# Web Structure Mining

## Rising Stars and Community Detection

Li Xiaoli

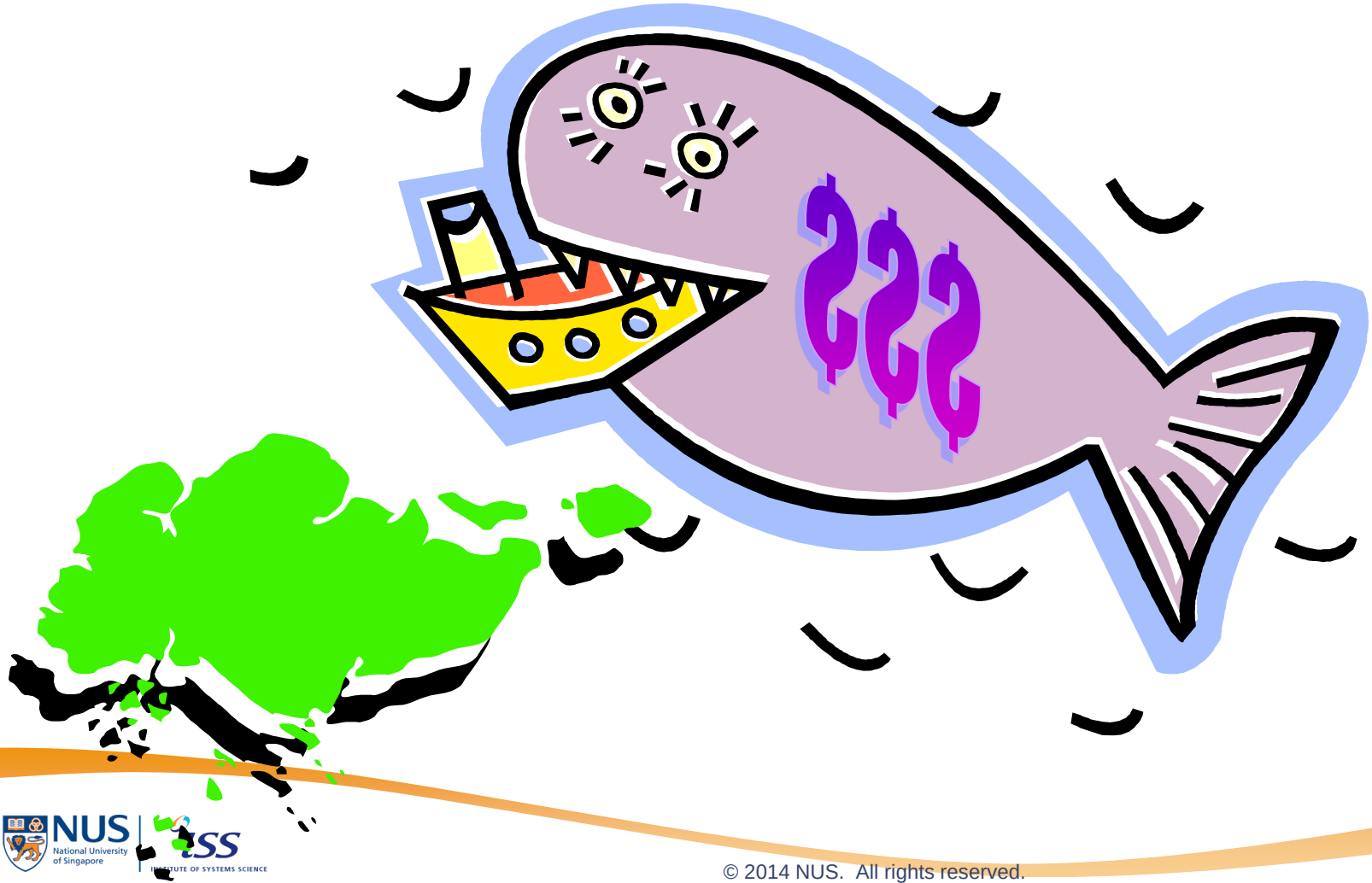
# Outlines

## PART I: Mining the Researcher Social Networks to Discover the Rising Stars

## PART II: Community Detection from Social Networks [focus]

Some organizations want to attract the "big whales" …

# But hiring the whales can be expensive…

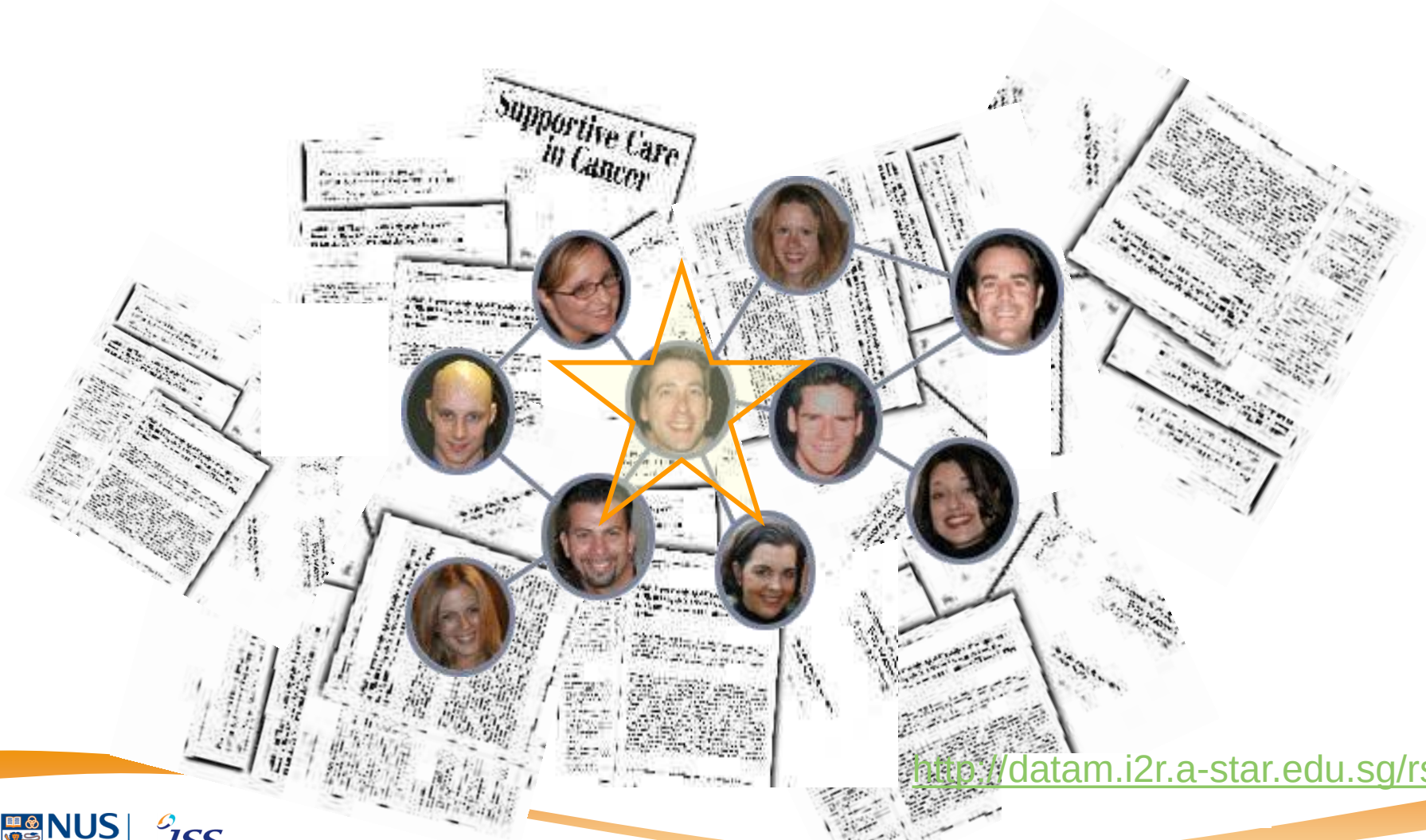


# Spot-the-Whales Problem

**Challenge**:

Is it possible to discover the “baby whales” (less $$$) to hire before they become really big and expensive?

# Searching for the Rising **Stars** in the Researcher Social Networks




http://datam.i2r.a-star.edu.sg/rstar/

# Raw Data

dblp.uni-trier.de

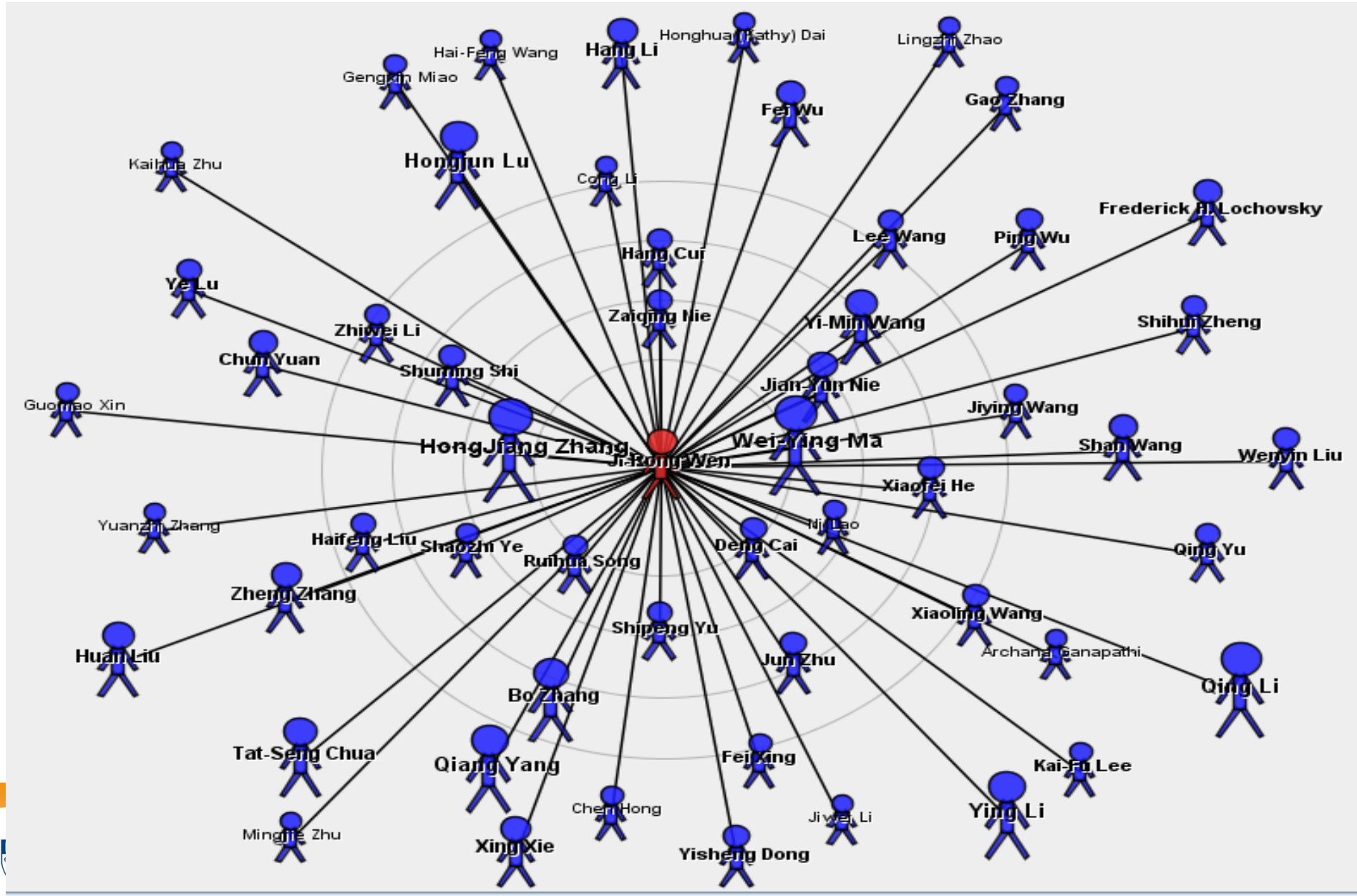
## See-Kiong Ng

List of publications from the DBLP Bibliography Server - FAQ

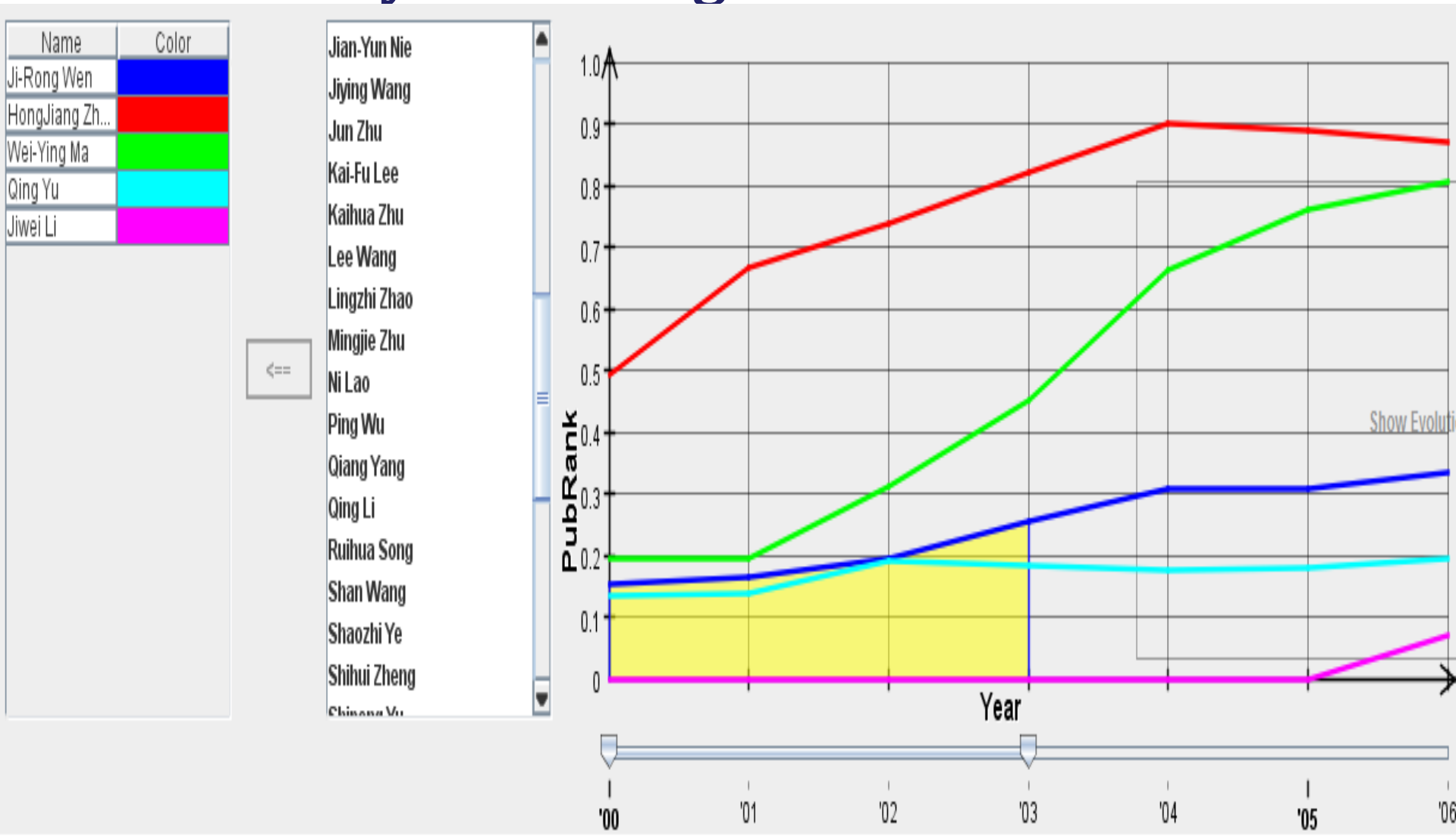
Coauthor Index - Ask others: ACM DL/Guide - CiteSeer - CSB - Google - MSN - Yahoo

| | | |
|---|---|---|
| | | **2007** |
| 32 | EE | Xiaoli Li, Bing Liu, See-Kiong Ng: Learning to Classify Documents with Only a Small Positive Training Set. ECML 2007: 201-213 |
| 31 | EE | Jin Chen, Wynne Hsu, Mong-Li Lee, See-Kiong Ng: Labeling network motifs in protein interactomes for protein function prediction. ICDE 2007: 546-555 |
| 30 | EE | Xiaoli Li, Bing Liu, See-Kiong Ng: Learning to Identify Unexpected Instances in the Test Set. IJCAI 2007: 2802-2807 |
| 29 | EE | Vincent Yan Fu Tan, See-Kiong Ng: Generic Probability Density Function Reconstruction for Randomization in Privacy-Preserving Data Mining. MLDM 2007: 76-90 |
| | | **2006** |
| 28 | EE | Xiaoli Li, Yin-Chet Tan, See-Kiong Ng: Systematic Gene Function Prediction Using a Fuzzy Nearest-Cluster Method on Gene Expression Data. IMSCCS (1) 2006: 171-178 |
| 27 | EE | Jin Chen, Wynne Hsu, Mong-Li Lee, See-Kiong Ng: NeMoFinder: dissecting genome-wide protein-protein interactions with meso-scale network motifs. KDD 2006: 106-115 |
| 26 | EE | Zhuo Zhang, Merlin Veronika, See-Kiong Ng, Vladimir B. Bajic: Intelligent Extraction Versus Advanced Query: Recognize Transcription Factors from Databases. PRIB 2006: 133-139 |
| 25 | EE | Jesper Jansson, See-Kiong Ng, Wing-Kin Sung, Hugo Willy: A Faster and More Space-Efficient Algorithm for Inferring Arc-Annotations of RNA Sequences through Alignment. Algorithmica 46(2): 223-245 (2006) |
| 24 | EE | Jin Chen, Wynne Hsu, Mong-Li Lee, See-Kiong Ng: Increasing confidence of protein interactomes using network topological metrics. Bioinformatics 22(16): 1998-2004 (2006) |
| 23 | EE | Jung-jae Kim, Zhuo Zhang, Jong C. Park, See-Kiong Ng: BioContrasts: extracting and exploiting protein-protein contrastive relations from biomedical literature. Bioinformatics 22(5): 597-605 (2006) |
| | | **2005** |
| 22 | EE | Wei-Khing For, Xiaoming Bao, Woon-Seng Gan, See-Kiong Ng: Reconfigurable Context-Sensitive Bio-Bridge Middleware for Smart Bio-Laboratories. AINA 2005: 561-566 |
| 21 | EE | Zhuo Zhang, Suisheng Tang, See-Kiong Ng: Toward discovering disease-specific gene networks from online literature. APBC 2005: 161-169 |
| 20 | EE | Xiaoming Bao, See-Kiong Ng, Eng-Huat Chua, Wei-Khing For: Virtual Lab Dashboard: Ubiquitous Monitoring and Control in a Smart Bio-laboratory. ICCSA (2) 2005: 1167-1176 |
| 19 | EE | Xiaoli Li, Soon-Heng Tan, See-Kiong Ng: Protein Interaction Prediction Using Inferred Domain Interactions and Biologically-Significant Negative Dataset. ICCSA (3) 2005: 318-326 |
| 18 | EE | See-Kiong Ng: Smart bio-laboratories of the future. ISCAS (5) 2005: 4779-4782 |
| 17 | EE | Jin Chen, Wynne Hsu, Mong-Li Lee, See-Kiong Ng: Discovering reliable protein interactions from high-throughput experimental data using network topology. Artificial Intelligence in Medicine 35(1-2): 37-47 (2005) |
| | | **2004** |
| 16 | EE | See-Kiong Ng, Zexuan Zhu, Yew-Soon Ong: Whole-Genome Functional Classification of Genes by Latent Semantic Analysis on Microarray Data. APBC 2004: 123-129 |
| 15 | EE | Soon-Heng Tan, Wing-Kin Sung, See-Kiong Ng: Discovering Novel Interacting Motif Pairs from Large Protein-Protein Interaction Datasets. BIBE 2004: 568-575 |
| 14 | EE | Jin Chen, Wynne Hsu, Mong-Li Lee, See-Kiong Ng: Systematic Assessment of High-Throughput Experimental Data for Reliable Protein Interactions Using Network Topology. ICTAI 2004: 368-372 |

