V7 – Biological PPI Networks

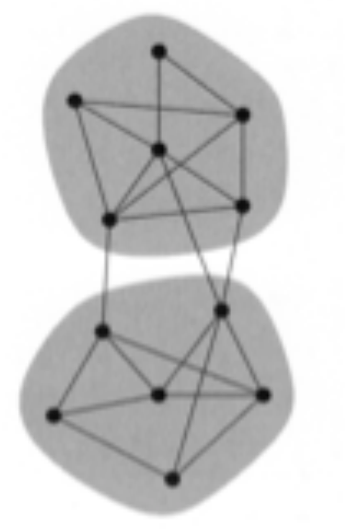
- graph bisection (-> communities)
- graph modularity
- network growth
- functional annotation in the network

Thu, Nov 7, 2019

Graph bisection

The simplest graph partitioning problem is the division of a network into just 2 parts. This is called **graph bisection**.

If we can divide a network into 2 parts, we can also divide it further by dividing one or both of these parts …



graph bisection problem: divide the vertices of a network into 2 non-overlapping groups of given sizes such that the **number of edges** running **between** vertices in **different groups is minimized**.

The number of edges between groups is called the **cut size**.

In principle, one could simply look through all possible divisions of the network into 2 parts and choose the one with smallest cut size.

Algorithms for graph partitioning

But this exhaustive search is prohibitively expensive!

Given a network of *n* vertices. There are $\frac{n!}{n_1!n_2!}$ different ways of dividing it into 2 groups of n_1 and n_2 vertices.

The amount of time to look through all these divisions will go up roughly exponentially with the size of the system.

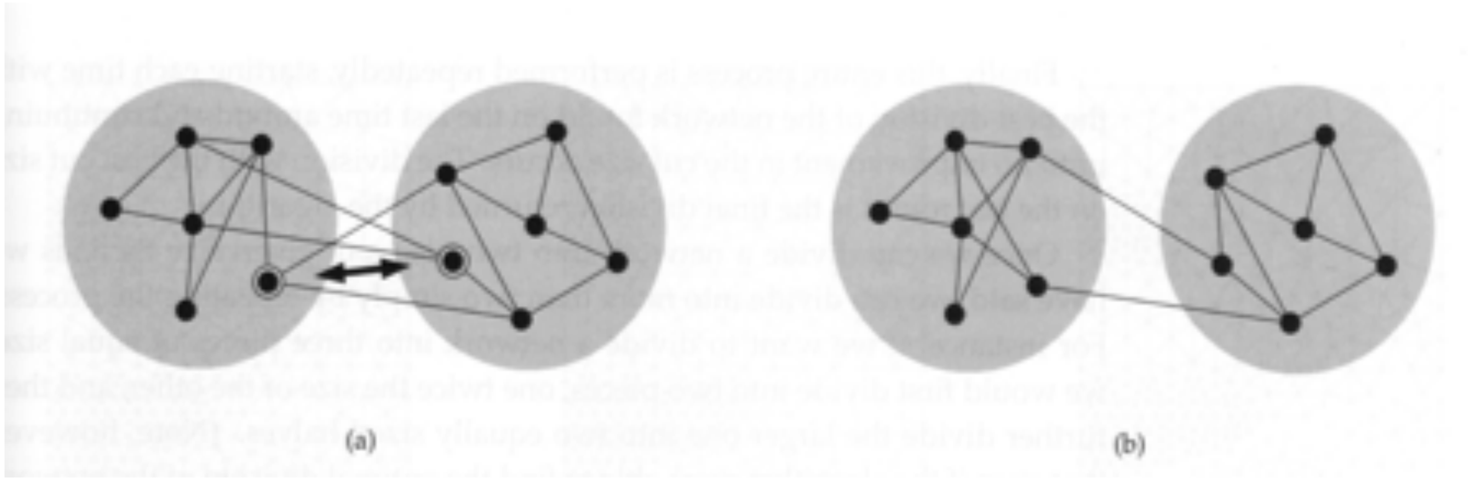
Only values of up to $n = 30$ are feasible with today's computers.

In computer science, one often encounters the following situation:
either an algorithm can be clever and run quickly, but will fail to provide the optimal answer in some (or perhaps in many) cases, or it will always find the optimal answer, but takes an impractical length of time to do so.

The Kernighan-Lin algorithm

This algorithm proposed by Brian Kernighan and Shen Lin in 1970 is one of the simplest and best known **heuristic** algorithms for the graph bisection problem.

(Kernighan is also one of the developers of the C language).



(a) The algorithm starts with any division of the vertices of a network into two groups (shaded) and then searches for pairs of vertices, such as the pair highlighted here, whose **interchange** would **reduce** the **cut size** between the groups.

(b) The same network after interchange of the 2 vertices.

The Kernighan-Lin algorithm

(1) **Divide** the vertices of a given network into 2 groups (e.g. randomly).

(2) For each pair *(i,j)* of vertices, where *i* belongs to the first group and *j* to the second group, calculate **how much** the **cut size** between the groups would **change** if *i* and *j* were interchanged between the groups.

(3) Find the pair that **reduces the cut size** by the largest amount and **swap the vertices**.

If no pair reduces it, find the pair that increases it by the smallest amount.

Repeat this process, but with the important restriction that **each vertex** in the network can **only** be **moved once**.

Stop when there is no pair of vertices left that can be swapped.

The Kernighan-Lin algorithm (II)

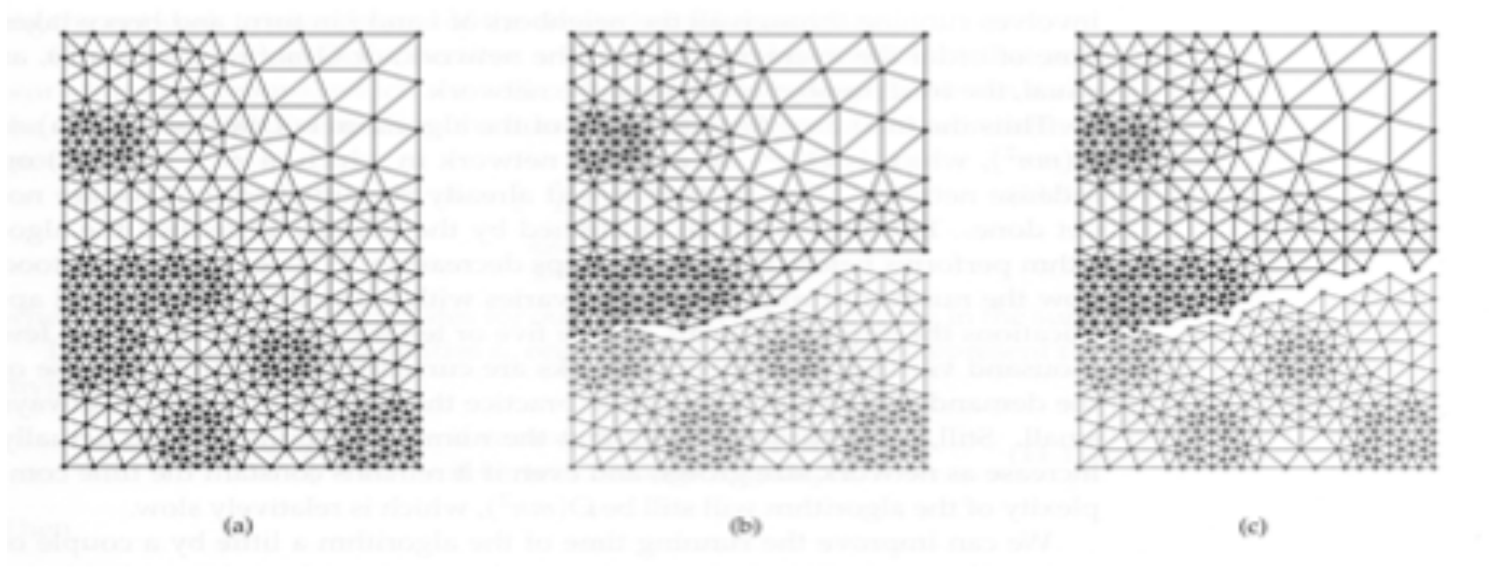
(3) **Go back** through every state that the network passed through during the swapping procedure and choose among them the state in which the **cut size** takes its **smallest value**.

(4) Perform the steps (2) – (4) repeatedly, starting each **iteration** with the **best division** of the network found in the last round (in step (3)).

(5) Stop when **no improvement** on the cut size occurs.

Note that if the initial assignment of vertices to groups is done randomly, the Kernighan-Lin algorithm may give (slightly) different answers when it is run twice on the same network.

The Kernighan-Lin algorithm (II)



(a) A mesh network of 547 vertices of the kind commonly used in finite element analysis.
(b) The best division found by the **Kernighan-Lin algorithm** when the task is to split the network into 2 groups of almost equal size.
This division involves cutting 40 edges in this mesh network and gives parts of 273 and 274 vertices.
(c) The best division found by **spectral partitioning** (alternative method).

Runtime of the Kernighan-Lin algorithm

The number of swaps performed during one round of the algorithm is equal to the smaller of the sizes of the two groups $\in [0, n / 2]$.

$\rightarrow$ in the worst case, there are $O(n)$ swaps.

For each swap, we have to examine all pairs of vertices in different groups to determine how the cut size would be affected if the pair was swapped.

At most (if both groups have the same size),
there are $n / 2 \times n / 2 = n^2 / 4$ such pairs, which is $O(n^2)$.

Runtime of the Kernighan-Lin algorithm (ii)

When a vertex *i* moves from one group to the other group, any edges connecting it to vertices in its current group become edges between groups after the swap.

Let us suppose that there are k_i^{same} such edges.

Similarly, any edges that *i* has to vertices in the other group, (say k_i^{other} ones) become within-group edges after the swap.

There is one exception. If *i* is being swapped with vertex *j* and they are connected by an edge, then the edge is still between the groups after the swap

→ the change in the cut size due to the movement of *i* is $-(k_i^{other} - k_i^{same} - A_{ij})$

A similar expression applies for vertex *j*.

→ the total change in cut size due to the swap is

$$-(k_i^{other} - k_i^{same} + k_j^{other} - k_j^{same} - 2A_{ij})$$

Runtime of the Kernighan-Lin algorithm (iii)

For a network stored in adjacency list form, the evaluation of this expression involves running through all the neighbors of i and j in turn, and hence takes time on the order of the average degree in the network, or $O\,(m/n)$ with m edges in the network.

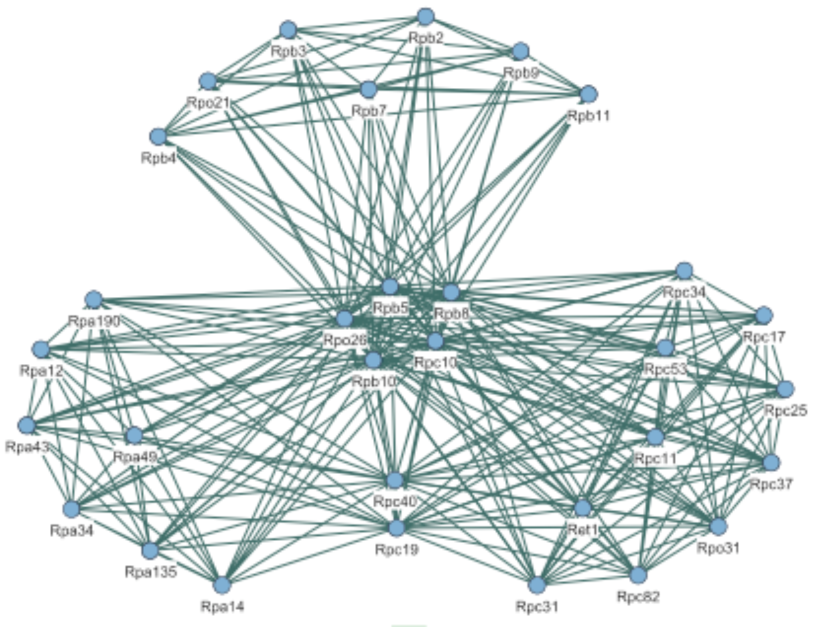
$\rightarrow$ the total running time is $O\,(\,n \times n^2 \times m/n\,) = O(mn^2)$.

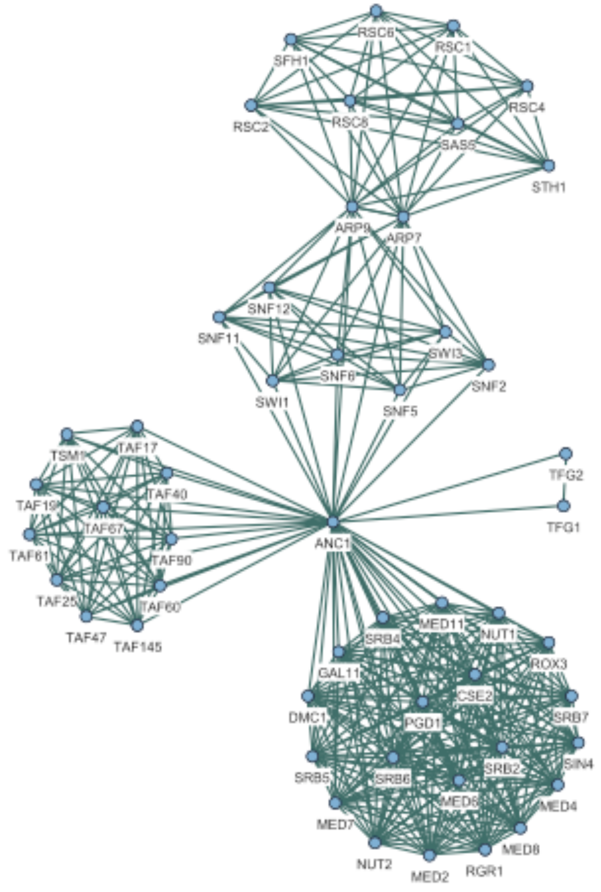
For a sparse network with $m \propto n$, this is $O(n^3)$.

For a dense network (with $m \rightarrow \frac{n(n-1)}{2}$) , this is $O(n^4)$.

This time still needs to be **multiplied** by the **number** of **rounds** the algorithm is run before the cut size stops decreasing.
For networks up to a few 1000 of vertices, this number may be between 5 and 10.

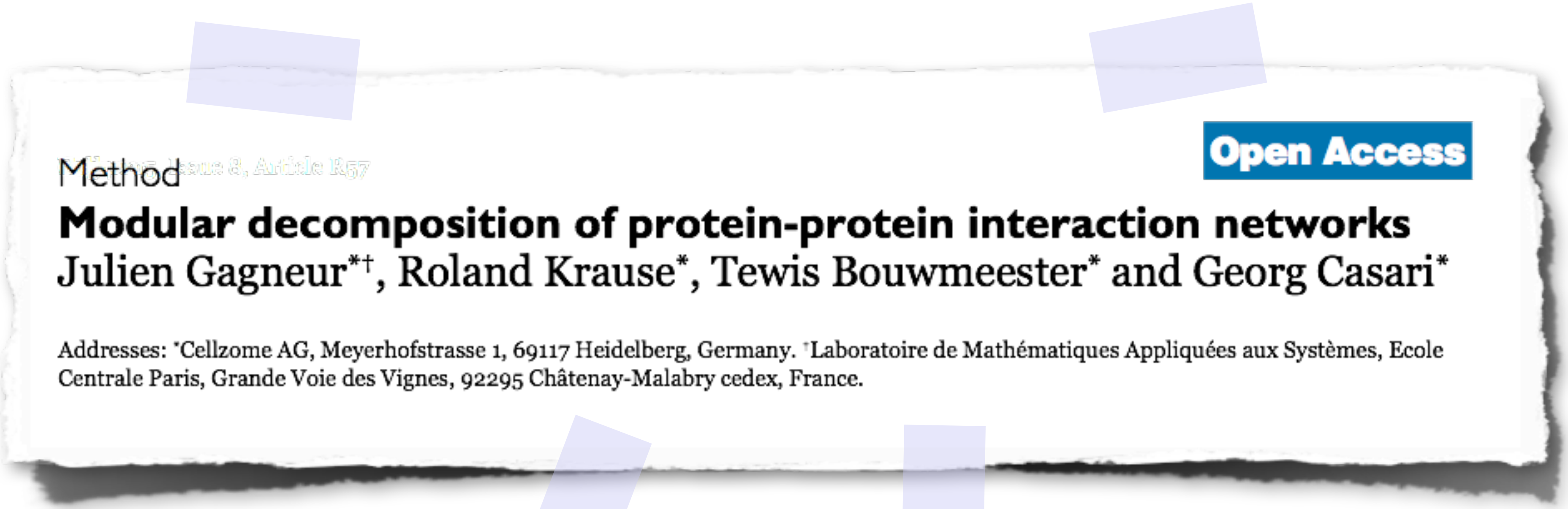
Reducing Network Complexity?





