V 6 – Network analysis

- Dijkstra algorithm: compute shortest pathways
- Graph layout
- Network robustness
- are biological networks really scale-free?

Tue, Nov 5, 2019

The Shortest Path Problem

Edsger Dijkstra (1930-2002):

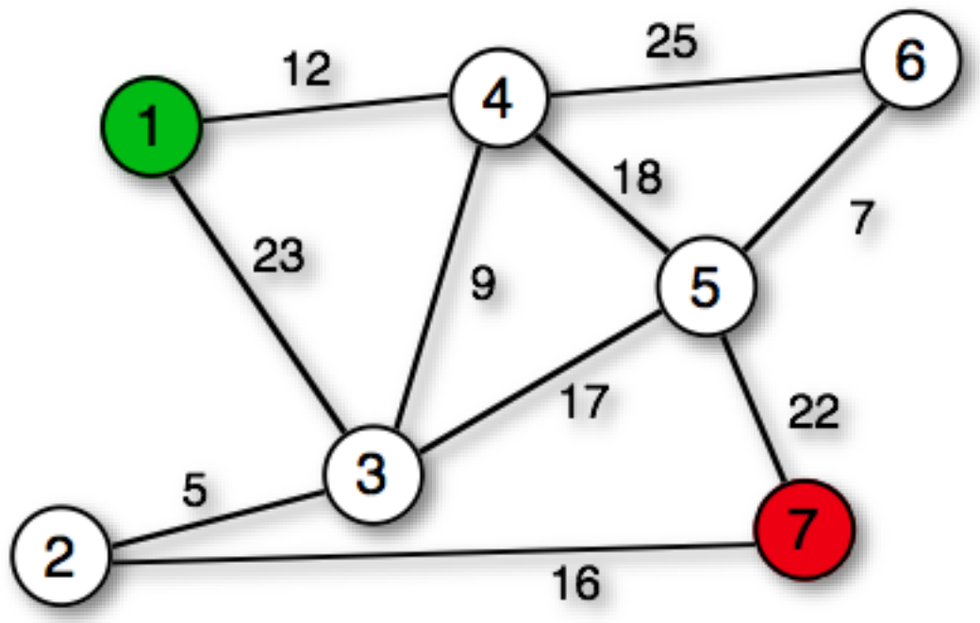
Problem:
Find the shortest path from a given vertex
to the other vertices of the graph (Dijkstra 1959).

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

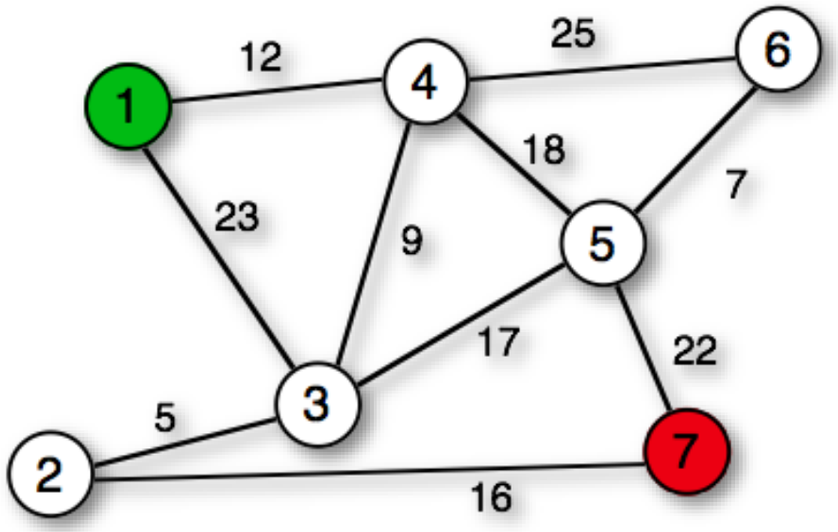
distance from source to source = 0

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*

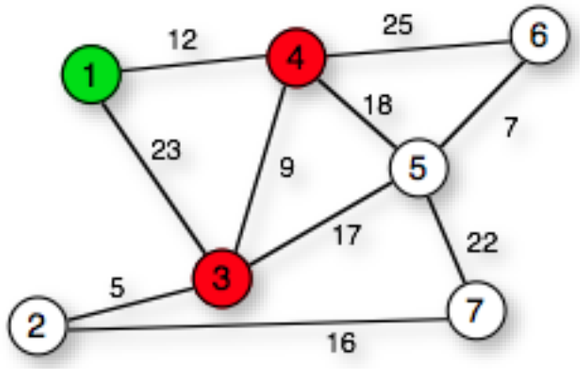
exit if all remaining nodes are inaccessible

calculate distance to *u*'s neighbors

if new path is shorter => update

Dijkstra-Example

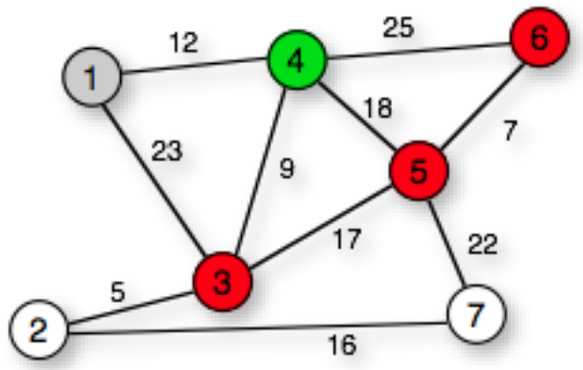
1)



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	–	–	–

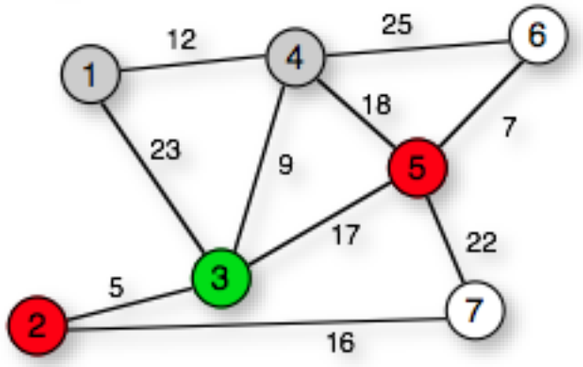
2)



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	–

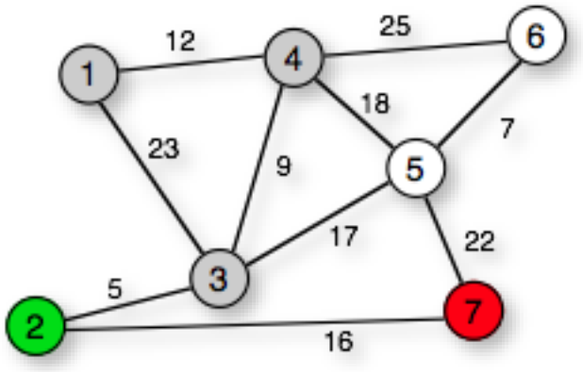
3)



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)



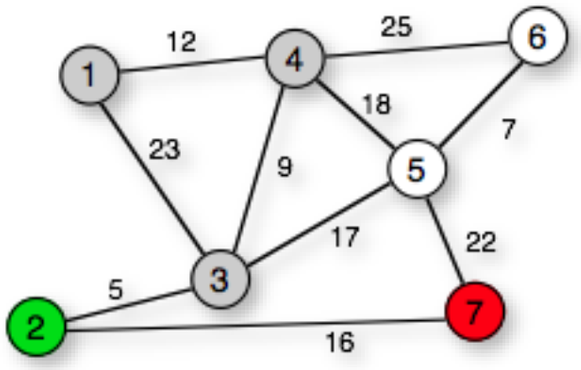
Q = (**2**, 5, 6, 7)

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.

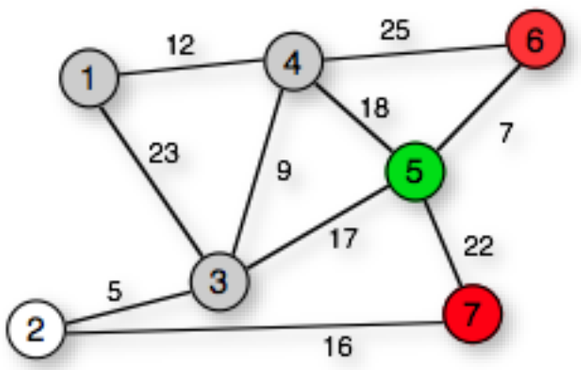
4)



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)

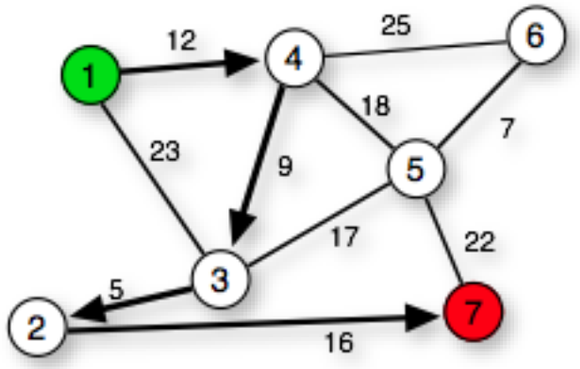
node	1	2	3	4	**5**	6	7
d	0	26	21	12	**30**	37	42
pred	–	3	4	1	**4**	4	2

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/miRNAs + regulated proteins/miRNAs
edges – regulatory interactions

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 are 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 (not covered)

Force-Directed Layout

Peter Eades (1984): graph layout heuristic

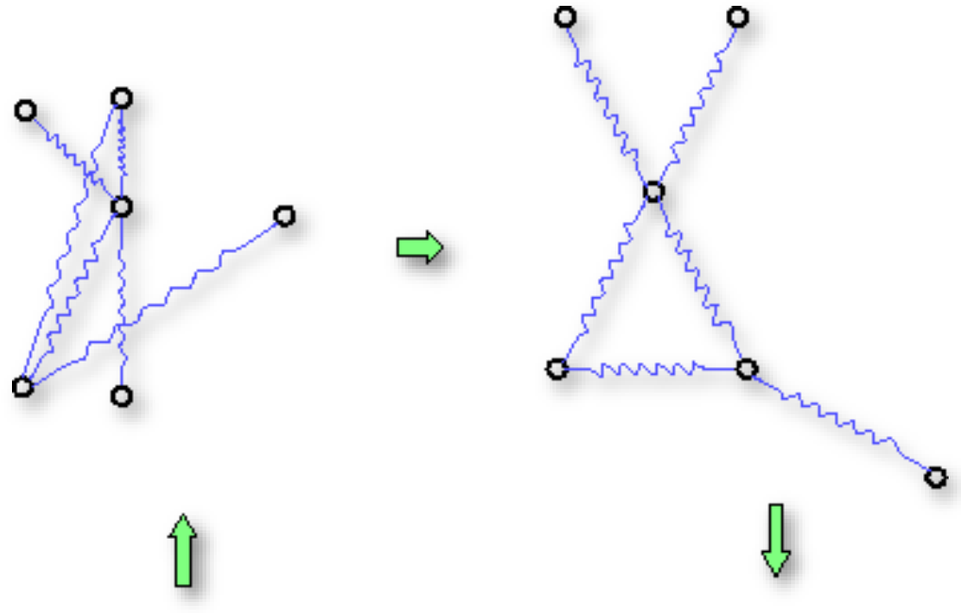
→ "**Spring Embedder**" algorithm.

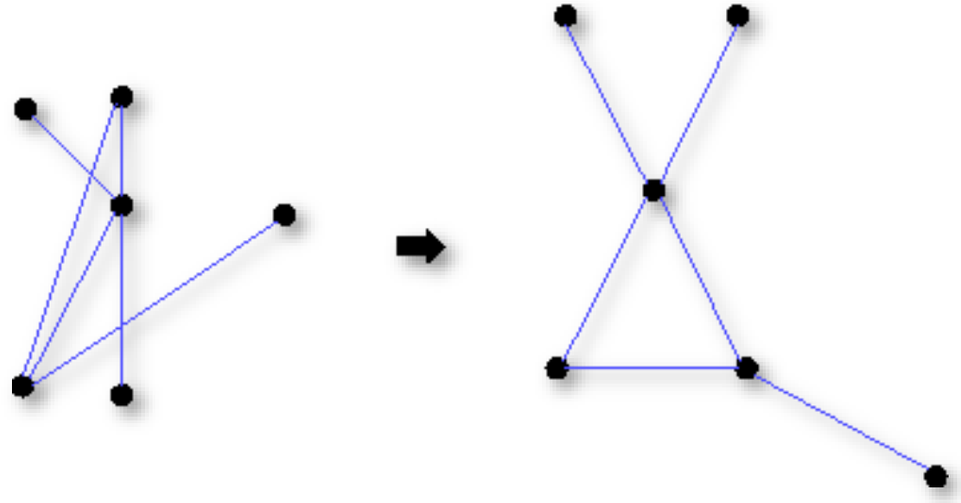
- edges → springs
  vertices → rings that connect the springs