# Step 1: From Data to Information: Construct the "Researcher Social Networks"

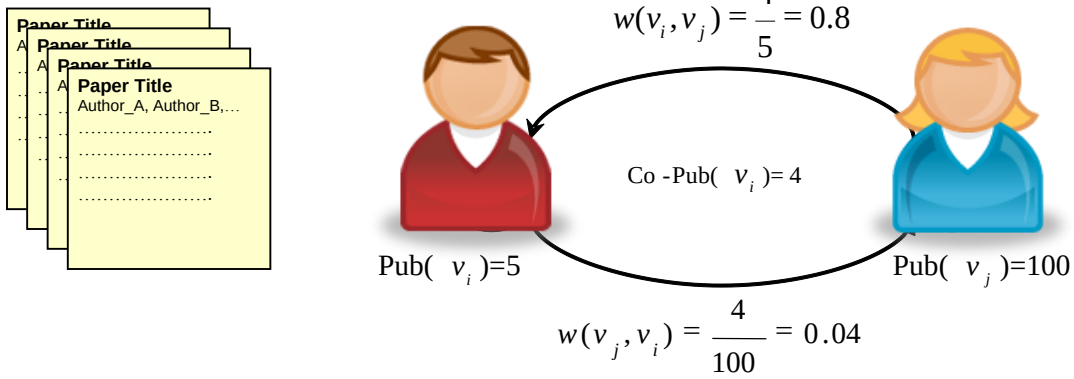


# Step 2: From Information to Knowledge: Identify the Rising Stars of Tomorrow



# How do we do it?

- Model the **mutual influence** of the social relationships as a directed weighted network (both nodes and links are weighted dynamically)




Paper:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.566.2777&rep=rep1&type=pdf

# Pubrank Algorithm

- Constructing weighted bi-directional network
  - ***Edge weighting***: model mutual influence between the two co-authors by weighting the edges in the network

$$w(v_i, v_j) = \frac{co\_pub(v_i, v_j)}{\sum_{k=1}^{|A_j|} co\_pub(v_j, v_k)}$$

  - ***Node weighting***: compute the impact factor of each researcher based on his/her track record

$$\lambda(v_i) = \frac{1}{|P|} * \sum_{i=1}^{|P|} \frac{1}{\alpha^{r(pub_i)-1}}$$

where $pub_i$ is the $i$-th publication, r($pub_i$) is the rank of publication $pub_i$ , and α (0 < α < 1) is a damping factor designed so that lower ranked publications have lower scores. The larger $\lambda(v_i)$ is, the higher the average quality of papers published by researcher $v_i$

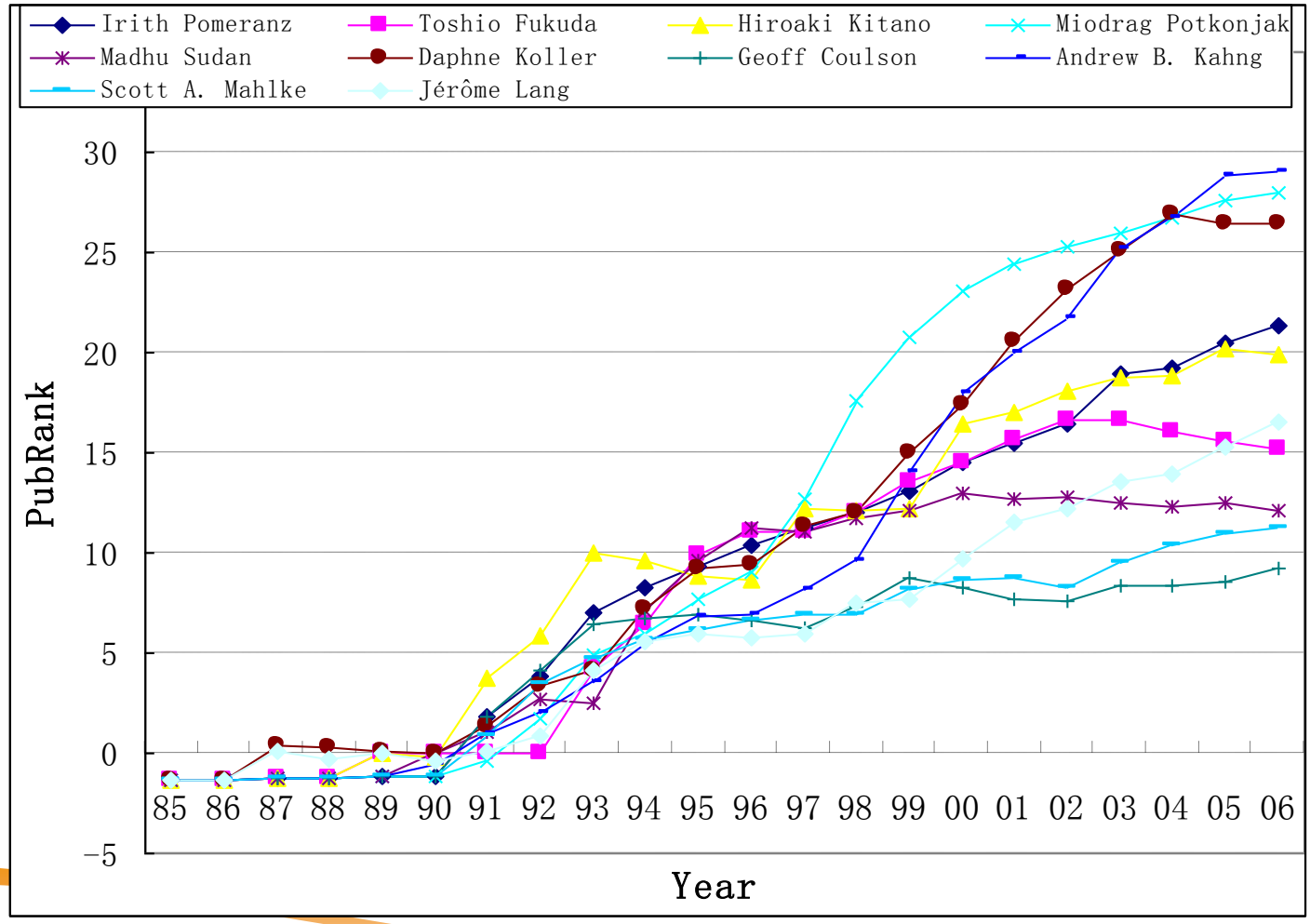
# Pubrank Algorithm

- Modeling the ***propagation*** of mutual influence cross the whole network

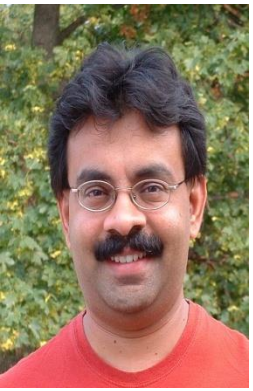
$$p(p_i) = \frac{1-d}{N} + d * \sum_{p_j \in M(p_i)} \frac{w(j,i) * \lambda(p_j) * P(p_j)}{\sum_{k=1}^{|A_i|} w(k,i) * \lambda(p_j)}$$

- We use a regression model to detect rising stars as those whose increase in their potential scores is significantly higher than the average researcher across the years.

# Results (The Increasing Trend for the Top Ten Rising Stars from 1990 to 1995)



# Top 10 Predicted Rising Stars and Their Respective Rank in 1990 and 2006



Madhu Sudan, MIT

Daphne Koller, Stanford

Geoff Coulson Lancaster

Andrew B. Kahng, UCSD

Scott A. Mahlke Michigan university

Irith Pomeranz Purdue

| Name | Gradient | Author Ranking | |
|---|---|---|---|
| | | Start (1990) | Final (2006) |
| Irith Pomeranz | 2.13 | 64753 | 85 |
| Toshio Fukuda | 2.09 | 17992 | 244 |
| Hiroaki Kitano | 1.91 | 38330 | 106 |
| Miodrag Potkonjak | 1.89 | 64753 | 24 |
| Madhu Sudan | 1.89 | 34896 | 466 |
| Daphne Koller | 1.85 | 34901 | 32 |
| Geoff Coulson | 1.65 | 64753 | 915 |
| Andrew B. Kahng | 1.49 | 50451 | 19 |
| Scott A. Mahlke | 1.49 | 64753 | 568 |
| Jérôme Lang | 1.47 | 43646 | 191 |
| **Average** | **1.79** | **47922.8** | **265** |

# Top Ten Predicted Rising Stars and Their Possible Nurturers and/or Strong Collaborators

| Name | Rank (1990) | Possible nurturers and their rank (in 1990) |
|---|---|---|
| Irith Pomeranz | 64753 | Sudhakar M. Reddy(67) |
| Hiroaki Kitano | 38330 | Hideto Tomabechi(5620) |
| Miodrag Potkonjak | 64753 | Alice C. Parker(32) |
| Madhu Sudan | 34896 | Richard J. Lipton(5), Daniel H. Greene(683) |
| Daphne Koller | 34901 | Danny Dolev(7), Amotz Bar-Noy(323) Rudiger Reischuk(681) |
| Geoff Coulson | 64753 | Gordon S. Blair(964) |
| Andrew B. Kahng | 50451 | C. K. Wong(543), Majid Sarrafzadeh(1613) |
| Scott A. Mahlke | 64753 | Wen-mei W. Hwu(696) |
| Jérôme Lang | 43646 | Henri Prade(665), Didier Dubois(1156) |

# Top Rising Stars from Database Domains(1990~1994)

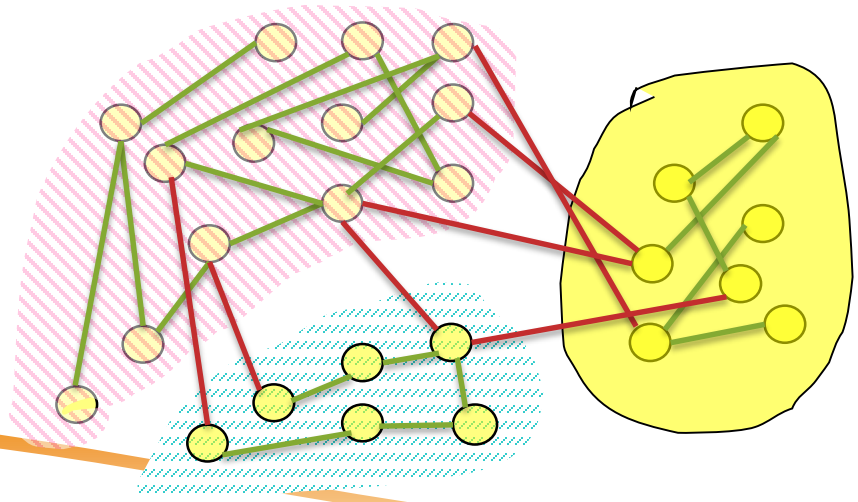
| Name | Position | Organization | Awards | Top Citation |
|---|---|---|---|---|
| Bharat K. Bhargava | Professor | Purdue University | IEEE Technical Achievement Award, IETE Fellow | 143 |
| H. V. Jagadish | Professor | University of Michigan, Ann Arbor | ACM Fellow | 457 |
| Hamid Pirahesh | Manager | IBM Almaden Research Center | IBM Fellow, IBM Master Inventor | 1428 |
| Ming-Syan Chen | Professor | National Taiwan University | ACM Fellow, IEEE Fellow | 1260 |
| Philip S. Yu | Professor | UIC | ACM Fellow, IEEE Fellow | 1260 |
| Rajeev Rastogi | Director | Bell Labs Research Center, Bangalore | Bell Labs Fellow | 1178 |
| Rakesh Agrawal | Head | Microsoft Search Labs | ACM Fellow, IEEE Fellow, a Member of the National Academy of Engineering | 6285 |
| Richard R. Muntz | Professor | UCLA | ACM Fellow, IEEE Fellow | 1191 |
| Shi-Kuo Chang | Professor | University of Pittsburgh | IEEE fellow | 171 |
| Jiawei Han | Professor | UIUC | ACM fellow | 6158 |

# 2. Community Detection

- Unravel the underlying community substructures for social networks, biological networks, citation networks, communication networks  etc
- Communities can reveal important functional information about these networks. For example, communities in the biological networks correspond to functional modules or biological pathways that are useful for understanding the causes of various diseases

# What is the community detection?

Qualitatively, detecting communities from networks involves *dividing the vertices into groups* such that there is a higher density of links within groups than between them.



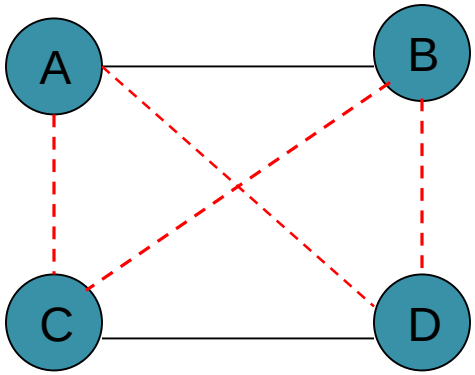
# 2.1 Clique detection (PNAS paper)

- Strategy: enumeration of complete subgraphs since it is difficult to directly detect big cliques.
  - starts from the smallest statistically significant size (4 cliques)
  - goes up in size one by one until all cliques are found (5, 6, …. cliques).

Spirin and Mirny [***PNAS*, 2003**]

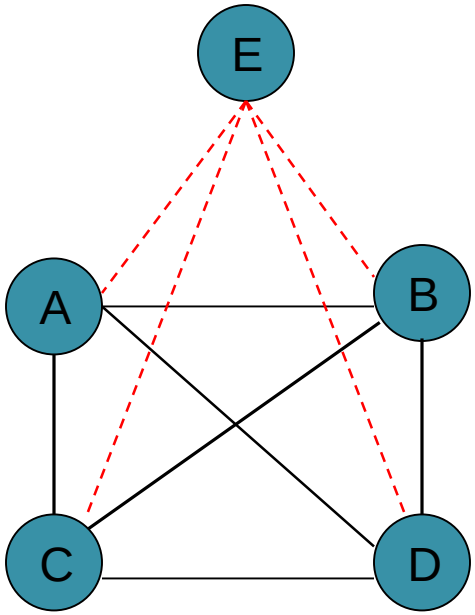
# 2.1 Clique detection (PNAS paper)

- Step1: To find cliques with size 4, all pairs of edges are picked successively.
  - For every pair $A$,$B$ and $C$,$D$ we check whether there are edges between $(A,C)$, $(A,D)$, $(B,C)$, and $(B,D)$. If all of these edges are present, $ABCD$ is a clique.




# Detect larger cliques

- Step 2: for every clique $ABCD$, we pick all known nodes/proteins successively. For every picked node/protein $E$, if the interactions $(E,A)$, $(E,B)$, $(E,C)$, and $(E,D)$ are known, then $ABCDE$ is a clique with size 5.





•This process is continued for cliques of size 6, 7, and larger until the biggest clique is found.

