NETWORK CLUSTERING METHODS

Dr. Alioune Ngom
School of Computer Science
University of Windsor
angom@uwindsor.ca

Winter 2013

Why clustering?

- A cluster is a group of related objects
  - In biological nets, a group of “related” genes/proteins

- Application in PPI nets:
  - Protein function prediction
  - Protein complex identification

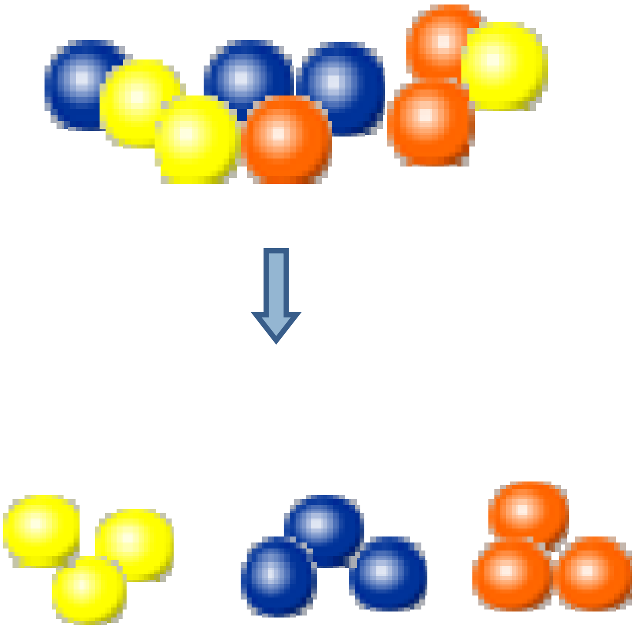
- Are you familiar with Gene Ontology?

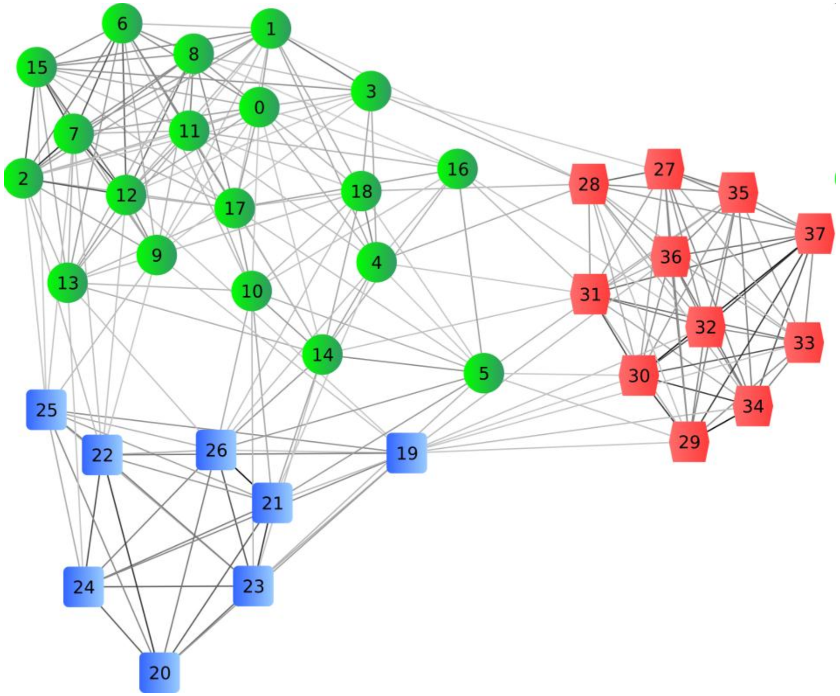
The Problem

- Clustering:
  - Group elements into subsets based on similarity between pairs of elements
- Requirements:
  - Elements in the *same* cluster are highly similar to each other
  - Elements in *different* clusters have low similarity to each other
- Challenges:
  - Large sets of data
  - Inaccurate and noisy measurements

Clustering

- Data clustering (Lecture 6) vs. Graph clustering





Graph clustering

Overlapping terminology:

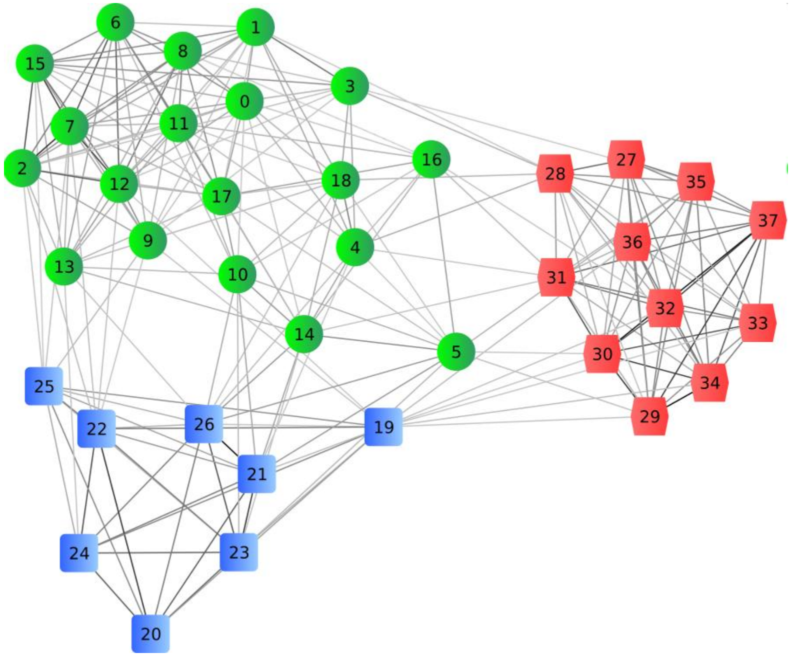
- **Clustering** algorithm for graphs =
  "**Community** detection" algorithm for networks

- Community structure in networks =
  Cluster structure in graphs

- Partitioning vs. clustering
  - Overlap?

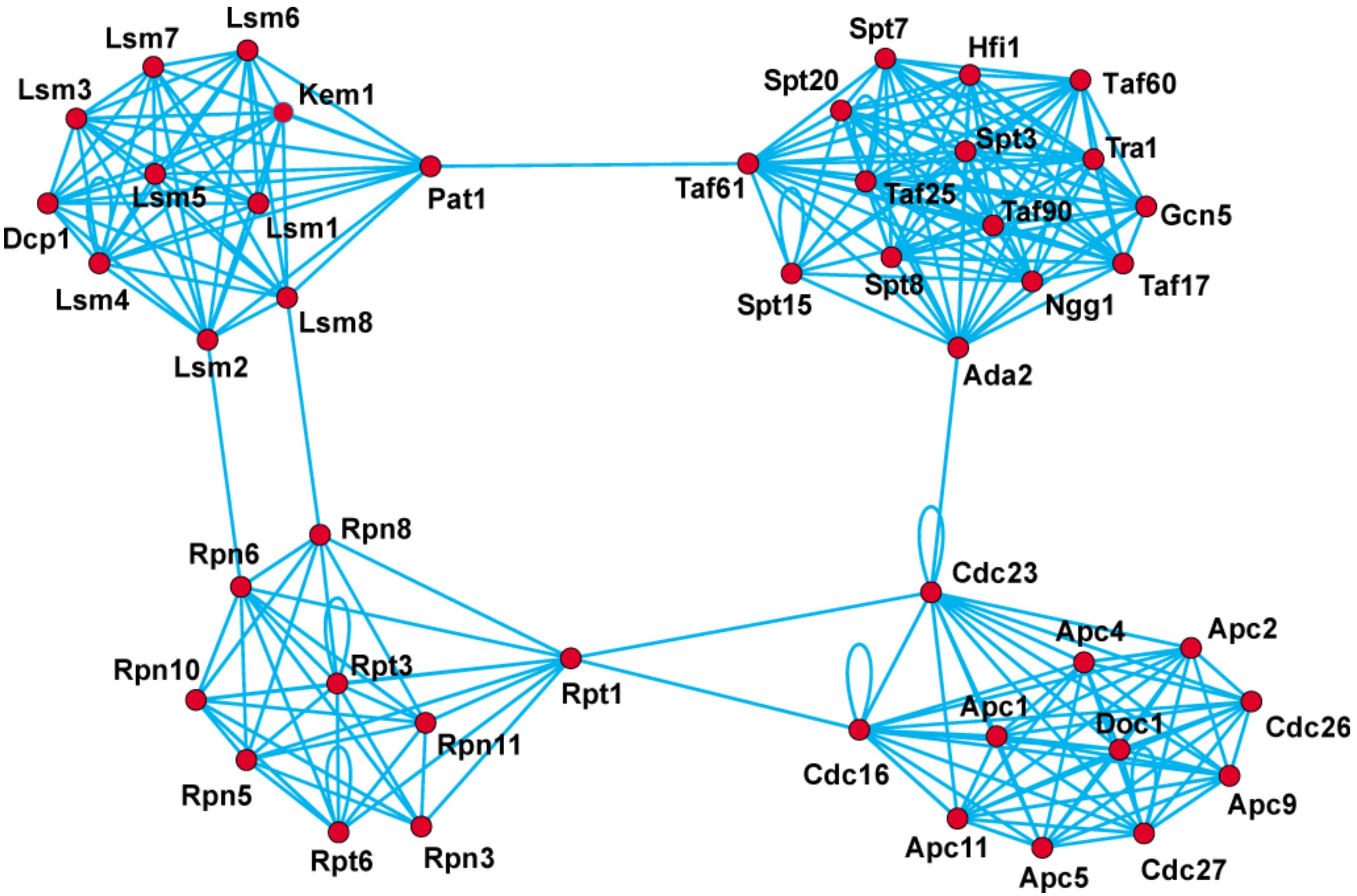
Graph clustering

- Decompose a network into subnetworks based on some topological properties
- Usually we look for *dense* subnetworks

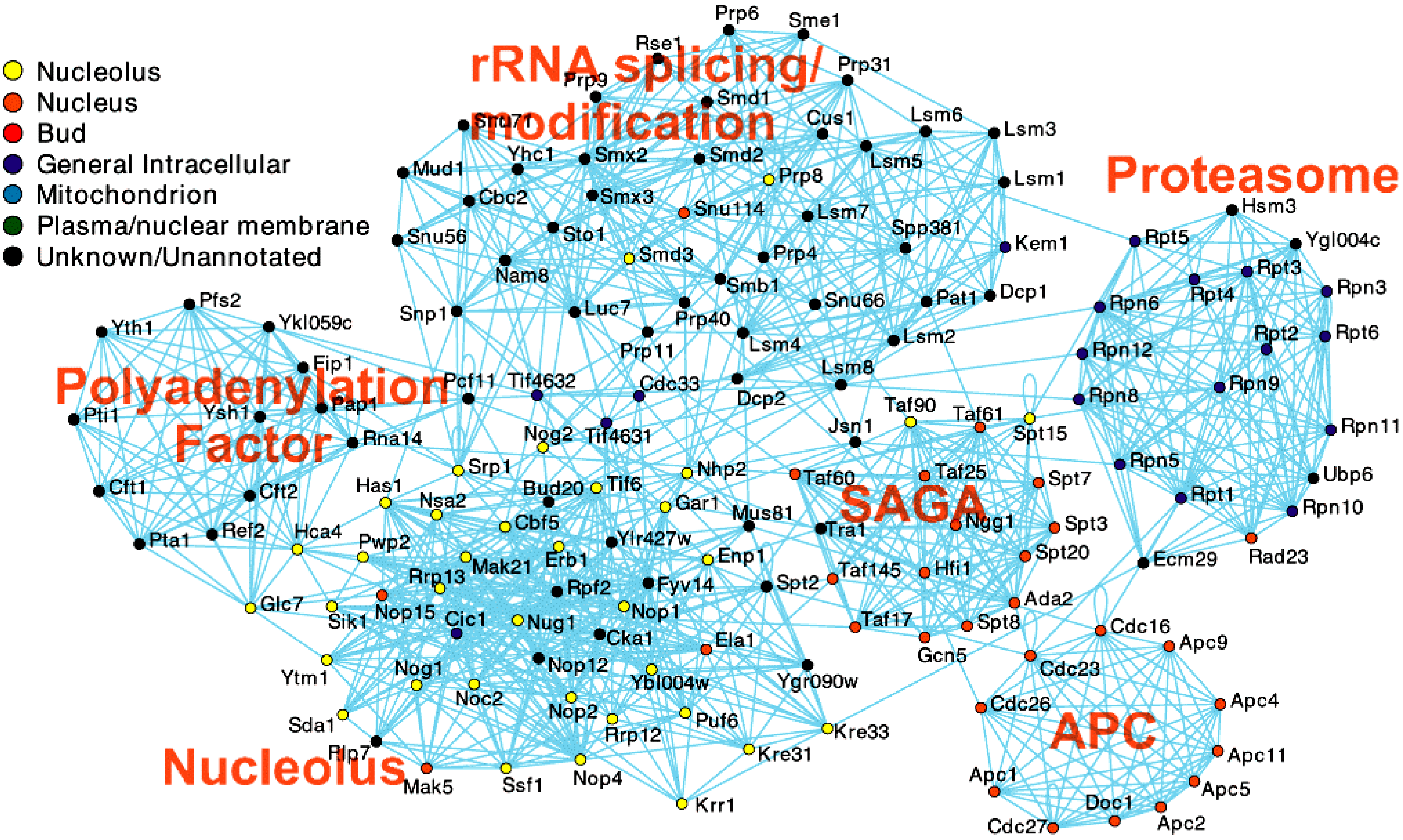


Graph clustering

- Why?
  - Protein complexes in a PPI network



E.g., Nuclear Complexes



Graph clustering

Algorithms:

- Exact: have proven solution quality and time complexity
- Approximate: heuristics are used to make them efficient

Example algorithms:

- Highly connected subgraphs (HCS)
- Restricted neighborhood search clustering (RNSC)
- Molecular Complex Detection (MCODE)
- Markov Cluster Algorithm (MCL)
- … etc

Hierarchical, k-means… clustering

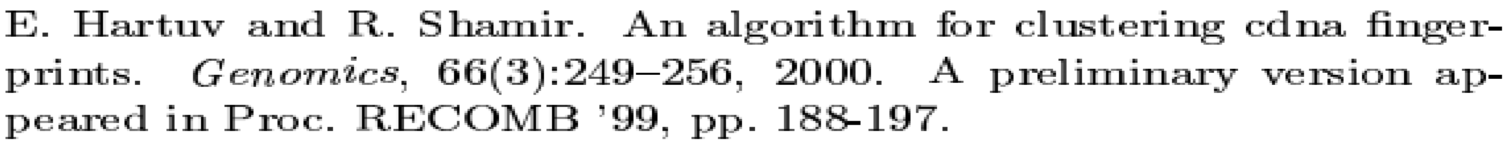
- Of course, you can always cluster data using these methods and an appropriate topological distance measure
  - Shortest path distances
    - Many ties
  - Czekanowski-Dice distance
    - Assigns the maximum distance value to two nodes having no common interactors
    - Assigns zero value to those nodes interacting with exactly the same set of neighbors
    - Form clusters of nodes sharing a high percentage of edges
  - GDV-similarity
  - Do they satisfy all of the distance metric rules?

Highly connected subgraphs (HCS)

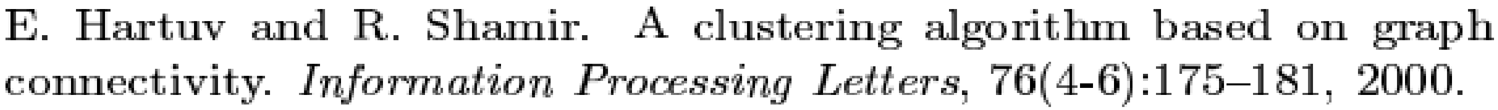
- Definitions:
  - HCS - a subgraph with n nodes such that more than n/2 edges must be removed in order to disconnect it
  - A cut in a graph - partition of vertices into two non-overlapping sets
  - A multiway cut - partition of vertices into several disjoint sets
  - The cut-set - the set of edges whose end points are in different sets
  - Edges are said to be crossing the cut if they are in its cut-set
  - The size/weight of a cut - the number of edges crossing the cut
- The HCS algorithm partitions the graph by finding the minimum graph cut and by repeating the process recursively until highly connected components (subgraphs) are found

Highly connected subgraphs (HCS)

- HCS algorithm:
  - ○ Input: graph G
  - ○ Does G satisfy a stopping criterion?
    - ○ If yes: it is declared a “kernel”
    - ○ Otherwise, G is partitioned into two subgraphs, separated by a minimum weight edge cut
    - ○ Recursively proceed on the two subgraphs
  - ○ Output: list of kernels that are basis of possible clusters

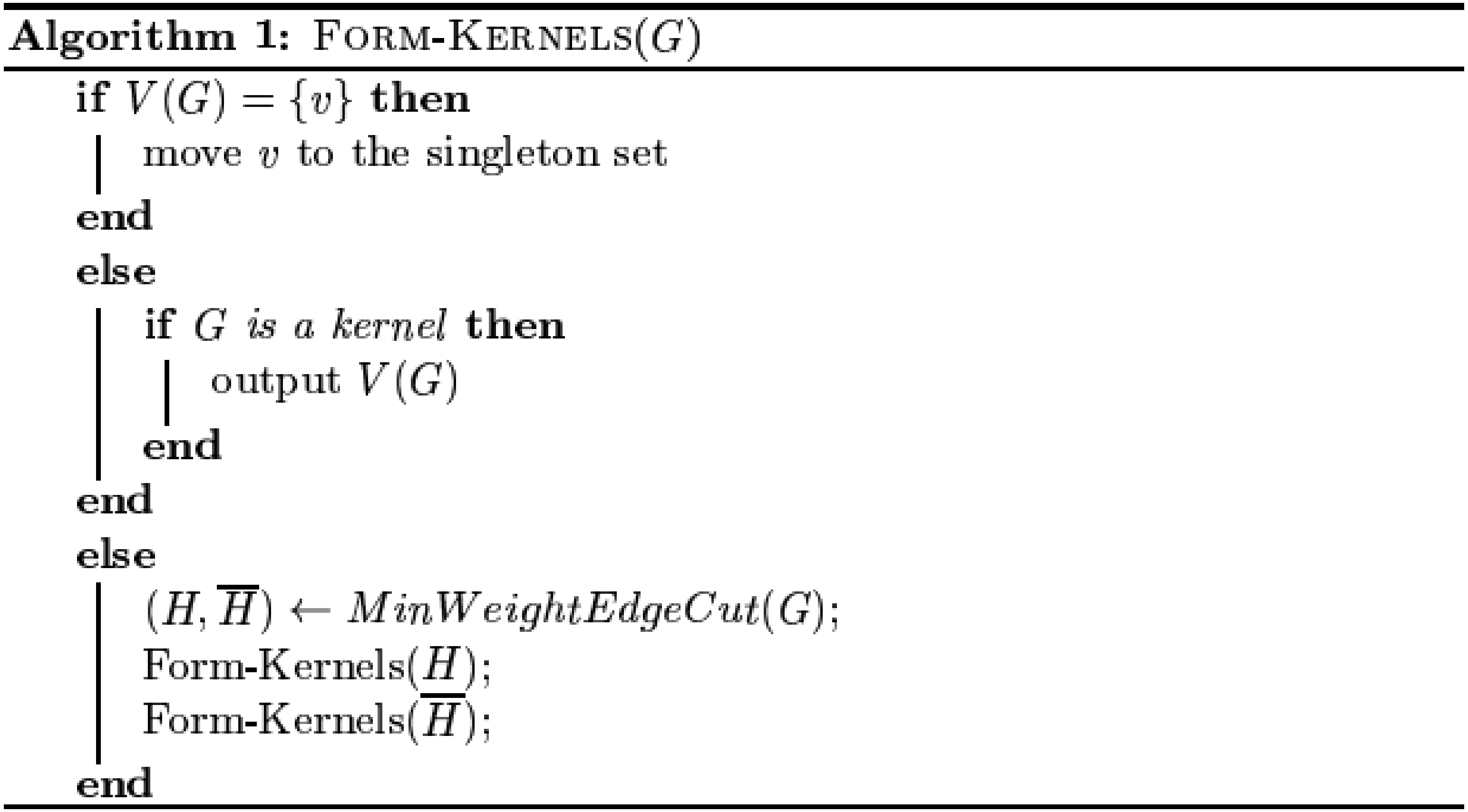


E. Hartuv and R. Shamir. An algorithm for clustering cdna fingerprints. *Genomics*, 66(3):249–256, 2000. A preliminary version appeared in Proc. RECOMB ’99, pp. 188-197.



E. Hartuv and R. Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(4-6):175–181, 2000.

Highly connected subgraphs (HCS)



Algorithm 1: Form-Kernels(G)

```
if $V(G) = \{v\}$ then
    move $v$ to the singleton set
end
else
    if $G$ is a kernel then
        output $V(G)$
    end
end
else
    $(H, \overline{H}) \leftarrow MinWeightEdgeCut(G)$;
    Form-Kernels($H$);
    Form-Kernels($\overline{H}$);
end
```

Highly connected subgraphs (HCS)

- Clusters satisfy two properties:
  - They are homogeneous, since the diameter of each cluster is at most 2 and each cluster is at least half as dense as clique
  - They are well separated, since any non-trivial split by the algorithm happens on subgraphs that are likely to be of diameter at least 3

- Running time complexity of HCS algorithm:
  - Bounded by 2N f(n,m)
  - N is the number of clusters found (often N << n)
  - f(n,m) is time complexity of computing a minimum edge cut of G with n nodes and m edges
  - The fastest deterministic min edge cut alg. for *unweighted* graphs has time complexity O(nm); for *weighted* graphs it's O(nm+n^{2}log n)

More in survey chapter: N. Przulj, "Graph Theory Analysis of Protein-Protein Interactions," a chapter in "Knowledge Discovery in Proteomics," edited by I. Jurisica and D. Wigle, CRC Press, 2005

Highly connected subgraphs (HCS)

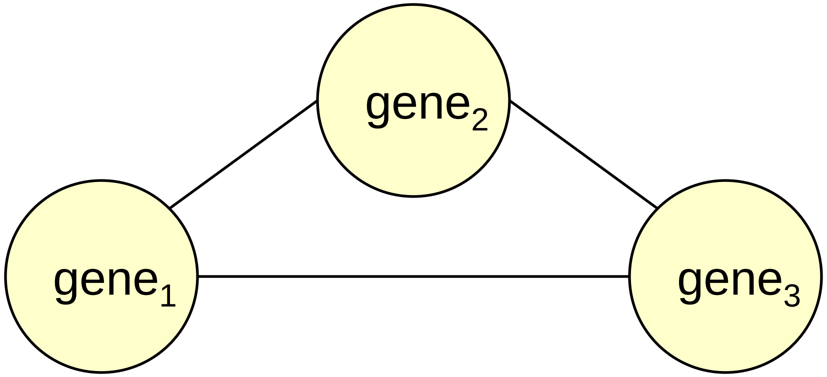
- Several heuristics used to speed it up
- E.g., *removing low degree nodes*
  - If an input graph has many low degree nodes (remember, bio nets have power-law degree distributions), one iteration of the minimum edge cut algorithm many only separate a low degree node from the rest of the graph contributing to increased computational cost at a low informative value in terms of clustering
  - After clustering is over, singletons can be “adopted” by clusters, say by the cluster with which a singleton node has the most neighbors

HCS Algorithm Overview

- **H**ighly **C**onnected **S**ubgraphs Algorithm
  - Uses graph theoretic techniques
- Basic Idea
  - Uses similarity information to construct a similarity graph
  - Groups elements that are highly connected with each other

HCS: Main Players

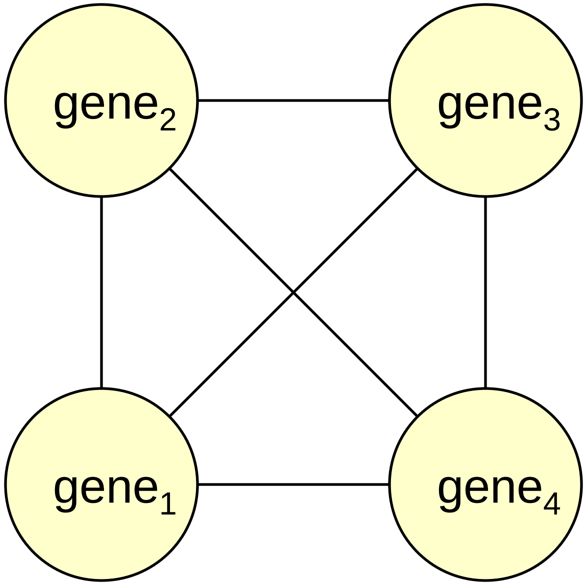
- Similarity Graph
  - Nodes correspond to elements (genes)
  - Edges connect similar elements (those whose similarity value is above some threshold)



Gene_1 similar to gene_2
Gene_1 similar to gene_3
Gene_2 similar to gene_3

HCS: Main Players

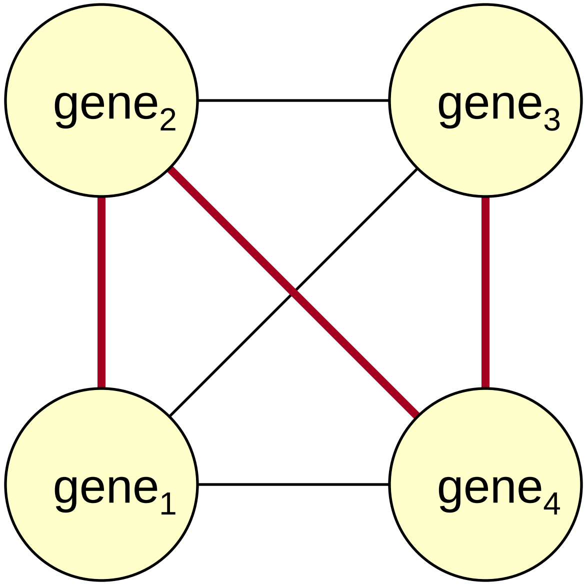
- Edge Connectivity
  - Minimum number of edges whose removal results in a disconnected graph



Must remove 3 edges to disconnect graph, thus has an edge connectivity $k(G) = 3$

HCS: Main Players

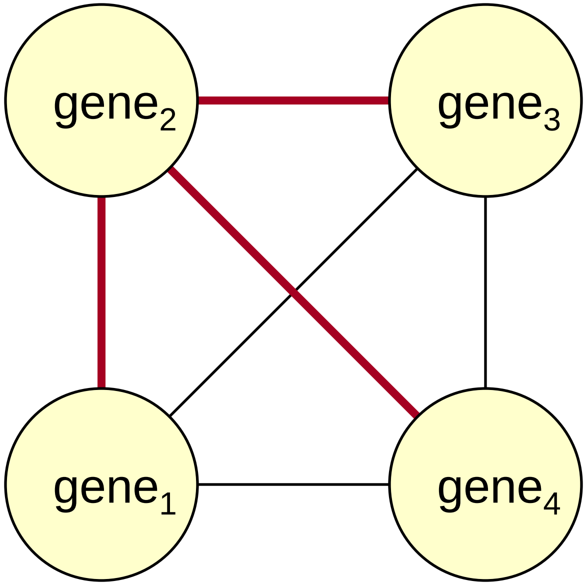
- Edge Connectivity
  - Minimum number of edges whose removal results in a disconnected graph



Must remove 3 edges to disconnect graph, thus has an edge connectivity $k(G) = 3$

HCS: Main Players

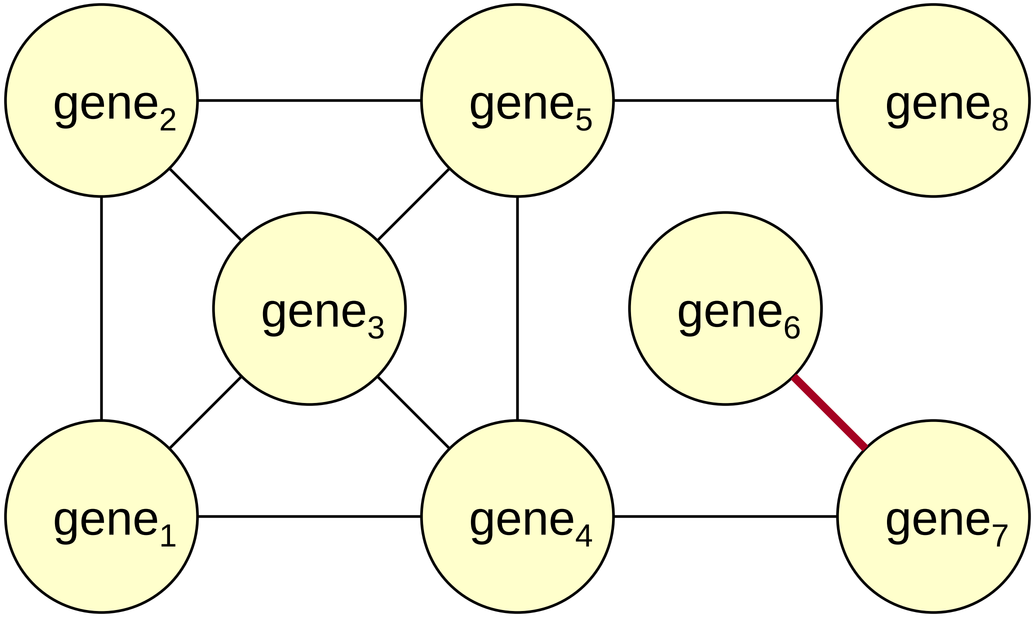
- Edge Connectivity
  - Minimum number of edges whose removal results in a disconnected graph



Must remove 3 edges to disconnect graph, thus has an edge connectivity $k(G) = 3$

HCS: Main Players

- Highly Connected Subgraphs
  - Subgraphs whose edge connectivity exceeds half the number of nodes

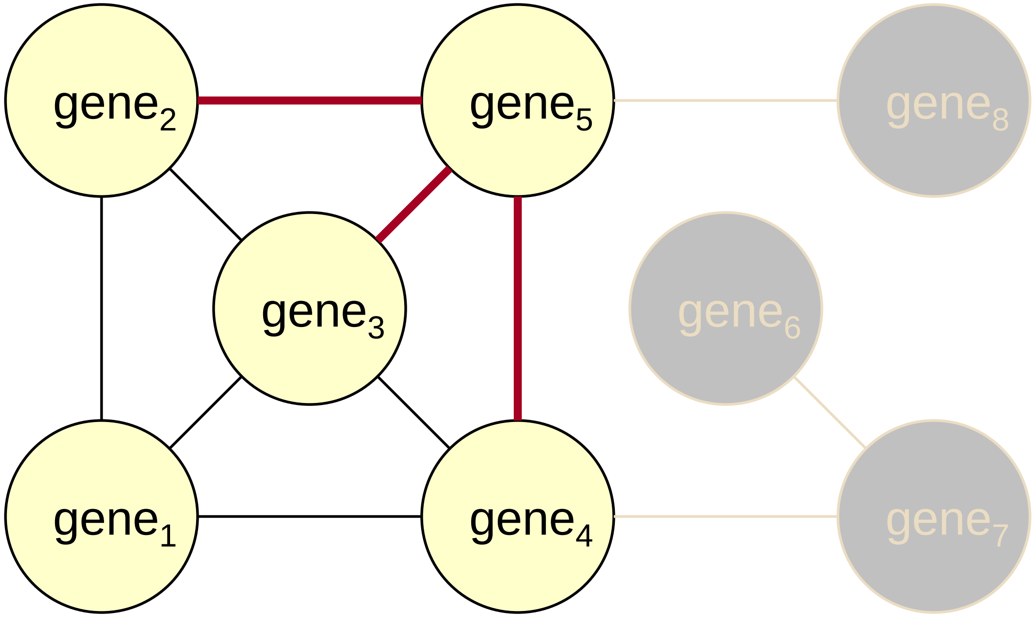


Entire Graph
Nodes = 8
Edge connectivity = 1

Not HCS!

HCS: Main Players

- Highly Connected Subgraphs
  - Subgraphs whose edge connectivity exceeds half the number of nodes

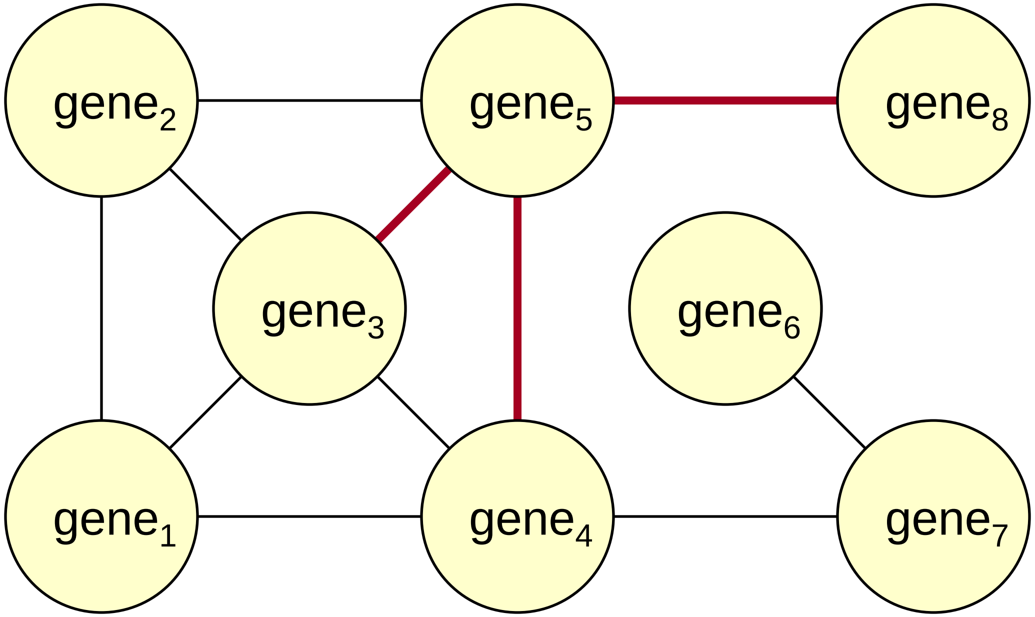


Sub Graph
Nodes = 5
Edge connectivity = 3

HCS!