- Layout by dynamic relaxation

→ lowest-energy conformation

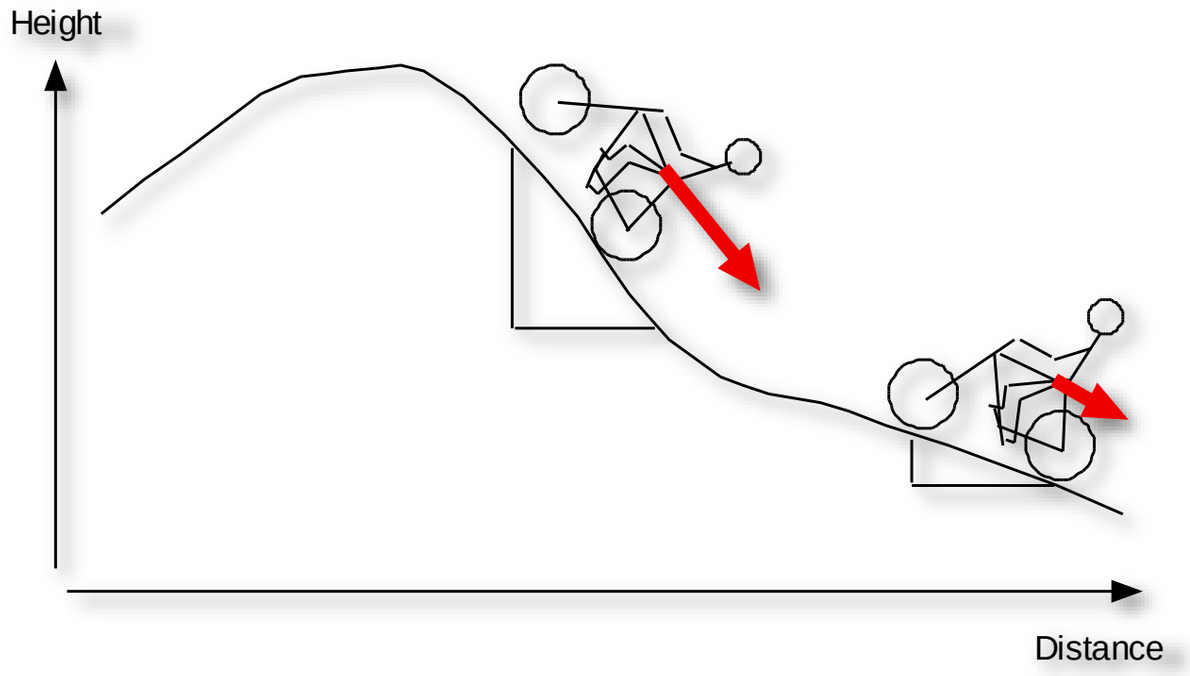
→ "**Force Directed**" algorithm

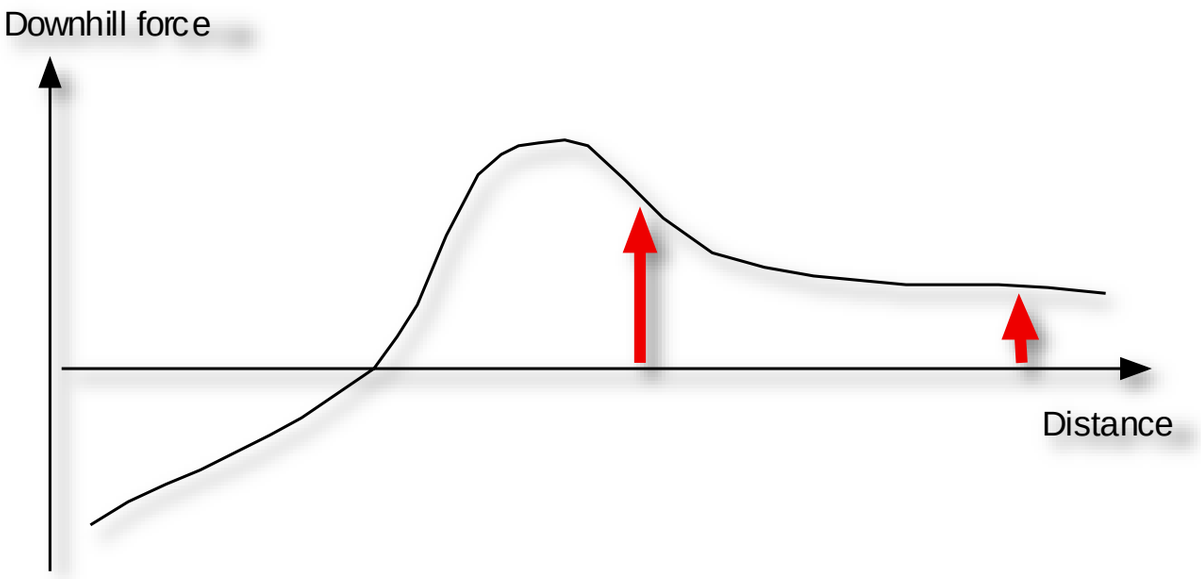




http://www.hpc.unm.edu/~sunls/research/treelayout/node1.html

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

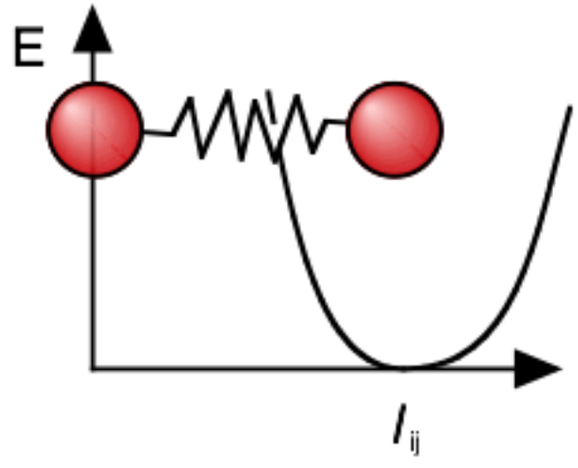
$$F(x) = -\frac{dE(x)}{dx}$$

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" $\rightarrow$ repulsion

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

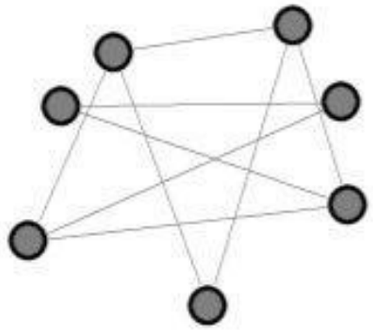
either $Q_{ij} = Q$ or, e.g., $Q_{ij} = k_i\, k_j$

Repulsion pushes all nodes apart, springs pull connected nodes together

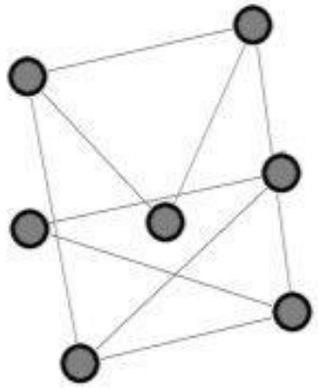
$\rightarrow$ **workhorse method** for small to medium sized graphs

$\rightarrow$ Do-it-yourself in Assignment 4 (?) <=

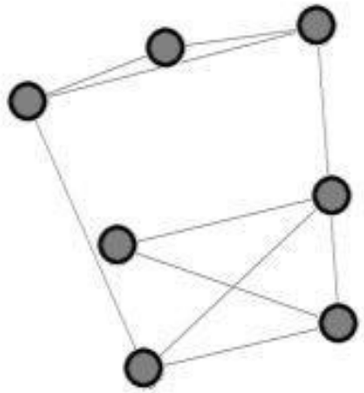
Spring-Electrical Example



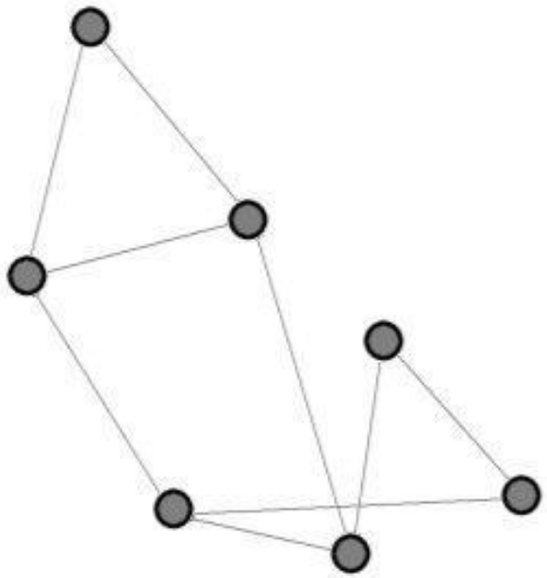
(a)



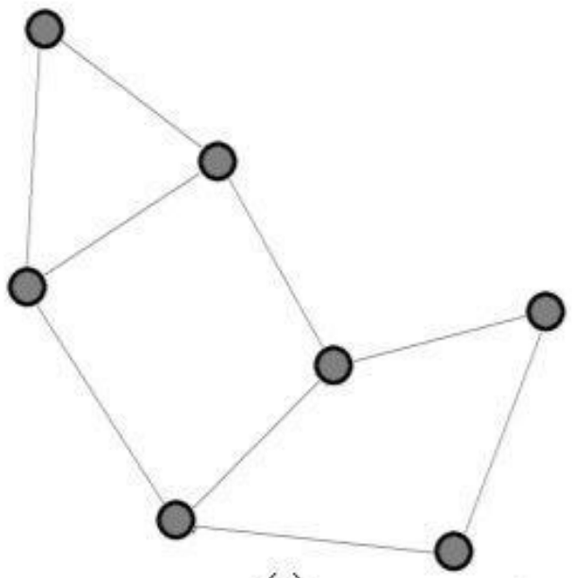
(b)



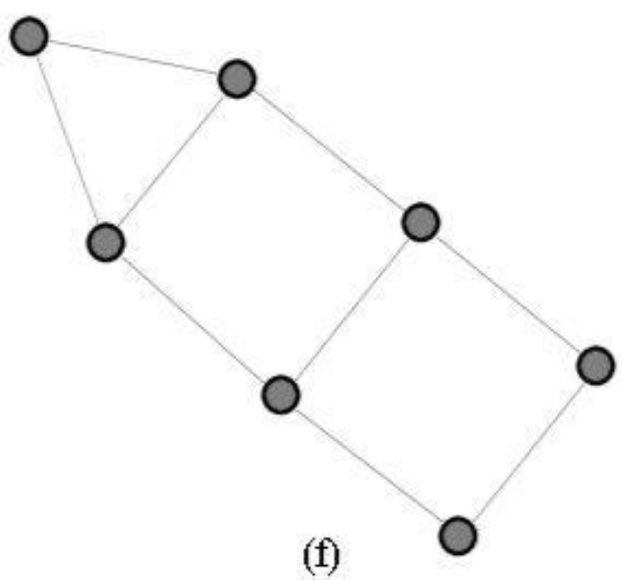
(c)



(d)



(e)



(f)