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

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



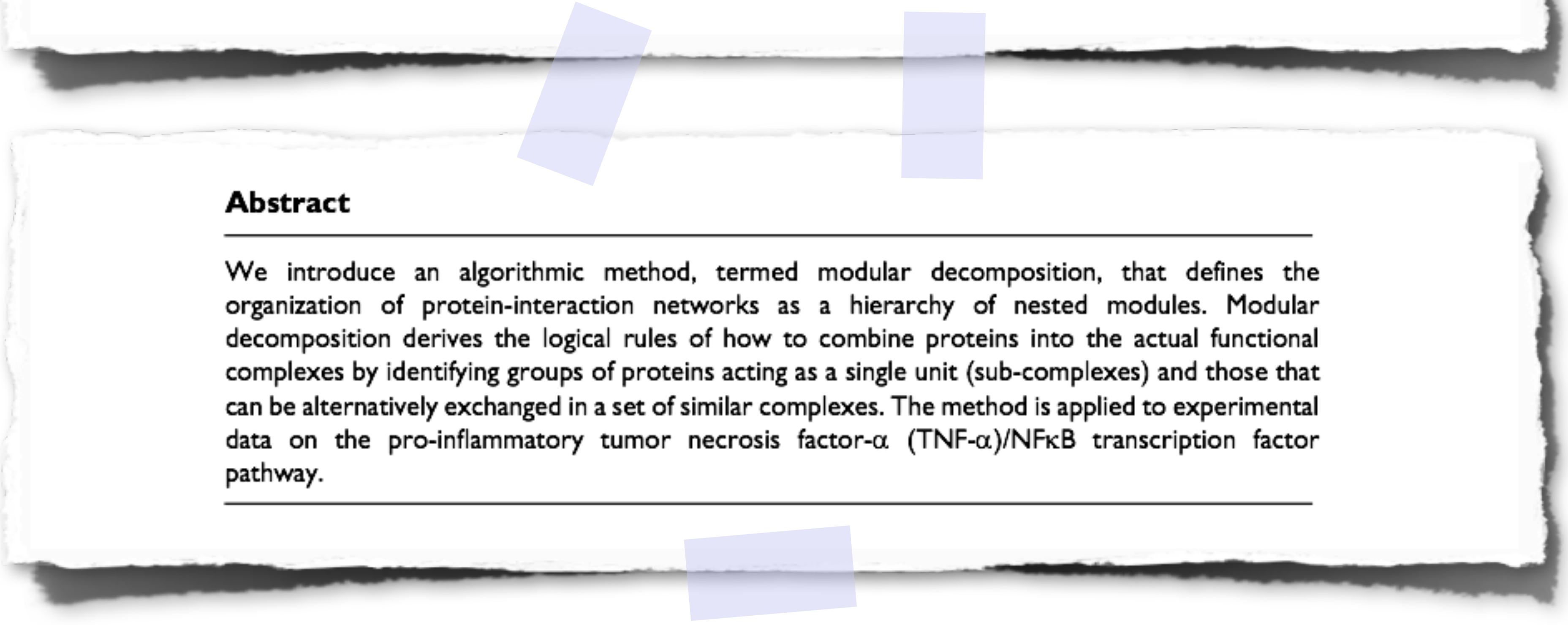
Method

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.

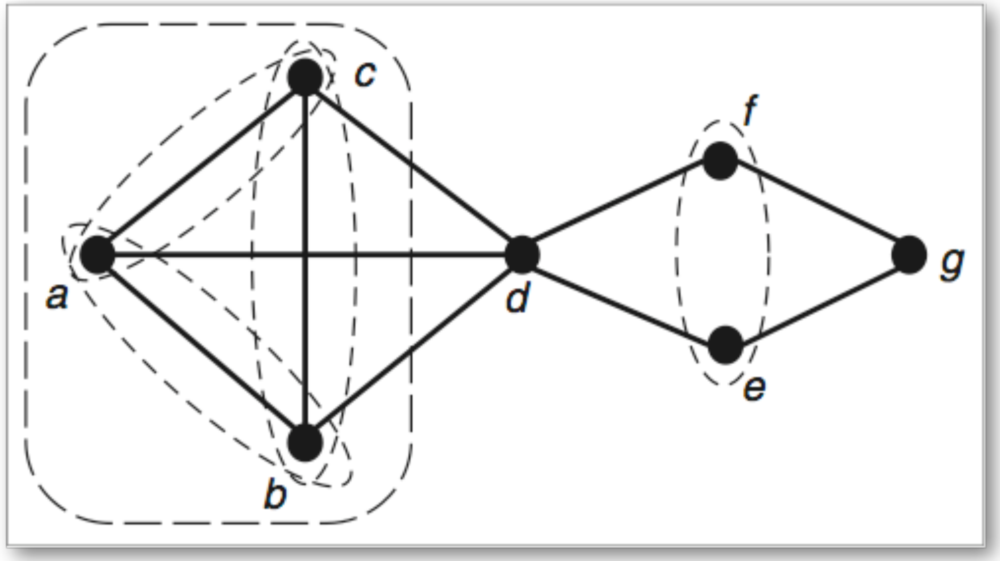
Aim: **identify** and properly **represent** the modularity of protein-protein interaction networks by identifying the **shared components** and the way they are arranged to generate **complexes**.

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

Georg Casari, Cellzome (Heidelberg)

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}

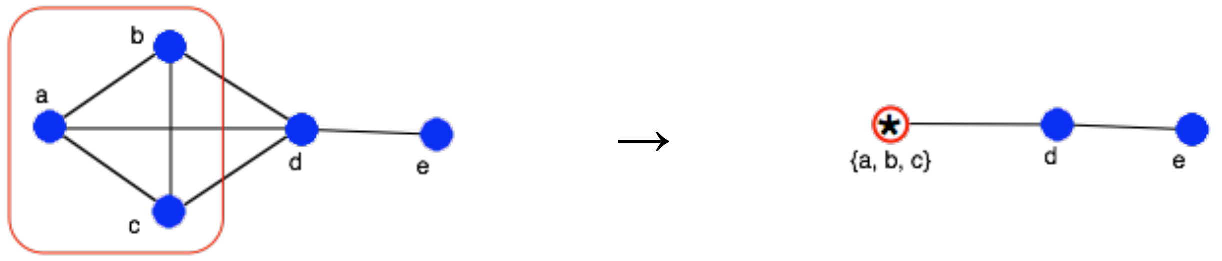
Quotient: representative node for a module

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

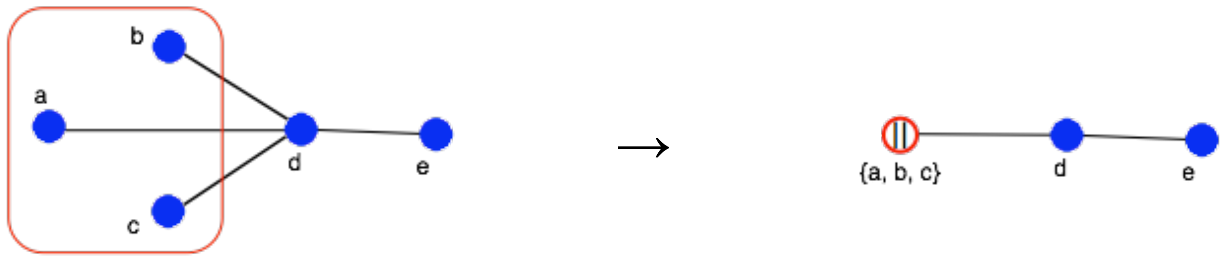
Gagneur et al, *Genome Biology* **5** (2004) R57

Quotients

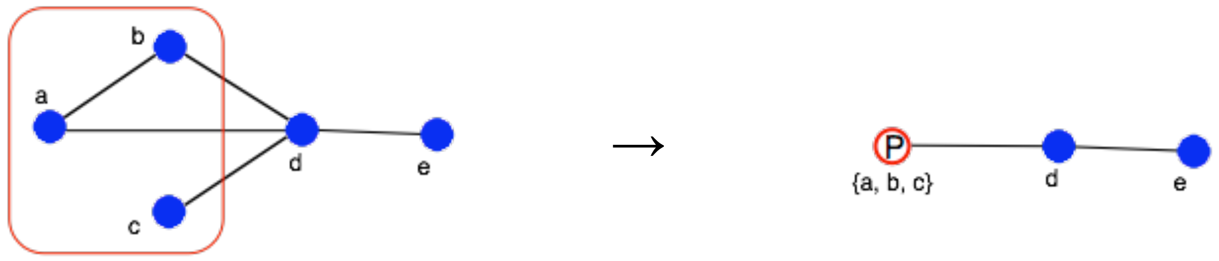
Series: all included nodes are direct **neighbors** (= **clique**)



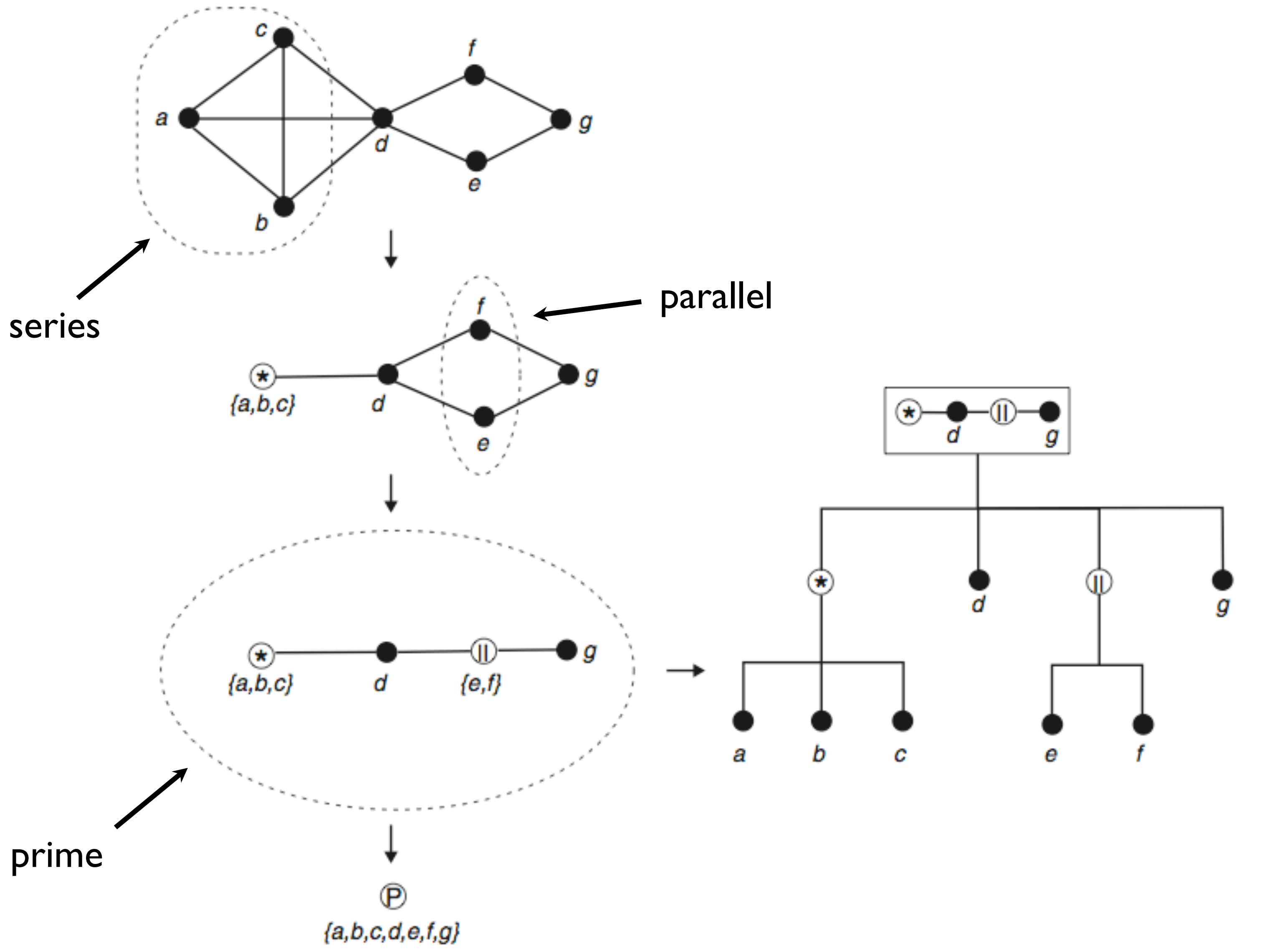
Parallel: all included nodes are **non-neighbors**



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

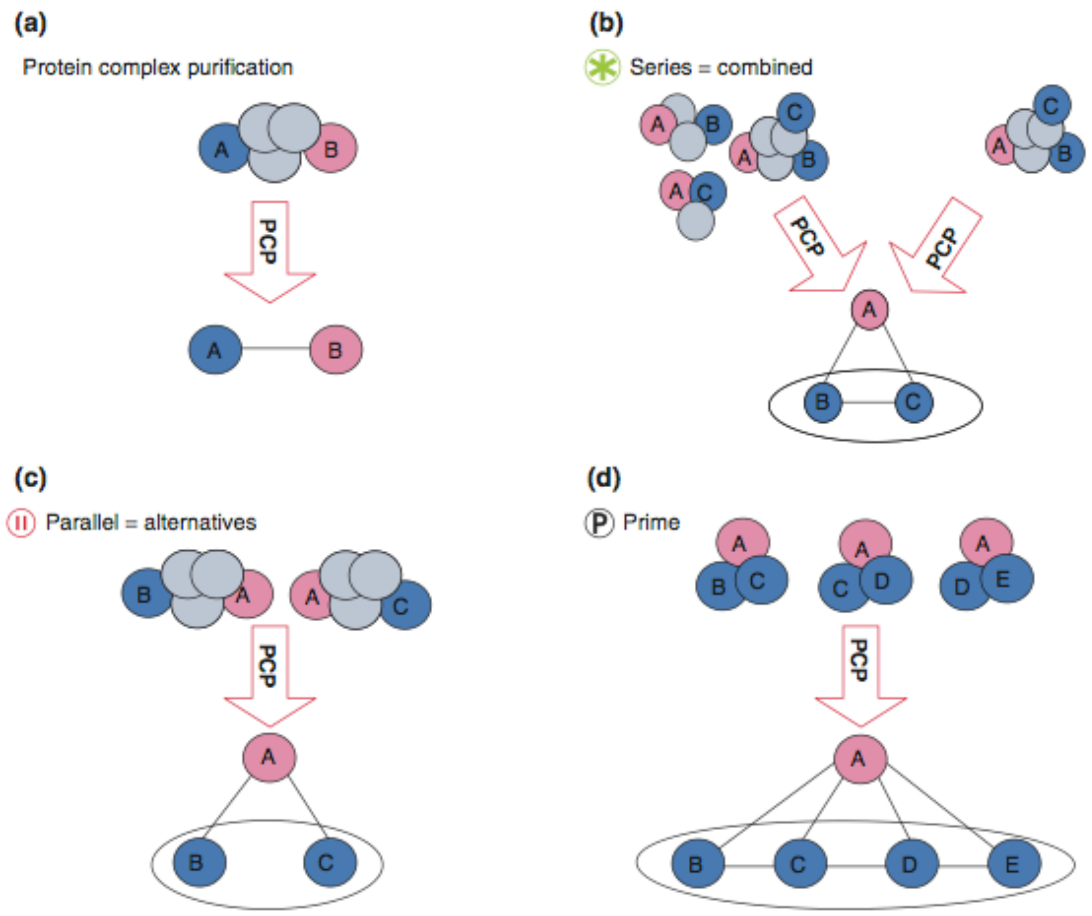
Graphs and module labels from systematic PCP experiments:

(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

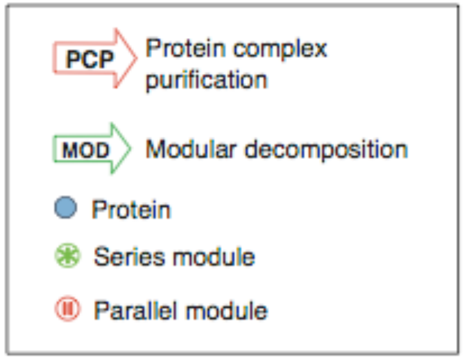
Two examples of modular decompositions of protein-protein interaction networks.