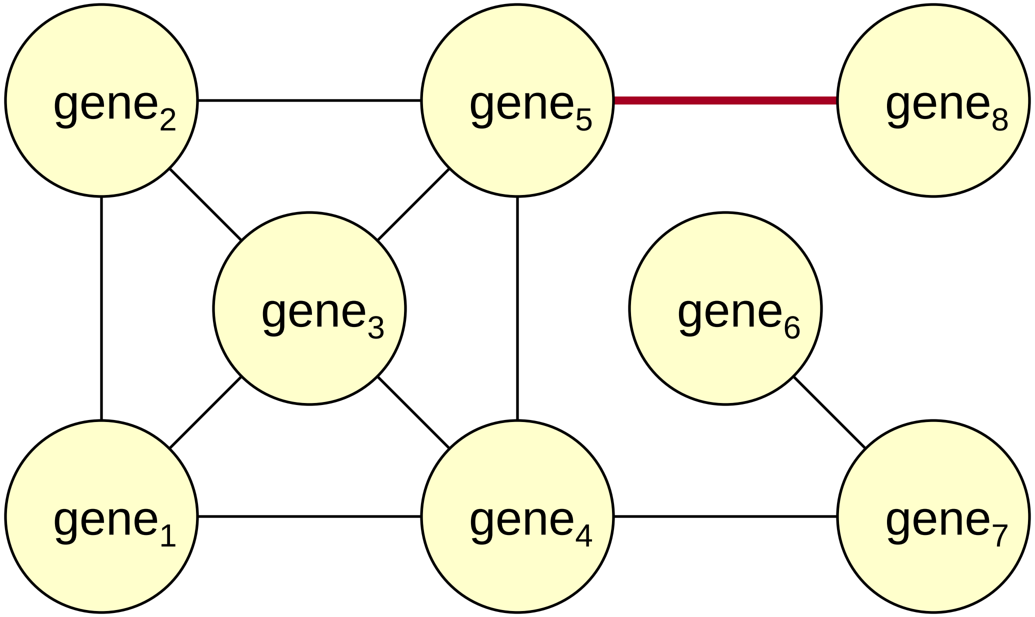
HCS: Main Players

- Cut
  - A set of edges whose removal disconnects the graph



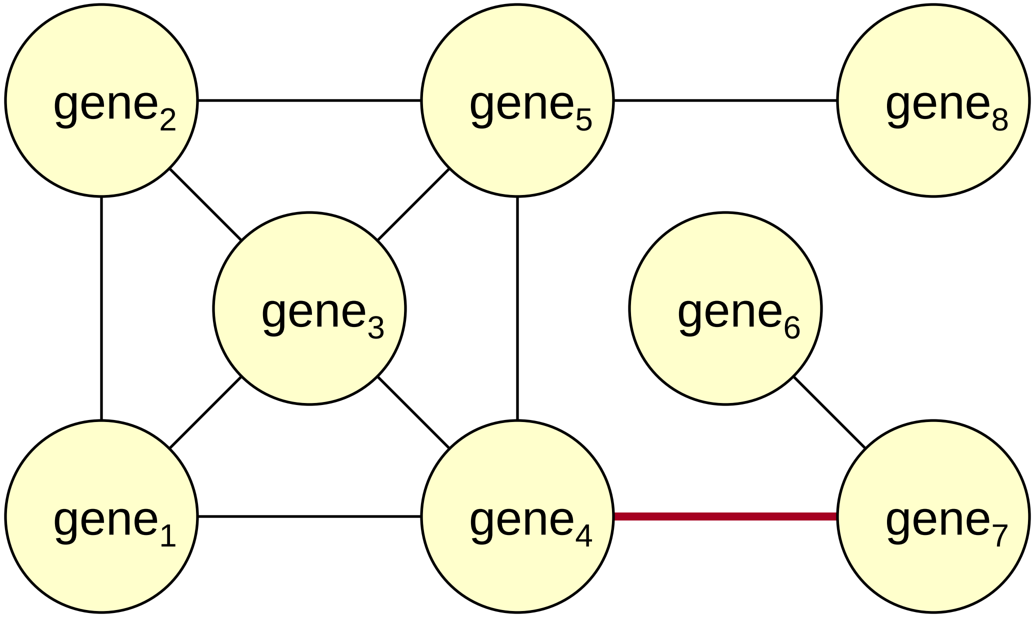
HCS: Main Players

- Minimum Cut
  - A cut with a *minimum* number of edges



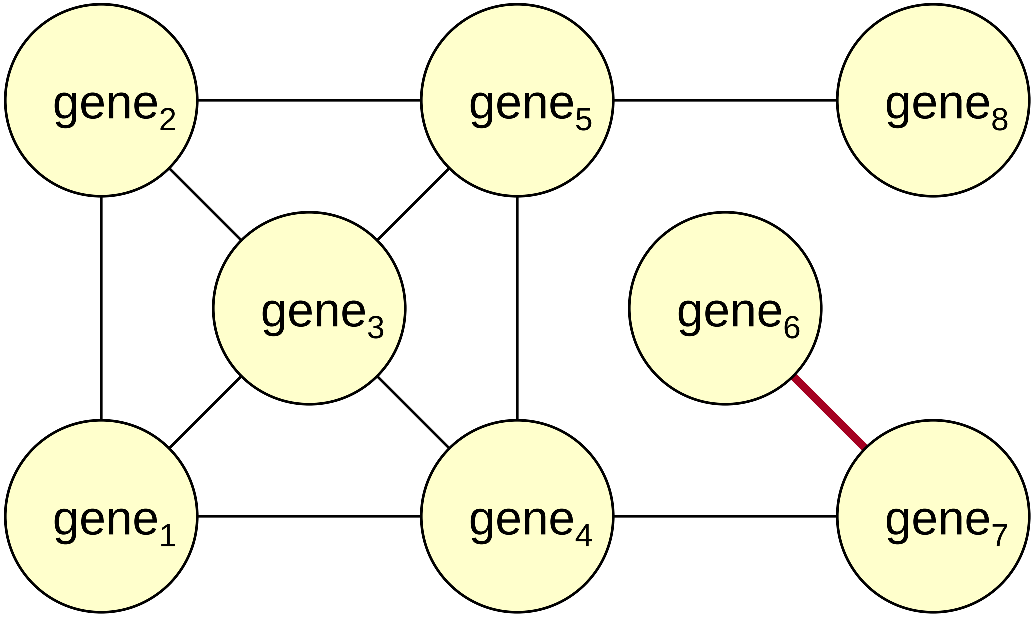
HCS: Main Players

- Minimum Cut
  - A cut with a *minimum* number of edges

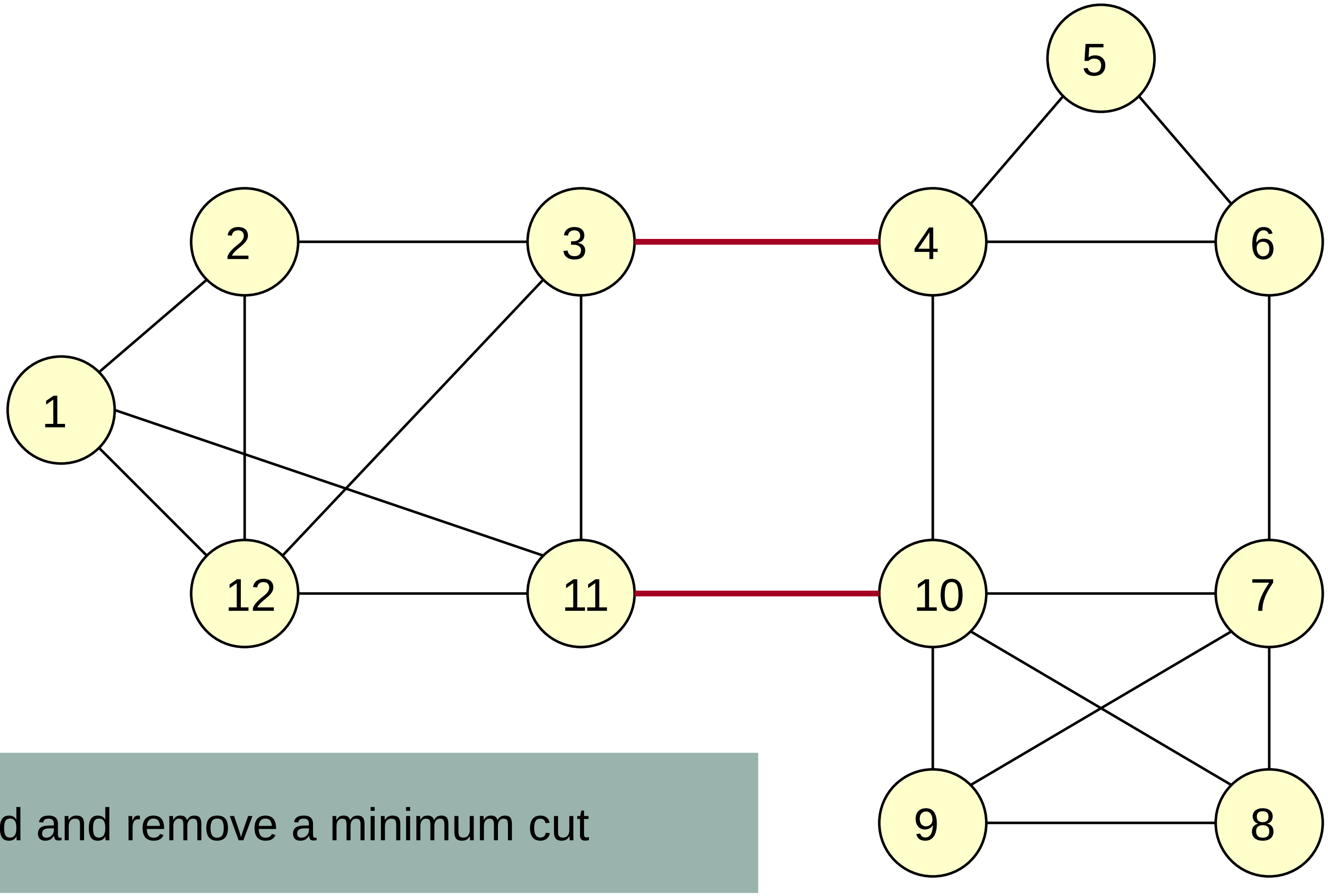


HCS: Main Players

- Minimum Cut
  - A cut with a *minimum* number of edges

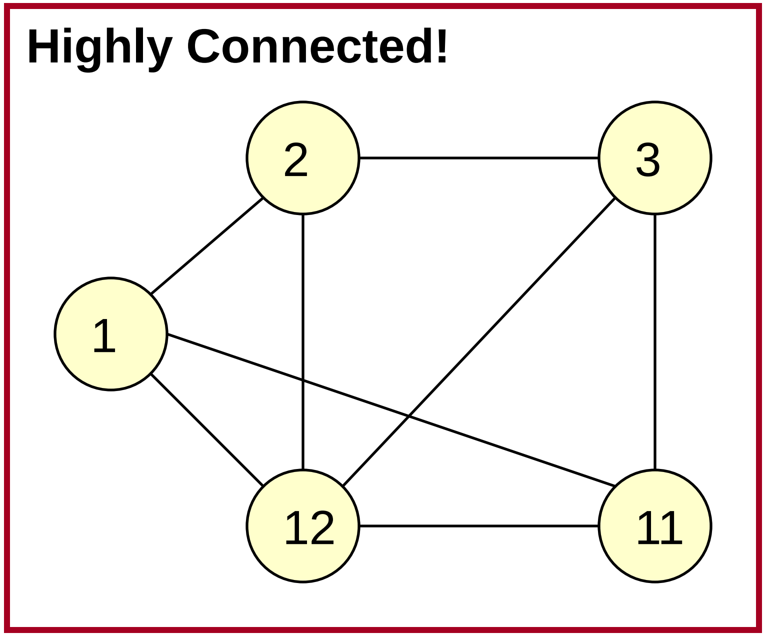


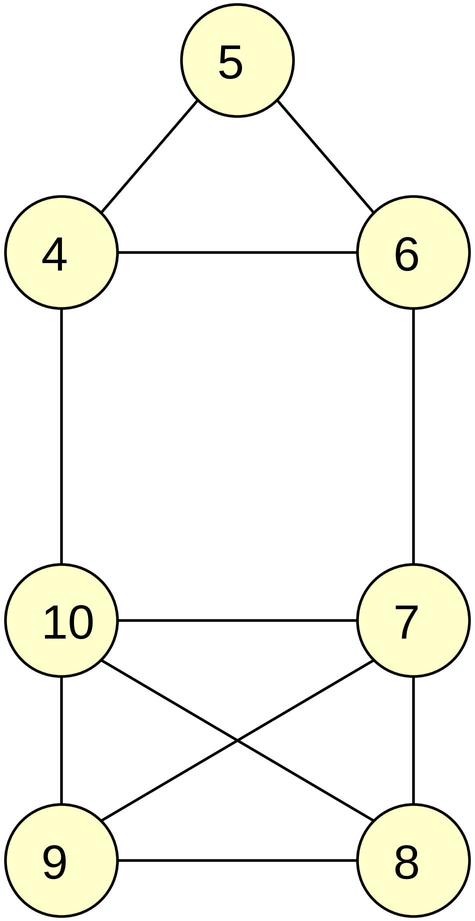
HCS: Algorithm (by example)



find and remove a minimum cut

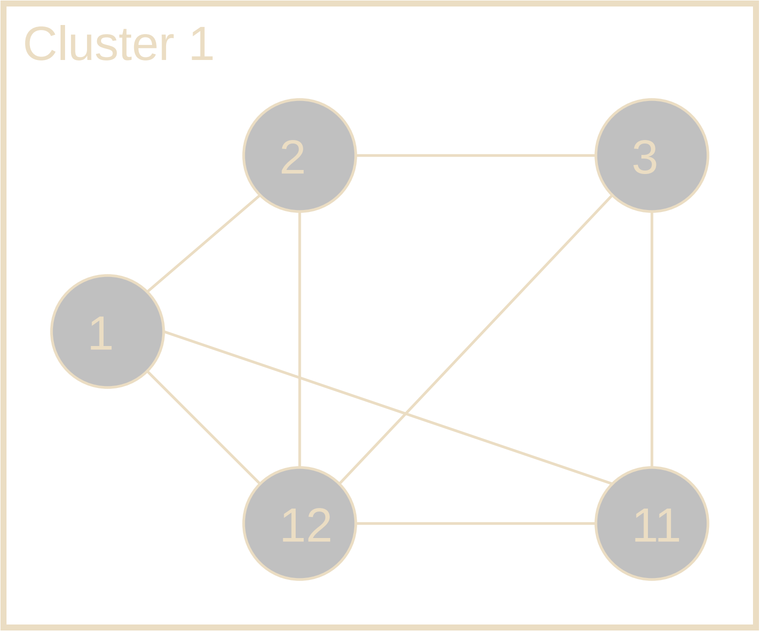
HCS: Algorithm (by example)

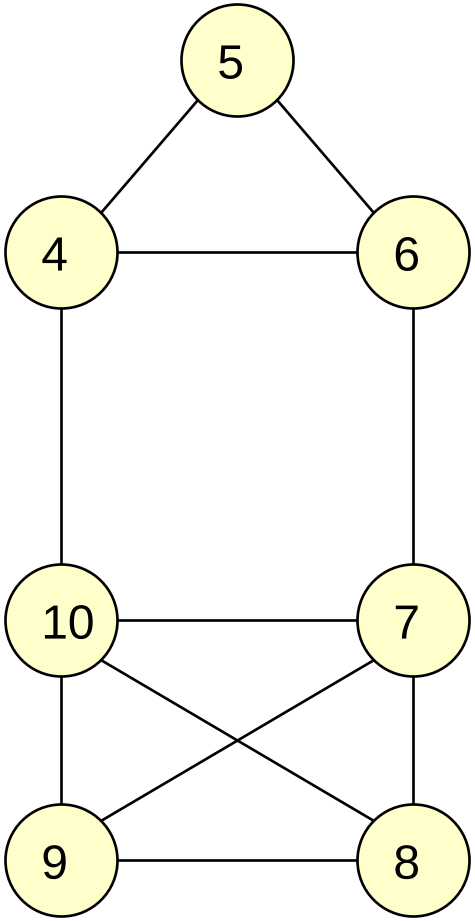




are the resulting
subgraphs highly connected?

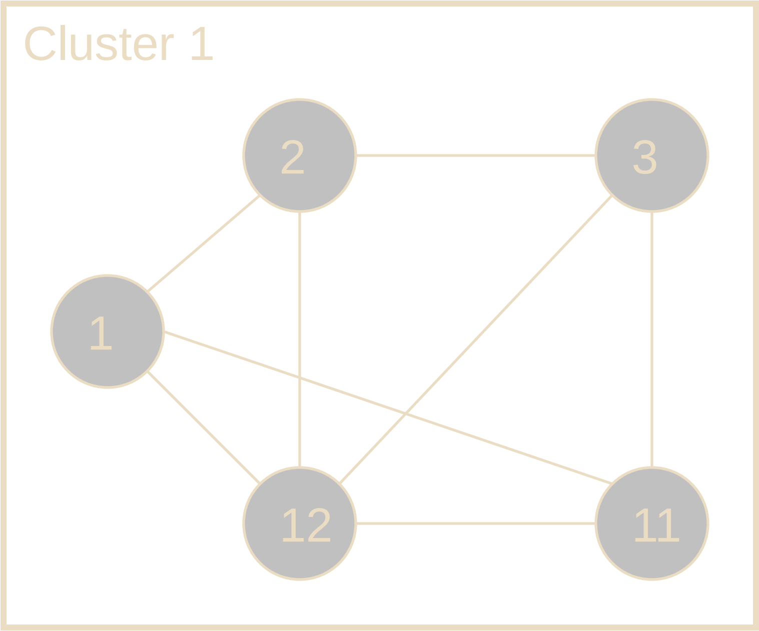
HCS: Algorithm (by example)

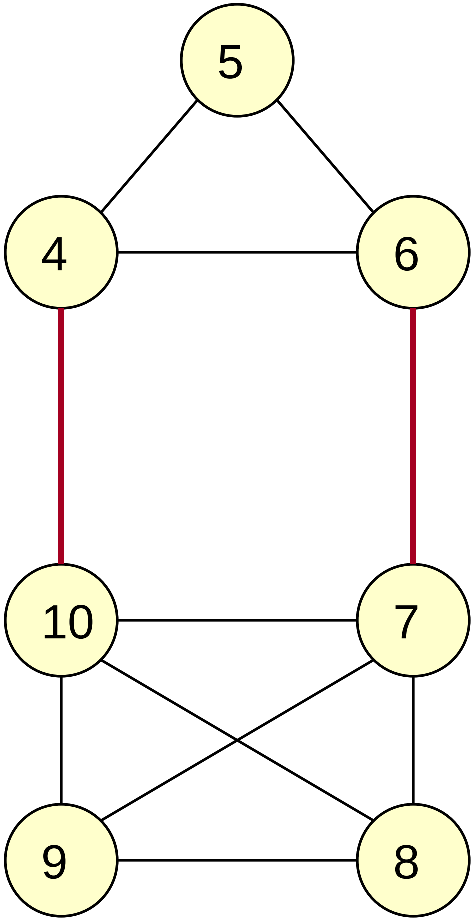




repeat process on non-highly connected subgraphs

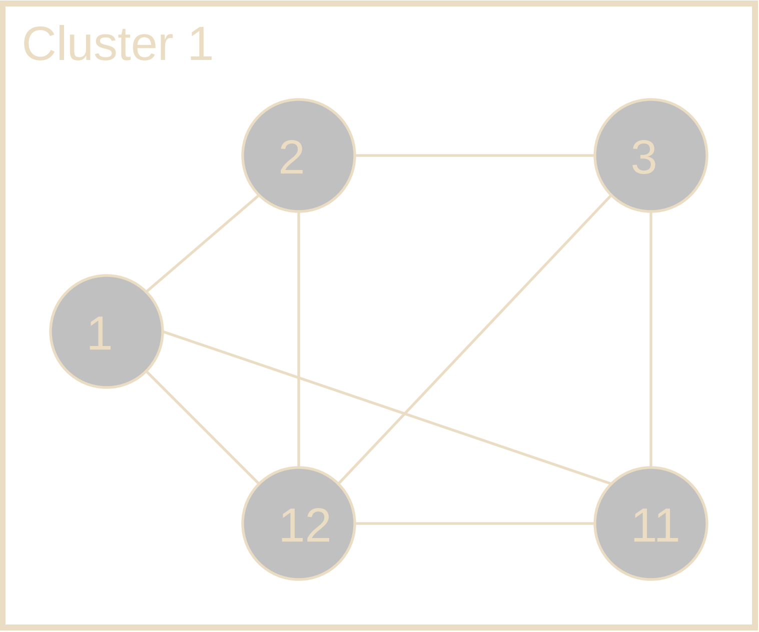
HCS: Algorithm (by example)

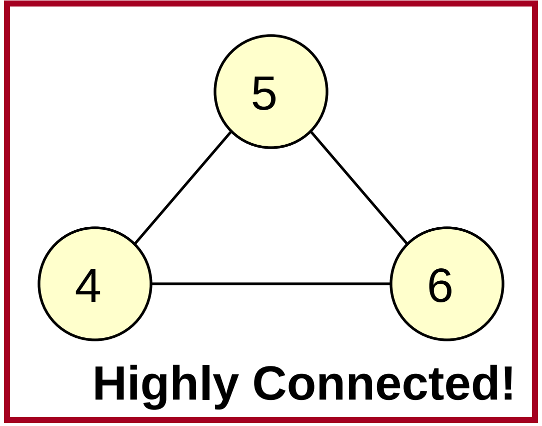




find and remove a minimum cut

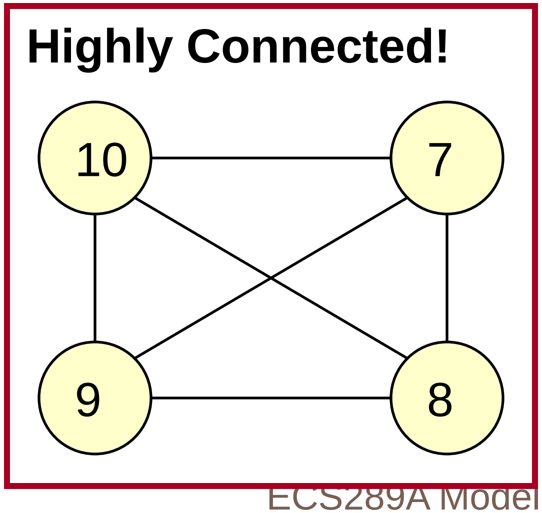
HCS: Algorithm (by example)





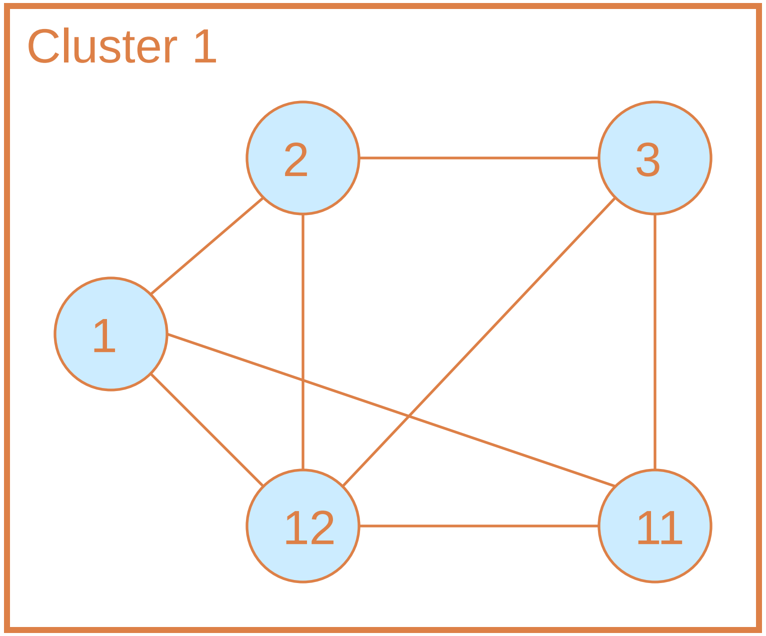
Highly Connected!

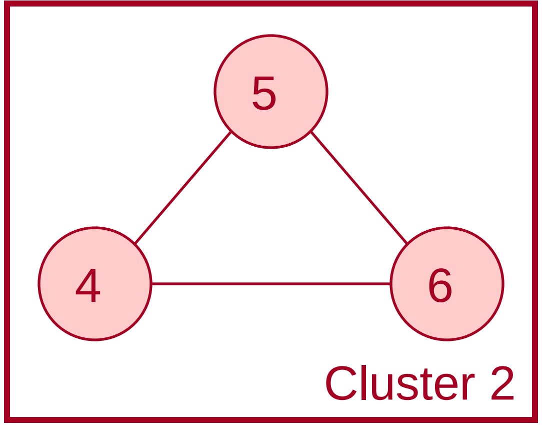
Highly Connected!

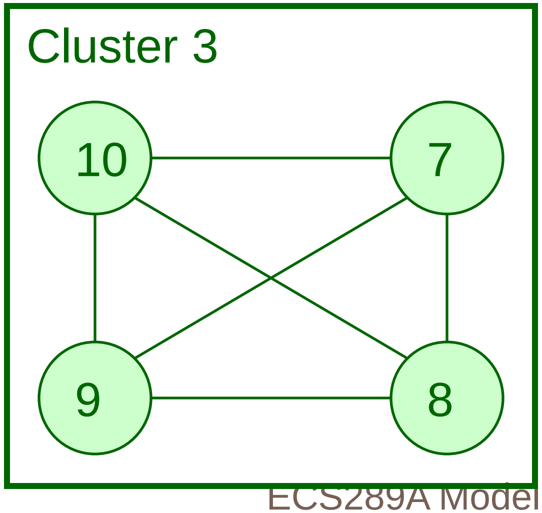


are the resulting
subgraphs highly connected?

HCS: Algorithm (by example)







resulting clusters

HCS: Algorithm

```
HCS( G )
{
    MINCUT( G ) = { $H_1$, … , $H_t$ }
    for each $H_i$, i = [ 1, t ]
  {
            if k( $H_i$ ) > n ÷ 2
                  return $H_i$
            else
                  HCS( $H_i$ )
    }
}
```

HCS: Algorithm

```
HCS( G )
{
    MINCUT( G ) = { $H_1$, … , $H_t$ }
    for each $H_i$, i = [ 1, t ]
    {
        if k( $H_i$ ) > n ÷ 2
            return $H_i$
        else
            HCS( $H_i$ )
    }
}
```

Find a minimum cut in graph G. This returns a set of subgraphs { H_1, … , H_t } resulting from the removal of the cut set.

HCS: Algorithm

```
HCS( G )
{
    MINCUT( G ) = { $H_1$, … , $H_t$ }
    for each $H_i$, i = [ 1, t ]
    {
            if k( $H_i$ ) > n ÷ 2
                return $H_i$
            else
                HCS( $H_i$ )
    }
}
```

For each subgraph…

HCS: Algorithm

```
HCS( G )
{
    MINCUT( G ) = { $H_1$, … , $H_t$ }
    for each $H_i$, i = [ 1, t ]
{
            if k( $H_i$ ) > n ÷ 2
                return $H_i$
            else
                HCS( $H_i$ )
    }
}
```

If the subgraph is highly connected, then return that subgraph as a cluster. (Note: k(H_i) denotes edge connectivity of graph H_i, n denotes number of nodes)

HCS: Algorithm

```
HCS( G )
{
    MINCUT( G ) = { $H_1$, … , $H_t$ }
    for each $H_i$, i = [ 1, t ]
    {
            if k( $H_i$ ) > n ÷ 2
                return $H_i$
            else
                HCS( $H_i$ )
    }
}
```

Otherwise, repeat the algorithm on the subgraph. (recursive function)

This continues until there are no more subgraphs, and all clusters have been found.

HCS: Algorithm

```
HCS( G )
{
    MINCUT( G ) = { $H_1$, … , $H_t$ }
    for each $H_i$, i = [ 1, t ]
    {
        if k( $H_i$ ) > n ÷ 2
            return $H_i$
        else
            HCS( $H_i$ )
    }
}
```

Running time is bounded by `2N × f( n, m )` where `N` is the number of clusters found, and `f( n, m )` is the time complexity of computing a minimum cut in a graph with `n` nodes and `m` edges.