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 → 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
```

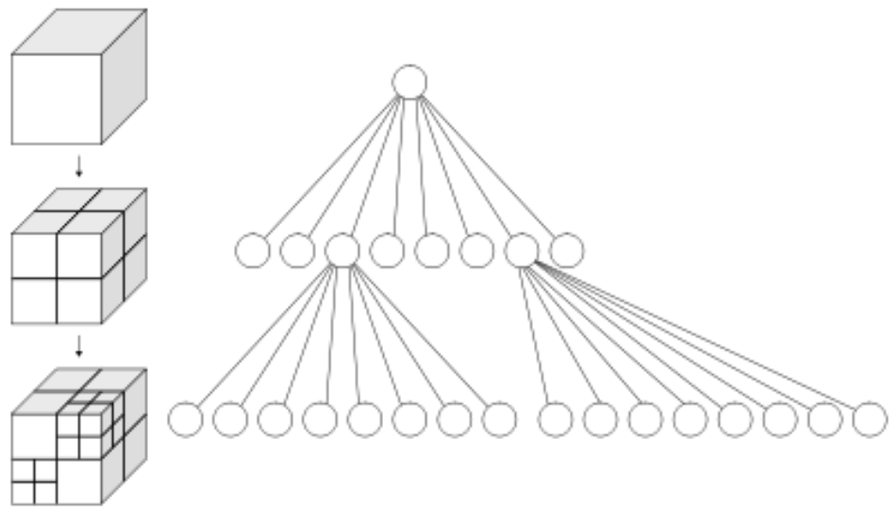
Several possible arrangements!!! (local minima)

$O(N^2)$!!!

→ 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)*



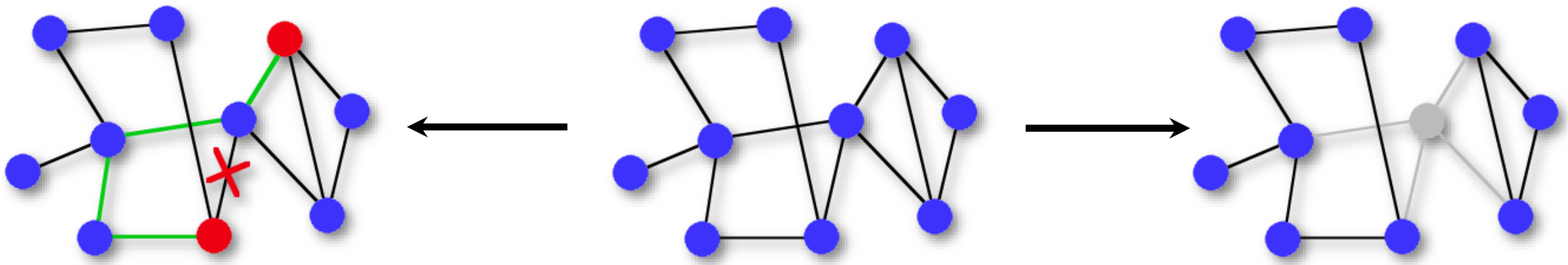
Network Robustness

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

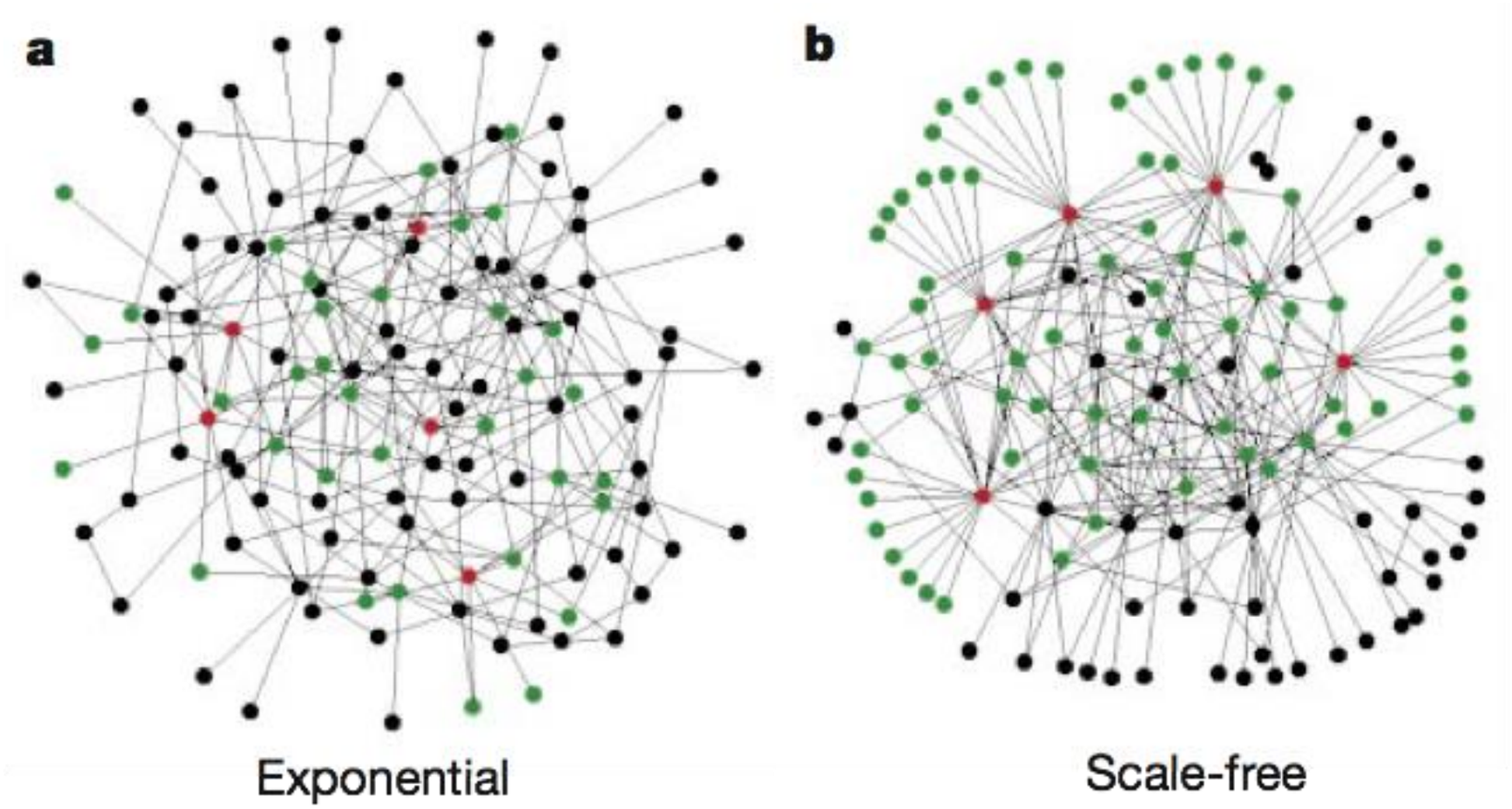
Réka Albert, Hawoong Jeong & Albert-László Barabási

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

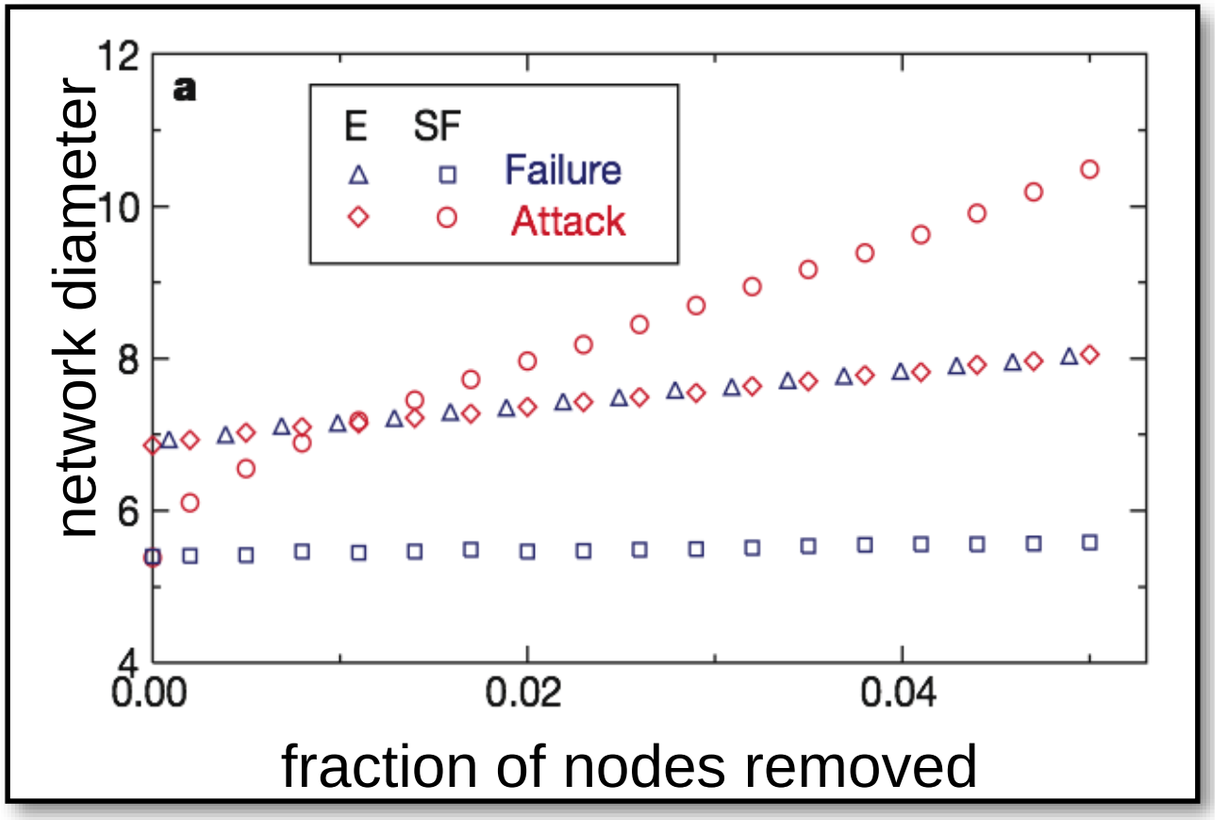
… 60% of the network

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

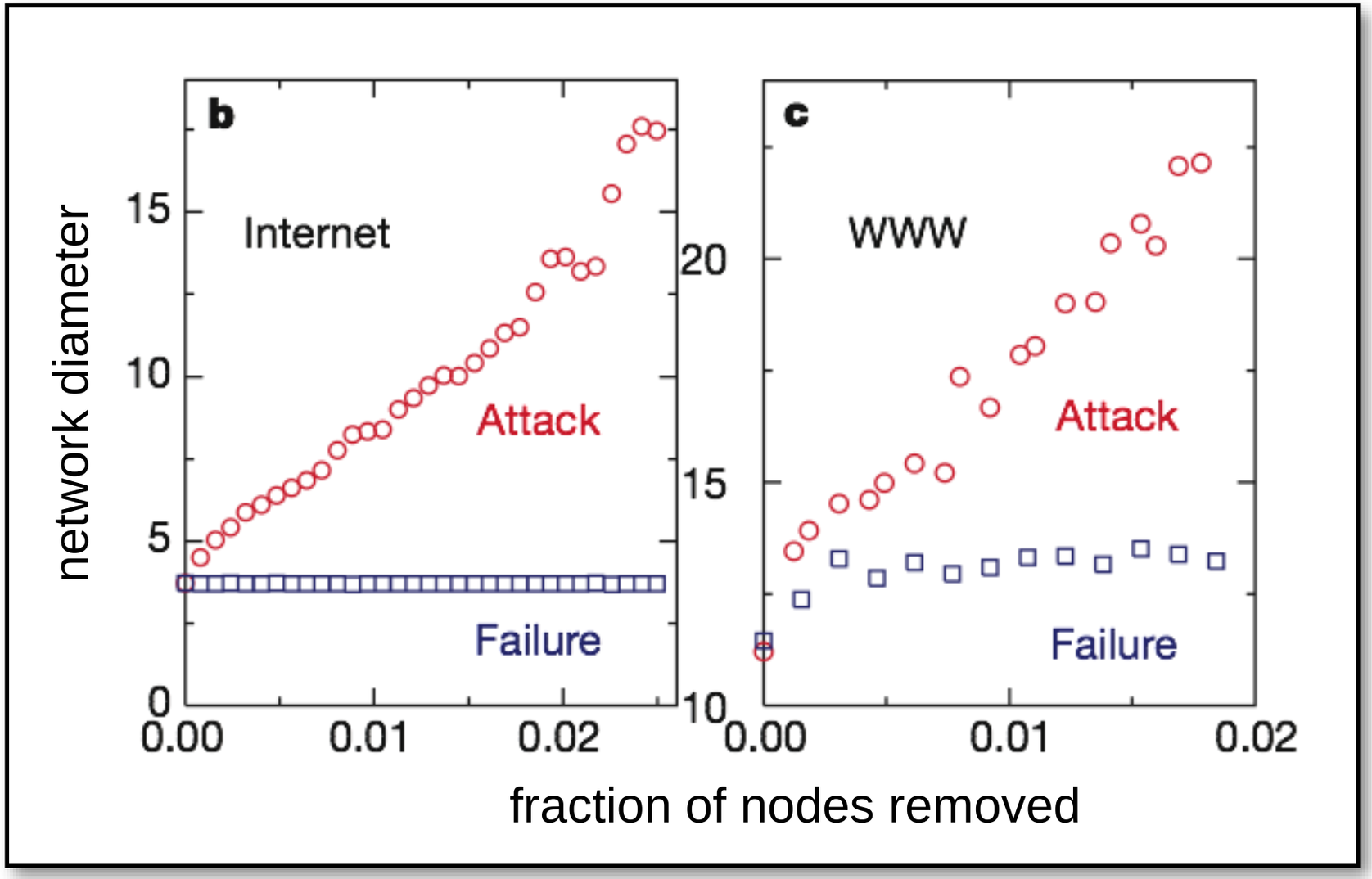
N = 10000, L = 20000, but effect is size-independent;

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)

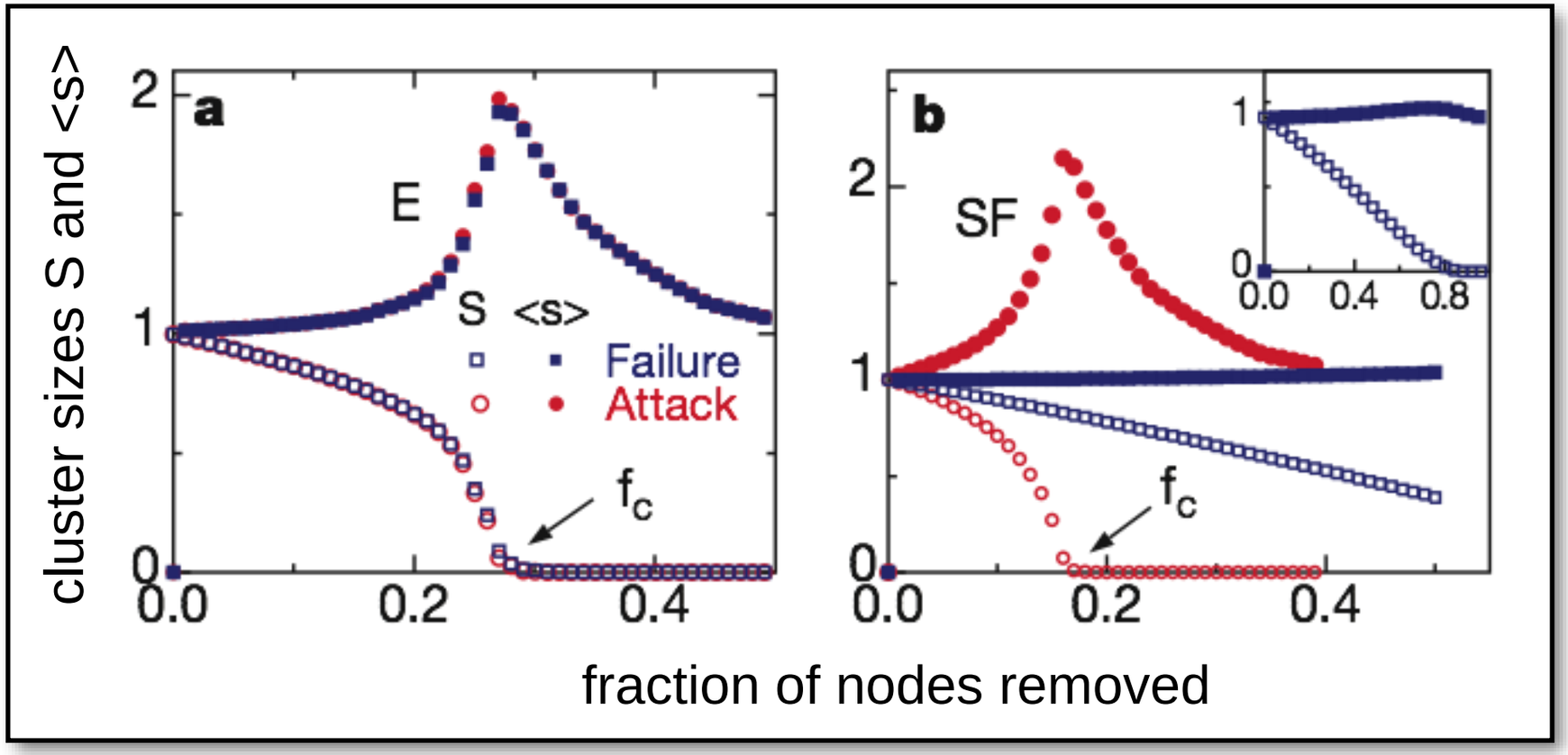
WWW-sample containing 325729 nodes and 1498353 links

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 $f_c \approx 0.18$

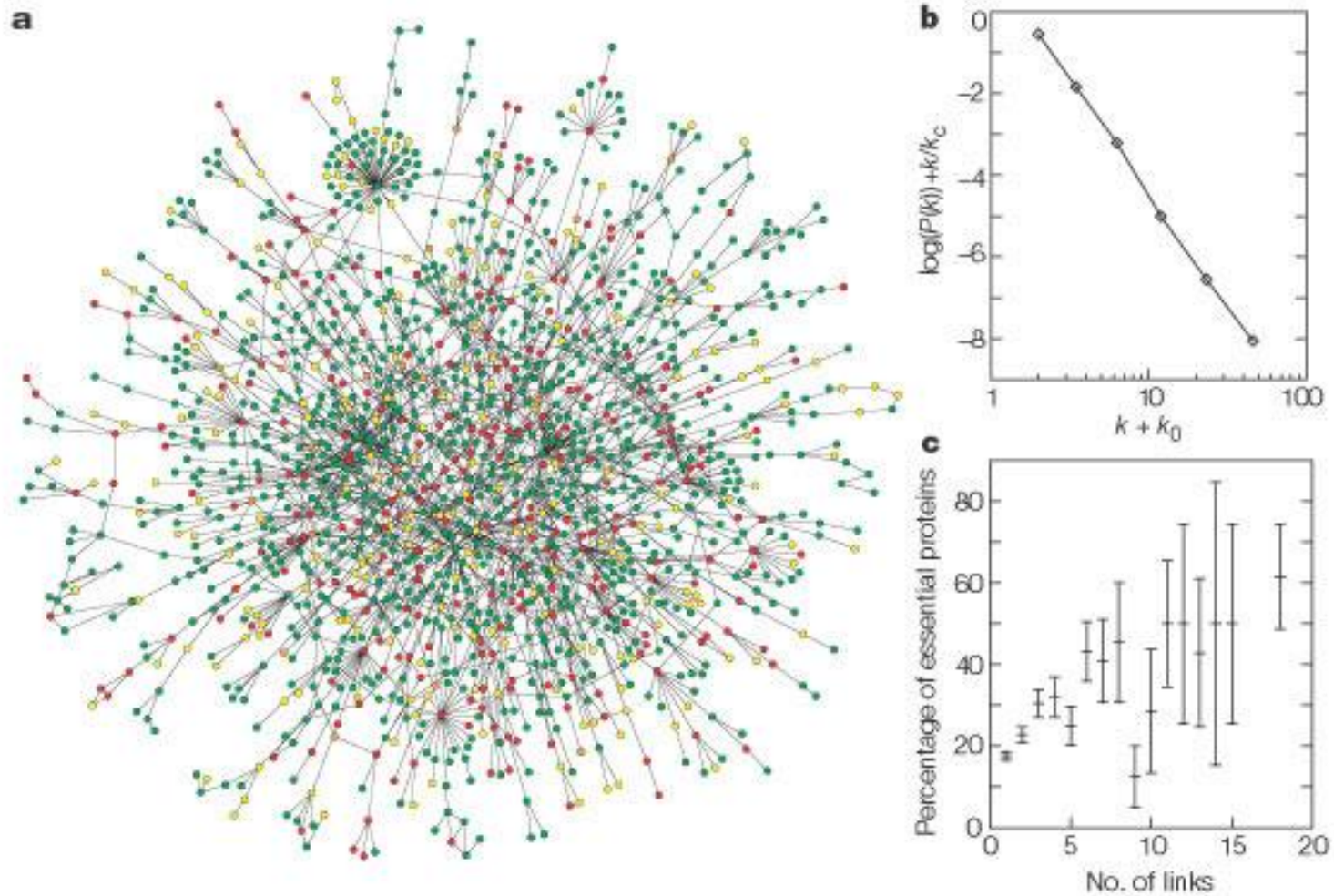
Albert, Jeong, Barabási,
Nature **406** (2000) 378

brief communications

Lethality and centrality in protein networks

The most highly connected proteins in the cell are the most important for its survival.

Jeong, Mason, Barabási, Oltvai, *Nature* **411** (2001) 41



largest cluster of the yeast proteome (at 2001)

→ "PPI networks apparently are scale-free…"

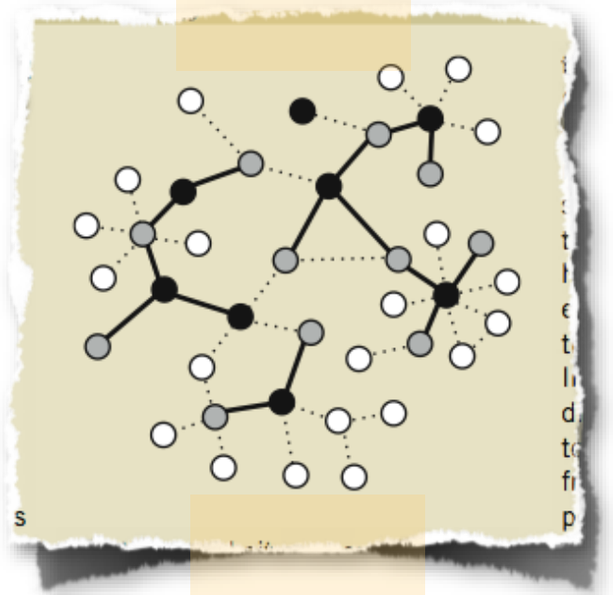
"**Are**" they scale-free
or
"Do they **look** like" scale-free???

Effect of sampling on topology predictions of protein-protein interaction networks

Jing-Dong J Han[1–3], Denis Dupuy[1,3], Nicolas Bertin[1], Michael E Cusick[1] & Marc Vidal[1]

Nature Biotech **23** (2005) 839

Generate networks of various types,
sample sparsely from them
→ determine degree distribution



- Random (ER / Erdös-Renyi) → $P(k)$ = Poisson
- Exponential (EX) → $P(k) \sim \exp[-k]$
- scale-free / power-law (PL) → $P(k) \sim k^{-\gamma}$
- $P(k)$ = truncated normal distribution (TN)

Partial Sampling

Estimated for yeast: 6000 proteins, 30000 interactions

Table 1 Topological properties of interactome maps

