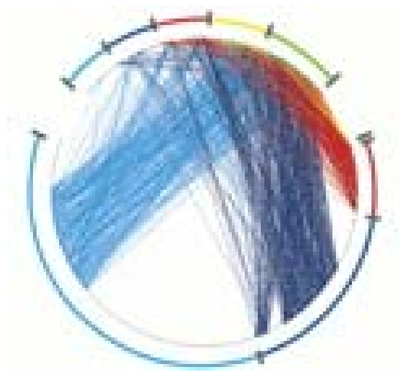
Lectures 7&8: Hierarchy & Modularity Measures



Networks: flow of information

- How information flows through the network?
- How different nodes can play structurally distinct roles in this process?
- How different links (short range vs. long range) play different roles in diffusion?

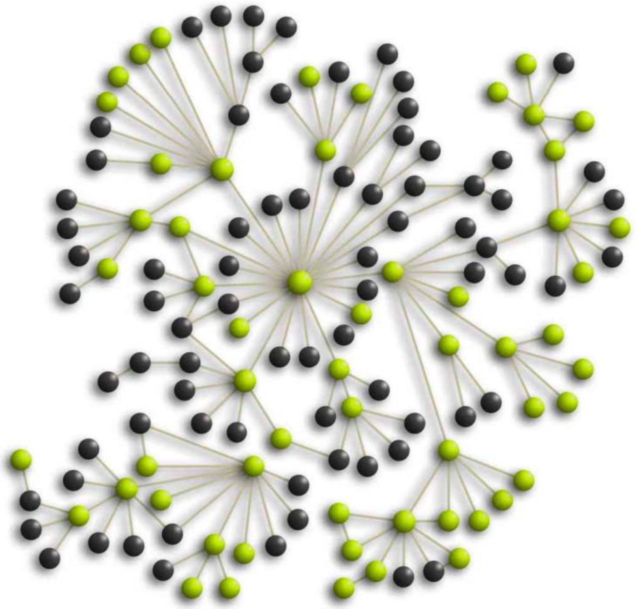
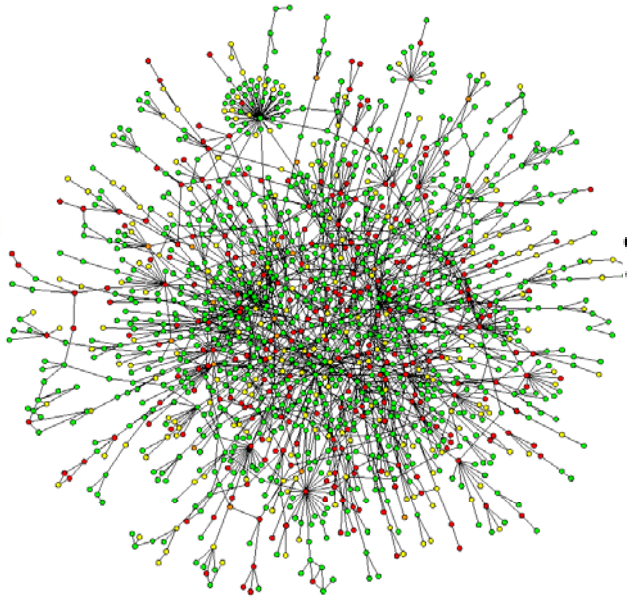
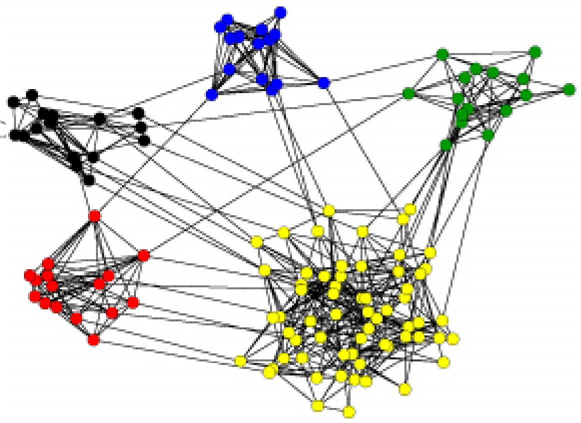
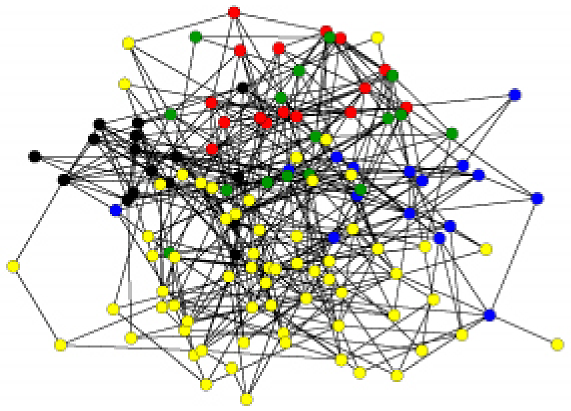
Communities

- Network of tightly connected nodes → communities
- They are important in the functionality of the network
- How to automatically find such densely connected groups of nodes
- Ideally such automatically detected groups might correspond to real groups

Communities

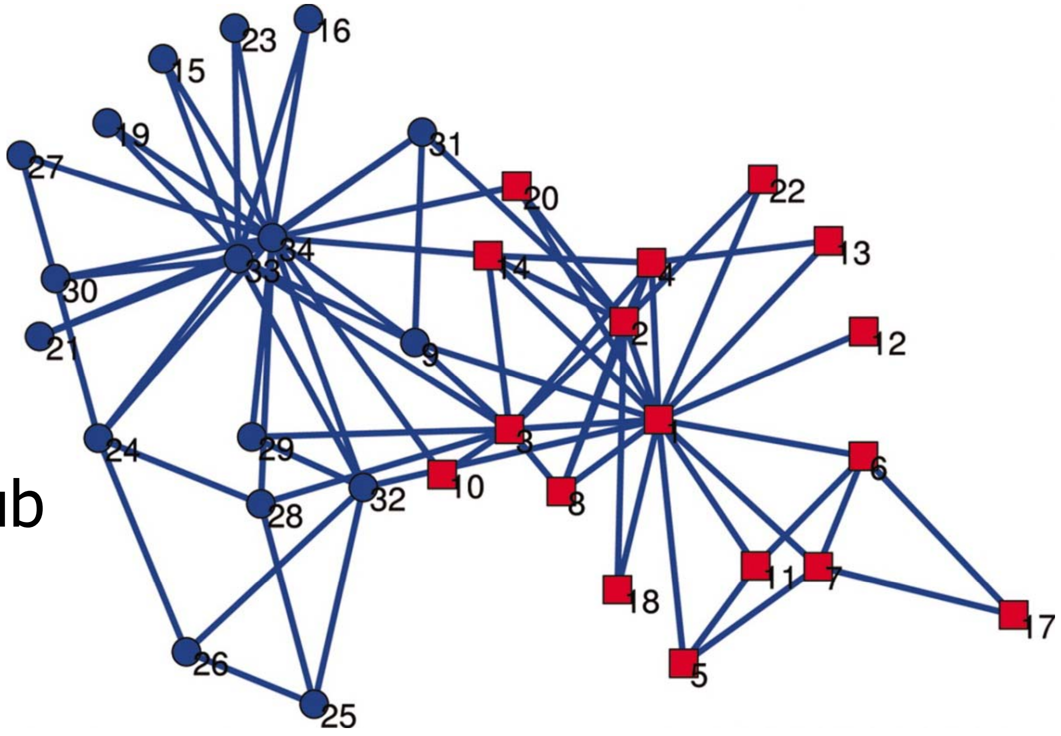
- The ability to detect community structure in a network could clearly have important benefits.
- Communities in a social network might represent real social groupings, perhaps by interest or background.
- Communities in a citation network might represent related papers on a single topic.
- Communities in a metabolic network might represent cycles and other functional groupings.
- Communities on the web might represent pages on related topics.
- Being able to identify these communities could help us to understand and exploit these networks more effectively.

Communities

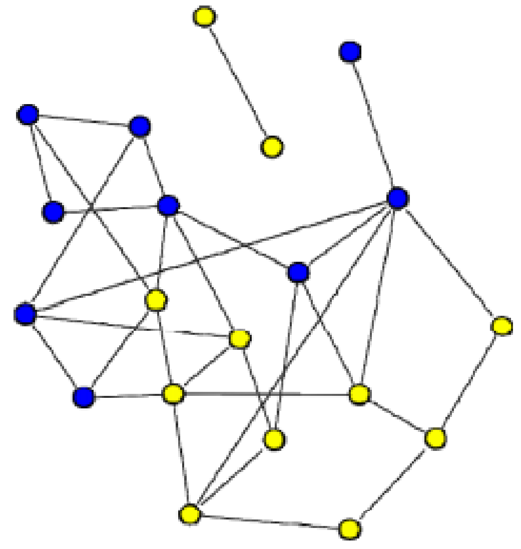
Communities in real examples

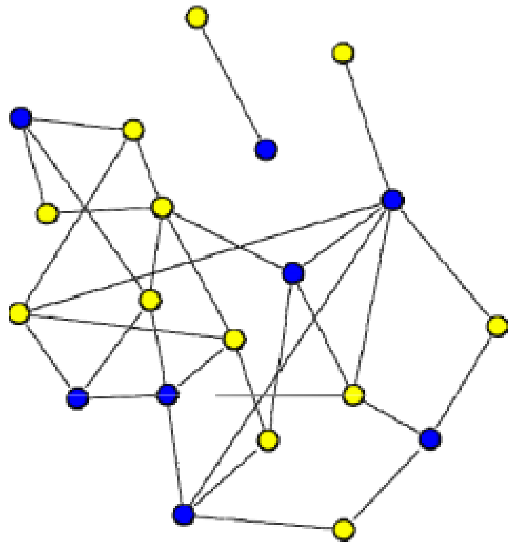
- Zachary's Karate club network
  - Observe social ties between the participants of the club
- (Newman, PRE 2002)



Group formation in networks

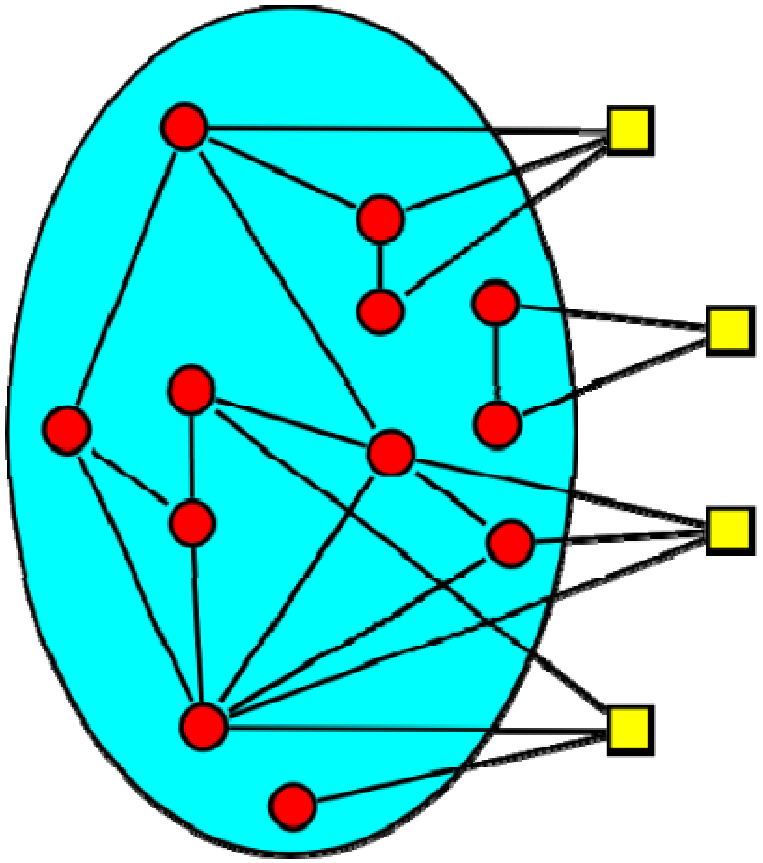
- In a social network nodes explicitly declare group membership
  - e.g. facebook groups
- We can think of groups as node colors
- It gives insight into social dynamics:
  - Recruiting friends?
  - Doesn’t recruit them?
  - What factors influences a person to join a group?



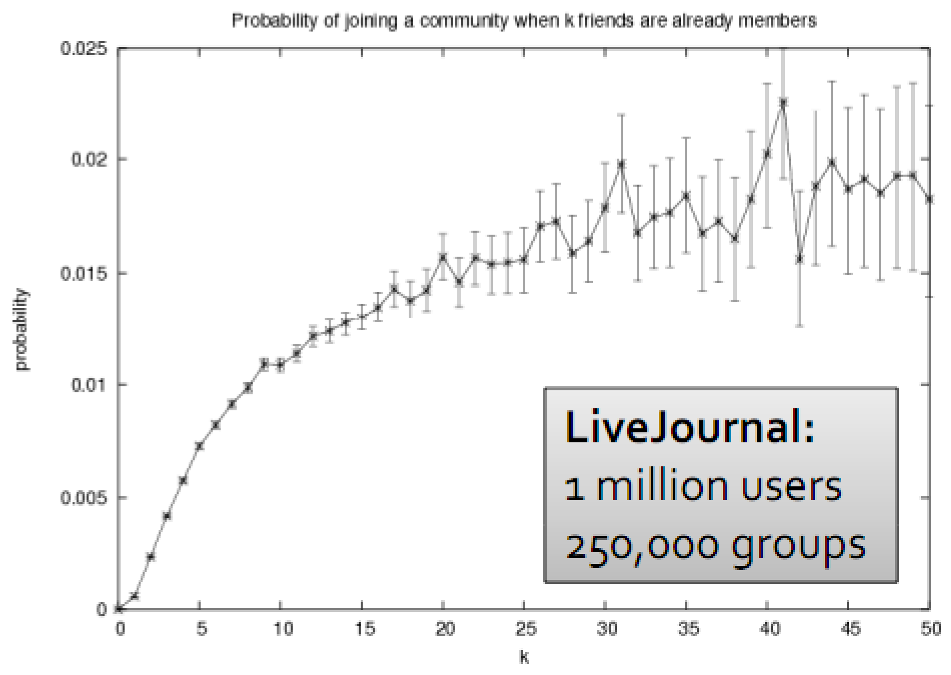


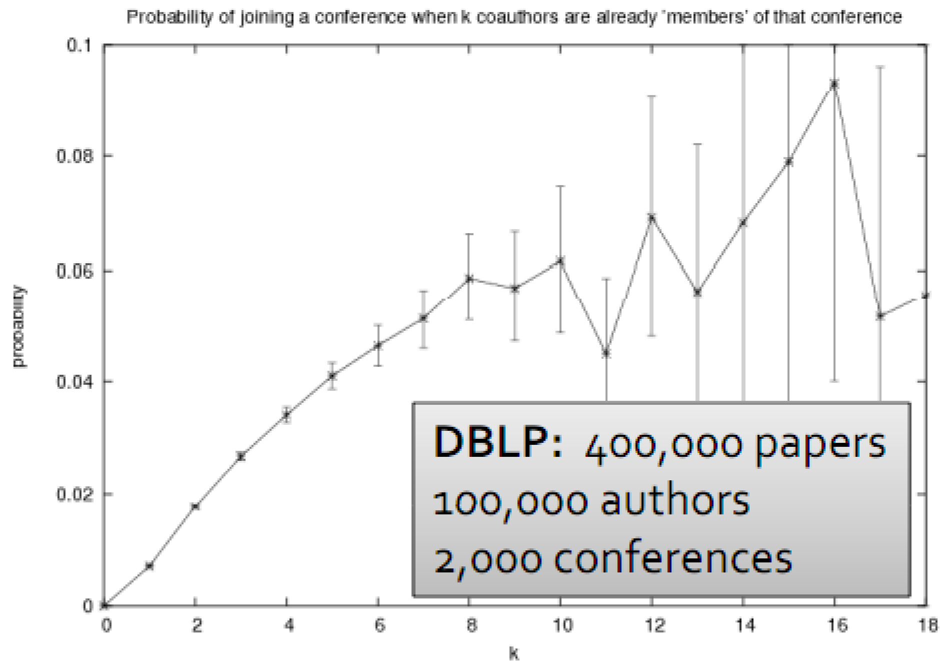
Group formation in networks

- We might think of spreading the group membership in the network
  - Node in red represent existing group members
  - Nodes in yellow may join the group
- Question:
  - How does the probability of joining a group depend on the number of friends already in the group?



