Department of Computer and IT Engineering
University of Kurdistan

Complex Networks

Communities

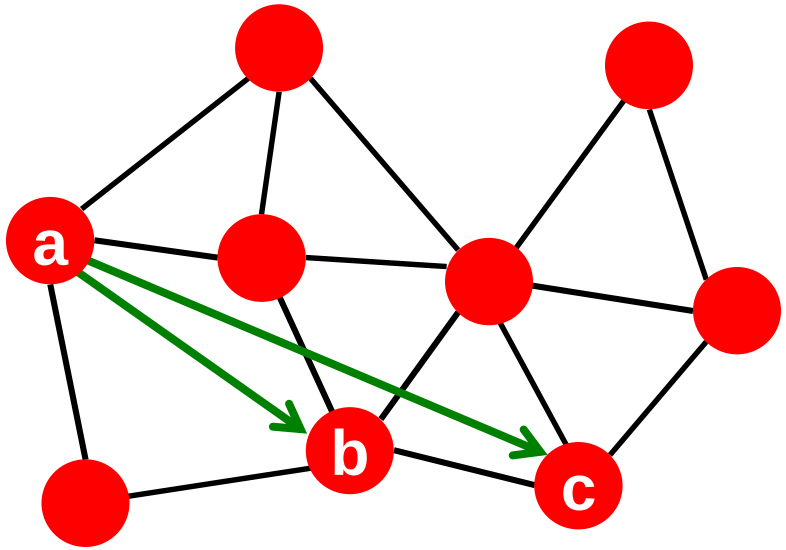
By: Dr. Alireza Abdollahpouri

Strength of Weak Ties

- **How people find out about new jobs?**
  - Mark Granovetter, part of his PhD in 1960s
  - People find the information through personal contacts
- **But:** Contacts were often **acquaintances** rather than close friends
  - **This is surprising:**
    - One would expect your friends to help you out more than casual acquaintances when you are between the jobs
    - why is it that distant acquaintances are most helpful?

Triadic Closure

- **Which edge is more likely to form, a-b or a-c?**

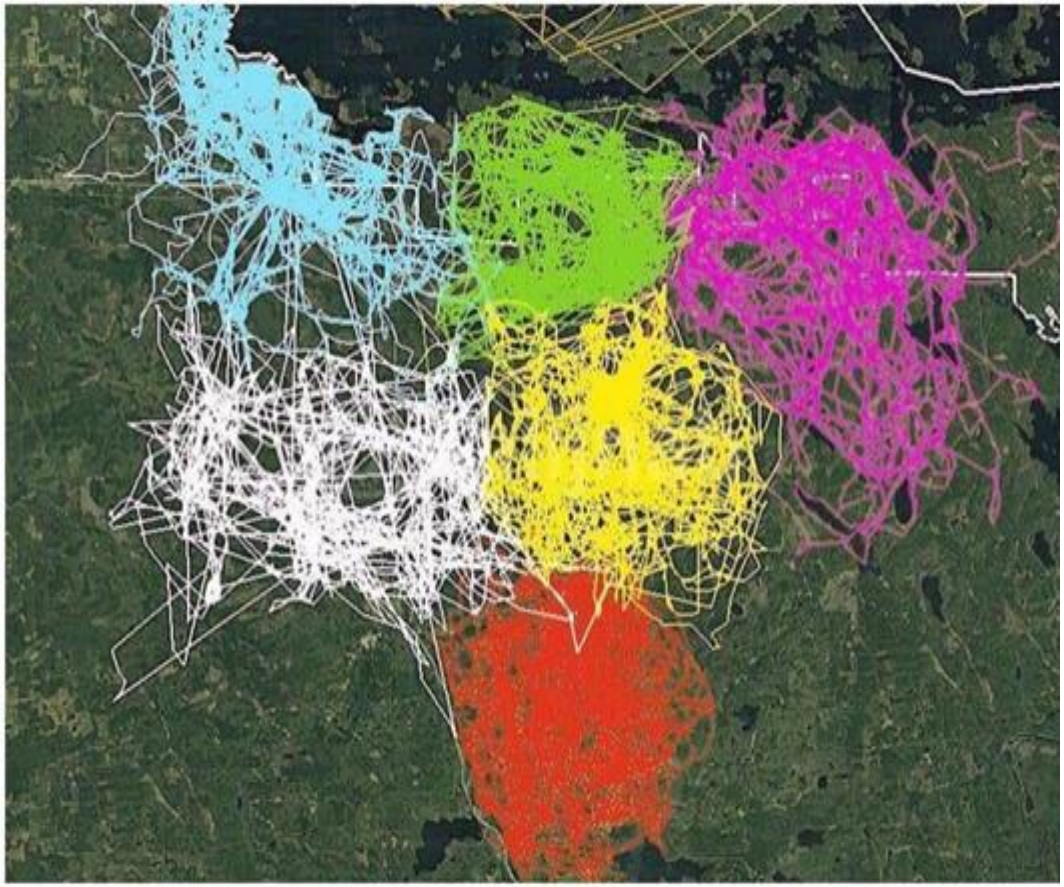


- **Triadic closure:** If two people in a network have a friend in common there is an increased likelihood they will become friends themselves

The concept of community

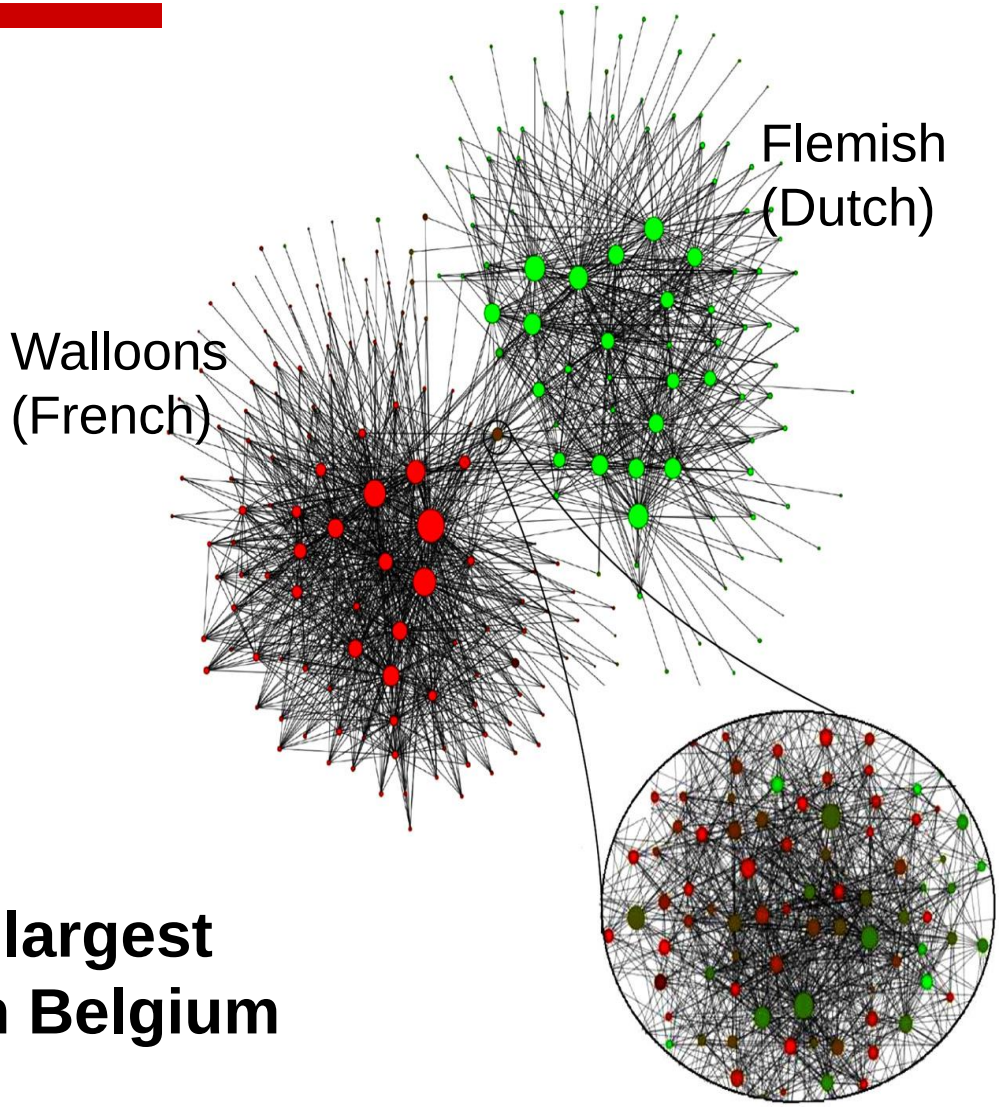
Birds of a feather, flock together

The concept of community



The images received with the aid of GPS mounted on the six wolf cattle flocks in the American National Park show how they respect each other's territory.

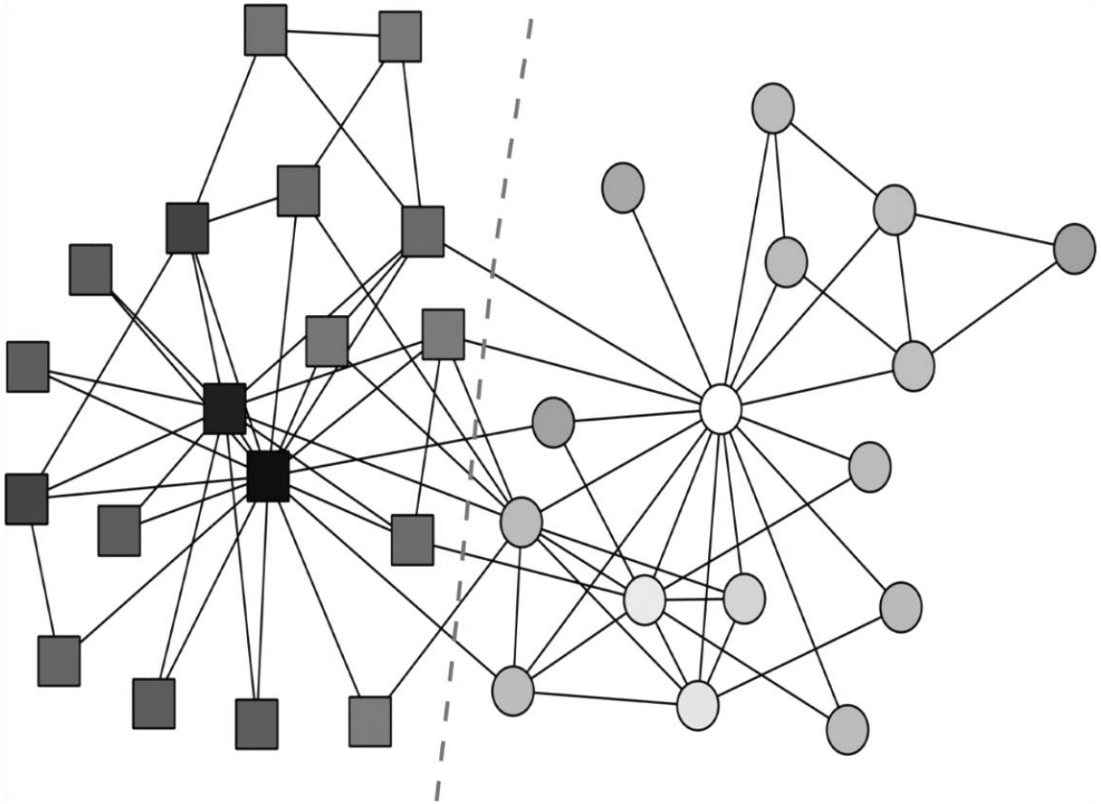
The concept of community



call patterns of one of the largest mobile phone operators in Belgium

The concept of community

Zachary's Karate Club



Similarity of Connected Nodes in Social Networks

- Race
- Religion
- Education
- Income level
- Job and skills
- Language
- Interests and preferences

More examples of Communities

- **Network: World Wide Web:**
  Communities: Sites on related topics

- **Network: Friendship network:**
  Communities: Group formation among people

- **Network: Metabolic networks:**
  Communities: Functional modules

- **Network: Collaboration network:**
  Communities: Research fields

Community- Definition

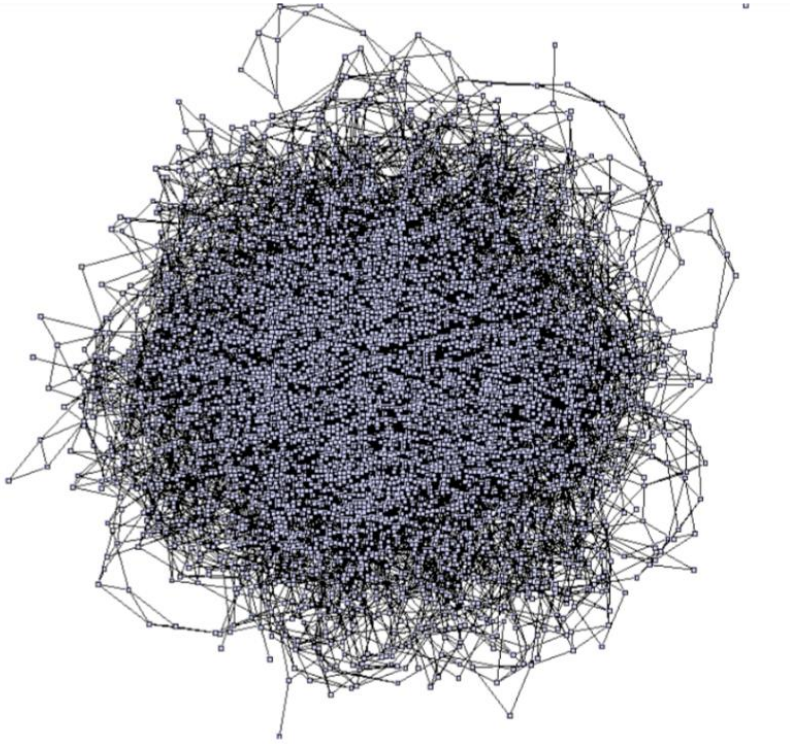
A variety of definitions of community/cluster/module exist:

- **A group of nodes which share common properties and/or play a similar role within the graph [Fortunato, 2010].**

- **A subset of nodes within which the node-node connections are dense, and the edges to nodes in other communities are less dense [Girvan & Newman, 2002].**

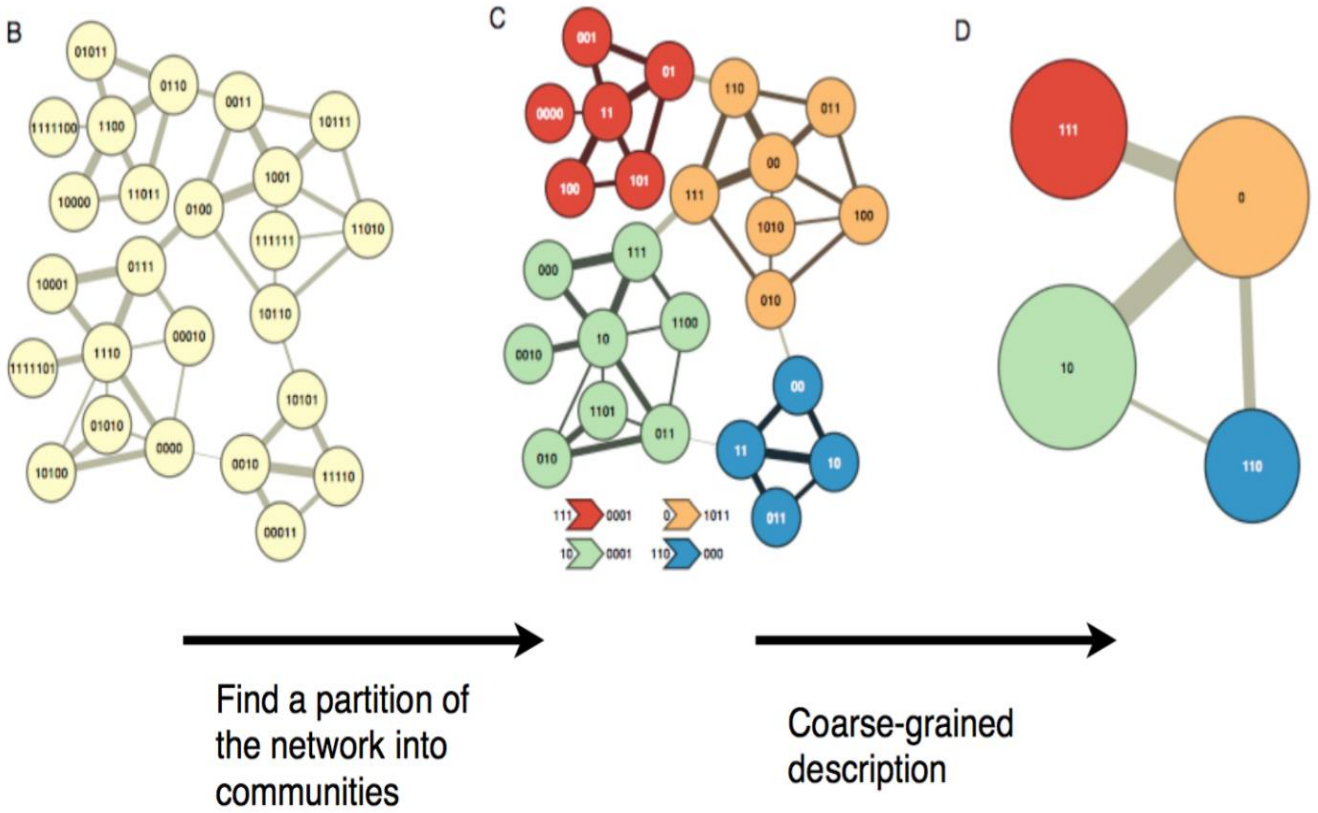
Why community detection?