Data set	Ito *et al.* (yeast)	Uetz *et al.* (yeast)	Ito-Uetz combined	Li *et al.* (worm)	Giot *et al.* (fly)	Minimum value	Maximum value
Total number of nodes	797	1,005	1,417	1,415	4,651	797	4,651
Nodes in main component	417 (52%)	473 (47%)	970 (68%)	1,260 (89%)	3,039 (65%)	47%	89%
Total number of interactions	806	948	1,520	2,135	4,787	806	4,787
Interactions in main component	544	558	1,229	2,038	3,715	544	3,715
R-square	0.843	0.954	0.899	0.885	0.91	0.843	0.954
γ	−1.82	−2.42	−1.91	−1.59	−2.75	−2.75	−1.59
$\langle k \rangle$	1.96	1.84	2.15	2.98	2.04	1.84	2.98
Average clustering coefficient	0.2	0.11	0.09	0.09	0.06	0.06	0.2
Number of network components	143	177	160	70	591	70	591
Average component size	5.6	5.7	8.9	20.2	7.9	5.6	20.2
Characteristic path length	6.14	7.48	6.55	4.91	9.43	4.91	9.43
Number of baits	455	512	827	502	2,820	455	2,820

The linear regression R-square measures the linearity between log(n(k)) and log(k) i.e. the fit to a power-law distribution. γ is the exponent of the power law distribution formula that best fits the observed distribution. $\langle k \rangle$ is the average number of interactions per protein observed in the network. For the Ito, Li and Giot data sets only the high confidence interactions were considered (core).

Y2H experiments **detected** only **3…9%** of the complete interactome!

Han et al, *Nature Biotech* **23** (2005) 839

R square

Given: a data set with n values $y_1,...,y_n$ and
a set of fitted / predicted / modelled values $f_1,...,f_n$ e.g. from linear regression.

We call their difference **residuals** $e_i = y_i - f_i$

and the mean value $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

The total sum of squares (proportional to the variance of the data) is:

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$

The sum of squares of residuals is:

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

The **coefficient of determination**, R^2 or r^2 is often defined as:

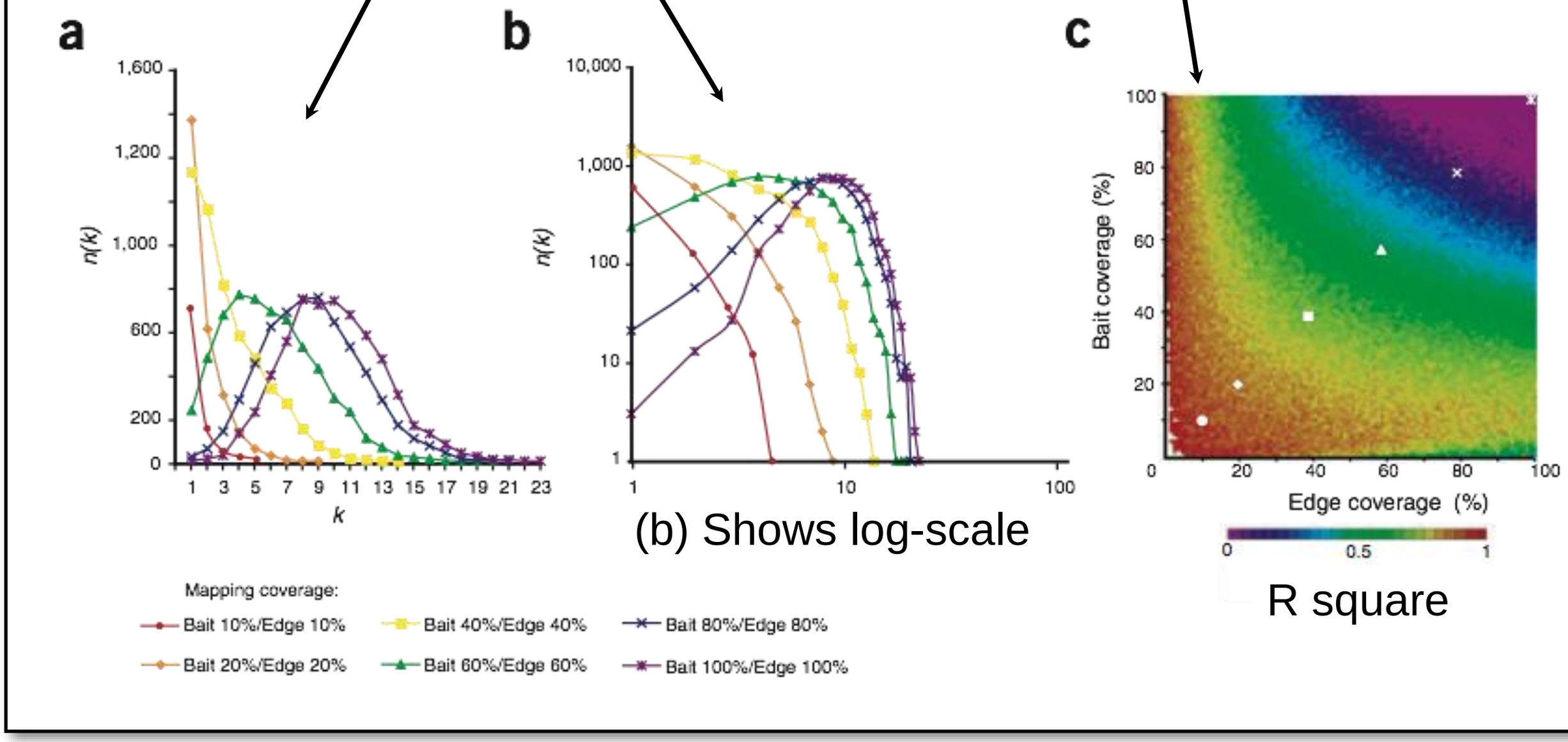
$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}.$$

www.wikipedia.org

Sparsely Sampled random (ER) Network

resulting *P(k)* for different coverages