HCS: Algorithm

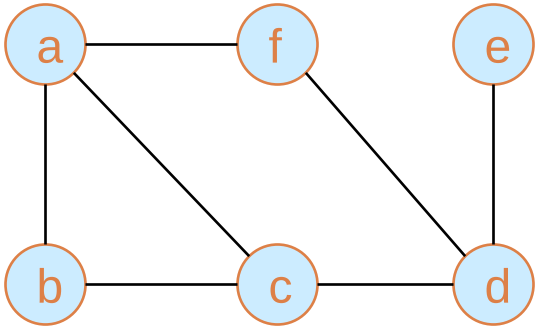
```
HCS( G ) {
    MINCUT( G ) = { $H_1$, … , $H_t$ }

    for each $H_i$, i = [ 1, t ] {
        if k( $H_i$
            retu
        else
            HCS(
    }
}
```

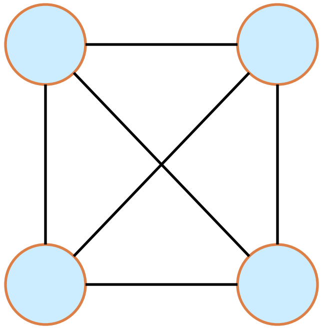
Deterministic for Un-weighted Graph: takes `O(nm)` steps where n is the number of nodes and m is the number of edges

HCS: Properties

- Homogeneity
  - Each cluster has a diameter of at most 2
    - *Distance* is the minimum length path between two nodes
      - Determined by number of EDGES traveled between nodes
    - *Diameter* is the longest distance in the graph
  - Each cluster is at least half as dense as a clique
    - Clique is a graph with maximum possible edge connectivity



Dist(a, d) = 2
Dist(a, e) = 3
Diam(G) = 4

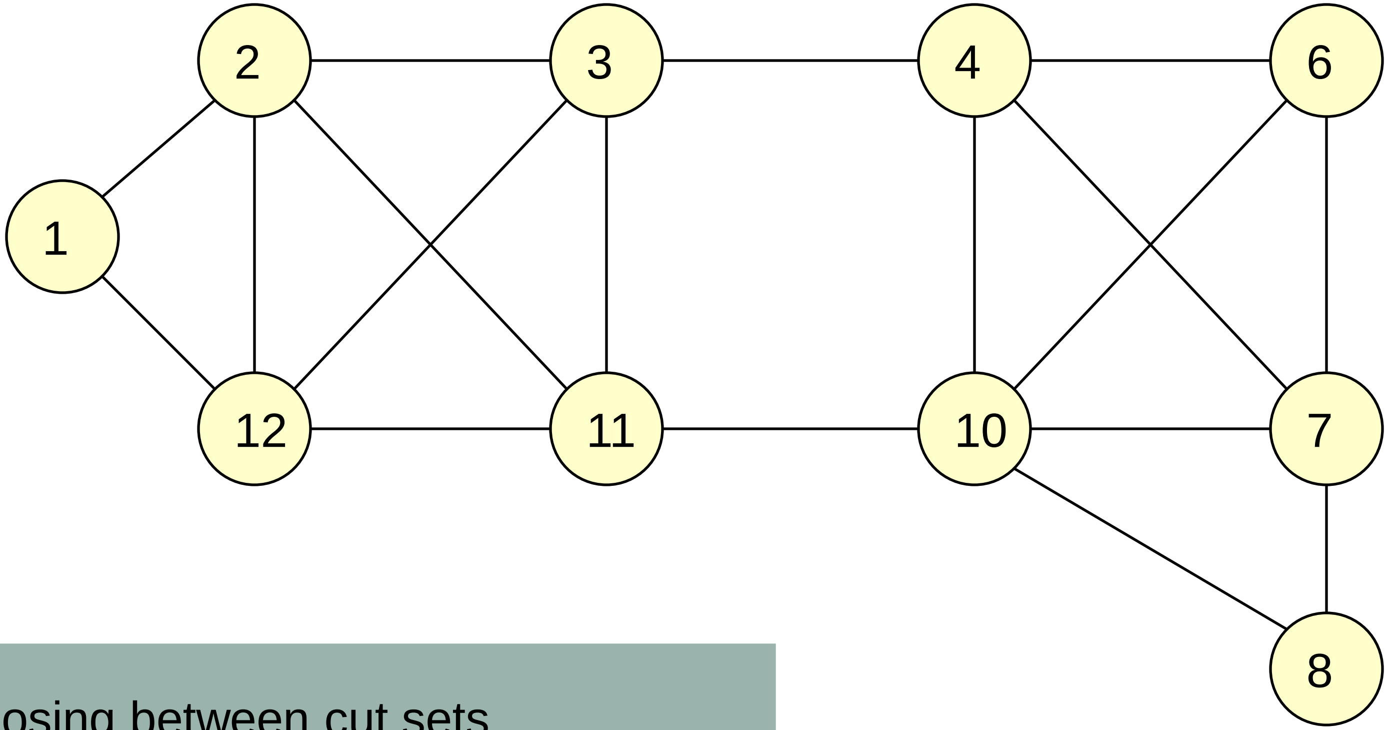


clique

HCS: Properties

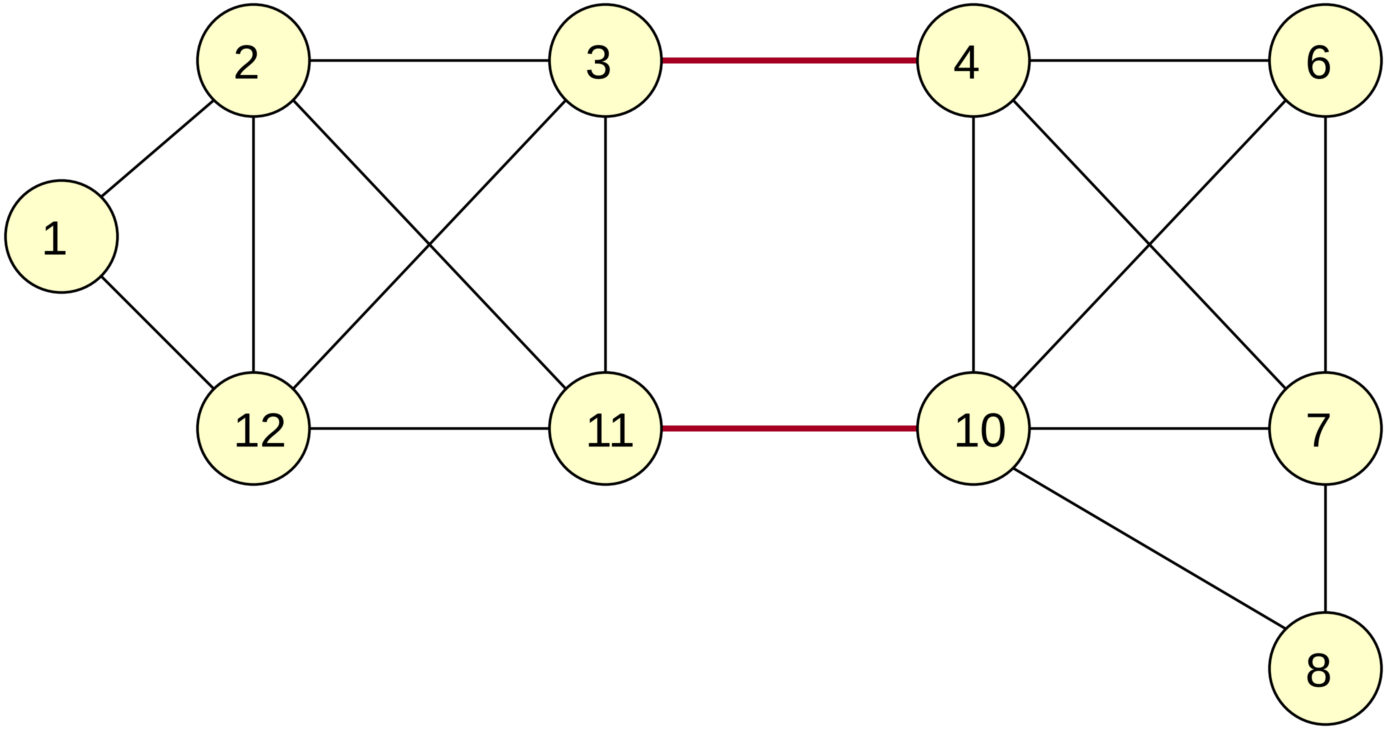
- Separation
  - Any non-trivial split is unlikely to have diameter of two
  - Number of edges removed by each iteration is linear in the size of the underlying subgraph
    - Compared to quadratic number of edges within final clusters
    - Indicates separation unless sizes are small
    - Does not imply number of edges removed overall

HCS: Improvements

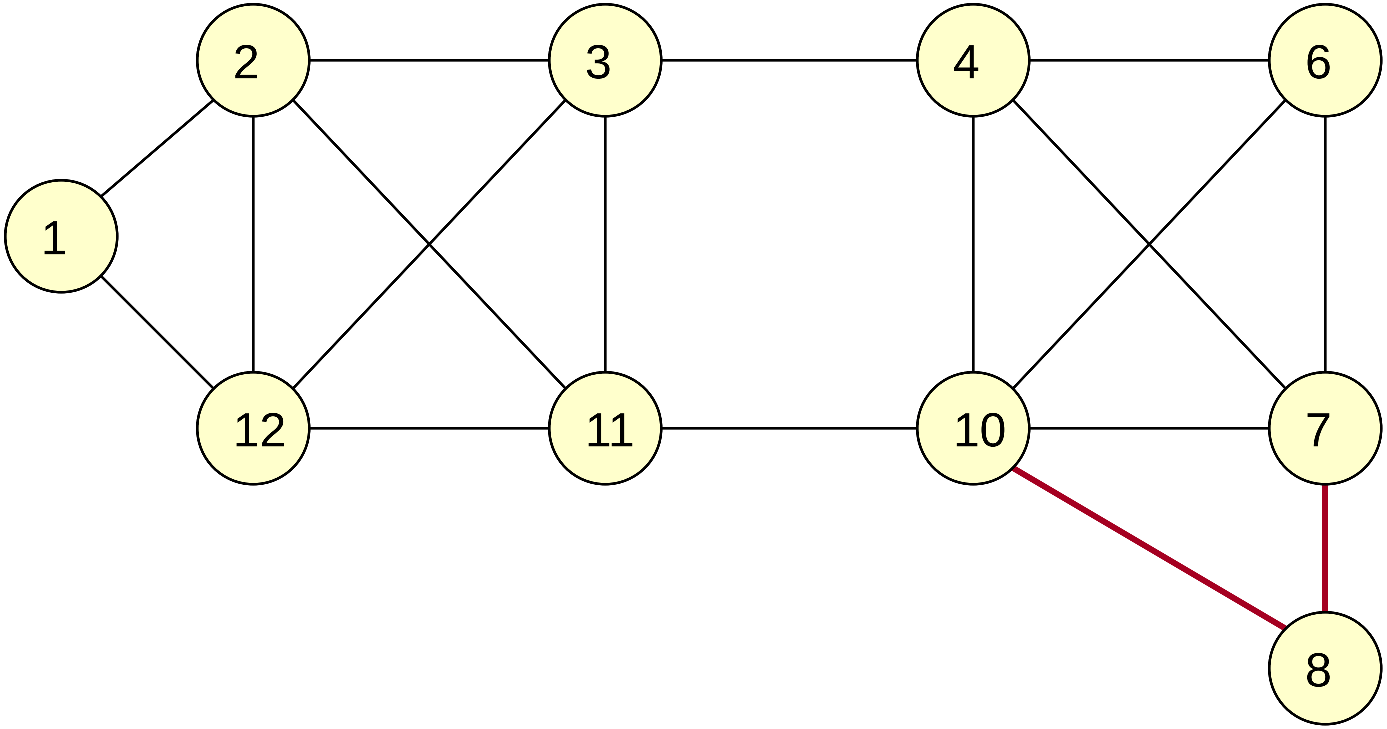


Choosing between cut sets

HCS: Improvements



HCS: Improvements



HCS: Improvements

- Iterated HCS
  - Sometimes there are multiple minimum cuts to choose from
    - Some cuts may create “singletons” or nodes that become disconnected from the rest of the graph
  - Performs several iterations of HCS until no new cluster is found (to find best final clusters)
    - Theoretically adds another O(n) factor to running time, but typically only needs 1 – 5 more iterations

HCS: Improvements

- Remove low degree nodes first
  - If node has low degree, likely will just be separated from rest of graph
  - Calculating separation for those nodes is expensive
  - Removal helps eliminate unnecessary iterations and significantly reduces running time

HCS Conclusion

- Performance
  - With improvements, can handle problems with up to thousands of elements in reasonable computing time
  - Generates clusters with high homogeneity and separation
  - More robust (responds better when noise is introduced) than other approaches based on connectivity

Identify connected subgraphs

The network of protein interactions is typically presented as an **undirected graph** with proteins as nodes and protein interactions as undirected edges.

Aim: identify highly connected subgraphs (**clusters**) that have more interactions within themselves and fewer with the rest of the graph.

A fully connected subgraph, or **clique**, that is not a part of any other clique is an example of such a cluster.

$$Q = \frac{2m}{n(n-1)}$$

In general, clusters need not to be fully connected.

Measure density of connections by

where *n* is the number of proteins in the cluster
and *m* is the number of interactions between them.

Identify all fully connected subgraphs (cliques)

Generally, finding all cliques of a graph is an NP-hard problem.
Because the protein interaction graph is sofar **very sparse** (the number of interactions (edges) is similar to the number of proteins (nodes), this can be done quickly.

To find cliques of size *n* one needs to enumerate only the cliques of size *n-1.*

The search for cliques starts with *n* = 4, pick all (known) pairs of edges (6500 × 6500 protein interactions) successively.
For every pair *A*-*B* and *C*-*D* check whether there are edges between *A* and *C, A* and *D, B* and *C,* and *B* and *D.* If these edges are present, *ABCD* is a clique.

For every clique identified, *ABCD,* pick all known proteins successively.
For every picked protein *E,* if all of the interactions *E*-*A, E*-*B, E*-*C,* and *E*-*D* are known, then *ABCDE* is a clique with size 5.

Continue for *n* = *6, 7, ...* The largest clique found in the protein-interaction network has size 14.

Identify all fully connected subgraphs (cliques)

These results include, however, many redundant cliques.
For example, the clique with size 14 contains 14 cliques with size 13.

To find all nonredundant subgraphs, mark all proteins comprising the clique of size 14, and out of all subgraphs of size 13 pick those that have at least one protein other than marked.

After all redundant cliques of size 13 are removed, proceed to remove redundant twelves etc.

Monte Carlo Simulation

Use MC to find a tight subgraph of a predetermined number of nodes *M*.

At time *t = 0,* a random set of *M* nodes is selected.

For each pair of nodes *i,j* from this set, the shortest path L_{ij} between *i* and *j* on the graph is calculated.

Denote the sum of all shortest paths L_{ij} from this set as L_0.

At every time step one of *M* nodes is picked at random, and one node is picked at random out of all its neighbors.

$$\exp^{-\frac{L_1 - L_0}{T}}$$

The new sum of all shortest paths, L_1, is calculated if the original node were to be replaced by this neighbor.

If $L_1 < L_0$, accept replacement with probability 1.

If $L_1 > L_0$, accept replacement with probability

where *T* is the effective temperature.

Monte Carlo Simulation

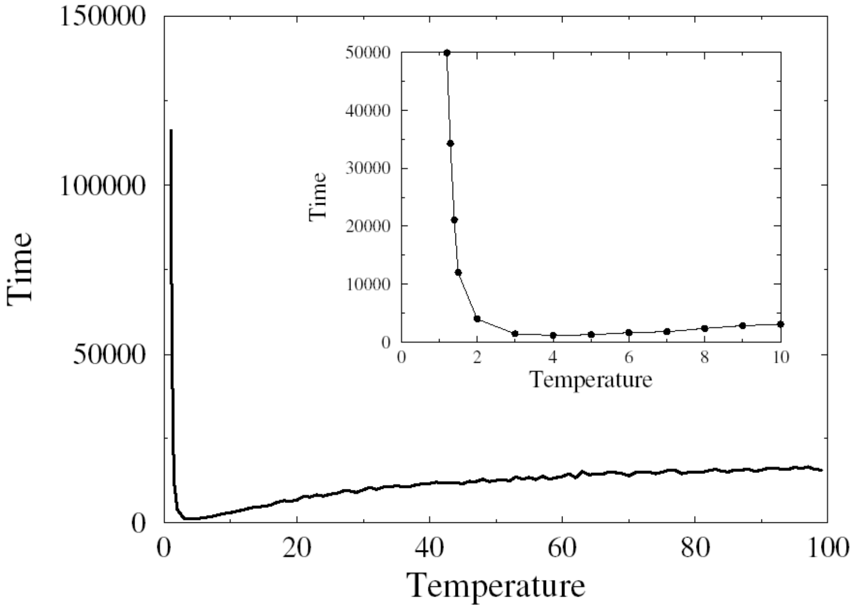
Every tenth time step an attempt is made to replace one of the nodes from the current set with a node that has no edges to the current set to avoid getting caught in an isolated disconnected subgraph.

This process is repeated
(i) until the original set converges to a complete subgraph, or
(ii) for a predetermined number of steps,
after which the tightest subgraph (the subgraph corresponding to the smallest L_0) is recorded.

The recorded clusters are merged and redundant clusters are removed.

Optimal temperature in MC simulation

For every cluster size there is an optimal temperature that gives the fastest convergence to the tightest subgraph.



Time to find a clique with size 7 in MC steps per site as a function of temperature *T*.

The region with optimal temperature is shown in *Inset*.

The required time increases sharply as the temperature goes to 0, but has a relatively wide plateau in the region 3 < *T* < 7.

Simulations suggest that the choice of temperature $T \approx M$ would be safe for any cluster size *M.*

Merging Overlapping Clusters

A simple statistical test shows that nodes which have only one link to a cluster are statistically insignificant. Clean such statistically insignificant members first.

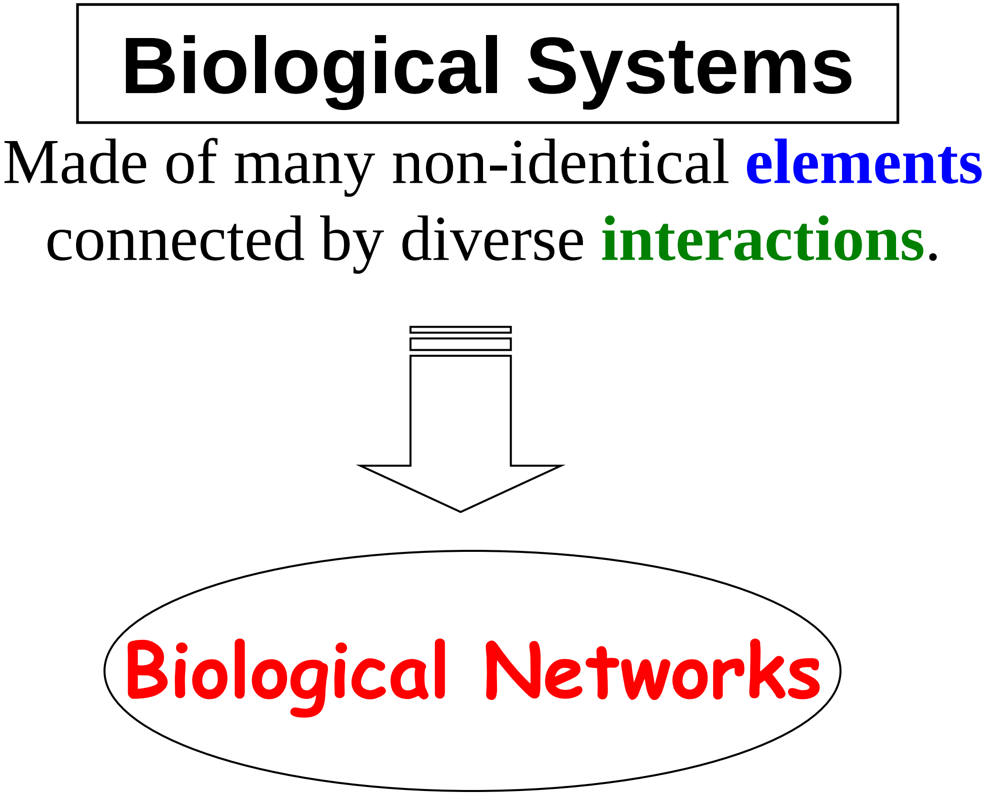
Then merge overlapping clusters:
For every cluster A_i find all clusters A_k that overlap with this cluster by at least one protein.
For every such found cluster calculate Q value of a possible merged cluster $A_i \cup A_k$. Record cluster $A_{best}(i)$ which gives the highest Q value if merged with A_i.

After the best match is found for every cluster, every cluster A_i is replaced by a merged cluster $A_i \cup A_{best}(i)$ unless $A_i \cup A_{best}(i)$ is below a certain threshold value for Q_C.
This process continues until there are no more overlapping clusters or until merging any of the remaining clusters witll make a cluster with Q value lower than Q_C.

Modularity and Community Detection in PPI networks

- Background.
- Network module definition.
- Algorithm for identifying modules in network.

Biological Networks



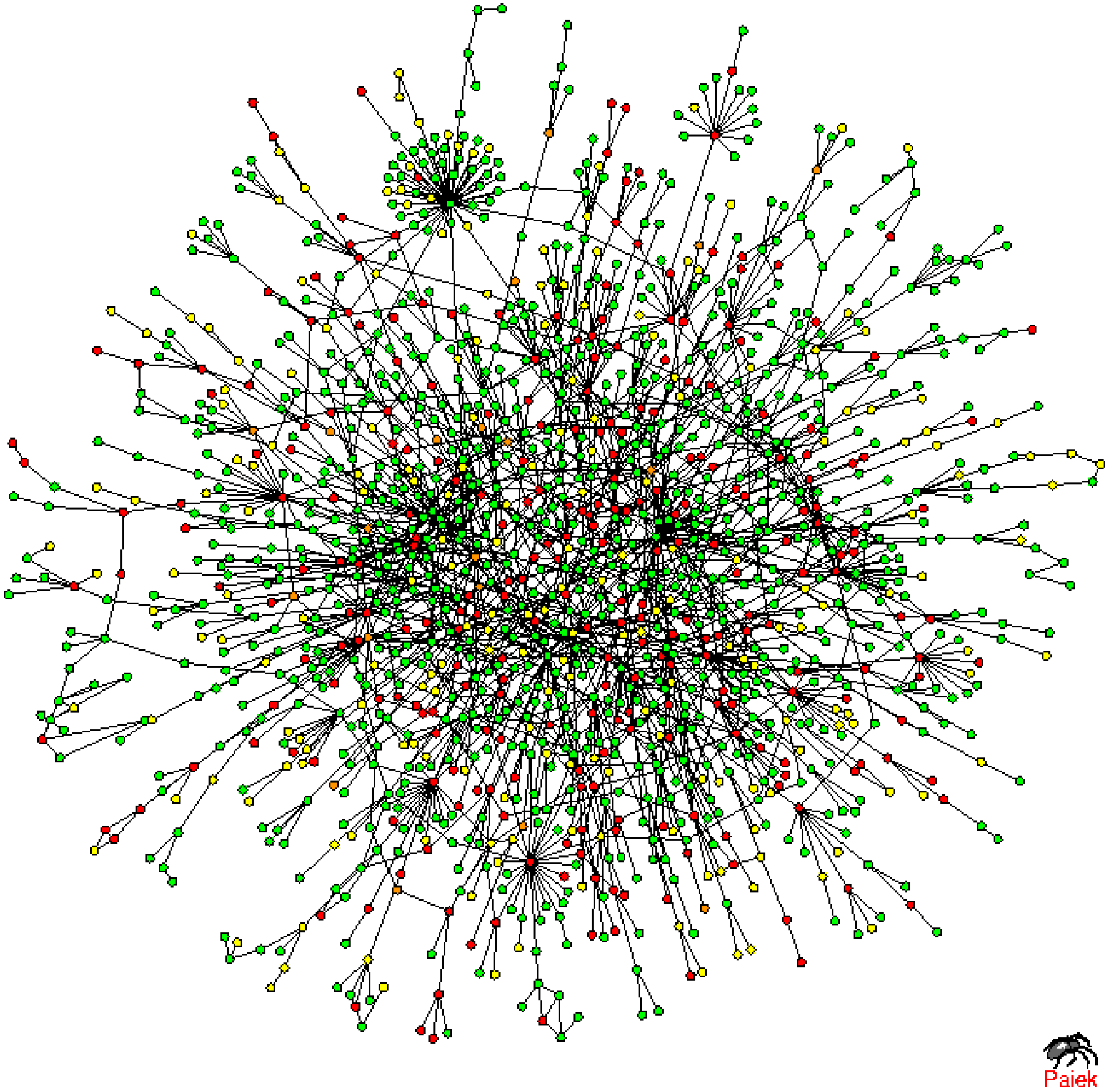
Biological Systems

Made of many non-identical **elements**
connected by diverse **interactions**.

Biological Networks

Biological networks as framework for the study of biological systems

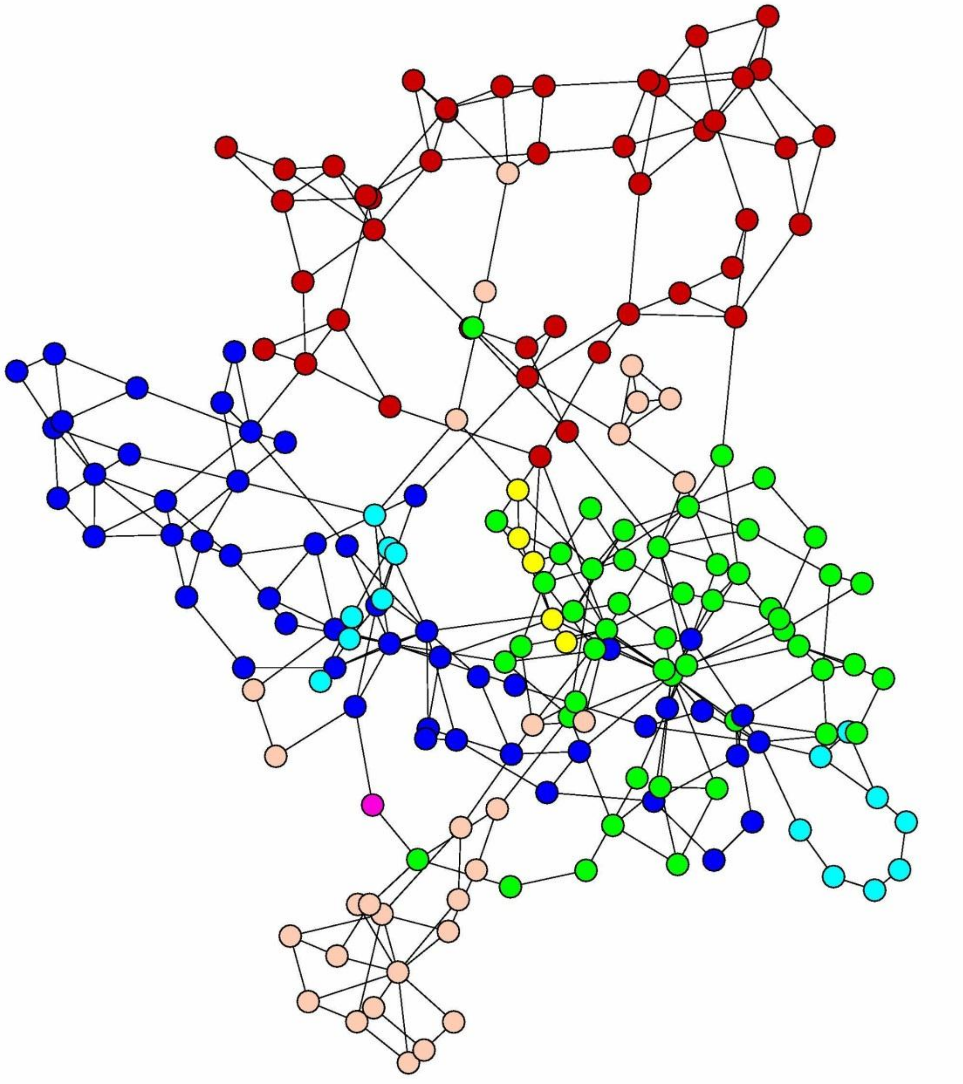
Protein Interaction Network



Nodes: proteins
Links: physical interactions

(Jeong et al., 2001)

Metabolic Network



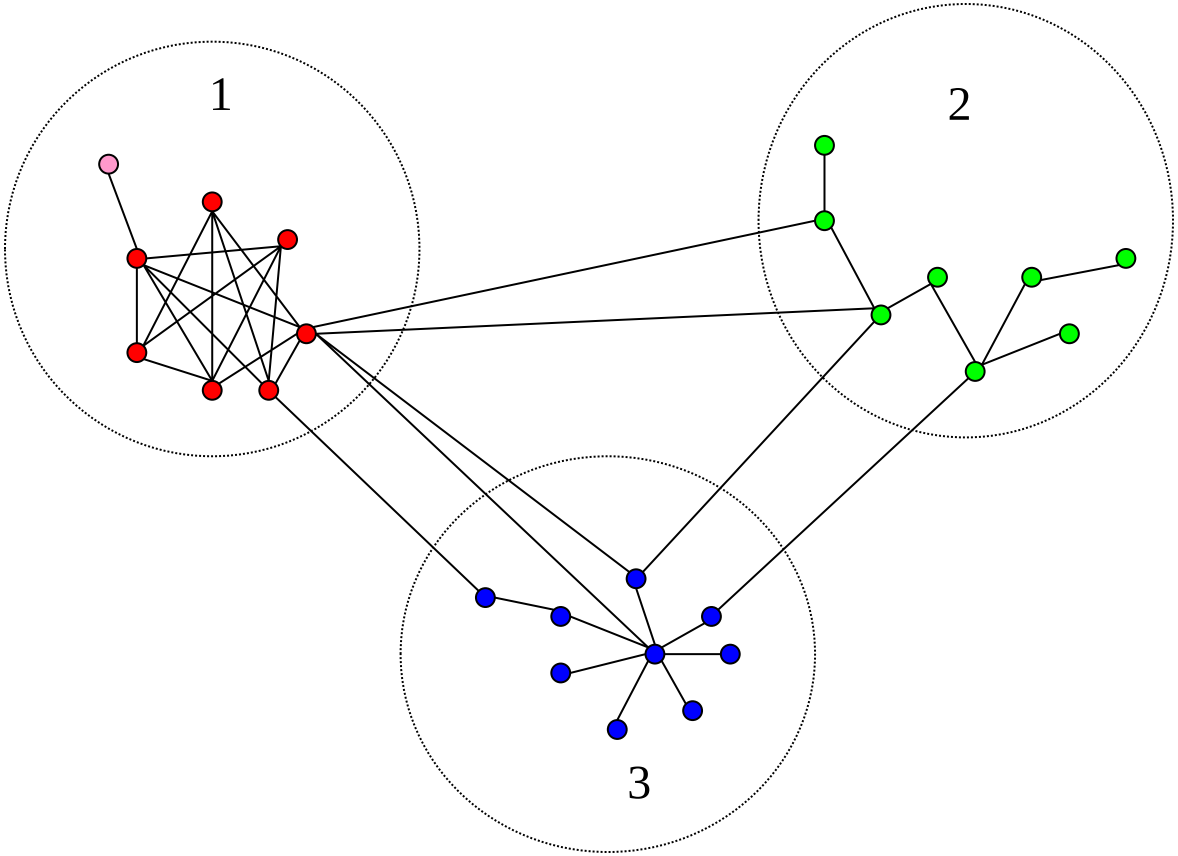
Nodes: chemicals (substrates)
Links: chemistry reactions

(Ravasz et al., 2002)

Biological System are Modular

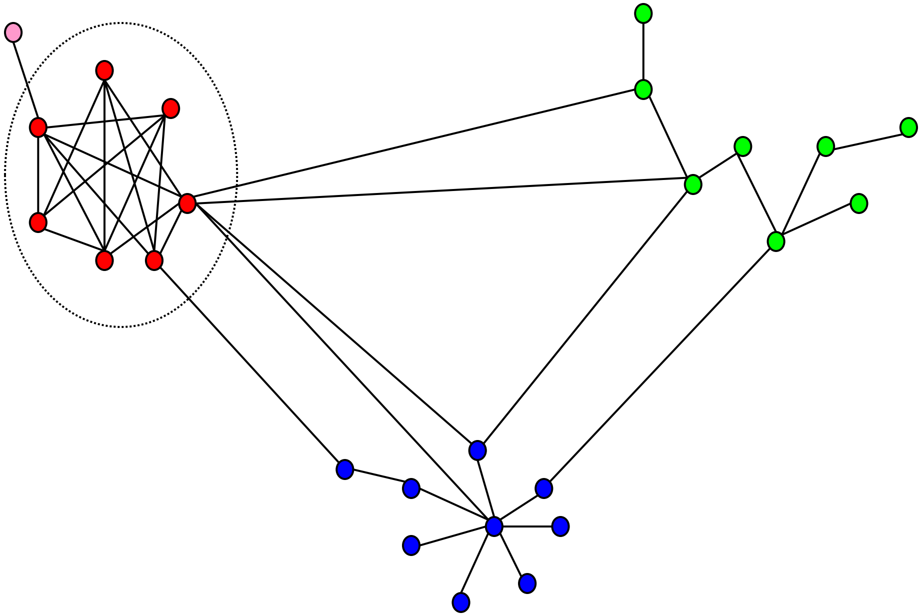
- There is increasing evidence that the cell system is composed of modules
  - A "module" in a biological system is a discrete unit whose function is separable from those of other modules
  - Modules defined based on functional criteria reflect the critical level of biological organization (Hartwell, et al.)
  - A modular system can reuse existing, well-tested modules
- Functional modules will be reflect in the topological structures of biological networks.
- Identifying functional modules and their relationship from biological networks will help to the understanding of the organization, evolution and interaction of the cellular systems they represent

Biological Modules in Biological Networks



Background: Identify Modules from Biological Networks

- Most efforts focused on detecting highly connected clusters.
  - Ignored the peripheral proteins.
  - Modules with other topology are not identified.
  - Modules are isolated and no inter relationship is revealed.



Background: Identify Modules from Biological Networks (continue)

- Traditional clustering algorithms have been applied to protein interaction networks (PIN) to find biological modules.
  - Need transforming PIN into weighted networks
    - Weight the protein interactions based on number of experiments that support the interaction (Pereira-Leal et al).
    - Weight with shortest path length (River et al. and Arnau et al.).
  - Drawbacks
    - Weights are artificial.
    - “tie in proximity” problem in hierarchical agglomerative clustering (HAC).

Background: Identify Modules from Biological Networks (continue)

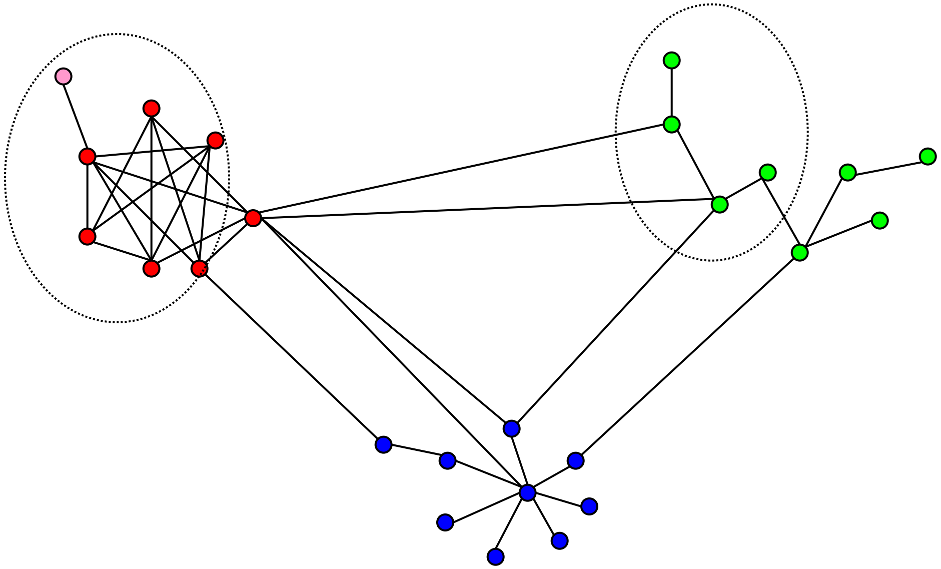
- Radicchi et al. (PNAS, 2004) proposed two new definitions of module in network.
  - For a sub-graph *V*⊂*G*, the degree definition of vertex i∈V in a undirected graph

    $$k_i^{in}(V) = \sum_{j \in V} A_{i,j} \qquad k_i^{out}(V) = \sum_{j \notin V} A_{i,j}$$

    $A_{i,j}$ equal to 1 if *i* and *j* are directly connected; it is equal to zero otherwise.
  - Strong definition of Module $k_i^{in}(V) > k_i^{out}(V) \quad \forall i \in V$
  - Weak definition of Module $\sum_{i \in V} k_i^{in}(V) > \sum_{i \in V} k_i^{out}(V)$

Background: Identify Modules from Biological Networks (continue)

- Two module definitions do not follow the intuitive concept of module exactly.