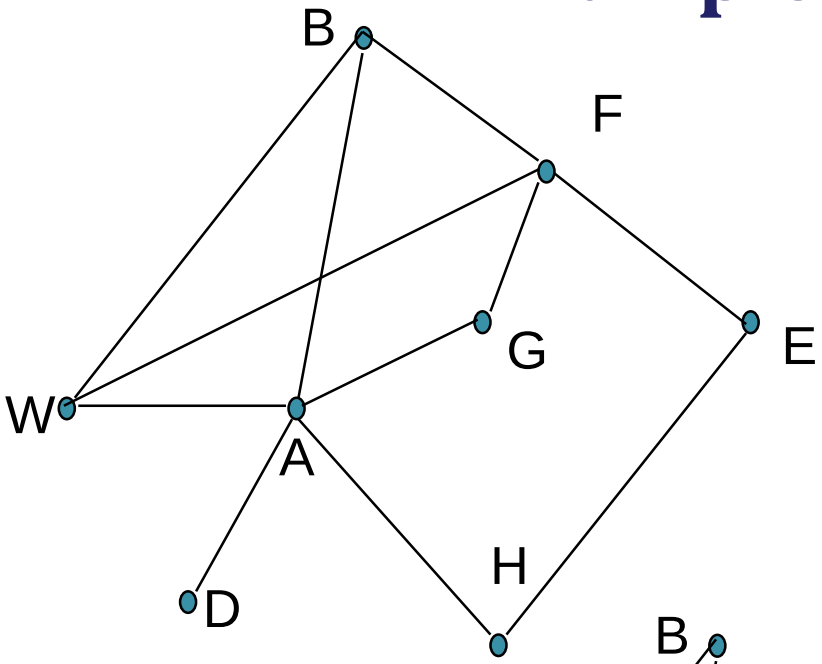
# Cliques Method Summary

- **Advantage**
  - Implementation is simple
- **Disadvantage**
  - Detecting cliques is a computational intensive problem (NP hard).
  - May not be used to handle large-scale networks

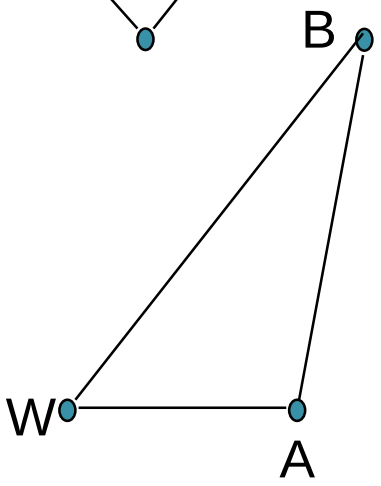
# 2.2 k-cores (Science paper)

- Let G = (V,E) be a graph, where V is the set of vertices and E is the set of edges.
- A subgraph H = (C, E|C) induced by the set C is a *k*-core iff
  - $\forall v \in C$: $\deg_H(v) \geq k$ and
  - H is a *maximum* subgraph with this property.
- E|C is the set of edges which connect the vertices in set C
- k-core is also called a core of order k

# Example of a subgraph



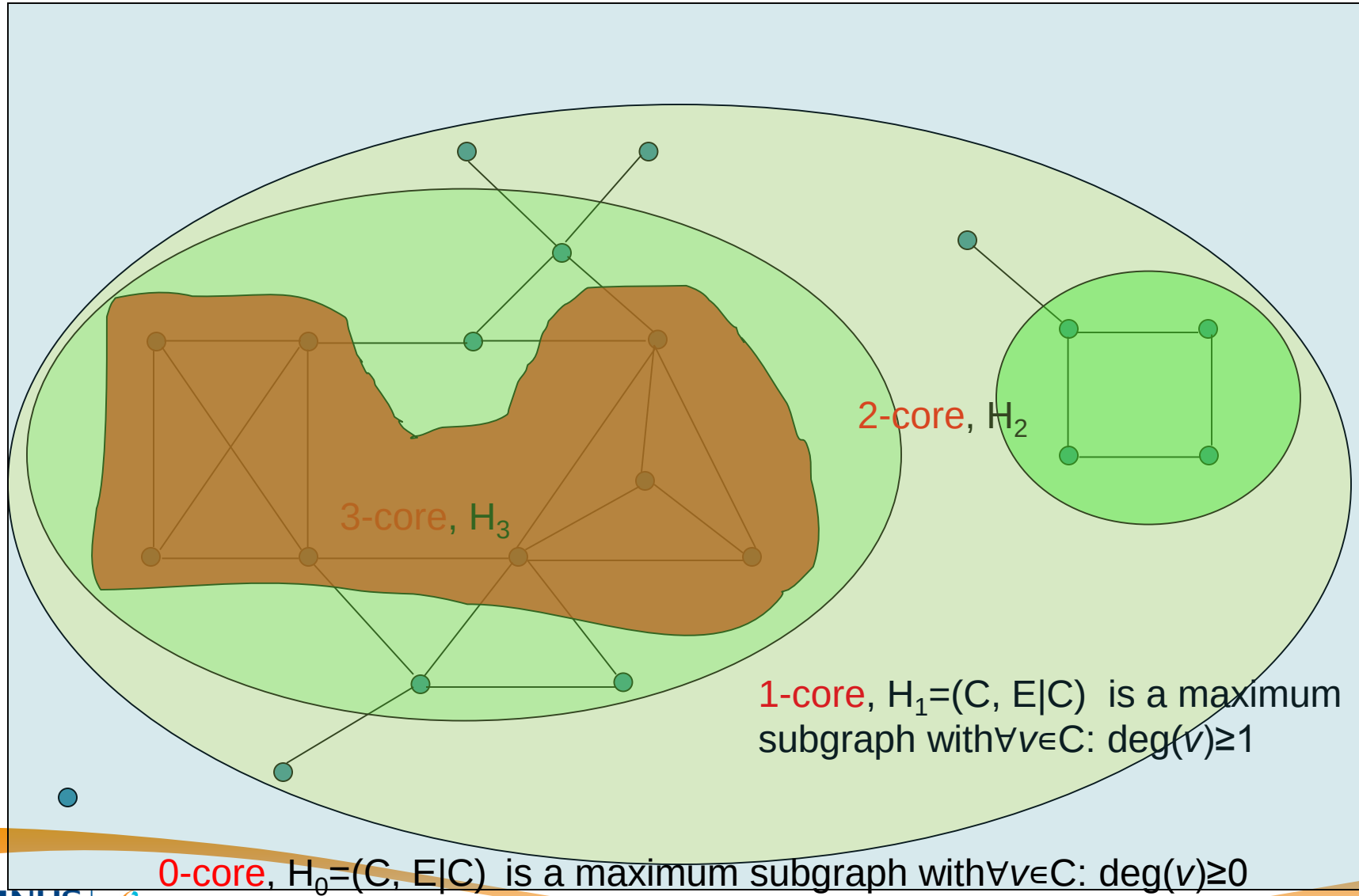

- G=(V,E)
- V={A,B,W,D,E,F,G,H}
- E={(A,B),(A,W),(A,D),(A,G), (A,H), (B,W), (B,F), (C,F), (E,F),(E,H),(F,G)}




- A subgraph H = (C, E|C) induced by the set C
- C={A,B,W}
- E|C={(A,B),(A,W),(B,W)}

E|C is the set of edges which connect the vertices in set C

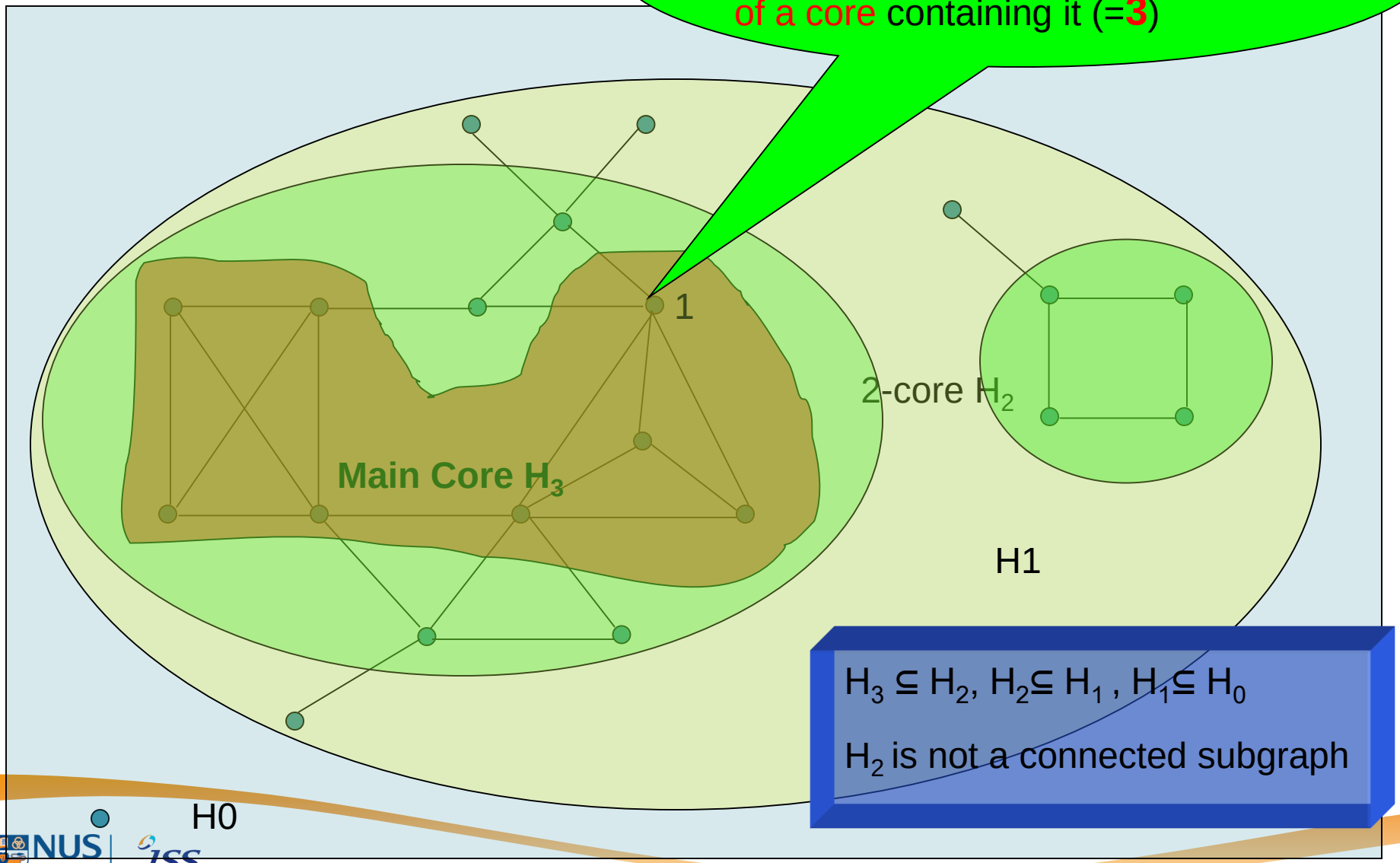
# An Example of k-core



# Some concepts of k-core

- main core: the core with maximum order, e.g. $H_3$
- The core number of vertex v is the **highest order of a core** that contains this vertex. The vertices within $H_3$, their core number is 3

# An E



# Properties of the cores

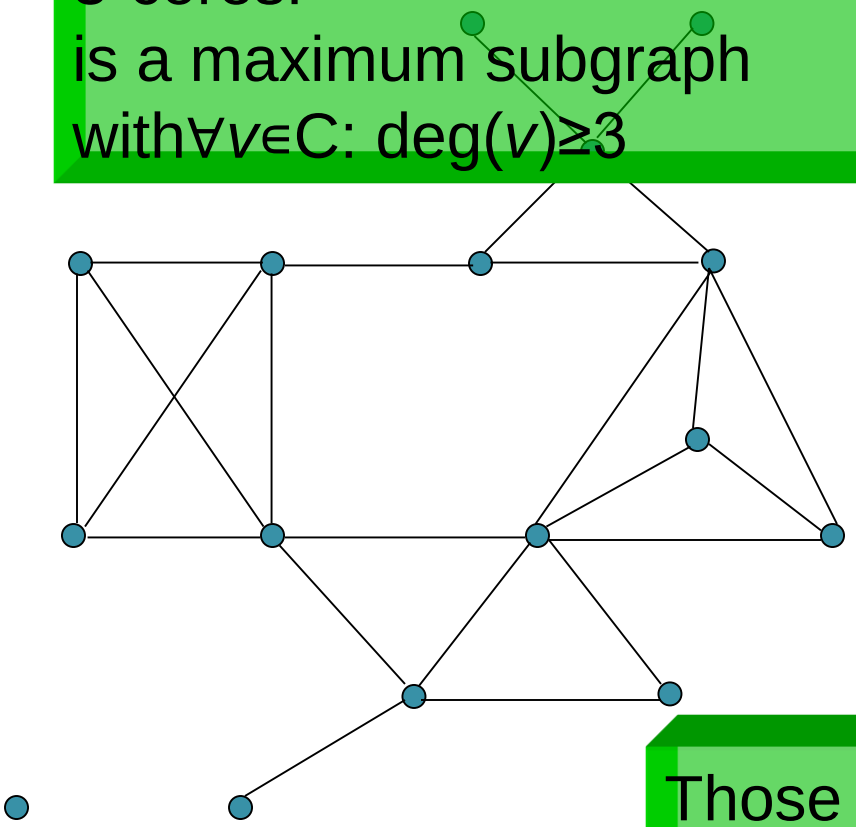
- The cores are nested: $i < j \Rightarrow H_j \subseteq H_i$, e.g $H_3 \subseteq H_2$ ($i$=2 < $j$=3)
- Cores are not necessarily connected subgraphs, e.g. $H_2$
- There exists very efficient algorithm to determine core numbers (or corresponding k-cores).

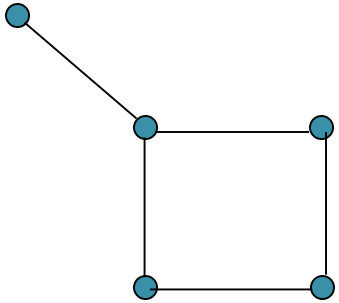
# The intuition to detect k-cores

- From a given graph G = (V,E), we recursively delete all vertices whose degree less than k (MUST not belong to k-core) and the edges incident with the remaining graph is the k-core.
- It can be implemented in O(|E|), with the help of bin-sort.

# An Example to detect 3-cores

3-cores:
is a maximum subgraph
with$\forall v \in C$: deg($v$)≥3



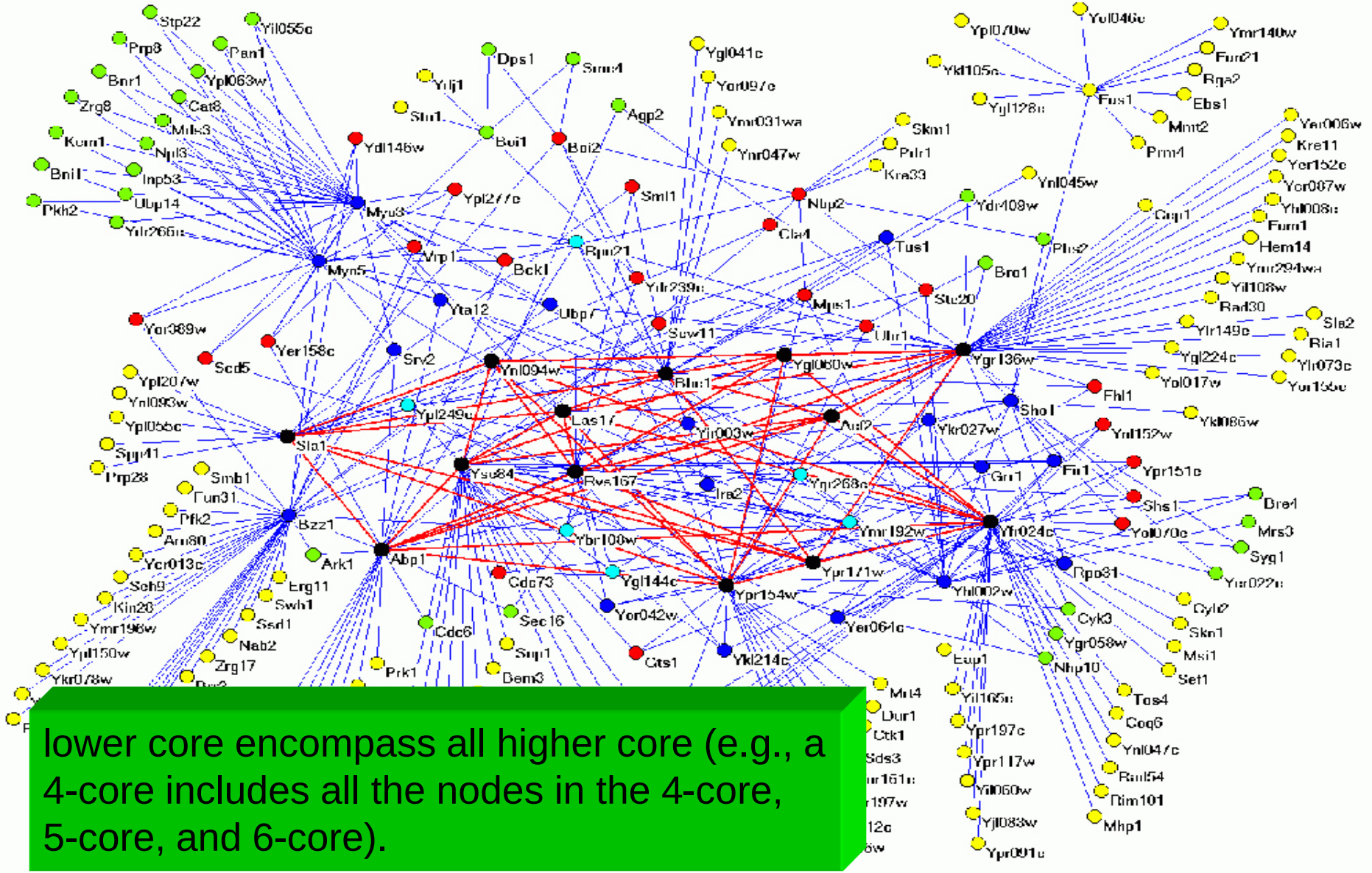


Those nodes with degree >=3 in the original graph may not belong to 3-core

# Apply k-core algorithms to PPI

- Amy Hin Yan Tong, *et al.* A Combined Experimental and Computational Strategy to Define Protein Interaction Networks for Peptide Recognition **Modules**, ***Science* 295,** 321 (2002).
- Yeast protein-protein interaction network predicted by means of phage display–selected peptides.
- In total, 394 interactions and 206 proteins

# Results




lower core encompass all higher core (e.g., a 4-core includes all the nodes in the 4-core, 5-core, and 6-core).