(c) Shows linearity (R square) between detected *P(k)* and ideal power law; good agreement (red; R ≈1 for low edge coverage)



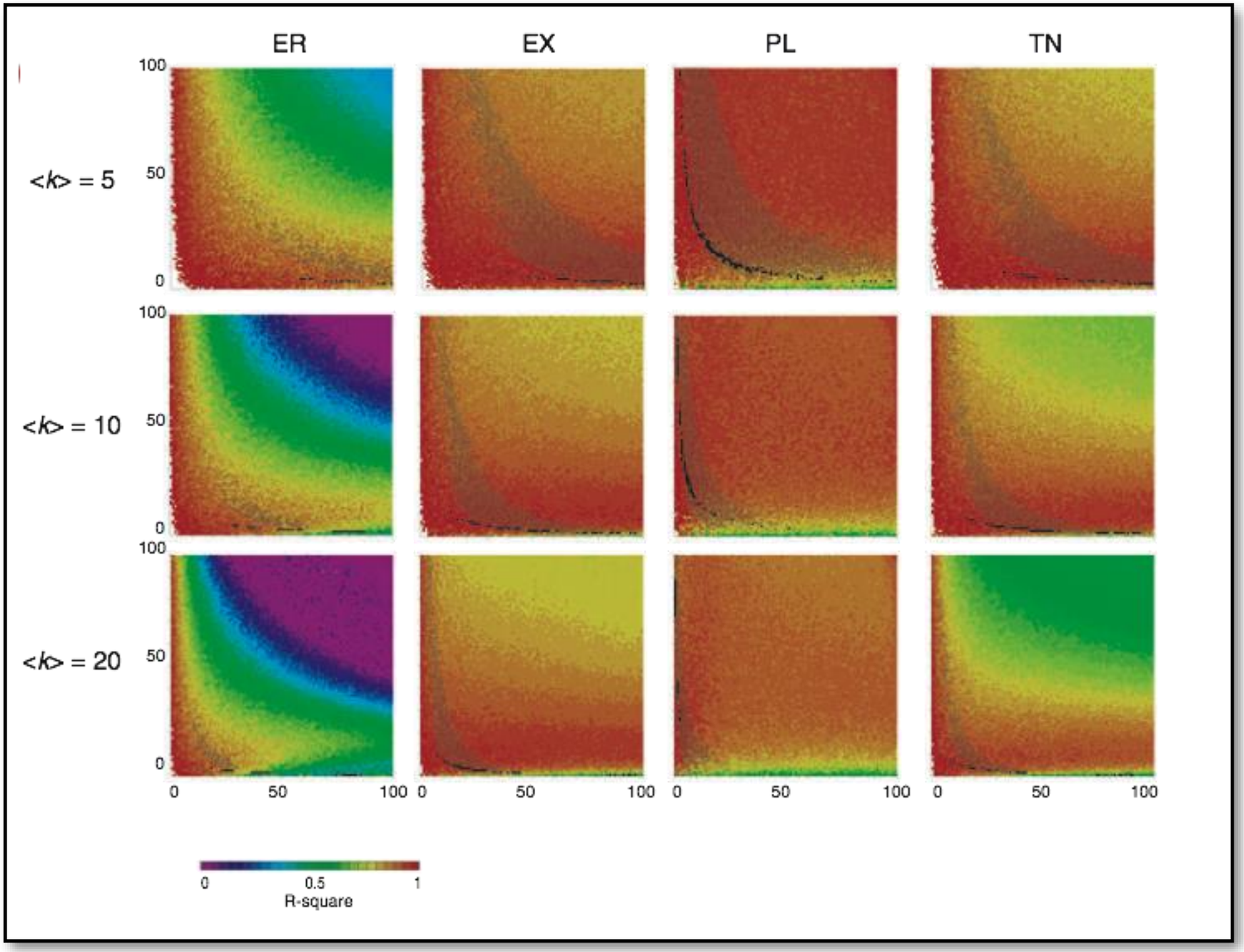
(b) Shows log-scale

→ for **sparse** sampling (10-20%), even an ER network

"**looks**" **scale-free** (when only *P(k)* is considered)

Han et al, *Nature Biotech* **23** (2005) 839

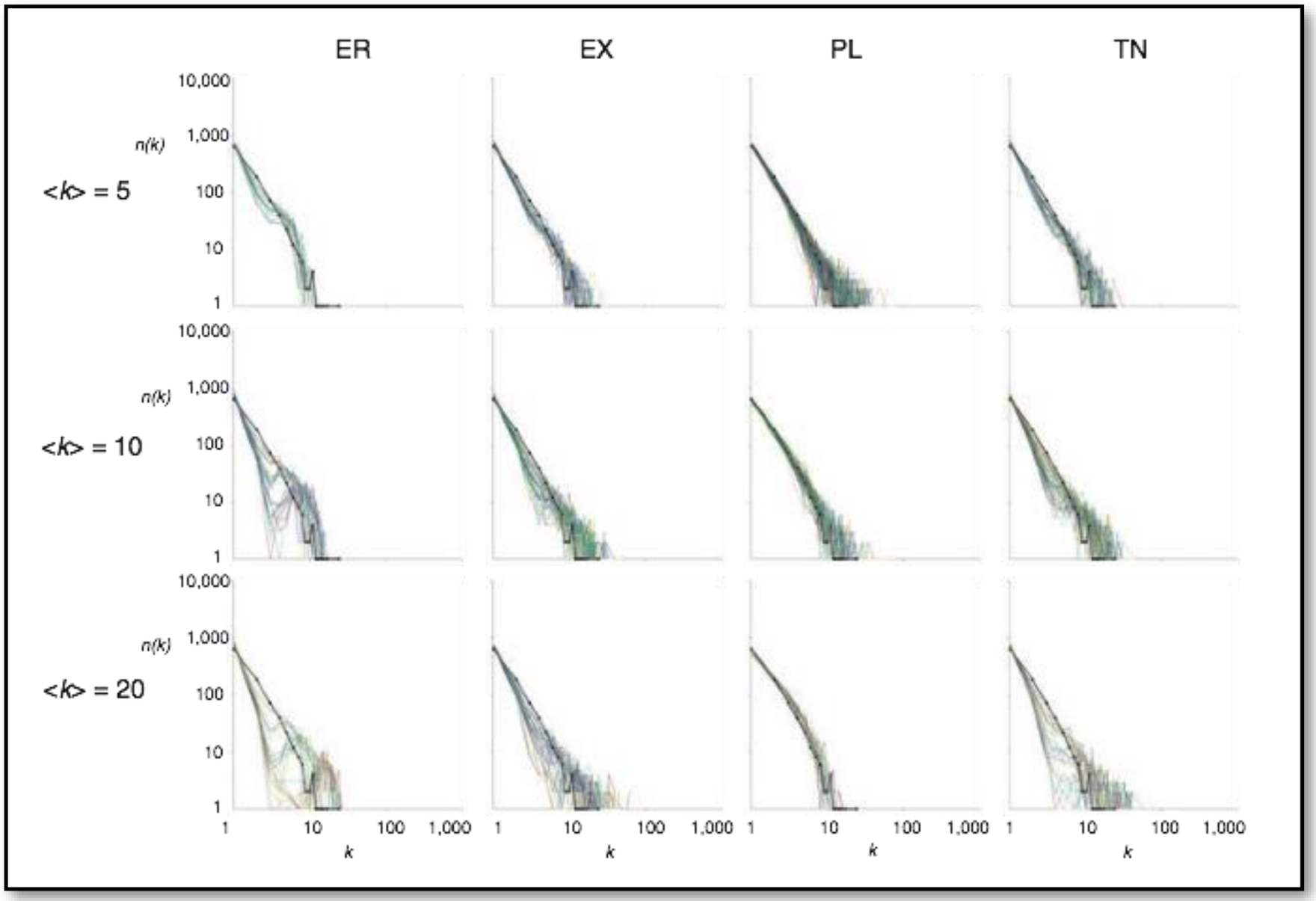
Anything Goes – different topologies



All network topologies look scale-free (red) when undersampled

Han et al, *Nature Biotech* **23** (2005) 839

Compare to Uetz et al. data

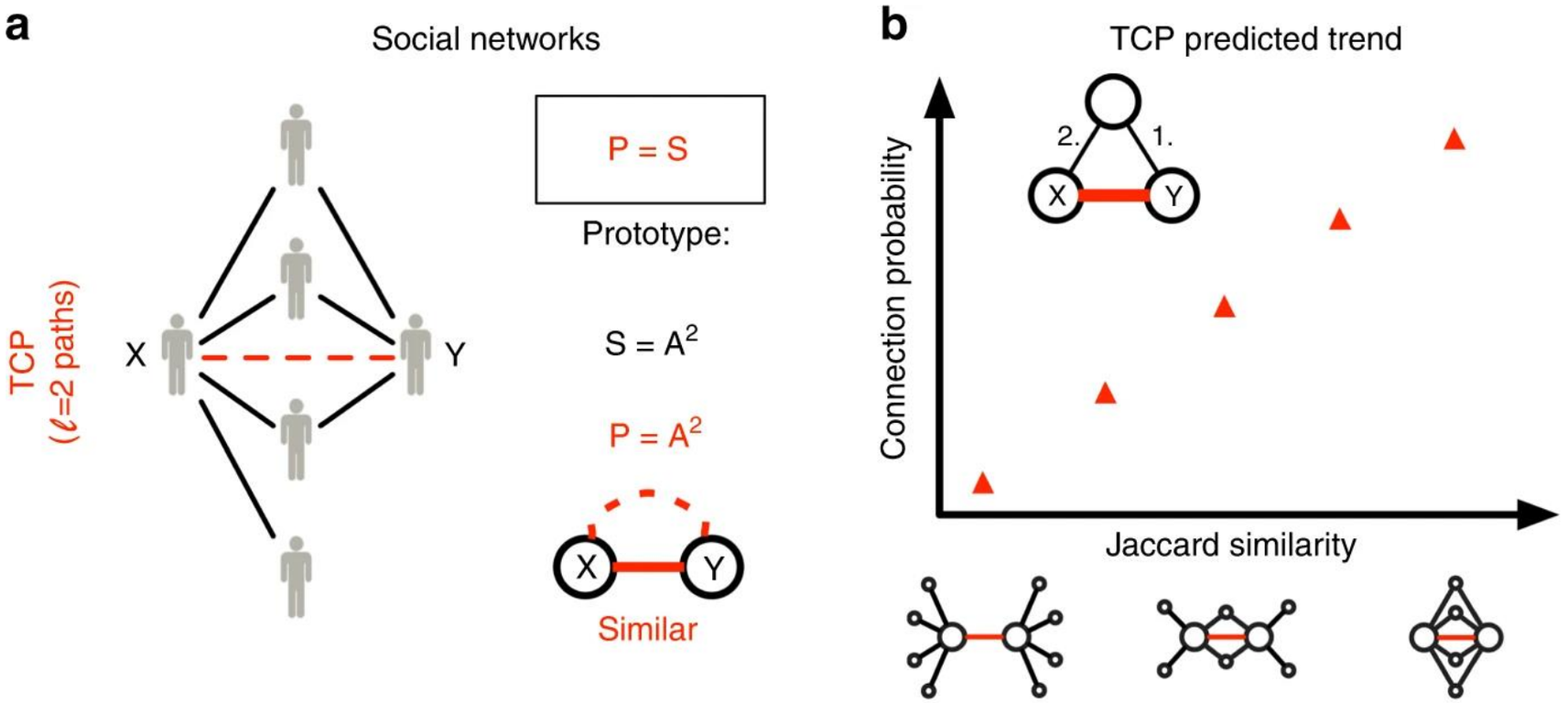


Uetz et al. data (solid line) is compared to sampled networks of similar size.

Sampling density affects observed degree distribution
→ true underlying network cannot be identified from available data

Han et al, *Nature Biotech* **23** (2005) 839

Link prediction based on PPI network

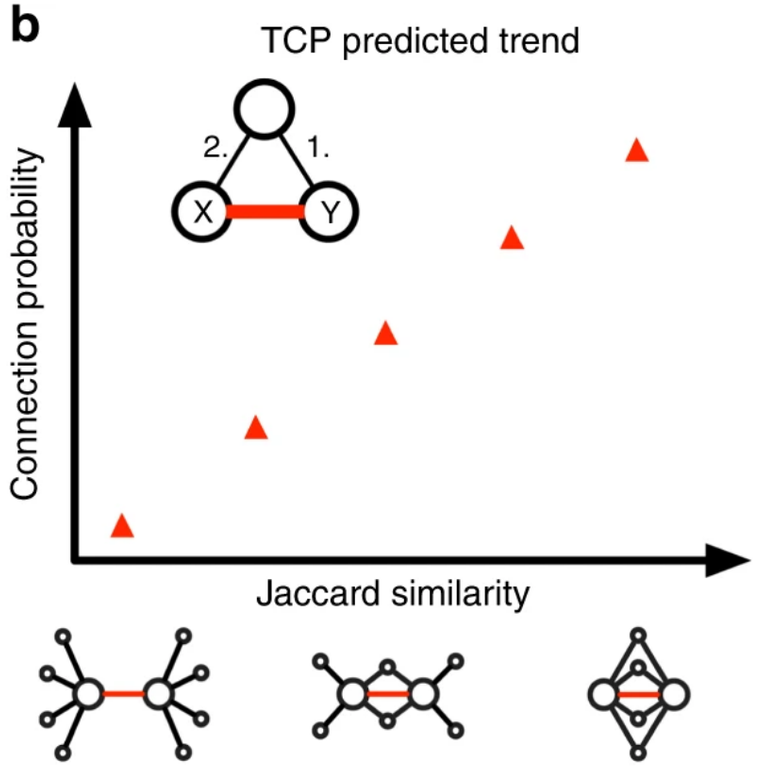


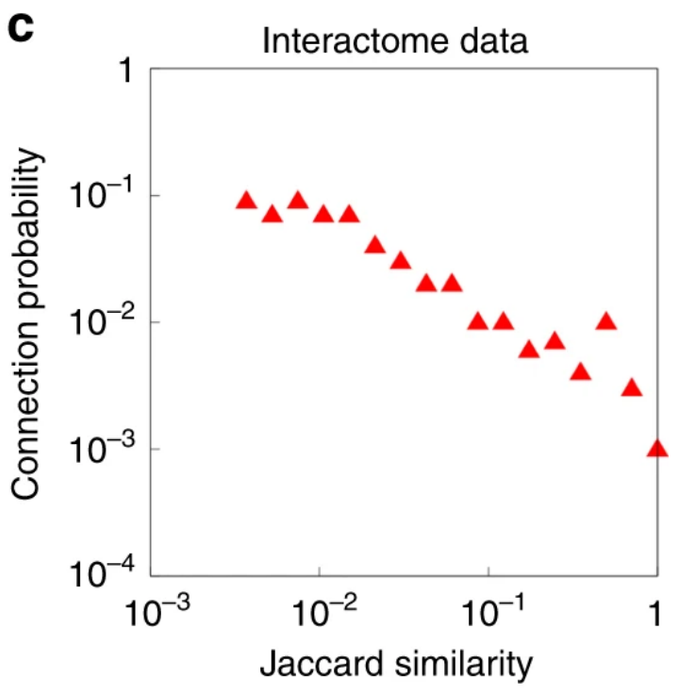
Kovács, ... Vidal & Barabási
Nature Commun. 10, 1240 (2019)

(a) In social networks, a large number of common friends implies a higher chance to become friends (red link between nodes X and Y), known as the **Triadic Closure Principle (TCP)**.
TCP predicts (*P*) links based on node similarity (*S*), quantifying the **number of shared neighbors** between each node pair (A^2).

(b) A basic mathematical formulation of TCP would imply that protein pairs of high Jaccard similarity are more likely to interact

TCP does not apply to PPI networks





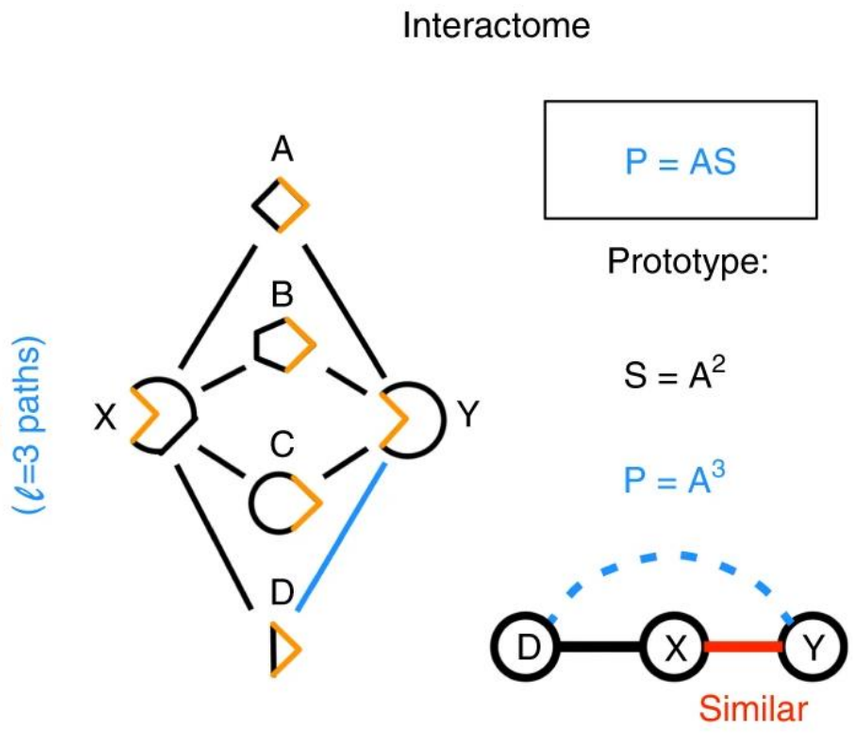
Kovács, ... Vidal & Barabási
Nature Commun. 10, 1240 (2019)

$J = |N_X \cap N_Y| / |N_X \cup N_Y|$, where N_X and N_Y are the interaction partners of X and Y.

However, Kovács and co-workers did not observe the expected trend in Protein-Protein Interaction (PPI) datasets, as illustrated here for a binary human PPI network (HI-II-14): high Jaccard similarity indicates a lower chance for the proteins to interact.

The data are binned logarithmically based on the Jaccard similarity values.

PPIs involve binding interfaces



PPIs often require complementary interfaces (see V8).
Hence, two proteins, X and Y, with similar interfaces share many of their neighbors.