Degree of Subgraph

- Given a graph G, let S be a subgraph of G (S⊂ G).
  - The adjacent matrix of sub-graph S and its neighbors N can be given as:

$$S_{ij} = \begin{cases} 1 & if\ \ vertices\ i\ and\ \ j\ connected\ ,\ and\ either\ \ i\ or\ \ j\ belongs\ to\ S \\ 0 & otherwise \end{cases}$$

  - **Indegree** of S, **Ind(S)**:

$$ind(S) = \sum_{i,j} S_{ij}\delta(i, j)$$

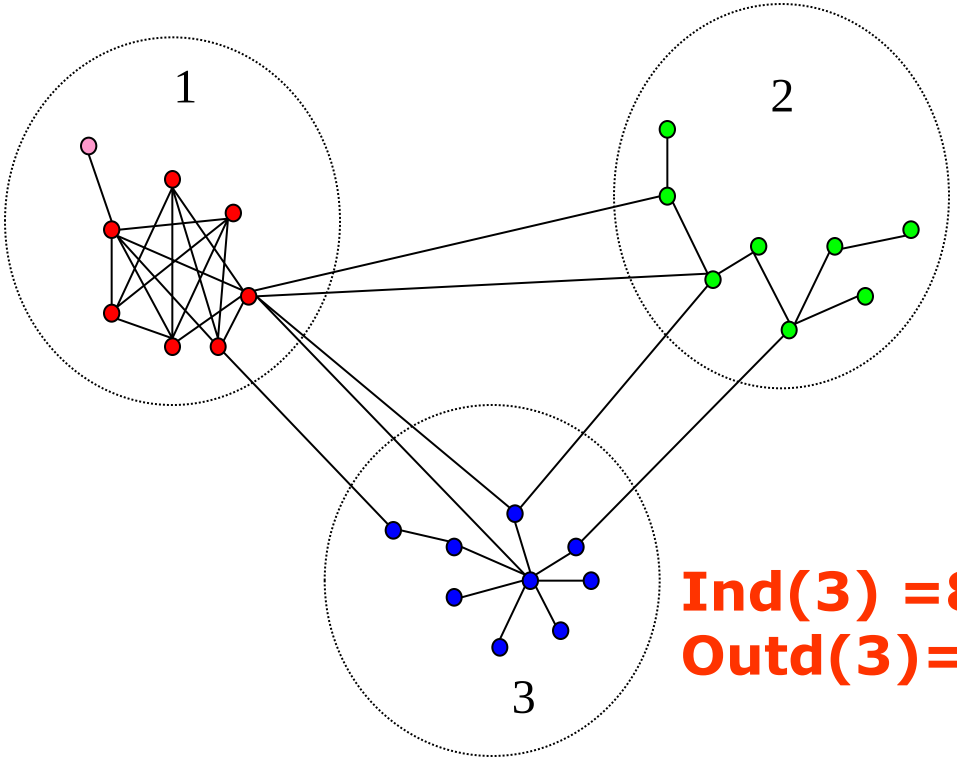
Where $\delta(i, j)$ is 1 if both vertex *i* and vertex *j* are in sub-graph S and 0 otherwise.

  - **Outdegree** of S, **Outd(S)**:

$$outd(S) = \sum_{i,j} S_{ij}\lambda(i, j)$$

Where $\lambda(i, j)$ is 1 if only one of vertex *i* and vertex *j* belong to sub-graph S and 0 otherwise.

Degree of Subgraph: Example



Ind(1) =16
Outd(1)=5

Ind(2) =7
Outd(2)=4

Ind(3) =8
Outd(3)=5

Modularity

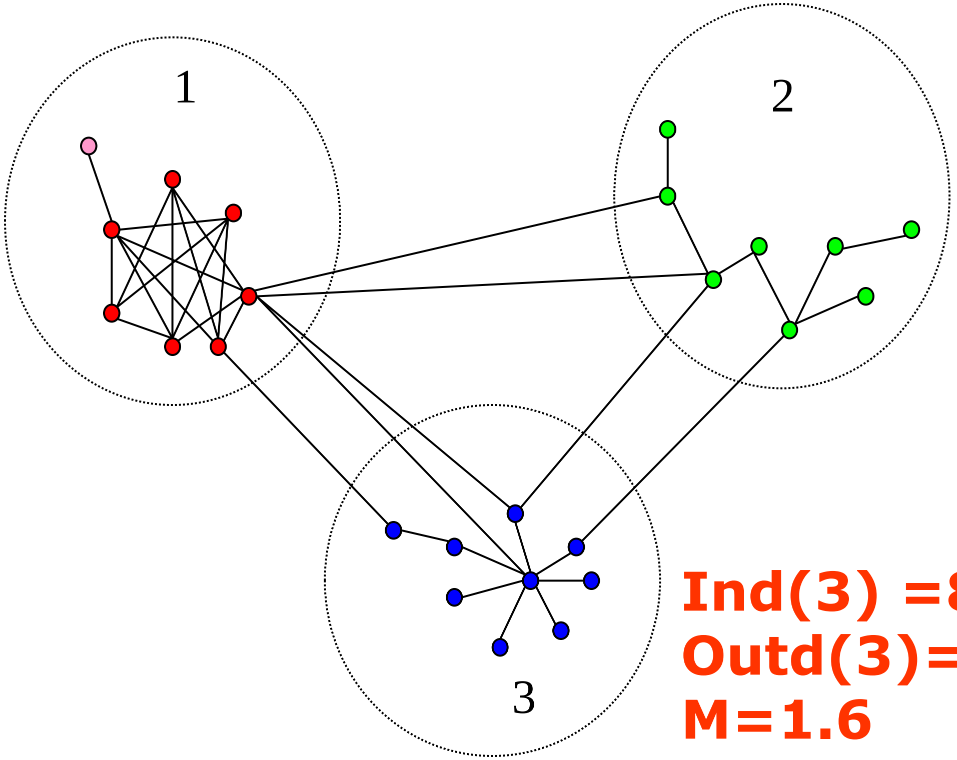
□ *The modularity **M** of a sub-graph S in a given graph G is defined as the ratio of its indegree, **ind(S)**, and outdegree, **outd(S)**:*

$$M = \frac{ind(S)}{outd(S)}$$

New Network Module Definition

- *A subgraph $S \subset G$ is a* ***module*** *if $M>1$.*

Ind(1) =16
Outd(1)=5
M=3.2

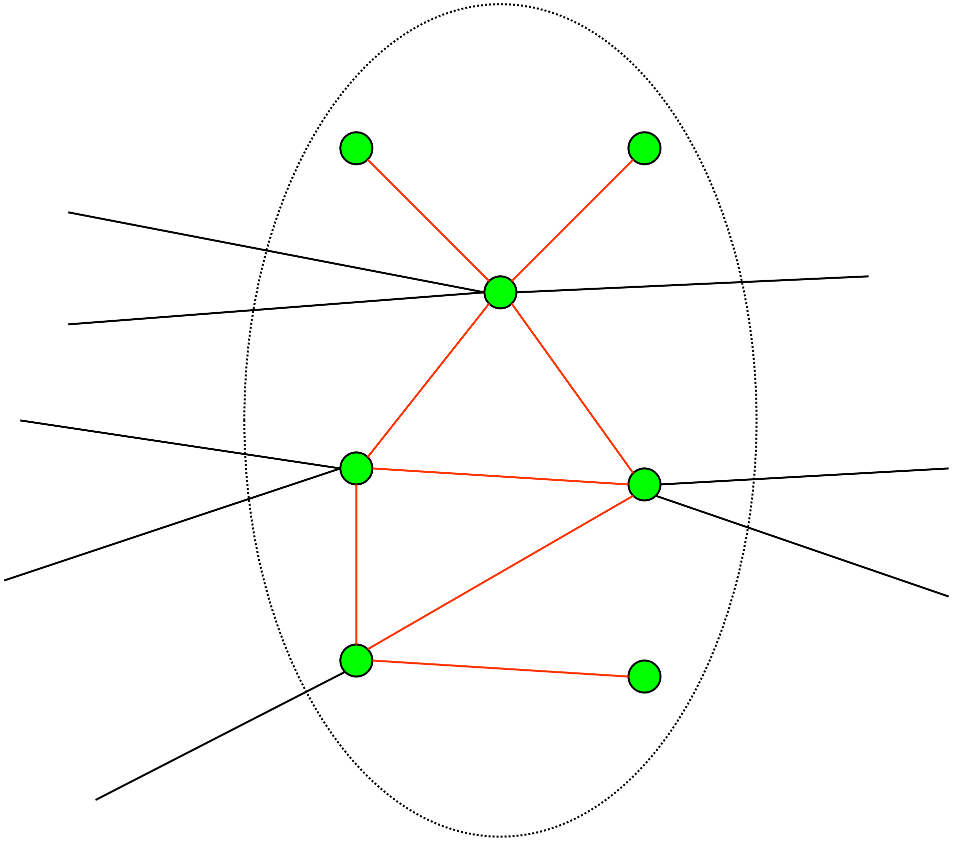


Ind(2) =7
Outd(2)=4
M=1.75

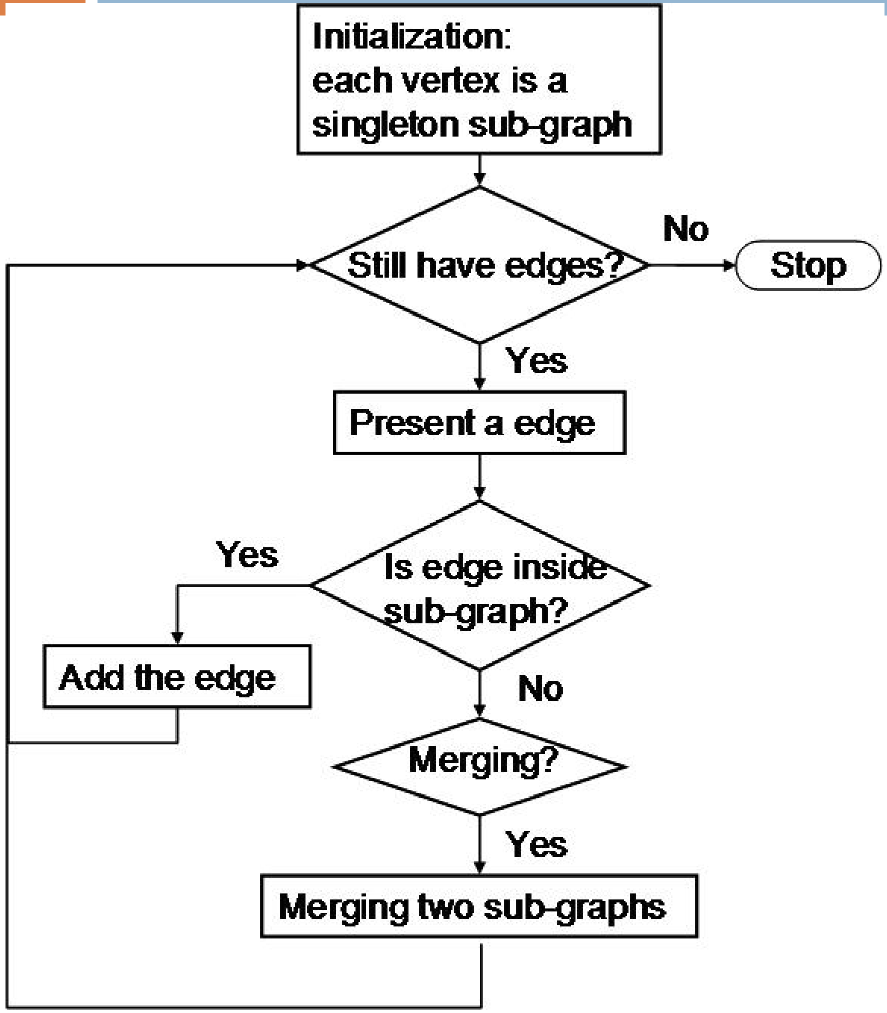
Ind(3) =8
Outd(3)=5
M=1.6

Comparison to Radicchi's Module Defintions

- This sample network is a Strong module, but is not a module by this new definition based on ***indegree*** vs ***outdegree*** criteria



Agglomerative Algorithm for Identifying Network Modules



Initialization: each vertex is a singleton sub-graph

Still have edges?

No

Stop

Yes

Present a edge

Is edge inside sub-graph?

Yes

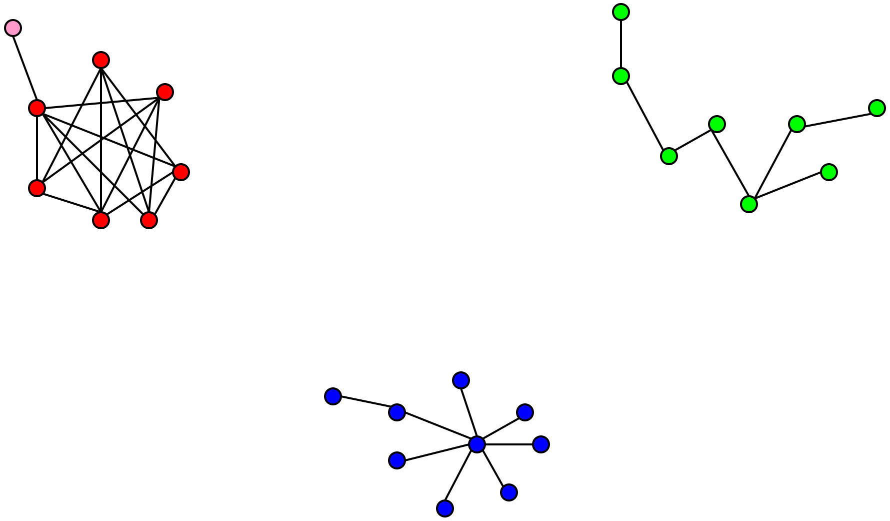
Add the edge

No

Merging?

Yes

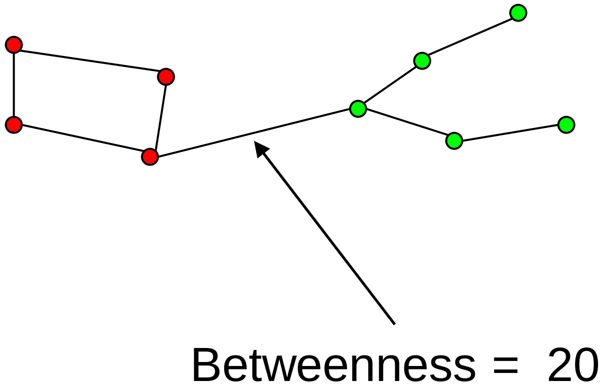
Merging two sub-graphs



Flow chart of the agglomerative algorithm

The Order of Merging

- Edge Betweenness (Girvan-Newman, 2002)
  - Defined as the number of shortest paths between all pairs of vertices that run through it.
  - Edges between modules have higher betweenness values.



The Order of Merging (continue)

- Gradually deleting the edge with the highest betweenness will generate an **order of edges**.
  - Edges between modules will be deleted earlier.
  - Edges inside modules will be deleted later.
- Reverse the deletion order of edges and use it as the merging order.

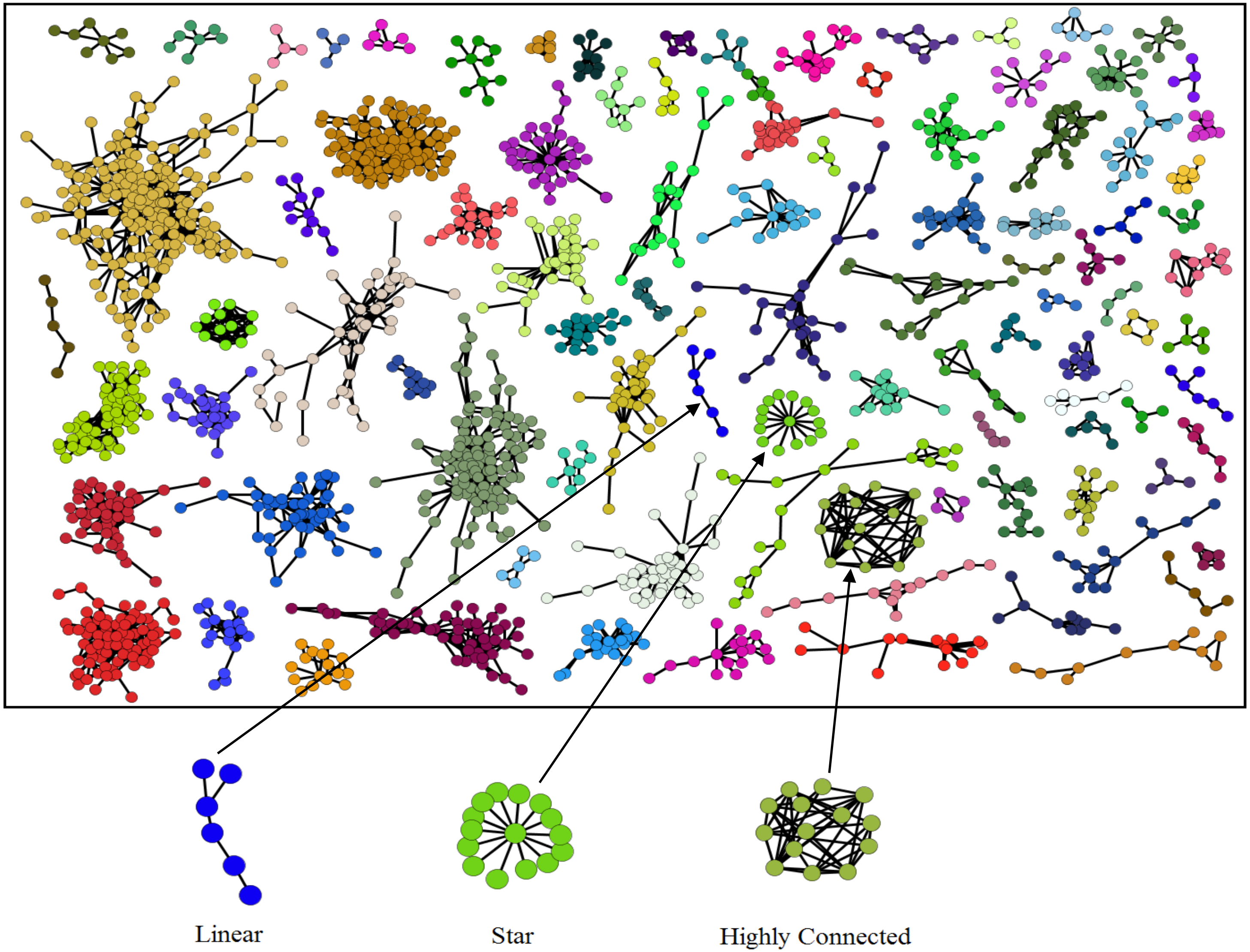
When Merging Occurs?

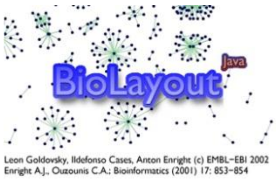
- Between two non-modules
- Between a non-module and a module
- Not between two modules

Testing Data Set

- Yeast Core Protein Interaction Network (PIN).
  - The yeast core PIN from Database of Interacting Proteins (DIP) (version ScereCR20041003).
  - Total: 2609 proteins; 6355 links.
  - Large component: 2440 proteins, 6401 interactions.

86 Modules Obtained from DIP Yeast core PIN





Leon Goldovsky, Ildefonso Cases, Anton Enright (c) EMBL−EBI 2002
Enright A.J., Ouzounis C.A.; Bioinformatics (2001) 17: 853−854

Validation of modules

- Annotated each protein with the Gene Ontology^{TM} (GO) terms from the *Saccharomyces* Genome Database (SGD) (Cherry et al. 1998; Balakrishna et al)

- Quantified the co-occurrence of GO terms using the hypergeometric distribution analysis supported by the Gene Ontology Term Finder of SGD(Balakrishna et al)

- The results show that each module has statistically significant co-occurrence of bioprocess GO categories

Validation of modules

Modules with 100% GO frequency

Module #	GOID	GO_term	Frequency	Genome Frequency	Probability
134	45851	pH reduction	14 out of 14 genes, 100%	21 out of 7274	2.79E-36
140	6402	mRNA catabolism	14 out of 14 genes, 100%	55 out of 7274	1.99E-30
23	6267	pre-replicative complex formation and maintenance	7 out of 7 genes, 100%	13 out of 7272	5.83E-20
99	6617	SRP-dependent cotranslational protein-membrane targeting, signal sequence recognition	6 out of 6 genes, 100%	7 out of 7274	7.94E-19
109	6207	'de novo' pyrimidine base biosynthesis	5 out of 5 genes, 100%	5 out of 7274	1.53E-16
54	42147	retrograde transport, endosome to Golgi	5 out of 5 genes, 100%	10 out of 7272	4.91E-15
108	6303	double-strand break repair via nonhomologous end-joining	5 out of 5 genes, 100%	19 out of 7274	1.21E-13
96	96	sulfur amino acid metabolism	5 out of 5 genes, 100%	31 out of 7274	1.40E-12
55	6896	Golgi to vacuole transport	4 out of 4 genes, 100%	18 out of 7272	3.75E-11
84	6109	regulation of carbohydrate metabolism	4 out of 4 genes, 100%	26 out of 7274	1.63E-10

Validation of modules

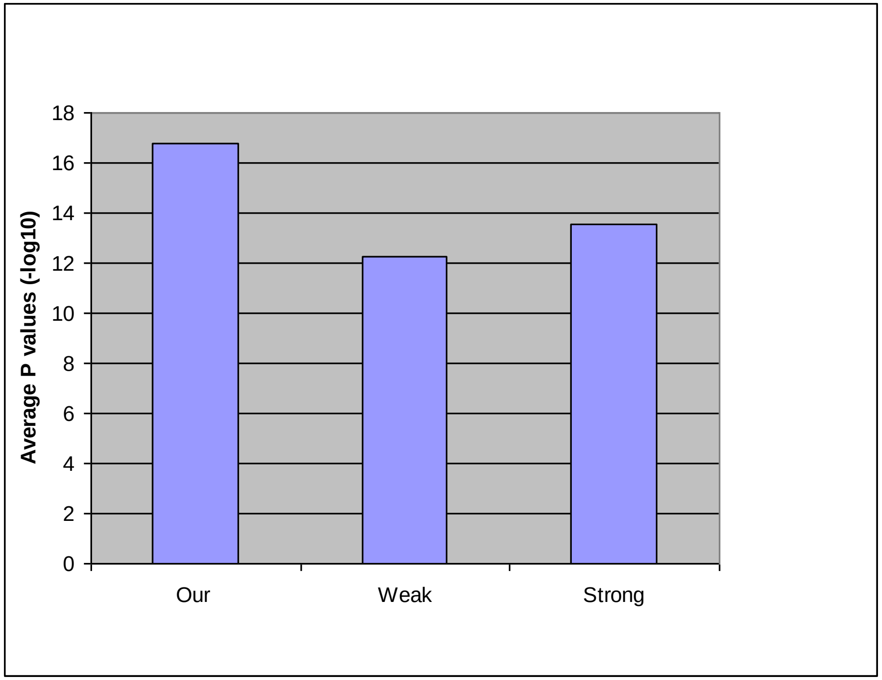
Most significant GO term in top 10 largest modules

Module #	Module Size	GOID	GO term	Frequency	Genome Frequency	Probability
202	201	6913	nucleocytoplasmic transport	62 out of 201 genes, 30.8%	105 out of 7274	5.48E-63
199	111	30163	protein catabolism	46 out of 111 genes, 41.4%	175 out of 7274	2.85E-44
193	93	16071	mRNA metabolism	58 out of 93 genes, 62.3%	184 out of 7274	4.69E-68
189	76	7028	cytoplasm organization and biogenesis	56 out of 76 genes, 73.6%	250 out of 7274	5.81E-65
187	59	30036	actin cytoskeleton organization and biogenesis	31 out of 59 genes, 52.5%	101 out of 7274	9.93E-42
182	50	6366	transcription from RNA polymerase II promoter	34 out of 50 genes, 68%	270 out of 7274	6.35E-37
185	45	16573	histone acetylation	17 out of 45 genes, 37.7%	28 out of 7274	8.90E-30
188	45	6364	rRNA processing	34 out of 45 genes, 75.5%	175 out of 7274	7.18E-46
175	44	48193	Golgi vesicle transport	36 out of 44 genes, 81.8%	137 out of 7274	1.20E-54
194	42	6338	chromatin remodeling	18 out of 42 genes, 42.8%	128 out of 7274	6.18E-21

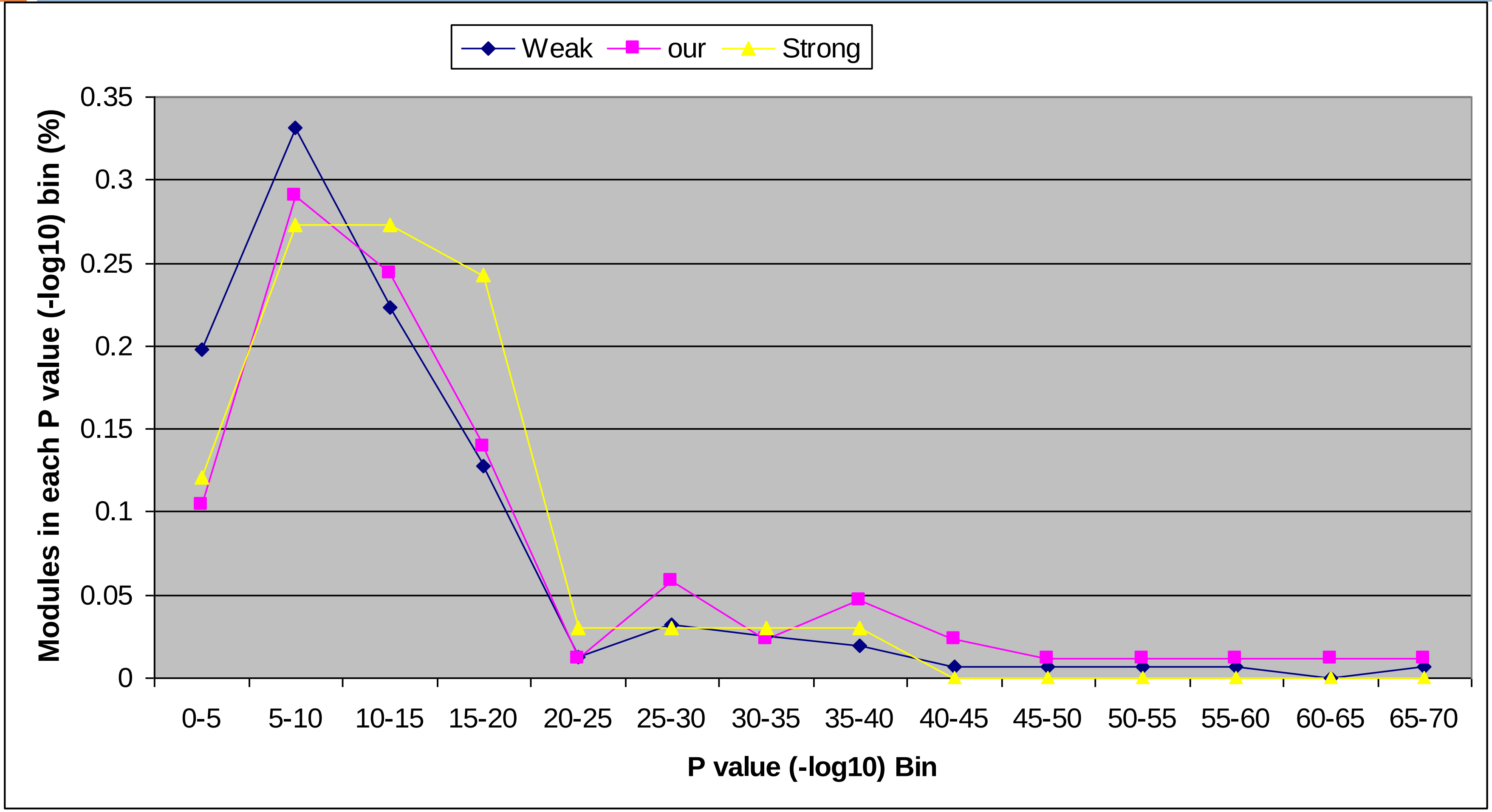
Validation of modules

- Comparison with module definitions of Radicchi et al.
  - Running the agglomerative algorithm based on different definitions

	Average lowest P value (-log10)	Number of Modules (larger than 3)
Our	16.77497	86
Weak	12.28661	157
Strong	13.5531	33

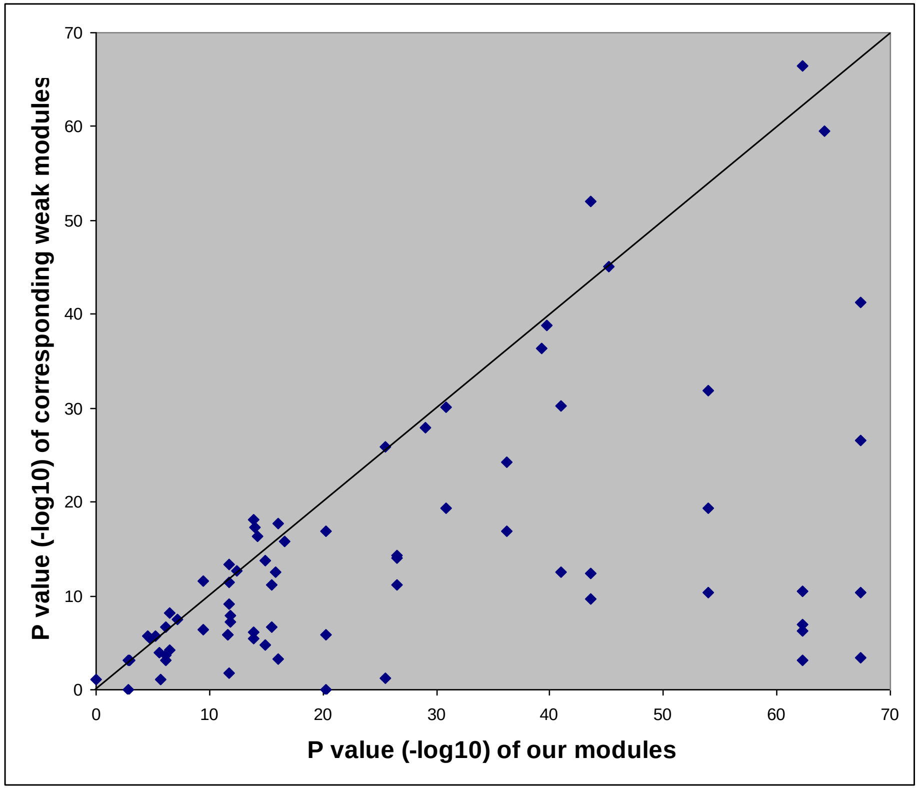


Validation of modules



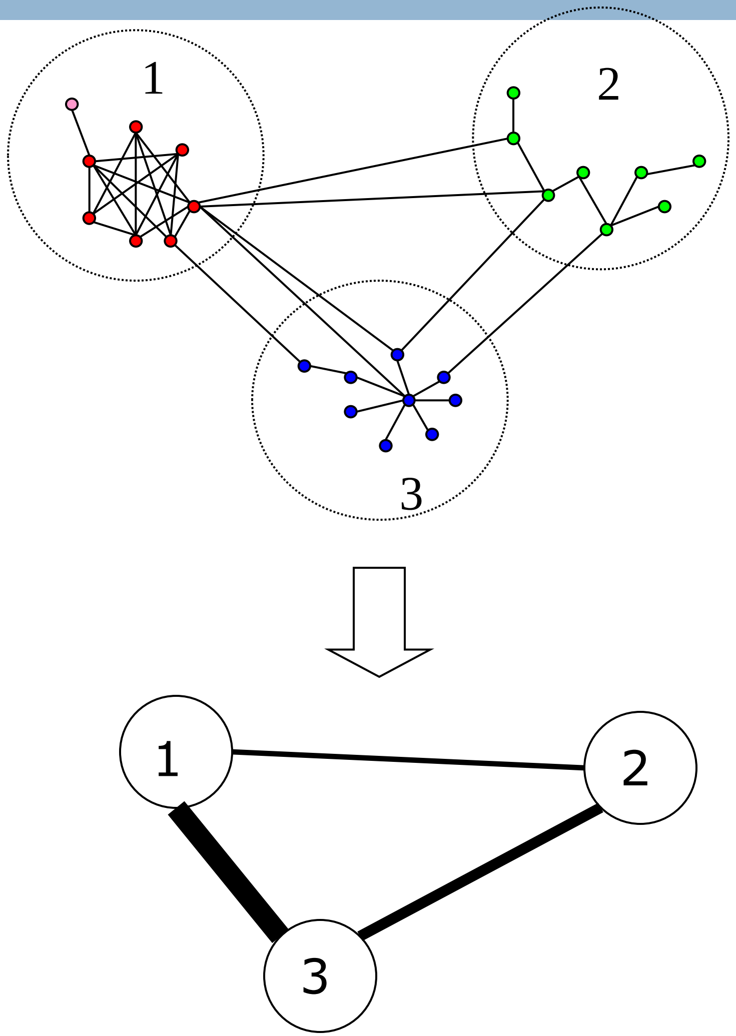
Validation of modules

- P values of modules obtained based our definition plot against P values of the corresponding weak modules (line is y=x).

