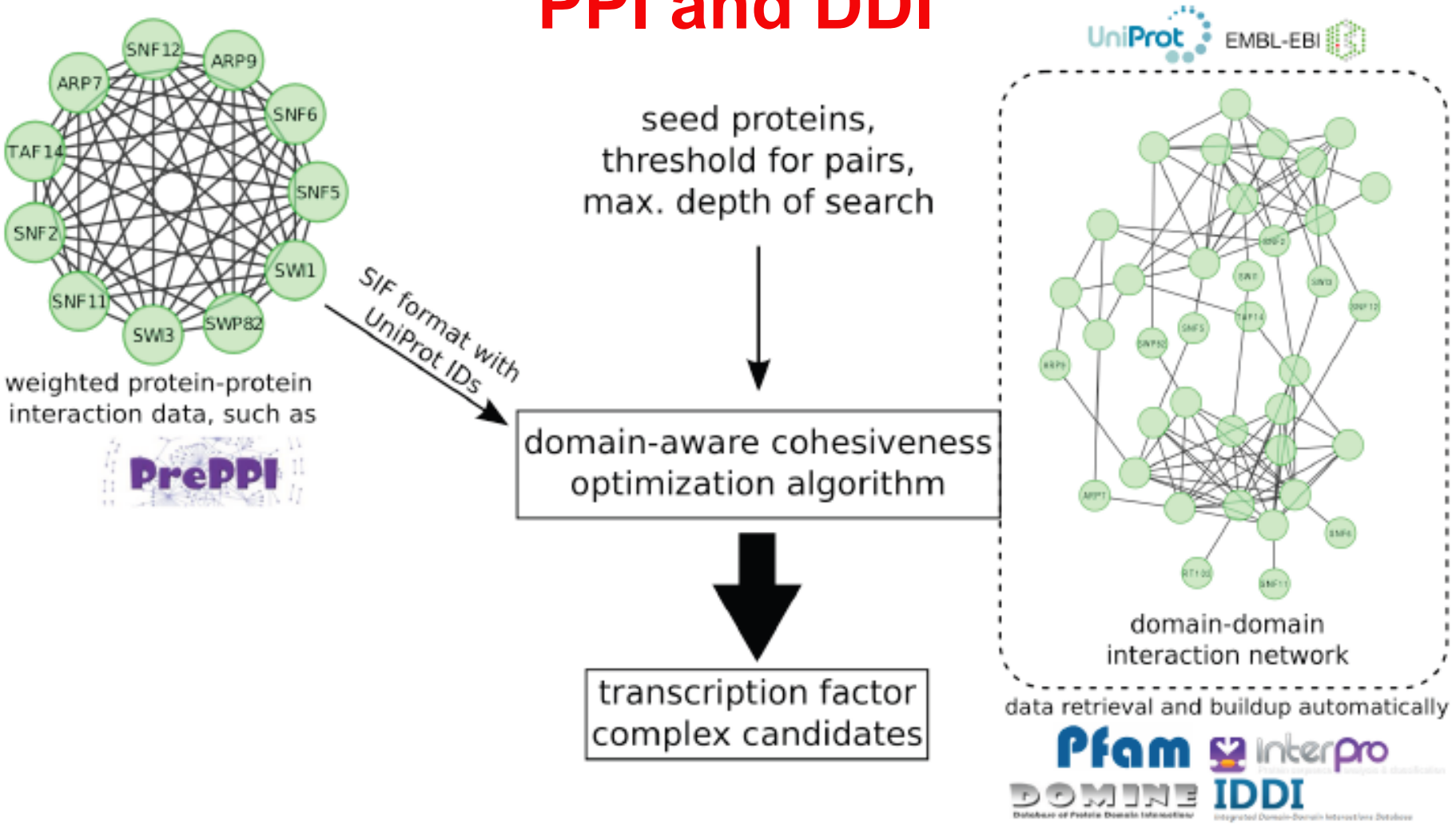
Green proteins A, C, E form actual complex.

Their red domains are connected by the two green edges.