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).

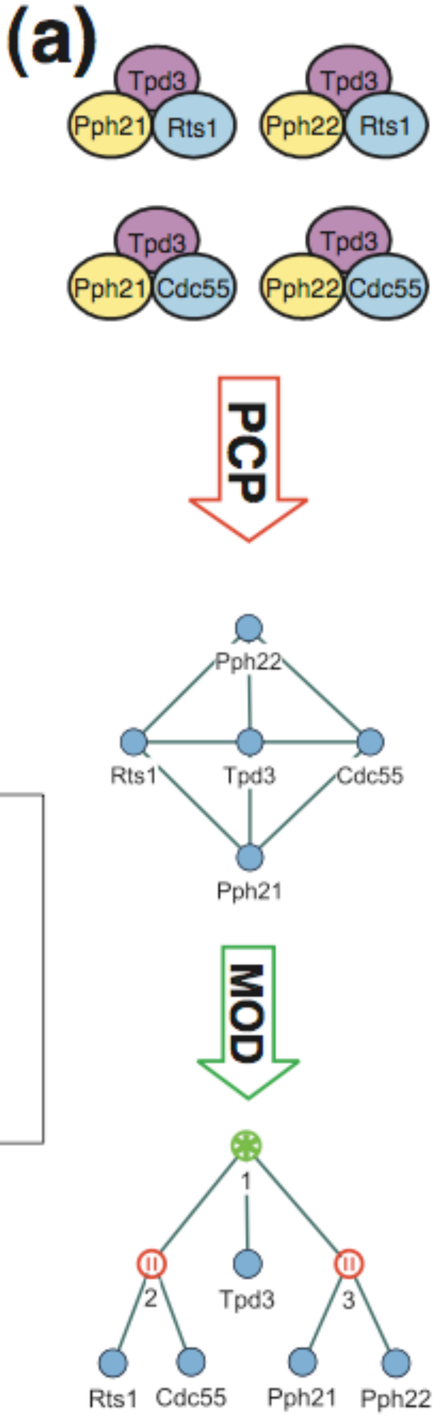
(a) Protein phosphatase 2A.

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

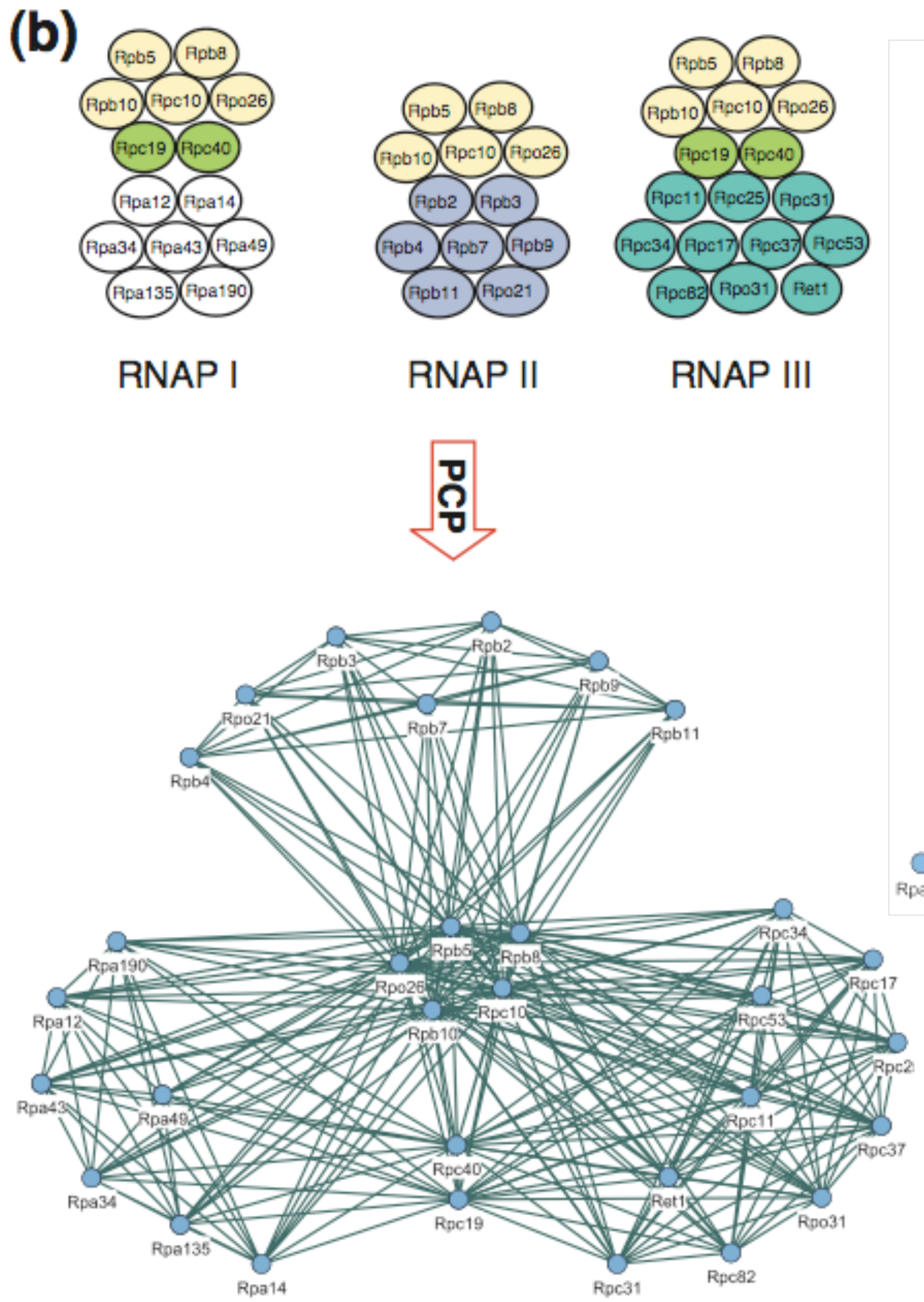
Here these are the catalytic proteins Pph21 and Pph22 (module 2) and the regulatory proteins Cdc55 and Rts1 (module 3), connected by the Tpd3 „backbone“.

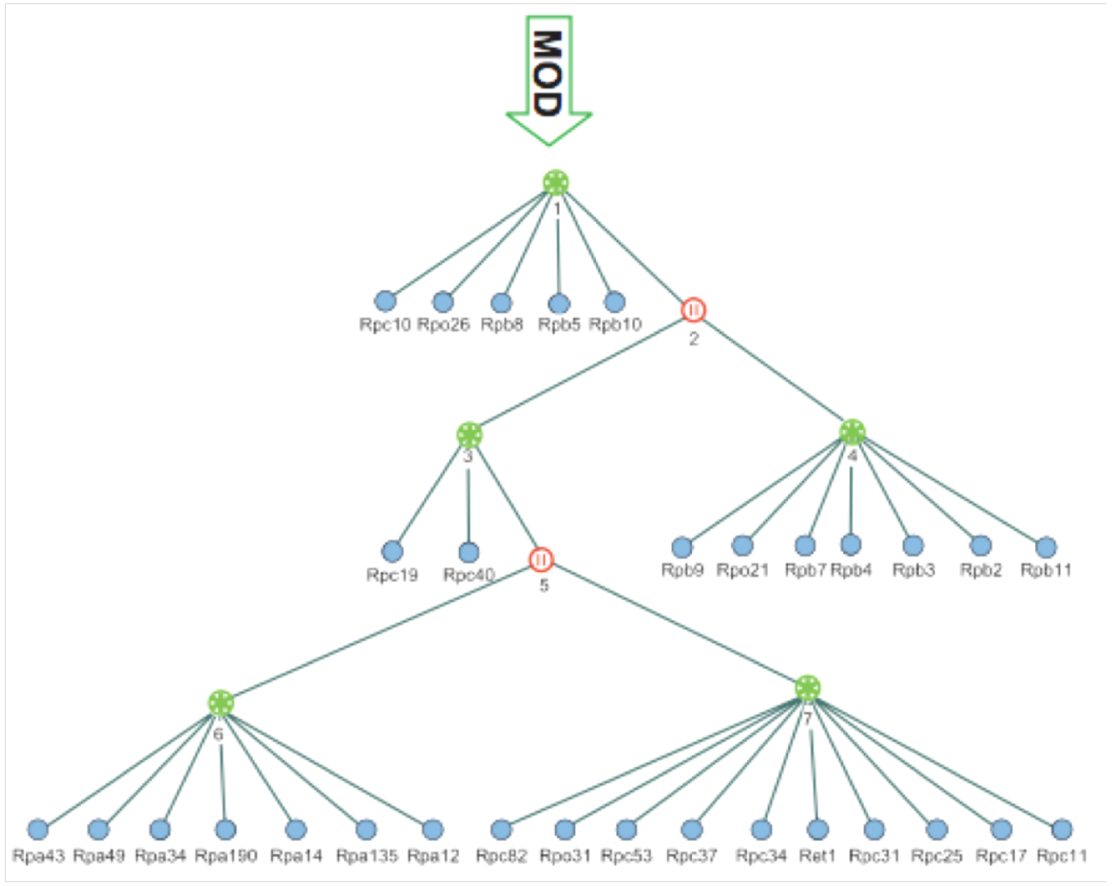


Notes: • Graph does not show functional alternatives!!!
• other decompositions also possible



RNA polymerases I, II and III





MOD
1
Rpc10 Rpo26 Rpb8 Rpb5 Rpb10
2
3
4
Rpc19 Rpc40
5
Rpb9 Rpo21 Rpb7 Rpb4 Rpb3 Rpb2 Rpb11
6
7
Rpa43 Rpa49 Rpa34 Rpa190 Rpa14 Rpa135 Rpa12 Rpc82 Rpo31 Rpc53 Rpc37 Rpc34 Ret1 Rpc31 Rpc25 Rpc17 Rpc11

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

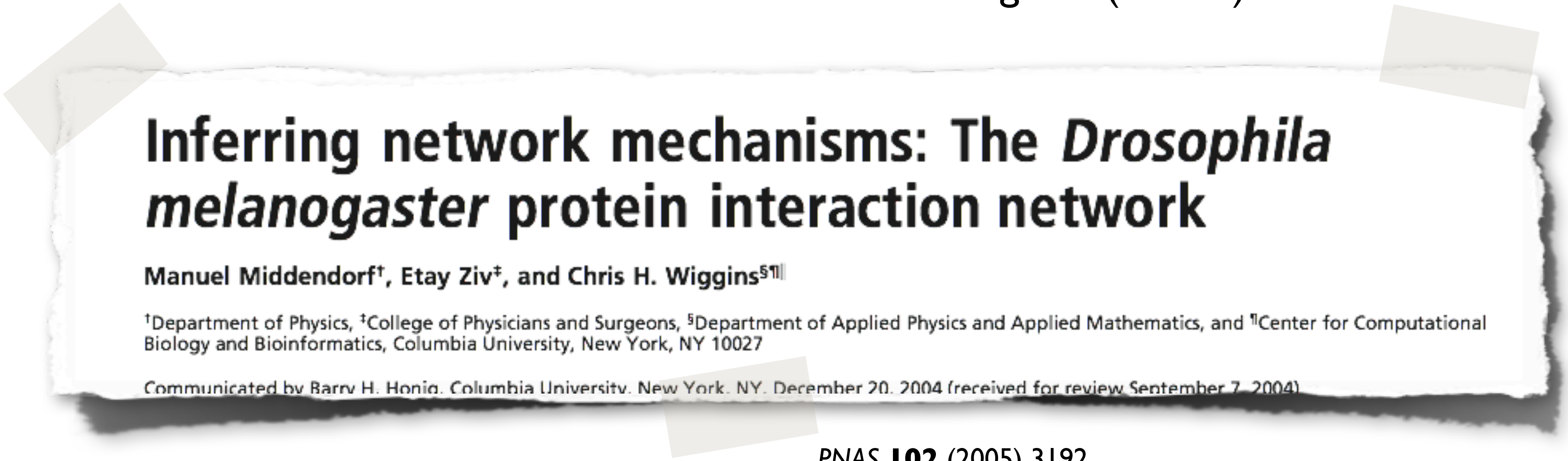
→ separate complexes do not always fulfill the strict requirements of modular graph decomposition.

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.

Network Growth Mechanisms

Given: an observed PPI network → how did it grow (evolve)?