Graphs help us to comprehend in a visual way the global organization of the system. This works extremely well when the graph is small but, as soon as the system is made of hundreds or thousands of nodes, a brute force representation typically leads to a meaningless cloud of nodes.



Why community detection?

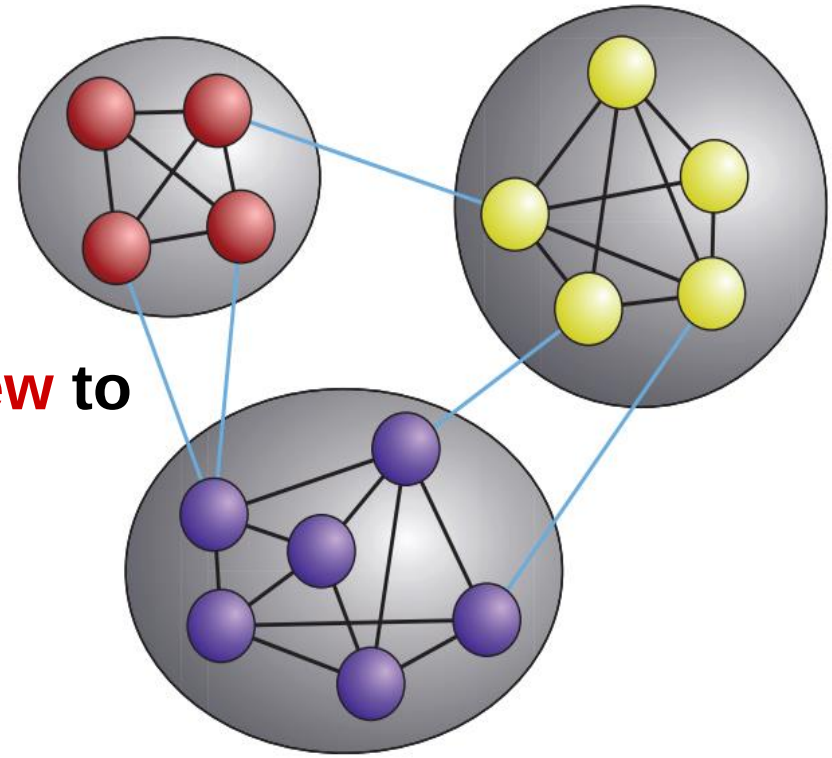
Uncovering communities/modules helps to change the resolution of the representation and to draw a readable map of the network



Community- Definition

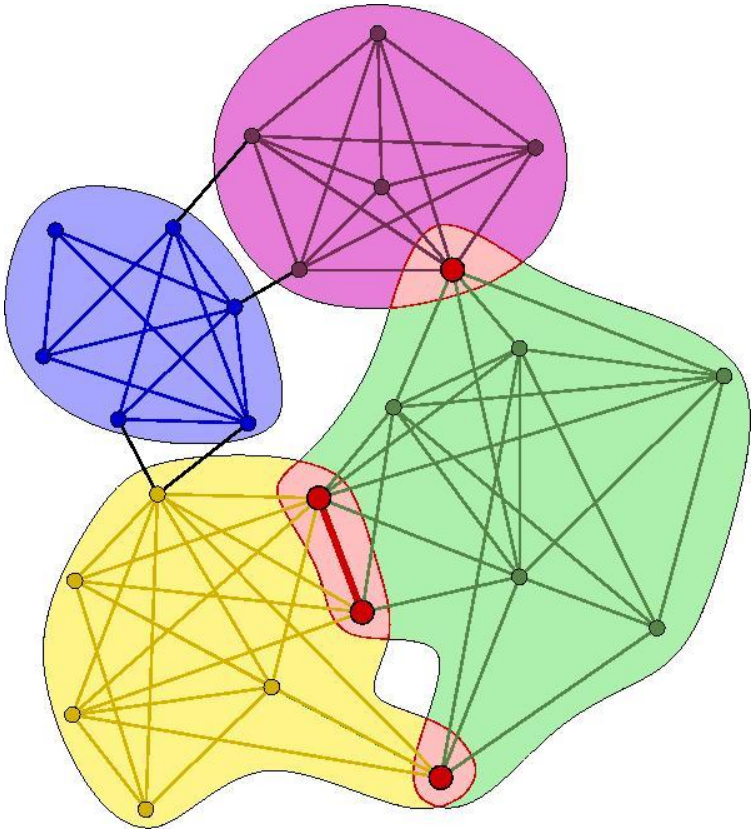
- **Networks of tightly connected groups**

- **Network communities:**
  - **Sets of nodes with lots of connections inside and few to outside (the rest of the network)**

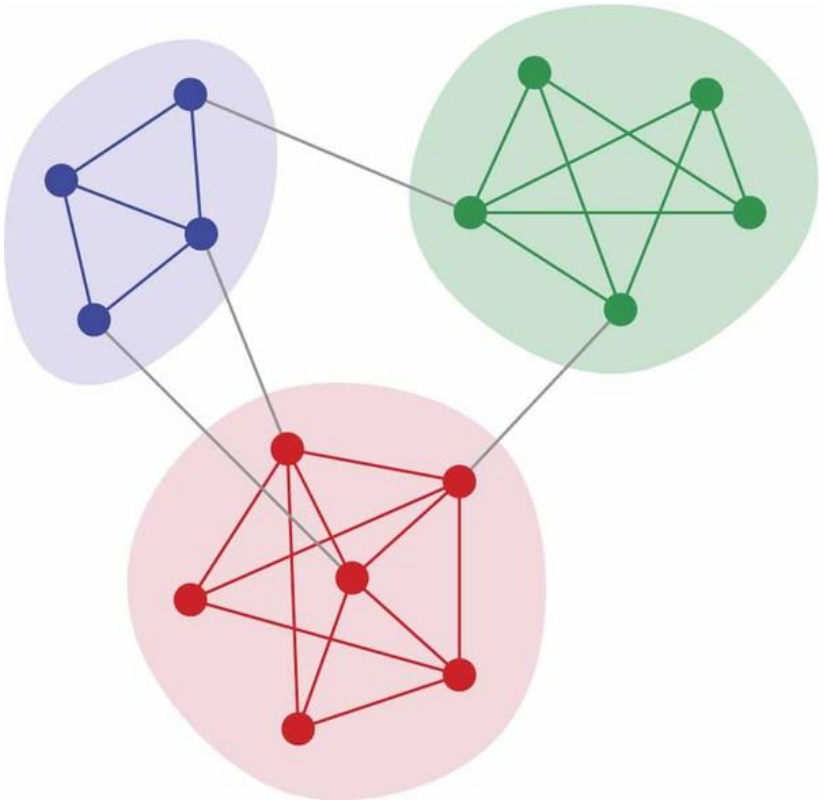


Communities, clusters, groups, modules

Overlapping vs. Disjoint Communities



Overlapping Communities



Disjoint Communities

Community detection- Questions

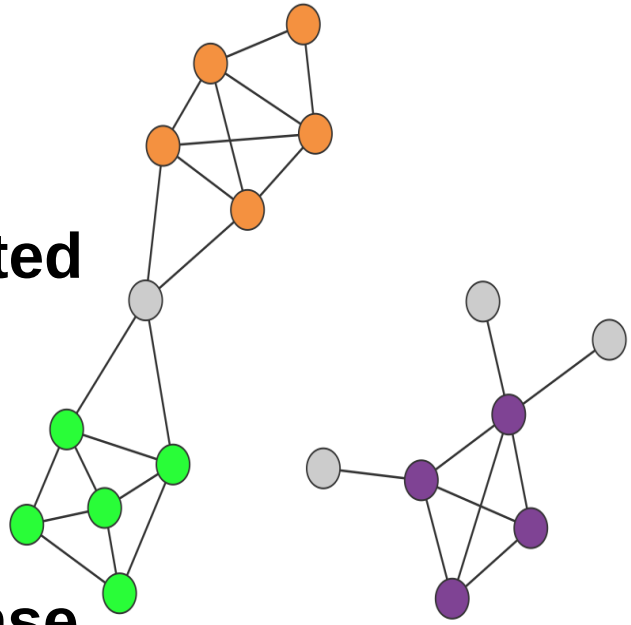
- What do we really mean by a community?
- How many communities are in a network?
- How many different ways can we partition a network into communities?
- How can we detect communities?
- How do communities evolve and how can we study evolving communities?
- How can we evaluate detected communities?

Community Hypotheses

H1: A network's community structure is uniquely encoded in its wiring diagram (there is a ground truth about a network's community organization, that can be uncovered by inspecting A_{ij}).

H2: Connectedness Hypothesis

A community corresponds to a connected subgraph.



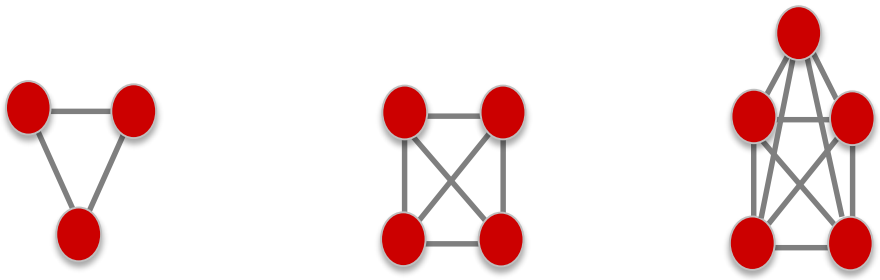
H3: Density Hypothesis

Communities correspond to locally dense neighborhoods of a network.

Basics of Communities

Cliques as communities

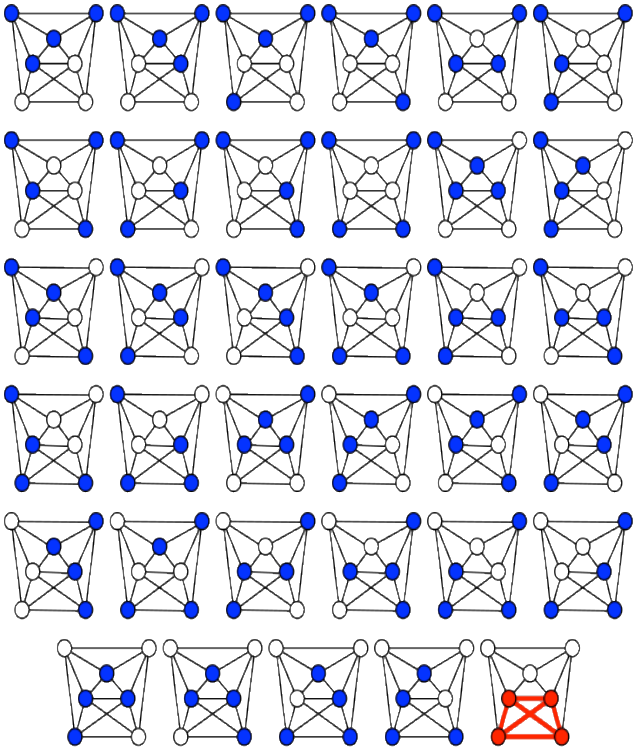
A clique is a complete subgraph of *k*-nodes



Basics of Communities

Cliques as communities

- Triangles are frequent; larger cliques are rare.
- Communities do not necessarily correspond to complete subgraphs, as many of their nodes do not link directly to each other.
- Finding the cliques of a network is computationally rather demanding, being a so-called NP-complete problem.



Basics of Communities

Strong and weak communities

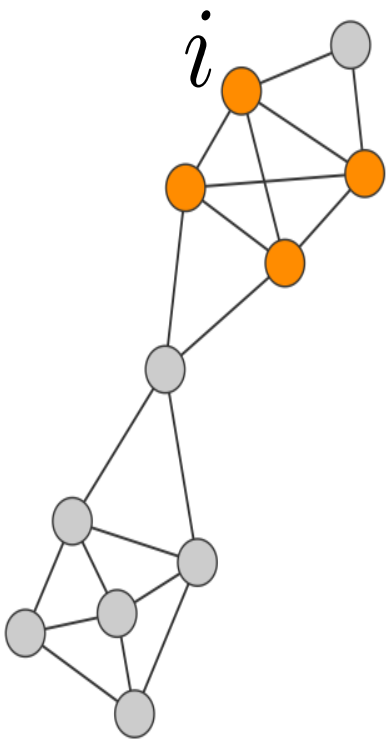
$$k_i^{int} = 3$$

$$k_i^{ext} = 1$$

Consider a connected subgraph C of N_c nodes

Internal degree, k_i^{int}: set of links of node i that connects to other nodes of the same community C.

External degree k_i^{ext}: the set of links of node i that connects to the rest of the network.



If $k_i^{ext}=0$: all neighbors of i belong to C, and C is a good community for i.

If $k_i^{int}=0$, all neighbors of i belong to other communities, then i should be assigned to a different community.

Basics of Communities

Strong community:

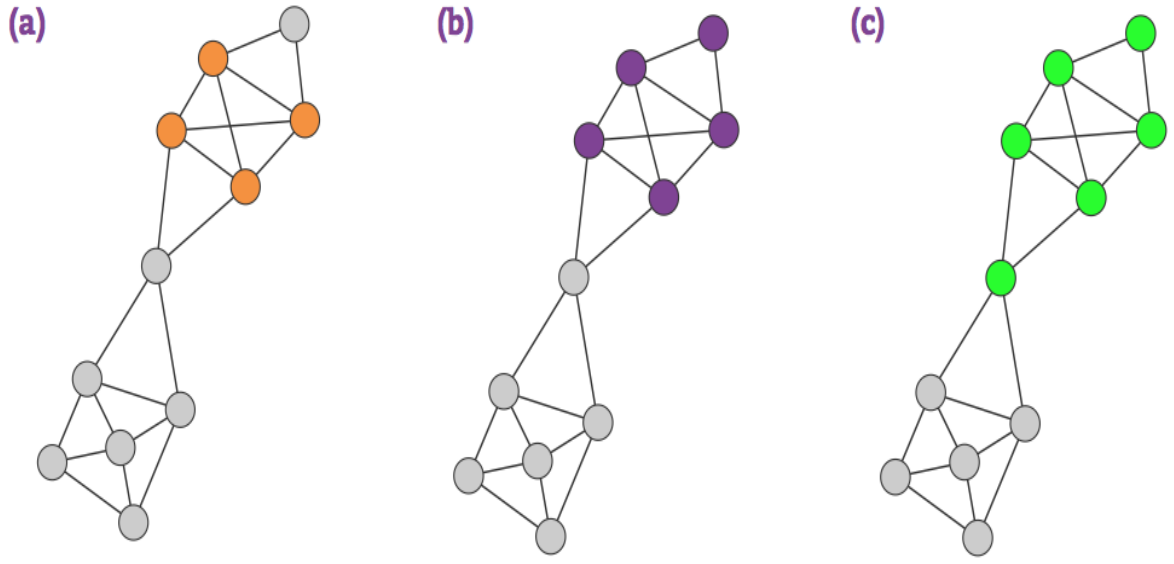
Each node of C has more links within the community than with the rest of the graph.

$$k_i^{\text{int}}(C) > k_i^{\text{ext}}(C)$$

Weak community:

The total internal degree of C exceeds its total external degree,

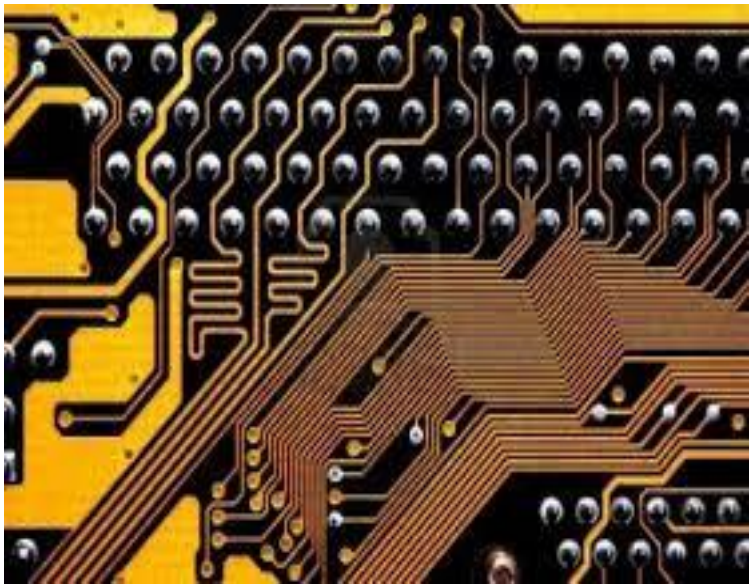
$$\sum_{i \in C} k_i^{\text{in}}(C) > \sum_{i \in C} k_i^{\text{out}}(C)$$



Clique Strong Weak

Graph partitioning

Partition the full wiring diagram of an integrated circuit into smaller subgraphs, so that they minimize the number of connections between them.