Group formation in networks



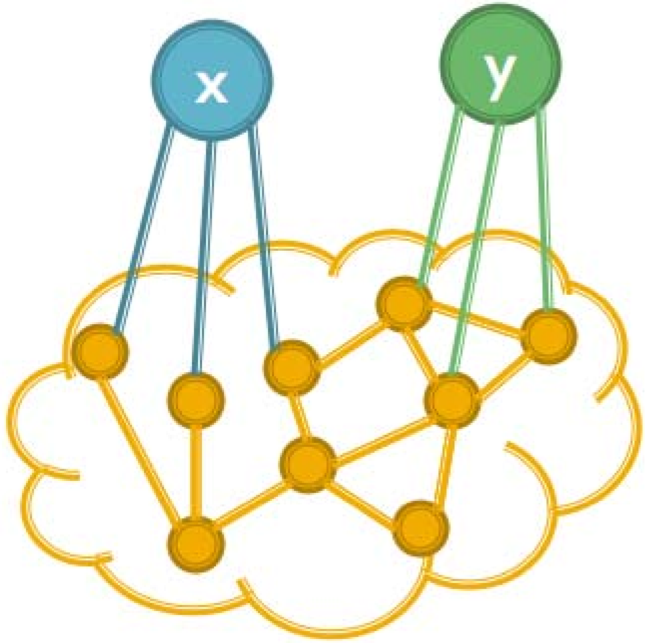


- (Backstrom et al. KDD 2006)
- Diminishing returns:
  - Probability of joining increases with the number of friends in the group
  - But increases get smaller and smaller

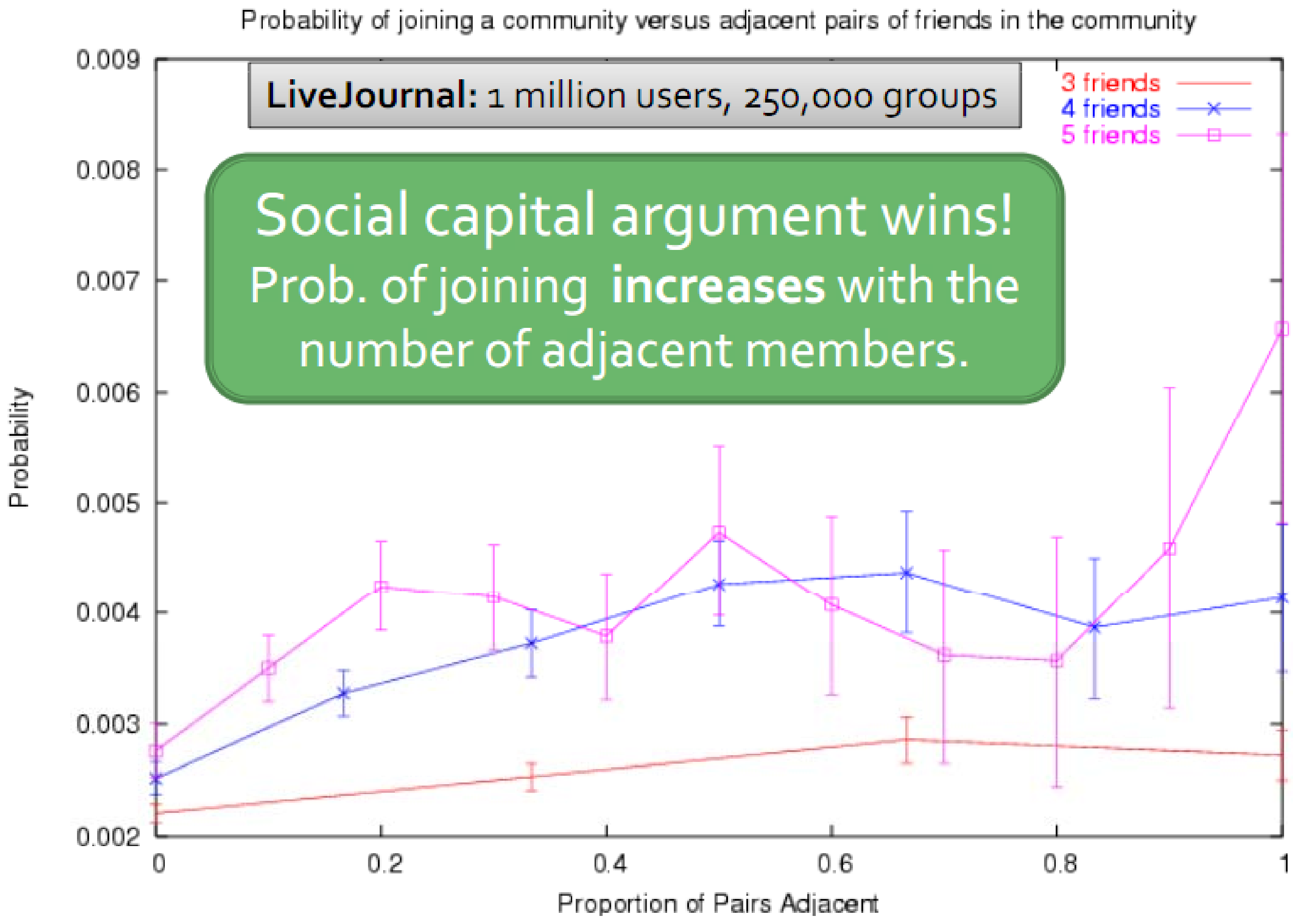
Group formation in networks

- Connectedness of friends:
  - x and y have three friends in the group
  - x's friends re independent
  - y's friends are all connected
- Who is more likely to join?

- Competing sociological theories are:
- Information argument (Granovetter '73):
  - Unconnected friends give independent support
- Social capital argument (Coleman '88):
  - Safety/trust advantage in having friends who know each other



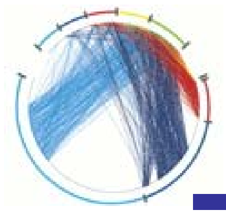
Group formation in networks



LiveJournal: 1 million users, 250,000 groups

Social capital argument wins!
Prob. of joining **increases** with the number of adjacent members.

(Backstrom et al. KDD 2006)

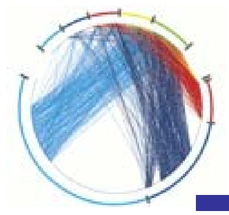


Group formation in networks

- So, this means that a person is more likely to join a group if
  - she has more friends who are already in the group
  - friends have more connections between themselves
- So, groups form clusters of tightly connected nodes, or communities
- the question is how to do clustering (community identification) in networks

Clustering

- Given a set of objects V, and a notion of similarity (or distance) between them, partition the objects into disjoint sets $S_1, S_2, \ldots, S_k$, such that objects within each set are similar, while objects across different sets are dissimilar
- Graph clustering:
  - Input: a graph $G=(V,E)$
    - edge (u,v) denotes similarity between u and v
    - weighted graphs: weight of edge captures the degree of similarity
  - Clustering: Partition the nodes in the graph such that nodes within clusters are well interconnected (high edge weights), and nodes across clusters are sparsely interconnected (low edge weights)
  - The number of edges between clusters is called cut-size
  - most graph partitioning problems are NP hard

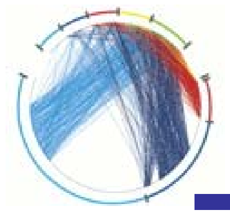


How good a cluster is?

- Random networks are supposed without modular structure
- A random network with the same degree distribution
- We may compare the modular structure of the network with that of a random network (Newman and Girvan 2002)

$$Q = \frac{1}{2m}\sum_{ij}\left(A_{ij} - P_{ij}\right)\delta\left(C_i, C_j\right) = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta\left(C_i, C_j\right)$$

  - m: number of edges
  - A: adjacency matrix
  - P: expected number of edges between the nodes in the random network
  - The δ-function yields one if i and j are in the same community ($C_i = C_j$), zero otherwise
  - k_i: degree of node i



How good a cluster is?

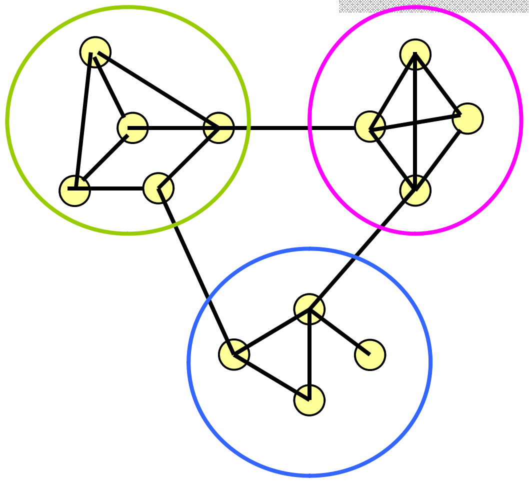
- Since the only contributions to the sum come from vertex pairs belonging to the same cluster, we can group these contributions together and rewrite the sum over the vertex pairs as a sum over the clusters

Structural definition of Community

- Groups of vertices within which connections are dense, but between which connections are sparser.
  - Because we don't have any prior information about network.

Modularity

$$Q = \sum_i e_{ii} - \sum_{ijk} e_{ij} e_{ki} = \sum_i \left(e_{ii} - \left(\sum_j \left(e_{ij} \right) \right)^2 \right) = \mathrm{Tr}(e) - \left\| \mathrm{e}^2 \right\|$$



e_{ij}: the fraction of edges in the original network connecting nodes in community i to those in community j

Note that the above representation is another form of modularity presented in the previous slide

Structural definition of Community

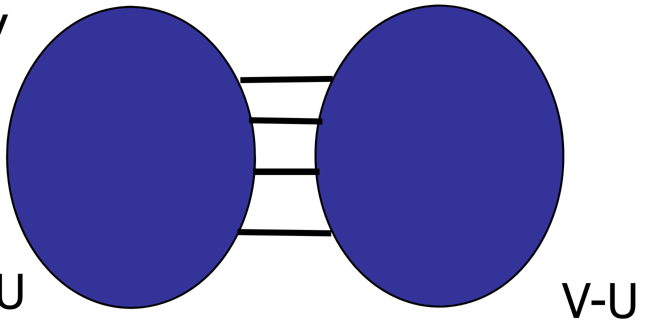
- An LS-set or strong community is a subgraph such that the internal degree of each vertex is greater than its external degree
- Weak community is a subgraph such that the internal degree of the subgraph exceeds its external degree

The methods

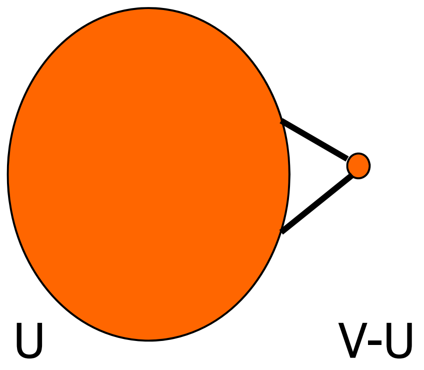
- What property or measure of network is used in this algorithm or method?
  - eigenvalue and eigenvector, spectrum of adjacency matrix (we briefly discussed this one)
  - Edge betweenness, information centrality
  - Distance, dissimilarity index, edge clustering coefficient, etc
- Agglomerative or divisive?
- What is the required prior information here?
  - Whether there is community or not
  - How many modules are there
- Performance of partitioning results and computational complexity

Measuring connectivity

- What does it mean that a set of nodes are well or sparsely interconnected?
- min-cut: the min number of edges such that when removed cause the graph to become disconnected
  - small min-cut implies sparse connectivity
  - $\min_{U} E(U, V-U) = \sum_{i \in U} \sum_{j \in V-U} A[i,j]$

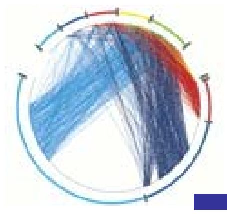


  - not always a good idea!