Inferring network mechanisms: The *Drosophila melanogaster* protein interaction network

Manuel Middendorf[†], Etay Ziv[‡], and Chris H. Wiggins[§¶‖]

[†]Department of Physics, [‡]College of Physicians and Surgeons, [§]Department of Applied Physics and Applied Mathematics, and [¶]Center for Computational Biology and Bioinformatics, Columbia University, New York, NY 10027

Communicated by Barry H. Honig, Columbia University, New York, NY, December 20, 2004 (received for review September 7, 2004)

PNAS **102** (2005) 3192

Look at **network motifs** (local connectivity):
compare motif distributions from various network prototypes to fly network

Idea: each growth **mechanism** leads to a typical motif **distribution**,
even if global measures are comparable

The Fly Network

Y2H PPI network for *D. melanogaster* from Giot et al. [*Science* **302** (2003) 1727]

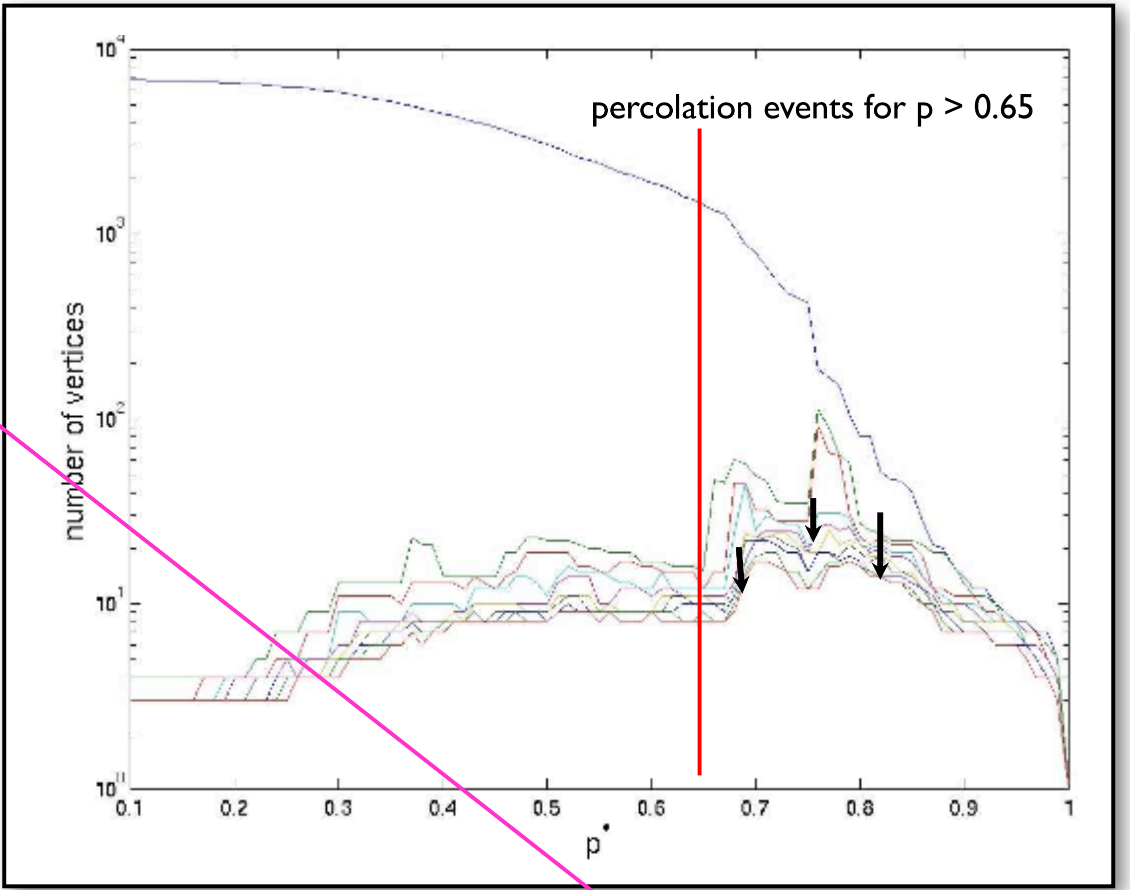
Giot *et al.* assigned a confidence score [0, 1] to every observed interaction.

$\rightarrow$ use only data with
p > 0.65 (0.5) because …

$\rightarrow$ remove self-interactions and isolated nodes

High confidence network with 3359 (4625) nodes and 2795 (4683) edges.

Use prototype networks of same size for training.

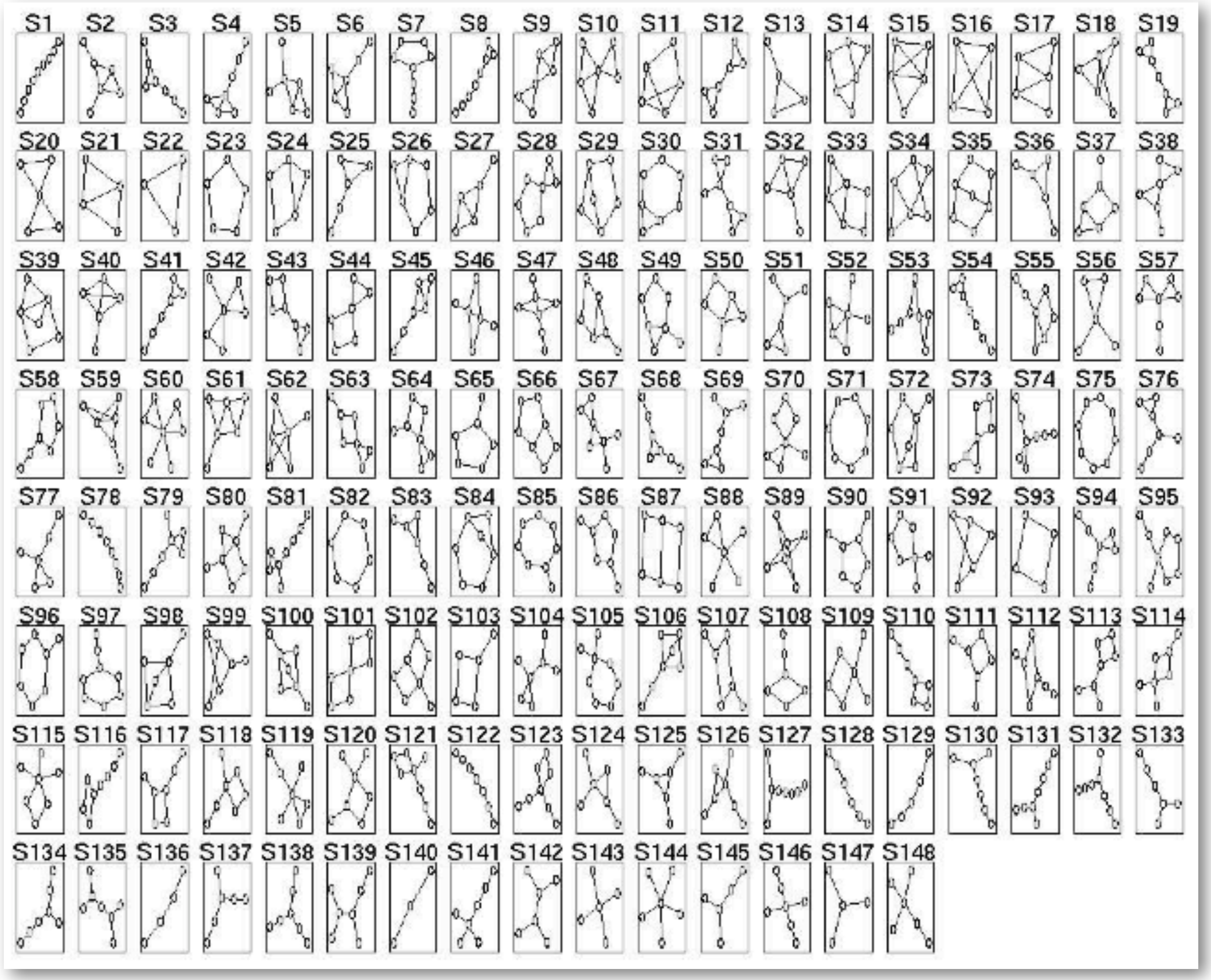


Size of largest components. At p = 0.65, there is one large component with 1433 nodes and the other 703 components contain at most 15 nodes.

Middendorf et al, *PNAS* **102** (2005) 3192

Network subgraphs -> motifs

All non-isomorphic subgraphs that can be generated with a walk of length 8



S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19
S20 S21 S22 S23 S24 S25 S26 S27 S28 S29 S30 S31 S32 S33 S34 S35 S36 S37 S38
S39 S40 S41 S42 S43 S44 S45 S46 S47 S48 S49 S50 S51 S52 S53 S54 S55 S56 S57
S58 S59 S60 S61 S62 S63 S64 S65 S66 S67 S68 S69 S70 S71 S72 S73 S74 S75 S76
S77 S78 S79 S80 S81 S82 S83 S84 S85 S86 S87 S88 S89 S90 S91 S92 S93 S94 S95
S96 S97 S98 S99 S100 S101 S102 S103 S104 S105 S106 S107 S108 S109 S110 S111 S112 S113 S114
S115 S116 S117 S118 S119 S120 S121 S122 S123 S124 S125 S126 S127 S128 S129 S130 S131 S132 S133
S134 S135 S136 S137 S138 S139 S140 S141 S142 S143 S144 S145 S146 S147 S148

Middendorf et al, *PNAS* **102** (2005) 3192

Growth Mechanisms

Generate 1000 networks, each, of the following 7 types
(same size as fly network, undefined parameters were scanned)

DMC	Duplication-mutation, preserving complementarity
DMR	Duplication with random mutations
RDS	Random static networks
RDG	Random growing network
LPA	Linear preferential attachment network (Albert-Barabasi)
AGV	Aging vertices network
SMW	Small world network

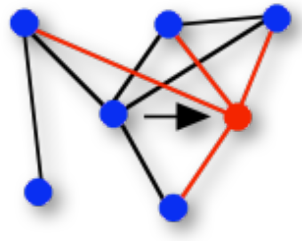
Growth Type 1: DMC

"Duplication – mutation with preserved complementarity"

Evolutionary idea: gene **duplication**, followed by a partial **loss** of function of one of the copies, making the other copy essential

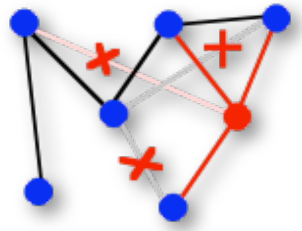
Algorithm:

Start from two connected nodes



• duplicate existing node with all interactions

• for all neighbors: delete with probability q_{del} either link from original node **or** from copy



Repeat these steps many (e.g. $N - 2$) times

Growth Type 2: DMR

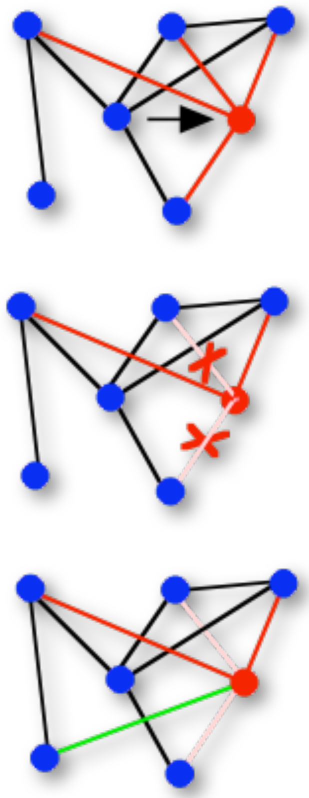
"Duplication with random mutations"

Gene duplication, but no correlation between original and copy (original unaffected by copy)

Algorithm:

Start growth from five-vertex cycle,
repeat *N* - 5 times:

- duplicate existing node with all interactions
- for all neighbors: delete with probability q_{del} link from copy
- add **new links** to non-neighbors with probability q_{new}/n



Growth Types 3–5: RDS, RDG, and LPA

RDS = static random network

Start from N nodes, add L links randomly

RDG = growing random network

Start from small random network, add nodes,
then edges between all existing nodes

LPA = linear preferential attachment

Add new nodes similar to Barabási-Albert algorithm,
but with preference according to ($k_i + \alpha$), $\alpha = 0\ldots5$
(BA for $\alpha = 0$)

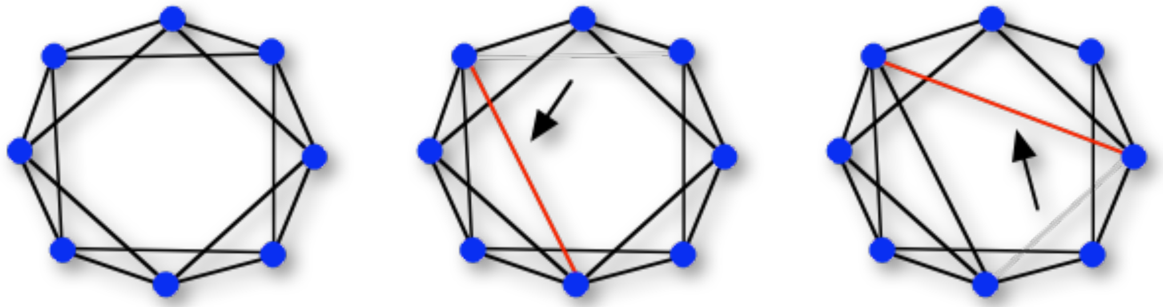
Growth Types 6-7: AGV and SMW

AGV = aging vertices network

Like growing random network,
but preference decreases with age of the node
→ citation network: more recent publications are cited more likely

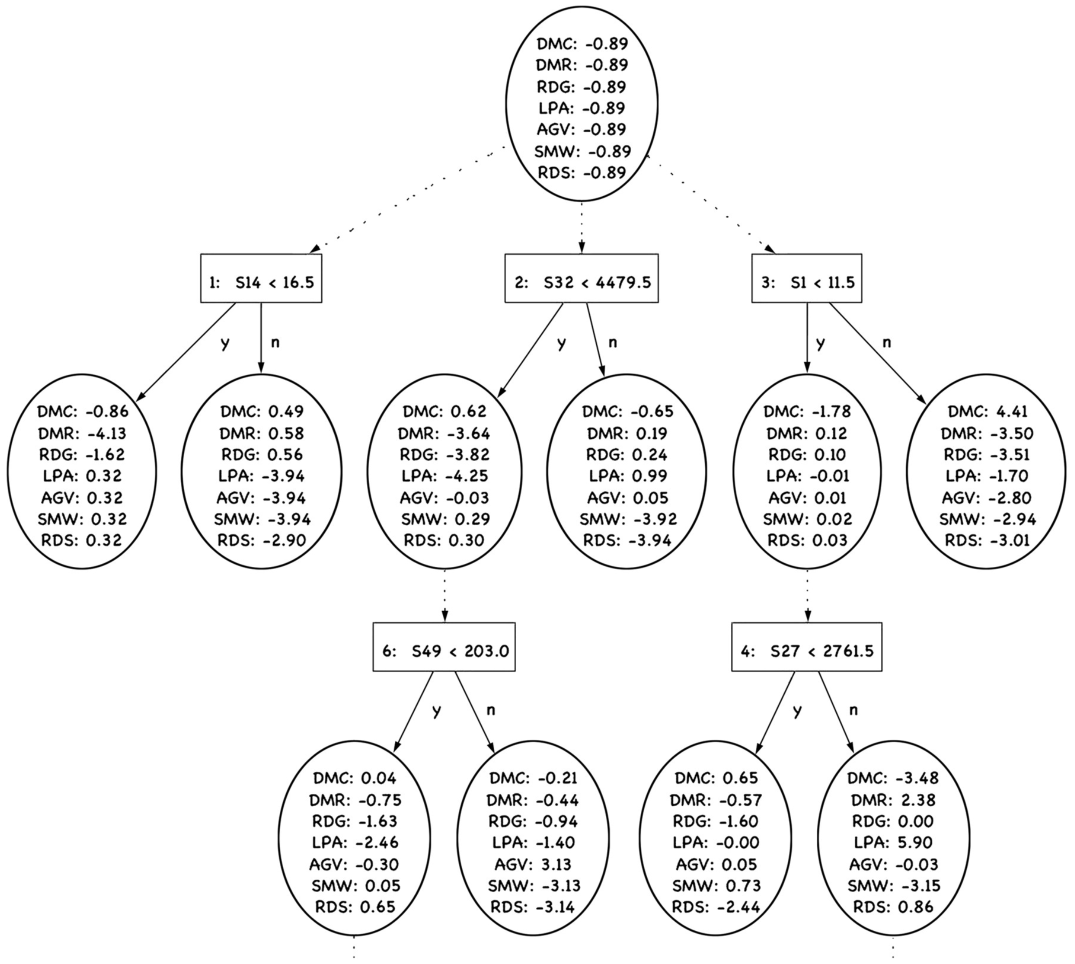
SMW = small world networks, see Watts, Strogatz, *Nature* **363**, 202 (1998)

Randomly rewire regular ring lattice



Alternating Decision Tree Classifier

Trained with the motif counts from 1000 networks of each of the 7 types
→ prototypes are well separated and can be reliably classified