The proteins are colored according to their k-core value

**6-core, 5-core, 4-core, 3-core, 2-core, 1-core**

# Summarize the K-cores

- **Advantage**
  - The most efficient algorithm (linear time) to find the dense graph
- **Disadvantage**
  - Note that vertices in a core may not be highly connected with each other (even non-connected)
  - It can not be used to detect many dense subgraphs (nested graphs)

# References


- Vladimir Batagelj, Matjaz Zaversnik, **An O(m) algorithm** for cores decomposition of networks, Preprint series, Vol. 40 (2002), 798.

http://vlado.fmf.uni-lj.si/pub/preprint/imfm0798.pdf

# Sometimes you just have normal data Can you detect community?

- You may have normal data points
- Data

Features: $f_1$ $f_2$ $f_d$

Objects/ $D_1=(V_{11}, V_{12}, \ldots, V_{1d})$

$D_2=(V_{21}, V_{22}, \ldots, V_{2d})$

Instances/ ……

points $D_n=(V_{n1}, V_{n2}, \ldots, V_{nd})$

Proximity Matrix

| | $D_1$ | $D_2$ | … | $D_n$ |
|---|---|---|---|---|
| $D_1$ | 1 | 0.2 | | 0.8 |
| $D_2$ | 0.2 | 1 | | 0.7 |
| … | | | | |
| $D_n$ | 0.8 | 0.7 | | 1 |

# 2.3 Shared Near Neighbor (SNN) Based Clustering

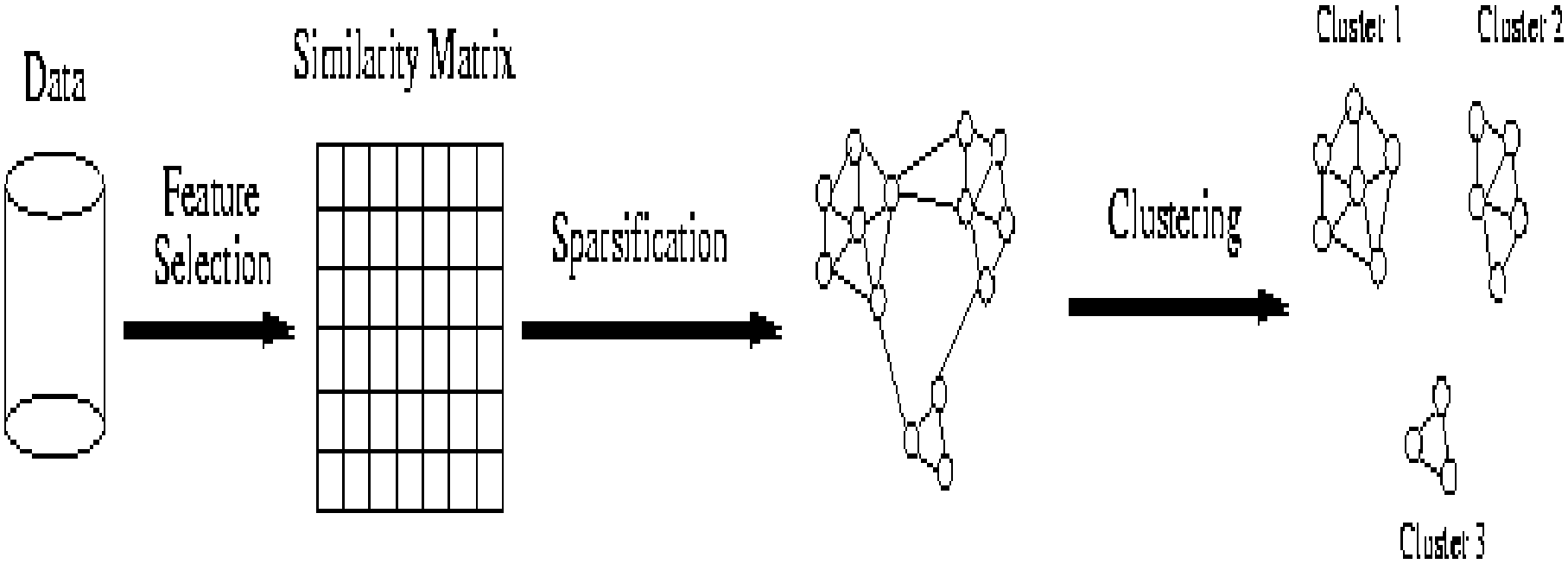
- Graph-Based clustering uses the **proximity graph**
  - Start with the proximity matrix
  - Consider each point/object as a node in a graph
  - Each edge between two nodes has a weight which is the proximity between the two points

We could have too many edges!

# Sparsification

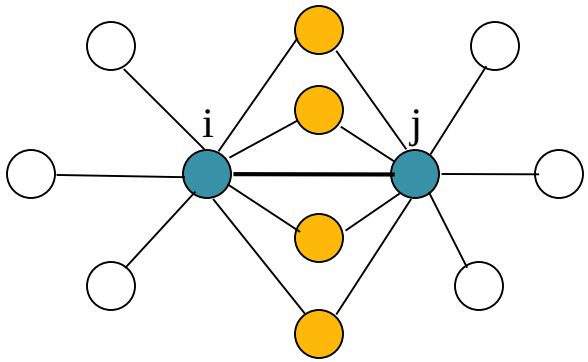
- Sparsification
  - ***k*-NN graph**: choose part of the links - each node just keeps top $k$ neighbors with largest similarities
  - **Threshold based**: use specified similarity threshold $t$ to choose the links with similarities larger than $t$
- Benefit: The amount of data that needs to be processed is drastically reduced
  - Sparsification can eliminate more than 99% of the entries in a proximity matrix
  - The amount of time required to cluster the data is drastically reduced
  - The size of the problems that can be handled is thus increased

# Sparsification in the Clustering Process



# Shared Near Neighbor (SNN) Approach

SNN graph: the **weight** of an edge is the number of shared neighbors between vertices given that the vertices are connected




Sim (i, j)=4

Social networks: We not only know each other; we also have some common friends

Two points to compute the similarity between node i and node j
1. i and j must occur in **each other's nearest neighbor lists**, representing they are similar according to the existing similarity measure
2. How many common neighbors (other than i, j) do they share? The more common neighbors, the more similar they are

# Example: SNN Similarity

- Given the following proximity matrix between 6 data points, calculate the Shared Near Neighbor similarity matrix with K=2 and K=3.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | 1.000 | 0.809 | 0.874 | 0.697 | 0.681 | 0.696 |
| B | 0.809 | 1.000 | 0.746 | 0.673 | 0.590 | 0.594 |
| C | 0.874 | 0.746 | 1.000 | 0.603 | 0.680 | 0.771 |
| D | 0.697 | 0.673 | 0.603 | 1.000 | 0.514 | 0.543 |
| E | 0.681 | 0.590 | 0.680 | 0.514 | 1.000 | 0.622 |
| F | 0.696 | 0.594 | 0.771 | 0.543 | 0.622 | 1.000 |

K=2

| | 1st NN | 2nd NN |
|---|---|---|
| A | C | B |
| B | A | C |
| C | A | F |
| D | A | B |
| E | A | C |
| F | C | A |

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 0 | 0 | 0 |
| B | | 0 | 0 | 0 | 0 | 0 |
| C | | | 0 | 0 | 0 | 1 |
| D | | | | 0 | 0 | 0 |
| E | | | | | 0 | 0 |
| F | | | | | | 0 |

Why SNN(A,D)=0?

# Example: SNN similarity (cont')

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | 1.000 | 0.809 | 0.874 | 0.697 | 0.681 | 0.696 |
| B | 0.809 | 1.000 | 0.746 | 0.673 | 0.590 | 0.594 |
| C | 0.874 | 0.746 | 1.000 | 0.603 | 0.680 | 0.771 |
| D | 0.697 | 0.673 | 0.603 | 1.000 | 0.514 | 0.543 |
| E | 0.681 | 0.590 | 0.680 | 0.514 | 1.000 | 0.622 |
| F | 0.696 | 0.594 | 0.771 | 0.543 | 0.622 | 1.000 |

- K=3

| | 1st NN | 2nd NN | 3rd NN |
|---|---|---|---|
| **A** | C | B | D |
| **B** | A | C | D |
| **C** | A | F | B |
| **D** | A | B | C |
| **E** | A | C | F |
| **F** | C | A | E |

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | 0 | 2 | 1 | 2 | 0 | 0 |
| B | | 0 | 1 | 2 | 0 | 0 |
| C | | | 0 | 0 | 0 | 1 |
| D | | | | 0 | 0 | 0 |
| E | | | | | 0 | 2 |
| F | | | | | | 0 |

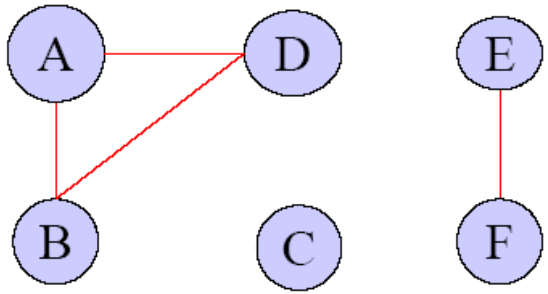
# Jarvis-Patrick Clustering

- First, the k-nearest neighbors of all points are found
  - In graph terms, this can be regarded as breaking all but the K strongest links from a point to other points in the proximity graph
- **A pair of points is put in the same cluster if**
  - any two points share ***at least*** **T** neighbors and
  - the two points are in each others K nearest neighbor list
- For instance, we might choose a nearest neighbor list of size 20 (K=20) and put points in the same cluster if they share more than 10 near neighbors (T=10)

# Example: Jarvis-Patrick Clustering

- Jarvis-Patrick Clustering for K=3 and T=2

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | 0 | 2 | 1 | 2 | 0 | 0 |
| B | | 0 | 1 | 2 | 0 | 0 |
| C | | | 0 | 0 | 0 | 1 |
| D | | | | 0 | 0 | 0 |
| E | | | | | 0 | 2 |
| F | | | | | | 0 |




- Thus, three clusters are obtained: {A, B, D}, {C}, {E, F}

T is a key parameter which decides if any two points should belong to a same cluster

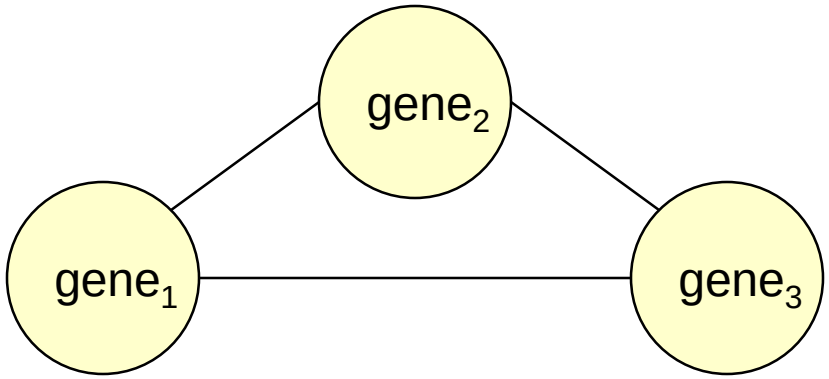
# 2.4 HCS Algorithm

- Once we have a graph (e.g. shared nearest neighbor graph, or sparsification proximate graph), we can also apply other graph partitioning techniques.

- **HCS: Highly Connected Subgraphs Algorithm**
  - Uses graph theoretic techniques
  - Applied HCS to biological data (gene expression data) and other data

# HCS: Similarity Graph

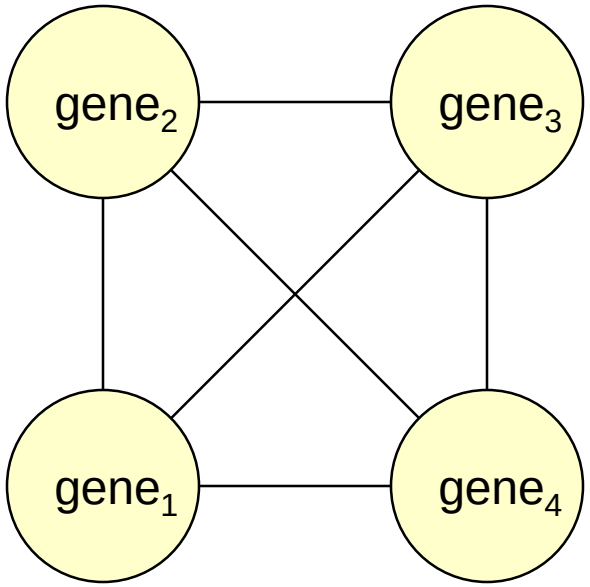
- Similarity Graph
  - Nodes correspond to elements
  - Edges connect similar elements (those whose similarity value is above a given threshold)




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

# HCS: Edge Connectivity

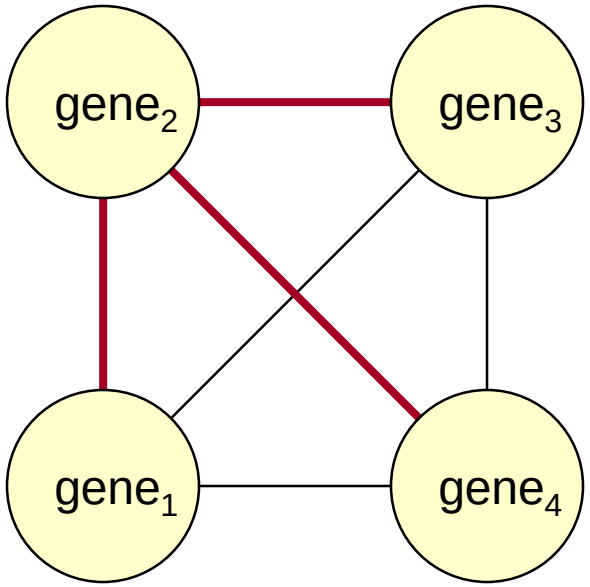
- Edge Connectivity: minimum number of edges whose removal results in a disconnected graph




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

# HCS: Edge Connectivity

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



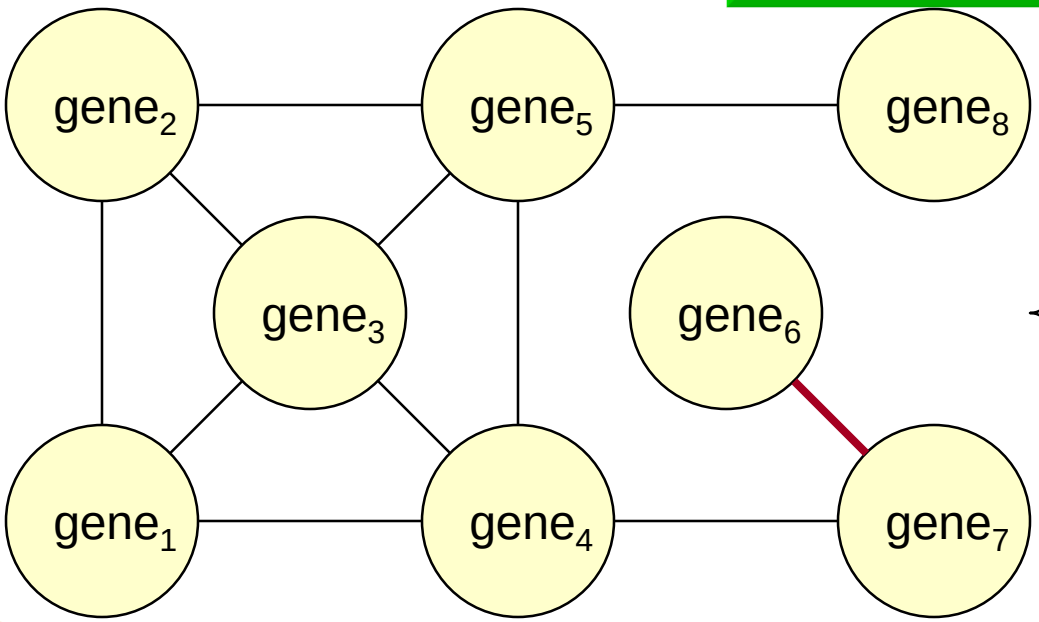

Hard to disconnect, wrt #nodes

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

# HCS: Highly Connected Subgraphs

- HCS: Subgraphs whose edge connectivity **exceeds** half the number of nodes (i.e. k(G)>|V|/2)

HCS is a graph that is not easily disconnected !!!