Graph expansion

- Normalize the cut by the size of the smallest component
- Cut ratio: $$\alpha = \frac{E(U, V-U)}{\min\{|U|, |V-U|\}}$$
- Graph expansion: $$\alpha(G) = \min_{U} \frac{E(U, V-U)}{\min\{|U|, |V-U|\}}$$
- The spectral properties of the Laplacian matrix L are related to the graph expansion, how?
  - The first eigenvalue of L is equal to zero (why?)
  - The second smallest eigenvalue (also known as Fielder value) λ_2 satisfies $$\lambda_2 = \min_{x, \|x\|=1} x^T L x$$
  - The vector that minimizes λ_2 is called the Fielder vector. It minimizes $$\lambda_2 = \min_{x \neq 0} \frac{\sum_{(i,j)\in E} (x_i - x_j)^2}{\sum_i x_i^2} \qquad \text{where} \qquad \sum_i x_i = 0$$

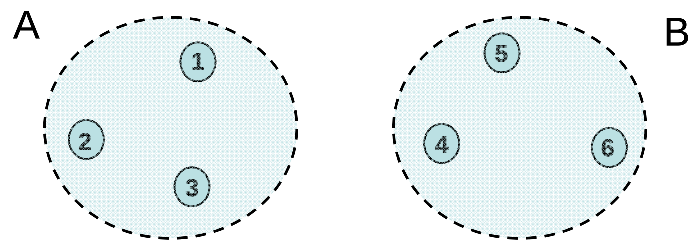


Spectral partitioning

- The ordering according to the x_i values (the entry i in the eigenvector corresponding to λ_2) will group similar (connected) nodes together
- Physical interpretation: The stable state of springs placed on the edges of the graph
- Partition the nodes according to the ordering induced by the Fielder vector
  - If $u = (u_1,u_2,\ldots,u_n)$ is the Fielder vector, then split nodes according to a value s
    - bisection: s is the median value in u
    - ratio cut: s is the value that minimizes α
    - sign: separate positive and negative values ($s=0$)
    - gap: separate according to the largest gap in the values of u
- This works well (provable for special cases)

Graph partitioning

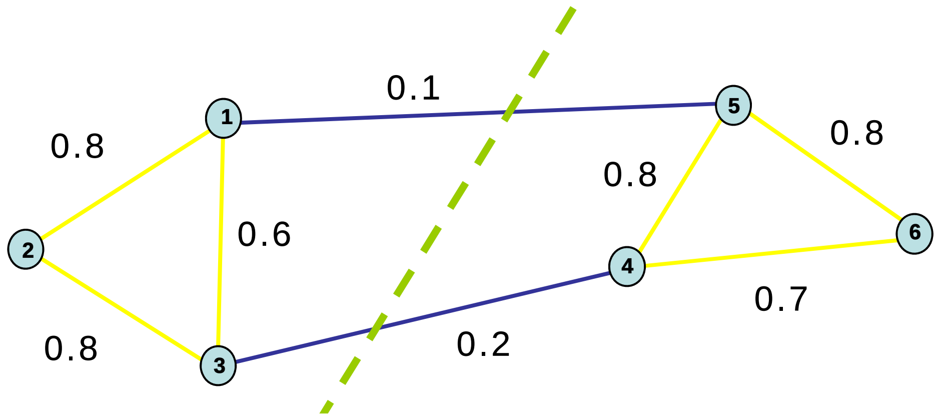
- Clustering can be viewed as partitioning a similarity graph
- *Bi-partitioning* task:
  - Divide vertices into two disjoint groups (A,B)



V=A U B ; Graph partition is NP hard

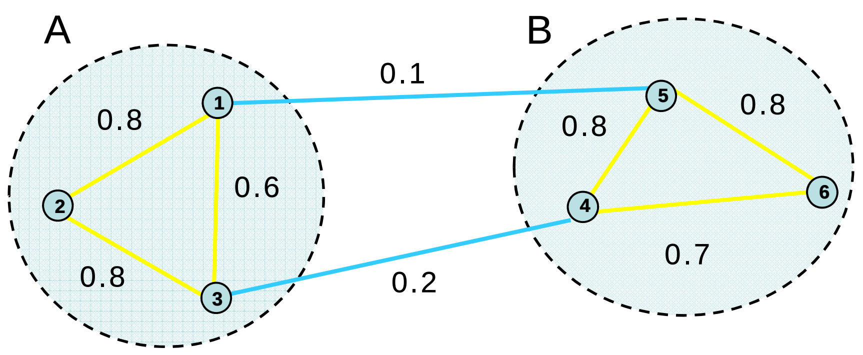
- Traditional definition of a "good" clustering:
  1. Points assigned to same cluster should be highly similar.
  2. Points assigned to different clusters should be highly dissimilar.

- Apply these objectives to our graph representation
- Minimize weight of between-group connections



Graph cuts

- Express partitioning objectives as a function of the "edge cut" of the partition.
- *Cut:* Set of edges with only one vertex in a group. We want to find the minimal cut between groups. The groups that has the minimal cut would be the partition



$$cut(A,B) = \sum_{i \in A, j \in B} w_{ij}$$

$$cut(A,B) = 0.3$$

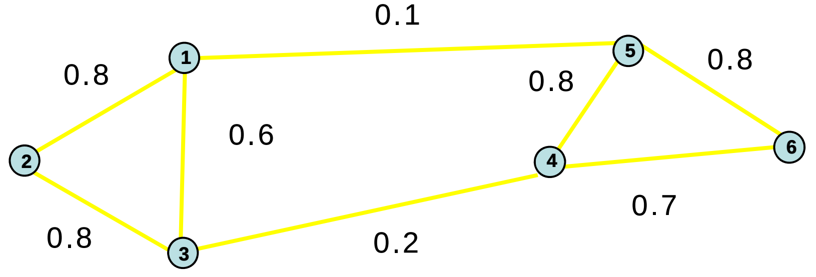
- **Criterion: Minimum-cut**
  - Minimise weight of connections between groups

$$\min\ cut(A,B)$$

Spectral graph theory

- **Laplacian matrix (L)**
  - *N-by-N* symmetric matrix

$$L = D - A$$



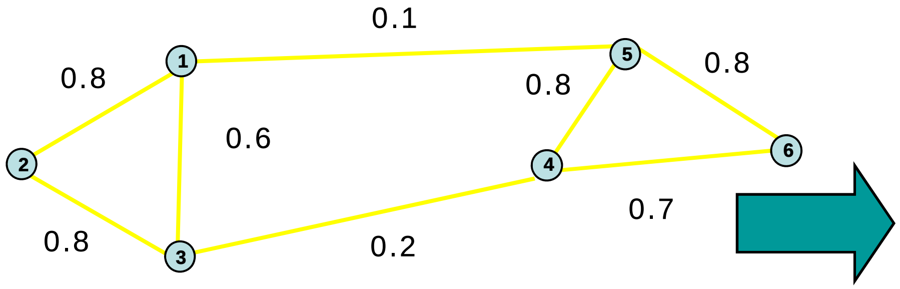
	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	0	0	-0.2	1.7	-0.8	-0.7
x_5	-0.1	0	0	-0.8	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

- Important properties:
  - Eigenvalues are non-negative real numbers
  - Eigenvectors are real and orthogonal
  - Eigenvalues and eigenvectors provide an insight into the connectivity of the graph…

Normalized Laplacian

- **Laplacian matrix (L)**
  - *N-by-N* symmetric matrix

$$D^{-0.5} \cdot (D - A) \cdot D^{-0.5}$$



1.00	-0.52	-0.39	0.00	-0.06	0.00
-0.52	1.00	-0.50	0.00	0.00	0.00
-0.39	-0.50	1.00	-0.12	0.00	0.00
0.00	0.00	-0.12	1.00	0.47-	0.44-
-0.06	0.00	0.00	-0.47	1.00	0.50-
0.00	0.00	0.00	0.44-	0.50-	1.00

- Important properties:
  - Eigenvectors are real and normalized
  - Each Aij (which i,j is not equal) = $\frac{-A_{ij}}{Dii}$

Optimal min cut (Fiedler 73)

- Express a bi-partition (A,B) as a vector

$$p_i = \begin{cases} +1 & \text{if } x_i \in \mathrm{A} \\ -1 & \text{if } x_i \in \mathrm{B} \end{cases} = p^T L\, p$$

Laplacian matrix

- We can minimise the cut of the partition by finding a non-trivial vector p that minimises the function $f(p) = \sum_{i,j \in V} w_{ij}(p_i - p_j)^2$

- The Laplacian is semi positive
- The *Rayleigh Theorem* shows:
  - The minimum value for $f(p)$ is given by the 2nd smallest eigenvalue of the Laplacian L.
  - The optimal solution for *p* is given by the corresponding eigenvector of λ_2, referred as the *Fiedler Vector*.

Fielder Value

- The value λ_2 is a good approximation of the graph expansion

$$\frac{\alpha(A)^2}{2d} \le \lambda_2 \le 2\alpha(A)$$

d = maximum degree

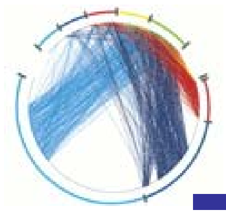
$$\frac{\lambda_2}{2} \le \alpha(G) \le \sqrt{\lambda_2(2d-\lambda_2)}$$

- For the minimum ratio cut of the Fielder vector we have that

$$\frac{\alpha^2}{2d} \le \lambda_2 \le 2\alpha(A)$$

- If the max degree d is bounded we obtain a good approximation of the minimum expansion cut

- A: the adjacency matrix of the graph



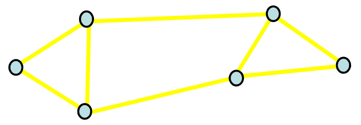
Spectral clustering algorithm

- **Three basic stages:**

1. Pre-processing
   - Construct a matrix representation of the dataset (Construct the network structure).
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 bi-partitioning algorithm

1. Pre-processing
   - Build Laplacian matrix L of the graph
2. Decomposition
   - Find eigenvalues X and eigenvectors Λ of the matrix L
   - Map vertices to corresponding components of λ_2



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	0	0	-0.2	1.7	-0.8	-0.7
x_5	-0.1	0	0	-0.8	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

$\Lambda =$

0.0
0.4
2.2
2.3
2.5
3.0

$X =$

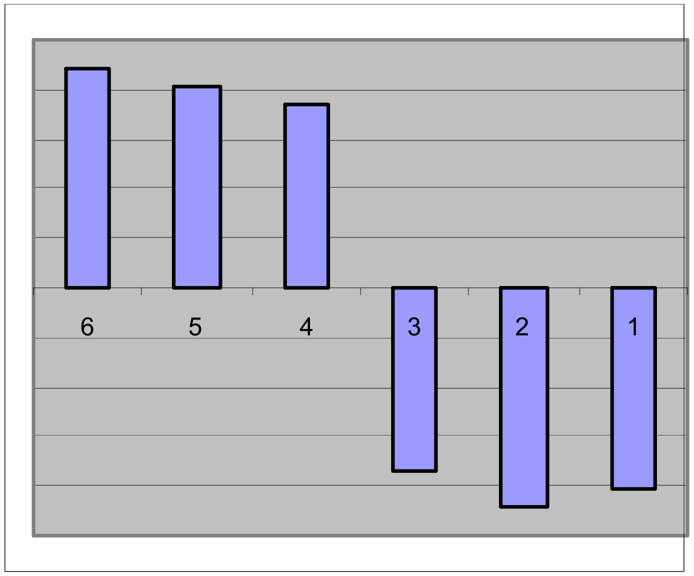
0.4	0.2	0.1	0.4	-0.2	-0.9
0.4	0.2	0.1	-0.	0.4	0.3
0.4	0.2	-0.2	0.0	-0.2	0.6
0.4	-0.4	0.9	0.2	-0.4	-0.6
0.4	-0.7	-0.4	-0.8	-0.6	-0.2
[illegible]	-0.7	-0.2	0.5	0.8	0.9

x_1	0.2
x_2	0.2
x_3	0.2
x_4	-0.4
x_5	-0.7
x_6	-0.7

Spectral bi-partitioning algorithm

The matrix which represents the eigenvector of the Laplacian - the eigenvector matched to the corresponded eigenvalues with increasing order

0.41	-0.41	0.65-	0.31-	0.38-	0.11
0.41	-0.44	0.01	0.30	0.71	0.22
0.41	-0.37	0.64	0.04	0.39-	0.37-
0.41	0.37	0.34	0.45-	0.00	0.61
0.41	0.41	0.17-	0.30-	0.35	0.65-
0.41	0.45	0.18-	0.72	0.29-	0.09



Spectral bi-partitioning algorithm

- 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, mean or median value
  - More expensive approaches
    - Attempt to minimise normalised cut criterion in 1-dimension

x_1	0.2
x_2	0.2
x_3	0.2
x_4	-0.4
x_5	-0.7
x_6	-0.7

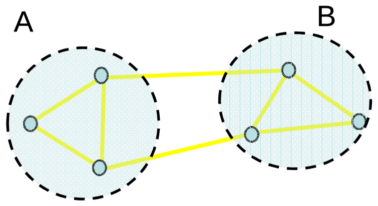
Split at 0

Cluster A: Positive points

Cluster B: Negative points

x_1	0.2
x_2	0.2
x_3	0.2

x_4	-0.4
x_5	-0.7
x_6	-0.7



k-way spectral clustering

- How do we partition a graph into k clusters?

- Two basic approaches:

1. **Recursive bi-partitioning** (Hagen et al.,'91)
   - Partition using only one eigenvector at a time
   - Recursively apply bi-partitioning algorithm in a hierarchical divisive manner.
   - Disadvantages: Inefficient, unstable
2. **Cluster multiple eigenvectors** (Shi & Malik,'00)
   - Build a reduced space from multiple eigenvectors.
   - Commonly used in recent papers
   - A preferable approach…but it is like to do PCA and then k-means

Why to use multiple eigenvalues?

1. **Approximates the optimal cut** (Shi & Malik,'00)
   - Can be used to approximate the optimal k-way normalised cut.
2. **Emphasises cohesive clusters** (Brand & Huang,'02)
   - Increases the unevenness in the distribution of the data.
   - $\Rightarrow$ Associations between similar points are <u>amplified</u>, associations between dissimilar points are <u>attenuated</u>.
   - $\Rightarrow$ The data begins to "approximate a clustering".
3. **Well-separated space**
   - Transforms data to a new "embedded space", consisting of *k* orthogonal basis vectors.

- NB: Multiple eigenvectors prevent instability due to information loss.

k-eigenvector clustering

- **K-eigenvector Algorithm** (Ng et al.,’01)

1. Pre-processing
   - Construct the *scaled adjacency matrix*

$$A' = D^{-1/2} A D^{-1/2}$$

2. Decomposition
   - Find the eigenvalues and eigenvectors of A'.
   - Build embedded space from the eigenvectors corresponding to the k largest eigenvalues.
3. Grouping
   - Apply *k*-means to reduced $n \; x \; k$ space to produce k clusters.

How to select k?