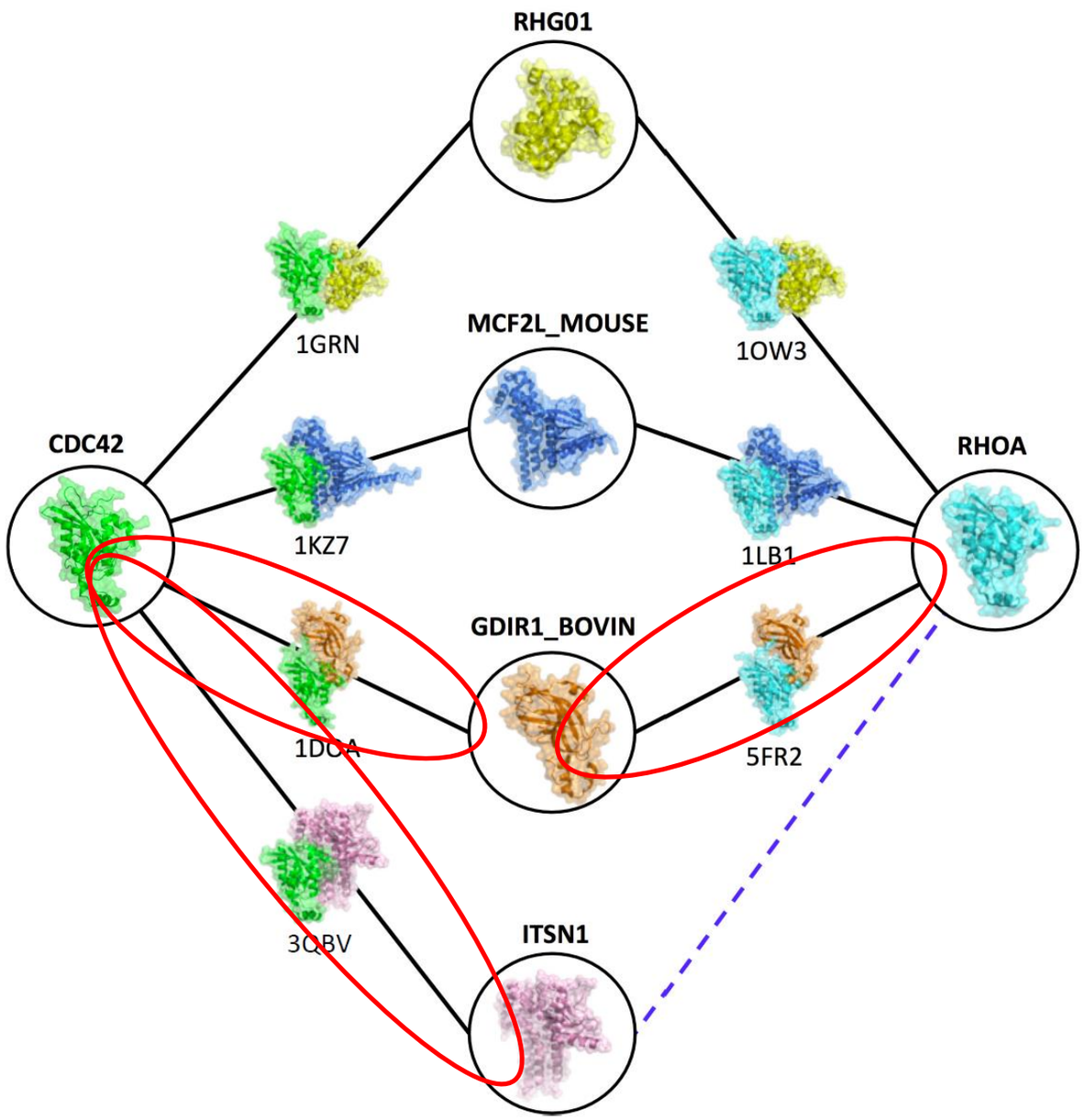
Yet, a shared interface does not typically guarantee that X and Y directly interact with each other.

Instead, an additional interaction partner of X (protein D) might be also shared with protein Y (**blue link**).

Such a link can be predicted by using paths of length 3 (L3). L3 identifies similar nodes to the known partners ($P = AS$), going one step beyond the similarity-based argument of TCP.

Kovács, ... Vidal & Barabási
Nature Commun. 10, 1240 (2019)

Structural illustration of L3 principle



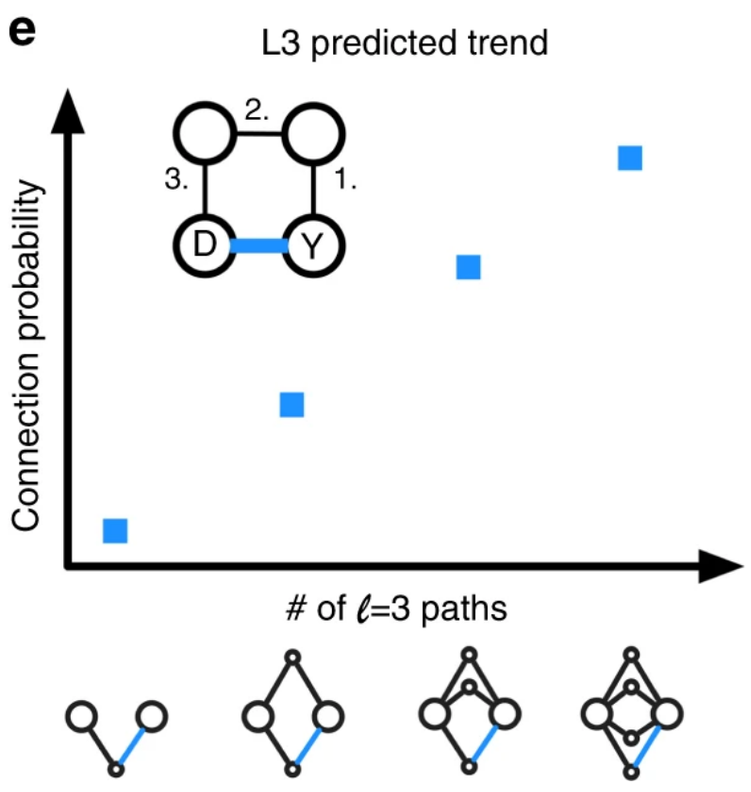
Kovács, ... Vidal & Barabási
Nature Commun. 10, 1240 (2019)

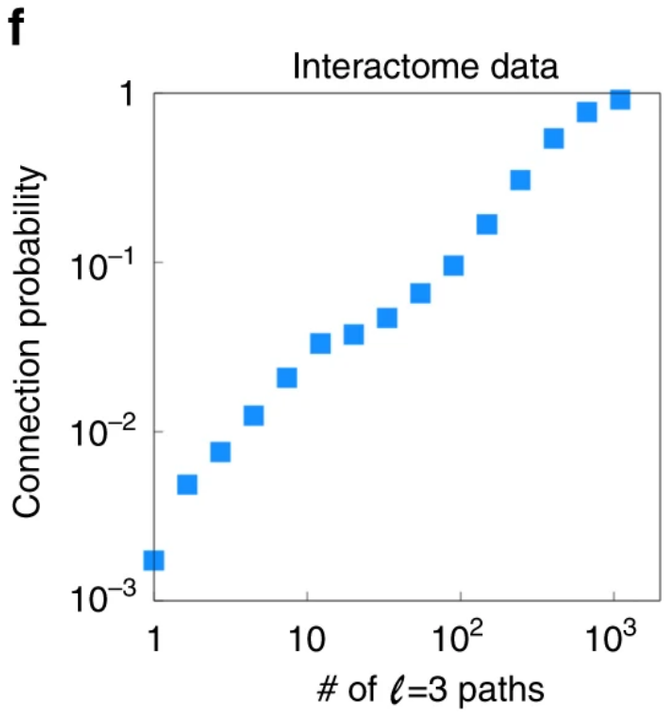
We will illustrate this link prediction principle with existing 3D structural data on two human proteins from PDB, CDC42 and RHOA that interact with some of their partners through the same shared interface.

CDC42 and RHOA are not known to interact with each other. But we expect them to share some additional interaction partners, interacting with the same shared interface.

From a network perspective, the structurally inferred (**blue**) interaction between ITSN1 and RHOA connects nodes that are linked by a larger number of paths of length l = 3.

L3 applies to PPI networks





e Even without using any structural information, two proteins, such as Y and D are expected to interact if they are linked by multiple $\ell = 3$ paths in the network (L3).

f A strong positive trend in HI-II-14 is observed between the probability of two proteins interacting and the number of $\ell=3$ paths between them, supporting the validity of the L3 principle

Kovács, ... Vidal & Barabási
Nature Commun. 10, 1240 (2019)

Apply degree normalization

High-degree nodes (hubs) induce multiple, unspecific shortcuts in the network, resulting in biased predictions that can only be avoided by proper degree normalization.

Such degree normalization is particularly important for L3, as it needs to choose candidates from nodes at l = 3 steps, an exponentially larger pool than the l = 2 distance pool utilized by TCP.

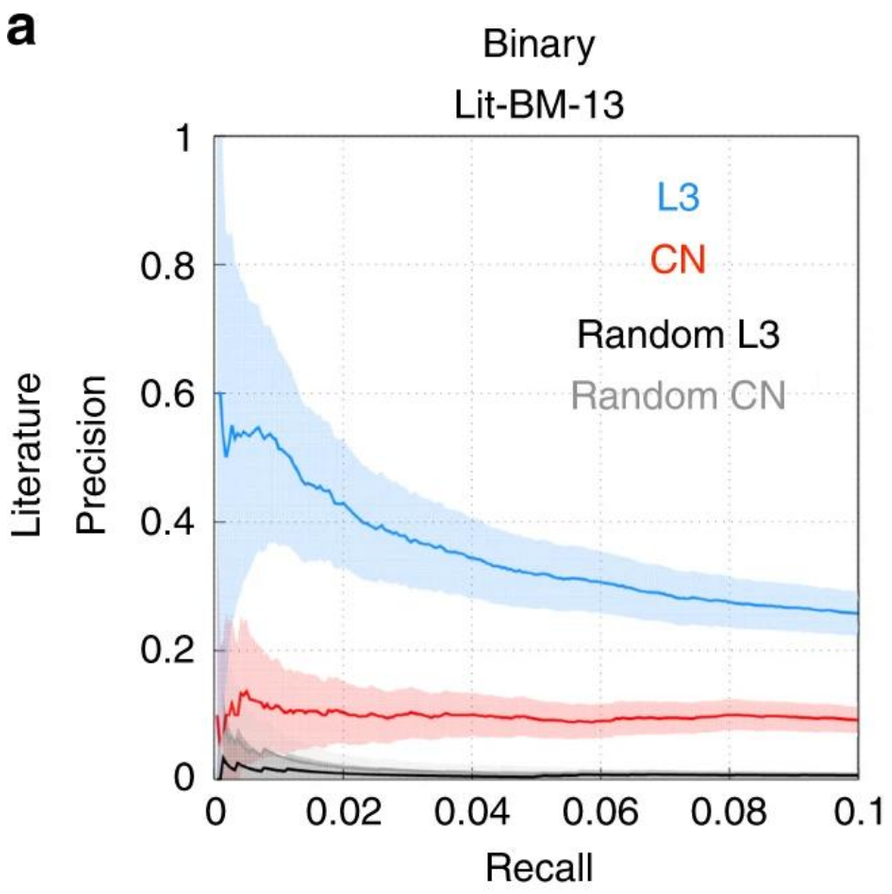