2.5 billion transistors

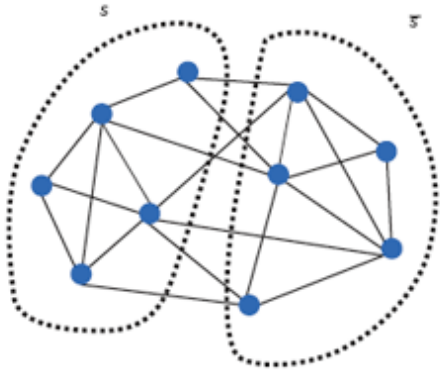
Graph partitioning

How many ways can we partition a network into 2 communities? (Graph bisection)

Divide a network into two non-overlapping subgraphs, such that the number of links between the nodes in the two groups is minimized.

Two subgroups of size n_1 and n_2. Total number of combinations:

$$\frac{N!}{n_1!n_2!}$$



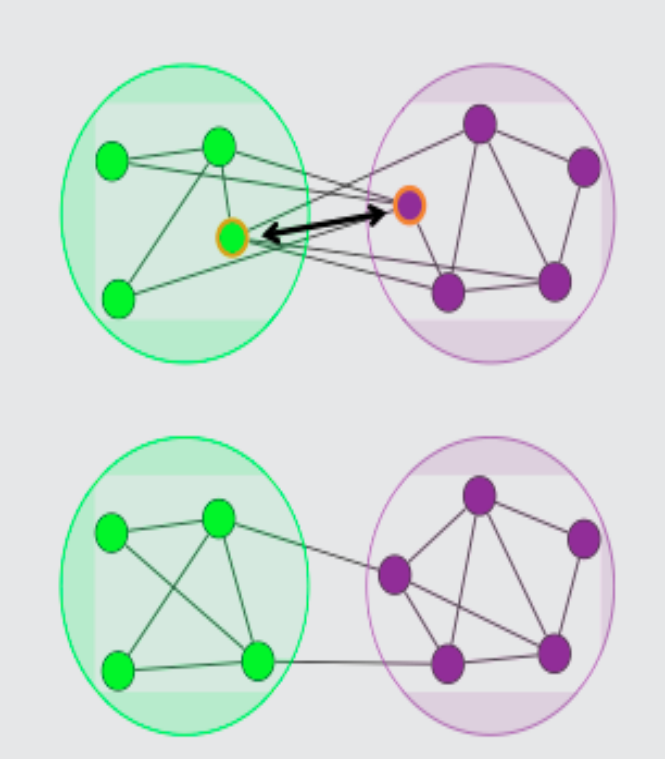
N=10 → 256 partitions (1 ms)

N=100→ 10^{26} partitions (10^{21} years)

Graph partitioning

Kerninghan-Lin Algorithm for graph bisection

- Partition a network into two groups of predefined size. This partition is called cut.
- Inspect each a pair of nodes, one from each group. Identify the pair that results in the largest reduction of the cut size (links between the two groups) if we swap them
- Swap them.
- If no pair deduces the cut size, we swap the pair that increases the cut size the least.
- The process is repeated until each node is moved once.

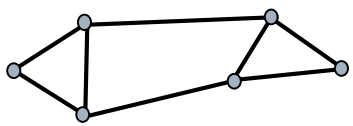


Spectral Clustering Algorithms

- **Three basic stages:**
  - **1) Pre-processing**
    - Construct a matrix representation of the graph
  - **2) Decomposition**
    - Compute eigenvalues and eigenvectors of the matrix
    - Map each point to a lower-dimensional representation based on one or more eigenvectors
  - **3) Grouping**
    - Assign points to two or more clusters, based on the new representation

Spectral Partitioning Algorithm

- **1) Pre-processing:**
  - Build Laplacian matrix L of the graph (L = D-A)



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

- **2) Decomposition:**
  - Find eigenvalues λ and eigenvectors x of the matrix L

$\lambda =$

0.0
1.0
3.0
3.0
4.0
5.0

$X =$

0.4	0.3	-0.5	-0.2	-0.4	-0.5
0.4	0.6	0.4	-0.4	0.4	0.0
0.4	0.3	0.1	0.6	-0.4	0.5
0.4	-0.3	0.1	0.6	0.4	-0.5
0.4	-0.3	-0.5	-0.2	0.4	0.5
0.4	-0.6	0.4	-0.4	-0.4	0.0

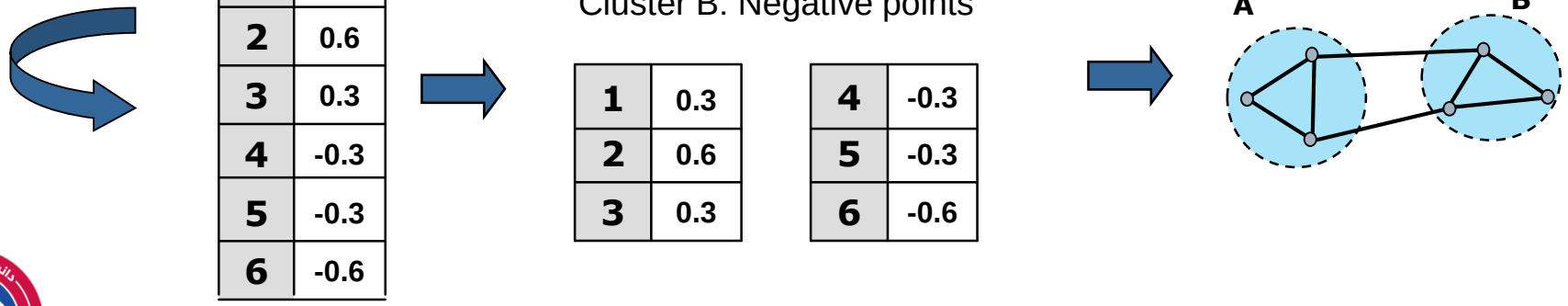
1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6

- Map vertices to corresponding components of λ_2

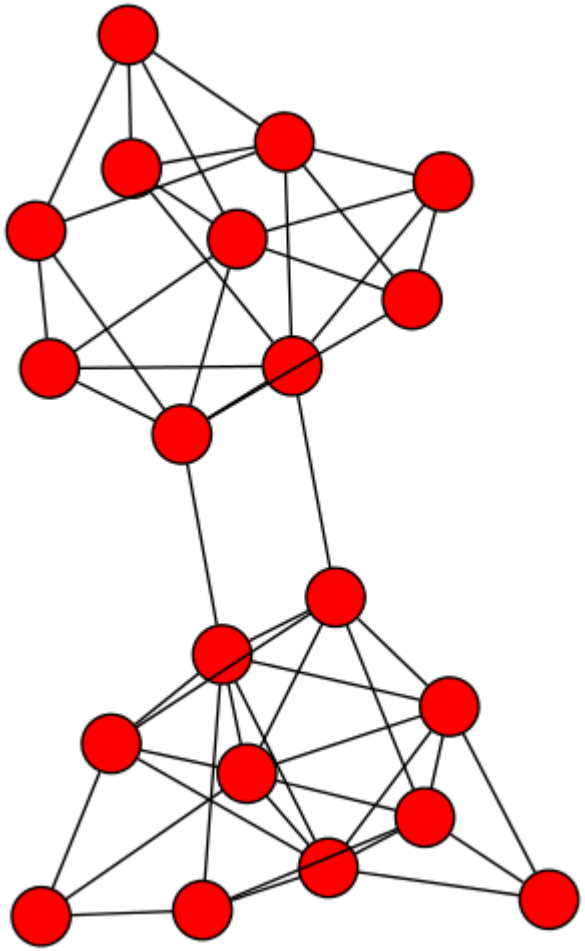
How do we now find the clusters?

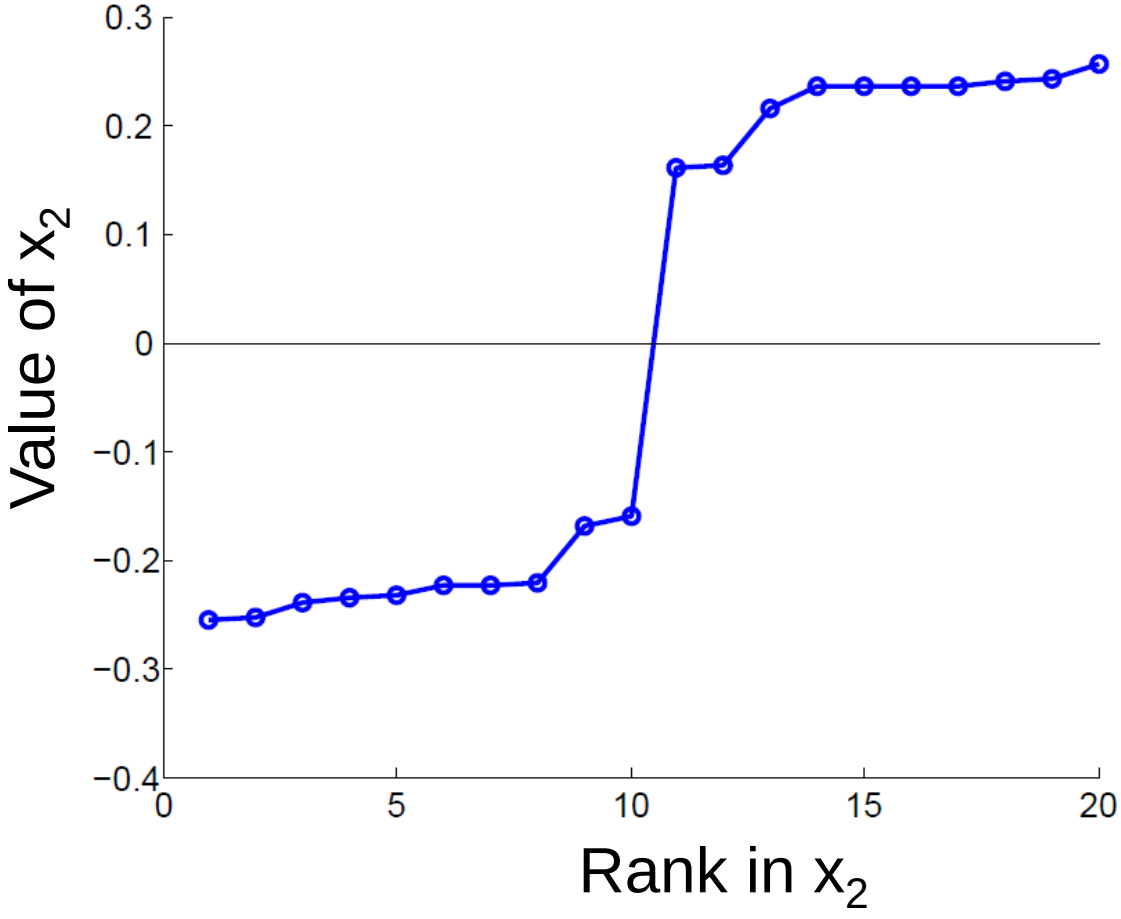
Spectral Partitioning

- **3) Grouping:**
  - Sort components of reduced 1-dimensional vector
  - Identify clusters by splitting the sorted vector in two
- **How to choose a splitting point?**
  - Naïve approaches:
    - Split at **0** or median value
  - More expensive approaches:
    - Attempt to minimize normalized cut in 1-dimension (sweep over ordering of nodes induced by the eigenvector)



Example: Spectral Partitioning





Community detection vs. Graph partitioning

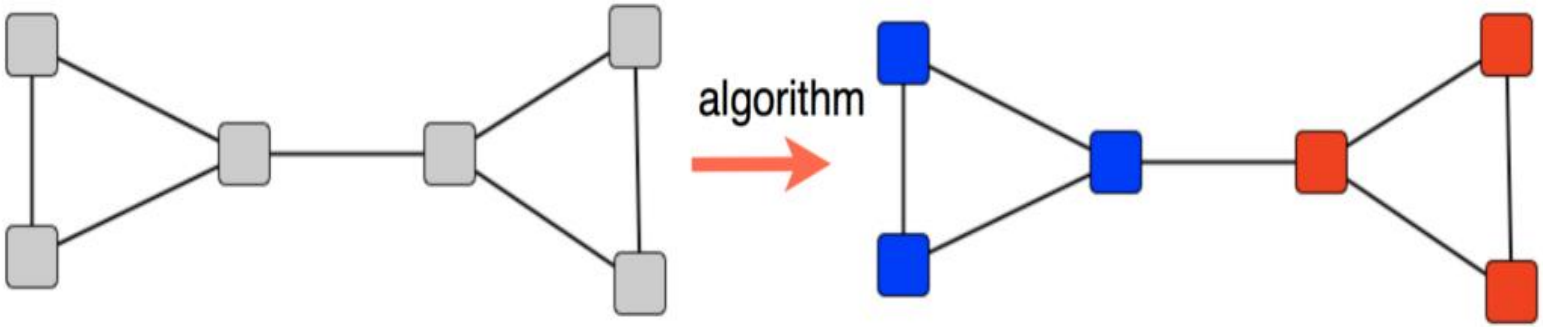
Both terms refer to the division of a network into dense groups

- **Graph partitioning:** the number and size of the groups is fixed by the user.

- **Community detection:** the number and size of the groups are unspecified (unknown), but determined by the organization of the network. Ideally, the method should be able to uncover a mixture of groups of different size in the same system. It should divide a network only when a good subdivision exists and leave it undivided otherwise

Community detection- Algorithm

Is it possible to uncover the (multi-scale) modular organization of networks in an automated fashion?
Given a graph, we look for an algorithm able to uncover its modules without specifying their number nor their size The method should be scalable to accommodate very large networks, as often observed in the real-world.