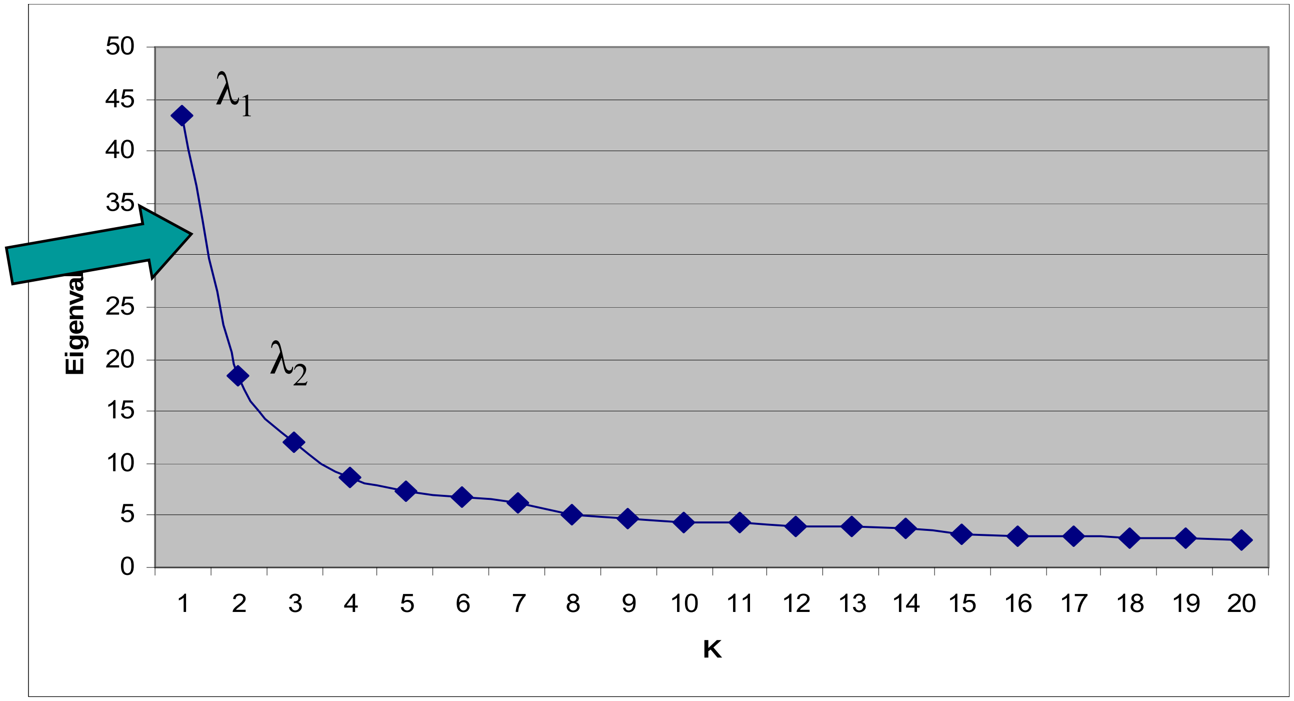
- *Eigengap*: the difference between two consecutive eigenvalues.
- Most stable clustering is generally given by the value k that maximises the expression $\Delta_k = |\lambda_k - \lambda_{k-1}|$

Largest eigenvalues

of a sample data

$$\max \Delta_k = |\lambda_2 - \lambda_1|$$

$\Rightarrow$ Choose $k=2$



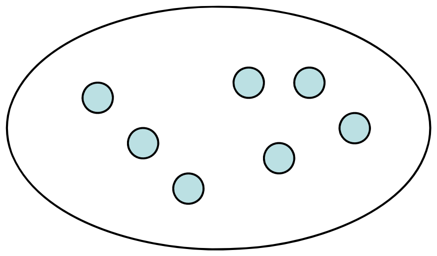
Conductance

- The expansion does not capture the inter-cluster similarity well
  - The nodes with high degree are more important
- Graph Conductance:

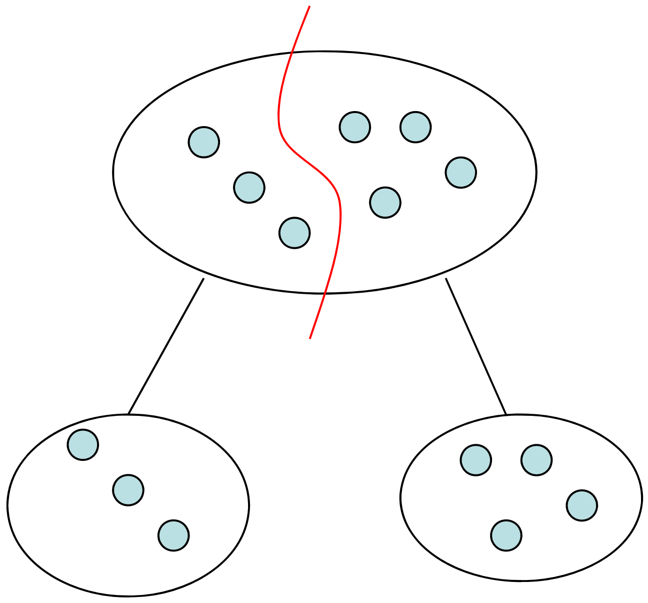
$$\varphi(G) = \min_{U} \frac{E(U, V - U)}{\min\{d(U), d(V - U)\}}$$

  - $d(U) = \sum_{i \in U} \sum_{j \in U} A[i, j]$ is weighted degrees of nodes in U
- Consider the normalized stochastic matrix $M = D^{-1}A$, where A is the adjacency matrix and D the diagonal degrees
  - Conductance φ is related to the second eigenvalue of the matrix M
- Low conductance means that there is some bottleneck and a subset of nodes not well connected with the rest of the graph
- High conductance means that the graph is well connected

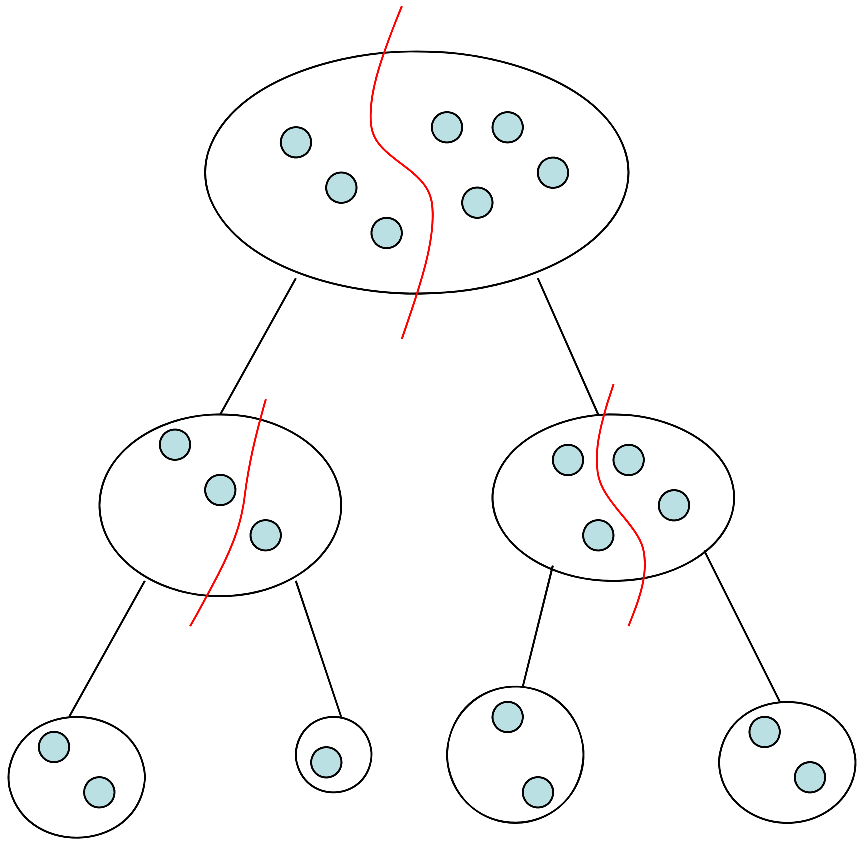
An example



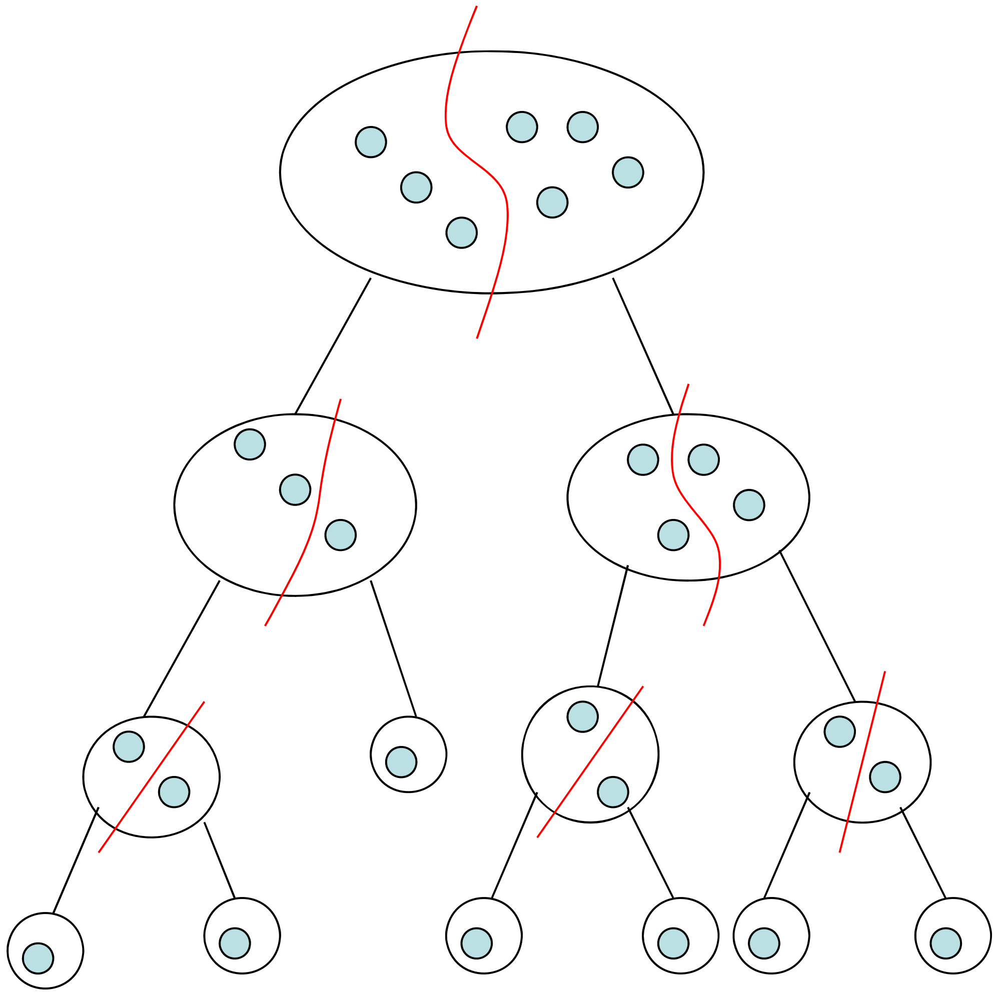
An example



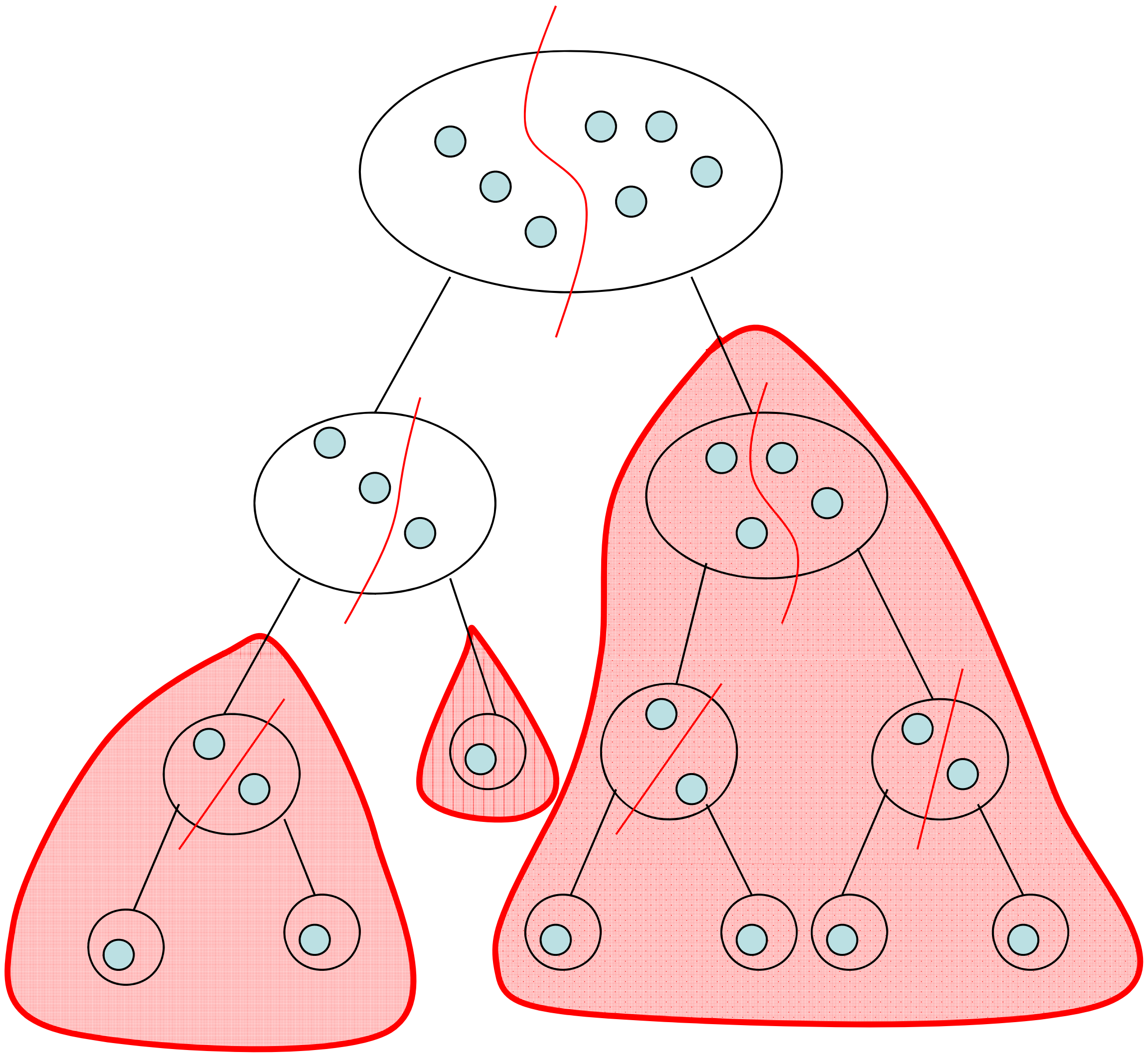
An example



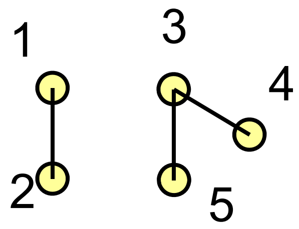
An example



An example



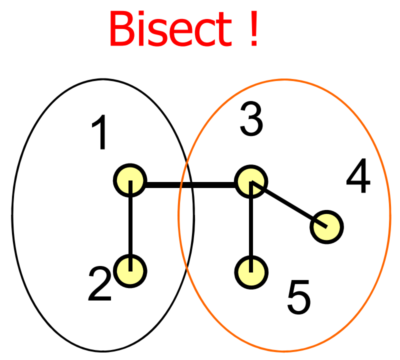
A numerical example



$$\Rightarrow \quad L = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

$1 = [1,1,1,1,1]$ is always eigenvector with eigenvalue 0.