DMC: -0.89 DMR: -0.89 RDG: -0.89 LPA: -0.89 AGV: -0.89 SMW: -0.89 RDS: -0.89
1: S14 < 16.5
2: S32 < 4479.5
3: S1 < 11.5
y
n
DMC: -0.86 DMR: -4.13 RDG: -1.62 LPA: 0.32 AGV: 0.32 SMW: 0.32 RDS: 0.32
DMC: 0.49 DMR: 0.58 RDG: 0.56 LPA: -3.94 AGV: -3.94 SMW: -3.94 RDS: -2.90
DMC: 0.62 DMR: -3.64 RDG: -3.82 LPA: -4.25 AGV: -0.03 SMW: 0.29 RDS: 0.30
DMC: -0.65 DMR: 0.19 RDG: 0.24 LPA: 0.99 AGV: 0.05 SMW: -3.92 RDS: -3.94
DMC: -1.78 DMR: 0.12 RDG: 0.10 LPA: -0.01 AGV: 0.01 SMW: 0.02 RDS: 0.03
DMC: 4.41 DMR: -3.50 RDG: -3.51 LPA: -1.70 AGV: -2.80 SMW: -2.94 RDS: -3.01
6: S49 < 203.0
4: S27 < 2761.5
DMC: 0.04 DMR: -0.75 RDG: -1.63 LPA: -2.46 AGV: -0.30 SMW: 0.05 RDS: 0.65
DMC: -0.21 DMR: -0.44 RDG: -0.94 LPA: -1.40 AGV: 3.13 SMW: -3.13 RDS: -3.14
DMC: 0.65 DMR: -0.57 RDG: -1.60 LPA: -0.00 AGV: 0.05 SMW: 0.73 RDS: -2.44
DMC: -3.48 DMR: 2.38 RDG: 0.00 LPA: 5.90 AGV: -0.03 SMW: -3.15 RDS: 0.86

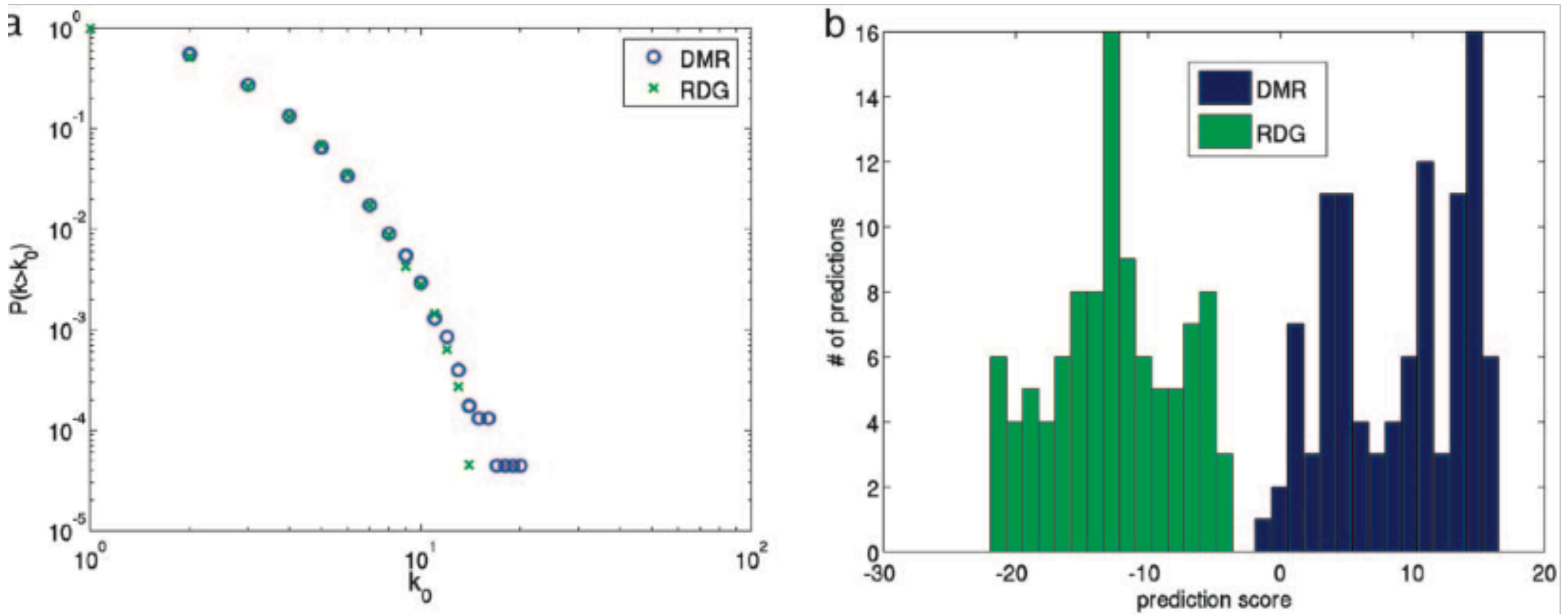
Decision nodes count occurrence of subgraphs

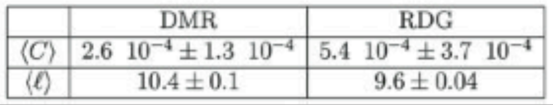
Prediction accuracy for networks similar to fly network with $p = 0.5$:

	Prediction						
Truth	DMR	DMC	AGV	LPA	SMW	RDS	RDG
DMR	99.3	0.0	0.0	0.0	0.0	0.1	0.6
DMC	0.0	99.7	0.0	0.0	0.3	0.0	0.0
AGV	0.0	0.1	84.7	13.5	1.2	0.5	0.0
LPA	0.0	0.0	10.3	89.6	0.0	0.0	0.1
SMW	0.0	0.0	0.6	0.0	99.0	0.4	0.0
RDS	0.0	0.0	0.2	0.0	0.8	99.0	0.0
RDG	0.9	0.0	0.0	0.1	0.0	0.0	99.0

Middendorf et al, *PNAS* **102** (2005) 3192

Are the generated networks different?





	DMR	RDG
$\langle C \rangle$	$2.6 \ 10^{-4} \pm 1.3 \ 10^{-4}$	$5.4 \ 10^{-4} \pm 3.7 \ 10^{-4}$
$\langle \ell \rangle$	10.4 ± 0.1	9.6 ± 0.04

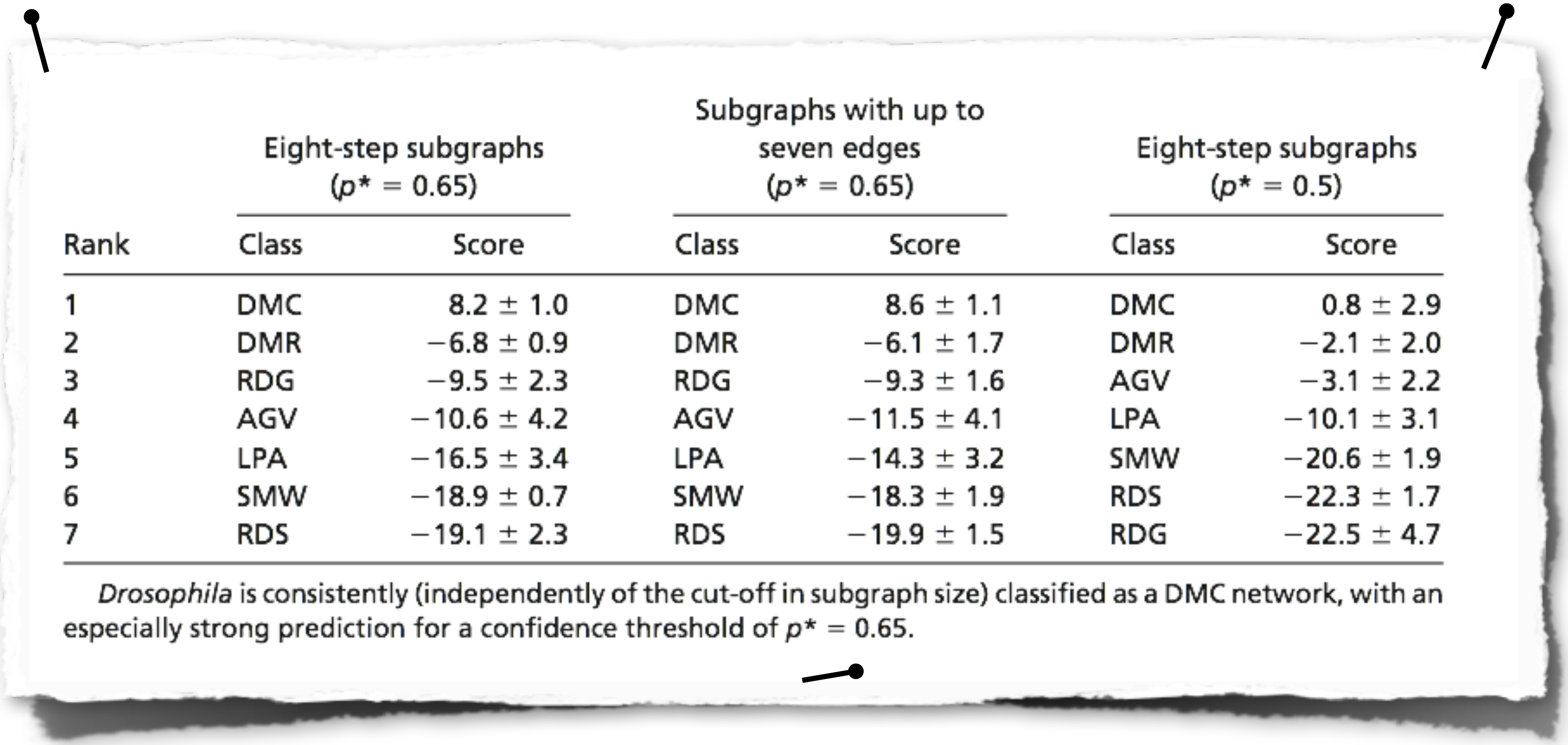
Clustering coefficient
Average shortest path length

Example: DMR vs. RDG: Similar global parameters <C> and <l> (left),
but different counts of the network motifs (right)

-> networks can (only) be perfectly separated by motif-based classifier

Middendorf et al, *PNAS* **102** (2005) 3192

How Did the Fly Evolve?



	Eight-step subgraphs ($p^* = 0.65$)		Subgraphs with up to seven edges ($p^* = 0.65$)		Eight-step subgraphs ($p^* = 0.5$)	
Rank	Class	Score	Class	Score	Class	Score
1	DMC	8.2 ± 1.0	DMC	8.6 ± 1.1	DMC	0.8 ± 2.9
2	DMR	-6.8 ± 0.9	DMR	-6.1 ± 1.7	DMR	-2.1 ± 2.0
3	RDG	-9.5 ± 2.3	RDG	-9.3 ± 1.6	AGV	-3.1 ± 2.2
4	AGV	-10.6 ± 4.2	AGV	-11.5 ± 4.1	LPA	-10.1 ± 3.1
5	LPA	-16.5 ± 3.4	LPA	-14.3 ± 3.2	SMW	-20.6 ± 1.9
6	SMW	-18.9 ± 0.7	SMW	-18.3 ± 1.9	RDS	-22.3 ± 1.7
7	RDS	-19.1 ± 2.3	RDS	-19.9 ± 1.5	RDG	-22.5 ± 4.7

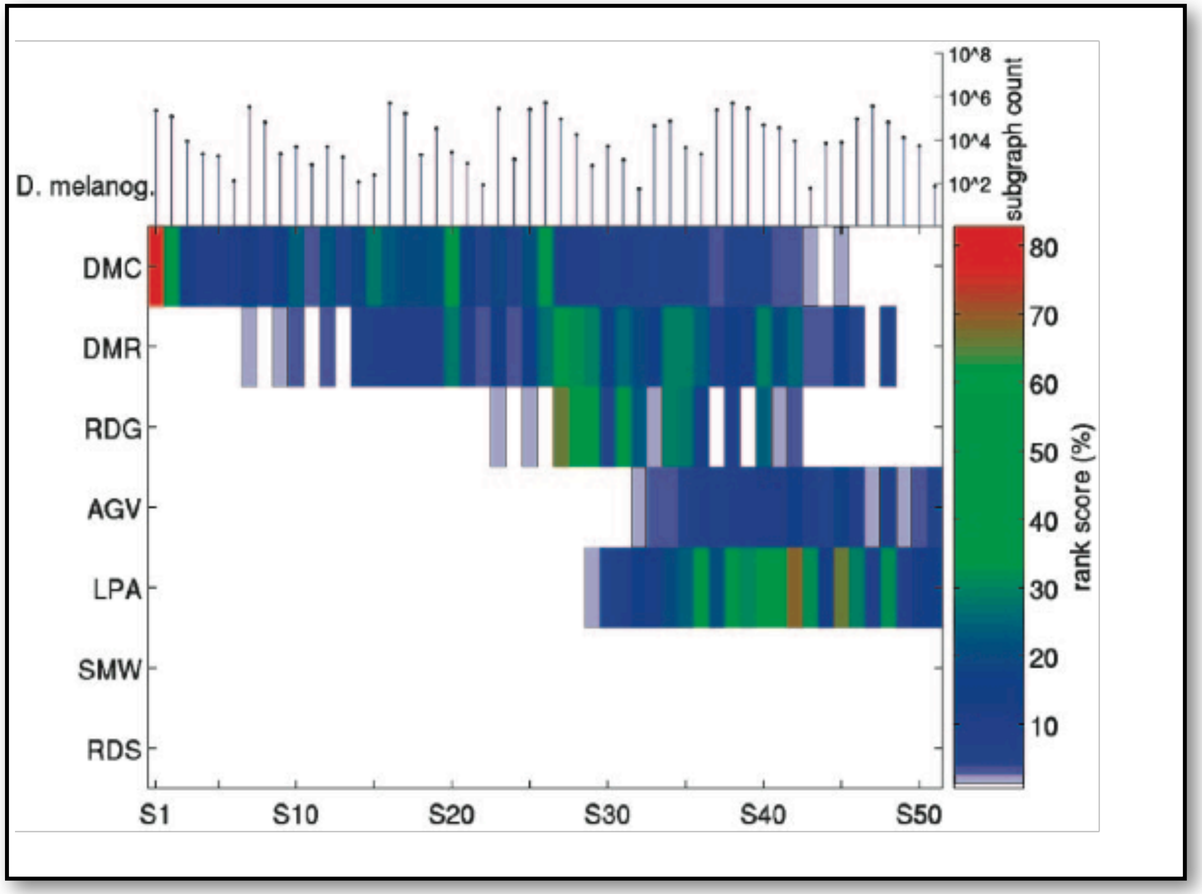
Drosophila is consistently (independently of the cut-off in subgraph size) classified as a DMC network, with an especially strong prediction for a confidence threshold of $p^* = 0.65$.

→ Best overlap with DMC (Duplication-mutation, preserved complementarity)

→ Scale-free (LPA) or random networks (RDS/RDG) are very unlikely

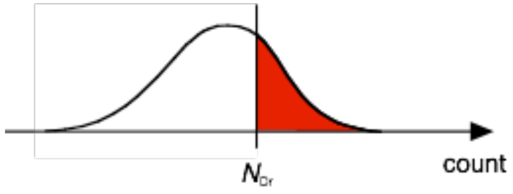
Middendorf et al, *PNAS* **102** (2005) 3192

Motif Count Frequencies



-> DMC and DMR networks contain most subgraphs in similar amount as fly network (top).