Clustering and Community Finding

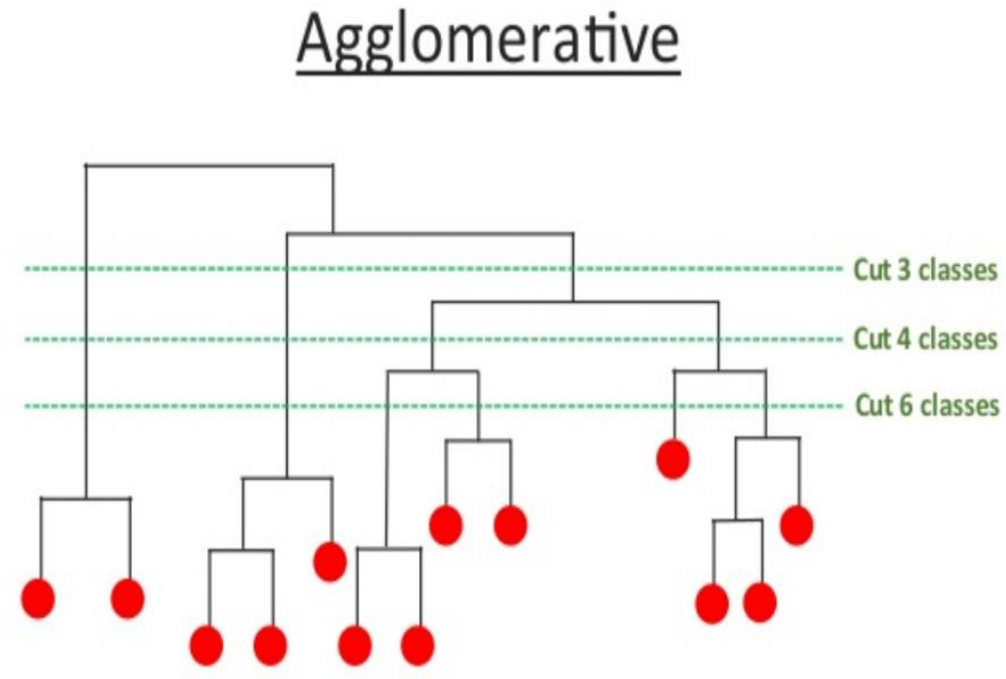
No community overlaps

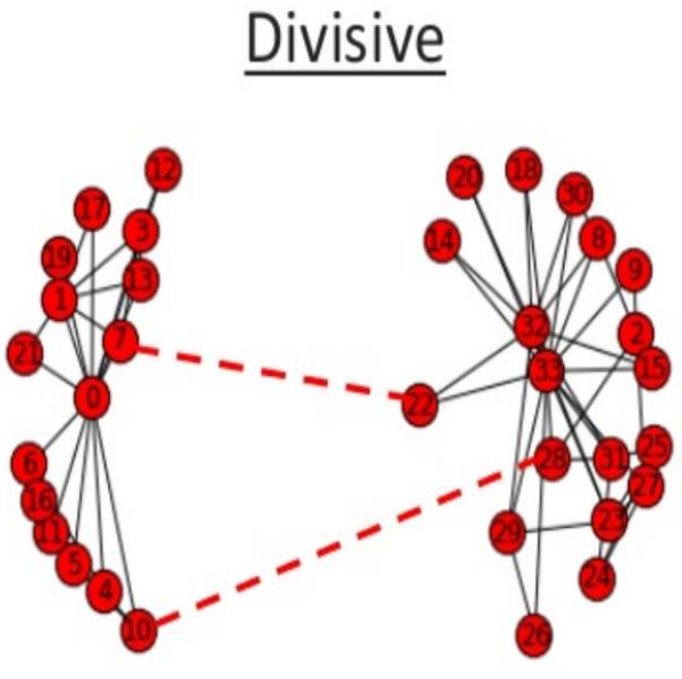
Many methods:

- Hierarchical methods:
  - Top-down/ bottom-up
- Graph partitioning methods:
  - Define “edge counting” metric – conductance, expansion, modularity, etc. – and optimize!
- Spectral methods:
  - Based on eigenvector decomposition of modified graph adjacency matrix
- Clique percolation method:
  - To extract **overlapping** communities in networks

Hierarchical Clustering

Hierarchical Methods





bottom-up

Top-down

Hierarchical Clustering

1. Build a ***similarity matrix*** for the network

2. Similarity matrix: how similar two nodes are to each other → we need to determine from the adjacency matrix

3. Hierarchical clustering iteratively identifies groups of nodes with high similarity, following one of two distinct strategies:

   - *Agglomerative algorithms* merge nodes and communities with high similarity. (A bottom-up strategy)

   - *Divisive algorithms* split communities by removing links that connect nodes with low similarity. (A top-down strategy)

4. *Hierarchical tree* or *dendrogram*: visualize the history of the merging or splitting process the algorithm follows. Horizontal cuts of this tree offer various community partitions.

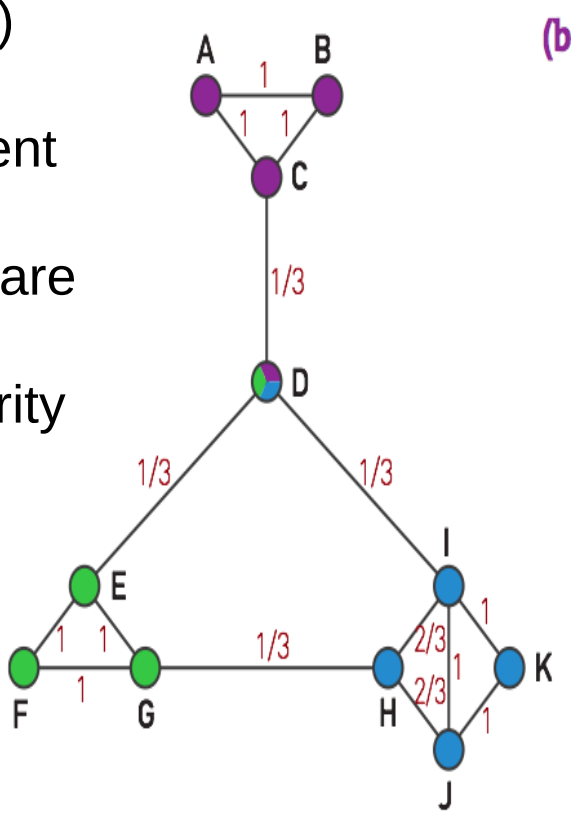
Agglomerative Algorithms

***Agglomerative algorithms* merge nodes and communities with high similarity.**

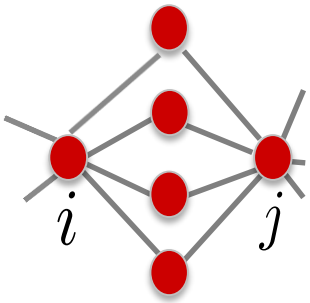
Step 1: Define the Similarity Matrix (Ravasz algorithm)

- High for node pairs that likely belong to the same community, low for those that likely belong to different communities.
- Nodes that connect directly to each other and/or share multiple neighbors are more likely to belong to the same dense local neighborhood, hence their similarity should be large.

Topological overlap matrix: $x_{ij}^{o} = \dfrac{J_N(i,j)}{\min(k_i, k_j)}$



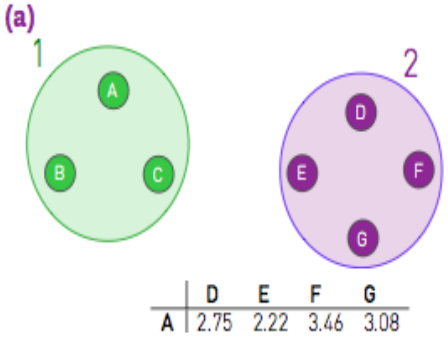
***$J_N(i,j)$*: number of common neighbors of node *i* and *j*; (+1) if there is a direct link between *i* and *j*;**



Agglomerative Algorithms

Step 2: Decide Group Similarity

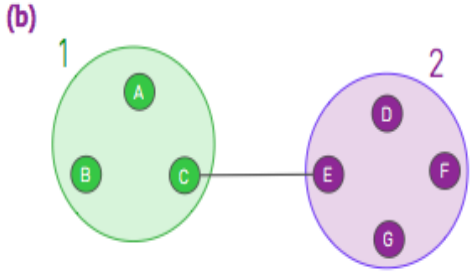
- Groups are merged based on their mutual similarity through *single*, *complete* or *average cluster linkage*



	D	E	F	G
A	2.75	2.22	3.46	3.08

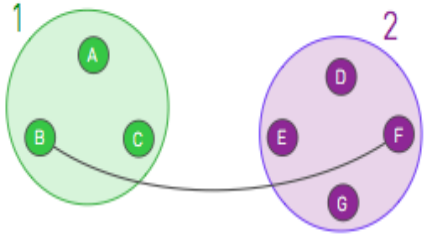
$$\frac{1}{x_{ij}} = r_{ij} =$$

	D	E	F	G
A	2.75	2.22	3.46	3.08
B	3.38	2.68	3.97	3.40
C	2.31	1.59	2.88	2.34



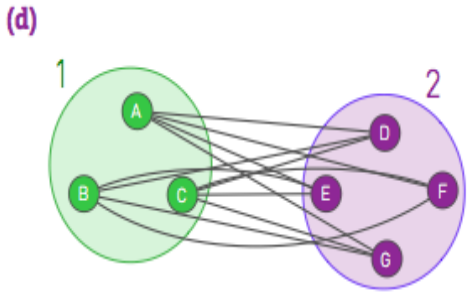
Single Linkage: $x_{12} = 1.59$

$$r_{12} = 1.59$$



Complete Linkage: $x_{12} = 3.97$

$$r_{12} = 3.97$$



Average Linkage: $x_{12} = 2.84$

$$r_{12} = 2.84$$

Agglomerative Algorithms

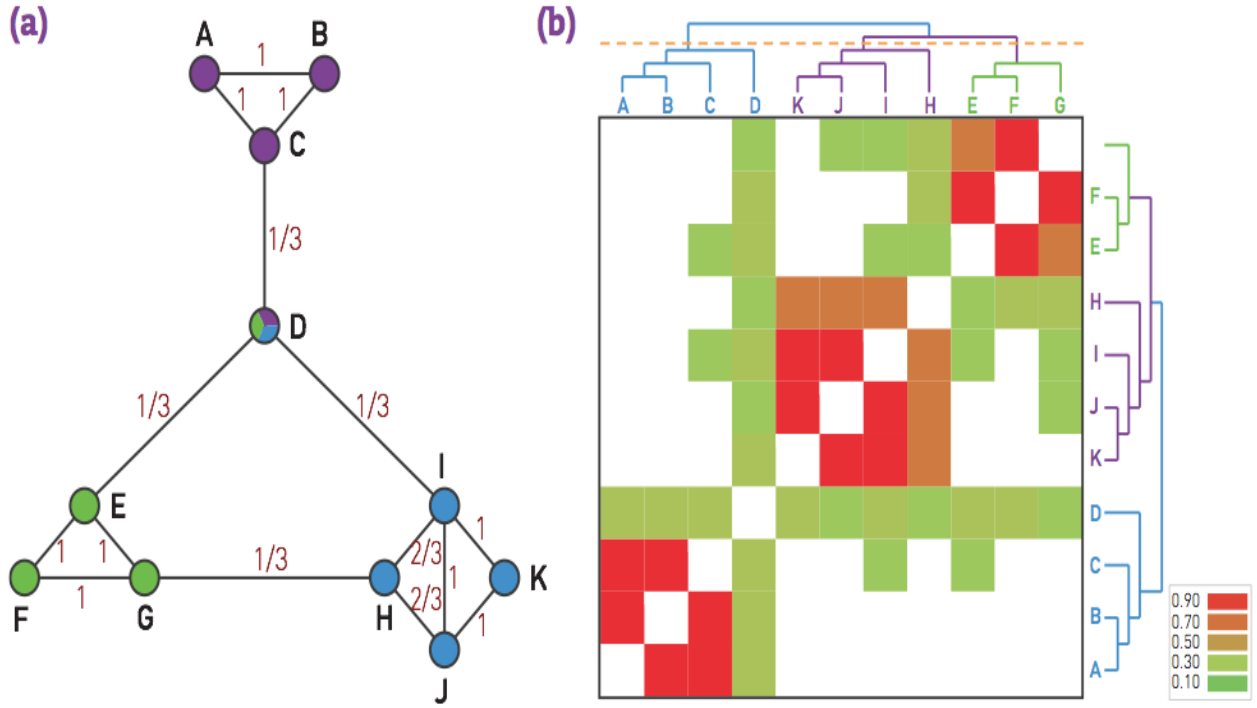
Step 3: Apply Hierarchical Clustering

- Assign each node to a community of its own and evaluate the similarity for all node pairs. The initial similarities between these “communities” are simply the node similarities.
- Find the community pair with the highest similarity and merge them to form a single community.
- Calculate the similarity between the new community and all other communities.
- Repeat from Step 2 until all nodes are merged into a single community.

Step 4: Build Dendrogram

- Describes the precise order in which the nodes are assigned to communities.

Agglomerative Algorithms



Computational complexity:

- Step 1 (calculation similarity matrix): $O\left(N^2\right)$
- Step 2-3 (group similarity): $O\left(N^2\right)$ → $O\left(N^2\right)$
- Step 4 (dendrogram): $O\left(N \log N\right)$

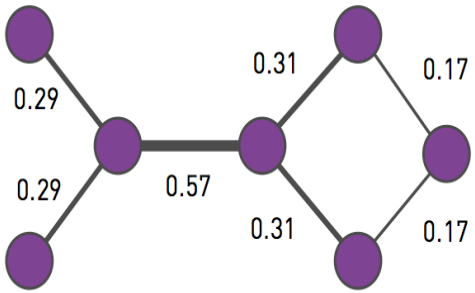
Divisive Algorithms

Divisive algorithms split communities by removing links that connect nodes with low similarity.

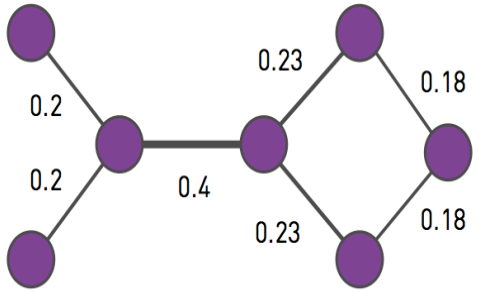
Step 1: Define a Centrality Measure (Girvan-Newman algorithm)

- ***Link betweenness*** **is the number of shortest paths between all node pairs that run along a link.**
- ***Random-walk betweenness*****. A pair of nodes m and n are chosen at random. A walker starts at m, following each adjacent link with equal probability until it reaches n. Random walk betweenness x_{ij} is the probability that the link $i \rightarrow j$ was crossed by the walker after averaging over all possible choices for the starting nodes m and n**

(a)



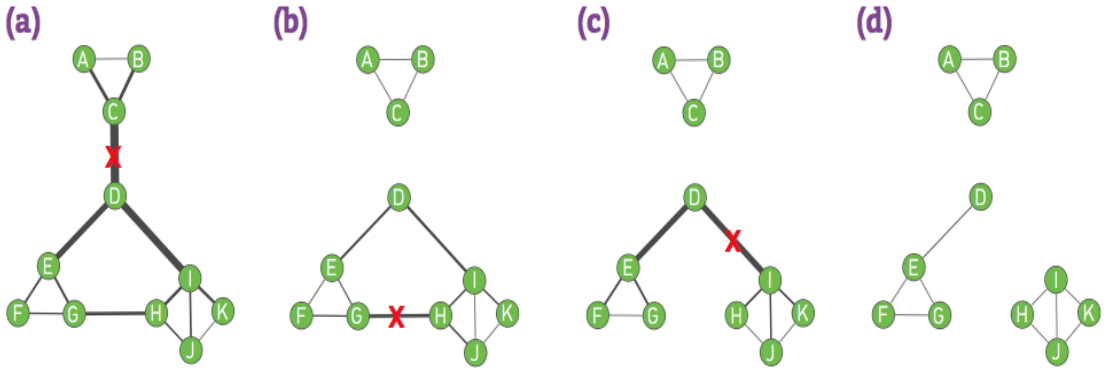
(b)



Divisive Algorithms

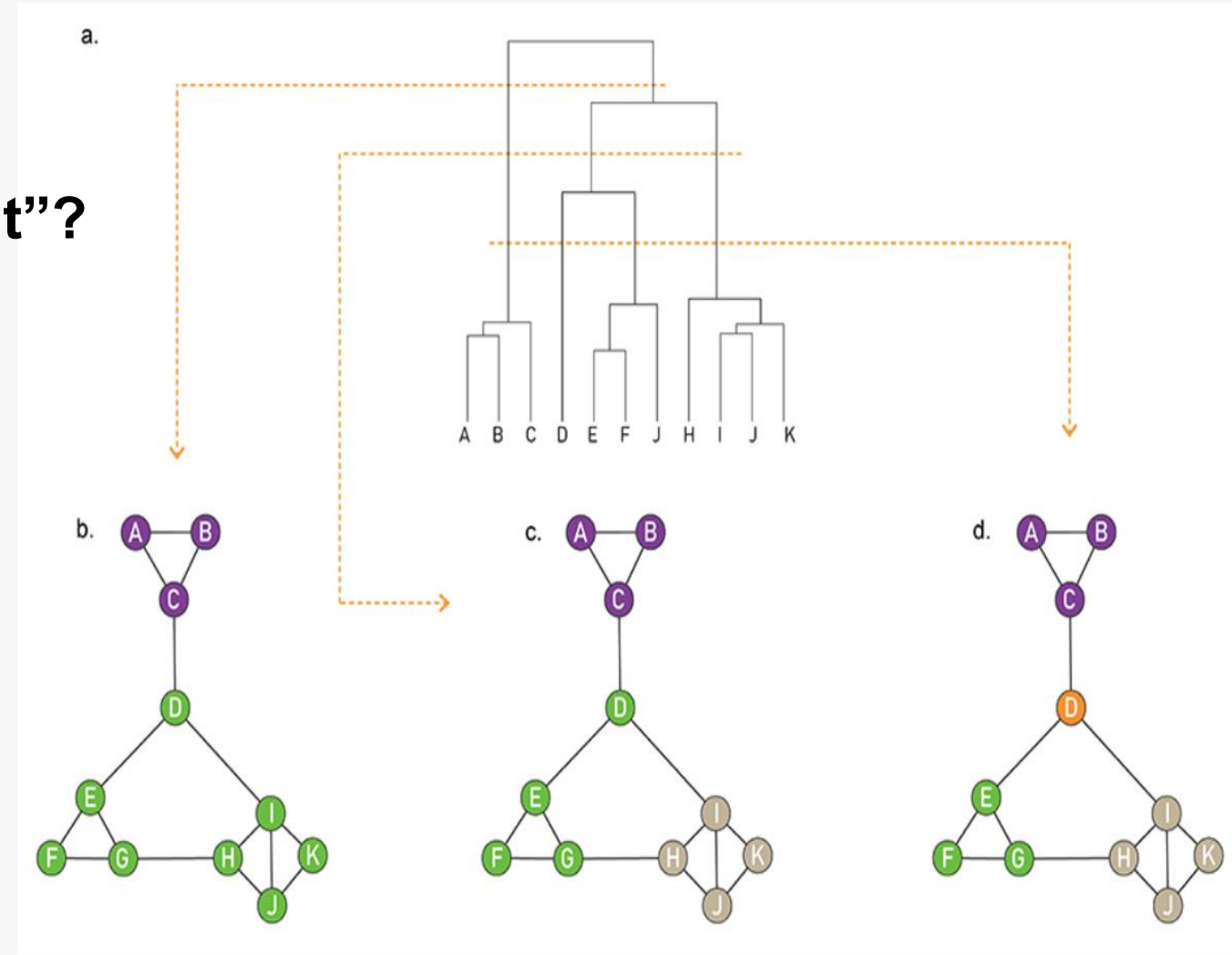
Step 2: Hierarchical Clustering

a) Compute of the centrality of each link.
b) Remove the link with the largest centrality; in case of a tie, choose one randomly.
c) Recalculate the centrality of each link for the altered network.
d) Repeat until all links are removed (yields a dendrogram).