A numerical example



$$L = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 3 & -1 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

$$V = \begin{bmatrix} 0.45 & \boxed{0.34} & 0 & -0.70 & 0.44 \\ 0.45 & 0.70 & 0 & 0.54 & -0.14 \\ 0.45 & -0.20 & 0 & -0.32 & -0.81 \\ 0.45 & -0.42 & -0.71 & 0.24 & 0.26 \\ 0.45 & -0.42 & 0.71 & 0.24 & 0.26 \end{bmatrix}$$

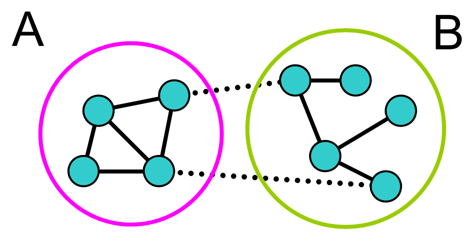
$$\text{Eigenvalues} = [0.00,\ \underbrace{0.52}_{\text{Algebraic connectivity}},\ 1.00,\ 2.31,\ 4.17]$$

- The eigenvector corresponding to the lowest eigenvalue must have both positive and negative elements.
- The spectral bisection method is reasonably fast.
- General n×n matrix case, $O(n^3)$ time complexity
- However, for sparse matrices, Lancozos method reduces it to approximately $O(m/(\lambda_3-\lambda_2))$, m being the number edges
- How good the split is, with smaller values corresponding to better splits

The Kernighan-Lin algorithm

Benefit function Q

The number of edges that lie within the two groups minus the number that lie between them.



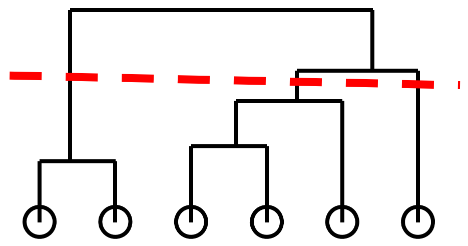
Bisect !

1. We should Specify the size of the two groups. N(A), N(B) Calculate the ΔQ for all possible exchange pair from A and B.
2. Choose the pair that maximizes the change of Q (greedy algorithm).
3. Repeat 2 & 3 until all vertices have been swapped once (any vertex that has been swapped is never swapped).
4. Go back over the sequence of swaps and find the highest Q.

- This algorithm requires a priori what the size of the groups will be
- It runs moderately quickly, in worst case time $O(n^2)$
- However, if we do not know the size, It will increase to $O(n^3)$
- The best values of Q are always achieved for very asymmetric trivial division

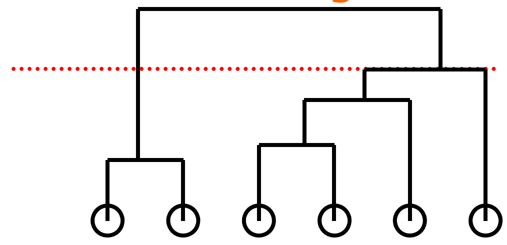
Dendrogram

- With weights, n nodes in the network, with no edges between them are initialized.
- Edges are added between pairs one by one in order of their weights, starting with the pair with strongest weight and progressing to the weakest.
- As edges are added, the resulting graph shows a nested set of increasingly large components which are taken to be the communities.
- In this graph, the lowest level at which two vertices are connected represents the strength of the edge that resulted in their first becoming members of the same community. Trees of this type are sometimes called dendrograms.



Hierarchal clustering

Dendrogram



Metrics:

1. Measure of similarity X_{ij} between pairs (i,j) of vertices.
2. Single linkage, complete linkage, or average linkage.

Structural equivalence :
Two vertices are said to be structurally equivalent if they have the same set of neighbors. How many same friends they have.

$$x_{ij} = \sqrt{\sum_{k \neq i,j} (A_{ik} - A_{ij})^2}$$

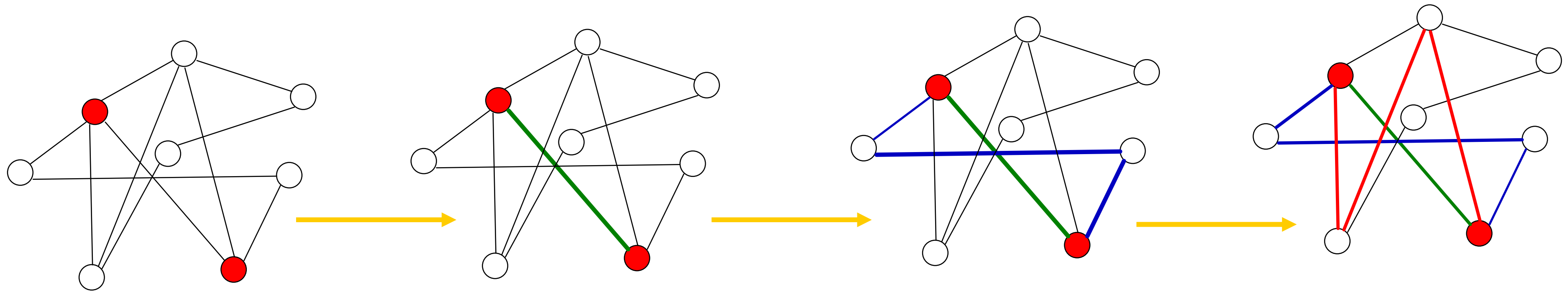
Euclidean distance

K-components :
Two nodes in the same community have at least k independent paths between them. The count of edge-independent path (max-flow) between-nodes.

Time complexity Max(O(mn), O(n2logn)), because of the sorting of n^2 similarity

Hierarchal clustering

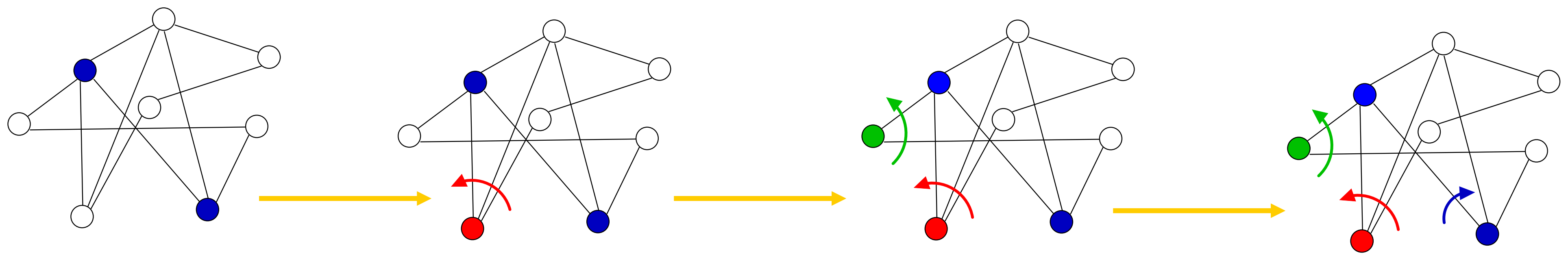
- Many weight calculation methods have been proposed for use with hierarchical clustering algorithms.
- One possible definition of the weight is the number of node-independent path between nodes.
- Two paths that connect the same pair of nodes are said to be node-independent if they share none of the same nodes other than their initial and final nodes.



3(2 without initial) Paths

Hierarchal clustering

- It is also possible to use Edge-Independent path.
- Edge-independent paths between two vertices i and j in a graph is equal to the minimum number of vertices (edges) that must be removed from the graph to disconnect i and j from one another.
- Thus these numbers are in a sense a measure of the robustness of the network to deletion of nodes.



Hierarchal clustering: implementation

- There are possible polynomial-time algorithms to calculate independent path in a network, e.g. :
  - Max-Flow Algorithm
  - Augmenting Path Algorithm
- Both method have a tendency to separate single peripheral nodes from the communities to which they should rightly belong. If a vertex is, for example, connected to the rest of a network by only a single edge then, to the extent that it belongs to any community.
- It is undesirable that single nodes often remain isolated from the network when the communities are constructed.
- Poor results from these methods in well known networks make the hierarchical clustering method, although useful, far from perfect.

Zhu dissimilarity index method

The distance d_{ij} from vertex i to vertex j is defined as the average number of steps needed for a Brownian particle on this network to move from node i to node j.

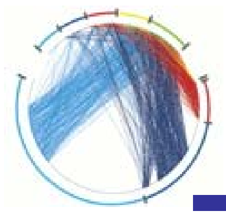
Transfer matrix (jumping probability) $$P_{ij} = \frac{A_{ij}}{\sum_{l=1}^{N} A_{il}} \sim \rho(k)$$

Distance $$d_{i,j} = \sum_{l=1}^{N} \left(\frac{1}{I - B(j)} \right)_{il}$$

- I is N×N identity matrix
- B(j) is equals to P except that $B_{lj}(j) = 0$ for all l.

Dissimilarity index $$\Lambda(i,j) = \frac{\sqrt{\sum_{k \neq i,j}^{N} \left| d_{i,k} - d_{j,k} \right|^2}}{N-2}$$

$$\because \left| I - B(j) \right| \begin{pmatrix} d_{1,j} \\ \vdots \\ d_{N,j} \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$



Edge-betweenness based methods

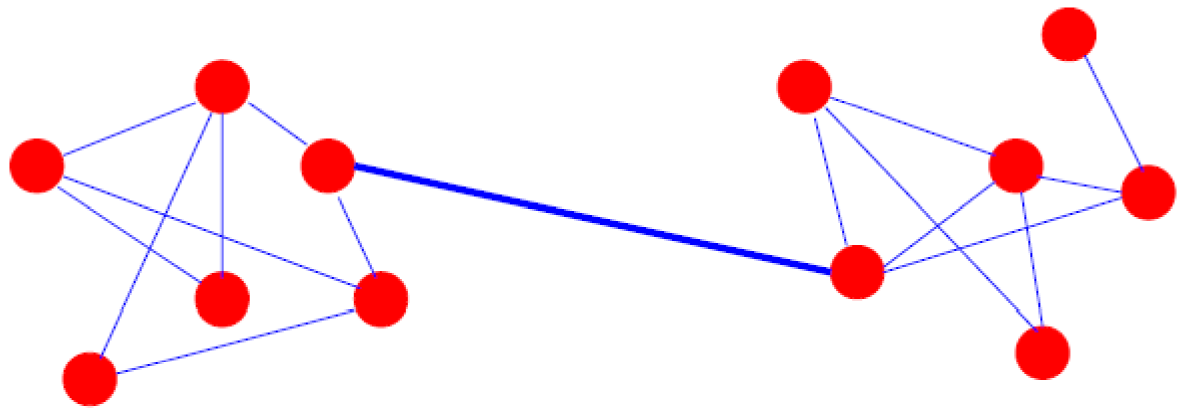
- Edge Betweenness is an alternative approach to the detection of the communities.
- Instead of trying to construct a measure that tells us which edges are most central to communities, it is focused on those edges that are least central or the edges that are most "between" communities.
- Rather than constructing communities by adding the strongest edges to an initially empty vertex set, the original graph is constructed by progressively removing edges from it.
- The most important method of this kind in the market is a series of methods proposed by Newman.

Girvan-Newman algorithm

- If a network contains communities or groups that are only loosely connected by a few intergroup edges, then all shortest paths between different communities must go along one of these few edges.
- Thus, the edges connecting communities will have high edge betweenness.
- By removing these edges, we separate groups from one another and so reveal the underlying community structure of the graph.
- The Girvan-Newman Algorithm:
  1. Calculate the betweenness for all edges in the network
  2. Remove the edge with the highest betweenness
  3. Recalculate betweennesses for all edges affected by the removal
  4. Repeat from step 2 until no edges remain

Girvan-Newman algorithm

- Edge betweenness is the largest for edges connecting communities. In the below figure, the edge in the middle has a much higher betweenness than all other edges, because all shortest paths connecting vertices of the two communities run through it.

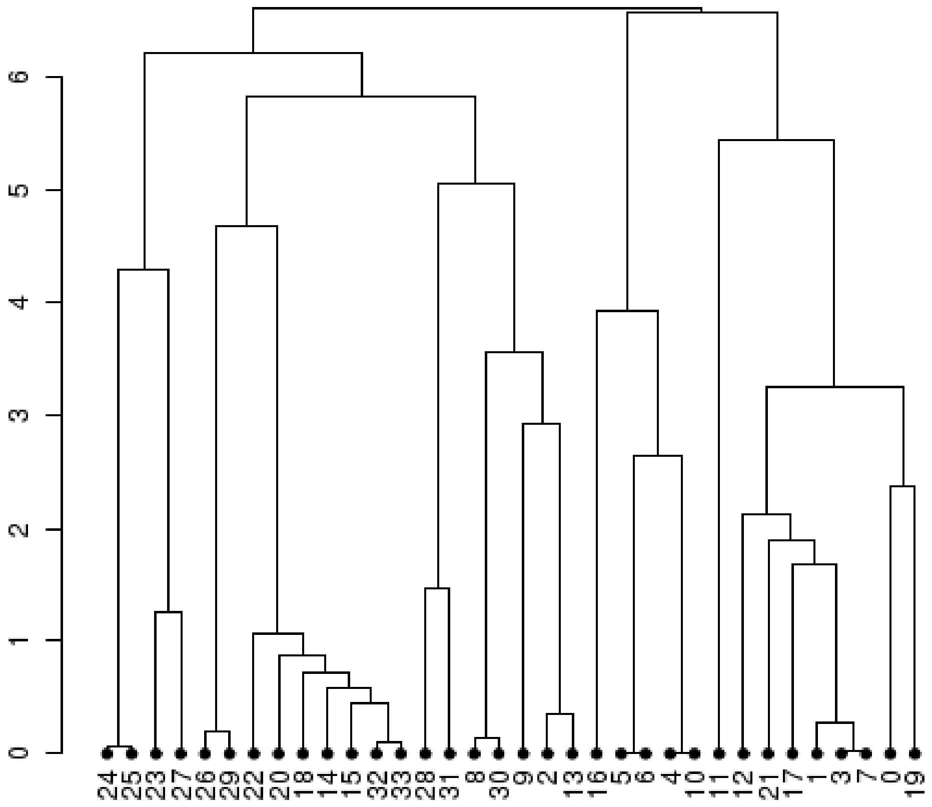


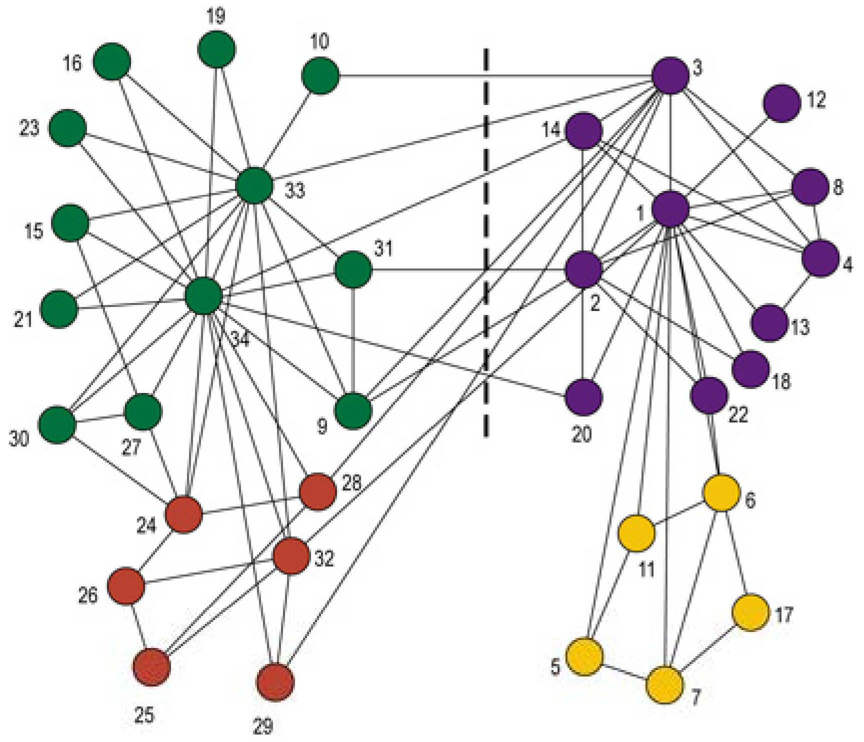
Girvan-Newman algorithm

- Because this calculation has to be repeated once for the removal of each-edge, the entire algorithm runs in worst-case time O(m^2n).
- However, after the removal of each edge, we only have to recalculate the betweennesses of those edges that were affected by the removal, which is at most only those in the same component as the removed edge. This means that running time may be better than the worst-case for networks with strong community structure.
- To improve performance, the betweennesses can be calculated by using the fast algorithm of Newman, which calculates betweenness for all m edges in a graph of n vertices in time O(mn).
- To try to reduce the running time of the algorithm further, one might be tempted to calculate the betweennesses of all edges only once and then remove them in order of decreasing betweenness. However, this strategy does not work well, because if two communities might be connected by more than one edge !