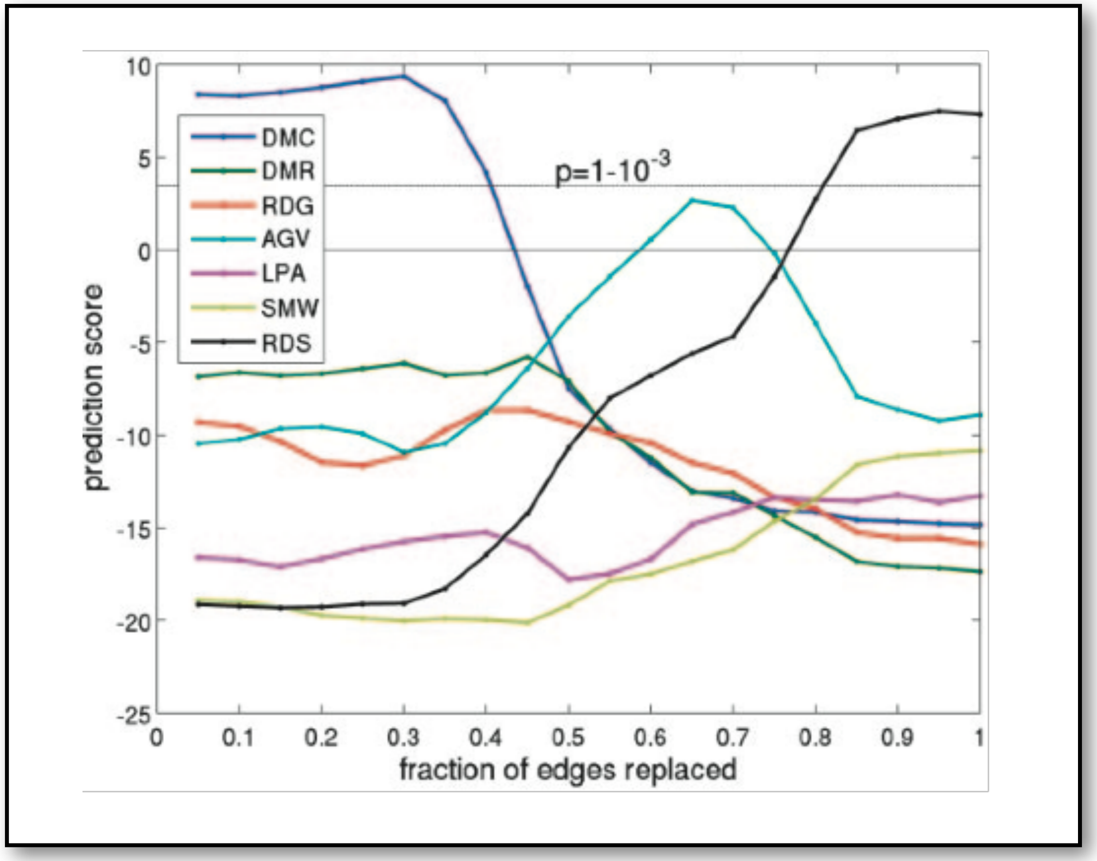
rank score: fraction of test networks with a higher count than Drosophila
(50% = same count as fly on avg.)



Middendorf et al, *PNAS* **102** (2005) 3192

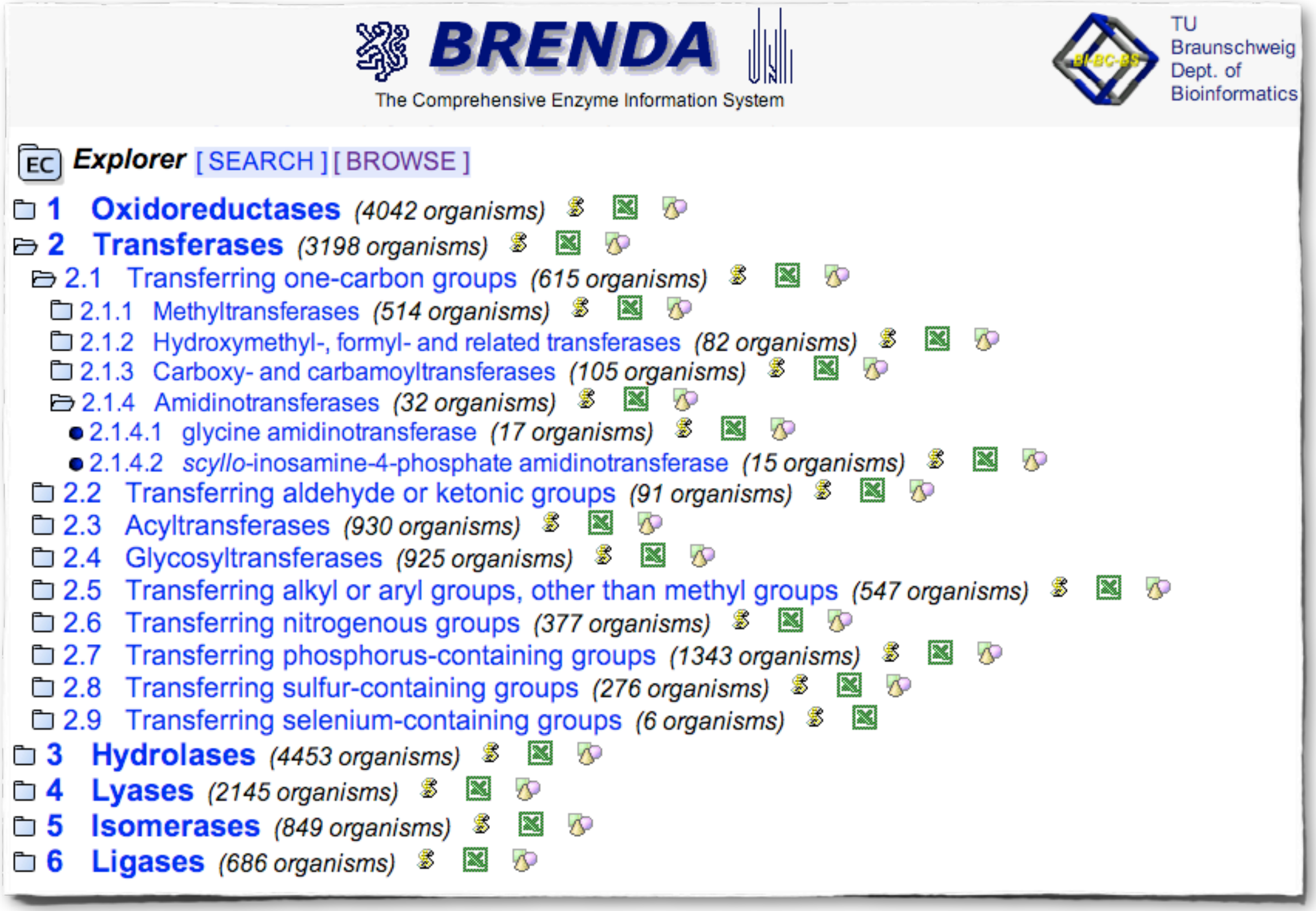
Experimental Errors?

Randomly replace edges in **fly** network and **classify** again:



→ Classification **unchanged** for **≤ 30%** incorrect edges,
at higher values RDS takes over (as to be expected)

What Does a Protein Do?



BRENDA
The Comprehensive Enzyme Information System

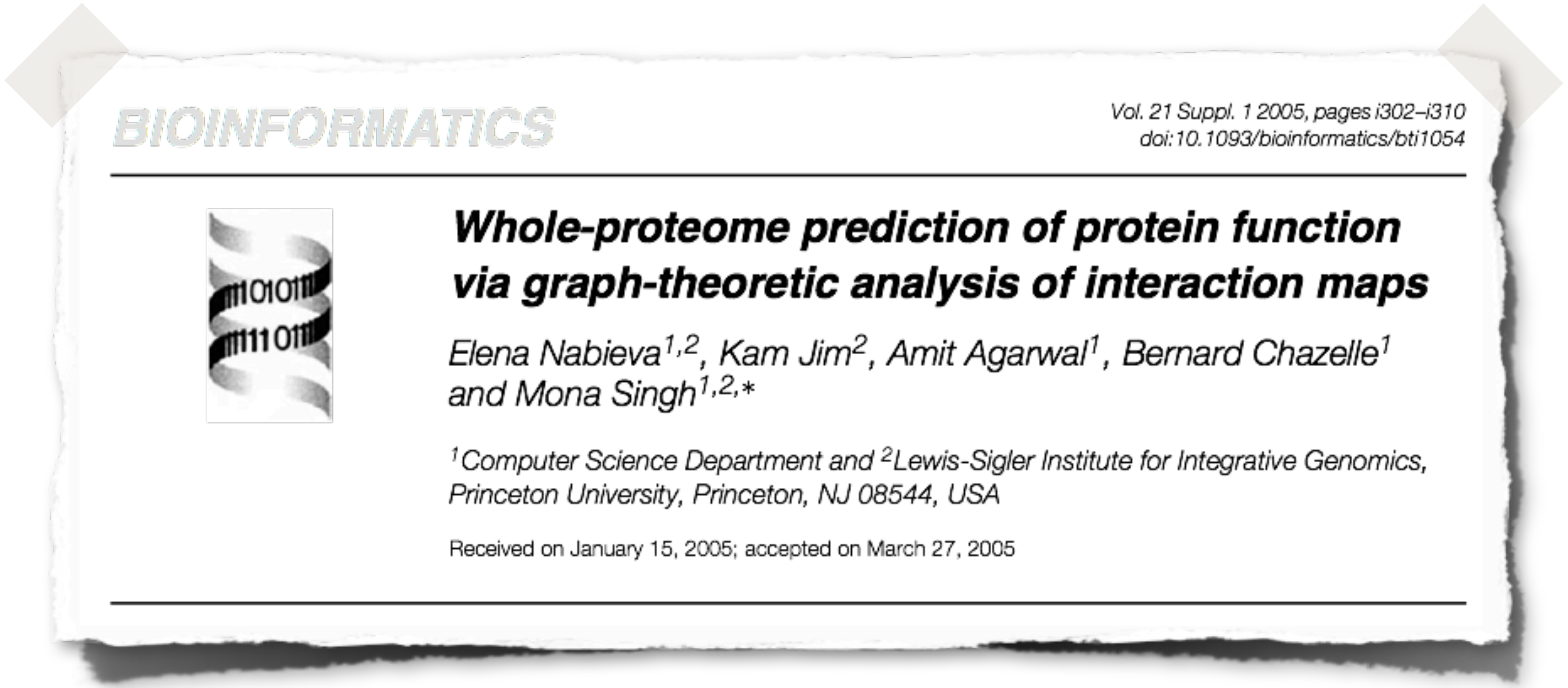
TU Braunschweig Dept. of Bioinformatics

EC Explorer [SEARCH] [BROWSE]

- 1 **Oxidoreductases** *(4042 organisms)*
- 2 **Transferases** *(3198 organisms)*
  - 2.1 Transferring one-carbon groups *(615 organisms)*
    - 2.1.1 Methyltransferases *(514 organisms)*
    - 2.1.2 Hydroxymethyl-, formyl- and related transferases *(82 organisms)*
    - 2.1.3 Carboxy- and carbamoyltransferases *(105 organisms)*
    - 2.1.4 Amidinotransferases *(32 organisms)*
      - 2.1.4.1 glycine amidinotransferase *(17 organisms)*
      - 2.1.4.2 *scyllo*-inosamine-4-phosphate amidinotransferase *(15 organisms)*
  - 2.2 Transferring aldehyde or ketonic groups *(91 organisms)*
  - 2.3 Acyltransferases *(930 organisms)*
  - 2.4 Glycosyltransferases *(925 organisms)*
  - 2.5 Transferring alkyl or aryl groups, other than methyl groups *(547 organisms)*
  - 2.6 Transferring nitrogenous groups *(377 organisms)*
  - 2.7 Transferring phosphorus-containing groups *(1343 organisms)*
  - 2.8 Transferring sulfur-containing groups *(276 organisms)*
  - 2.9 Transferring selenium-containing groups *(6 organisms)*
- 3 **Hydrolases** *(4453 organisms)*
- 4 **Lyases** *(2145 organisms)*
- 5 **Isomerases** *(849 organisms)*
- 6 **Ligases** *(686 organisms)*

Enzyme **C**lassification scheme
(from http://www.brenda-enzymes.org/)

What about Un-Classified Proteins?


BIOINFORMATICS

Vol. 21 Suppl. 1 2005, pages i302–i310
doi:10.1093/bioinformatics/bti1054

Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps

Elena Nabieva[1,2], Kam Jim[2], Amit Agarwal[1], Bernard Chazelle[1] and Mona Singh[1,2,*]

[1]Computer Science Department and [2]Lewis-Sigler Institute for Integrative Genomics, Princeton University, Princeton, NJ 08544, USA

Received on January 15, 2005; accepted on March 27, 2005

Many **unclassified proteins**:
→ estimate: ~1/3 of the yeast proteome not annotated functionally
→ BioGRID: 4495 proteins in the largest cluster of the yeast physical interaction map.
only 2946 have a MIPS functional annotation

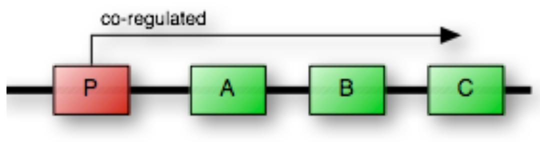
Partition the Graph

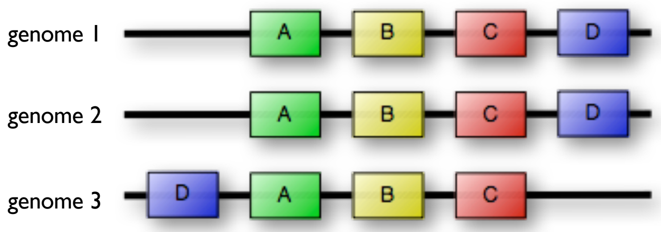
Large **PPI networks** can be built from (see V3, V4, V5):

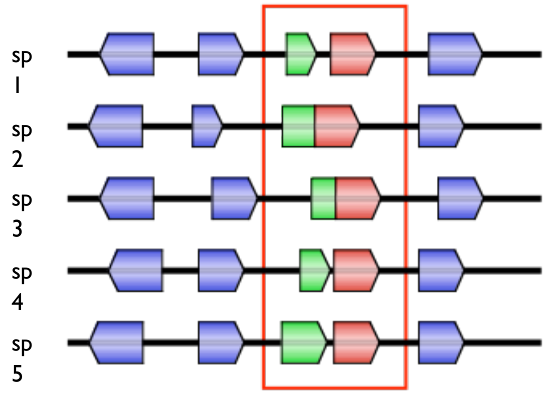
• HT experiments (Y2H, TAP, synthetic lethality, coexpression, coregulation, …)

• predictions (gene profiling, gene neighborhood, phylogenetic profiles, …)

→ proteins that are functionally linked







Identify **unknown functions** from **clustering** of these networks by, e.g.:

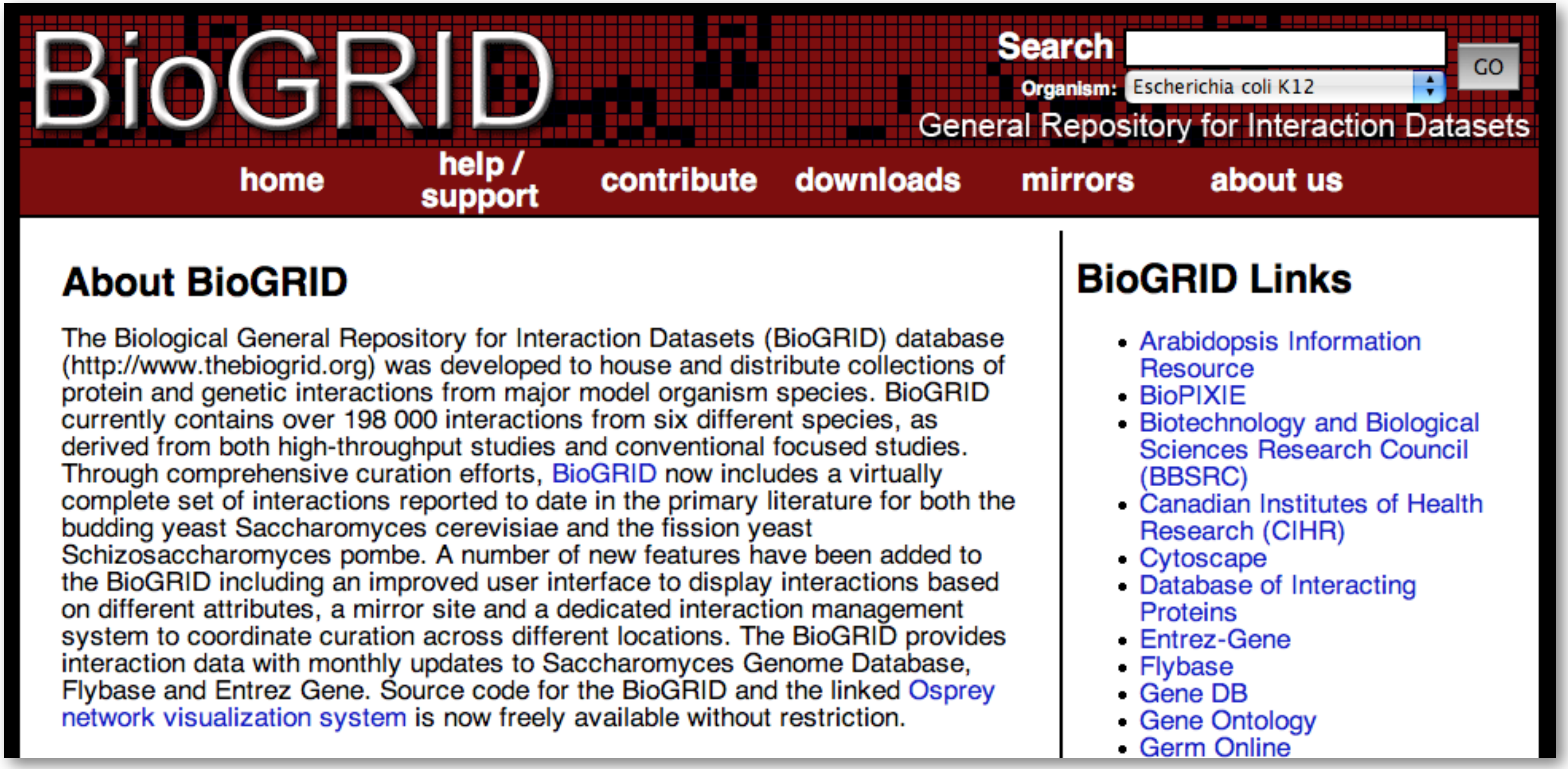
• shared interactions (similar neighborhood)

• membership in a community

• similarity of shortest path vectors to all other proteins (= similar path into the rest of the network)

Protein Interactions

Nabieva et al used the *S. cerevisiae* dataset from GRID of 2005 (now BioGRID)
→ 4495 proteins and 12 531 physical interactions in the largest cluster


BioGRID
Search
Organism: Escherichia coli K12
GO
General Repository for Interaction Datasets
home
help / support
contribute
downloads
mirrors
about us

About BioGRID

The Biological General Repository for Interaction Datasets (BioGRID) database (http://www.thebiogrid.org) was developed to house and distribute collections of protein and genetic interactions from major model organism species. BioGRID currently contains over 198 000 interactions from six different species, as derived from both high-throughput studies and conventional focused studies. Through comprehensive curation efforts, BioGRID now includes a virtually complete set of interactions reported to date in the primary literature for both the budding yeast Saccharomyces cerevisiae and the fission yeast Schizosaccharomyces pombe. A number of new features have been added to the BioGRID including an improved user interface to display interactions based on different attributes, a mirror site and a dedicated interaction management system to coordinate curation across different locations. The BioGRID provides interaction data with monthly updates to Saccharomyces Genome Database, Flybase and Entrez Gene. Source code for the BioGRID and the linked Osprey network visualization system is now freely available without restriction.

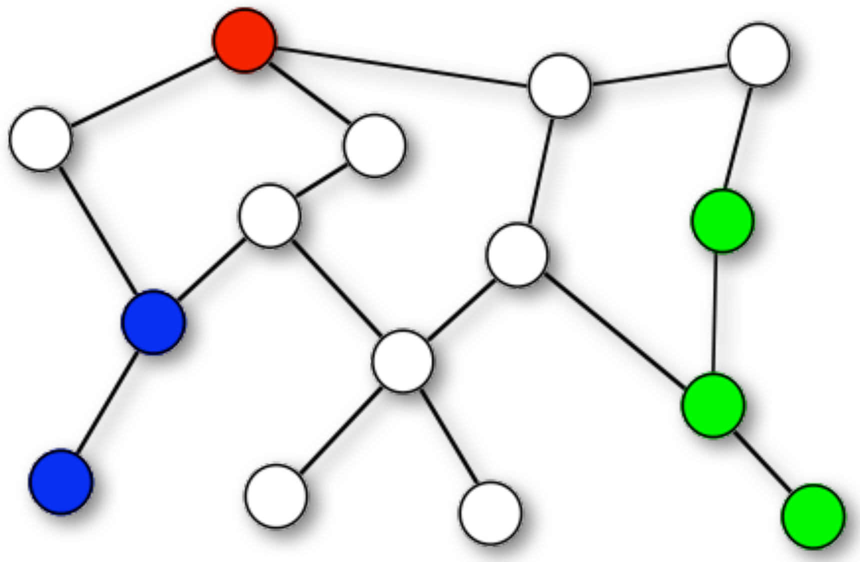
BioGRID Links

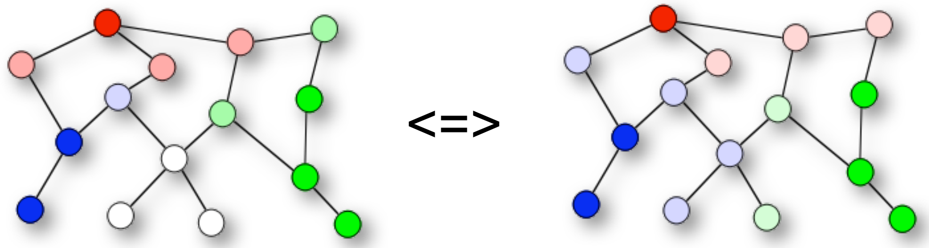
- Arabidopsis Information Resource
- BioPIXIE
- Biotechnology and Biological Sciences Research Council (BBSRC)
- Canadian Institutes of Health Research (CIHR)
- Cytoscape
- Database of Interacting Proteins
- Entrez-Gene
- Flybase
- Gene DB
- Gene Ontology
- Germ Online

http://www.thebiogrid.org/about.php

Function Annotation

Task: **predict** function (= functional annotation) for an unlabeled protein from the **available** annotations of other proteins in the network





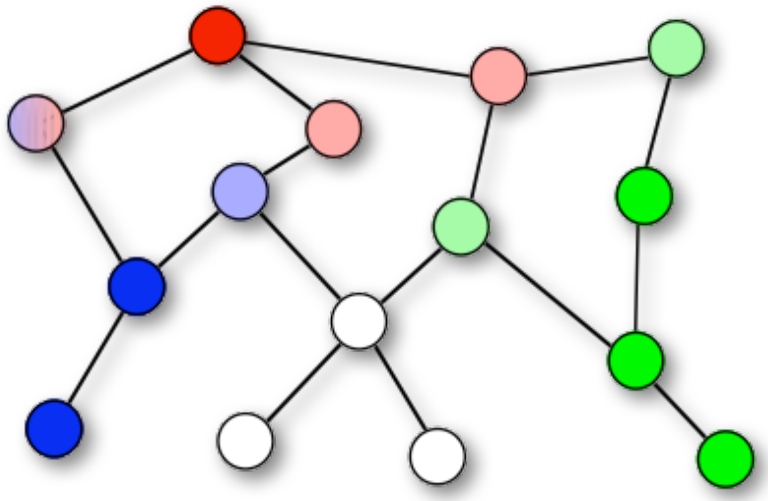
Similar task:
How to **assign colors** to the white nodes?

Use information on:
- distance to colored nodes
- local connectivity
- reliability of the links
- …

Algorithm I: Majority

This concept was presented in
Schwikowski, Uetz, and Fields, " A network of protein–protein interactions in yeast"
Nat. Biotechnol. **18** (2000) 1257

Consider all direct neighbors and **sum up** how often a certain **annotation occurs**
→ score for an annotation = count among the direct neighbors
→ take the 3 most frequent functions



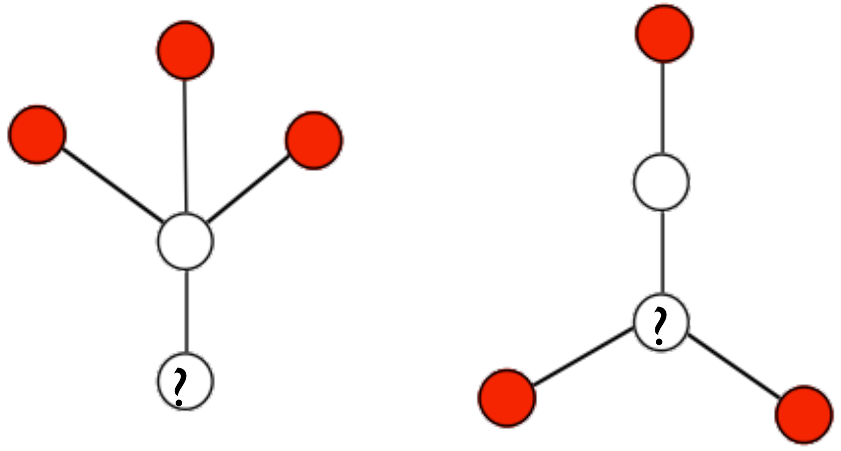
Majority makes only limited use of the local connectivity
→ cannot assign function to next-neighbors

For weighted graphs:
→ use weighted sum

Extended Majority: Neighborhood

This concept was presented in
Hishigaki, Nakai, Ono, Tanigami, and Takagi, "Assessment of prediction accuracy of protein function from protein–protein interaction data",
Yeast **18** (2001) 523

Look for **overrepresented** functions within a given **radius** of 1, 2, or 3 links
→ use as function score the value of a χ^2–test



Neighborhood algorithm does not consider local network topology

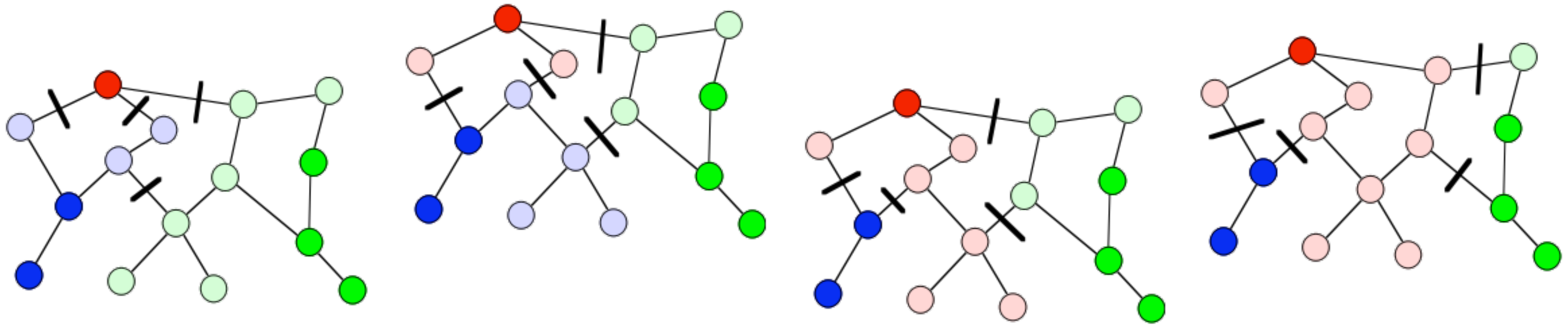
Both examples (left) are treated **identically** with *r* = 2
although the right situation feels more certain (2 direct neighbors of ? are labeled)

Minimize Changes: GenMultiCut

This concept was presented in
Karaoz, Murali, Letovsky, Zheng, Ding, Cantor, and Kasif, "Whole-genome annotation by using evidence integration in functional-linkage networks"
PNAS **101** (2004) 2888

"Annotate proteins so as to **minimize** the number of times that **different** functions are associated to **neighboring** (i.e. interacting) proteins"

→ generalization of the multiway *k*-cut problem for weighted edges,
can be stated as an integer linear program (ILP)



Multiple possible solutions → scores from **frequency** of annotations

Nabieva *et al*: FunctionalFlow

Extend the idea of **"guilty by association"**

$\rightarrow$ each annotated protein is considered as a source of "function"-flow

$\rightarrow$ propagate/simulate for a few time steps

$\rightarrow$ choose the annotation *a* with the highest accumulated flow

Each node *u* has a reservoir $R_t(u)$, each edge a capacity constraint (weight) $w_{u,v}$

Initially: $R_0^a(u) = \begin{cases} \infty, & \text{if } u \text{ is annotated with } a, \\ 0, & \text{otherwise.} \end{cases}$ and $g_0^a(u,v) = 0$

Then: **downhill flow** from node *u* to neighbor node *v*:

$$g_t^a(u,v) = \begin{cases} 0, & \text{if } R_{t-1}^a(u) < R_{t-1}^a(v) \\ \min\left(w_{u,v}, \frac{w_{u,v}}{\sum_{(u,y)\in E} w_{u,y}}\right), & \text{otherwise.} \end{cases}$$

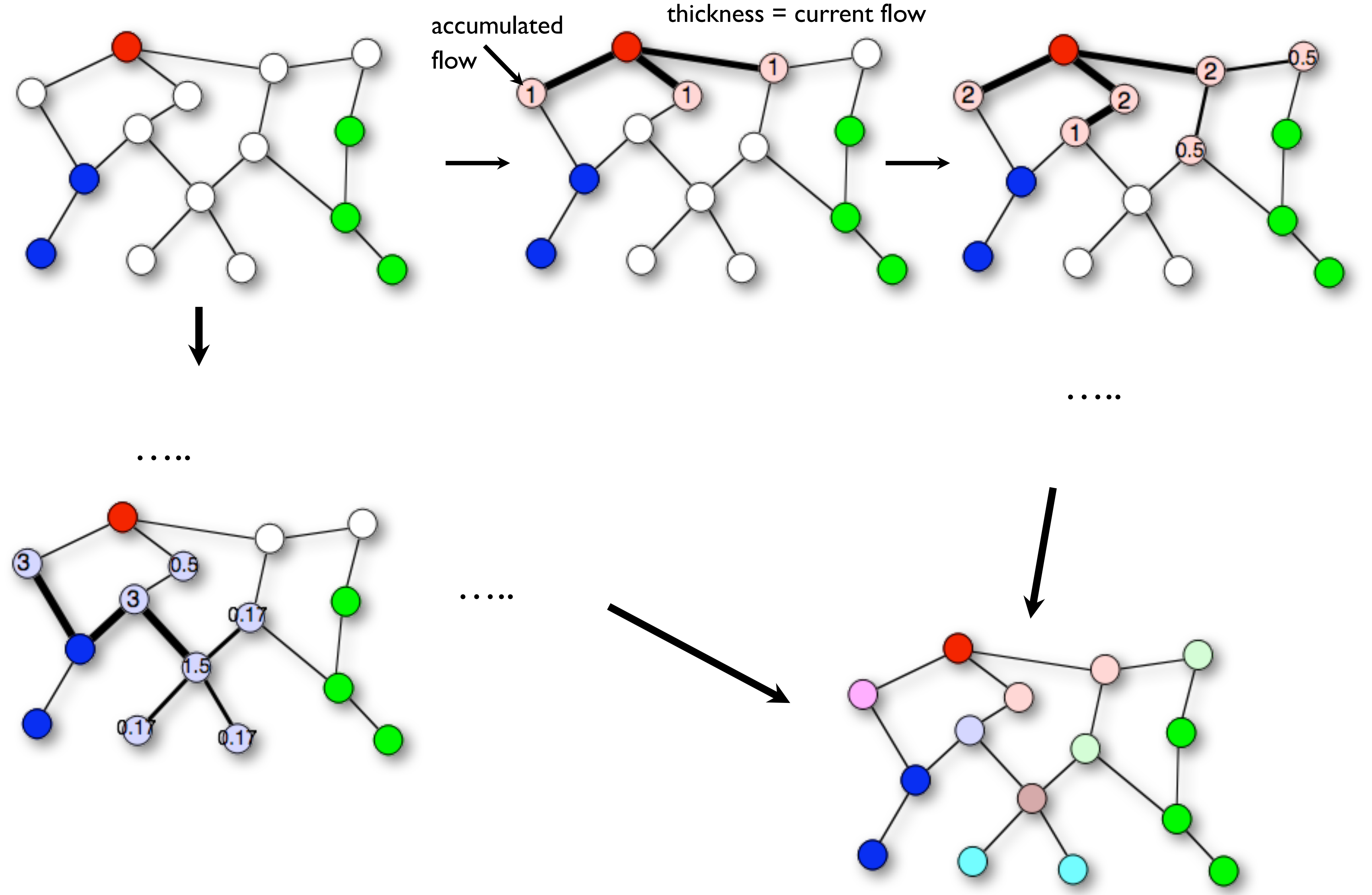
Idea: Node v has already „more function“ than node u → no flow uphill