Ambiguity in Hierarchical clustering

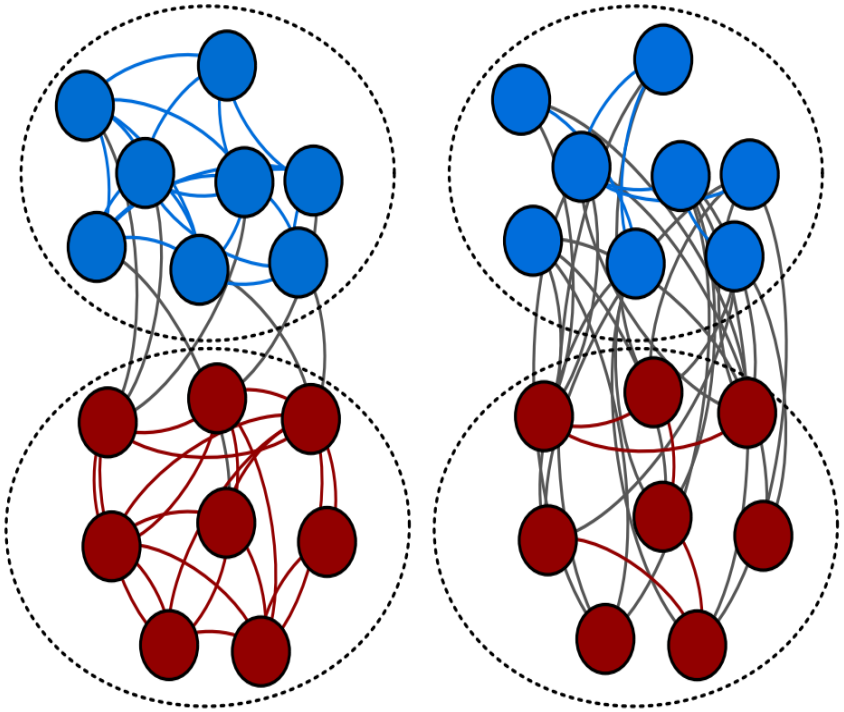
Where to “cut”?



Modularity

Modularity

Randomly wired networks are not expected to have a community structure.



Modularity measures how different the clusters of the partition are from the corresponding clusters of the ensemble of random graphs obtained by randomly joining the vertices, such to preserve their degrees, on average

the fraction of the edges that fall within communities minus the expected fraction if edges were distributed at random

Modularity

Imagine a partition in n_c communities $\{C_c,\ c = 1,\ n_c\}$

Modularity $$M(C_c) = \frac{1}{2L}\sum_{i,j=1}^{N}(A_{ij} - P_{ij})\delta(C_i\ ,\ C_j)$$

Original data

Expected connections, a model

Relative to a specific partition

→ **Random network** $P_{ij} = 2Lp_ip_j = \frac{k_ik_j}{2L}$

→ **Modularity is a <u>measure associated to a partition</u>**

Modularity

Another way of writing *M*

Total number of links in Community *C*

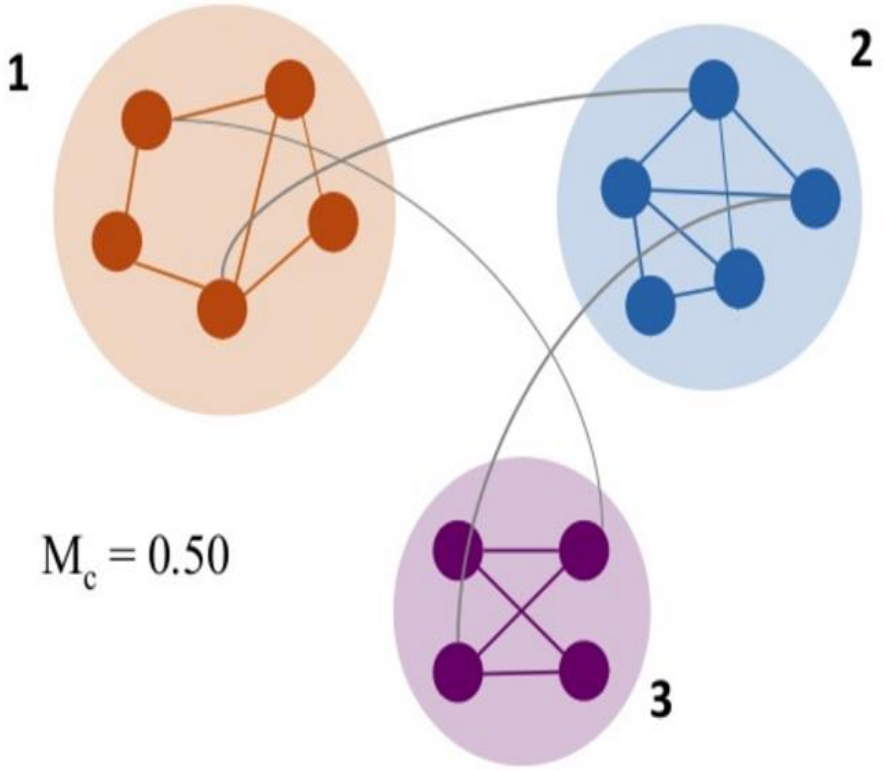
Total node degrees in Community *C*

Number of communities

$$M(C_c) = \sum_{c=1}^{n_c} \left[\frac{L_C}{L} - \left(\frac{k_c}{2L} \right)^2 \right]$$

Total number of links

Modularity - Example



Total number of links in Community C

Total node degrees in Community C

Number of communities

$$M(C_c) = \sum_{c=1}^{n_c} \left[\frac{L_C}{L} - \left(\frac{k_c}{2L} \right)^2 \right]$$

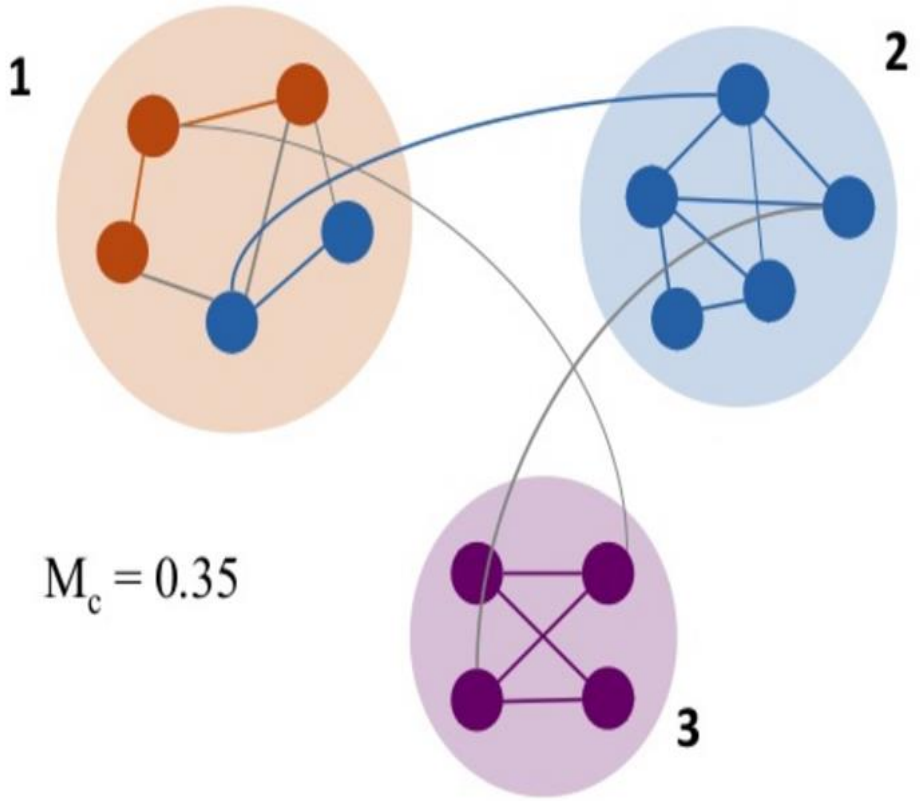
Total number of links

Community 1: $\left[\frac{6}{20} - \left(\frac{14}{40} \right)^2 \right]$

Community 2: $\left[\frac{7}{20} - \left(\frac{16}{40} \right)^2 \right]$

Community 3: $\left[\frac{4}{20} - \left(\frac{10}{40} \right)^2 \right]$

Modularity - Example

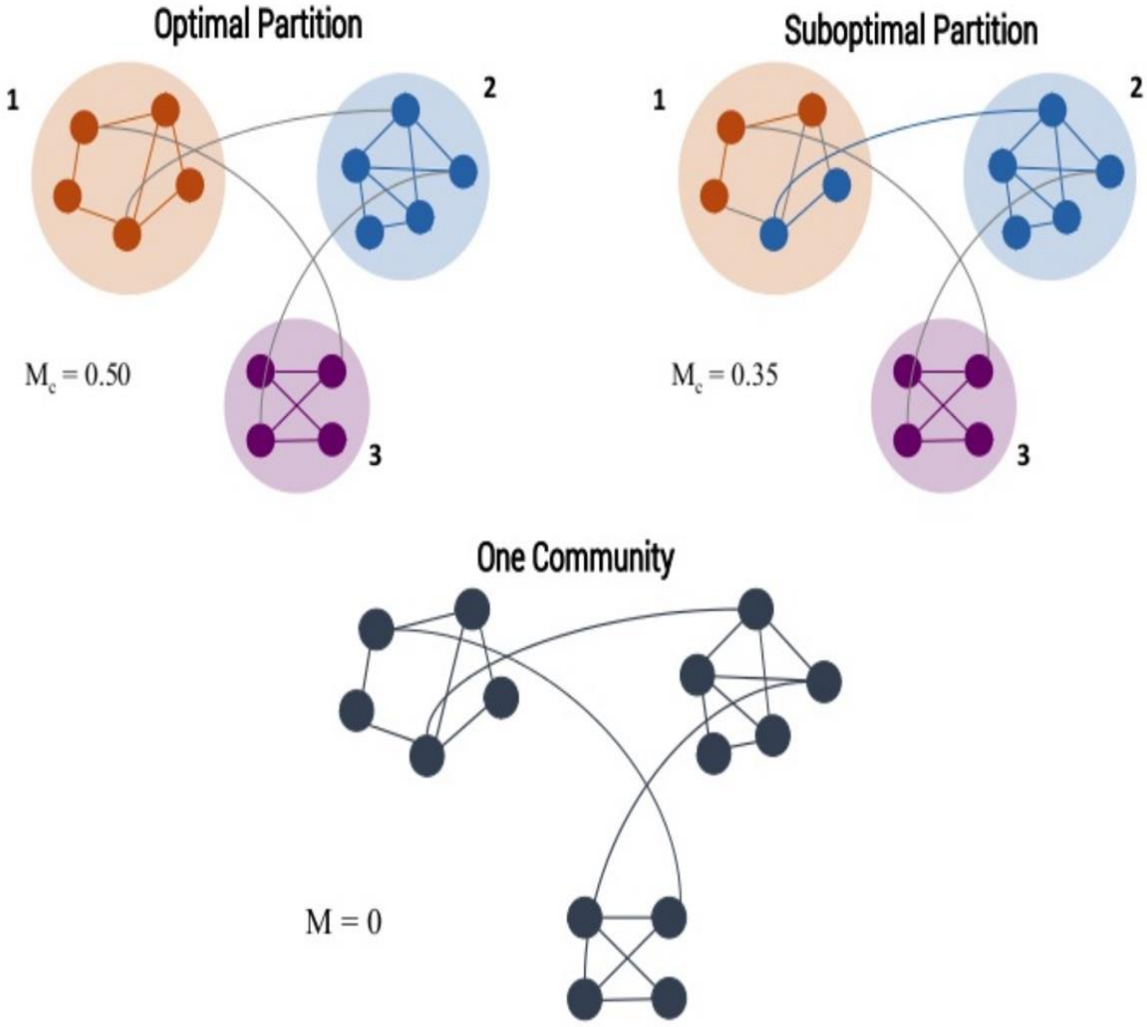


Community 1: $\left[\frac{2}{20} - \left(\frac{8}{40}\right)^2\right]$

Community 2: $\left[\frac{9}{20} - \left(\frac{22}{40}\right)^2\right]$

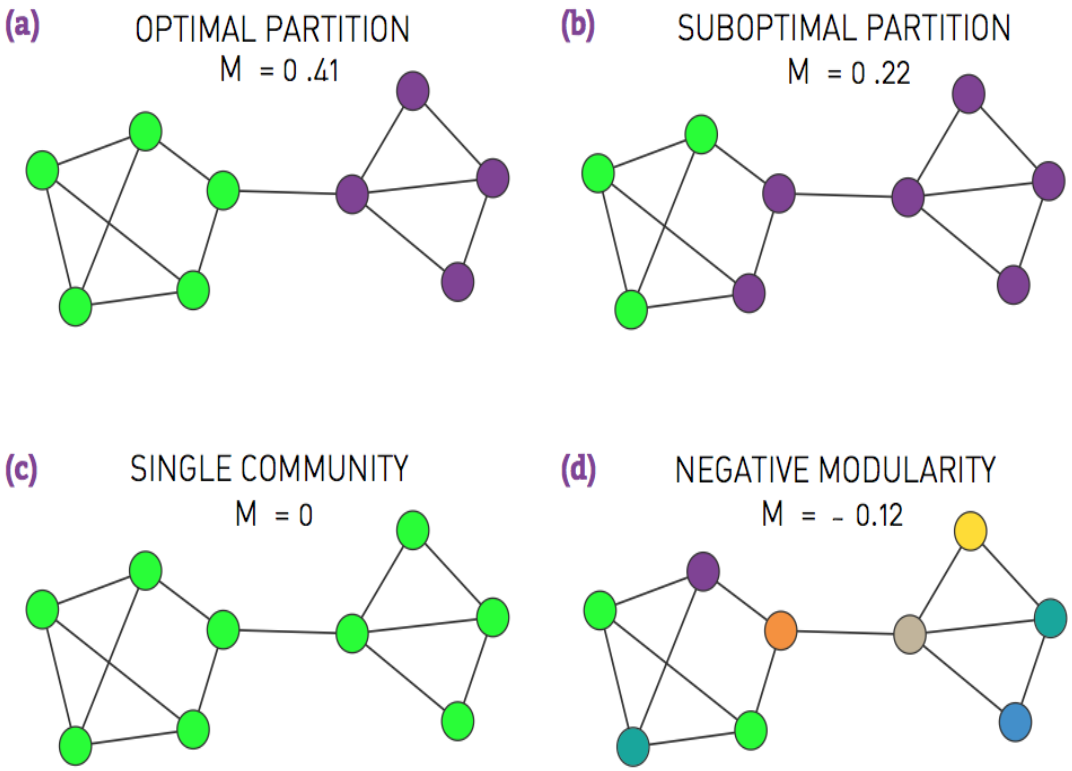
Community 3: $\left[\frac{4}{20} - \left(\frac{10}{40}\right)^2\right]$

Modularity - Example



Modularity - Example

Which partition?



- **(a)** *Optimal partition*, that maximizes the modularity.
- **(b)** *Sub-optimal* but positive modularity.
- **(c)** *Zero modularity:* Assigning all nodes to the same community, independent of the network structure.
- **(d)** *Negative Modularity*. If we assign each node to a different community.

Modularity based community identification

A ***greedy algorithm***, which iteratively joins nodes if the move increases the new partition's modularity.

Step 1. Assign each node to a community of its own. Hence we start with N communities.