To eliminate potential degree biases caused by intermediate hubs, we assign a degree-normalized L3 score to each node pair, X and Y

$$p_{XY} = \sum_{U,V} \frac{a_{XU} a_{UV} a_{VY}}{\sqrt{k_U k_V}}$$

where k_U is the degree of node U and $a_{XU} = 1$ if proteins X and U interact, and zero otherwise.

Kovács, ... Vidal & Barabási
Nature Commun. 10, 1240 (2019)

Cross-validation



We randomly select 50% of the PPIs and use it as the input network to predict the rest of the PPIs.

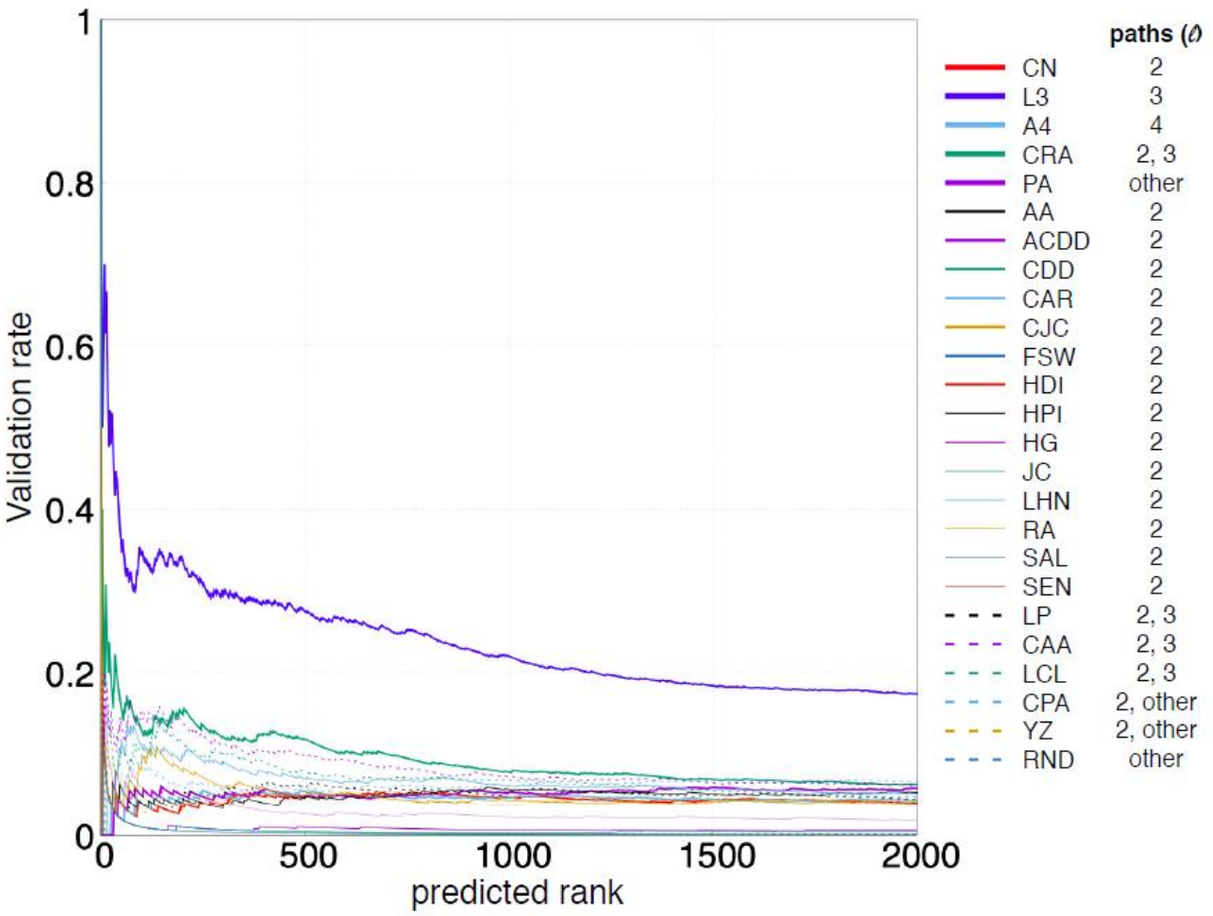
L3 outperforms Common Neighbors (CN) on PPI networks. Monte Carlo cross-validation of CN (a TCP implementation).

Precision: fraction of interacting proteins vs. all predicted pairs.

Recall : fraction of predicted PPIs compared to the number of test PPIs.

Kovács, ... Vidal & Barabási
Nature Commun. 10, 1240 (2019)

High-throughput validation



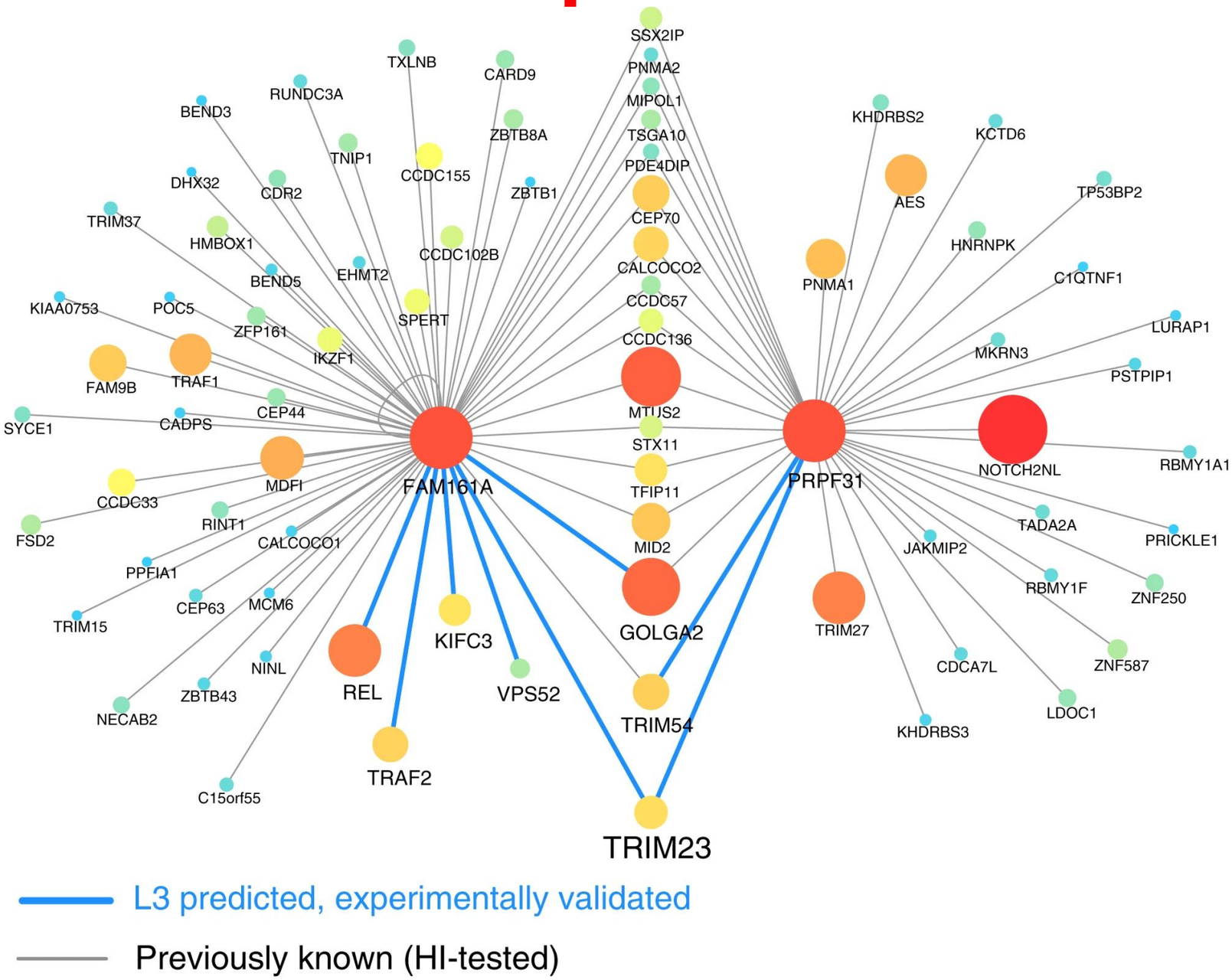
Top 500 predicted interactions were tested by Y2H method (positives and negative combinations).

-> High validation rate

-> L3 method outperforms all other link prediction methods (such as PrePPI) at least 2-fold.

Kovács, ... Vidal & Barabási
Nature Commun. 10, 1240 (2019)

L3 predicted interaction

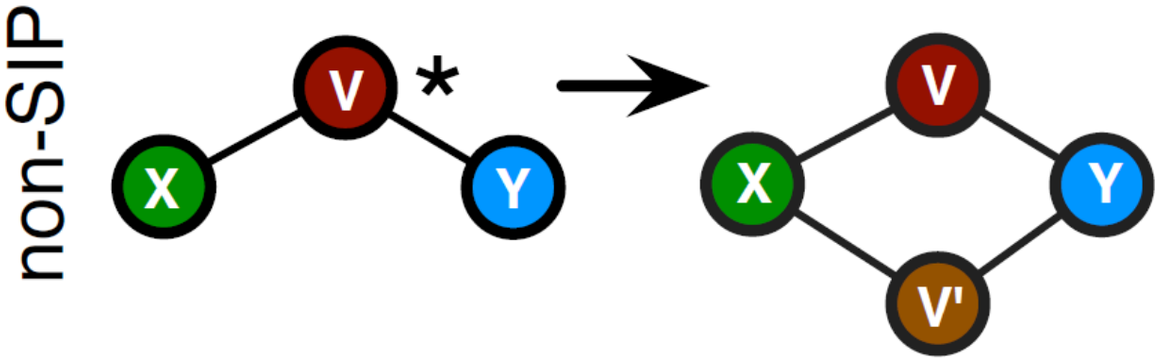


For 2 proteins involved in retinitis pigmentosa, FAM161A and PRPF31, we show all known interacting partners (gray), together with those predicted by the L3 algorithm and confirmed by pairwise tests (blue).

The top L3 predicted interaction is connecting FAM161A to GOLGA2, two proteins without any shared interaction partners. The node size and color illustrates the degree of the proteins in HI-tested.

Kovács, ... Vidal & Barabási
Nature Commun. 10, 1240 (2019)

Link to evolution



A key evolutionary mechanism responsible for the emergence of novel proteins is **gene duplication** (see V7).

If protein V duplicates, the duplicated node (V') will (at least initially) retain the links of the original protein.

This may partly explain the success of L3.

Kovács, ... Vidal & Barabási
Nature Commun. 10, 1240 (2019)

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!