Score from accumulated in-flow:

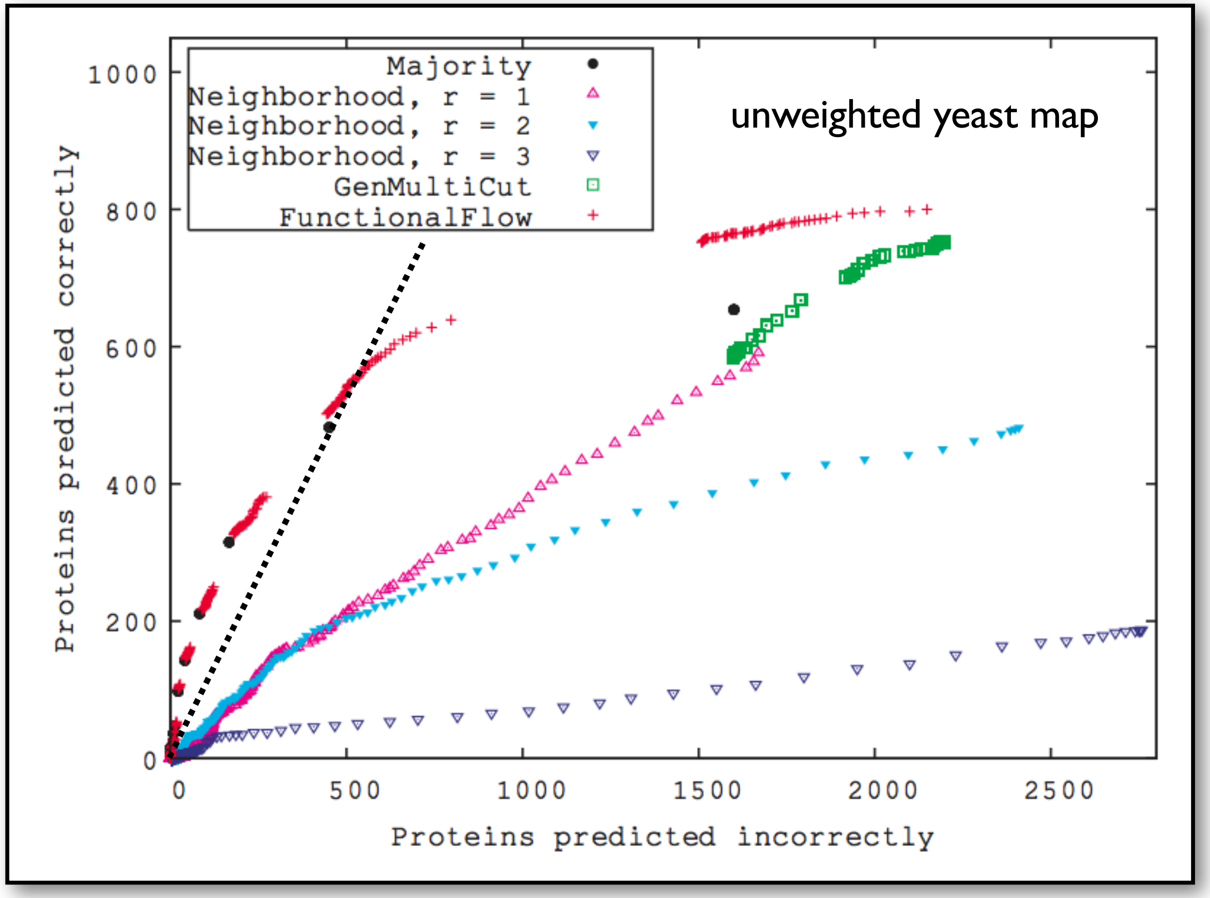
$$f_a(u) = \sum_{t=1}^{d} \sum_{v:(u,v)\in E} g_t^a(v,u)$$

Nabieva et al, Bioinformatics 21 (2005) i302

An Example


accumulated flow
thickness = current flow
1
1
1
2
2
2
1
0.5
0.5
3
3
0.5
0.17
1.5
0.17
0.17
…..
…..
…..

Comparison



For FunctionalFlow:
six propagation steps were simulated; this is comparable to the diameter of the yeast network ≈ 12

Majority results are initially very good, but method has limited coverage.

Results with neighborhood get more imprecise for larger radii r

Change **score threshold** for accepting annotations → ratio **TP/FP**
→ **FunctionalFlow** performs **best** in the high-confidence region
→ but generates still many false predictions!!!

Nabieva et al, Bioinformatics 21 (2005) i302

Going the Distance for Protein Function Prediction: A New Distance Metric for Protein Interaction Networks

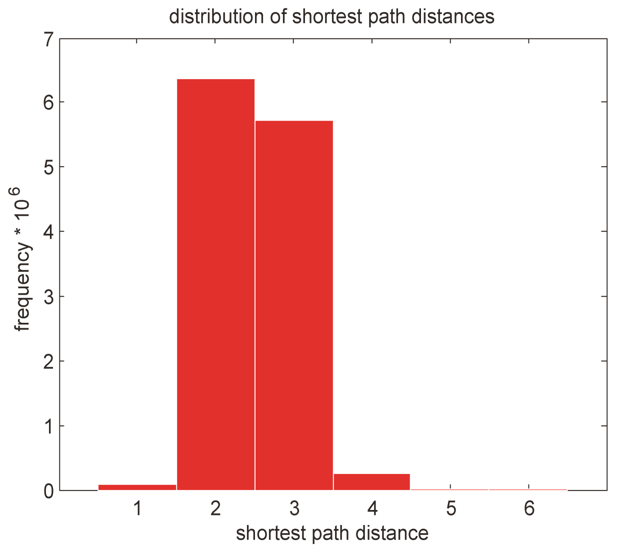
Citation: Cao M, Zhang H, Park J, Daniels NM, Crovella ME, et al. (2013) Going the Distance for Protein Function Prediction: A New Distance Metric for Protein Interaction Networks. PLoS ONE 8(10): e76339. doi:10.1371/journal.pone.0076339

Relying on the ordinary shortest-path distance metric in PPI networks is problematic because PPI networks are "small world" networks.
Most nodes are "close" to all other nodes.

$\rightarrow$ any method that infers similarity based on proximity will find that a large fraction of the network is proximate to any typical node.

Largest connected component of *S. cerevisiae* PPI network (BioGRID) has 4990 nodes and 74,310 edges (physical interactions).

Right figure shows the histogram of shortest-path lengths in this network. Over 95% of all pairs of nodes are either 2 hops or 3 hops apart



(a)

What nodes mediate short contacts?

The 2-hop neighborhood of a typical node
probably includes around half of **all nodes** in the graph.

One of the **reasons** that paths are typically short in biological networks like the PPI network is due to the **presence of hubs**.

But hub proteins often represent proteins with
different functional roles than their neighbors.

Hub proteins likely also have multiple, distinct functions.

$\rightarrow$ not all short paths provide equally strong evidence
of similar function in PPI networks.

DSD Distance Metric

Given some fixed $k > 0$, we define $He^{\{k\}}(A,B)$ to be the expected number of times that a random walk starting at A and proceeding for k steps, will visit B. If there is no ambiguity about *k*, we can drop *k*.

$$He(v_i) = (He(v_i,v_1), He(v_i,v_2), \ldots, He(v_i,v_n))$$

He(v_i) is a „random walk distance vector" of node v_i from all other nodes.

$$DSD(u,v) = ||He(u) - He(v)||_1$$ where

$||He(u) - He(v)||_1$ denotes the L_1 norm of the He vectors

Two nodes *u* and *v* have small DSD if they have similar distance from all other nodes.

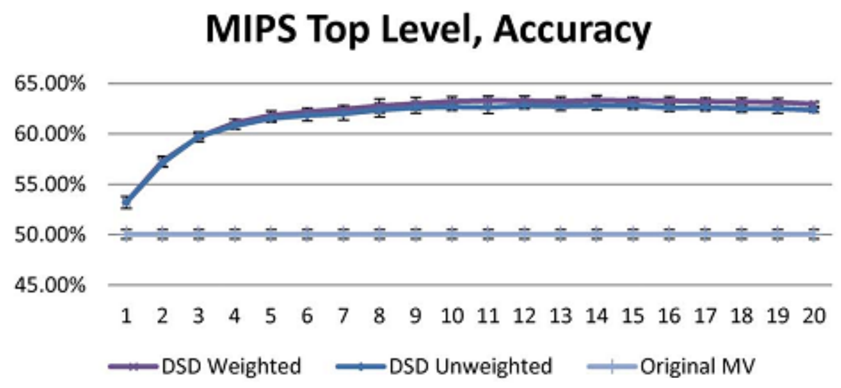
Explanation: The one-norm (also known as the L_1-norm, ℓ_1 norm, or mean norm) of a vector $\vec{v}$ is denoted $\|\vec{v}\|_1$ and is defined as the sum of the absolute values of its components:

$$\|\vec{v}\|_1 = \sum_{i=1}^{n} |v_i| \tag{1}$$

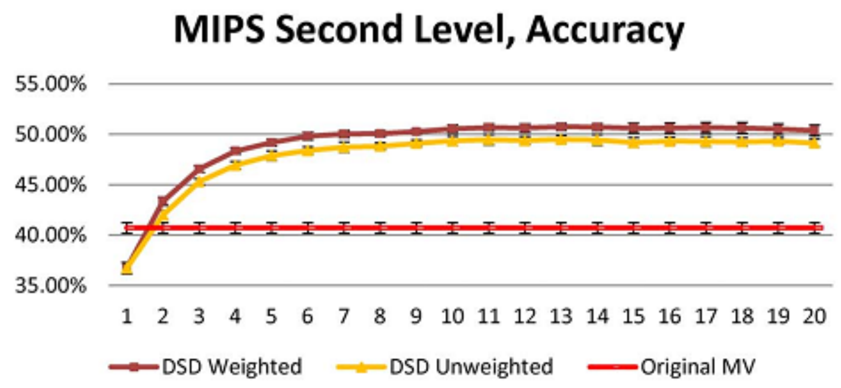
for example, given the vector $\vec{v} = (1, -4, 5)$, we calculate the one-norm:

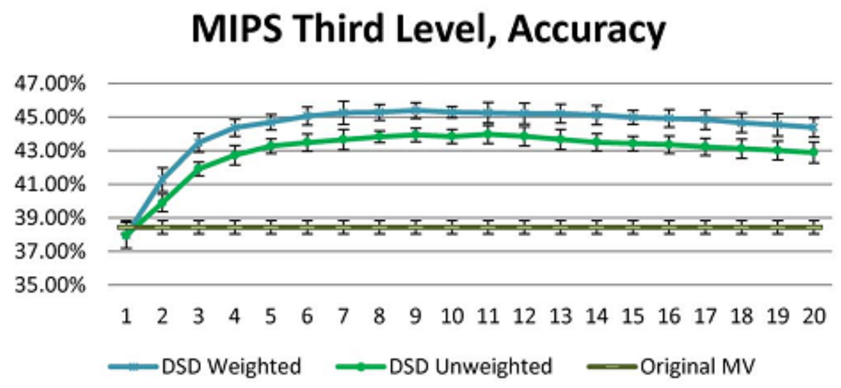
$$\|(1,-4,5)\|_1 = |1| + |-4| + |5| = 10$$

DSD clearly improves functional predictions



MV: majority voting





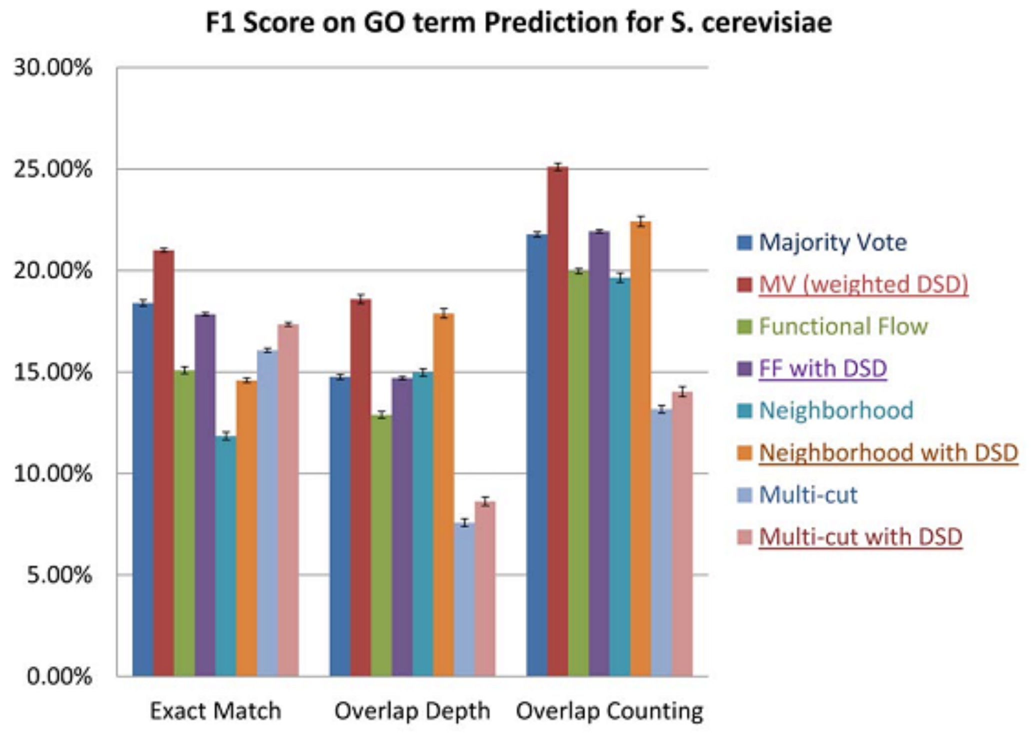


Figure 6. Improvement on F1 Score for DSD using three evaluation methods: exact match, overlap depth and overlap counting, on informative GO terms for the four algorithms for *S. cerevisiae* in 10 runs of 2-fold cross validation.

Summary

What you learned **today**:

- **Graph bisection**
  => Kernighan Lin algorithm

- **Modules** in networks
  => modular decomposition

- Postulated modes of **network evolution**
  => DMC yields networks that mimicking real networks most closely

- Predicting unknown **protein functions**
from a protein's connectivity in PPI network

V8: wrap up protein interaction networks

Then next block of the lecture: gene-regulatory networks