Step 2. Inspect each pair of communities connected by at least one link and compute the modularity variation obtained if we merge these two communities.

Step 3. Identify the community pairs for which ΔM is the largest and merge them. Note that modularity of a particular partition is always calculated from the full topology of the network.

Step 4. Repeat step 2 until all nodes are merged into a single community.

Step 5. Record for each step and select the partition for which the modularity is maximal.

Modularity for Girvan-Newman method

- Divisive hierarchical clustering based on the notion of edge **betweenness**:

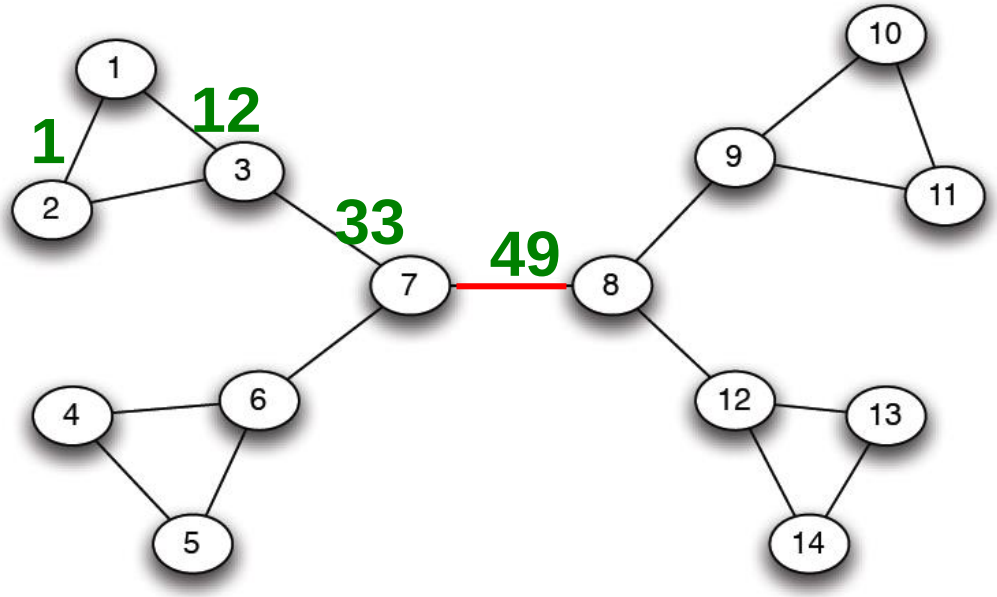
  Number of shortest paths passing through the edge

- **Girvan-Newman Algorithm:**

  - Undirected unweighted networks

  - Repeat until no edges are left:
    - Calculate betweenness of edges
    - Remove edges with highest betweenness
  - Connected components are communities
  - Gives a hierarchical decomposition of the network

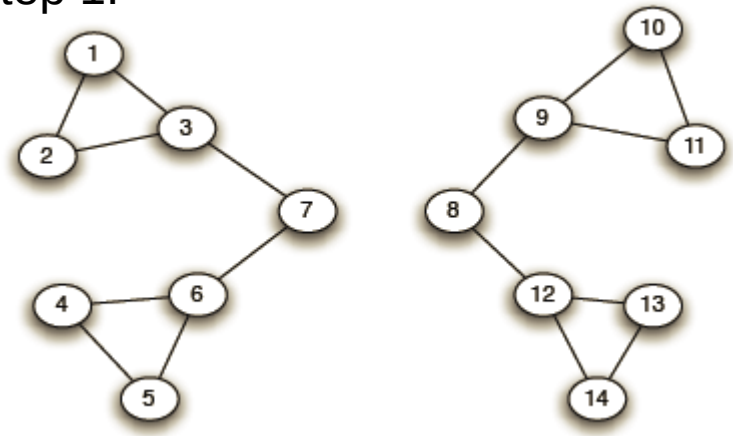
Girvan-Newman: Example



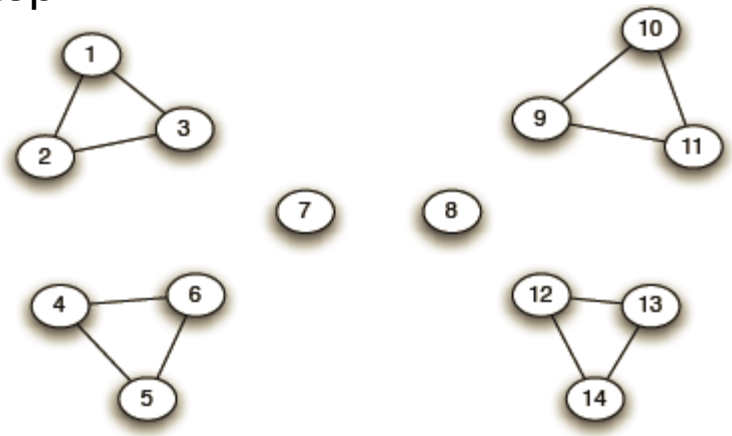
Need to re-compute betweenness at every step

Girvan-Newman: Example

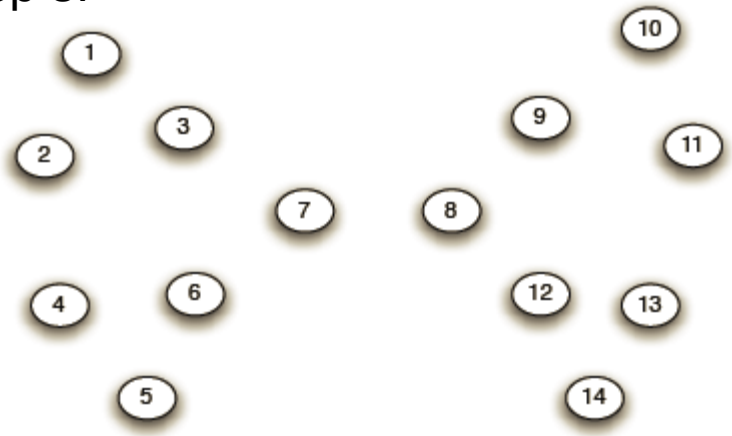
Step 1:



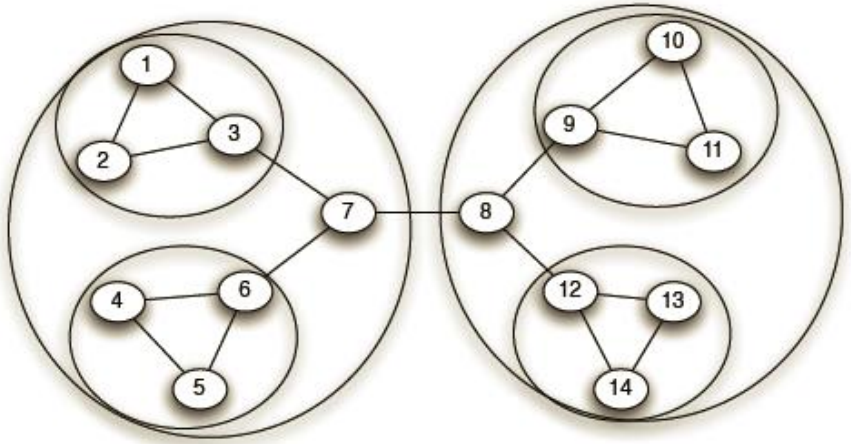
Step 2:



Step 3:

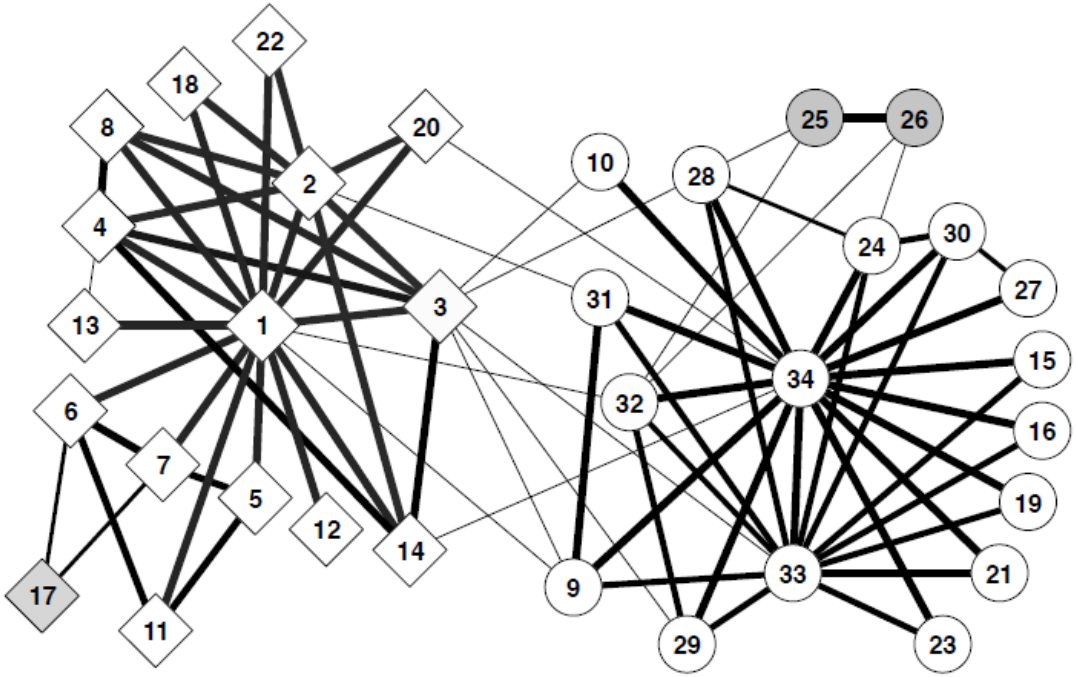


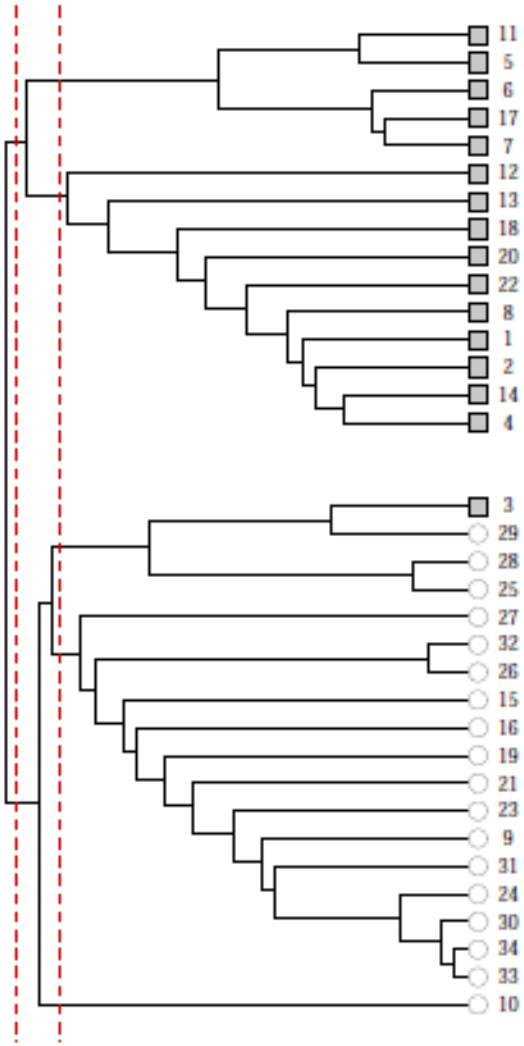
Hierarchical network decomposition:



Girvan-Newman: Results

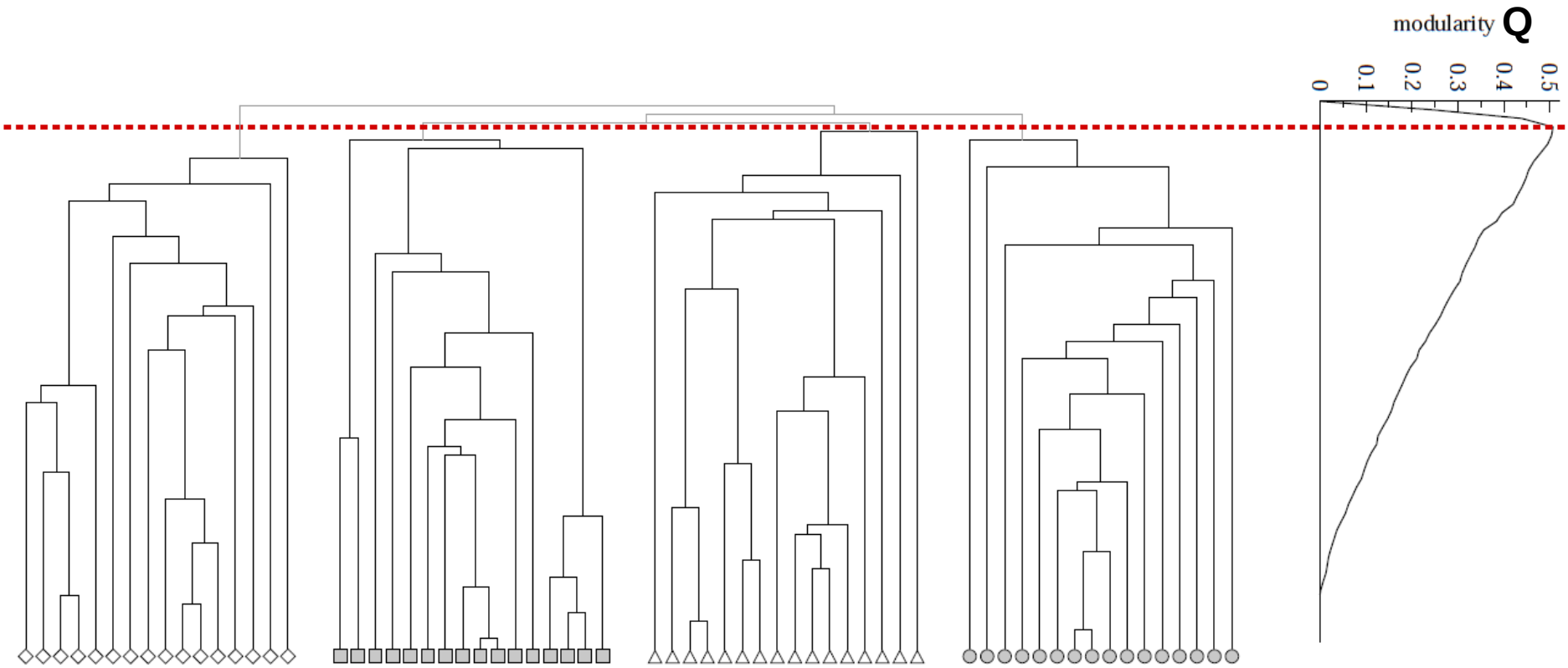
- **Zachary's Karate club:**
  hierarchical decomposition





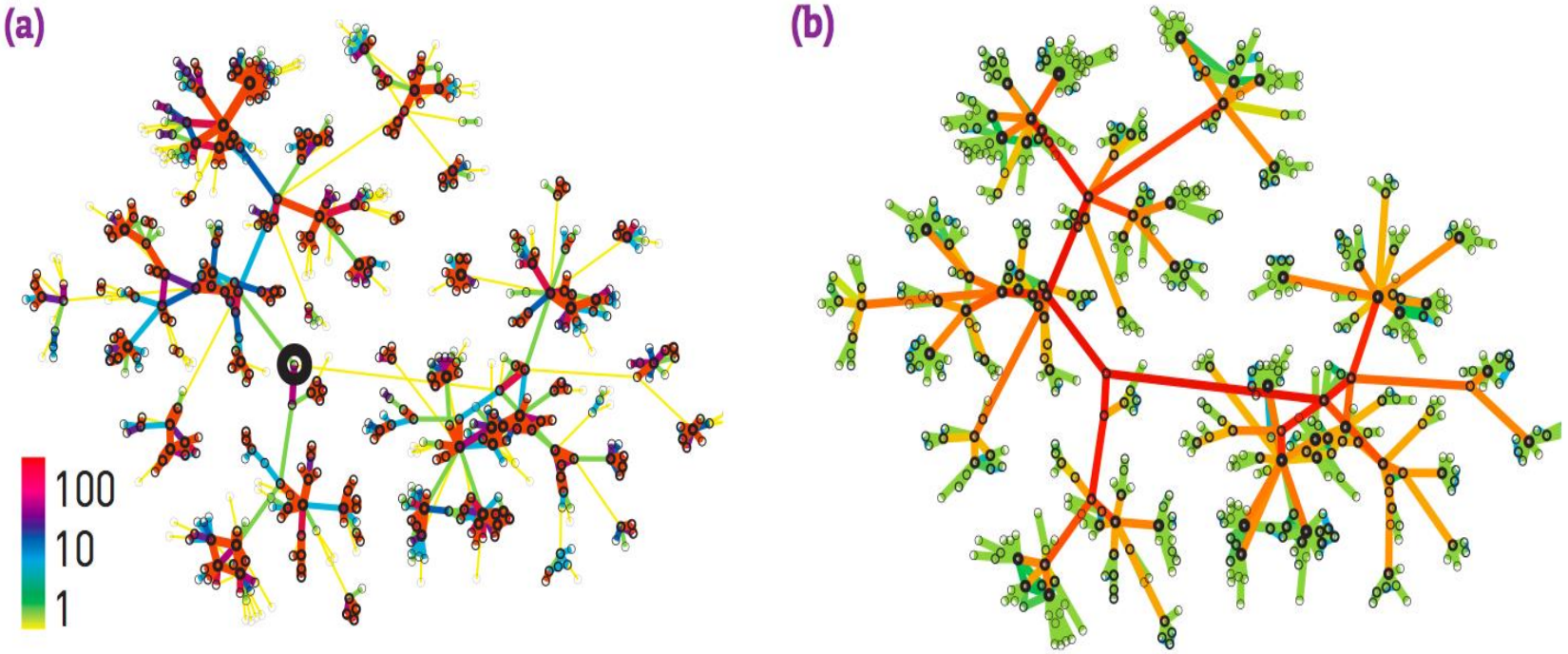
Modularity: Number of clusters

- Modularity is useful for selecting the number of clusters:



Characterizing Communities

Weights can help identifying communities



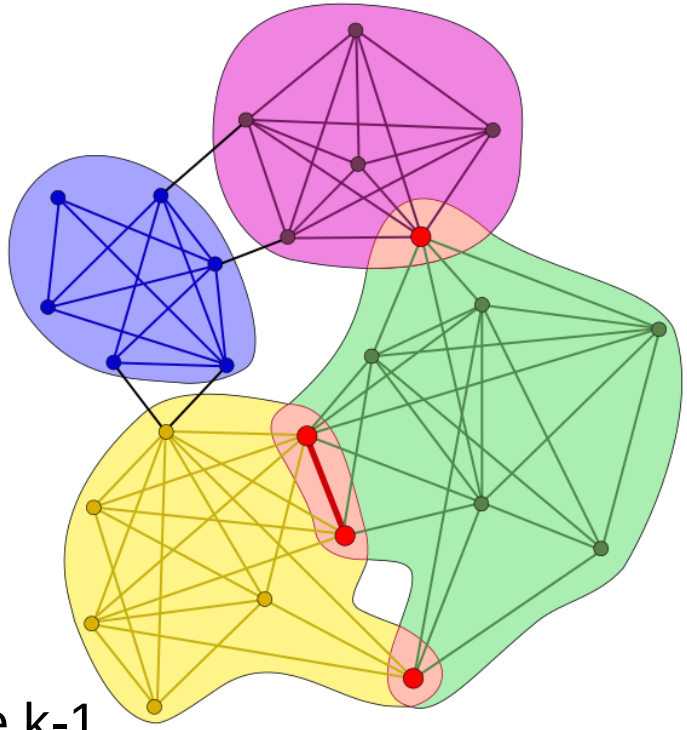
Caution!

- In social networks (a), the weak tie hypothesis says that communities are connected by links with smaller weight
- In transport systems, high betweenness links (correlates with weight, (b)) connect different communities

Using Cliques as a seed of Community

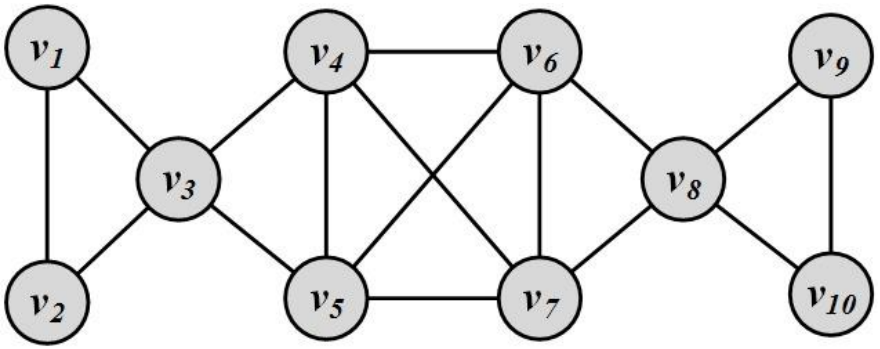
Clique Percolation Method (CPM):

- Input
  - A parameter k, and a network
- Procedure
  - Find out all cliques of size k in the given network
  - Construct a clique graph.
    - Two cliques are adjacent if they share k-1 nodes
  - These connected components in the clique graph form a community

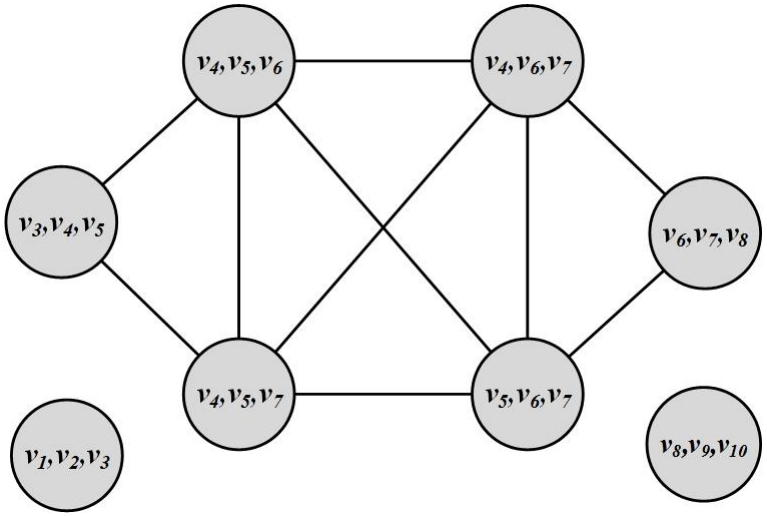


Overlapping Communities

Clique Percolation Method: Example



Cliques of size 3:
{1, 2, 3}, {3, 4,5}, {4, 5, 7},
{4,5, 6}, {4,6,7}, {5,6, 7},
{6, 7, 8}, {8,9,10}

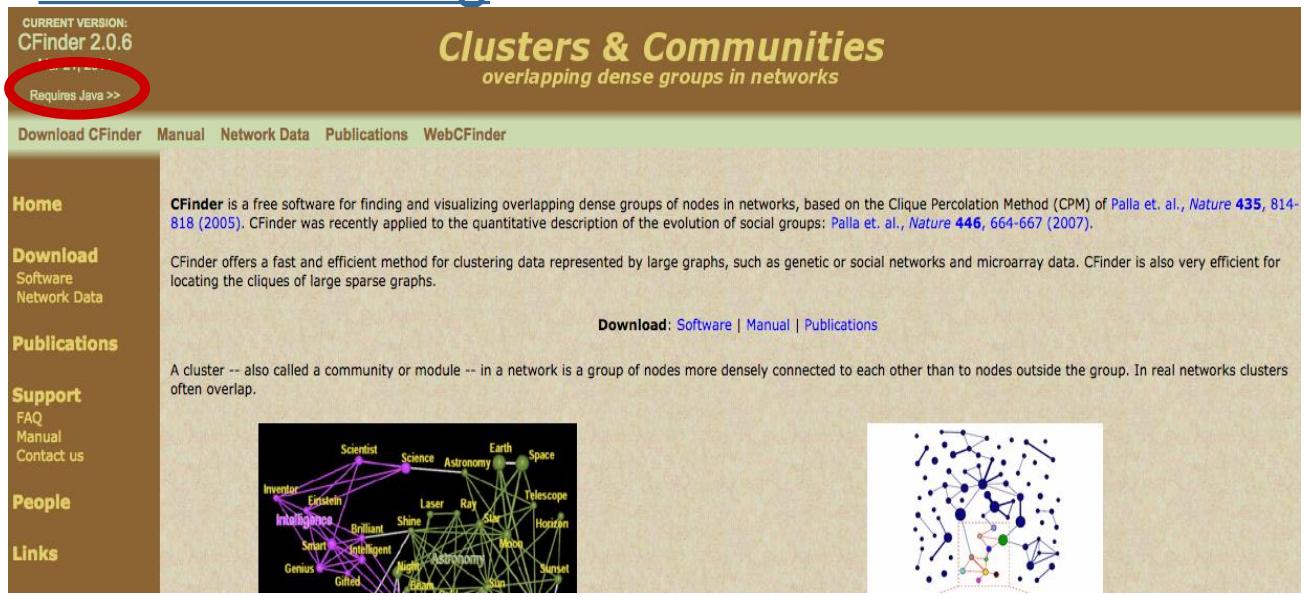


Communities:
{1, 2, 3} {8,9,10}
{3,4, 5, 6, 7, 8}

Online Resources (CFinder)

The CFinder software package that implements the Clique Percolation Method can be downloaded at

www.cfinder.org



→ **NetworkX**

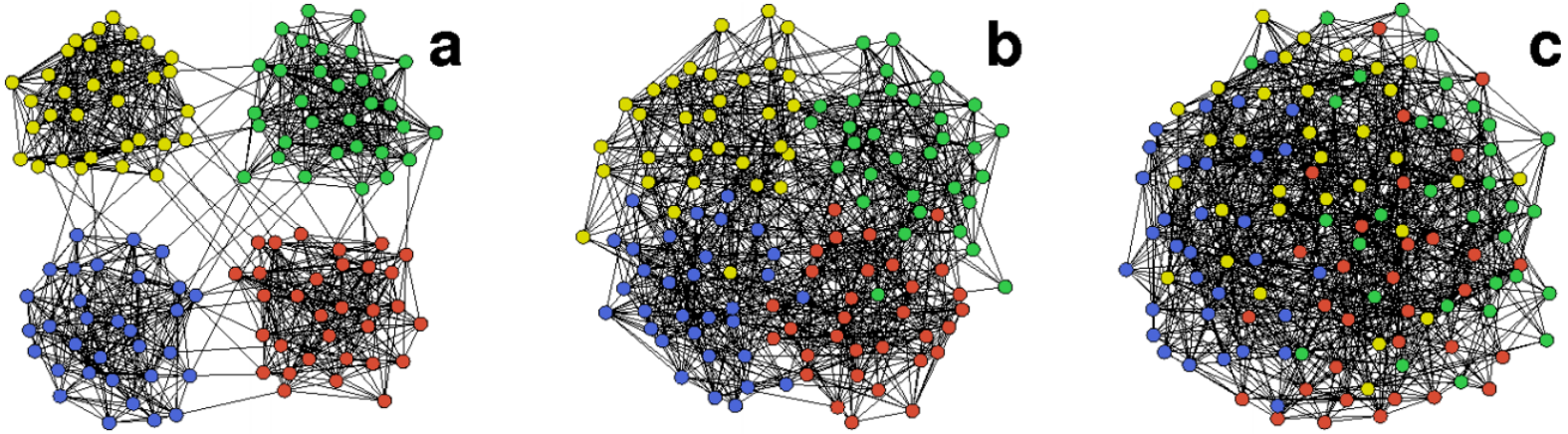
Benchmarcs and metrics

Testing Communities -Benchmarks

Newman-Girvan (NG) Benchmark

- N nodes in n_c=4 communities
- Nodes in the same community connected with probability p^{int}, otherwise p^{ext} $\mu = \frac{k^{ext}}{k^{ext} + k^{int}}$
- Control parameter

$k^{ext} \ll k^{int}$ $k^{ext} < k^{int}$ $k^{ext} = k^{int}$

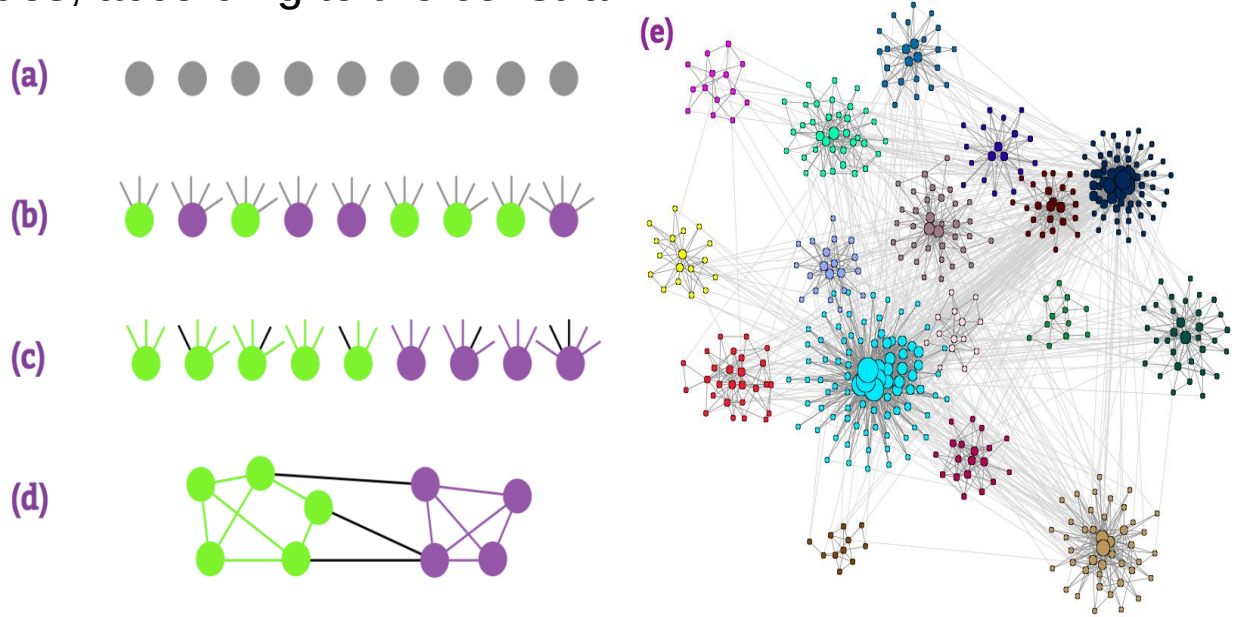


Testing Communities -Benchmarks

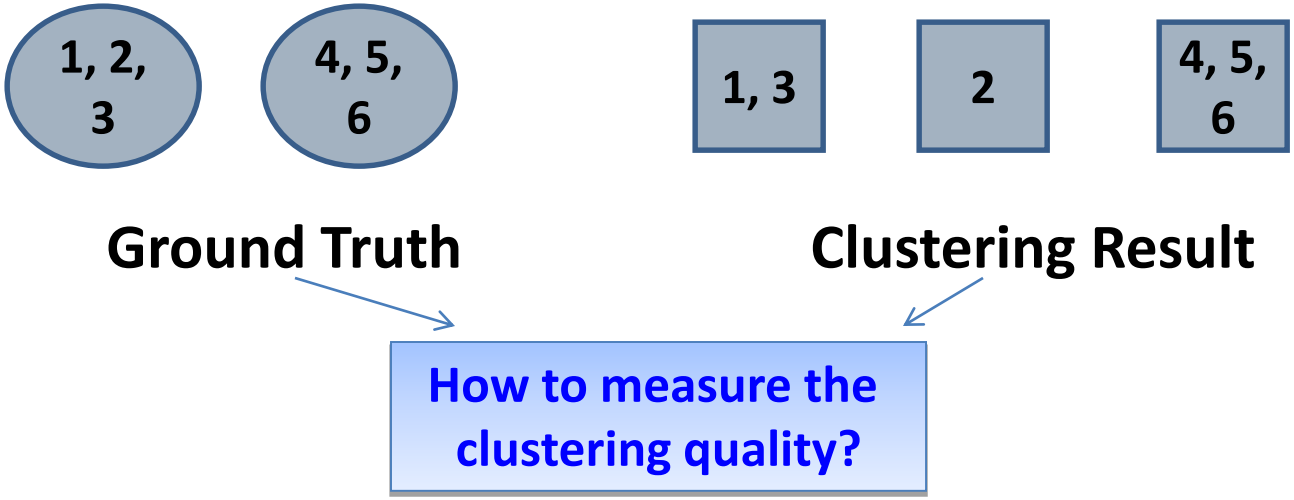
Lancichinetti-Fortunato-Radicchi (LFR) Benchmark

- N nodes in N_C communities, taken from $P_{Nc} \sim N_C^{-\zeta}$
- Each node is assigned a degree k from $P_k \sim k^{-\gamma}$
- Each node i is assigned to a community, and receives an internal degree $(1-\mu)k_i$ and an external degree μk_i.
- Connect randomly nodes, according to the constraints above. until there are no more free stubs

Vertex degree and **community size** are **power-law distributed**, to account for the heterogeneity observed in real networks with community structure



Testing Communities -Metrics



- The number of communities after grouping can be different from the ground truth
- No clear community correspondence between clustering result and the ground truth
- **Normalized Mutual Information (NMI)** can be used