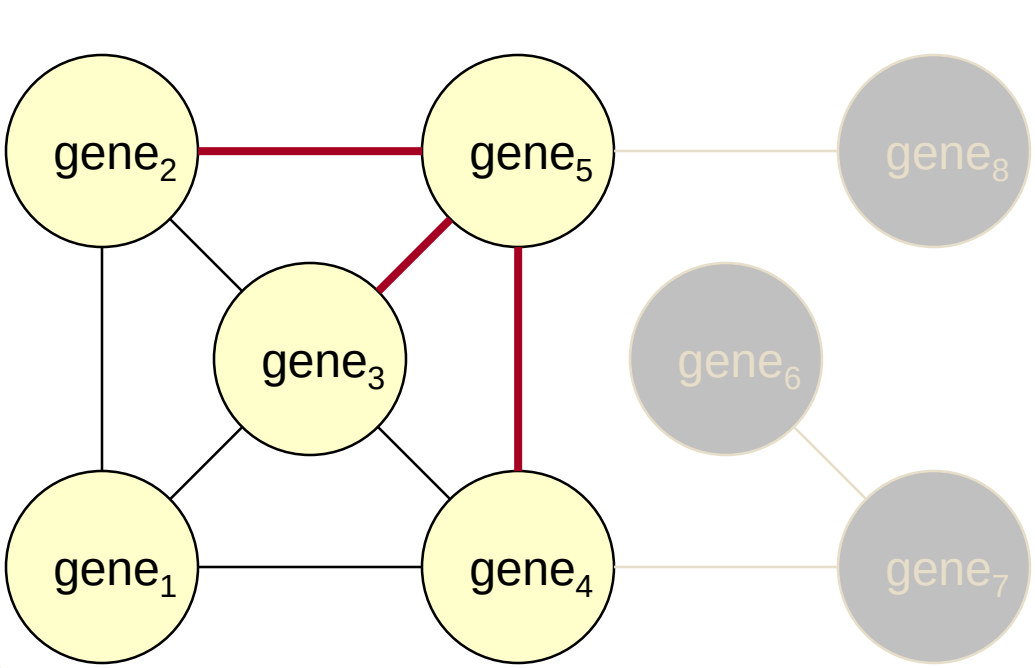



**Entire Graph**
Nodes =|V|= 8
Edge connectivity = k(G)=1

Not HCS! since 1<8/2
Easily disconnected

# HCS: Highly Connected Subgraphs

- Subgraphs whose edge connectivity **exceeds** half the number of nodes

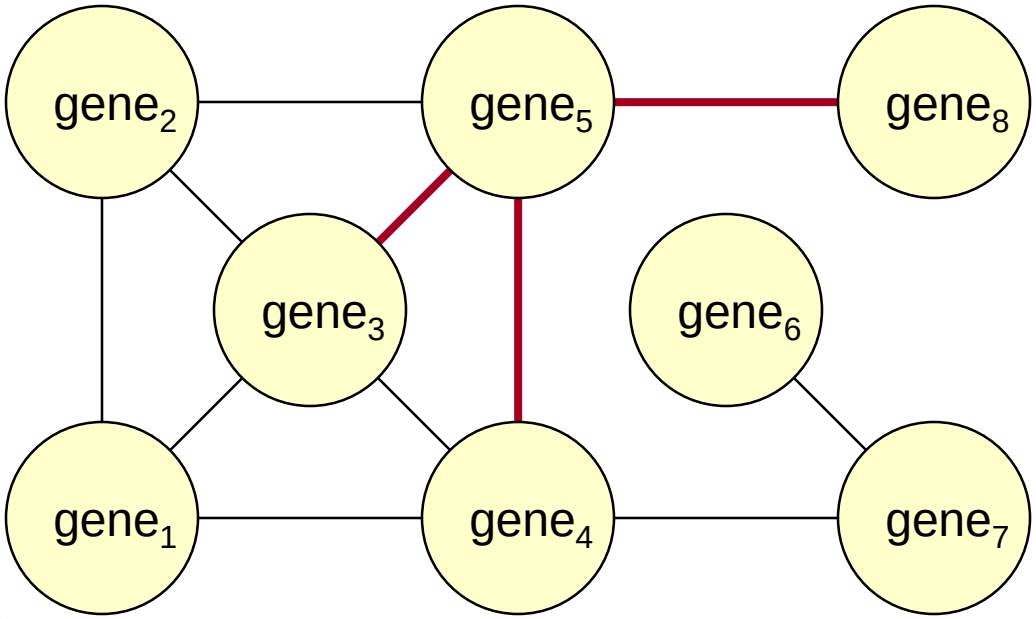



HCS! Since 3>5/2

**Sub Graph**
Nodes = 5
Edge connectivity = 3

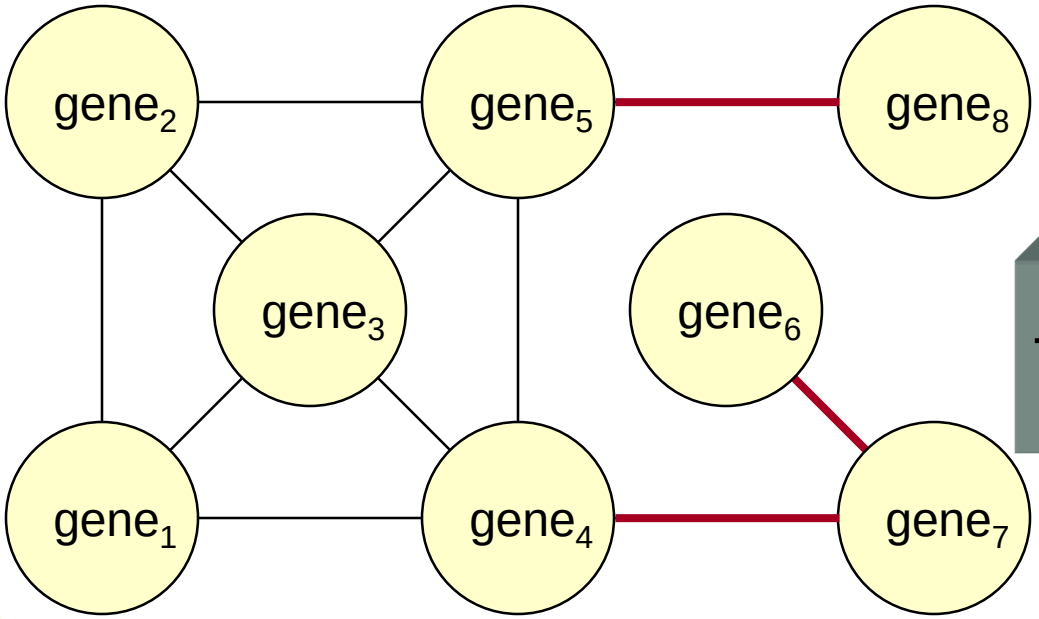
# HCS: Cut

- Cut: a set of edges whose removal disconnects the graph



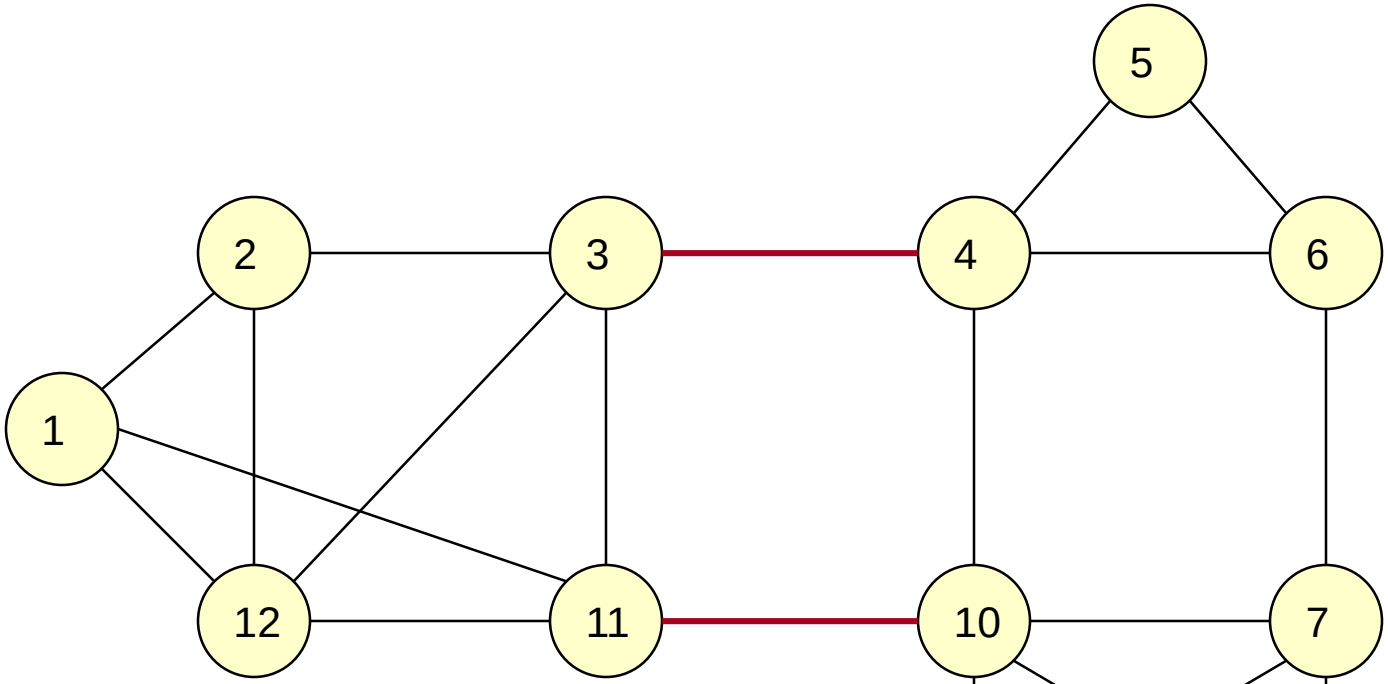
# HCS: Minimum Cut

- Minimum cut: a cut with a *minimum* number of edges



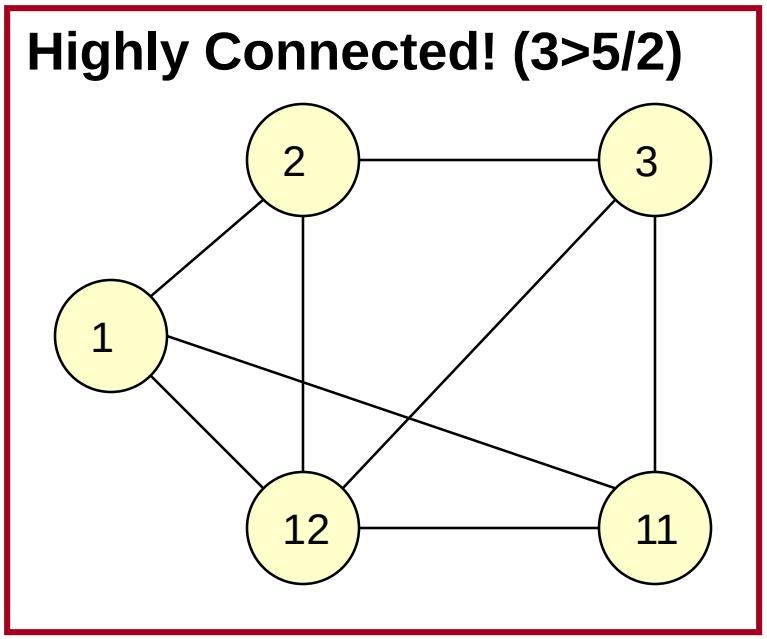

Three minimum cuts !!!

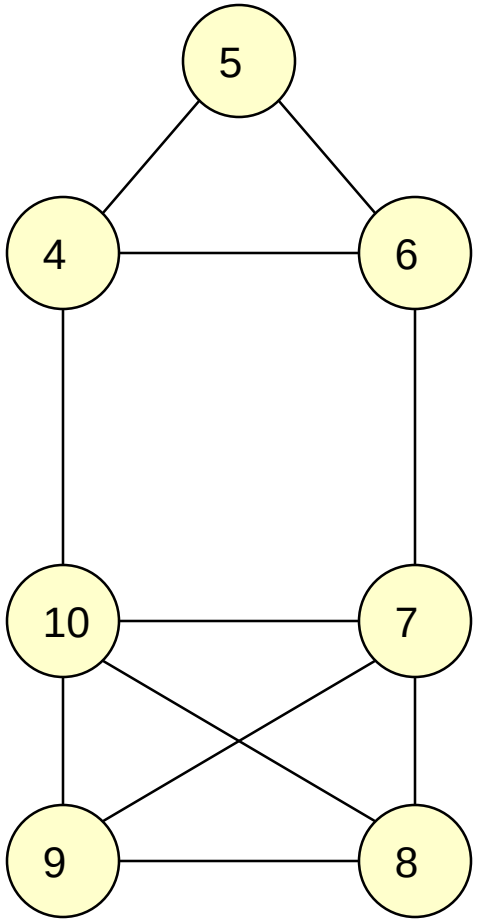
# HCS: Algorithm (by example)




find and remove a minimum cut

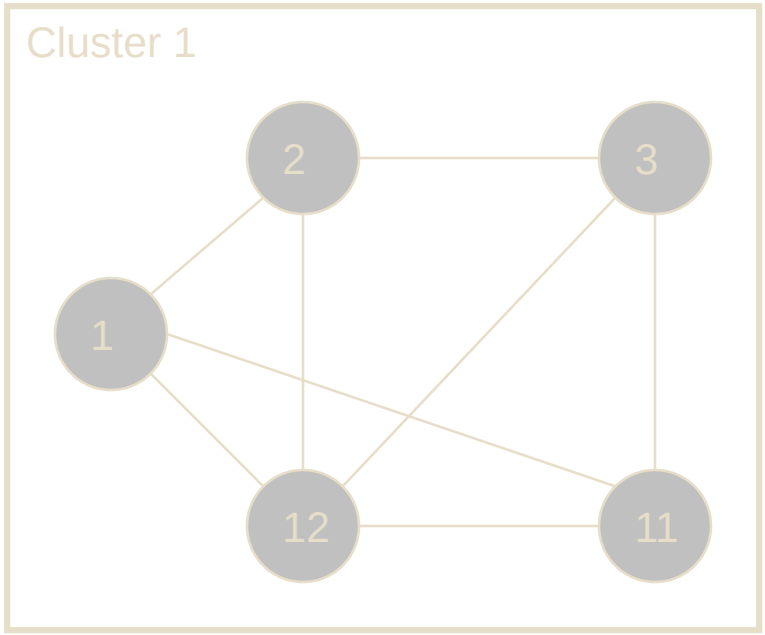
# HCS: Algorithm (by example)



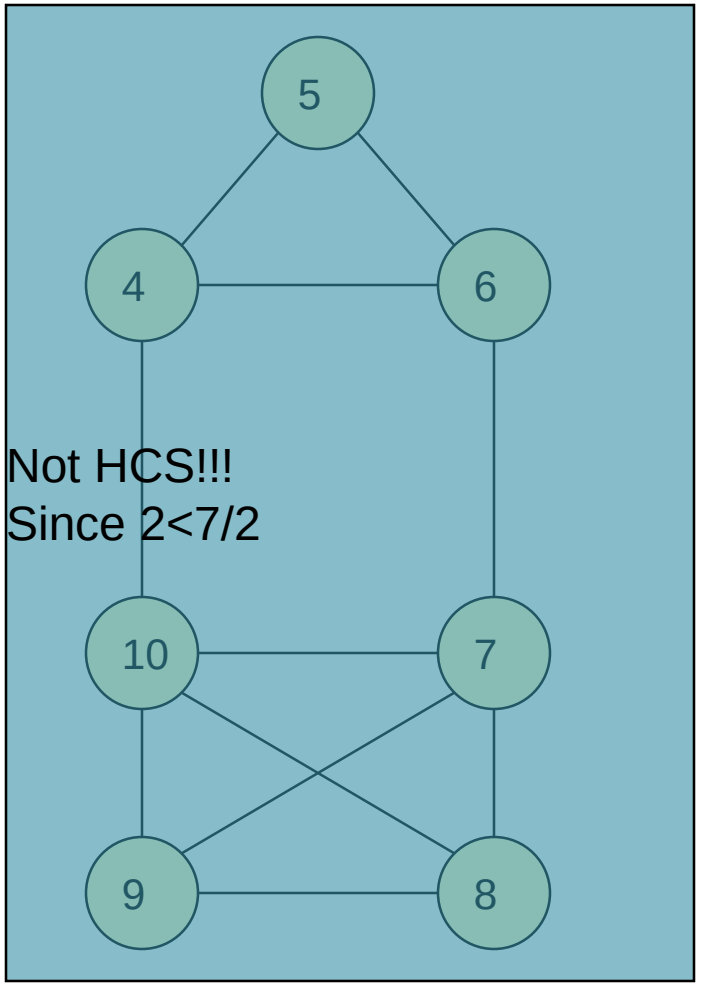




are the resulting subgraphs highly connected?

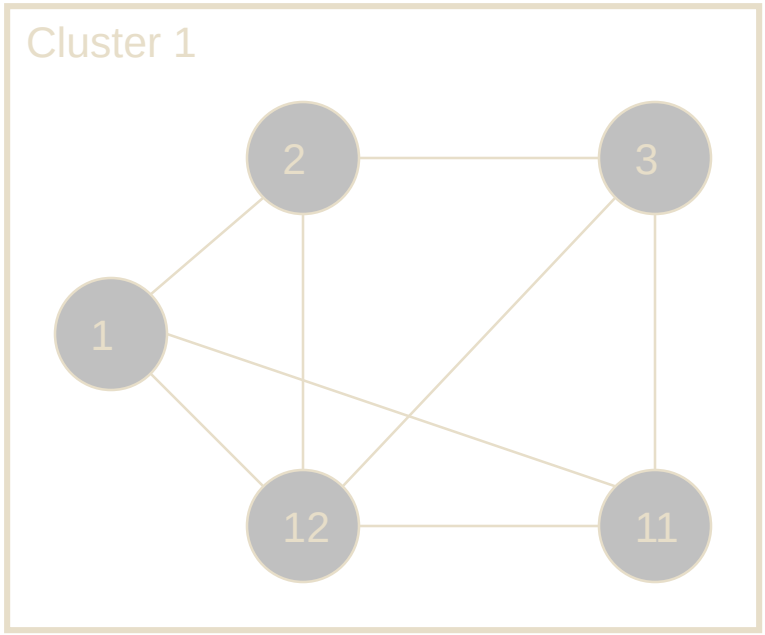
# HCS: Algorithm (by example)