Girvan-Newman algorithm

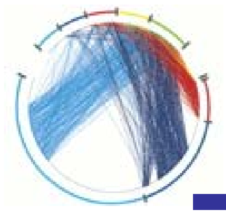
- Application to Zachary Karate club
- A well-studied social network with 34 nodes and 78 edges
- Data available at http://www-personal.umich.edu/~mejn/netdata/





Tyler-Wilkinson-Huberman method

- Variation of Girvan-Newman algorithm to improve the calculating speed.
- They suggest instead of summing up over all nodes, only a subset of vertices i be summed over, giving partial betweenness score for all edges; if a random sample is chosen, this will give a Monte Carlo estimate of betweenness.
- The number of vertices sampled is chosen so as to make the betweenness of at least one edge in the network greater than a certain threshold.
- This stochastic approach reduces the time complexity from $O(m^2n)$ to $O(m^2)$.



Newman's fast algorithm

Modularity

$$Q = \sum_i e_{ii} - \sum_{ijk} e_{ij} e_{ki} = \mathrm{Tr}\,\mathbf{e} - \left\| \mathbf{e}^2 \right\|$$

Maximize Q by greedy algorithm !

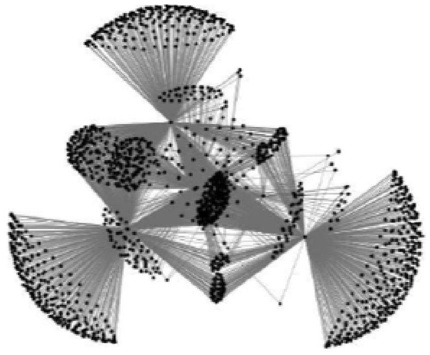
1. Separate each vertex solely into n community.
2. Calculate the increase of Q for all possible community pairs.
3. Choose the mergence of the greatest increase in Q.
4. Repeat 2 & 3 until the modularity Q reaches the maximal value.

Time Complexity - O(mn) $\longrightarrow$ O(n^2) on sparse graph.

Agglomerative hierarchical clustering method!

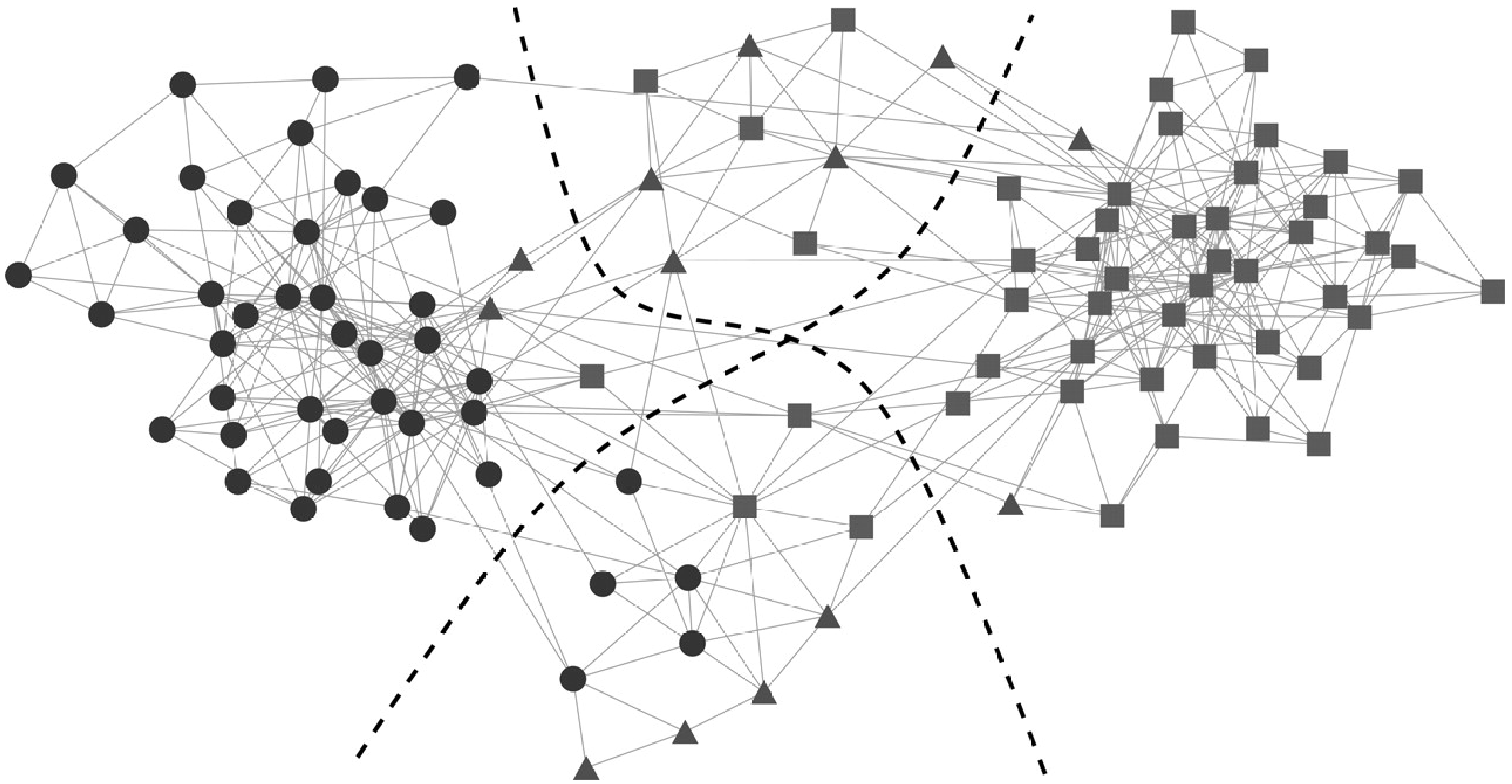
Newman's fast algorithm

- Algorithm (again)
  - start with all nodes as isolates
  - follow a greedy strategy:
    - successively join clusters with the greatest increase ΔQ in modularity
    - stop when the maximum possible $0 \geq \Delta Q$ from joining any two
  - successfully used to find community structure in a graph with more than 400,000 nodes and with more than 2 million edges
    - Amazon's people who bought this also bought that…
  - alternatives to achieving optimum ΔQ:
    - simulated annealing rather than greedy search



Example

Krebs' network of books on American politics.



Newman M E J PNAS 2006;103:8577-8582

Information centrality method

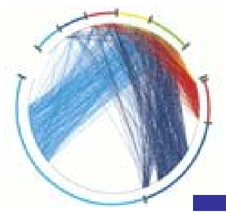
- This method considers information centrality instead of the betweenness centrality
- Consider the global efficiency of the network E
- If the efficiency changes to E_k by removing an edge k, the information centrality of this edge is defined as

$$C_k^I = \frac{\Delta E}{E} = \frac{E[G] - E[G_k^{'}]}{E[G]}$$

- The rest of the algorithm is like Girvan-Newman but based on the information centralities
- Iterative removal of the edges with the highest information centrality
- Time complexity is $O(m^3 n)$

Radicchi's method

- Communities are characterized by a high density of edges, so it is reasonable to expect that such edges form cycles.
- Edges lying between communities will hardly be part of cycles.
- Radicchi et al. proposed a new measure, the edge clustering coefficient; low values of the measure are likely to correspond to intercommunity edges.
- The edge clustering coefficient generalizes to edges the notion of clustering introduced by Watts and Strogatz (i.e. the first algorithm for calculating the clustering coefficient)



Radicchi's method

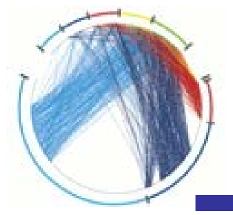
- The edge clustering coefficient is defined as

$$C_{i,j}^{(g)} = \frac{Z_{i,j}^{(g)} + 1}{s_{i,j}^{(g)}}$$

  - The edge (i,j) is between nodes i and j
  - $Z^{(g)}_{i,j}$: the number of cycles of length g built upon edge (i,j)
  - $s^{(g)}_{i,j}$: the possible number of cycles of length g that one could build based on the existing edges of (i,j) and their neighbors
- The number of actual cycles in the numerator is augmented by 1 to enable a ranking among edges without cycles, which would all yield a coefficient equal to zero, independently of the degrees of the extremes i and j and their neighbors
- Usually, cycles of length g = 3 (triangles) or 4 are considered
- The measure is (anti)correlated with edge betweenness: edges with low edge clustering coefficient usually have high betweenness and vice versa, although the correlation is not perfect

Radicchi’s method

- The method works as the algorithm by Girvan and Newman
  - At each iteration, the edge with smallest clustering coefficient is removed
  - The measure is recalculated again, and so on
- If the removal of an edge leads to a split of a subgraph in two parts, the split is accepted only if both clusters are (``strong") or ``weak" communities
- The verification of the community condition on the clusters is performed on the full adjacency matrix of the initial graph
- If the condition were satisfied only for one of the two clusters, the initial subgraph may be a random graph
- Enforcing the community condition on both clusters, it is more likely that the subgraph to be split indeed has a cluster structure.
- The algorithm stops when all clusters produced by the edge removals are communities in the strong or weak sense, and further splits would violate this condition



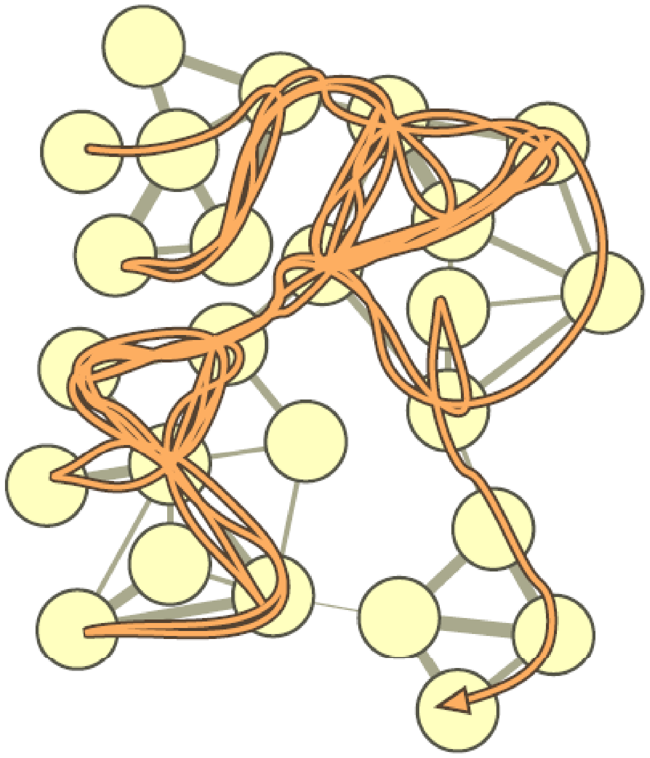
InfoMap

- Works for all types including directed and weighted networks
- Partitions the network into disjoint node sets

(M. Rosvall and C. T. Bergstrom, PNAS, 2008)

➢ Intuition: flow of information is more likely to be trapped within a community as its denser relative to its environment.

➢ Information propagates through a network like a random walker (mathematical model)



InfoMap

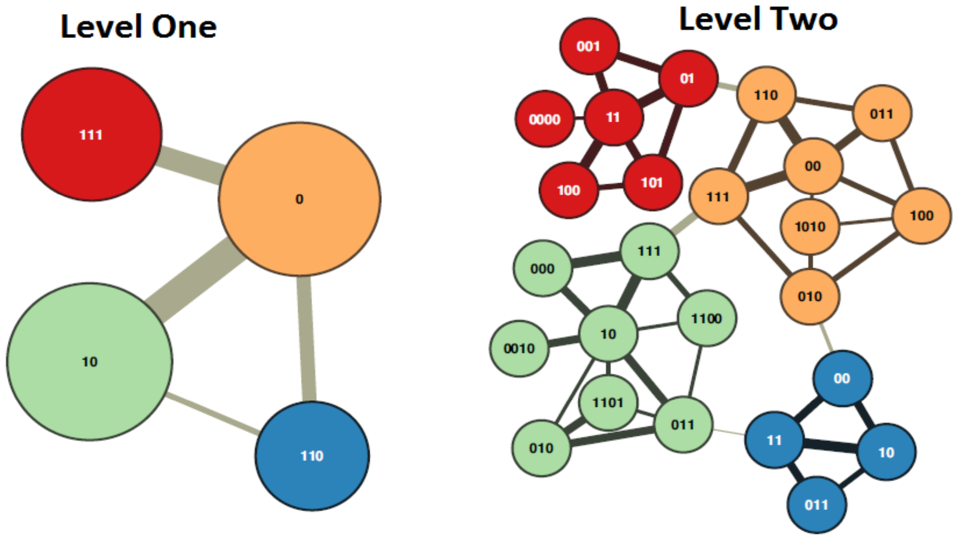
- **Objective**: find a coding that describes traversed path of a random walker as short as possible
  1) Without community structure: assign a code to each node using Hoffman coding (one level coding).
     Given P(node(i)) = probability of node(i) being visited = stationary distribution (node(i))
  2) However, we could do better using a two level coding. How?