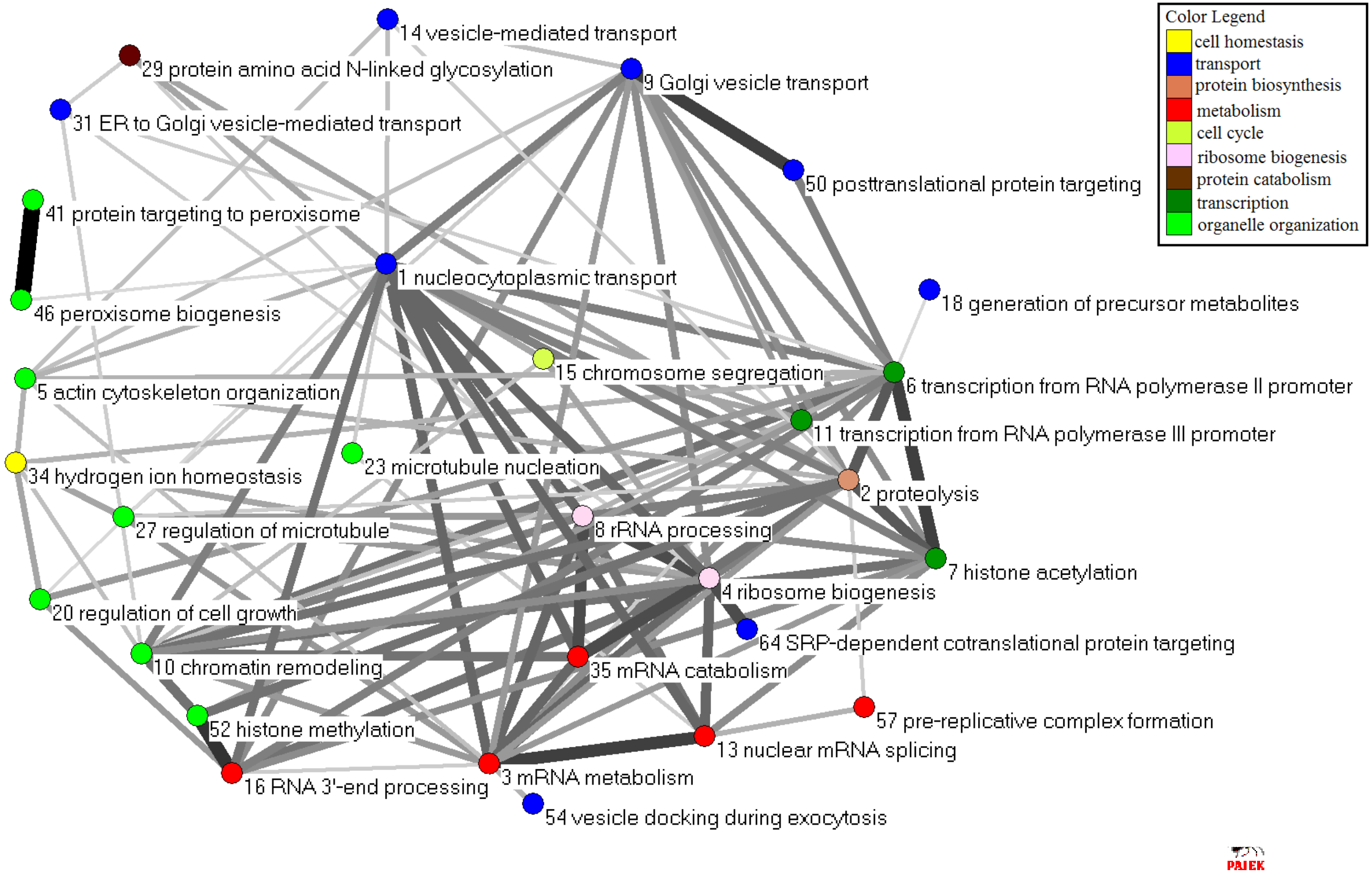


Constructing the Network of Modules

- Assembling the 86 MoNet modules to form an interconnected network of modules.
  - For **each adjacent module pair**, the edge that is deleted **last** by the G-N algorithm was selected from all the edges that connect two modules to represent the link between two modules.



A Section of Module Network of 30 Largest Modules

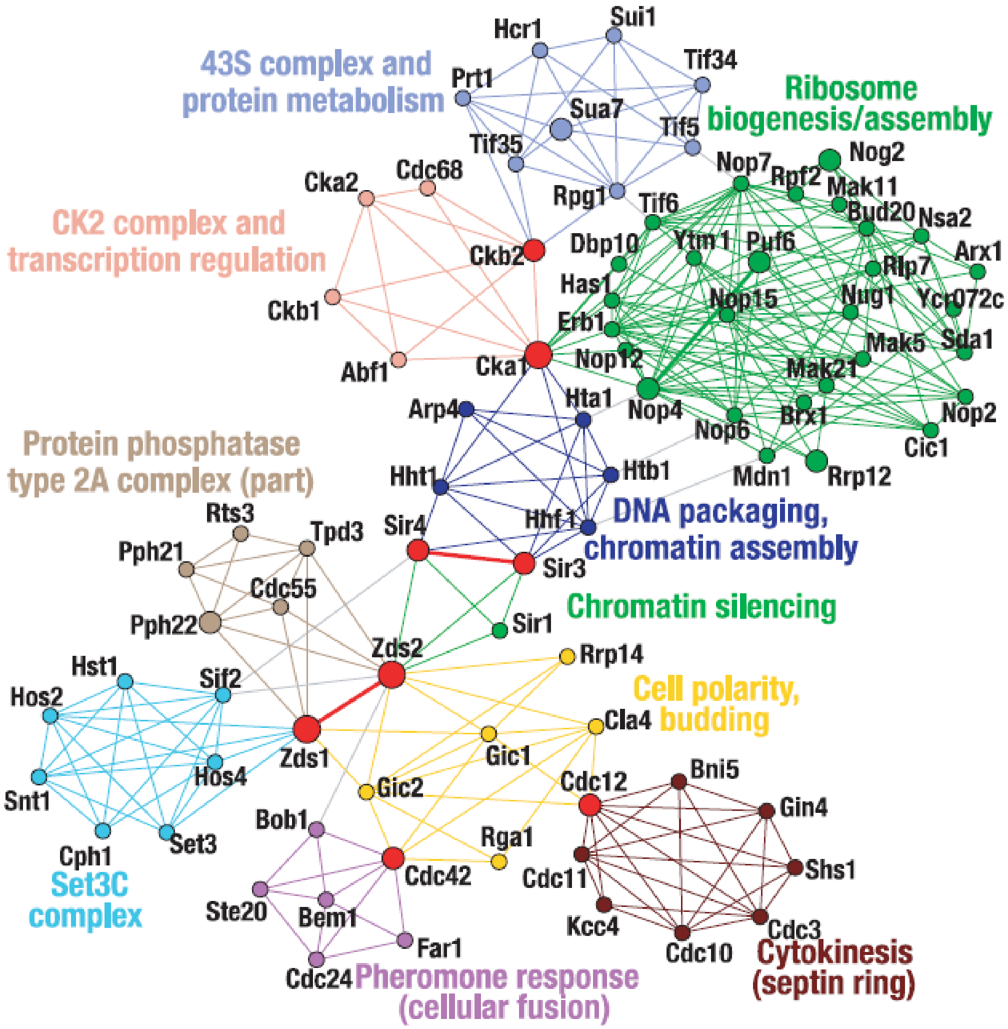

Color Legend
🟨 cell homestasis
🟦 transport
🟧 protein biosynthesis
🟥 metabolism
🟩 cell cycle
🟪 ribosome biogenesis
🟫 protein catabolism
🟩 transcription
🟩 organelle organization
14 vesicle-mediated transport
29 protein amino acid N-linked glycosylation
9 Golgi vesicle transport
31 ER to Golgi vesicle-mediated transport
50 posttranslational protein targeting
41 protein targeting to peroxisome
1 nucleocytoplasmic transport
18 generation of precursor metabolites
46 peroxisome biogenesis
15 chromosome segregation
6 transcription from RNA polymerase II promoter
5 actin cytoskeleton organization
11 transcription from RNA polymerase III promoter
23 microtubule nucleation
34 hydrogen ion homeostasis
2 proteolysis
27 regulation of microtubule
8 rRNA processing
7 histone acetylation
4 ribosome biogenesis
20 regulation of cell growth
64 SRP-dependent cotranslational protein targeting
10 chromatin remodeling
35 mRNA catabolism
57 pre-replicative complex formation
52 histone methylation
13 nuclear mRNA splicing
16 RNA 3'-end processing
3 mRNA metabolism
54 vesicle docking during exocytosis
PAJEK

Conclusions

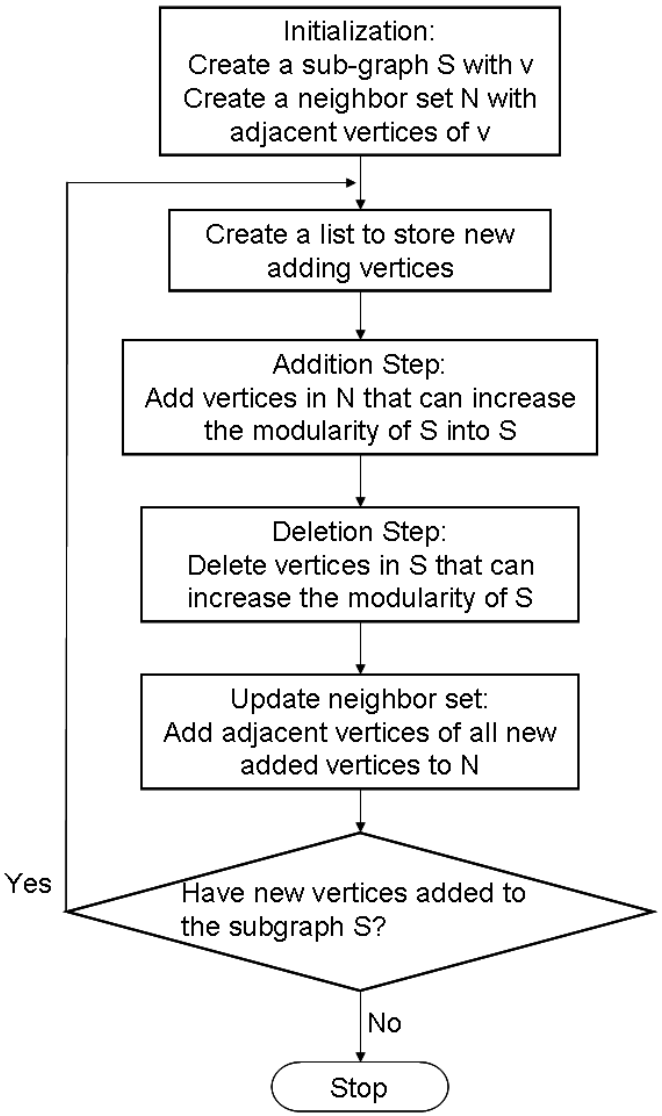
- Provide a framework for decomposing the protein interaction network into functional modules
- The modules obtained appear to be biological functional modules based on clustering of Gene Ontology terms
- The network of modules provides a plausible way to understanding the interactions between these functional modules
- With the increasing amounts of protein interaction data available, our approach will help construct a more complete view of interconnected functional modules to better understand the organization of the whole cellular system

Limitation of Global Algorithms

- □ Biological networks are incomplete.
- □ Each vertex can only belong to one module.

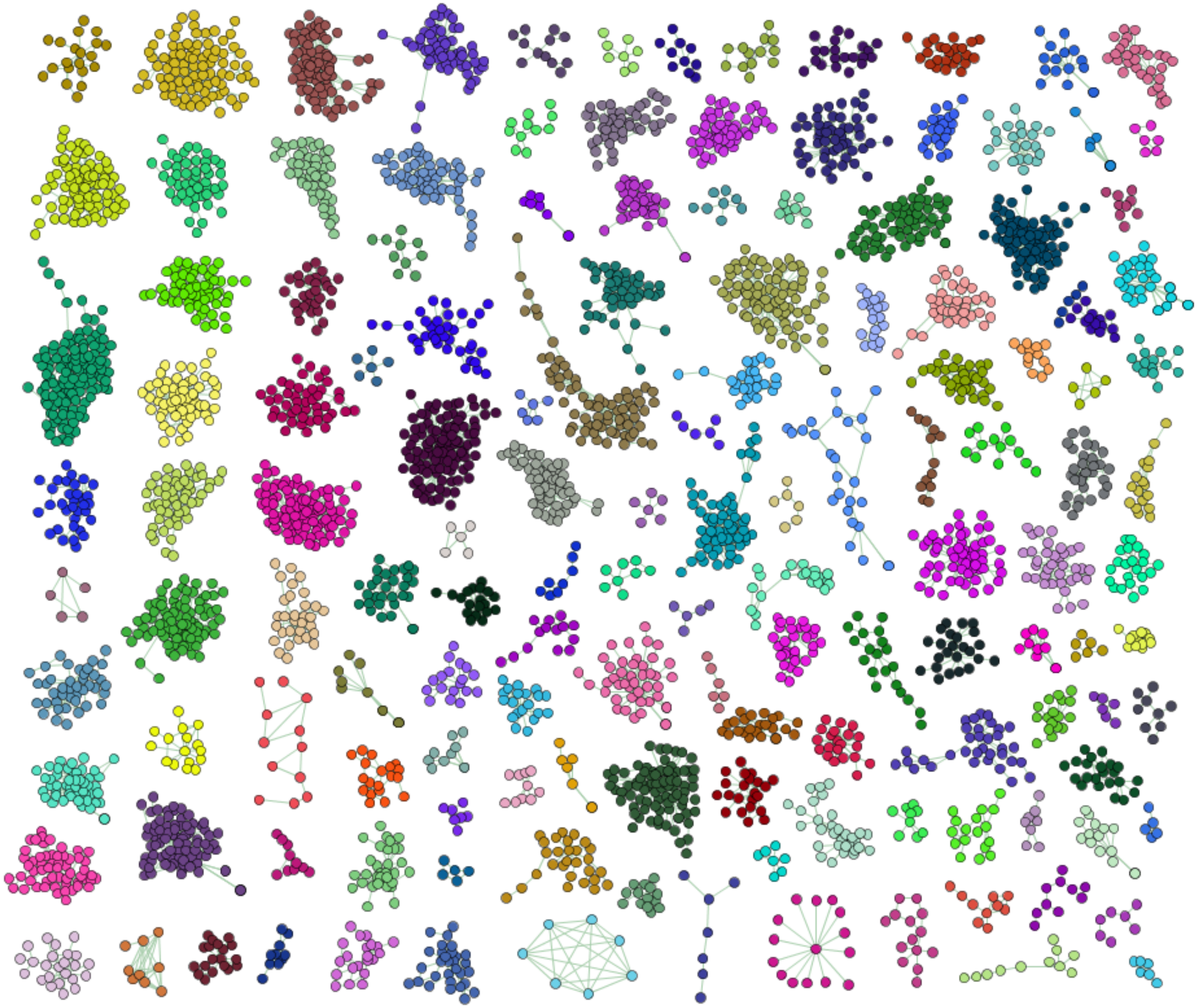


Local Optimization Algorithm

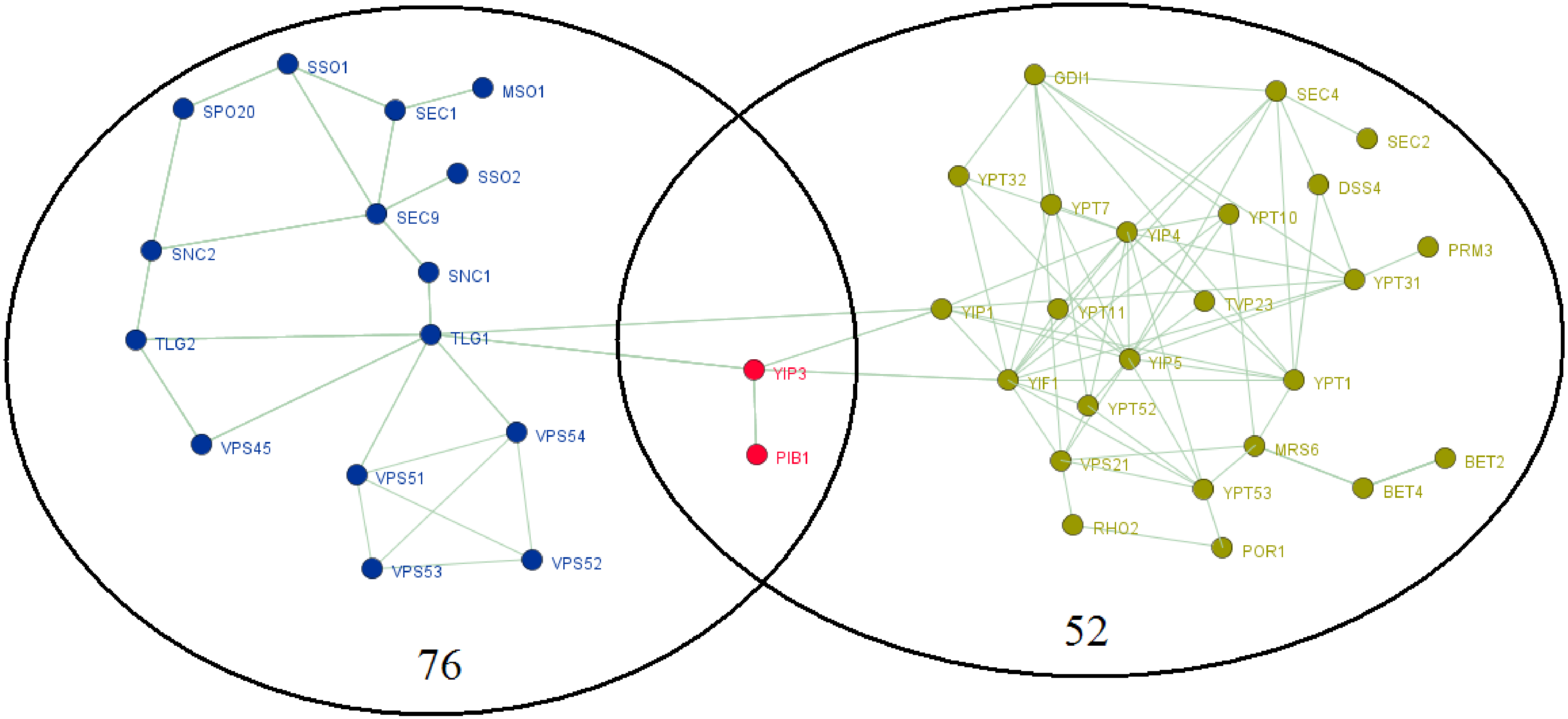


Initialization:
Create a sub-graph S with v
Create a neighbor set N with adjacent vertices of v
Create a list to store new adding vertices
Addition Step:
Add vertices in N that can increase the modularity of S into S
Deletion Step:
Delete vertices in S that can increase the modularity of S
Update neighbor set:
Add adjacent vertices of all new added vertices to N
Have new vertices added to the subgraph S?
Yes
No
Stop

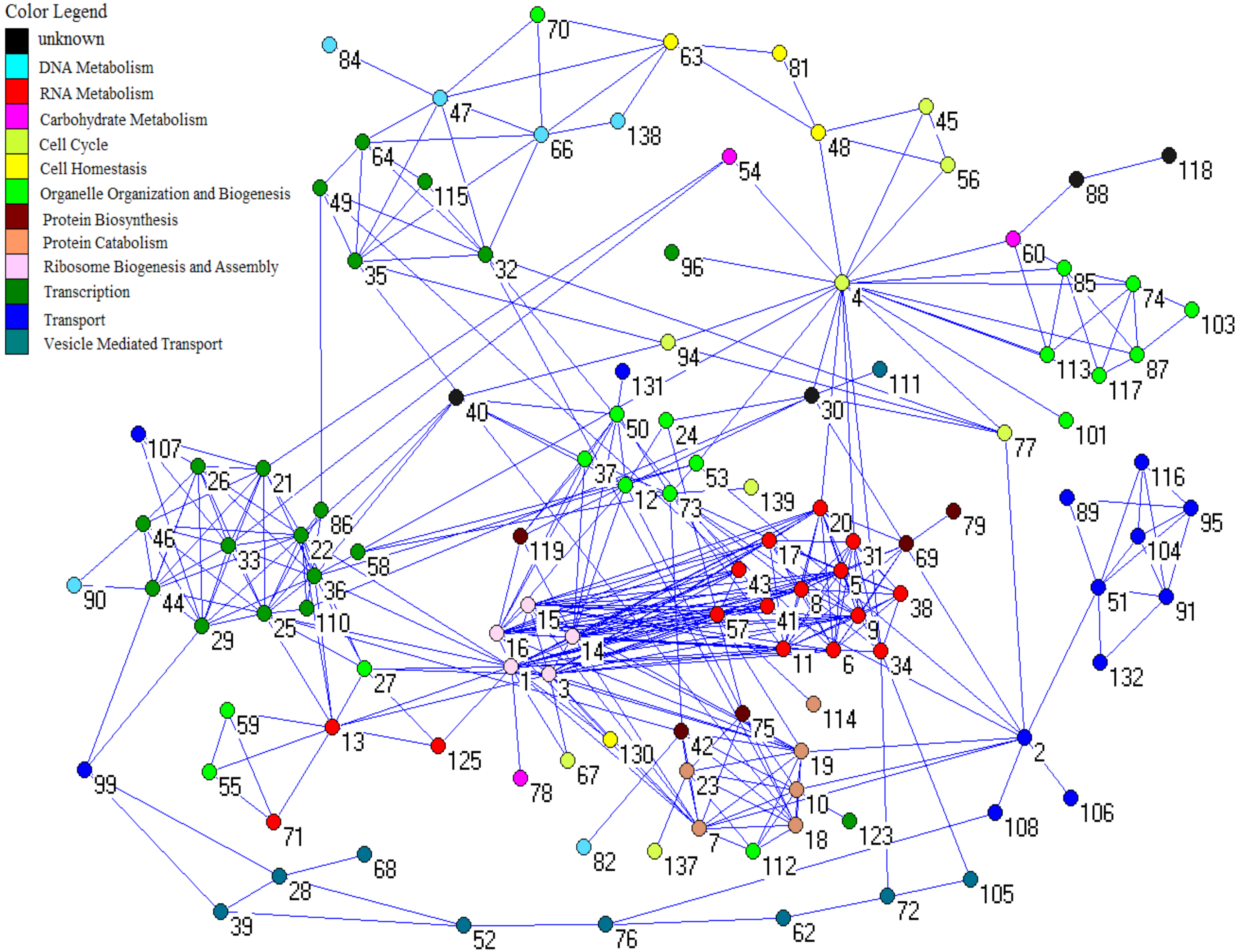
139 Modules Obtained from DIP Yeast core PIN



Example of Module Overlap



Interconnected Module Network



Restricted neighborhood search clust. (RNSC)

- RNSC algorithm - partitions the set of nodes in the network into clusters by using a *cost function* to evaluate the partitioning
- The algorithm starts with a random cluster assignment
- It proceeds by reassigning nodes, so as to maximize the scores of partitions
- At the same time, the algorithm keeps a list of already explored partitions to avoid their reprocessing
- Finally, the clusters are filtered based on their size, density and functional homogeneity

A. D. King, N. Przulj and I. Jurisica, “Protein complex prediction via cost-based clustering,” *Bioinformatics*, 20(17): 3013-3020, 2004.

Restricted neighborhood search clust. (RNSC)

- A *cost function* to evaluate the partitioning:
  - Consider node v in G and clustering C of G
  - α_v is the number of "bad connections" incident with v
  - A bad connection incident to v is an edge that exist between v and a node in a different cluster from that where v is, or one that does not exist between v and node *u* in the same cluster as v
  - The cost function is then:
    - $C_n(G,C) = ½ \sum_{v \in V} \alpha_v$
  - There are other cost functions, too
  - Goal of each cost function: clustering in which the nodes of a cluster are all connected to each other and there are no other connections

A. D. King, N. Przulj and I. Jurisica, "Protein complex prediction via cost-based clustering," *Bioinformatics*, 20(17): 3013-3020, 2004.

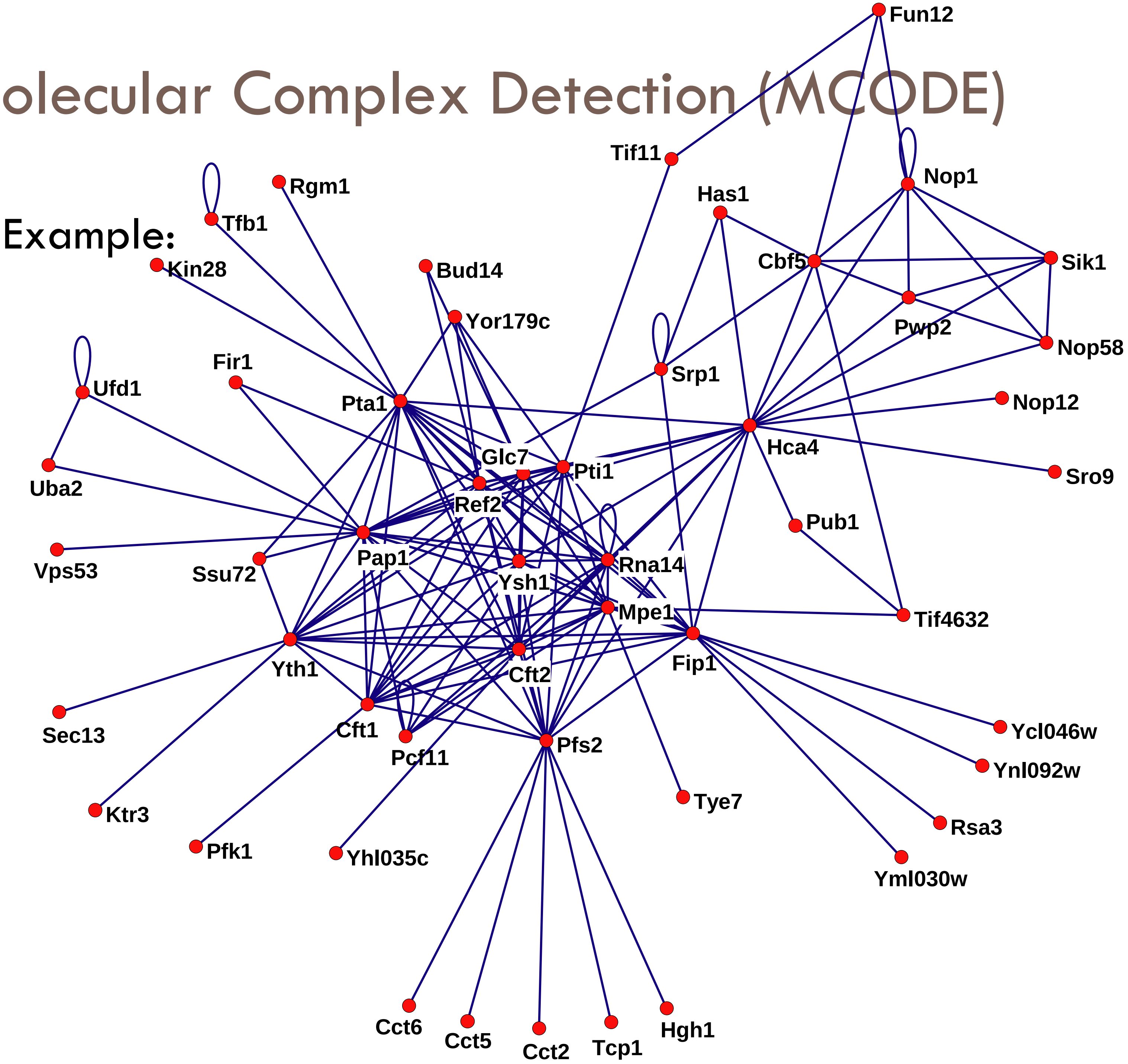
Molecular Complex Detection (MCODE)

- Step 1: node weighting
  - Based on the core clustering coefficient
    - Clustering coefficient of a node: the density of its neighborhood
    - A graph is called a “k-core” if the minimal degree in it is k
    - “Core clustering coefficient” of a node: the density of the k-core of its immediate neighborhood
    - It increases the weights of heavily interconnected graph regions while giving small weights to the less connected vertices, which are abundant in the scale-free networks
- Step 2: the algorithm traverses the weighted graph in a greedy fashion to isolate densely connected regions
- Step 3: The post-processing step filters or adds proteins based on connectivity criteria
- Implementation available as a Cytoscape plug-in

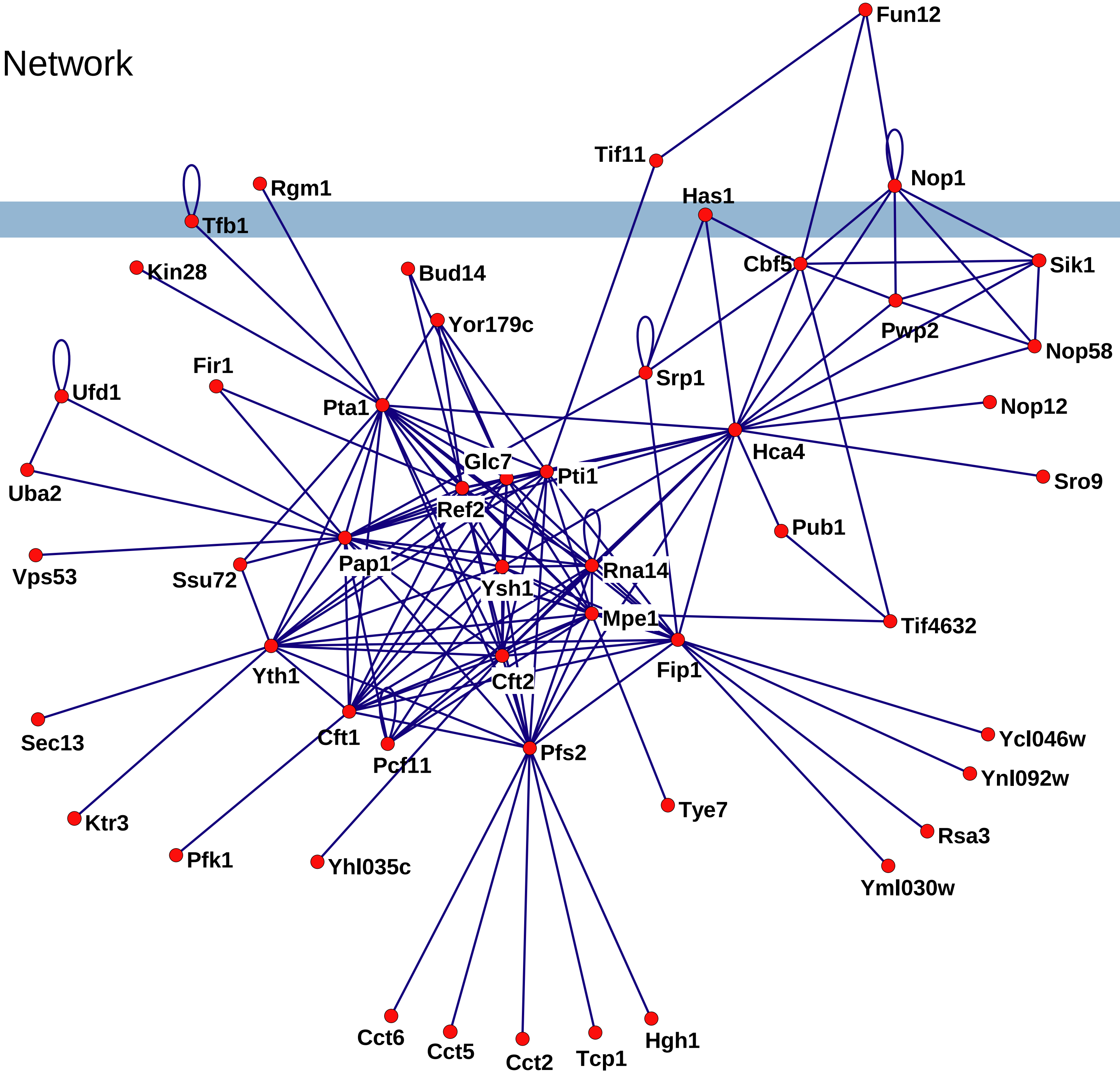
http://baderlab.org/Software/MCODE -- a Cytoscape plugin

Molecular Complex Detection (MCODE)