Normalized Mutual Information

- Entropy: the information contained in a distribution

$$H(X) = -\sum_{x \in X} p(x) \log p(x)$$

- Mutual Information: the shared information between two distributions

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p_1(x) p_2(y)} \right)$$

- Normalized Mutual Information (between 0 and 1)

$$NMI(X;Y) = \frac{I(X;Y)}{\sqrt{H(X)H(Y)}}$$ **JMLR03, Strehl** or $$NMI(X;Y) = \frac{2I(X;Y)}{H(X)+H(Y)}$$ **KDD04, Dhilon**

- Consider a partition as a distribution (probability of one node falling into one community), we can compute the matching between the clustering result and the ground truth

Normalized Mutual Information

$$H(X) = \sum_{x \in X} p(x) \log p(x)$$

$$H(\pi^a) = \sum_{h}^{k^{(a)}} \frac{n_h^a}{n} \log(\frac{n_h^a}{n})$$

$$H(\pi^b) = \sum_{\ell}^{k^{(b)}} \frac{n_\ell^b}{n} \log(\frac{n_\ell^b}{n})$$

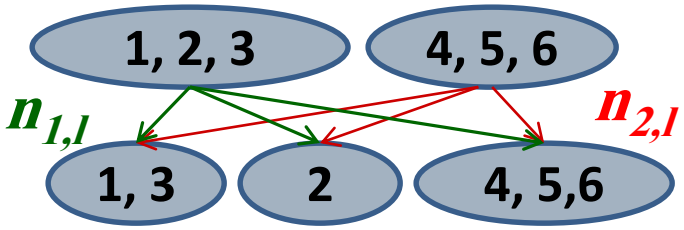
$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p_1(x) p_2(y)} \right) \Rightarrow I(\pi^a, \pi^b) = \sum_{h} \sum_{\ell} \frac{n_{h,\ell}}{n} \log \left(\frac{\frac{n_{h,\ell}}{n}}{\frac{n_h^a}{n} \frac{n_\ell^b}{n}} \right)$$

$$NMI(X;Y) = \frac{I(X;Y)}{\sqrt{H(X)H(Y)}}$$

$$NMI(\pi^a, \pi^b) = \frac{\sum_{h=1}^{k^{(a)}} \sum_{\ell=1}^{k^{(b)}} n_{h,\ell} \log \left(\frac{n \cdot n_{h,l}}{n_h^{(a)} \cdot n_\ell^{(b)}} \right)}{\sqrt{\left(\sum_{h=1}^{k^{(a)}} n_h^{(a)} \log \frac{n_h^a}{n} \right) \left(\sum_{\ell=1}^{k^{(b)}} n_\ell^{(b)} \log \frac{n_\ell^b}{n} \right)}}$$

NMI-Example

- Partition a: [1, 1, 1, 2, 2, 2]
- Partition b: [1, 2, 1, 3, 3, 3]



$n = 6$

$k^{(a)} = 2$

$k^{(b)} = 3$

	n_h^a
h=1	3
h=2	3

	n_l^b
l=1	2
l=2	1
l=3	3

$n_{h,l}$	l=1	l=2	l=3
h=1	2	1	0
h=2	0	0	3

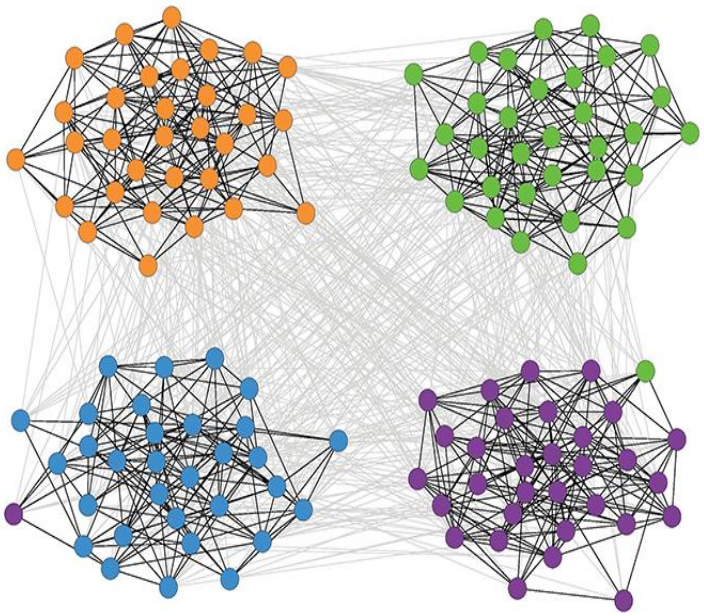
contingency table or confusion matrix

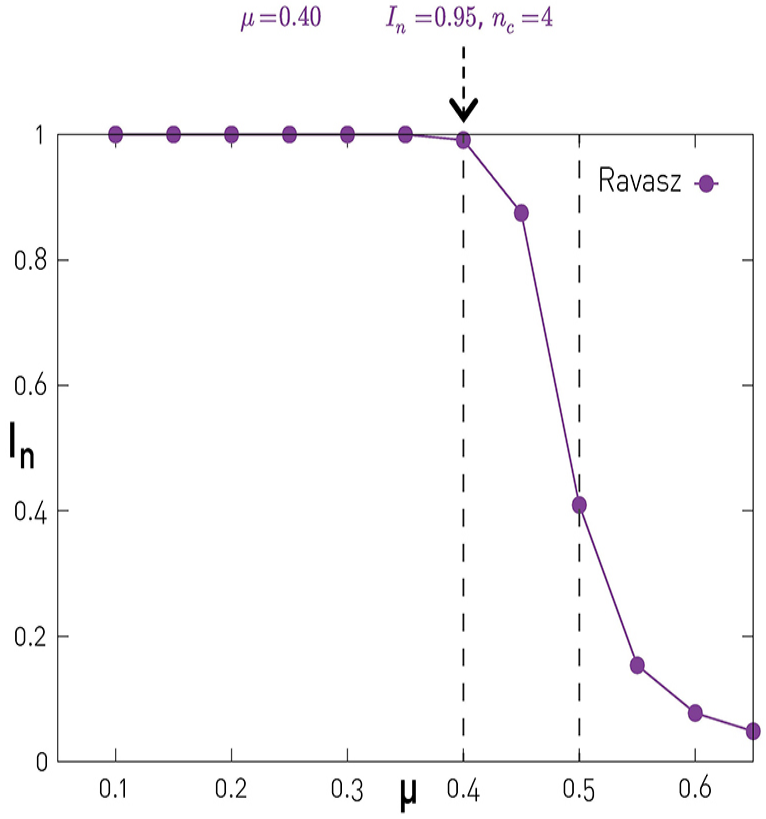
$$NMI(\pi^a, \pi^b) = \frac{\sum_{h=1}^{k^{(a)}} \sum_{\ell=1}^{k^{(b)}} n_{h,\ell} \log\left(\frac{n \cdot n_{h,l}}{n_h^{(a)} \cdot n_\ell^{(b)}}\right)}{\sqrt{\left(\sum_{h=1}^{k^{(a)}} n_h^{(a)} \log \frac{n_h^a}{n}\right)\left(\sum_{\ell=1}^{k^{(b)}} n_\ell^{(b)} \log \frac{n_\ell^b}{n}\right)}}$$

=0.8278

Normalized Mutual Information

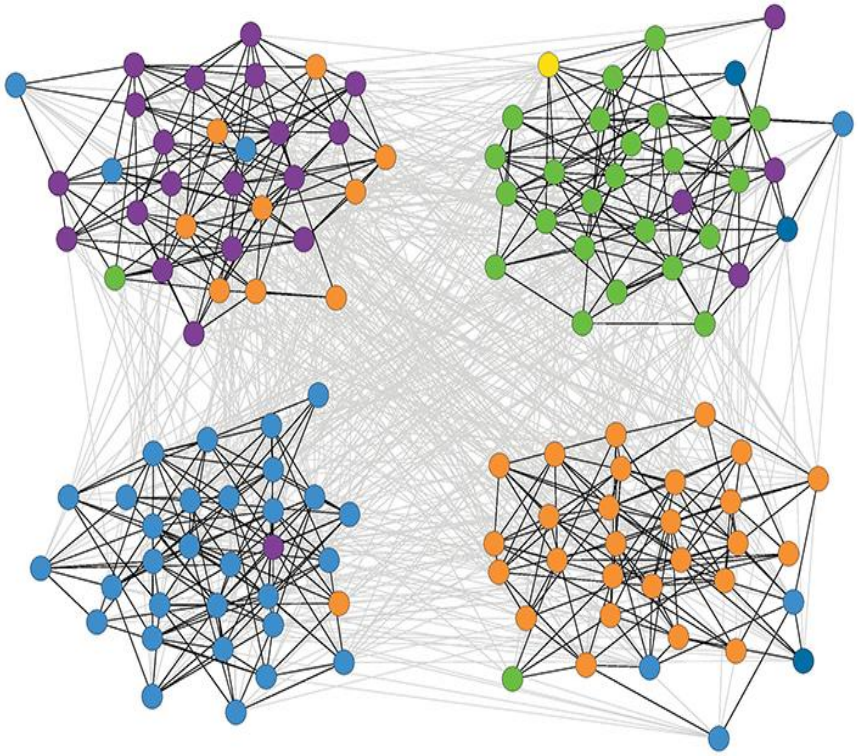
Testing Accuracy with the NG Benchmark

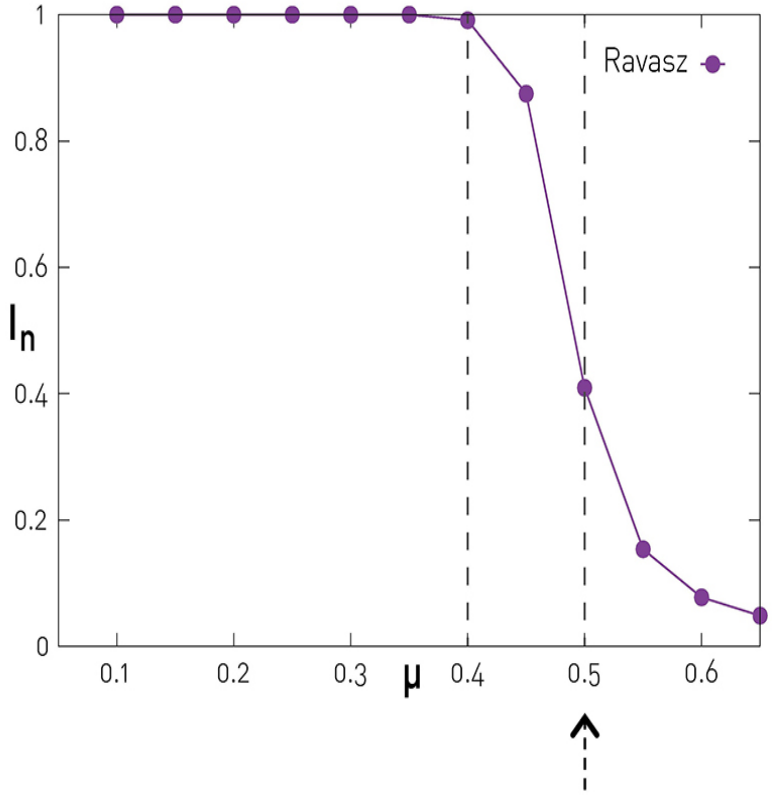




Normalized Mutual Information

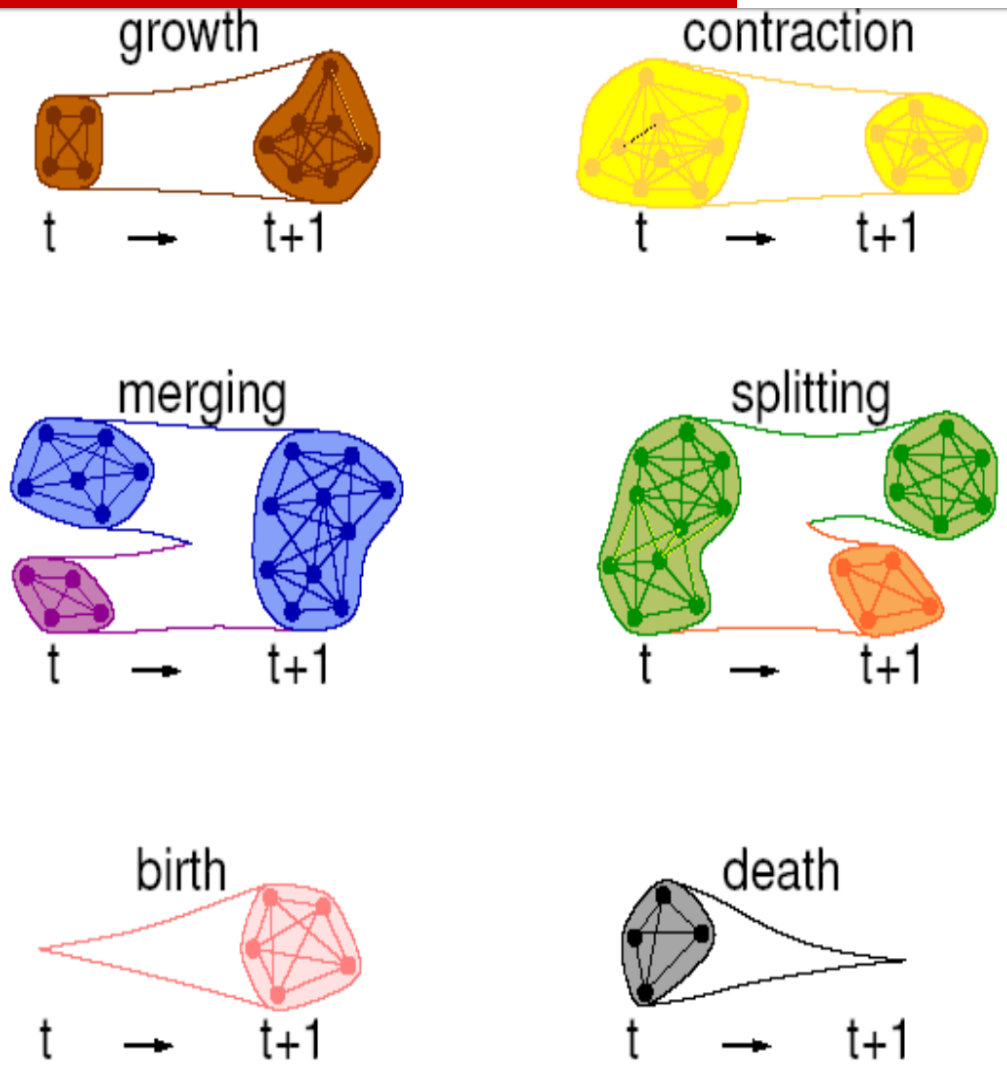
Testing Accuracy with the NG Benchmark





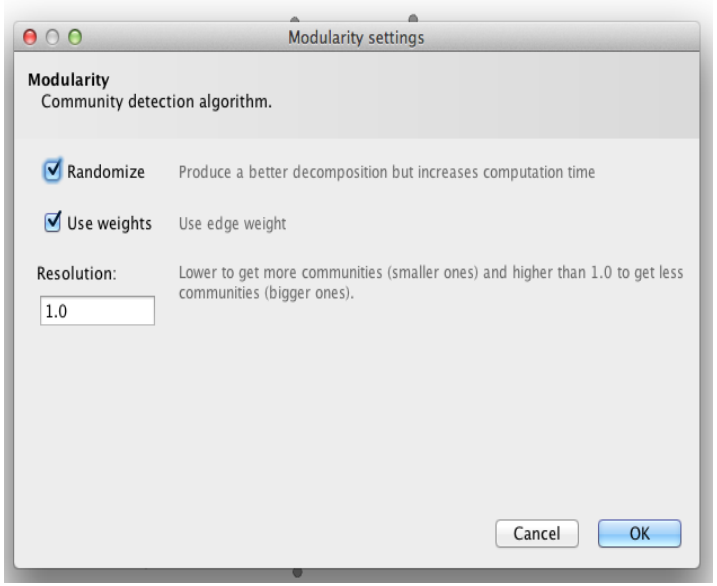
$\mu = 0.50$ $I_n = 0.56,\ n_c = 6$

Community dynamics(evolution)



Online Resources (Modularity)

Gephi



R assigns self-loops to nodes to increase or decrease the aversion of nodes to form communities

NetworkX

`community.best_partition(graph, partition=None)` ¶

Compute the partition of the graph nodes which maximises the modularity (or try..) using the Louvain heuristices

This is the partition of highest modularity, i.e. the highest partition of the dendogram generated by the Louvain algorithm.

Finds the partition that maximizes modularity (considers weights and direction)

`community.modularity(partition, graph)`

Compute the modularity of a partition of a graph

Calculates the modularity of the partition you provide


Questions