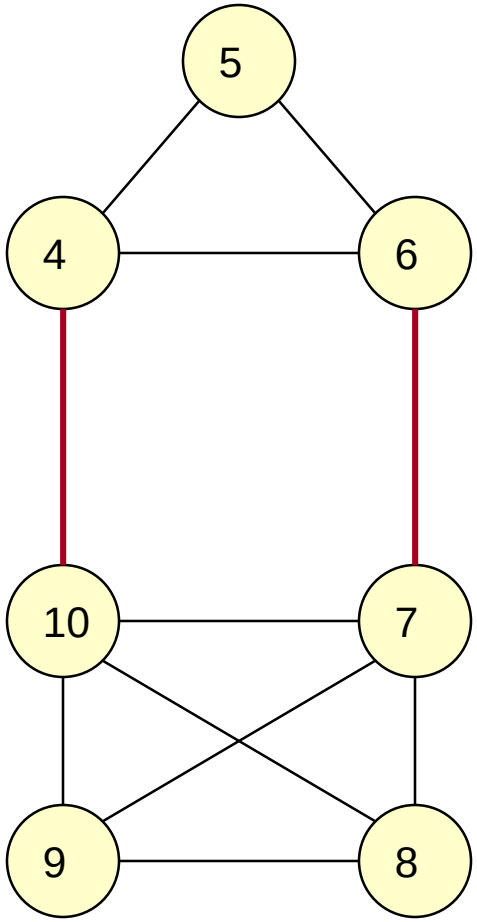

repeat process on non-highly connected subgraphs



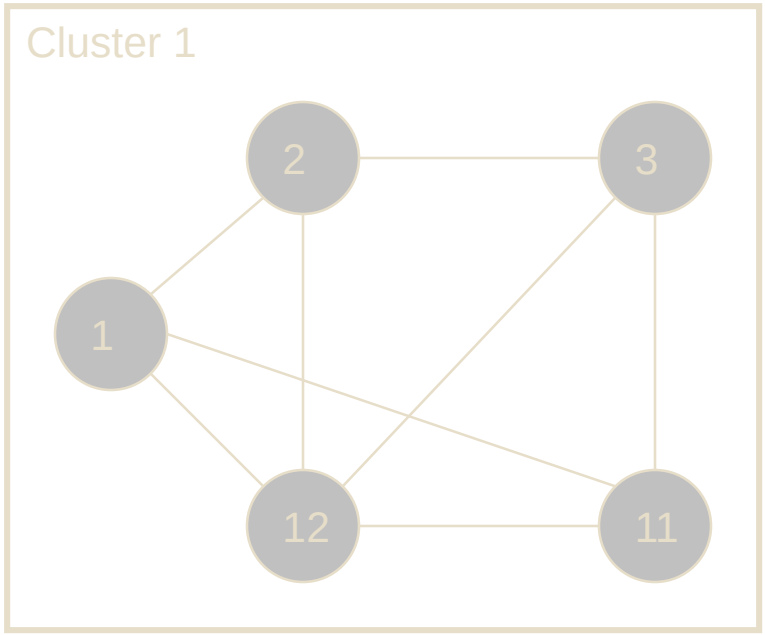
# HCS: Algorithm (by example)

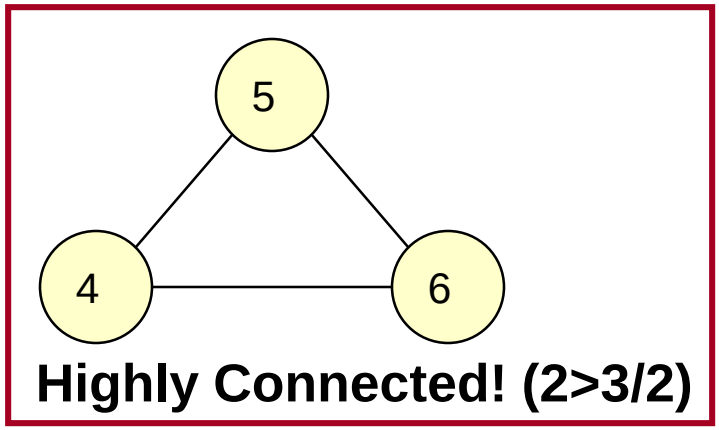


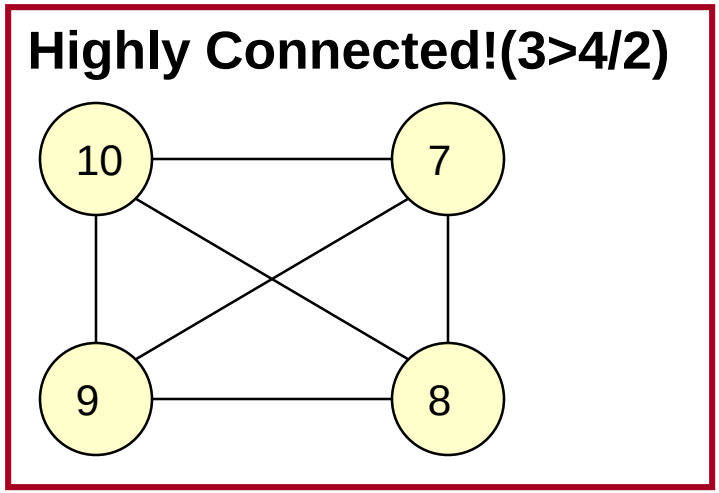

find and remove a minimum cut



# HCS: Algorithm (by example)

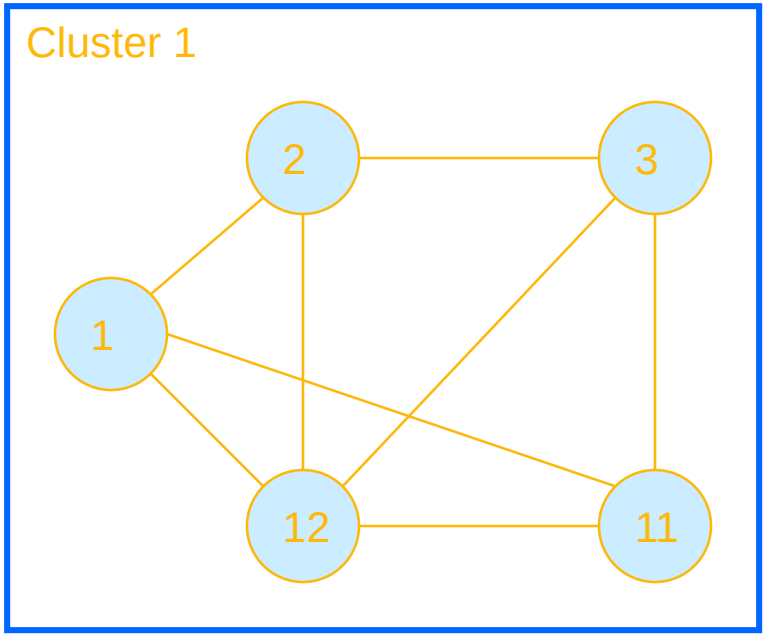


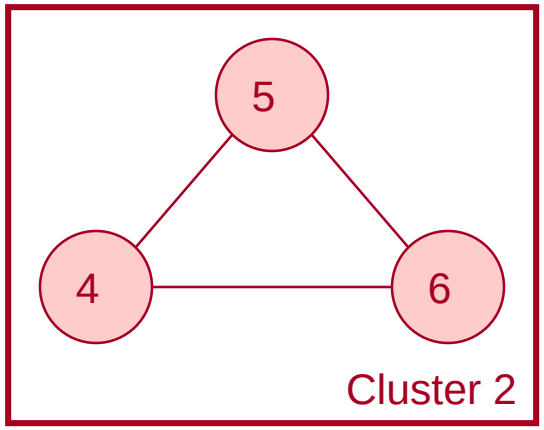



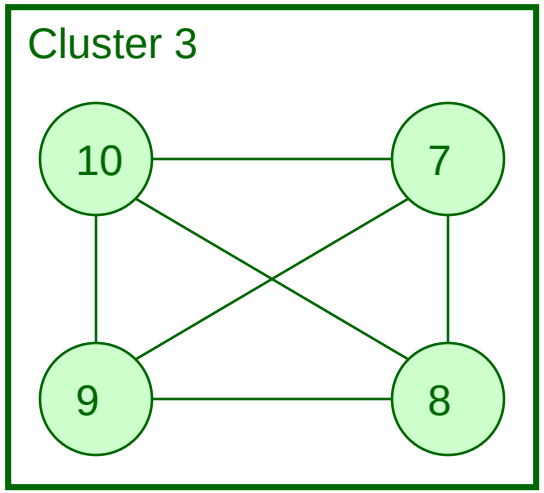




are the resulting subgraphs highly connected?

# HCS: Algorithm (by example)

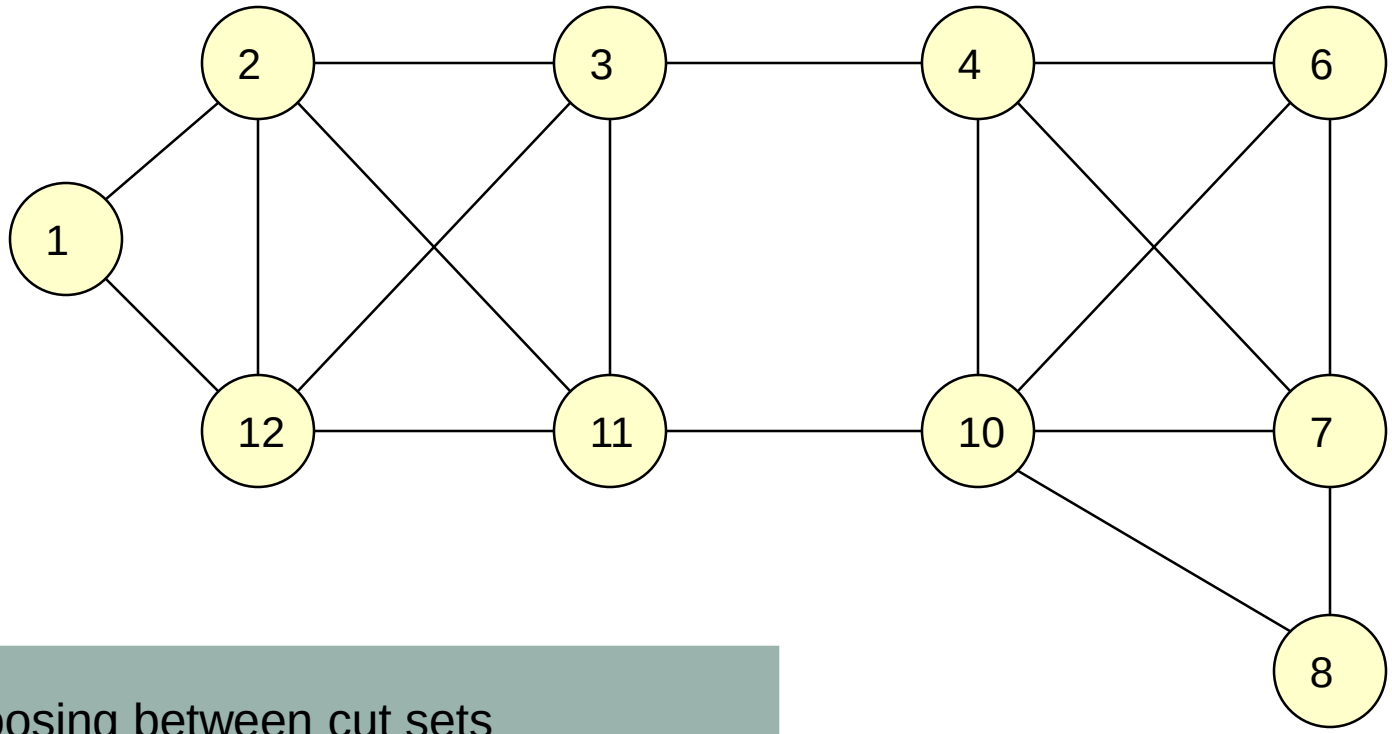


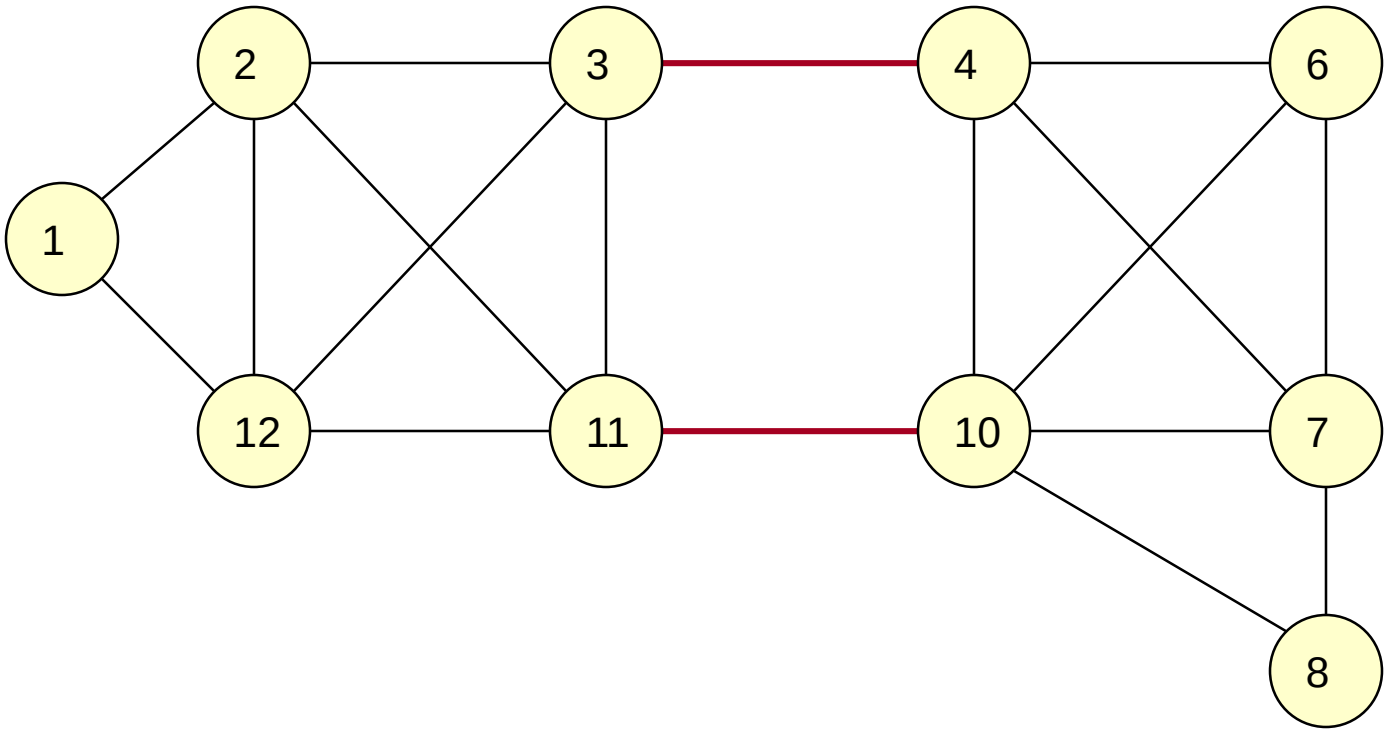

resulting clusters

# HCS: Improvements

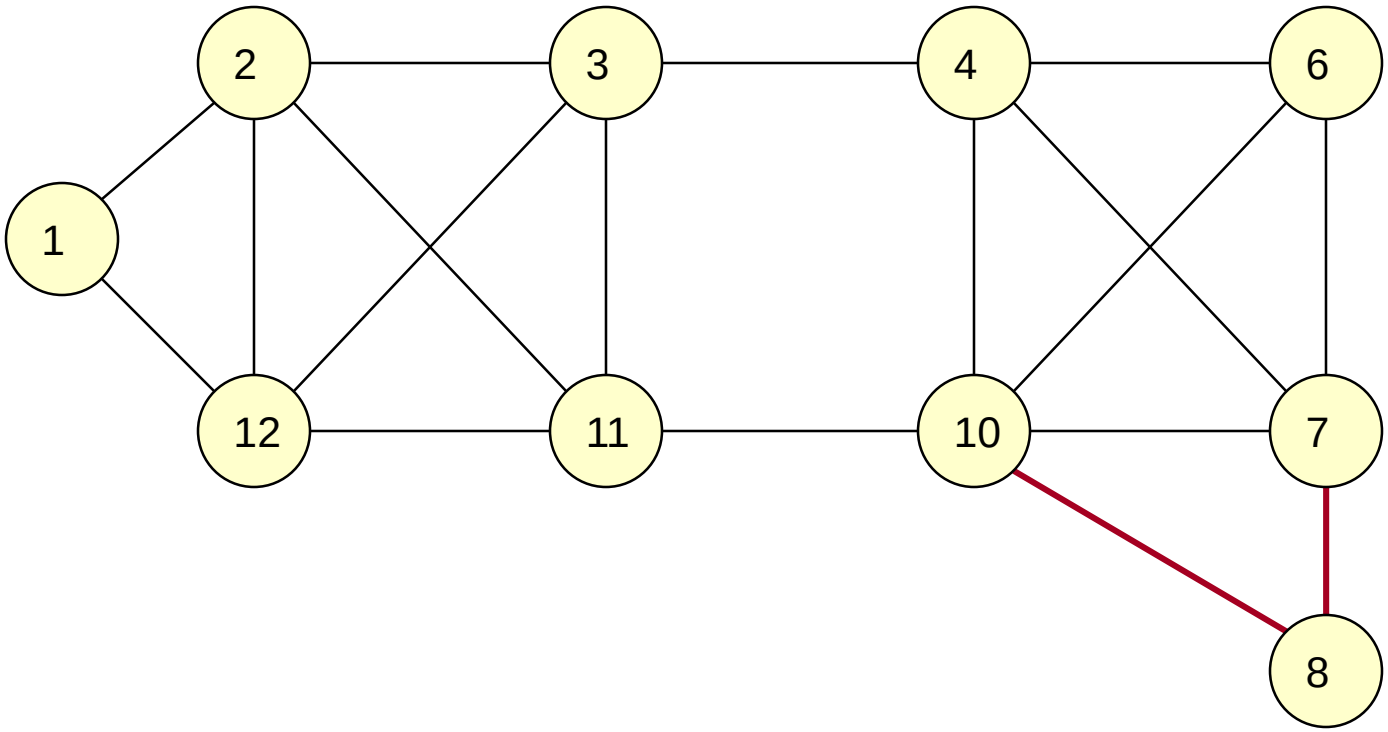



Choosing between cut sets

# 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

# References

**“A Clustering Algorithm based on Graph Connectivity”**

*By Erez Hartuv and Ron Shamir*

*RECOMB’99*

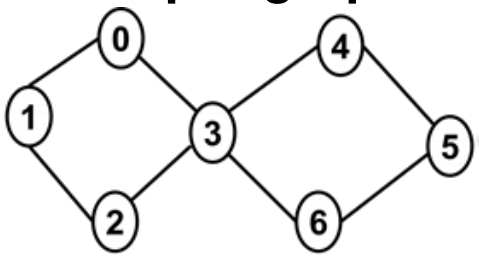
*Apply HCS in the gene expression data*

# 2.5 Detecting community – Divisive methods

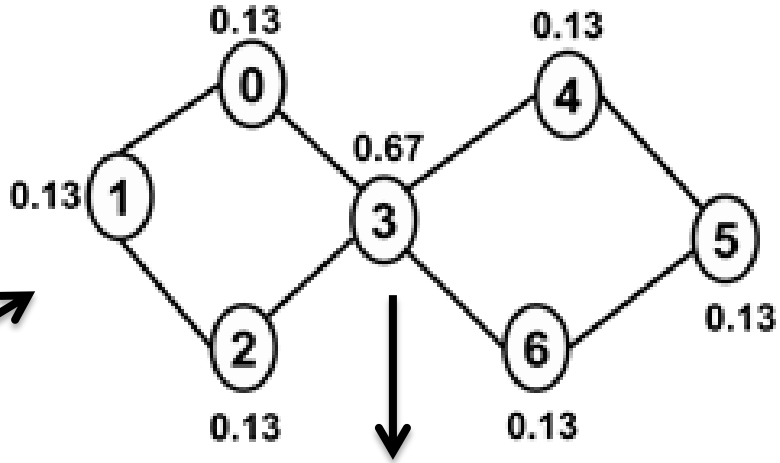
– **Divisive methods**: iteratively calculate *the cutting nodes/edges* based on some measures, e.g. betweenness scores.