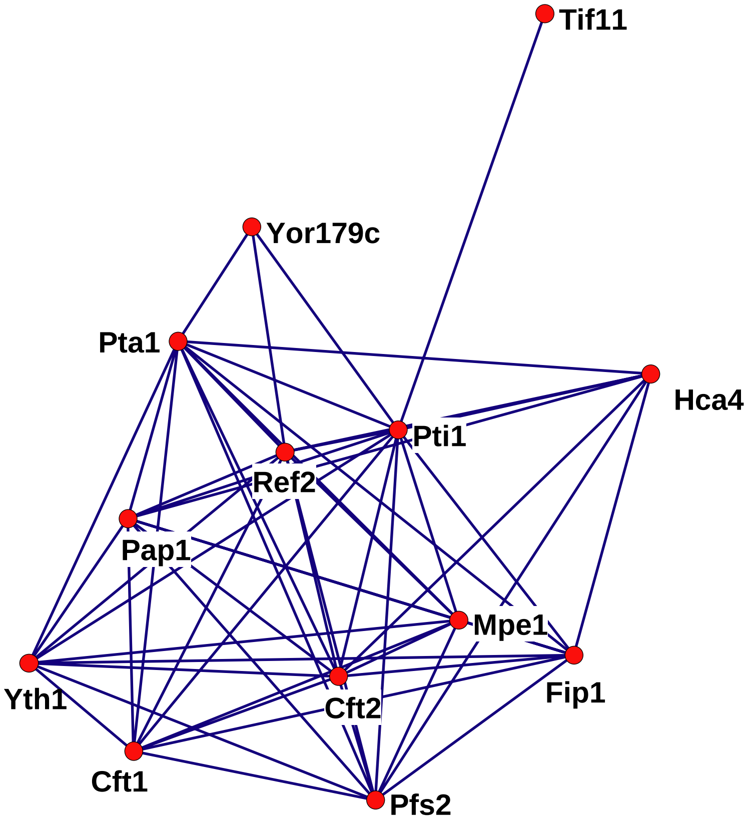
- Example:



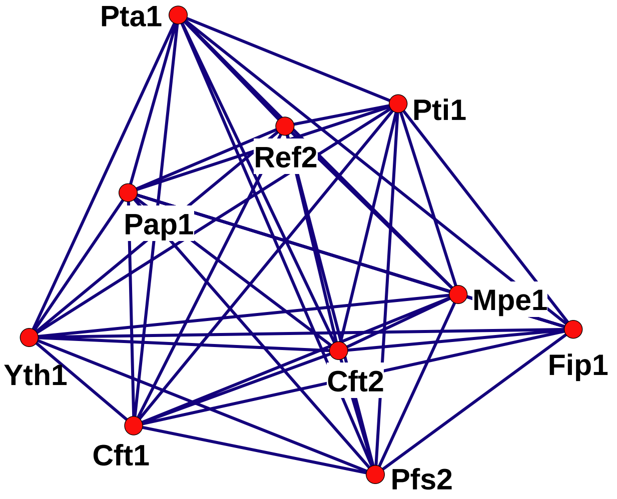
Input Network



Find neighbors of Pti1

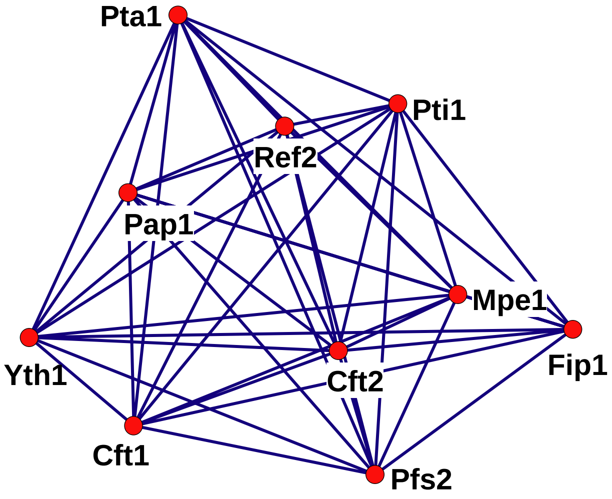


Find highest k-core (8-core)



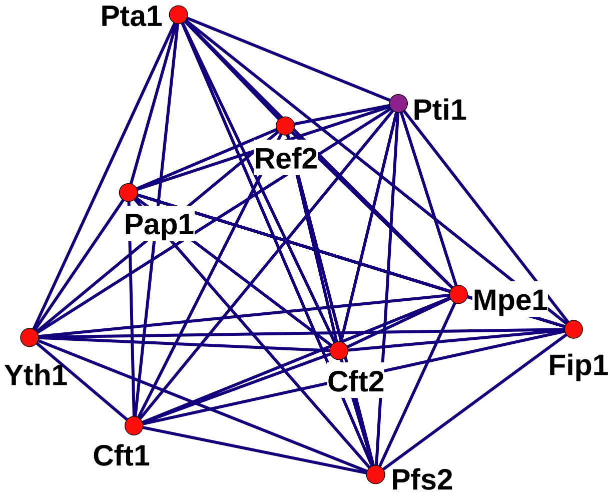
Removes low degree nodes
in power-law networks

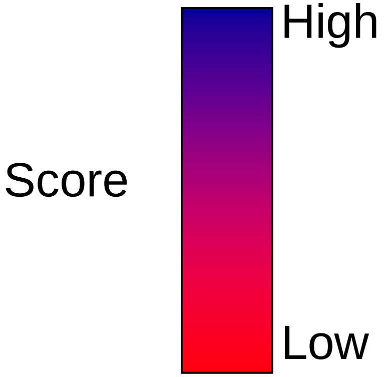
Find graph density



$$\text{Density} = \frac{\text{Number edges}}{\text{Number possible edges}} = 44/55 = 0.8$$

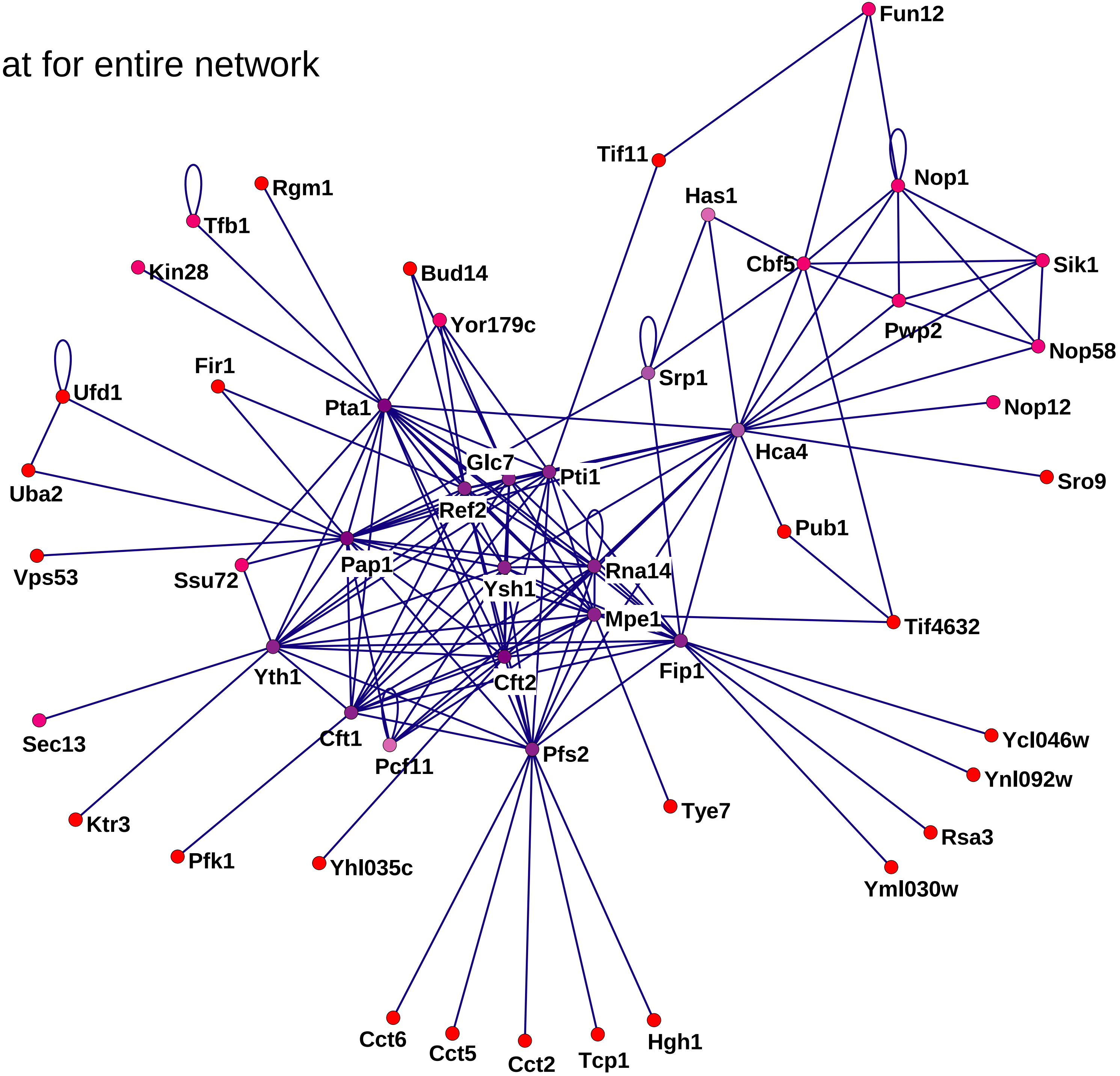
Calculate score for Pti1



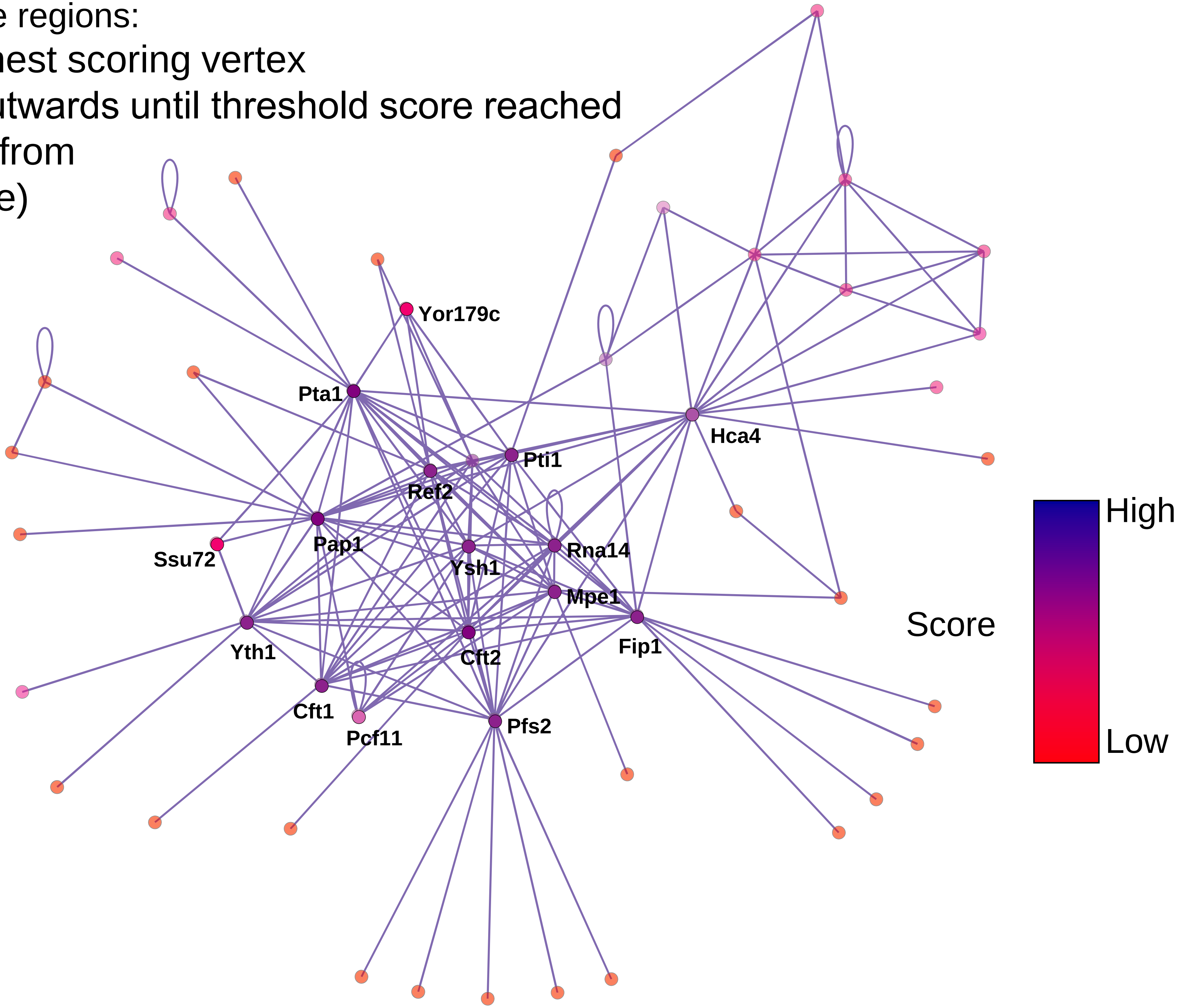


Score = highest k-core * density = 8 * 0.8 = 6.4 = ●

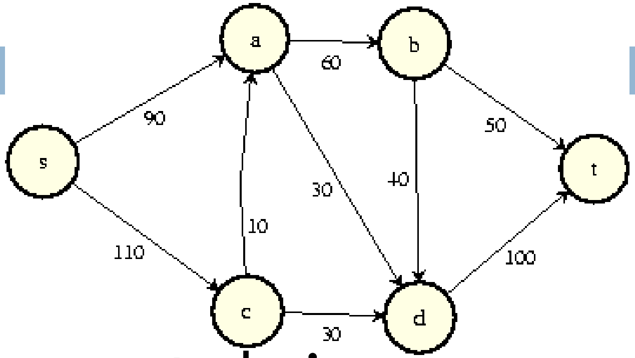
Repeat for entire network



Find dense regions:
-Pick highest scoring vertex
-'Paint' outwards until threshold score reached
(% score from
seed node)



Markov Cluster Algorithm (MCL)



- Network flow
  - Imagine a graph as a network of interconnected pipes
  - Suppose water gets into one or more vertices (sources) from the outside, and can exit the network at certain other vertices (sinks)
  - Then, it will spread in the pipes and reach other nodes, until it exits at sinks
  - The capacities of the edges (i.e., how much the pipe can carry per unit time) and the input at the sources determine the amount of flow along every edge (i.e., how much each pipe actually carries) and the amount exiting at each sink

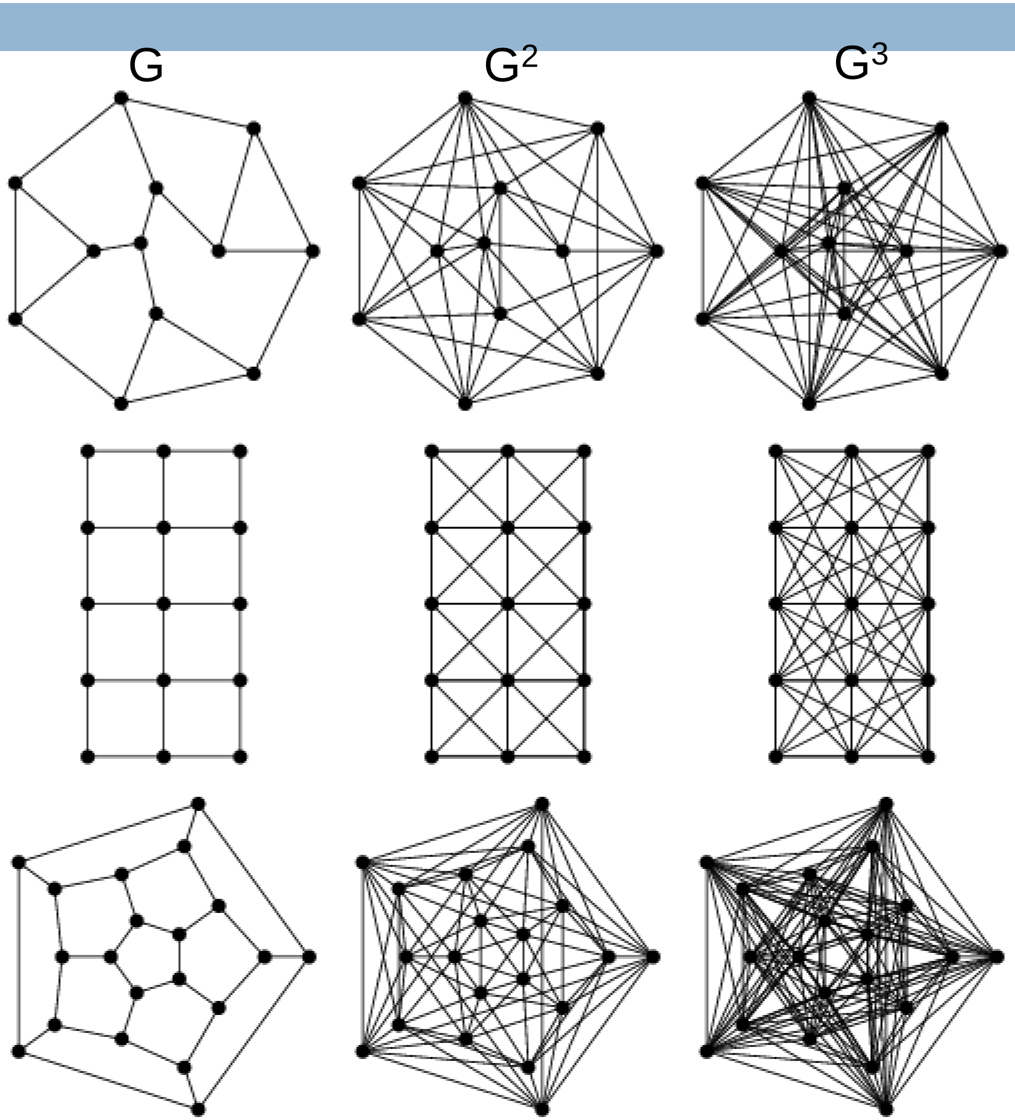
S. M. van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, The Netherlands, 2000.

Markov Cluster Algorithm (MCL)

- Graph power
  - The k^{th} power of a graph G: a graph with the same set of vertices as G and an edge between two vertices iff there is a path of length at most k between them
  - The number of paths of length *k* between any two nodes can be calculated by raising adjacency matrix of G to the exponent *k*
  - Then, G's k^{th} power is defined as the graph whose adjacency matrix is given by the sum of the first *k* powers of the adjacency matrix:

$$\operatorname{adj}\left(G^k\right) = \sum_{i=1}^{k} [\operatorname{adj}(G)]^i,$$

Markov Cluster Algorithm (MCL)

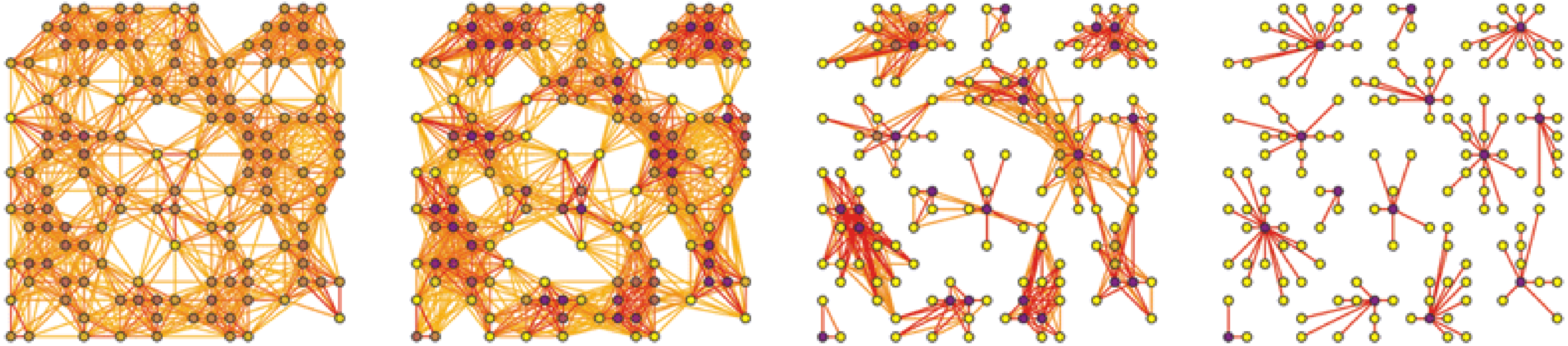


Markov Cluster Algorithm (MCL)

- The MCL algorithm simulates flow on a graph and computes its successive powers to increase the contrast between regions with high flow and regions with a low flow
- This process can be shown to converge towards a partition of the graph into high-flow regions separated by regions of no flow

- Very efficient for PPI networks
  - Brohee S, van Helden J: Evaluation of clustering algorithms for protein-protein interaction networks. *BMC bioinformatics* 2006, 7:488.
  - Vlasblom, J, Wodak, SJ: Markov clustering versus affinity propagation for the partitioning of protein interaction graphs, *BMC Bioinformatics* 2009, 10:99.

Markov Cluster Algorithm (MCL)



Flow between different dense regions that are sparsely connected eventually “evaporates,” showing cluster structure present in the input graph.

Correctness of methods

- ***Clustering*** is used for **making predictions:**
  - E.g., protein function, involvement in disease, interaction prediction
- Other methods are used for ***classifying*** the data (have disease or not) and **making predictions**
- Have to evaluate the correctness of the predictions made by the approach
- Commonly used method for this is ***ROC Curves***

Correctness of methods

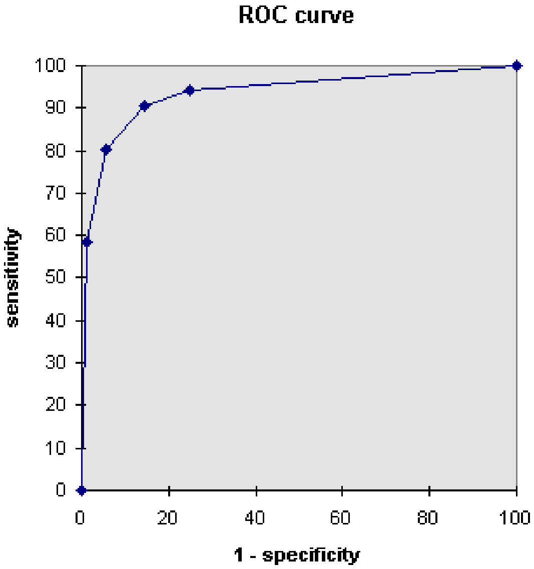
Definitions (e.g., for PPIs):

- A **true positive (TP)** interaction:
  - an interaction exists in the cell and is discovered by an experiment (biological or computational).
- A **true negative (TN)** interaction:
  - an interaction does not exist and is not discovered by an experiment.
- A **false positive (FP)** interaction:
  - an interaction does not exist in the cell, but is discovered by an experiment.
- A **false negative (FN)** interaction:
  - an interaction exists in the cell, but is not discovered by an experiment.

Correctness of methods

- If TP stands for true positives, FP for false positives, TN for true negatives, and FN for false negatives, then:
- *Sensitivity = TP / (TP + FN)*
- *Specificity = TN / (TN + FP)*
- *Sensitivity* measures the fraction of items out of all possible ones that truly exist in the biological system that our method successfully identifies (fraction of correctly classified existing items)
- *Specificity* measures the fraction of the items out of all items that truly do not exist in the biological system for which our method correctly determines that they do not exist (fraction of

  correctly classified non-existing items)
- Thus, *1-Specificity* measures the fraction of all non-existing items in the system that are incorrectly identified as existing



ROC Curve

□ **Receiver Operating Curves (ROC curves)** provide a standard measure of the ability of a test to correctly classify objects.

□ E.g., the biomedical field uses ROC curves extensively to assess the efficacy of diagnostic tests in discriminating between healthy and diseased individuals.

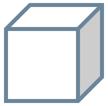
□ **ROC curve** is a graphical plot of the **true positive rate**, i.e., ***sensitivity***, vs. **false positive rate**, i.e., (***1*−*specificity***), for a binary classifier system as its discrimination threshold is varied (see above for definitions).

□ It shows the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity).

□ The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test; the closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test. The **area under the curve (AUC)** is a measure of a test's accuracy.

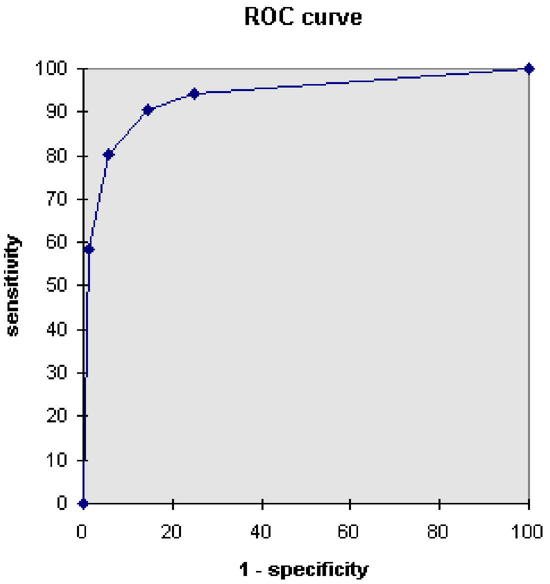
ROC curve

Example:



- Embed nodes of a PPI network into 3-D Euclidean unit box

  (use MDS – knowledge of MDS not required in this class, see reference in the footer, if interested)
- Like in GEO, choose a radius *r* to determine node connectivity
- Vary *r* between *0* and *sqrt(3)* (diagonal of the box)
  - *r=0* makes a graph with no edges (TP=0, FP=0)
  - *r=sqrt(3)* makes a complete graph (all possible edges, FN=TN=0)
- For each *r* in *[0, sqrt(3)]:*
  - measure TP, TN, FP, FN
  - compute *sensitivity* and *1- specificity*
  - draw the point
- Set of these points is the ROC curve



Sensitivity = TP / (TP + FN)
Specificity = TN / (TN + FP)

Note:

- For *r=0*, *sensitivity=0* and *1-specificity=0*, since TP=0, FP=0 (no edges)
- For *r=sqrt(3)*, *sensitivity=1* and *1-specificity=1 (or 100%)*, since FN=0, TN=0

D. J. Higham, M. Rasajski, N. Przulj, "Fitting a Geometric Graph to a Protein-Protein Interaction Network", *Bioinformatics*, 24(8), 1093-1099, 2008.

Precision and recall

- Information about true negatives is often not available
- Precision - a measure of exactness
- Recall - a measure of completeness

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

Sensitivity = *TP / (TP + FN)*

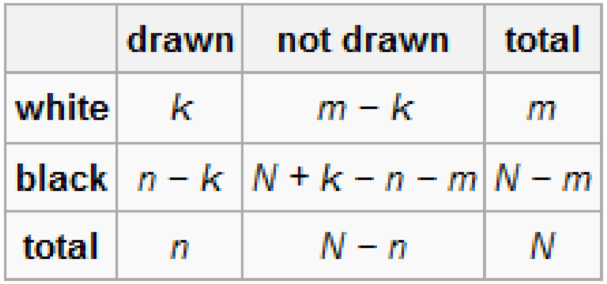
Specificity = *TN / (TN + FP)*

- E.g., given that we produce *n* cancer gene predictions
  - Precision is the number of known cancer genes in our *n* predictions, divided by *n*
  - Recall is the number of known cancer genes in our *n* predictions divided by the total number of known cancer genes
- F-score – measures test accuracy, weighted average of precission and recall (in [0,1]):

$$F = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

Hypergeometric distribution

- Probability distribution that describes the number of successes in a sequence of *n* draws from a finite population of size N *without* replacement
  - For draws with replacement, use binomial distribution



	drawn	not drawn	total
white	k	$m - k$	m
black	$n - k$	$N + k - n - m$	$N - m$
total	n	$N - n$	N

$$P(X = k) = \frac{\binom{m}{k}\binom{N-m}{n-k}}{\binom{N}{n}}$$

N -

m - the number of objects out of n objects with a given "function" (color)

n - number of draws from N (e.g., the size of a cluster)

k - the number of objects out of n objects that have the given function

- To get the <u>enrichment p-value</u> for a cluster of size n, sum over i=k,k+1,…,m

- Use *hygecdf* function in *Matlab* (but use *1-hygecdf(…)),* since it computes probability to get 0 to k elements of a given function

Network Topology → Biology

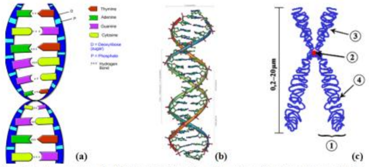
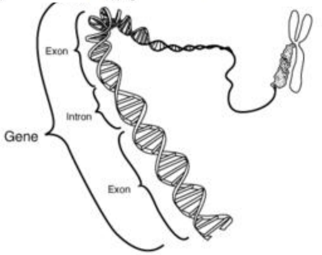
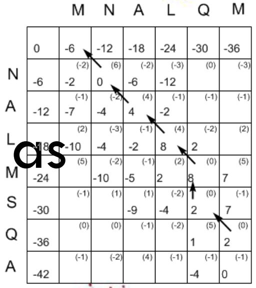


Figure 1.1: The structure of a DNA double helix: (a) an illustration of complementary nucleotides pairs; (b) a bio-chemical view of DNA; (c) organization of DNA into chromosome.