B and D are incident proteins. They could form new interactions

(red edges) with unused domains (blue) of A, C, E

data source used: Yeast Promoter Atlas, PPI and DDI



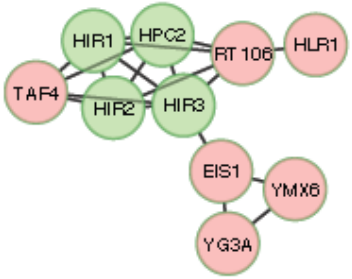
Will, T. and Helms, V. (2014)
Bioinformatics, 30, i415-i421

Daco identifies far more TF complexes than other methods

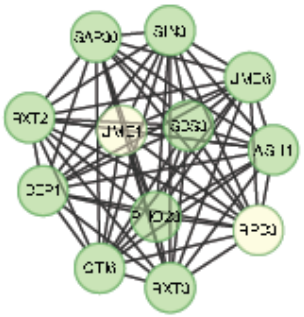
	DACO	Cl1ps	Cl1s	Cl1	MCD	MCL
TF complexes	1375	175/176	61/63	106/106	16/38	75/79
TF variants	412	134/138	59/61	80/80	16/38	75/79

Examples of TF complexes – comparison with ClusterONE

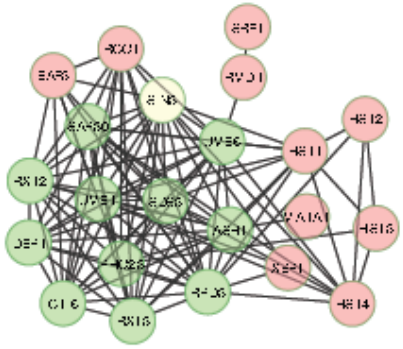
(a) HIR(SGD) / DACO



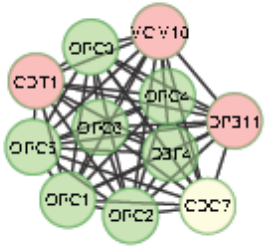
(b) HIR(SGD) / ClusterONE



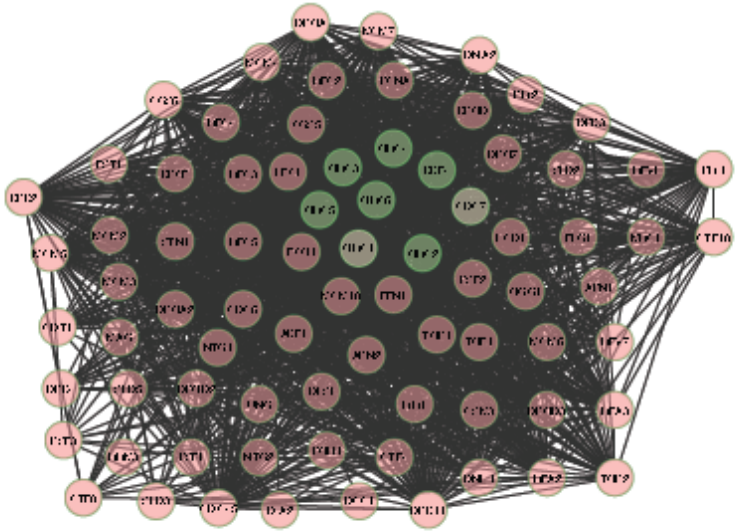
(c) RPD3L(CYC2008) / DACO



(d) RPD3L(CYC2008) / ClusterONE



(e) ORC(MIPS) / DACO

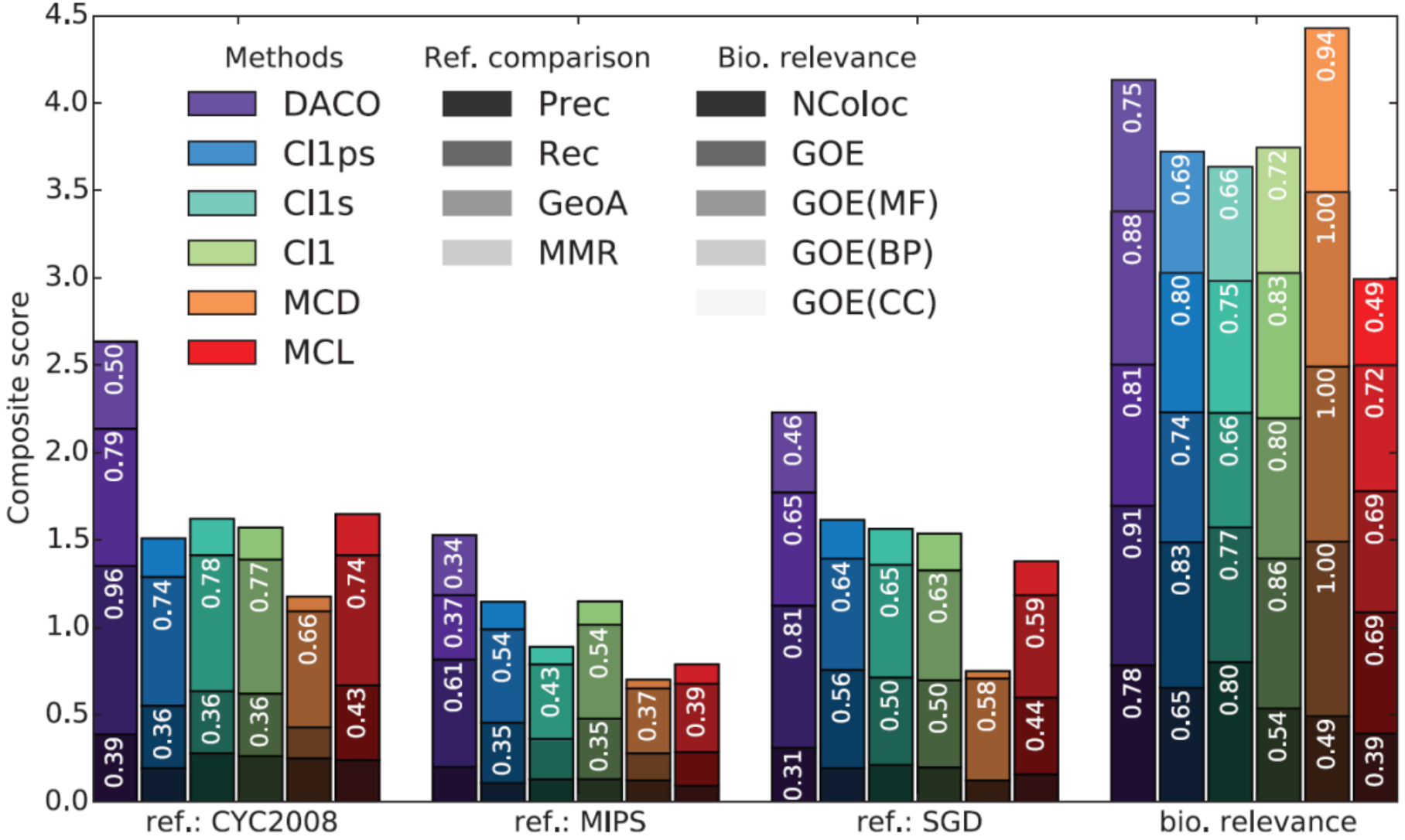


(f) ORC(MIPS) / ClusterONE

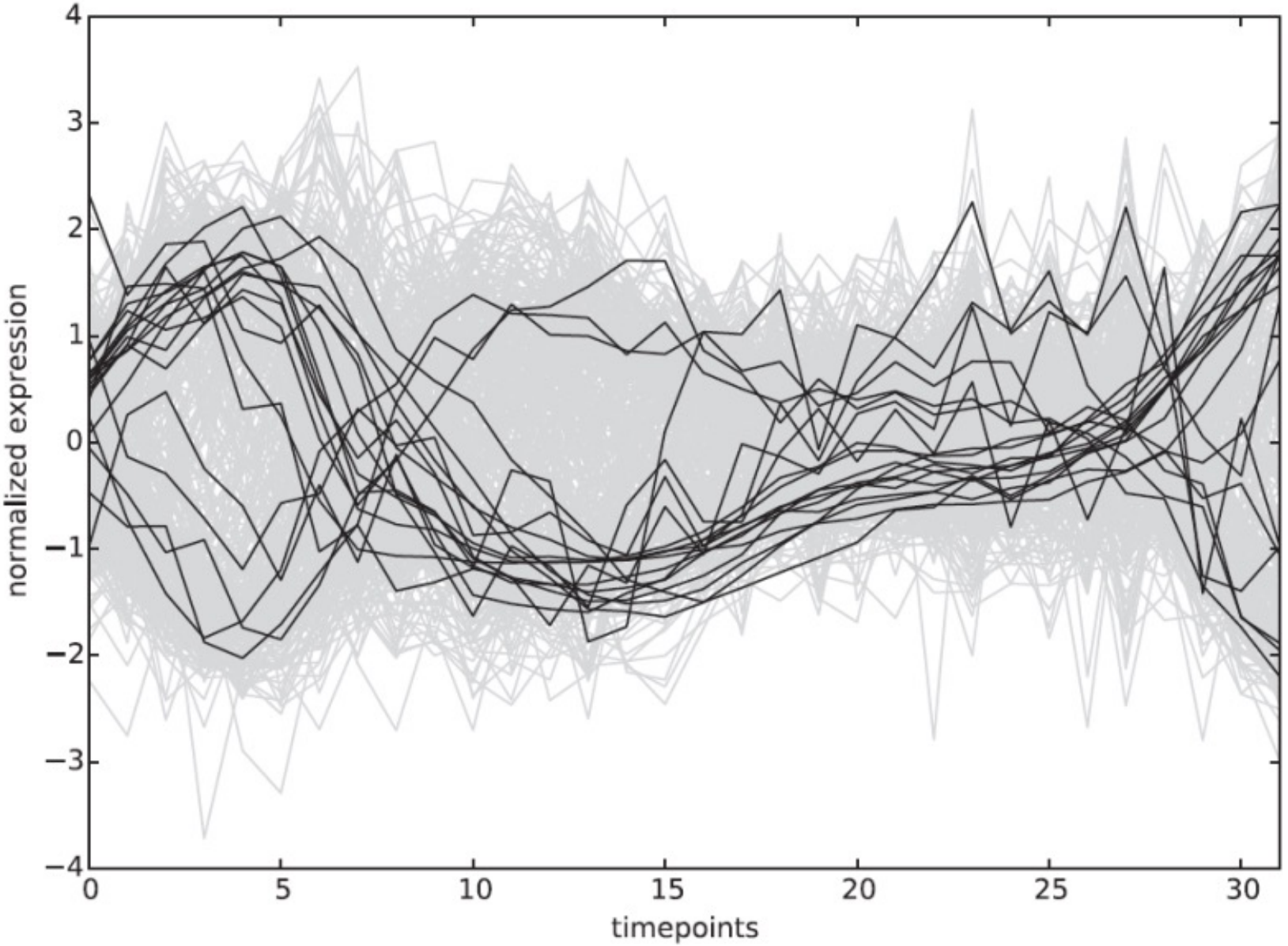
Green nodes: proteins in the reference that were matched by the prediction