# Divisive methods
# Vertex Betweenness Clustering

**Given Input graph G**



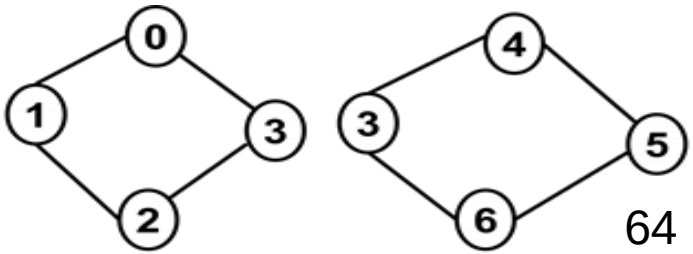

**Betweenness for each vertex**




**Repeat until highest vertex betweenness ≤ μ**

**1. Disconnect graph at selected vertex (e.g., vertex 3 )**
**2. Copy vertex to both Components**

**Select vertex v with the highest betweenness**
**E.g., Vertex 3 with value 0.67**

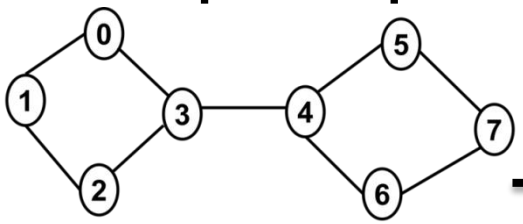


# Divisive methods

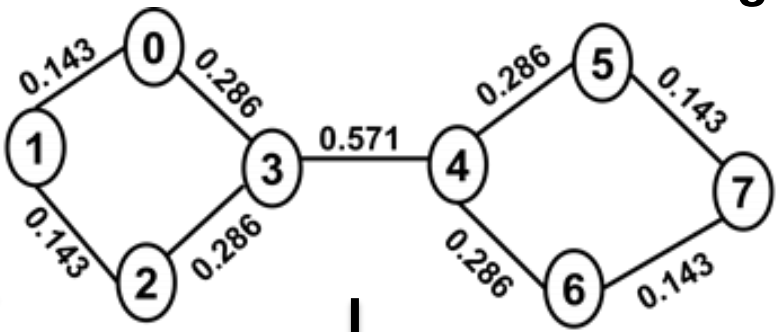
## Edge-Betweenness Clustering

## *Girvan and Newman Algorithm*

**Given Input Graph G**

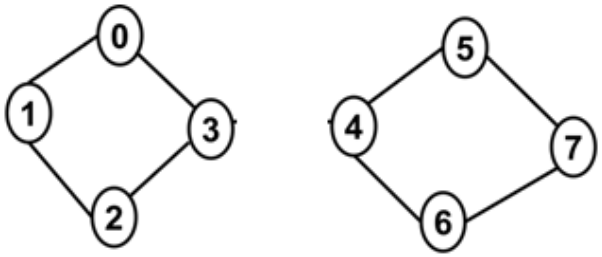



**Betweenness for each edge**




**Select edge with Highest Betweenness E.g., edge (3,4) with value 0.571**

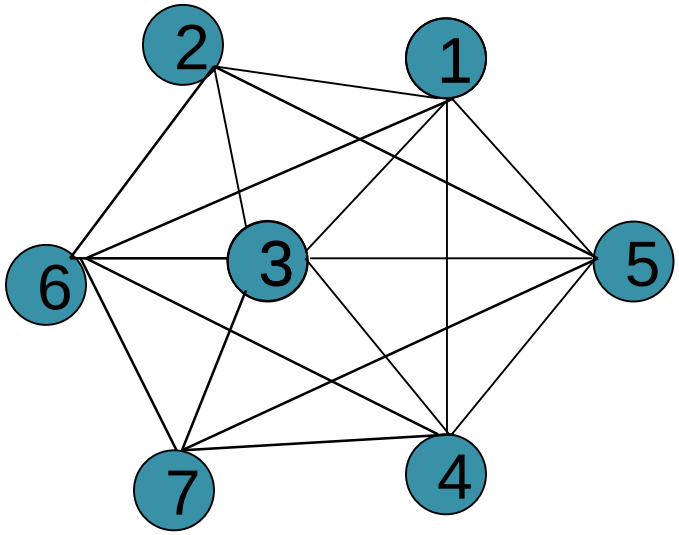
**Disconnect graph at selected edge (E.g., (3,4 ))**




**Repeat until highest edge betweenness ≤ μ**

# 2.6 LCMA: Local Clique Merging Algorithm

- Observation that a maximal dense region covering vertices $\{v_1, \ldots, v_k\}$ in $G_{ppi}$ must necessarily contain the local cliques (if any) of the vertices from $\{v_1, \ldots, v_k\}$.




{1, 3, 4, 5},

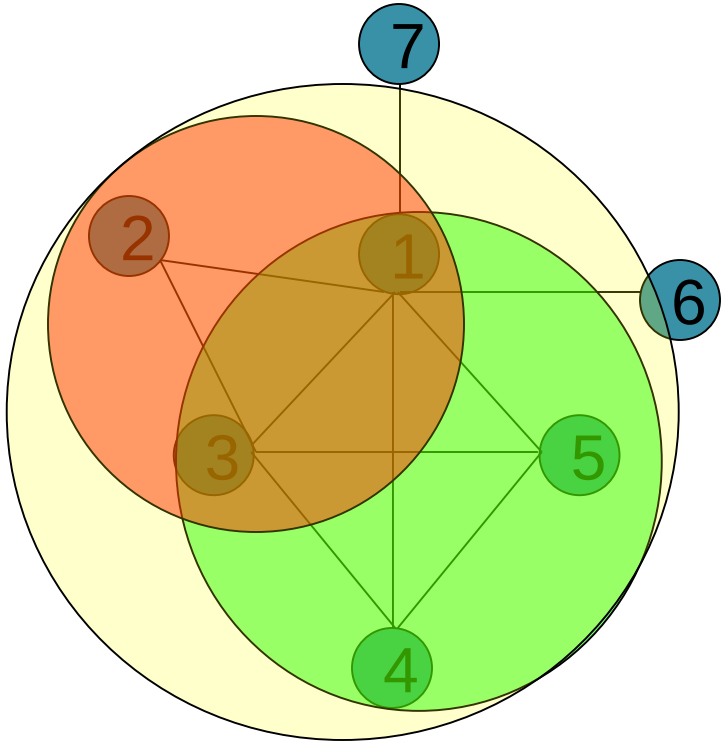
{1,2, 3, 6},

{1,2, 3, 5},

{7, 3, 4, 5},

{7, 3, 4, 6},

# Local Clique Merging Algorithm (LCMA): from local cliques to maximal dense subgraphs



# Two Steps of LCMA Algorithm

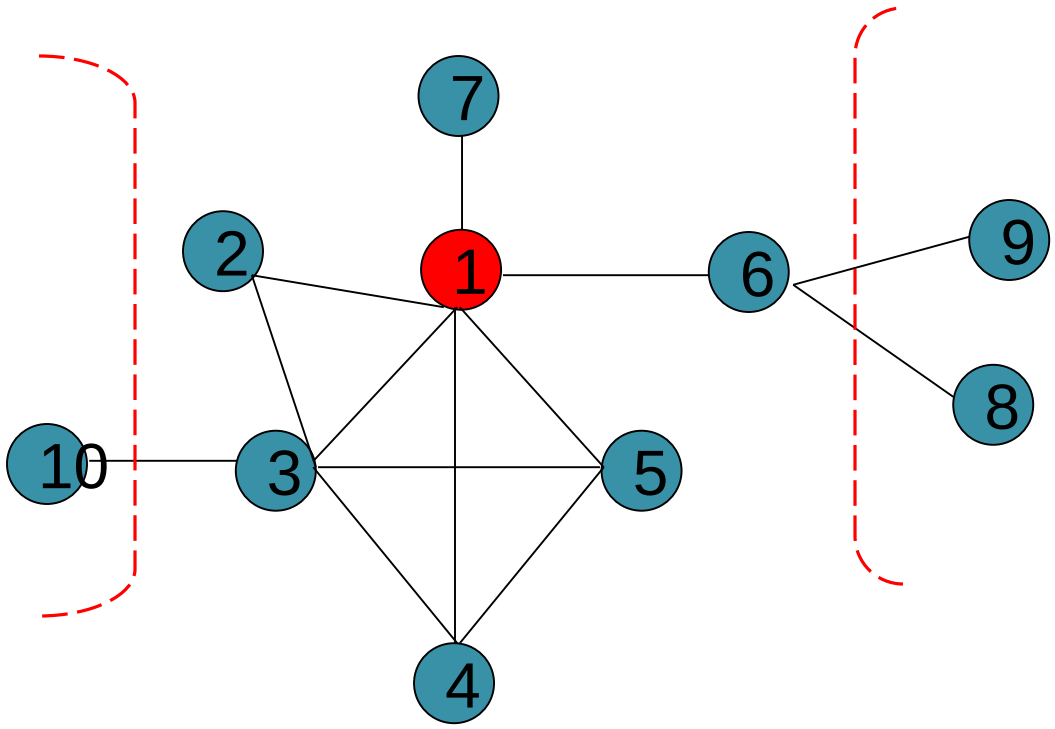
- 1. Computes the local cliques for all the vertices in $G_{ppi}$.
- 2.merge these local cliques to form maximal dense graphs.

# Local neighborhood graph

For each vertex $v_i$ from graph $G_{ppi}$, we first get its initial local neighborhood graph - namely, $v_i$, all its neighbors and the edges between the neighbors in graph $G_{ppi}$.

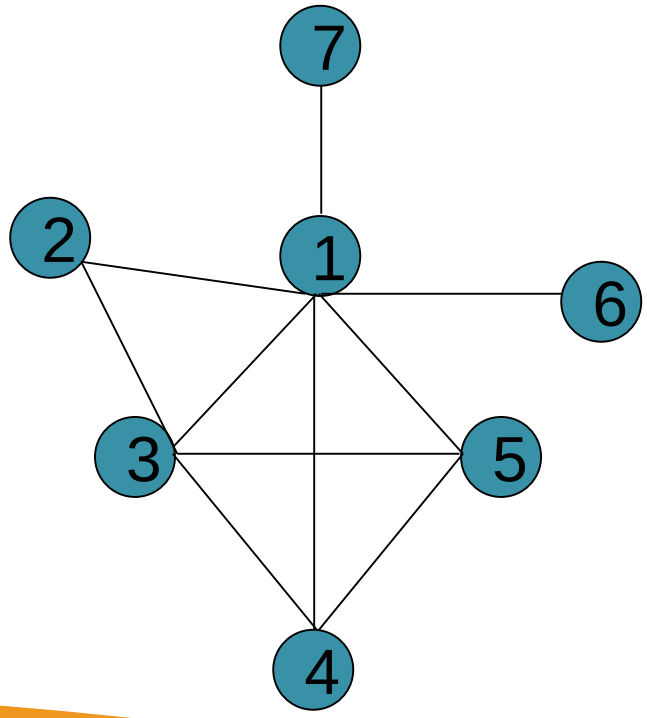
**Definition 2** *Let a graph* $G = (V, E)$. *For each vertex* $v_i \in V$, *its local neighborhood graph* $G_{v_i} = (V_{v_i}, E_{v_i})$, *where* $V_{v_i} = \{v_i\} \cup \{v | v \in V, (v, v_i) \in E\}$, $E_{v_i} = \{(v_j, v_k) | (v_j, v_k) \in E, v_j, v_k \in V_{v_i}\}$.

# Local neighborhood graph



# LCMA 1: Mining for local cliques

- Iteratively remove the loosely connected vertices




Density=

|V|=, |E|=.

# LCMA 1: Mining for local cliques from local neighborhood graph

- For each node in its local neighborhood graph, **iteratively remove the loosely connected vertices** until the density of local neighborhood graph does not increase.

- Paper proved that **the resulting graph is a fully connected graph, namely, clique.**

# LCMA 2: Merging local cliques for maximal dense neighborhoods

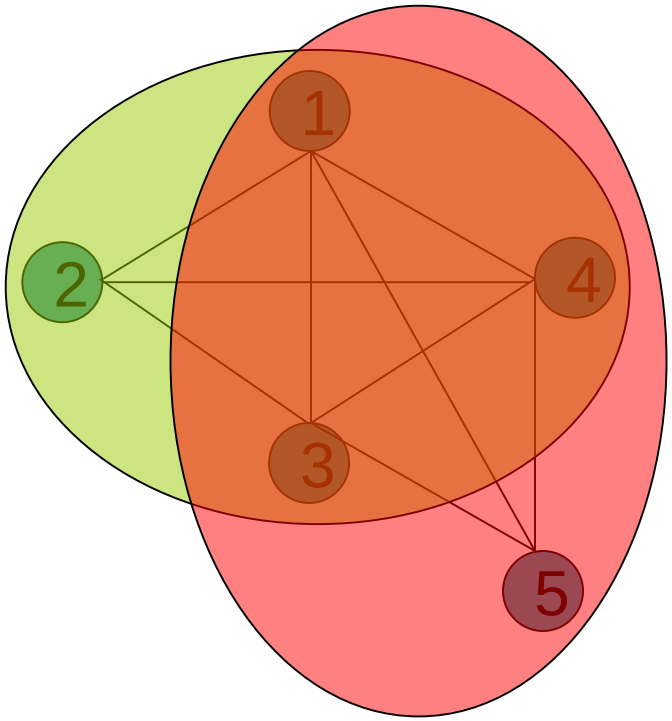
To detect *larger* dense graphs which can match the larger complexes better, the LCMA algorithm performs a merging step after the local cliques have been identified

**Definition 3** *Neighborhood Affinity. Given two neighborhoods (subgraphs) A and B, we define the Neighborhood Affinity NA between them as*

$$NA(A,B) = \frac{|A \cap B|^2}{|A| * |B|} \quad (10)$$

Equation quantifies the degree of similarity between neighborhoods. If two neighborhoods have larger intersection sets and similar sizes, then they are more similar and have bigger neighborhood affinity.

# Example of the Neighborhood Affinity




$$A = (V_1, E_1)$$

$$V_1 = \{1,2,3,4\}$$

$$E_1 = \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$$

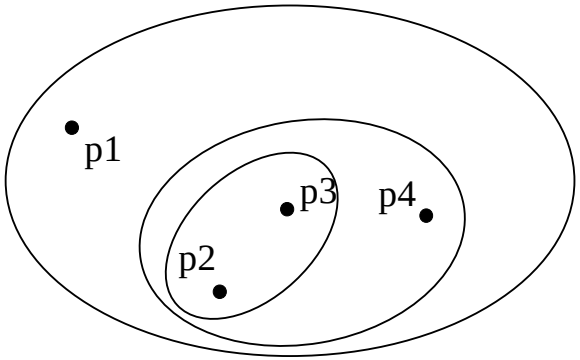
$$B = (V_2, E_2)$$

$$V_2 = \{1,3,4,5\}$$

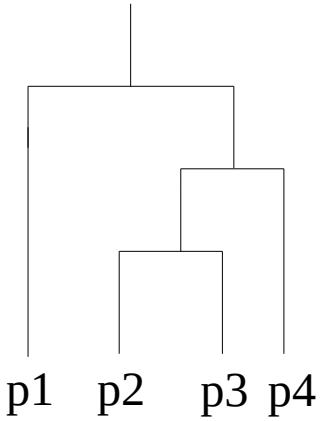
$$E_2 = \{(1,3),(1,4),(1,5),(3,4),(3,5),(4,5)\}$$

$$NA(A,B) = \frac{|A \cap B|^2}{|A| * |B|} = \frac{3*3}{4*4} = \frac{9}{16} = 0.5625$$