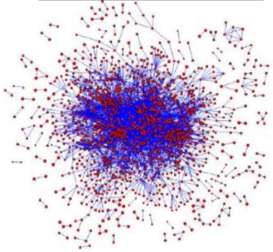
Motivation



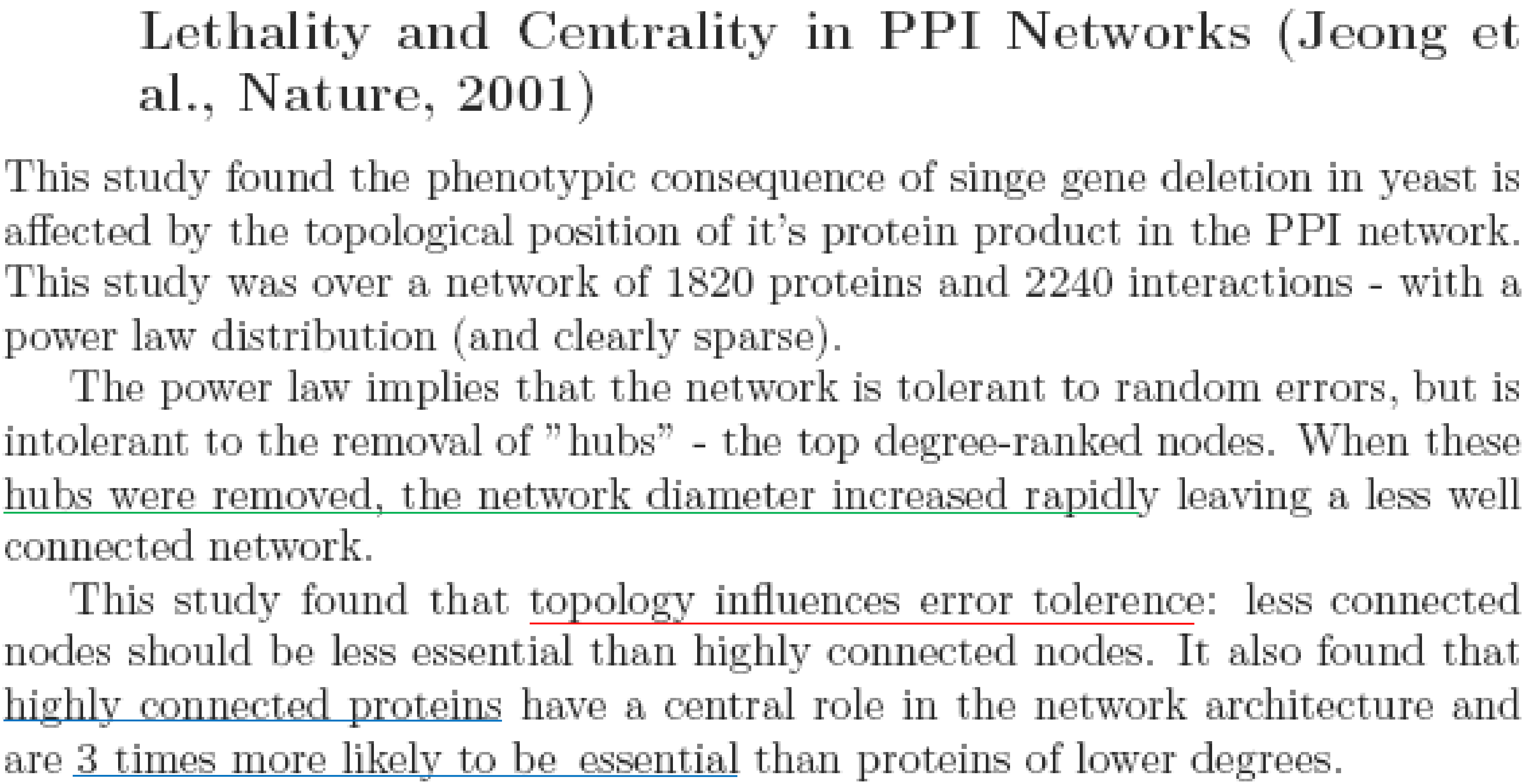
- □ Genetic sequence research – valuable insights
- □ Genes produce thousands of different proteins
- □ Proteins interact in complex ways to perform a function
  - They do not act in isolation
- □ Biological network research – at least as valuable insights as genetic sequence research
- □ However, the field is still in its infancy:
  - Incomplete/noisy network data
  - Computational intractability of many graph theoretic problems
- □ Defining the relationship between network topology and biological function – one of the most important problems in post-genomic era



		M	N	A	L	Q	M
	0	-6	-12	-18	-24	-30	-36
N	-6	(-2) -2	(6) 0	(-2) -6	(-3) -12	(0)	(-3)
A	-12	(-1) -7	(-2) -4	(4) 4	(-1) -2	(-1)	(-1)
L	-18	(2) -10	(-3) -4	(-1) -2	(4) 8	(-2) 2	(2)
M	-24	(5)	(-2) -10	(-1) -5	(2) 2	(0) 8	(5) 7
S	-30	(-1)	(1)	(1) -9	(-2) -4	(0) 2	(-1) 7
Q	-36	(0)	(0)	(-1)	(-2)	(5) 1	(0) 2
A	-42	(-1)	(-2)	(4)	(-1)	(-1) -4	(-1) 0



Network Topology → Biology



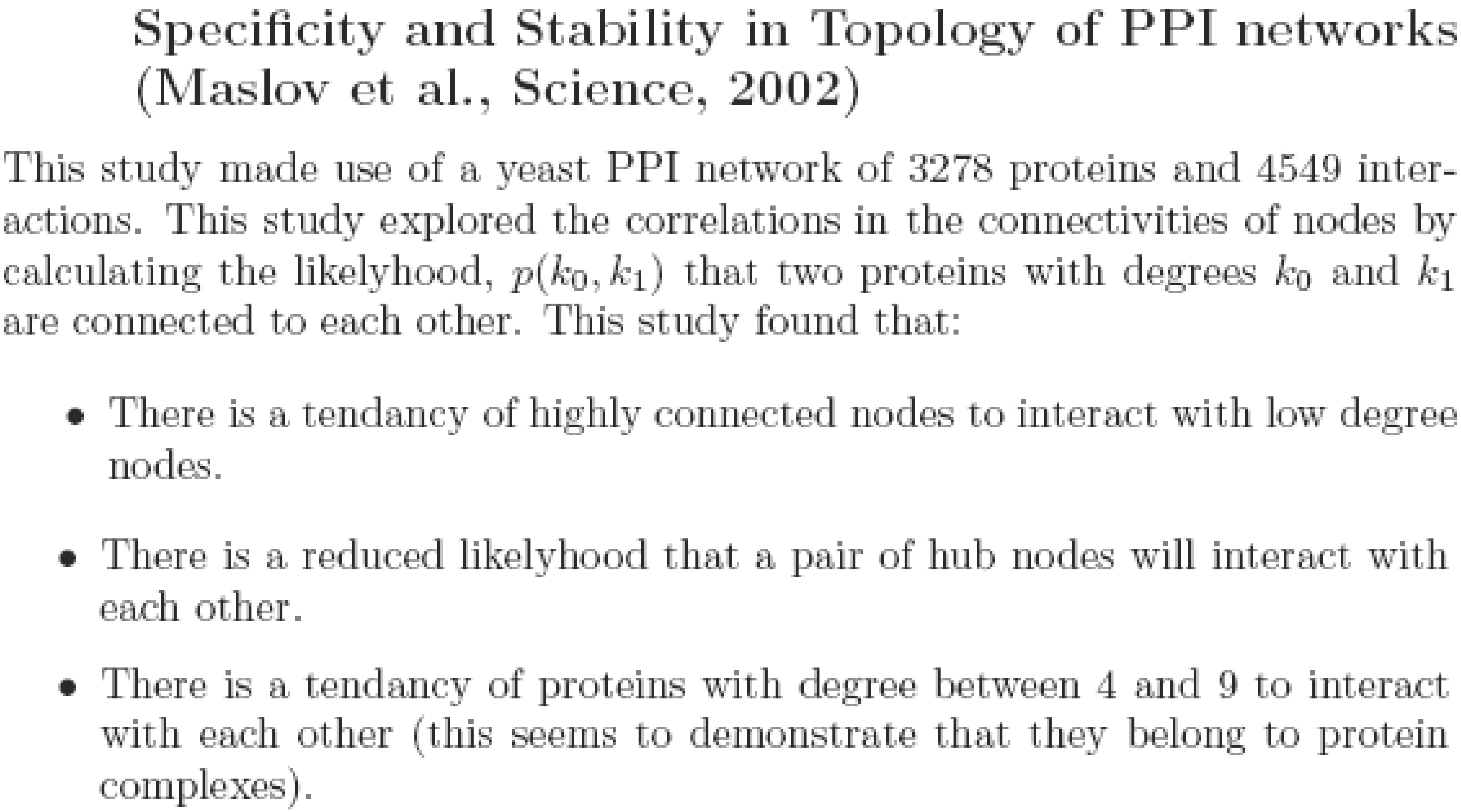
Lethality and Centrality in PPI Networks (Jeong et al., Nature, 2001)

This study found the phenotypic consequence of singe gene deletion in yeast is affected by the topological position of it's protein product in the PPI network. This study was over a network of 1820 proteins and 2240 interactions - with a power law distribution (and clearly sparse).

The power law implies that the network is tolerant to random errors, but is intolerant to the removal of ”hubs” - the top degree-ranked nodes. When these hubs were removed, the network diameter increased rapidly leaving a less well connected network.

This study found that topology influences error tolerence: less connected nodes should be less essential than highly connected nodes. It also found that highly connected proteins have a central role in the network architecture and are 3 times more likely to be essential than proteins of lower degrees.

Network Topology → Biology

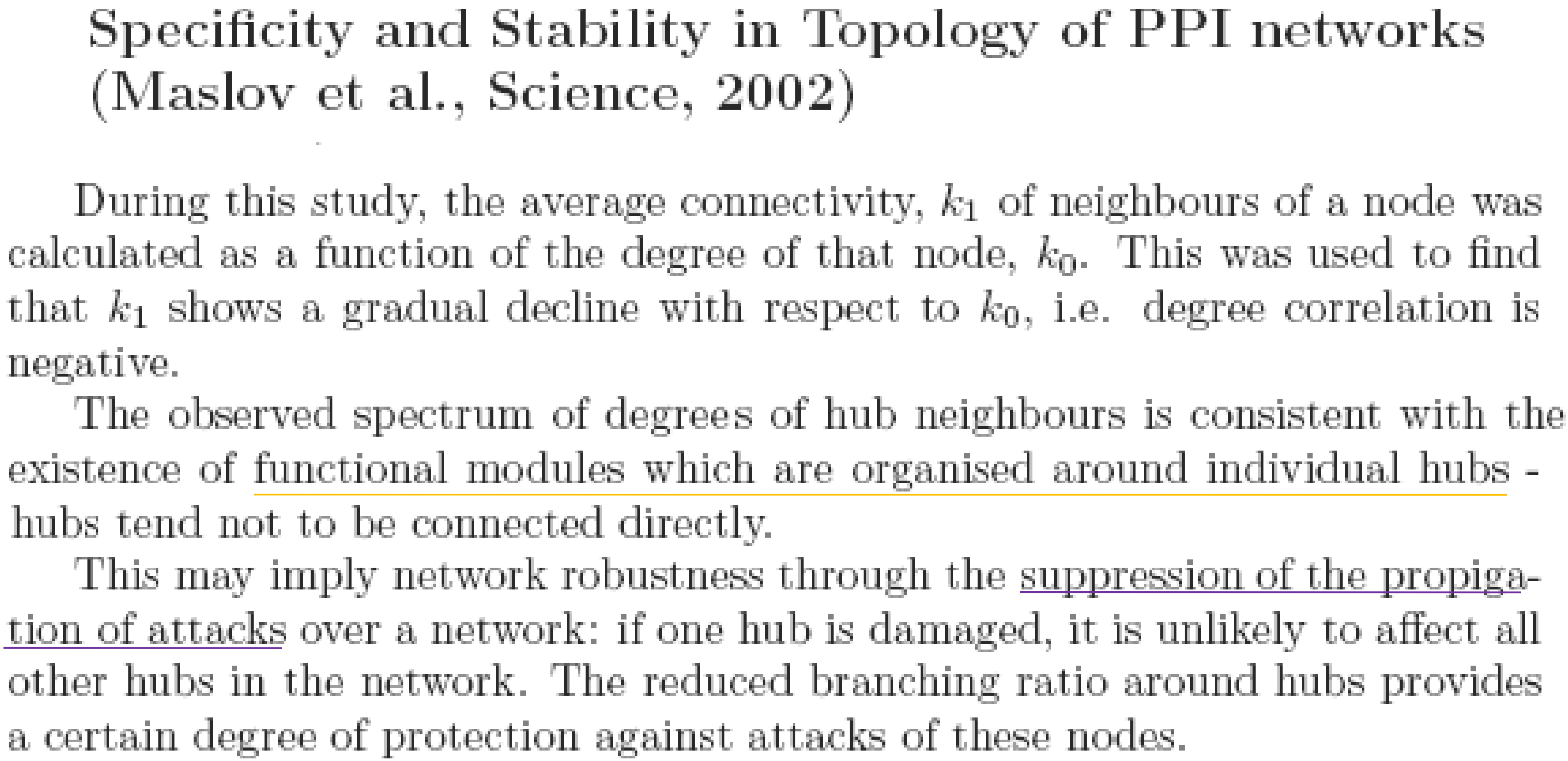


Specificity and Stability in Topology of PPI networks (Maslov et al., Science, 2002)

This study made use of a yeast PPI network of 3278 proteins and 4549 interactions. This study explored the correlations in the connectivities of nodes by calculating the likelyhood, $p(k_0, k_1)$ that two proteins with degrees k_0 and k_1 are connected to each other. This study found that:

- There is a tendancy of highly connected nodes to interact with low degree nodes.
- There is a reduced likelyhood that a pair of hub nodes will interact with each other.
- There is a tendancy of proteins with degree between 4 and 9 to interact with each other (this seems to demonstrate that they belong to protein complexes).

Network Topology → Biology

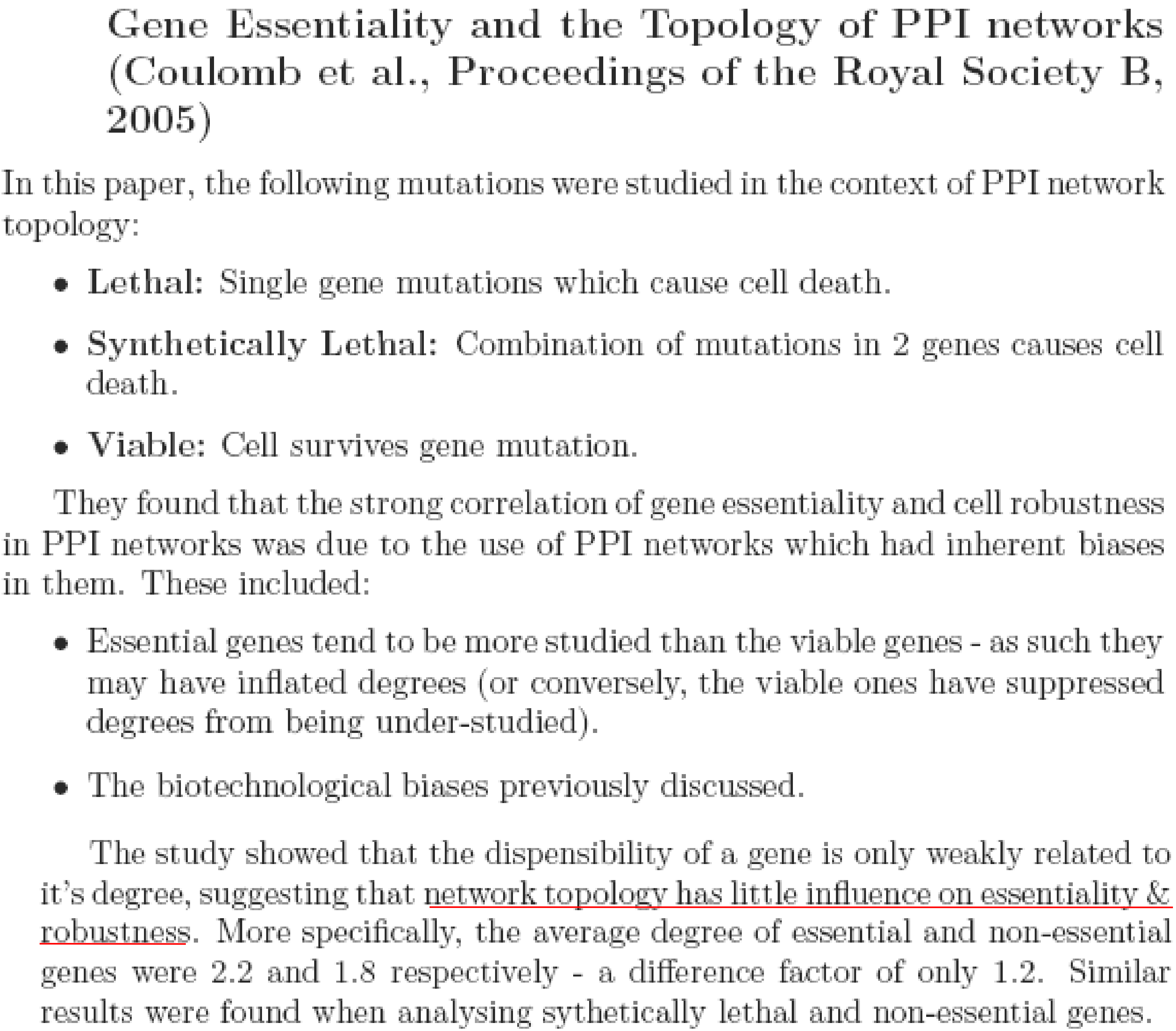


Specificity and Stability in Topology of PPI networks (Maslov et al., Science, 2002)

During this study, the average connectivity, k_1 of neighbours of a node was calculated as a function of the degree of that node, k_0. This was used to find that k_1 shows a gradual decline with respect to k_0, i.e. degree correlation is negative.

The observed spectrum of degrees of hub neighbours is consistent with the existence of functional modules which are organised around individual hubs - hubs tend not to be connected directly.

This may imply network robustness through the suppression of the propigation of attacks over a network: if one hub is damaged, it is unlikely to affect all other hubs in the network. The reduced branching ratio around hubs provides a certain degree of protection against attacks of these nodes.



Gene Essentiality and the Topology of PPI networks (Coulomb et al., Proceedings of the Royal Society B, 2005)

In this paper, the following mutations were studied in the context of PPI network topology:

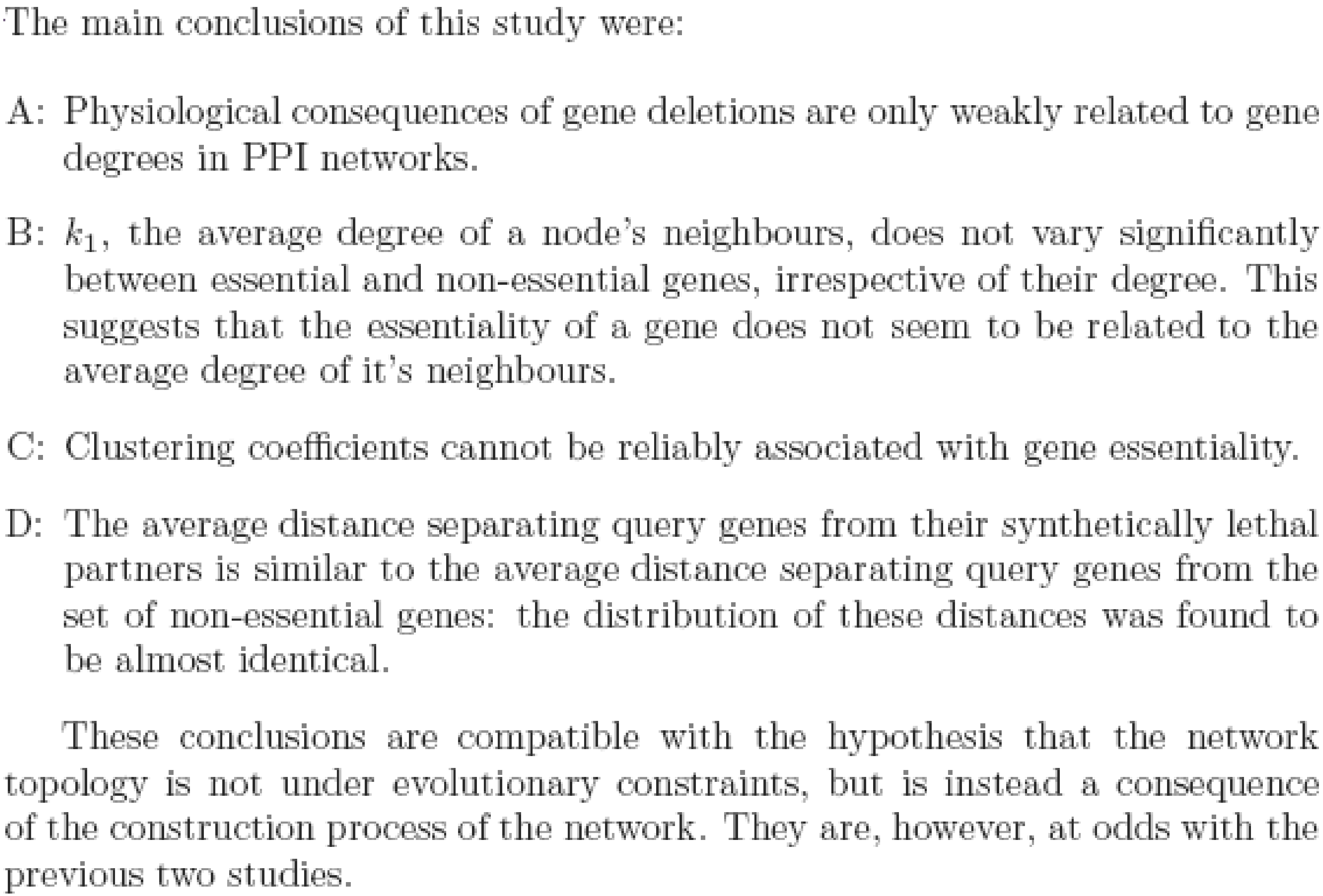
- **Lethal:** Single gene mutations which cause cell death.
- **Synthetically Lethal:** Combination of mutations in 2 genes causes cell death.
- **Viable:** Cell survives gene mutation.

They found that the strong correlation of gene essentiality and cell robustness in PPI networks was due to the use of PPI networks which had inherent biases in them. These included:

- Essential genes tend to be more studied than the viable genes - as such they may have inflated degrees (or conversely, the viable ones have suppressed degrees from being under-studied).
- The biotechnological biases previously discussed.

The study showed that the dispensibility of a gene is only weakly related to it's degree, suggesting that network topology has little influence on essentiality & robustness. More specifically, the average degree of essential and non-essential genes were 2.2 and 1.8 respectively - a difference factor of only 1.2. Similar results were found when analysing sythetically lethal and non-essential genes.

Gene Essentiality and the Topology of PPI networks (Coulomb et al., Proceedings of the Royal Society B, 2005)



The main conclusions of this study were:

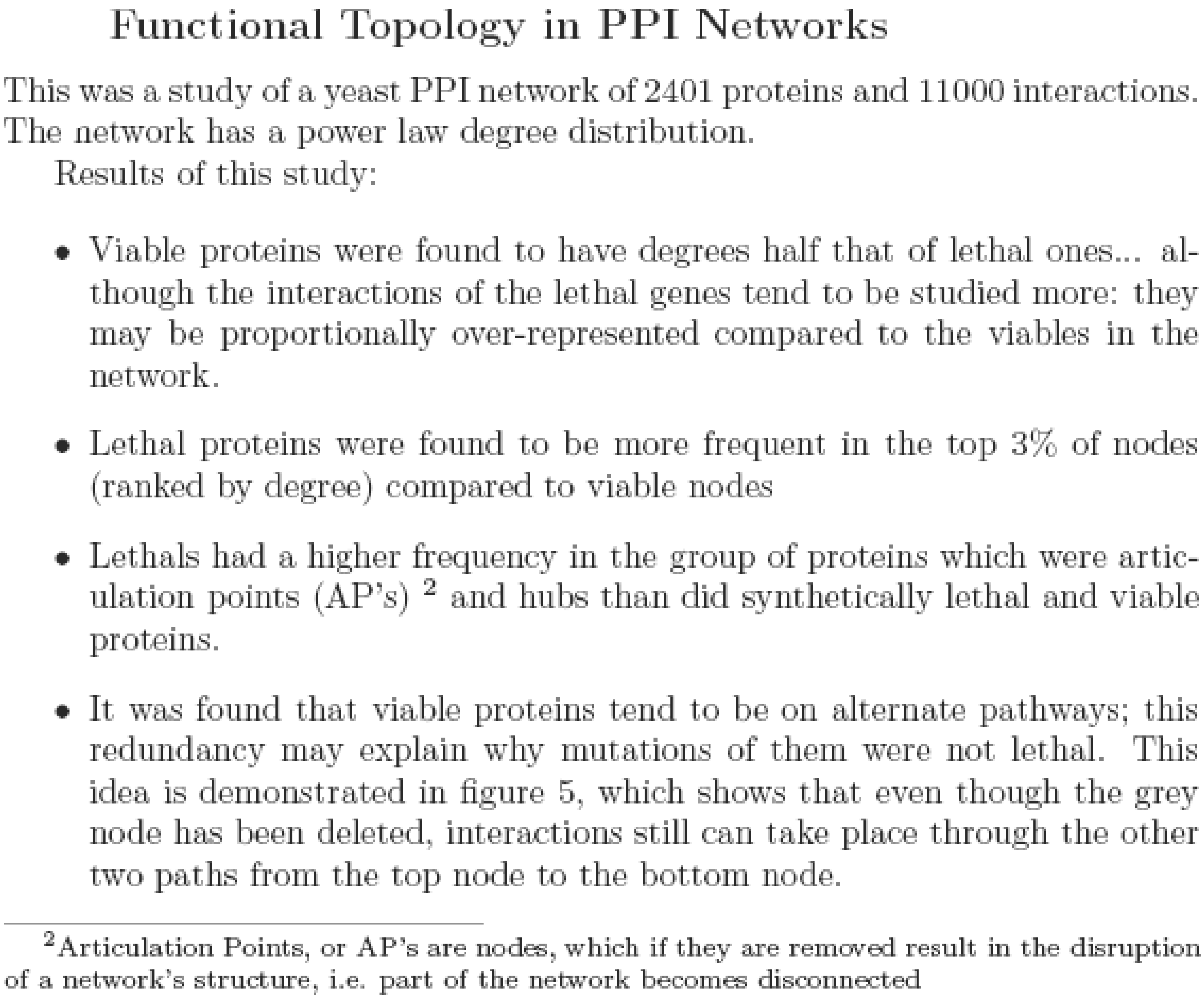
A: Physiological consequences of gene deletions are only weakly related to gene degrees in PPI networks.

B: k_1, the average degree of a node's neighbours, does not vary significantly between essential and non-essential genes, irrespective of their degree. This suggests that the essentiality of a gene does not seem to be related to the average degree of it's neighbours.

C: Clustering coefficients cannot be reliably associated with gene essentiality.

D: The average distance separating query genes from their synthetically lethal partners is similar to the average distance separating query genes from the set of non-essential genes: the distribution of these distances was found to be almost identical.

These conclusions are compatible with the hypothesis that the network topology is not under evolutionary constraints, but is instead a consequence of the construction process of the network. They are, however, at odds with the previous two studies.



Functional Topology in PPI Networks

This was a study of a yeast PPI network of 2401 proteins and 11000 interactions. The network has a power law degree distribution.

Results of this study:

- Viable proteins were found to have degrees half that of lethal ones... although the interactions of the lethal genes tend to be studied more: they may be proportionally over-represented compared to the viables in the network.
- Lethal proteins were found to be more frequent in the top 3% of nodes (ranked by degree) compared to viable nodes
- Lethals had a higher frequency in the group of proteins which were articulation points (AP's) [2] and hubs than did synthetically lethal and viable proteins.
- It was found that viable proteins tend to be on alternate pathways; this redundancy may explain why mutations of them were not lethal. This idea is demonstrated in figure 5, which shows that even though the grey node has been deleted, interactions still can take place through the other two paths from the top node to the bottom node.

[2] Articulation Points, or AP's are nodes, which if they are removed result in the disruption of a network's structure, i.e. part of the network becomes disconnected

Network Topology → Biology

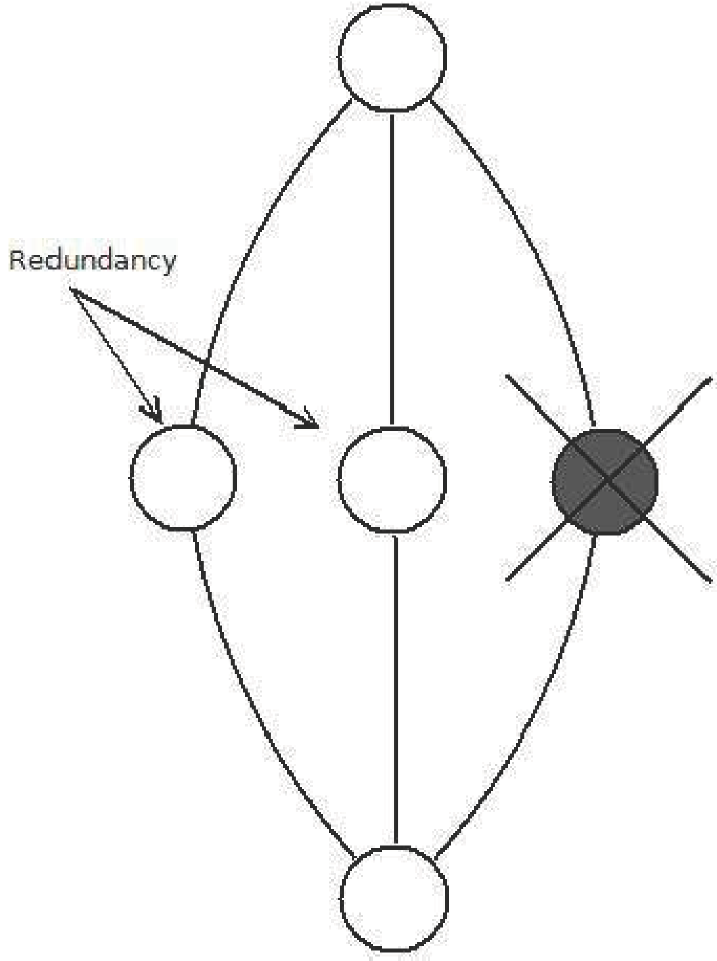
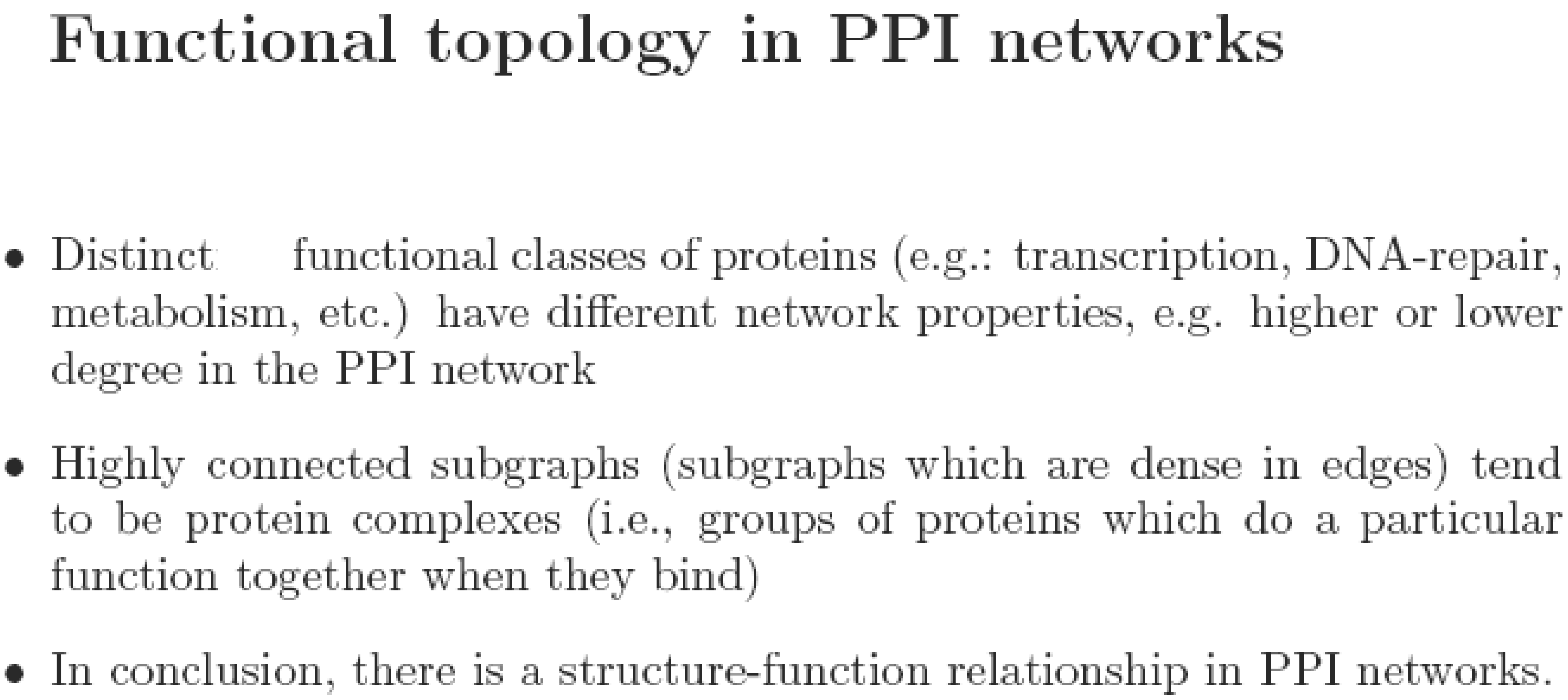


Figure 5: Redundancy in PPI networks

Network Topology → Biology

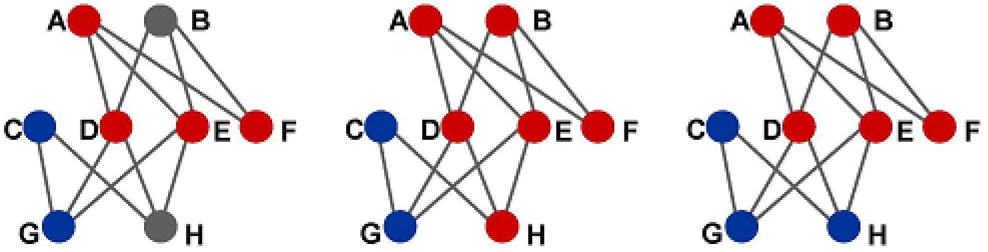


Functional topology in PPI networks

- Distinct functional classes of proteins (e.g.: transcription, DNA-repair, metabolism, etc.) have different network properties, e.g. higher or lower degree in the PPI network
- Highly connected subgraphs (subgraphs which are dense in edges) tend to be protein complexes (i.e., groups of proteins which do a particular function together when they bind)
- In conclusion, there is a structure-function relationship in PPI networks.

Protein function prediction

- One of the major challenges in the post-genomic era:
  - Relationship between PPI network topology and biological function?
- Methods for protein function prediction:
  - Direct methods: proteins that are closer in the PPI network are more likely to have similar function
    - Majority-rule (Schwikowski et al., 2000)
    - n-neighborhood (Hishigaki et al., 2001)
    - 1- and 2-neighborhood with different weights (Chua et al., 2006)
    - Global optimization strategies (Vazquez et al., 2003)
    - “Functional flow” (Nabieva et al., 2005)

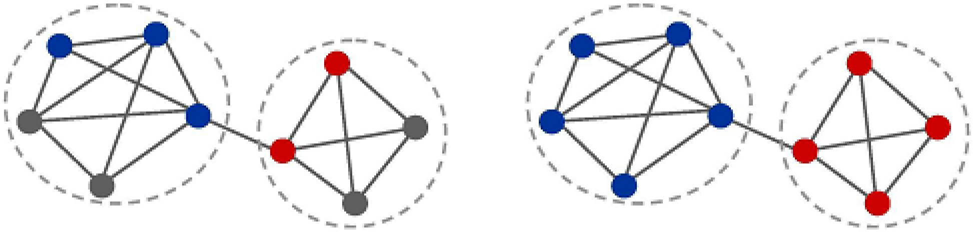


Direct neighborhood

Shared neighborhood

Protein function prediction

- One of the major challenges in the post-genomic era:
  - Relationship between PPI network topology and biological function?
- Methods for protein function prediction:
  - Cluster-based methods: partition the network into clusters (i.e., functional modules) and assign the entire cluster with a function
    - Detecting dense network regions:
      - MCODE (Bader and Hogue, 2003), HCS (Przulj et al., 2003); RNSC (King et al., 2004)…
    - Hierarchical clustering:
      - The key step: defining the similarity measure between protein pairs
      - E.g., the shortest path length (Arnau et al.2005) or Czekanowski-Dice distance (Brun et al., 2004)

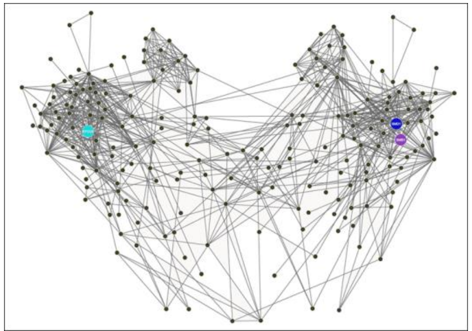


Uncovering Biological Network Function via Graphlet Degree Signatures: (Milenkovic, and Przulj, Cancer Informatics, 2008)

- Biological function of a protein and its local network structure (as described by graphlet degree vectors, a.k.a. "node signatures," covered in previous classes) are closely related.