red nodes: proteins that are in the predicted complex, but not part of the reference.

Performance evaluation



Co-expressed target genes of MET4/MET32 TF complex during yeast cell cycle



Functional role of TF complexes

TFs	P_{dECS}	Binding mode	Targets	Regulatory influence	GO process enrichment (P <0.05, Bonferroni corrected) in targets
MET4/MET32	0.0010	coloc.	19	+	Methionine metabolic process
TBP/HAP5	0.0335	med.	47	+	/
GLN3/DAL80	0.0009	med.	28	/	Allantoin catabolic process
DIG1/STE12/SWI6	0.0369	all	15	/	Fungal-type cell wall organization
FHL1/RAP1	0.0001	coloc.	116	+	rRNA transport
RPH1/GIS1	0.0001	med.	100	–	Hexose catabolic process
CBF1/MET32	0.0002	coloc.	33	o	Sulfate assimilation
DIG1/STE12	0.0003	med.	34	–	Response to pheromone
GCN4/RAP1	0.033	med.	62	+	/
MSN4/MSN2	0.0021	med.	105	+	Oligosaccharide biosynthetic process
DAL80/GZF3	0.0044	med.	20	–	Purine nucleobase metabolic process
SWI6/SWI4	0.0039	med.	53	+	Regulation of cyclin-dependent protein serine/threonine kinase activity
STB1/SWI6	0.0275	all	47	+	/
TBP/SWI6	0.0159	med.	14	+	/
GLN3/GZF3	0.0120	adj.	31	/	Allantoin catabolic process
MBP1/SWI6/SWI4	0.0307	med.	18	+	Regulation of cyclin-dependent protein serine/threonine kinase activity
MBP1/SWI6	0.0124	adj.	25	/	Cell cycle process

Note: Owing to the number of permutations of the test, the lowest possible value is $P_{dECS} = 10^{-4}$. The calculations were conducted for different conceivable modes of targeting (all shared target proteins, direct adjacency, mediated adjacency and colocalization) to have a detailed picture of the possible target–gene sets. Only the most enriched GO process term is shown for each target set. The inferred regulatory influence on the rate of transcription is abbreviated as follows: + (increase), – (decrease), o (no statement possible), / (conflicting annotations).

Summary

What you learned **today**:

- Graph layout: spring-electric layout algorithm
  produces aesthetic graphs
- Network **robustness**
  scale-free networks are failure-tolerant, but fragile to attacks
  <=> the few **hubs** are important
  => immunize hubs!
- **Modules** in networks
  => modular decomposition
  => optimization of cohesiveness (DACO)

Next lecture:

- Are biological networks scale-free? (other models?)
- Network growth mechanisms