# 2.7 Agglomerative methods




Traditional Hierarchical Clustering

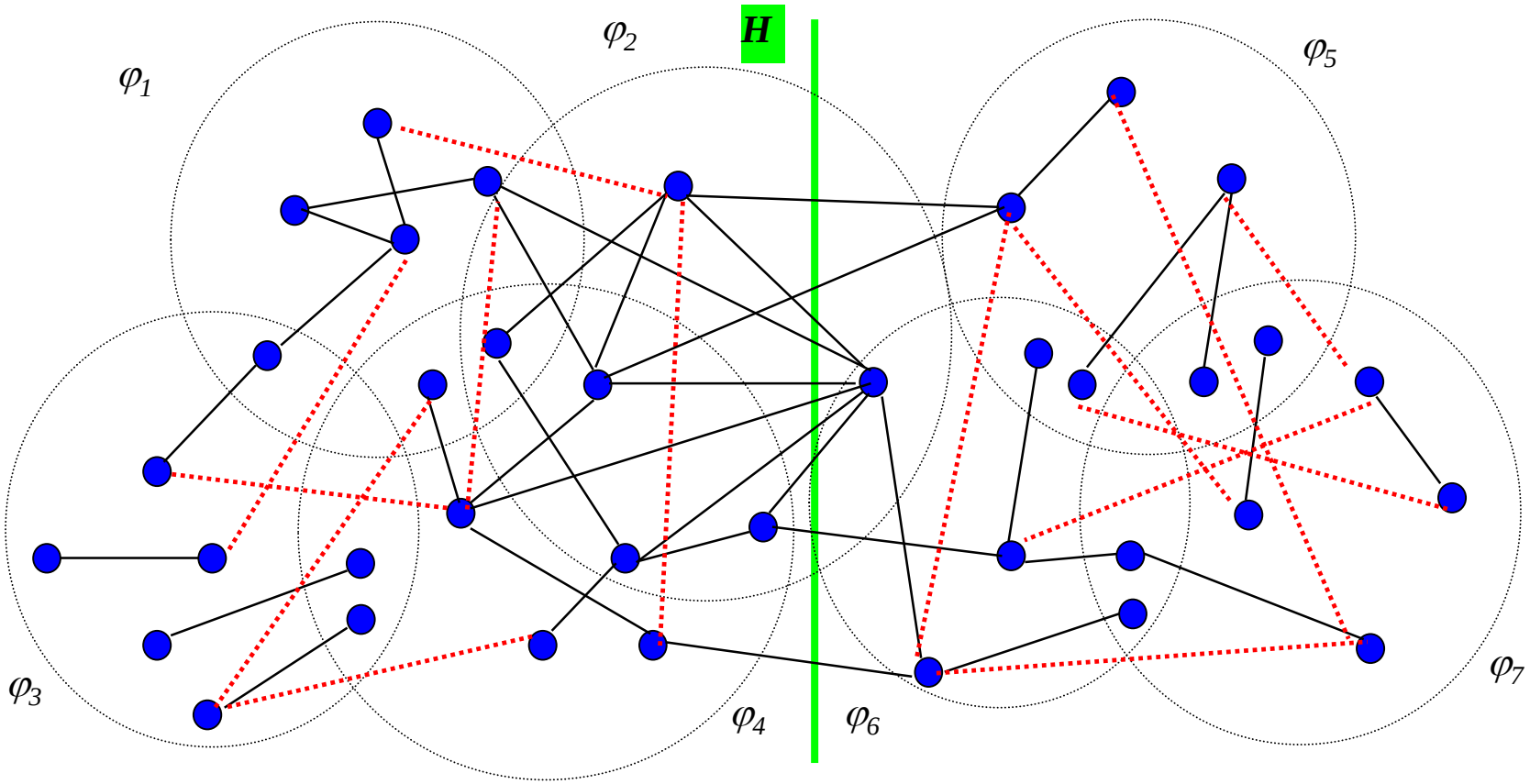



Traditional Dendrogram

# Less links among the community members

- Social entities only *interact with limited community members*. As such, *there exist communities which do not have very dense connections among all its members*. This will make existing algorithms (mostly density-based) suffer.

# *Virtual Links* enhance the connectivity among members within same communities




Physical links

Virtual links

***Virtual Links are content based***

# How to add virtual links

- For each node (crawl Web to get its additional information)
- Compute pair-wise content similarity
- If they are larger than certain threshold (i.e. average similarity among the known community members), then we regard them have virtual links
- Virtual links can be used to do friend recommendation

# Virtual Link - Predicting friendship

- Input: two people
- Output: should they be Facebook friends?

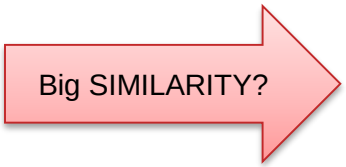
# Virtual Links

- Input: two people
- Output: should they be Facebook friends?

- **Features:**
  - **friends list**
  - **school**
  - **home town**
  - **Music**
  - **hobbies**
  - **…**

| Peter, Julia, … |
|---|
| NUS, IIT, |
| Hyderabad, India |
| Rock |
| Tennis, running |
| |
| |
| |
| … |

| Celia, Julia, … |
|---|
| NUS, Tsinghua |
| Beijing, China |
| Rock |
| Tennis, |
| |
| |
| |
| … |

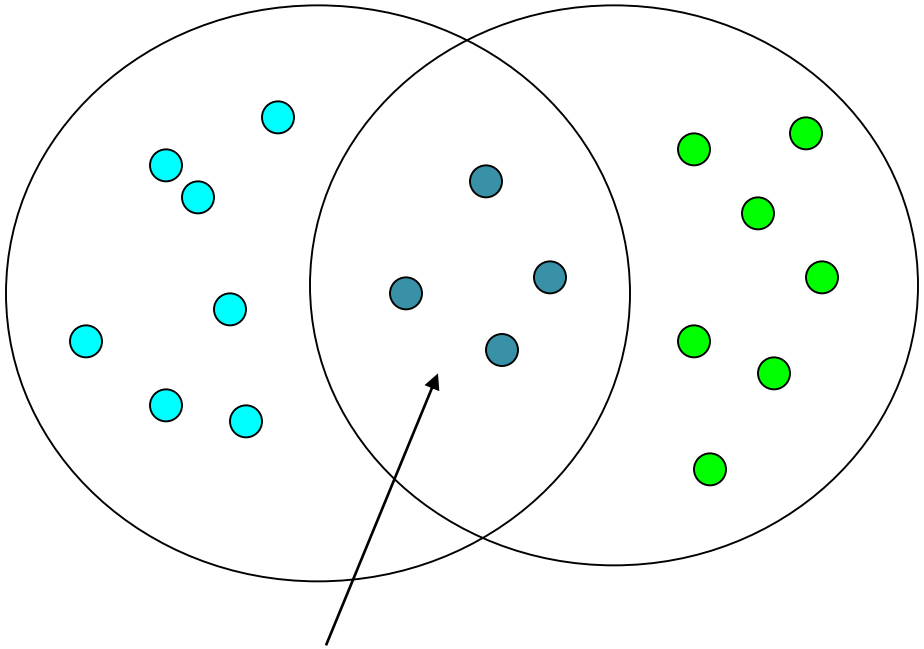



# Merge small dense graphs by computing similarity measures

- Communities can consist of the people from different small dense graphs . It is thus necessary to combine them together to form those bigger communities.
- We evaluate the similarities between graphs by the following *three different similarity measures*.

# Vertex overlapping based similarity

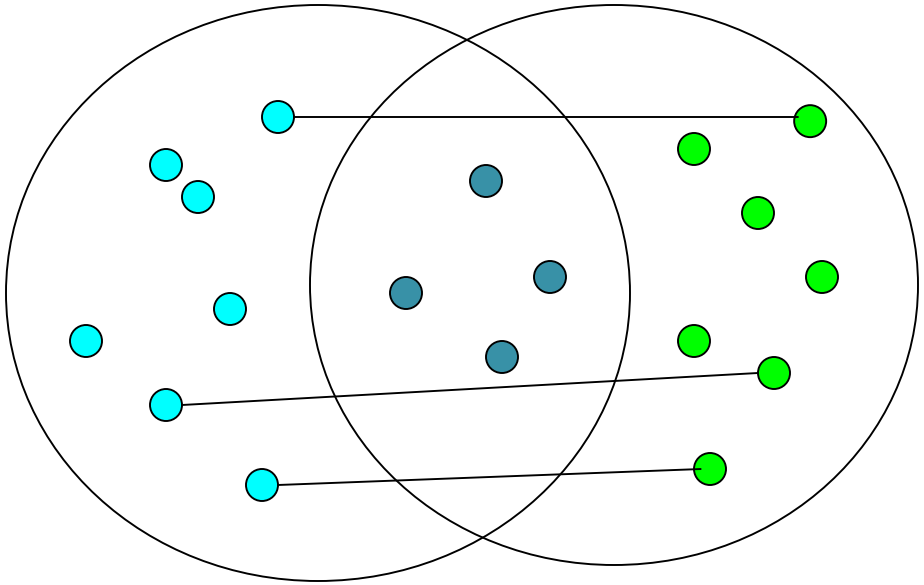
$$Vex\_sim(\varphi_i, \varphi_j) = \frac{|V_i \bigcap V_j|}{|V_i \bigcup V_j|} / K_{vex}$$



If two graphs share a high proportion of members, then they should be combined into same community.

# Physical link based similarity

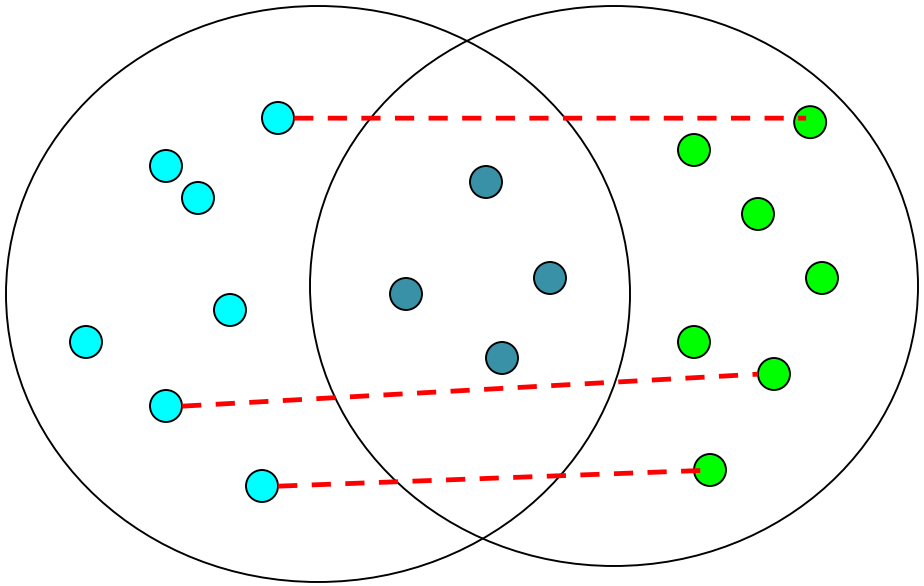
$$PL\_sim(\varphi_i,\varphi_j)=\frac{|\{(v_i,v_j)\,|\,(v_i,v_j)\in\varphi_k,k\neq i,k\neq j,v_i\in V_i\setminus V_j,v_j\in V_j\setminus V_i\}|}{|V_i\setminus V_j|*|V_j\setminus V_i|}/K_{PL}$$



Physical link based similarity: evaluates how closely the members from different graphs interact with each other.

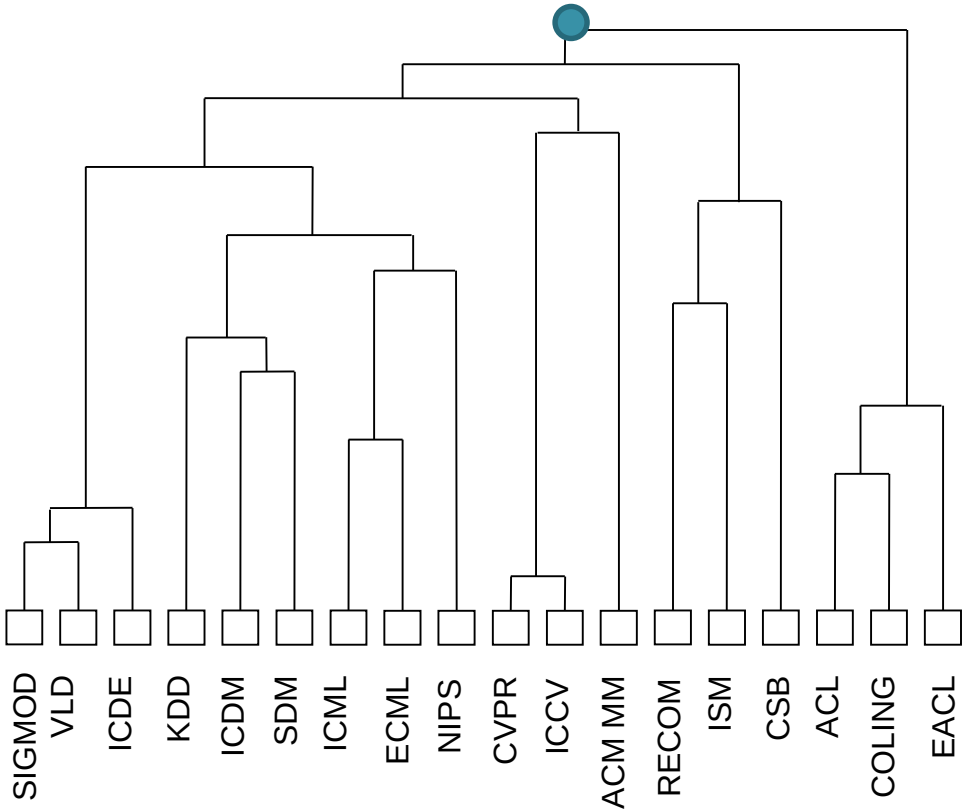
# Virtual link based similarity

$$VL\_sim(\varphi_i, \varphi_j) = \frac{|\{(v_i, v_j) | v_i \in \varphi_i, v_j \in \varphi_j, consim(v_i, v_j) > \delta, v_i \in V_i \setminus V_j, v_j \in V_j \setminus V_i\}|}{|V_i \setminus V_j| * |V_j \setminus V_i|} / K_{VL}$$



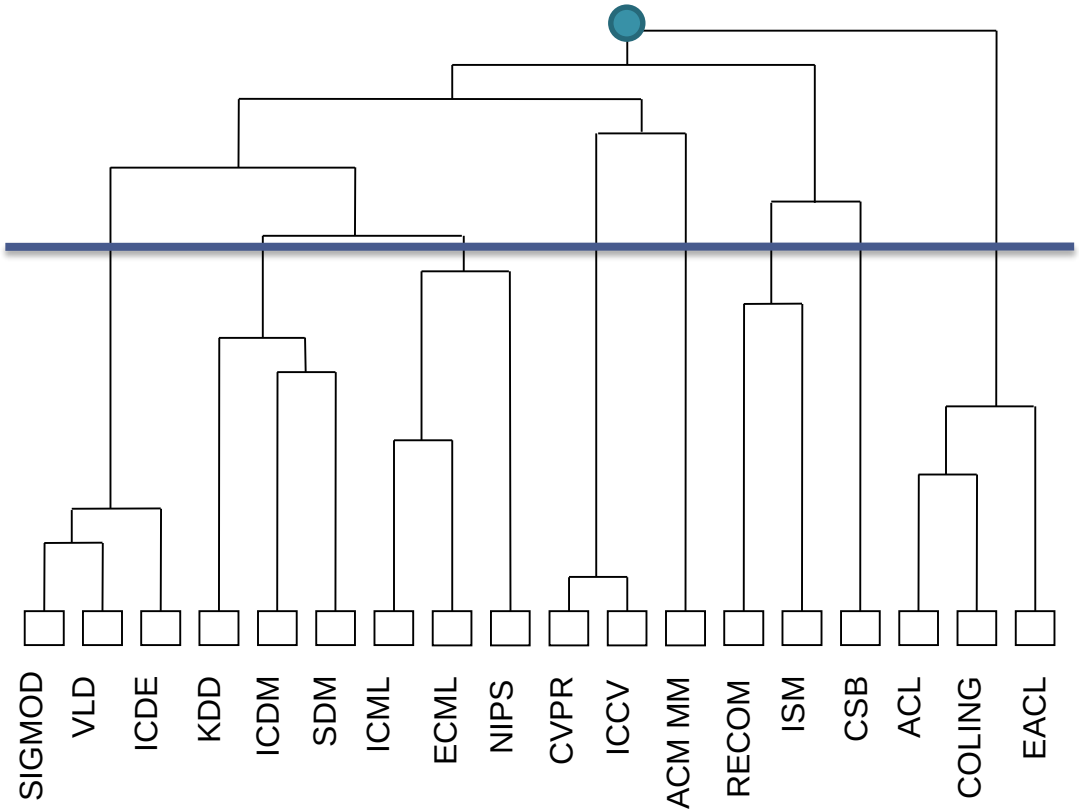
Connect those people from different graphs whose content similarity is equal to or higher than the average similarity between people within randomly selected events.

# Overall Idea of ECODE Algorithm




- Hierarchical clustering approach
- Detect similar graphs in terms of *overlapping vertices* and *physical/virtual links*, and then merging them to form bigger communities

# ECODE Algorithm




**Automatic stopping criteria:** automatically terminates when the *quality of the detected communities become maximal*

# Compute the quality of the current level of the tree
# Link based method

- The hierarchical clustering will result in a tree (one big community). The merging process can be stopped if *the current merging step does not improve the quality of the current level of tree*.
- Newman has proposed a quality function $Q$ (**modularity**) to evaluate the goodness of a tree

$$Q = \sum_i (e_{ii} - a_i^2)$$

where $e_{ii}$ is the number of links in the same group connecting the vertices (intralinks) and $a_i$ is the sum of edges from the vertices in group $i$ to another group $j$ (interlinks)

# Compute the quality of the current level of the tree
# Content based method

- There are many interactions across different communities, instead of using the physical links, we use the content/feature-based approach.

$$Q = \sum_i (cossim(i,i) - \sum_j cossim(i,j)^2)$$

- It favors a community substructure which has in overall bigger intra-similarity and less inter-similarity in terms of their topics and content.
- The algorithm can stop at a level of tree with the maximal $Q$ value.

# References


1. Xiao-Li Li, Chuan-Sheng Foo, Kar Leong Tew, See-Kiong Ng, "Searching for Rising Stars in Bibliography Networks", DASFAA 2009, Australia.
2. Xiao-Li Li, Soon-Heng Tan, Chuan-Sheng Foo and See-Kiong Ng. "Interaction Graph Mining for Protein Complexes Using Local Clique Merging." in *Genome Informatics, Vol. 16, No.2.* 2005.
3. Xiao-Li Li, Aloysius Tan, Philip S. Yu, See-Kiong Ng, ECODE: Event-Based Community Detection from Social Networks, DASFAA 2011, Hong Kong.
4. Nagiza F. Samatova, William Hendrix, John Jenkins, Kanchana Padmanabhan, Arpan Chakraborty, Practical Graph Mining With R

# Thank You

Contact: xlli@i2r.a-star.edu.sg if you have questions