- Proteins with topologically similar neighborhoods are clustered together and the resulting clusters are statistically significantly enriched in:
  - protein complexes
  - biological function
  - sub-cellular localization
  - tissue expression (in human)
  - involvement in (human) disease

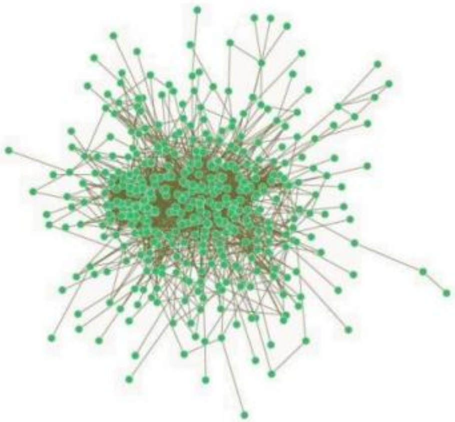
- Used to predict function and new proteins involved in disease.

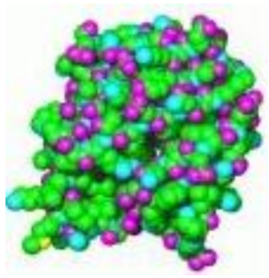


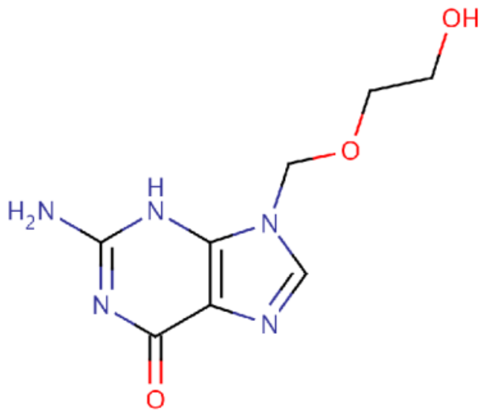
Disease-genes and drug-targets

- Emerging research field: understanding the networks underlying human disease
  - Analyzing topological properties of disease genes in PPI networks & identifying novel disease genes
  - Defining the relationship between disorders and disease genes

ıtionship between drugs

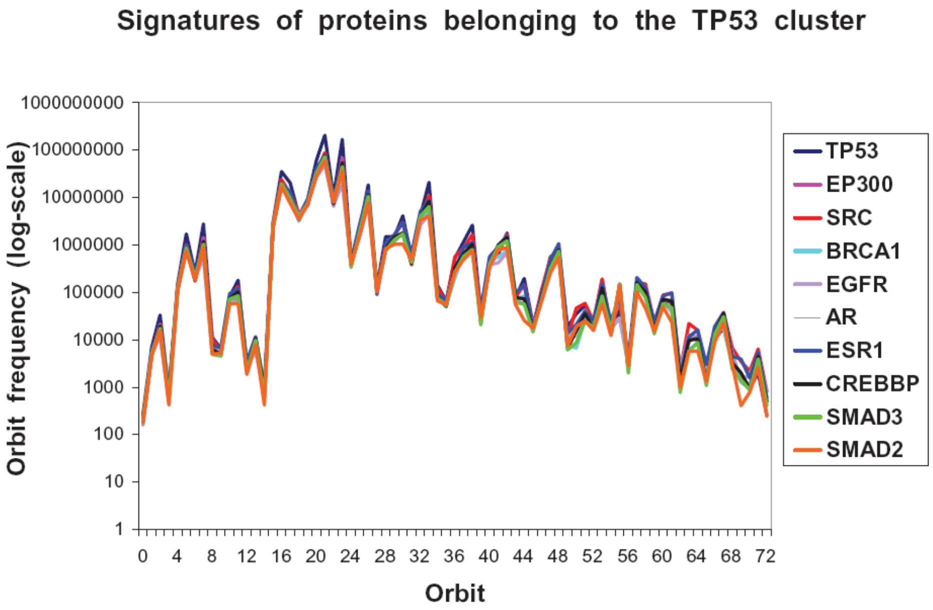






Topological properties of disease genes

- Cancer genes: greater connectivity and centrality (Jonsson and Bates, '06)
- Only essential disease genes show higher connectivities (Goh et al., '07)
- Disease genes have similar graphlet degree signatures in PPI networks (Milenkovic and Przulj, '08)



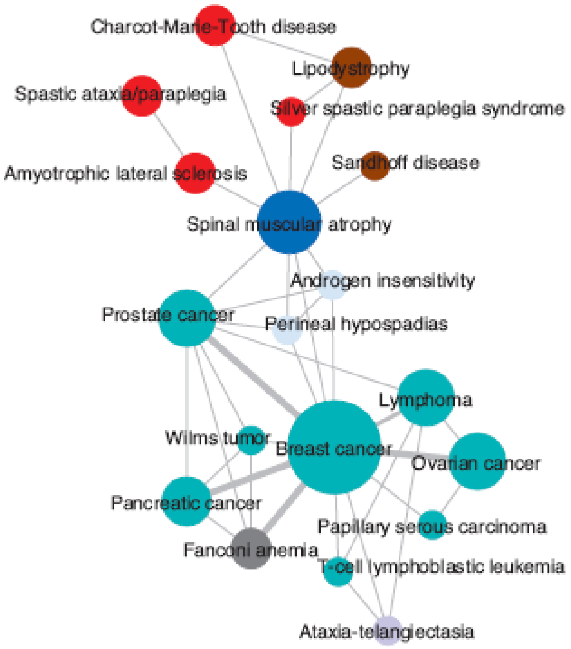
Cancer gene identification

- Network neighbors of cancer genes also involved in cancer (Aragues et al., ‘08)
- Do the genes that are involved in cancer have similar topological signatures without necessarily being adjacent in the network? (Milenkovic et al., ‘10)
  - 96% of signature-similar known cancer gene pairs not direct neighbors
- Apply a series of clustering methods to proteins’ signature similarities
  - Hierarchical clustering (HIE)
  - K-medoids (KM)
  - K-nearest neighbors (KNN)
  - Signature-threshold based clustering (ST)
- Analyze if the obtained clusters are statistically significantly enriched with known cancer genes
- Predict novel cancer gene candidates
- Measure prediction accuracy of our approach
- Validate predictions in the literature and biologically
- Demonstrate superiority over other approaches (Aragues et al. ‘08)

The disease network (Goh et al., 2007)

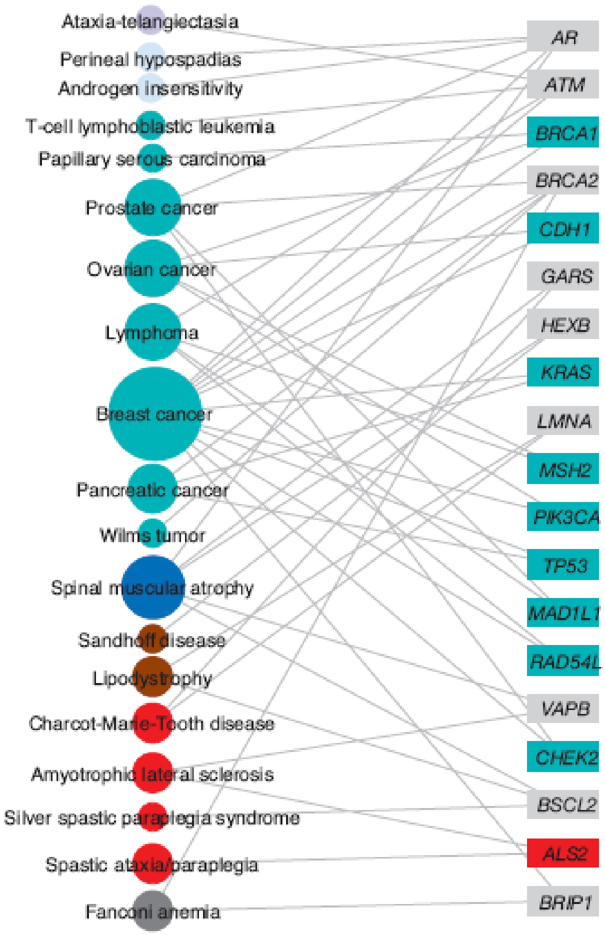
- The challenge: many-to-many relationships
- Global view from a higher level of cellular organization
  - The “disease phenome”: a systematic linkage of all genetic disorders
  - The “disease genome”: the complete list of disease genes
  - The “diseasome”, the combined set of all known associations between disorders and disease genes.
- Two projections of the diseasome:
  - The human disease network (HDN)
  - The disease gene network (DGN)
  - Both projections are far from being disconnected
  - Clustering of disorders and disease genes
- Overlaying DGN with the human PPI network
  - Overlap of 290 interactions
  - Genes involved in the same disease tend to interact in the PPI network
  - Only essential disease genes are topologically and functionally central

Human Disease Network (HDN)

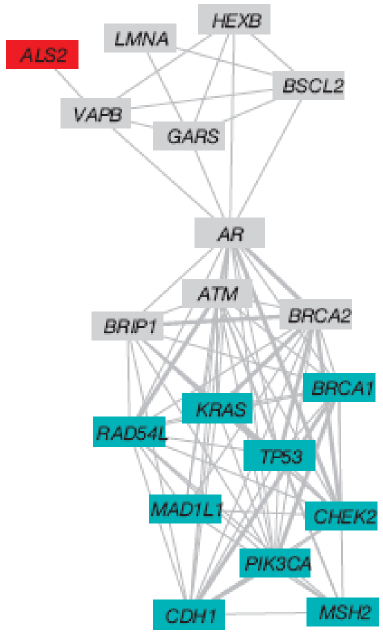


DISEASOME

disease phenome **disease genome**

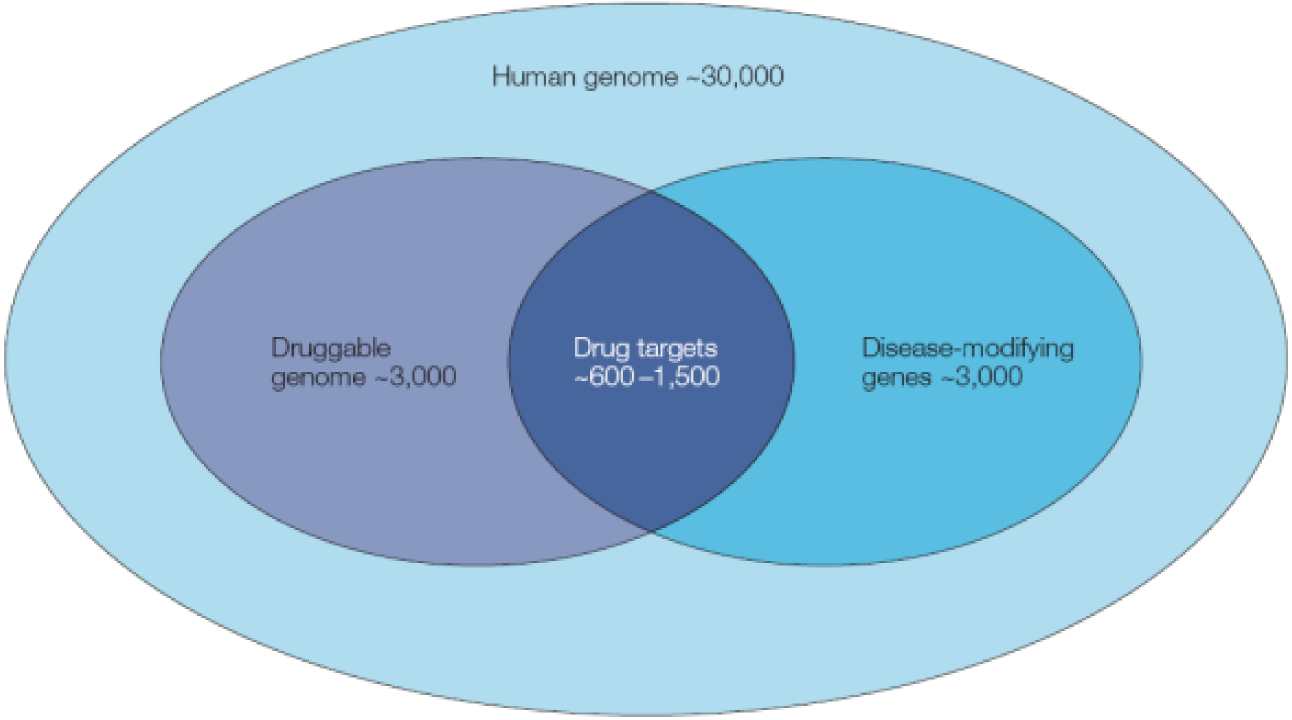


Disease Gene Network (DGN)



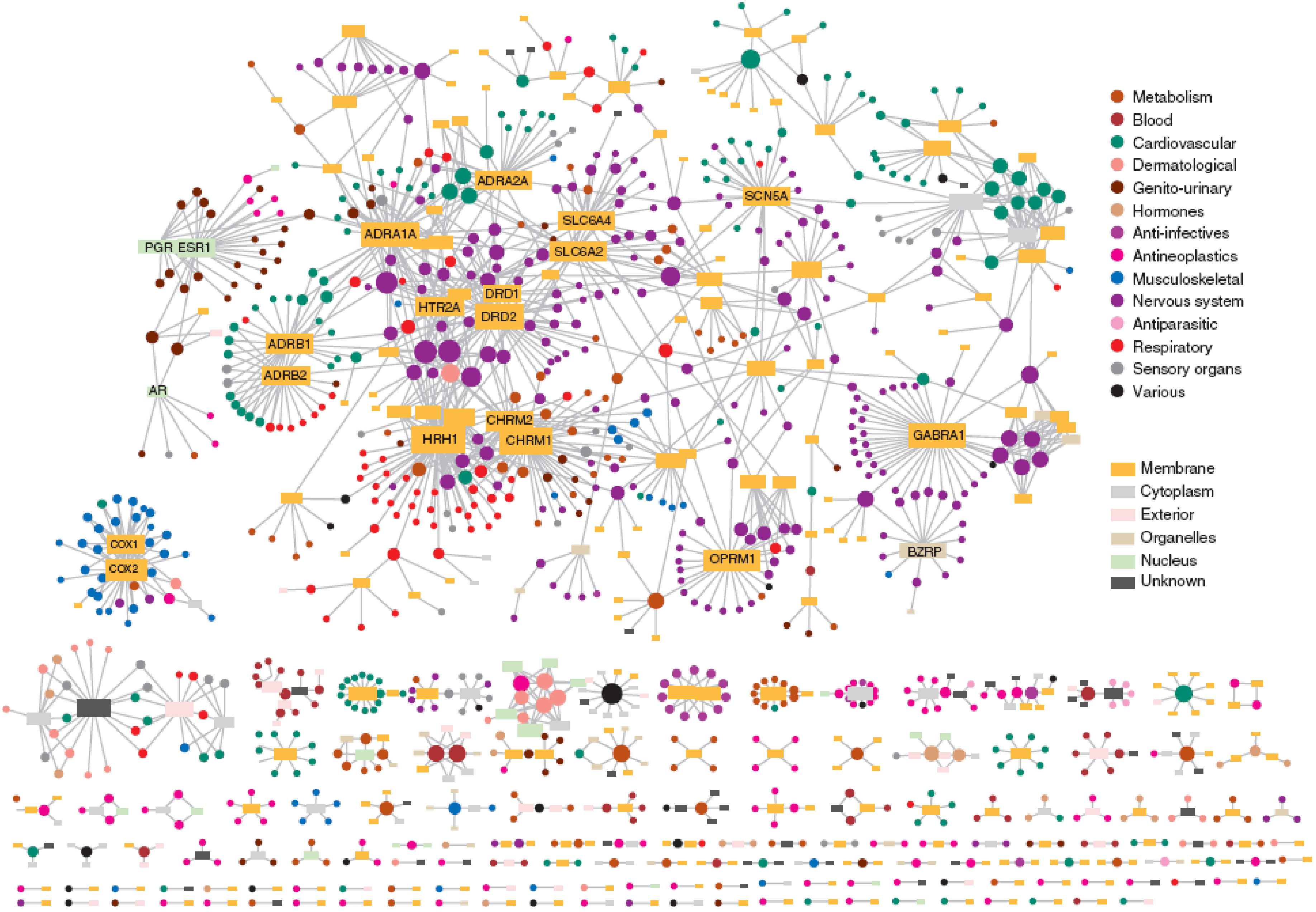
Drugs and drug targets

- □ Druggable genome
- □ DrugBank



Drugs and drug targets

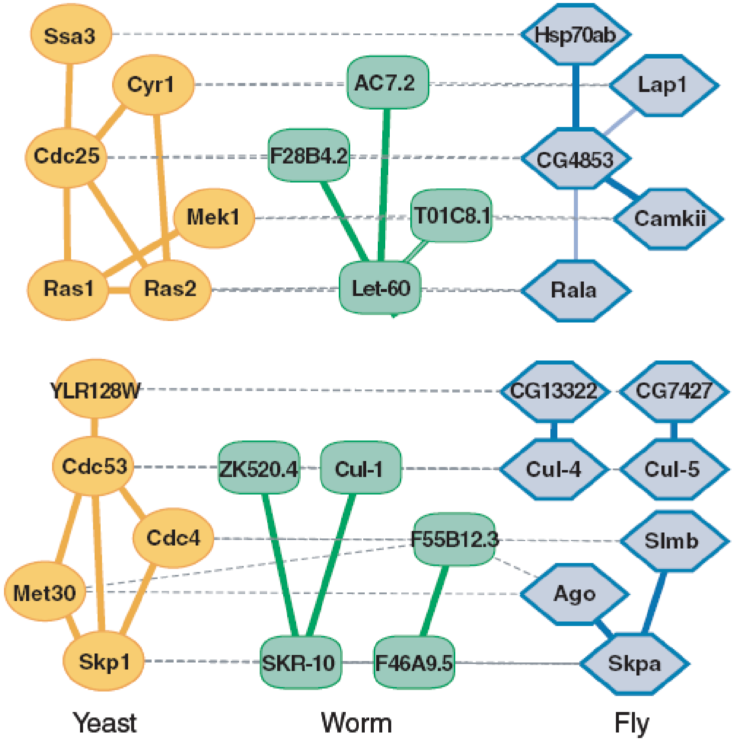
- The drug target network (Yildirim et al. 2007)
  - Two projections: “Drug network” & “Target-protein network”
  - The majority of drugs shared targets with other drugs
    - Industry trends: new drugs target already known targets
    - But, experimental drugs target more diverse set of proteins
  - Overlying target-protein network with human PPI network
    - 262 drug targets present in the human PPI network
    - These targets have higher degrees, but are not essential proteins
    - Do drug targets correspond to disease genes?
      - Most drugs target disease-genes indirectly
      - However, cancer drugs directly target the actual cause of disease



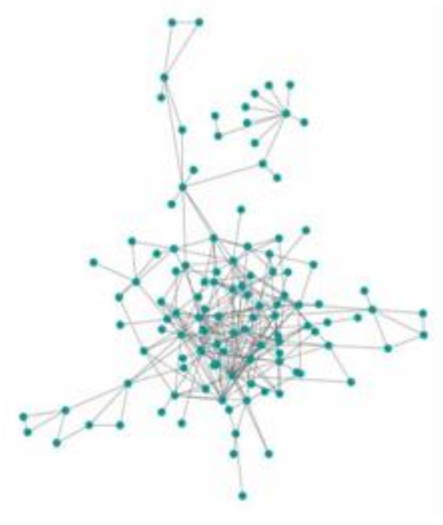
Circles = drugs, rectangles = target proteins, edge = the protein is a known target of the drug, size = degree.

Network alignment

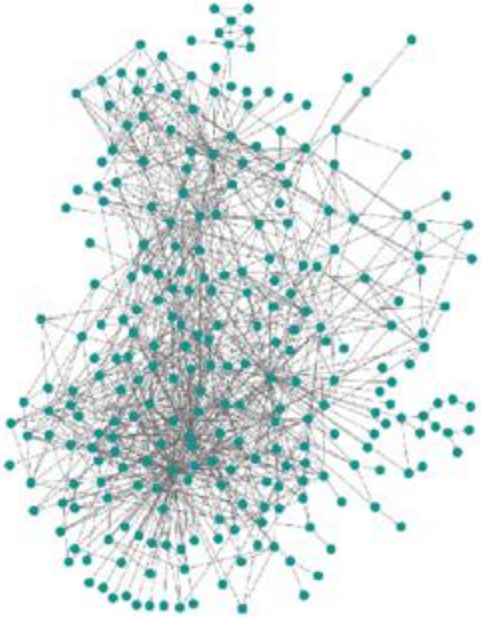
- Pathblast/Networkblast



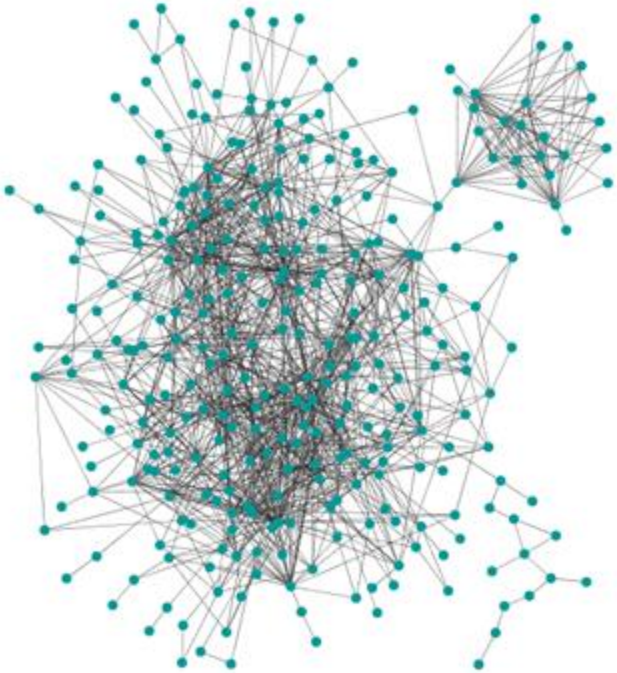
Network alignment



IsoRank: 116 proteins
261 interactions



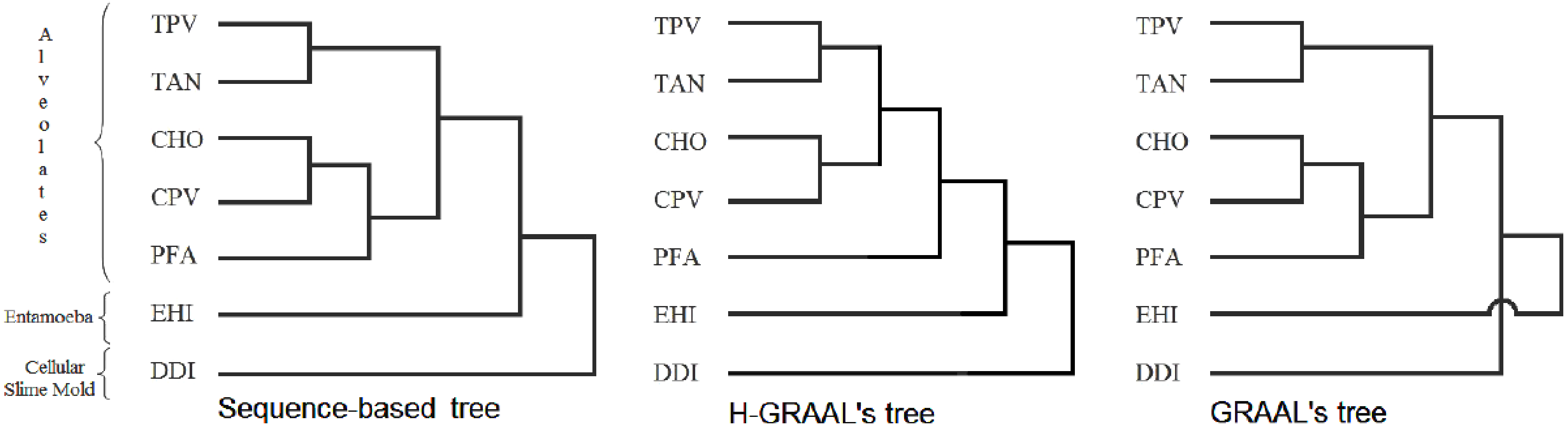
GRAAL: 267 proteins
900 interactions



H-GRAAL: 317 proteins
1,290 interactions

Network alignment

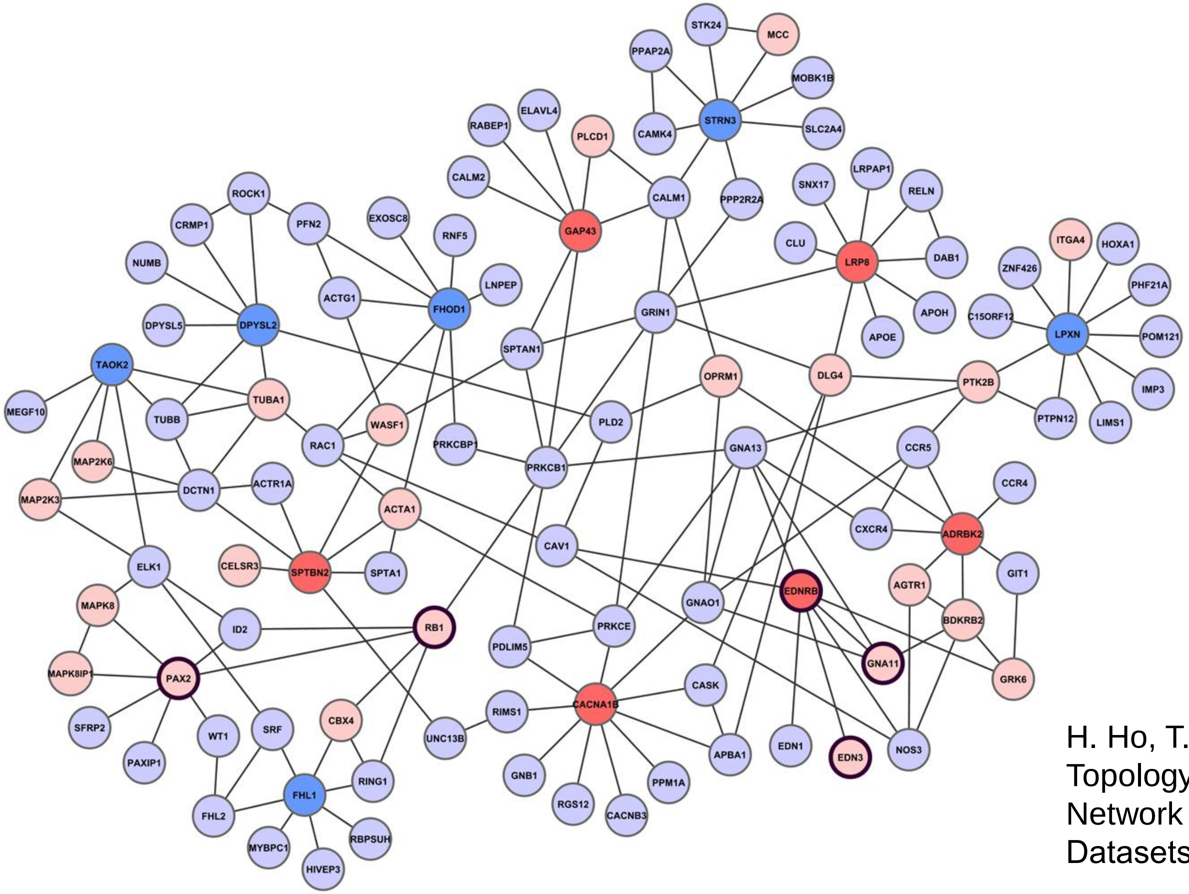
- Applications:
  - Protein function prediction
  - Prediction of protein interactions
  - Identification of the core interactome
  - Identification of evolutionary conserved subgraphs
  - Construction of phylogenetic trees



Uncovering components of pathways/networks underlying certain biological processes

Experiments→ computational predictions → experiments

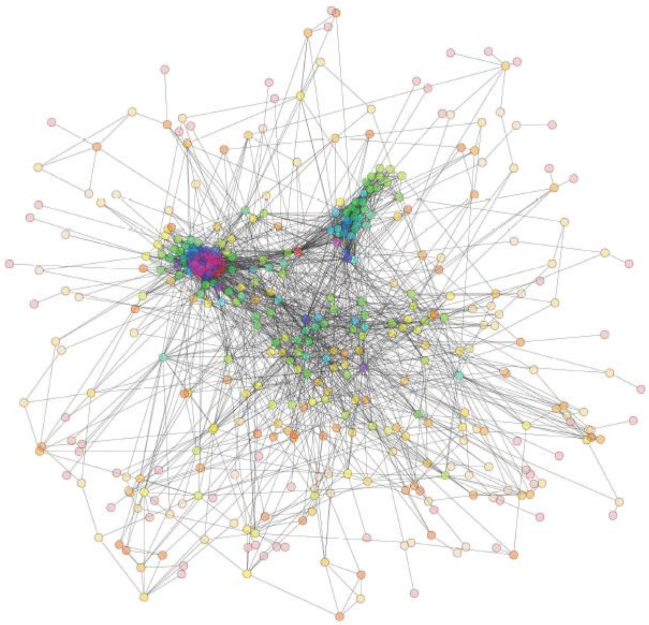
E.g., EDNRB-focused melanogenesis network

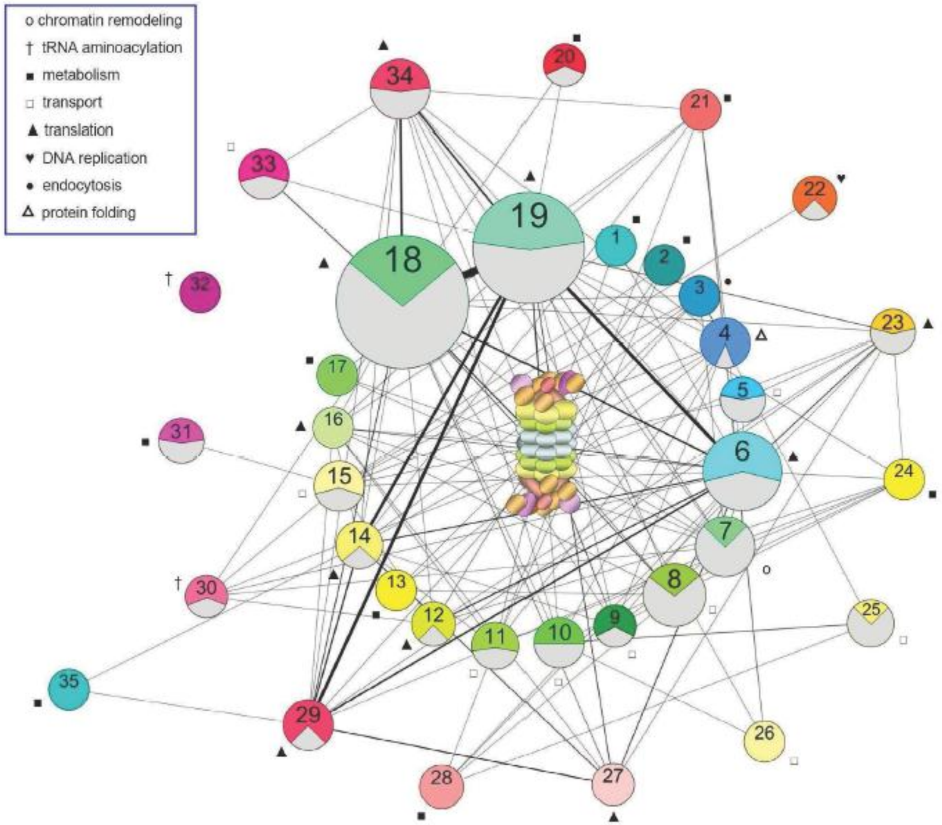


H. Ho, T. Milenković, et al. “Protein Interaction Network Topology Uncovers Melanogenesis Regulatory Network Components Within Functional Genomics Datasets,” BMC Systems Biology, 2010.

Uncovering components of pathways/networks underlying certain biological processes

- E.g., yeast proteasome network
- Reveal the interconnectivity of the proteasome complex with other protein complexes




o chromatin remodeling
† tRNA aminoacylation
■ metabolism
□ transport
▲ translation
♥ DNA replication
● endocytosis
Δ protein folding
34
20
21
33
22
19
18
1
2
3
32
4
23
17
5
16
31
6
15
24
14
7
30
13
8
12
11
10
9
25
35
29
26
28
27