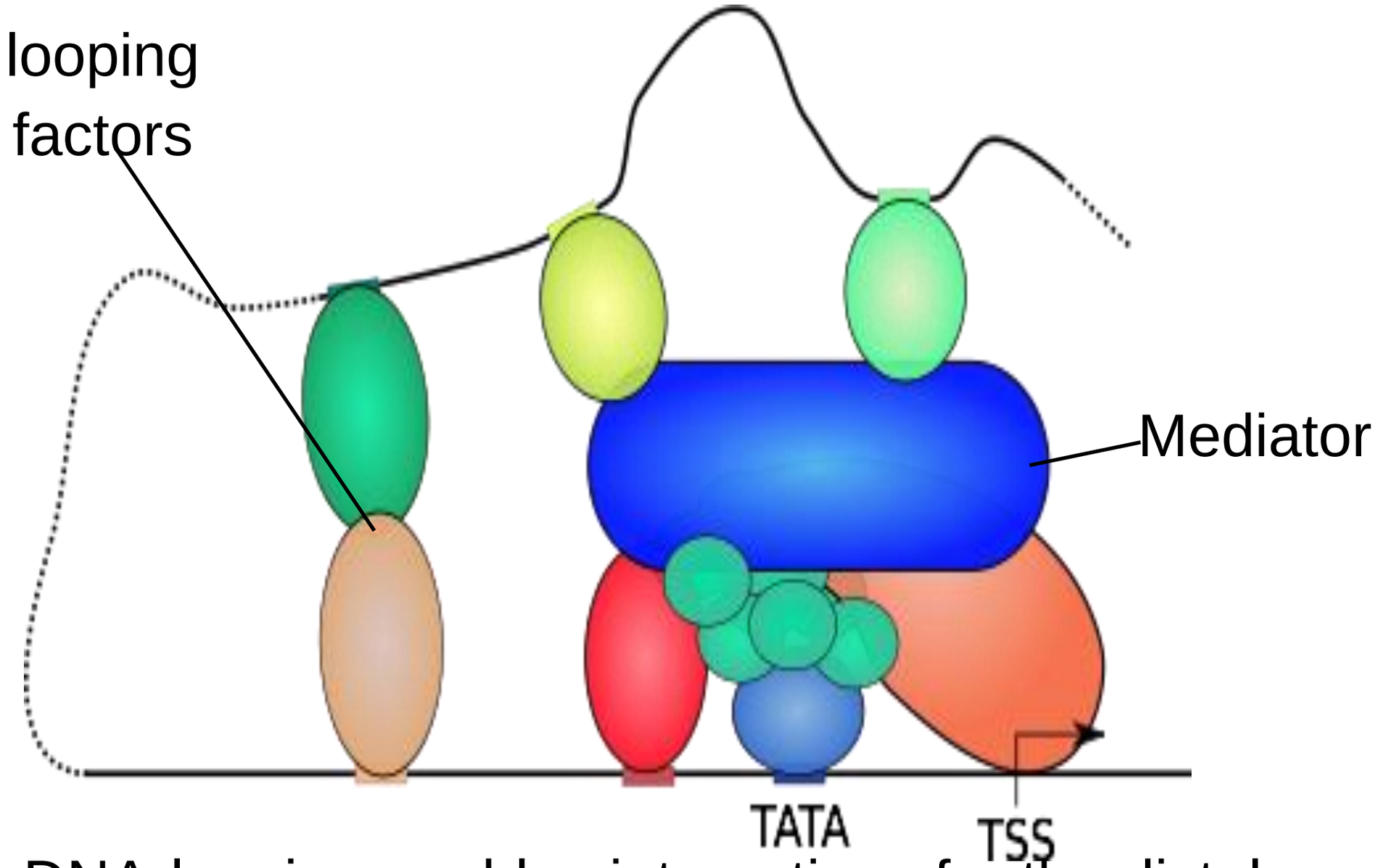
- L3 principle suitable for **link prediction**

Next lecture:

- graph bisection (-> communities)
- graph modularity
- network growth
- functional annotation in the network

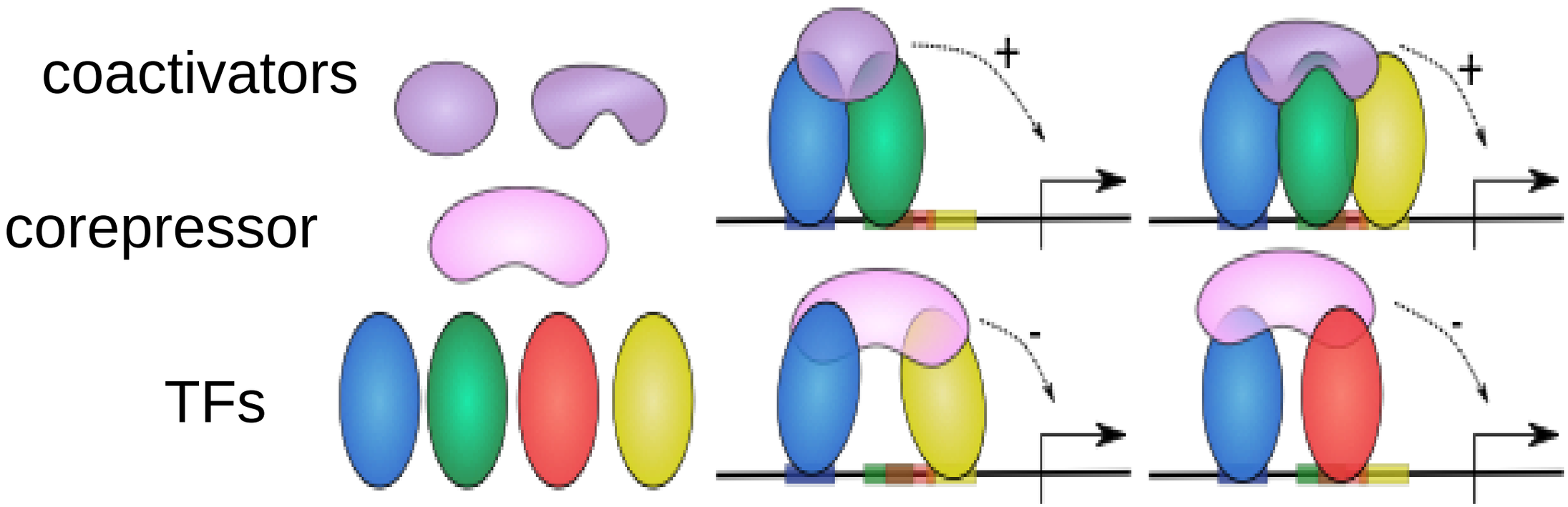
Additional slides (not used)

Transcriptional activation



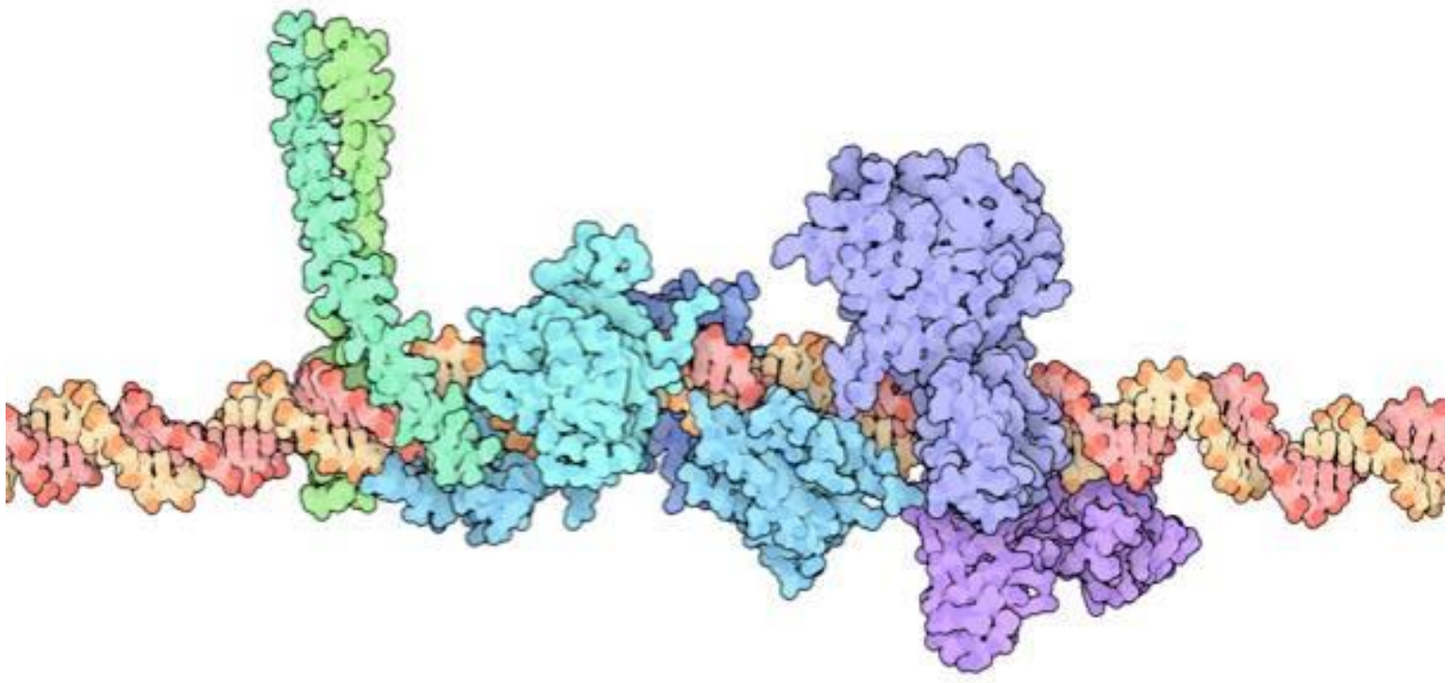
DNA-looping enables interactions for the distal promotor regions,
Mediator cofactor-complex serves as a huge linker

cis-regulatory modules



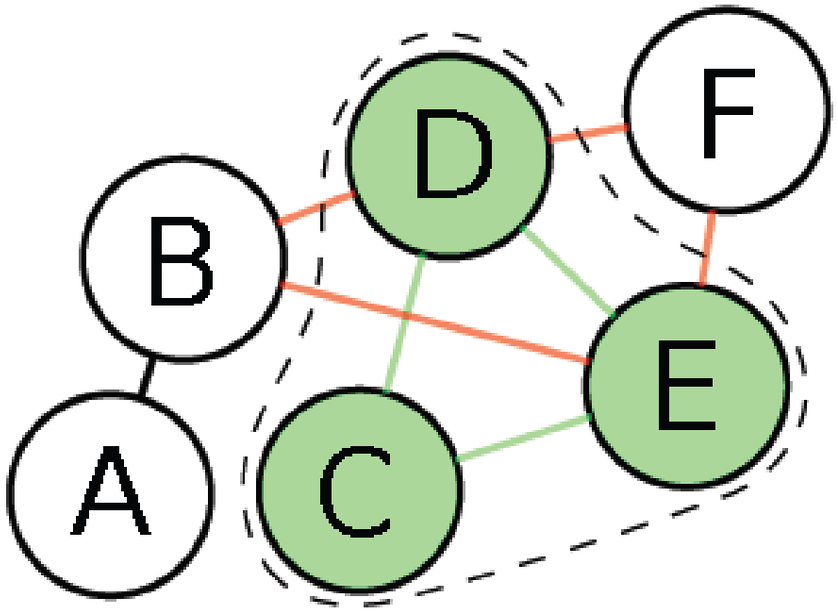
TFs are not dedicated activators or respressors!

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

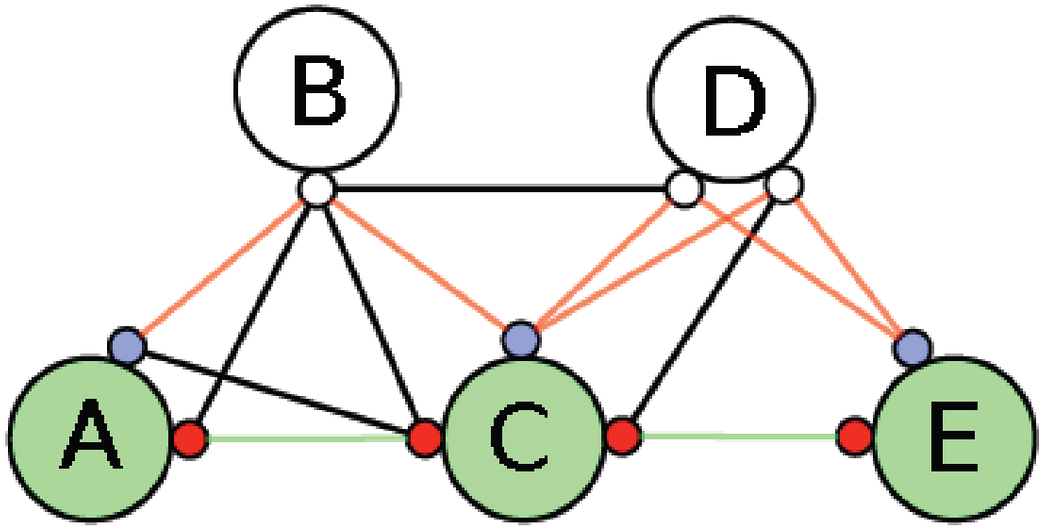
in protein-protein interaction graph

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.

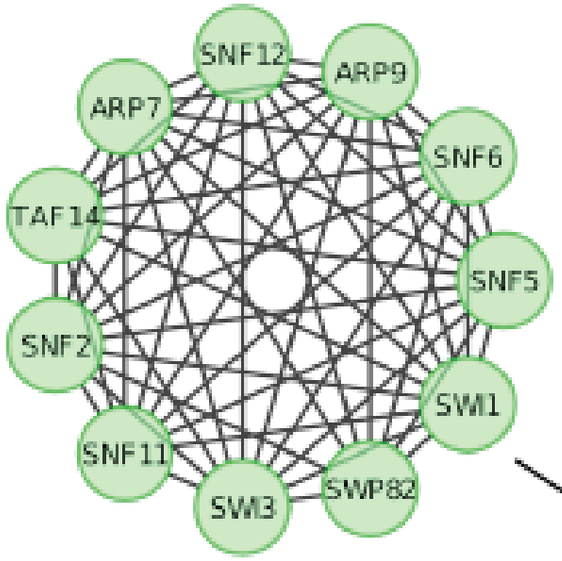


Green proteins A, C, E form actual complex.

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



weighted protein-protein interaction data, such as

PrePPI

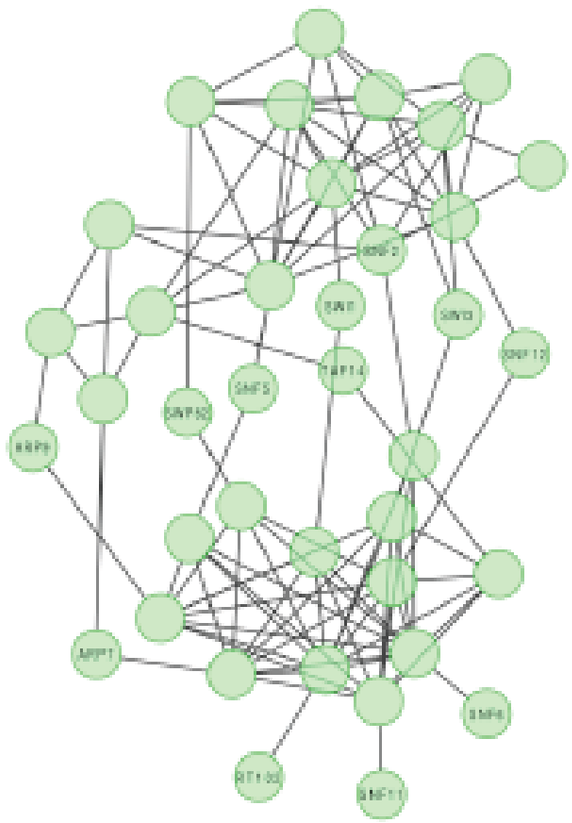
SIF format with UniProt IDs

seed proteins,
threshold for pairs,
max. depth of search

domain-aware cohesiveness optimization algorithm

transcription factor complex candidates

UniProt EMBL-EBI



domain-domain interaction network

data retrieval and buildup automatically

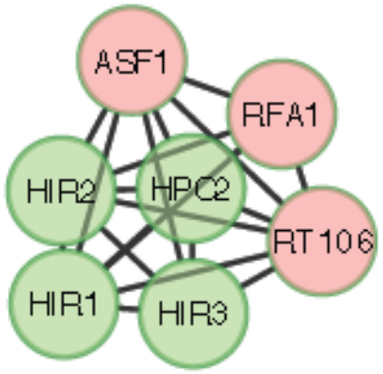
Pfam InterPro DOMINE IDDI

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

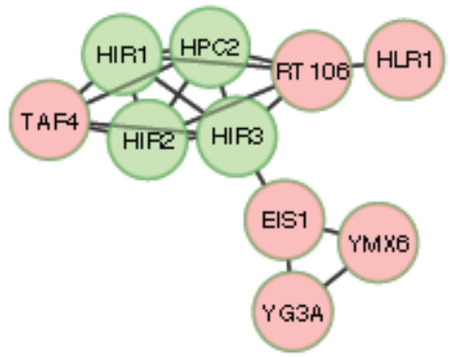
Daco identifies far more TF complexes than other methods

	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

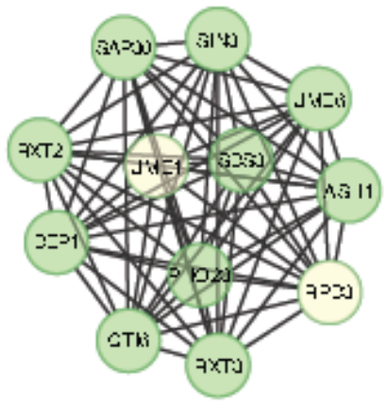
Examples of TF complexes – comparison with ClusterONE



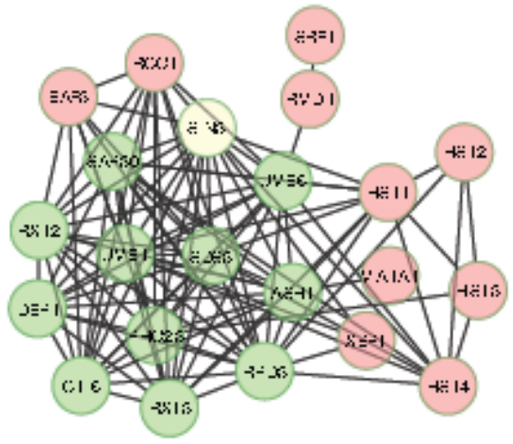
(a) HIR(SGD) / DACO



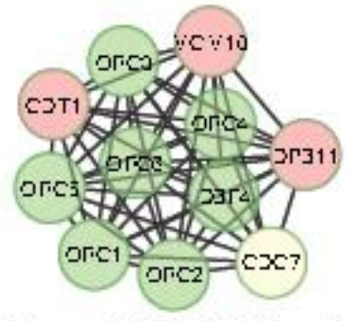
(b) HIR(SGD) / ClusterONE



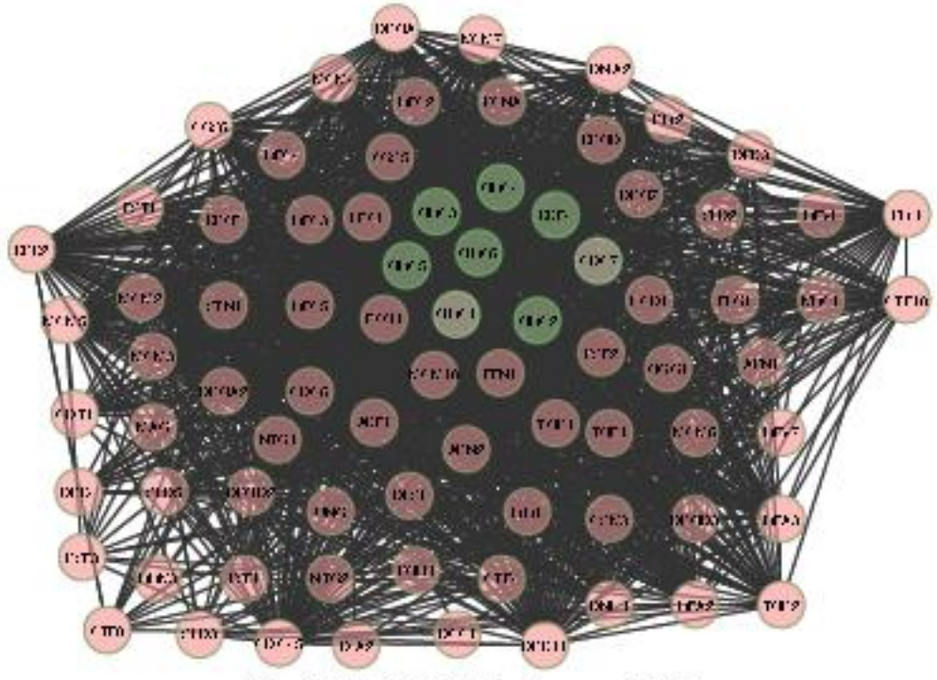
(c) RPD3L(CYC2008) / DACO



(d) RPD3L(CYC2008) / ClusterONE



(e) ORC(MIPS) / DACO

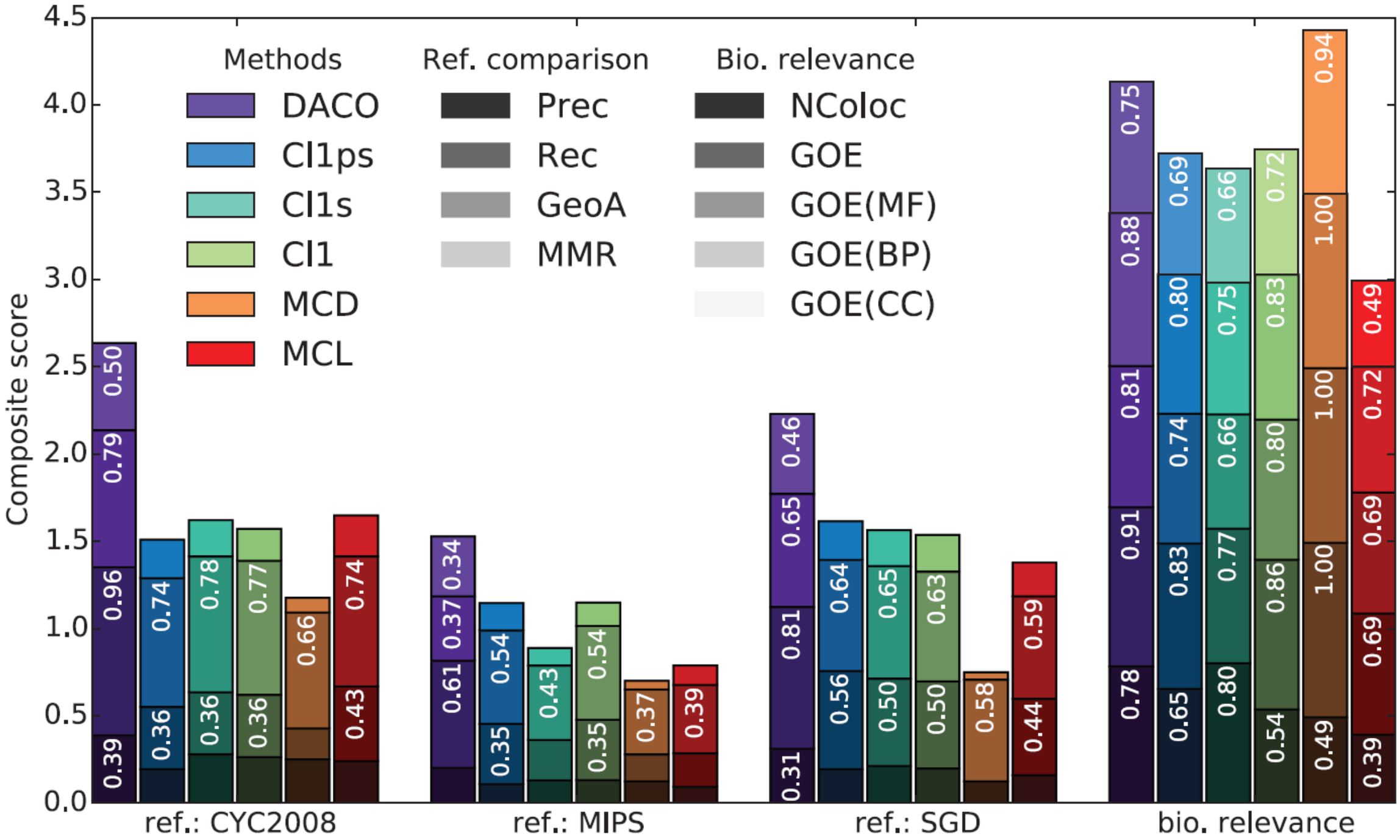


(f) ORC(MIPS) / ClusterONE

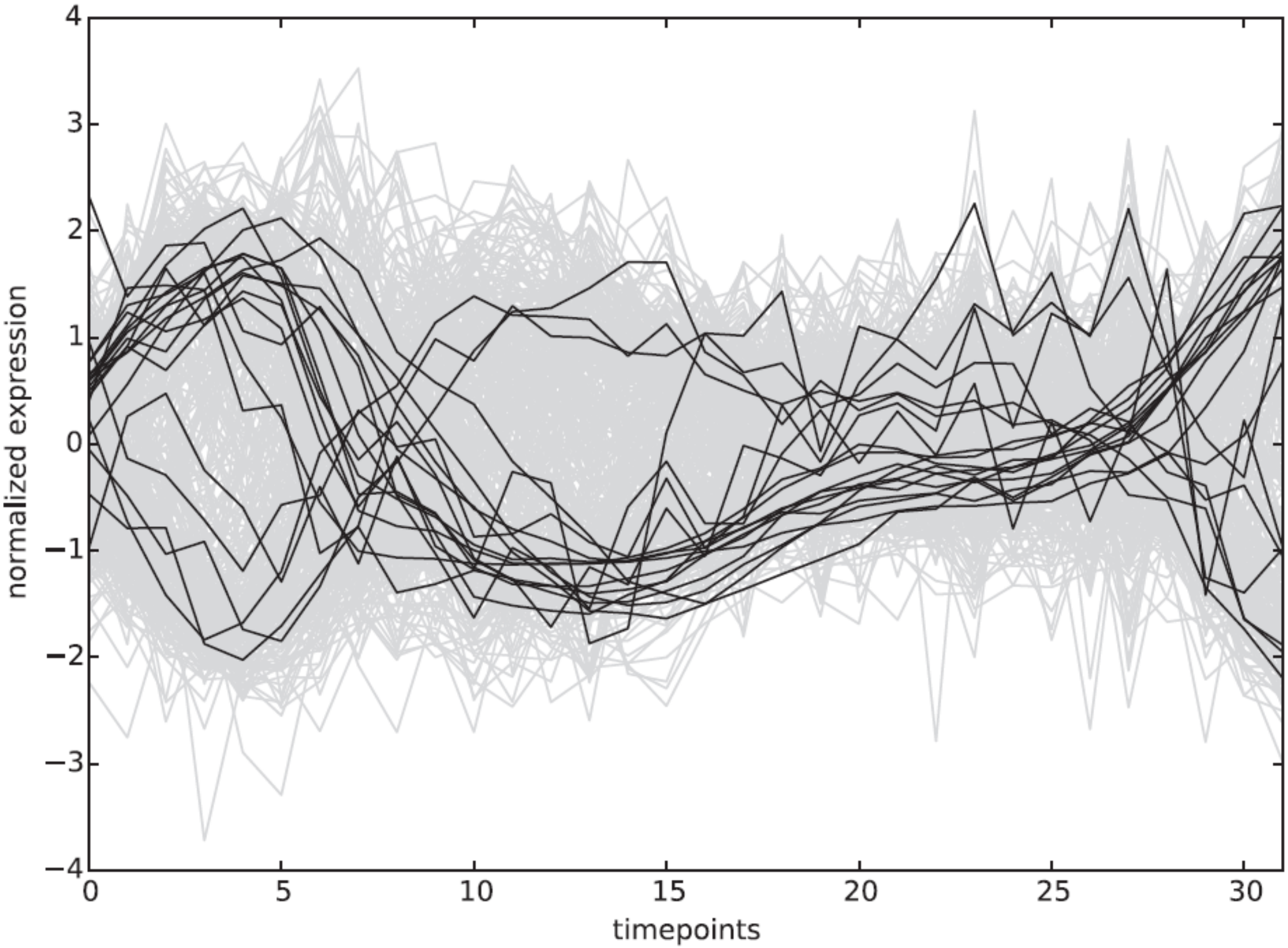
Green nodes: proteins in the reference that were matched by the prediction

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).