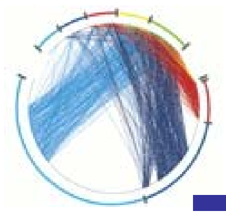
     Use a coding system for communities and a separate coding system for nodes inside each community (intra-coding of two communities may be identical !)

     - A code will be produced when walker enters a community, then the second level coding system will be used and finally a code when walker exits the community
     - Implication: we could use shorter codes for communities and their nodes

InfoMap

- **New Objective:** find a partitioning that minimizes the two level coding
  - Why this would lead us to densely connected communities?
  - If a random walker spends more time inside a partition and less frequently transits between partitions, the length of the code will be shorter. This could be achieved by densely connected partitions which are less connected to the rest of network





Modification of modularity index

- Modularity index can be extended to weighted graphs

$$Q = \frac{1}{2S}\sum_{ij}\left(W_{ij} - \frac{s_i s_j}{2S}\right)\delta\left(C_i, C_j\right) = \sum_{c=1}^{n_c}\left[\frac{W_c}{S} - \left(\frac{S_c}{2S}\right)^2\right]$$

  - W: the weighted adjacency matrix
  - s_i: strength of node I
  - S: the sum of the total weights of the edges
  - n_c: number of clusters
  - W_c: the sum of the weights of the internal edges of module c
  - S_c: the sum of the strengths of the vertices of c

Modification of modularity index

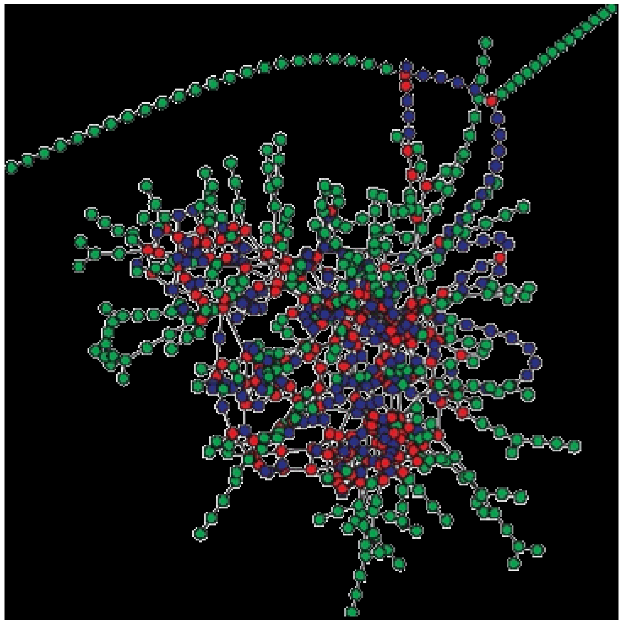
- The extension to directed graphs

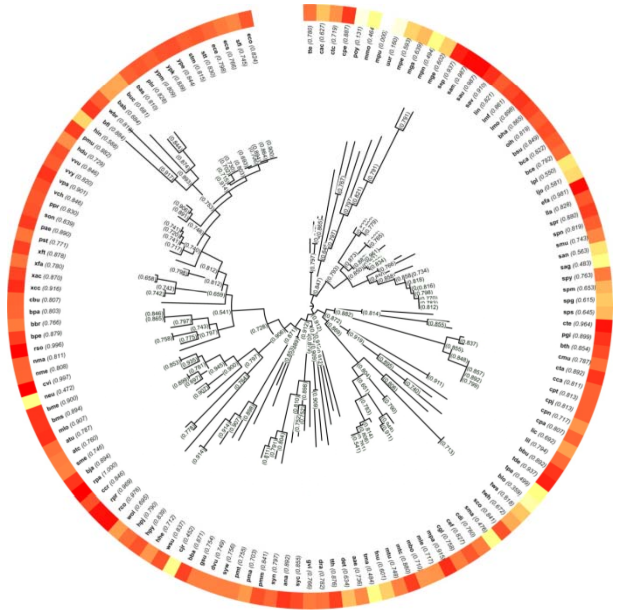
$$Q = \frac{1}{m} \sum_{ij} \left(A_{ij} - \frac{k_i^{out} k_j^{in}}{m} \right) \delta\left(C_i, C_j\right)$$

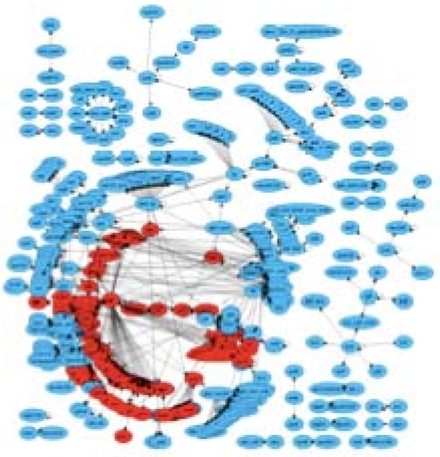
- The extension to directed weighted graphs

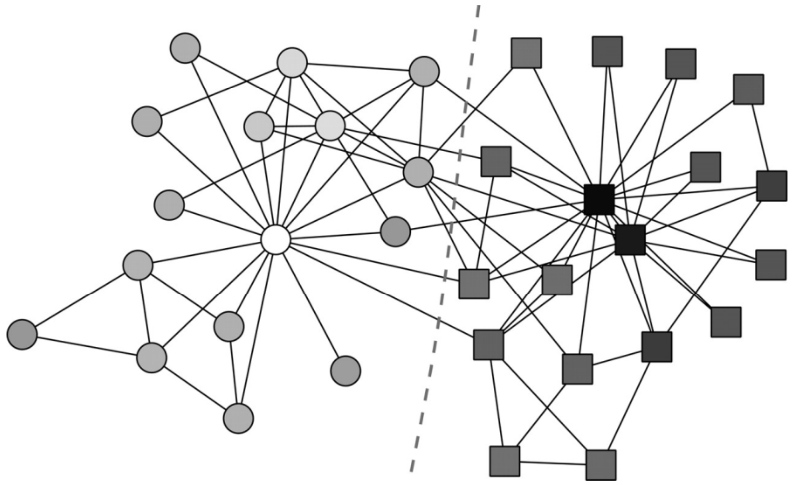
$$Q = \frac{1}{W} \sum_{ij} \left(W_{ij} - \frac{s_i^{out} s_j^{in}}{W} \right) \delta\left(C_i, C_j\right)$$

Modularity helps visualization



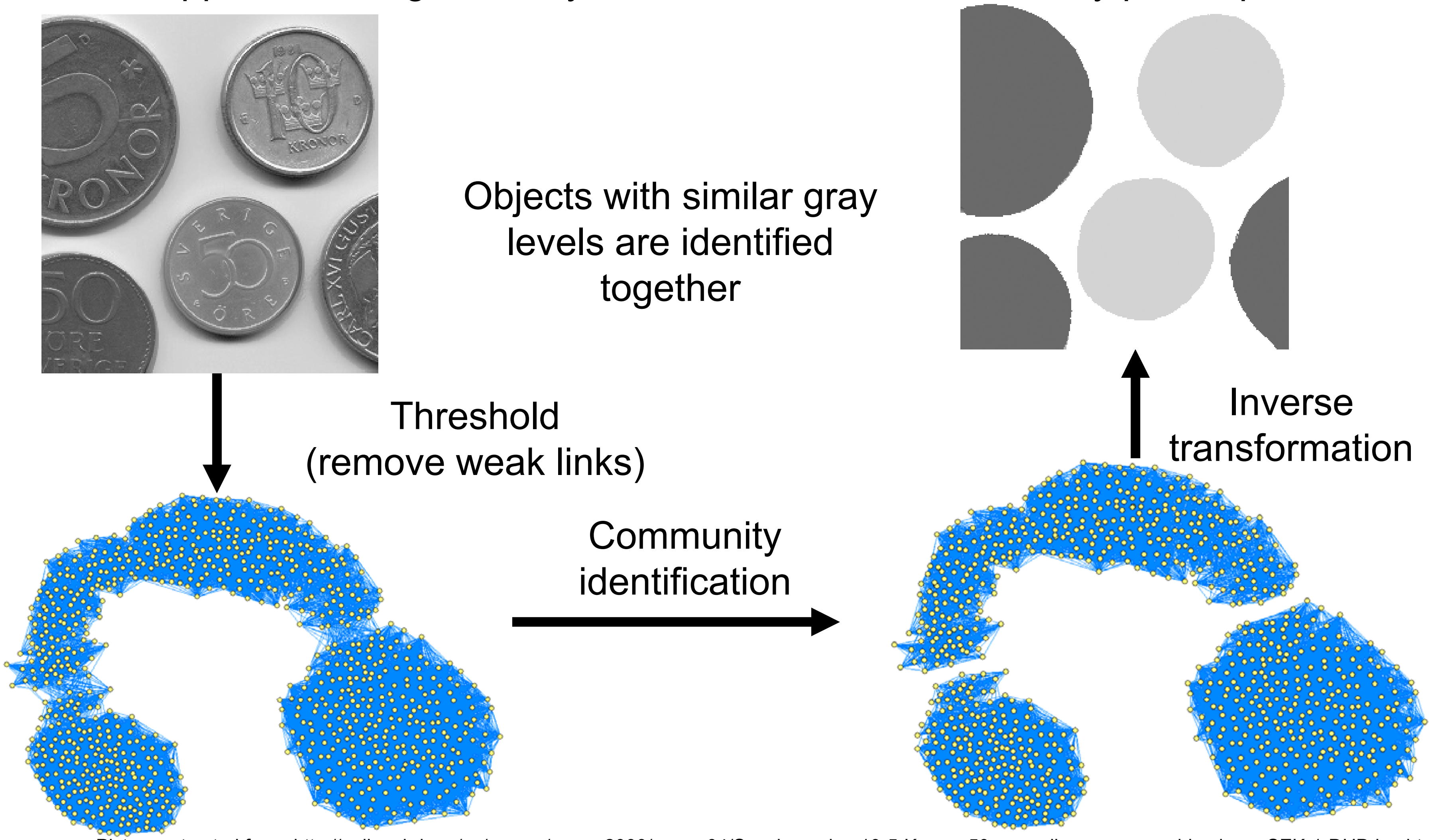






An application: segmentation

First approach: Weight = Gray level difference between every pair of pixel



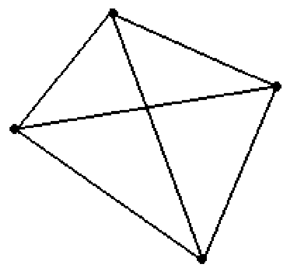
Objects with similar gray levels are identified together

Picture extracted from: http://gallery.hd.org/_c/money/_more2000/_more04/Sweden-coins-10-5-Kronor-50-oere-silver-copper-gold-colours-SEK-1-DHD.jpg.html

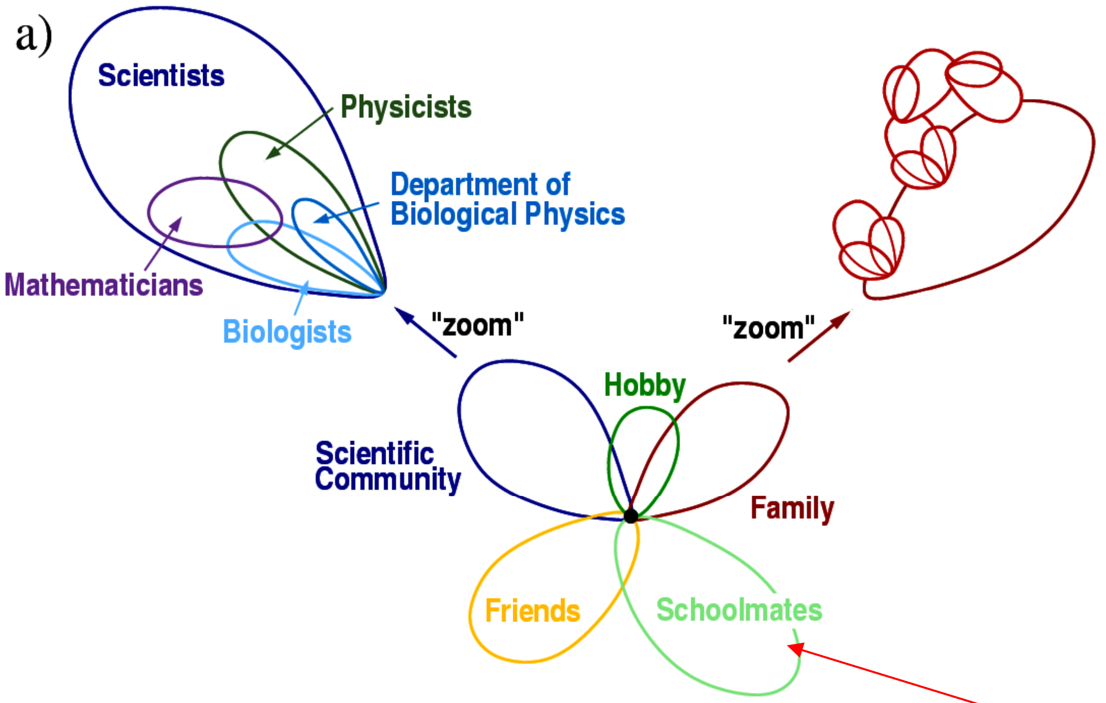
Finding overlapping communities

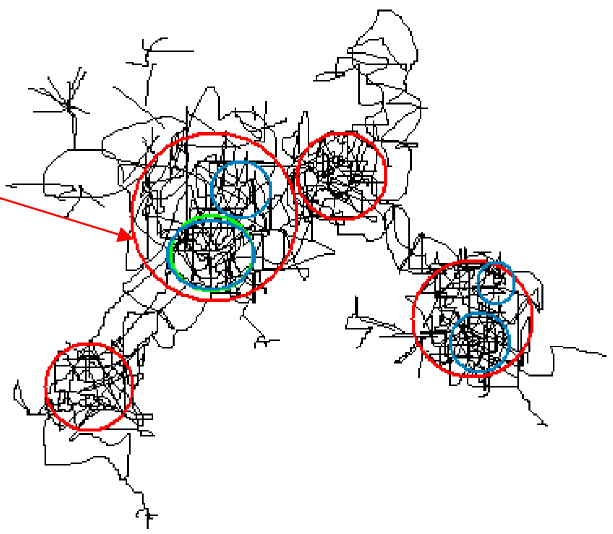
- In real graphs vertices are often shared between communities
- The issue of detecting overlapping
- The most popular technique is the Clique Percolation Method
- Based on the concept that the internal edges of a community are likely to form cliques due to their high density
- It is unlikely that intercommunity edges form cliques
- Two k-cliques are adjacent if they share k-1 nodes
- The union of adjacent k-cliques is called k-clique chain
- Two k-cliques are connected if they are part of a k-clique chain
- Finally, a k-clique community is the largest connected subgraph obtained by the union of a k-clique and of all k-cliques which are connected to it

a 4-clique



Role of overlaps

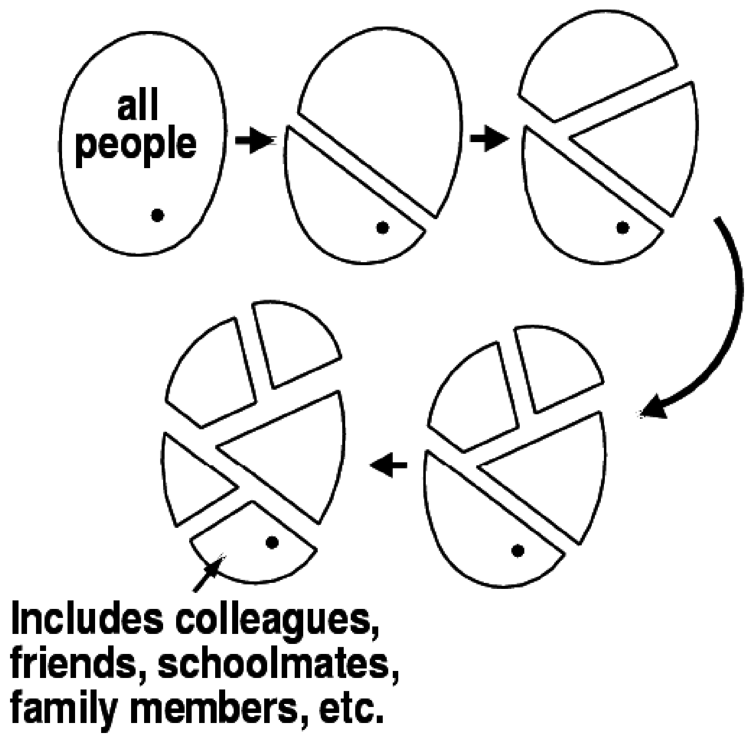




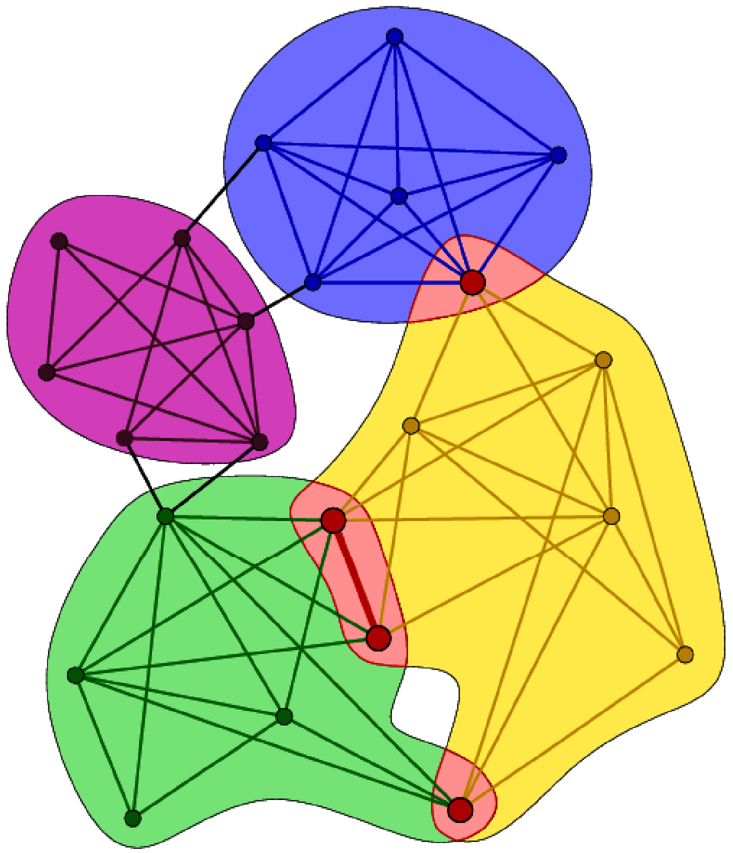
Is this like a tree? (hierarchical methods)

Finding overlapping communities

Hierarchical methods


all people
Includes colleagues, friends, schoolmates, family members, etc.

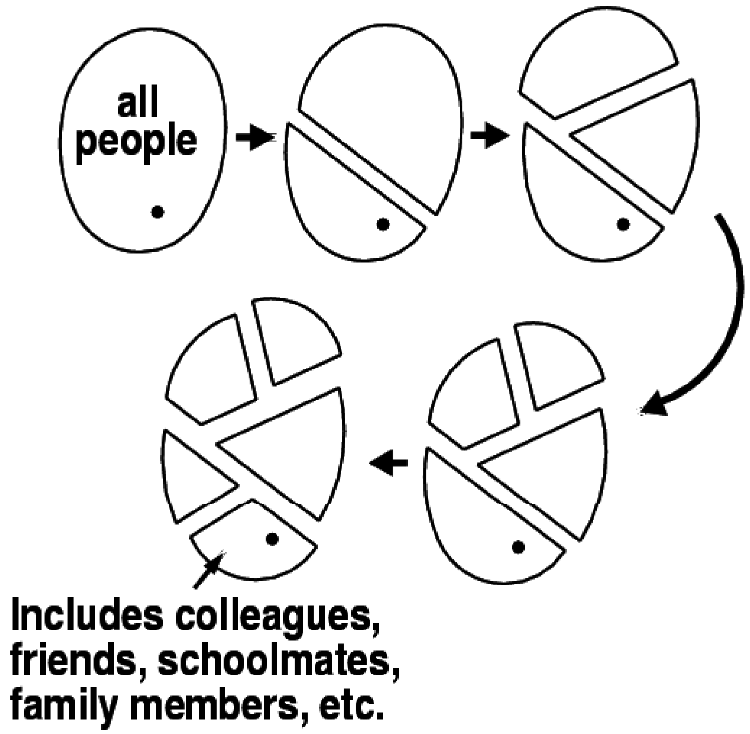
k-clique template rolling



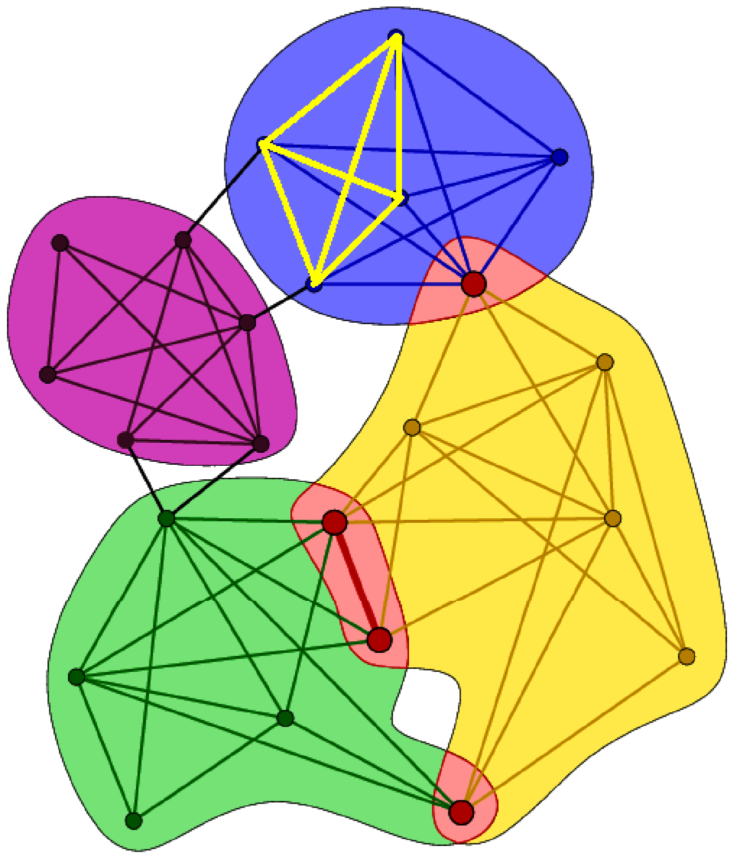
Two nodes belong to the same community if they can be connected through adjacent *k*-cliques

Finding overlapping communities

Hierarchical methods


all people
Includes colleagues, friends, schoolmates, family members, etc.

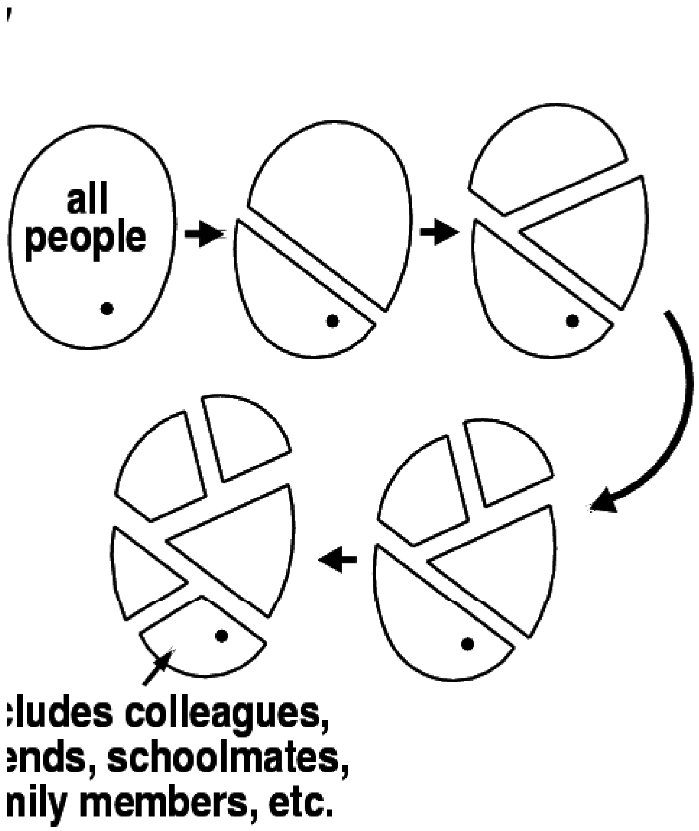
k-clique template rolling



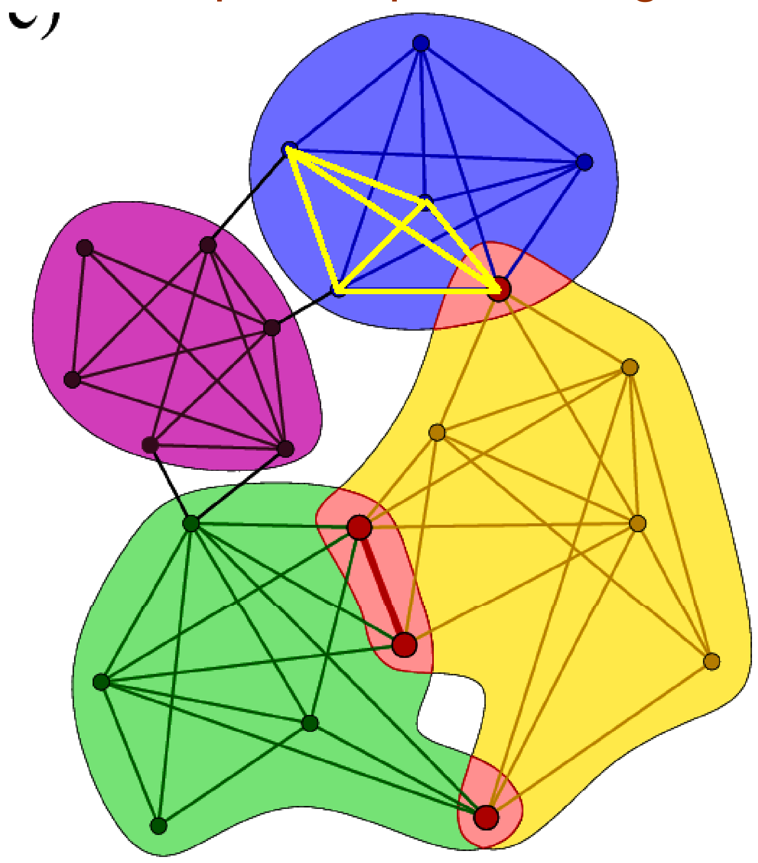
Two nodes belong to the same community if they can be connected through adjacent *k*-cliques

Finding overlapping communities

Hierarchical methods



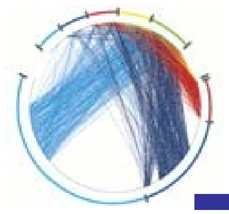
k-clique template rolling



Two nodes belong to the same community if they can be connected through adjacent *k*-cliques

Hierarchal vs. clique percolation

- Common clustering methods lead to a partitioning in which someone (a node) can belong to a single community at a time only.
- For example, one can be located as a member of the community “physicists”, but not, at the same time, be found as a member of my community “family” or “friends”, etc.
- k-clique template rolling allows large scale, systematic (deterministic) analysis of the network of overlapping communities (network of networks)



OSLOM for overlapping communities

- ❑ Works for all types including directed and weighted networks
- ❑ Partitions the network into overlapping/Hierarchical node sets

(A. Lancichinetti et al., PLoS ONE, 2011)

- ➢ Intuition: similar to the idea of modularity, a community is significant if it have a low probability of being created by chance

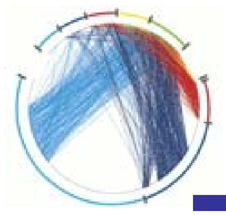
- ➢ Different from Modularity: more realistic null model which yields to complicated (yet tractable) formulations

Coarse graining

- Often real-networks comprise of many nodes
- The complexity can be reduced by reducing the number of nodes
- This process is called coarse graining
- Reducing the complexity of large systems described as complex networks is key to understand them and a crucial issue is to know which properties of the initial system are preserved in the reduced one.
- One possible solution is spectral coarse graining

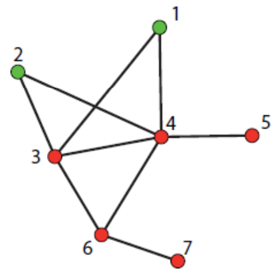
Coarse graining

- Reduce the network complexity by reducing the number of nodes
- Classical approaches
  - Clustering
  - Grouping nodes together or removing some nodes.
  - Do they preserve information!?
- **Coarse Graining**: complexity reduction techniques satisfying the goal of preserving some (maybe all) properties of a system.
  - Fine details have been smoothed over or averaged out.
  - Here the properties of interest are the main characteristics of random walk.



A spectral method

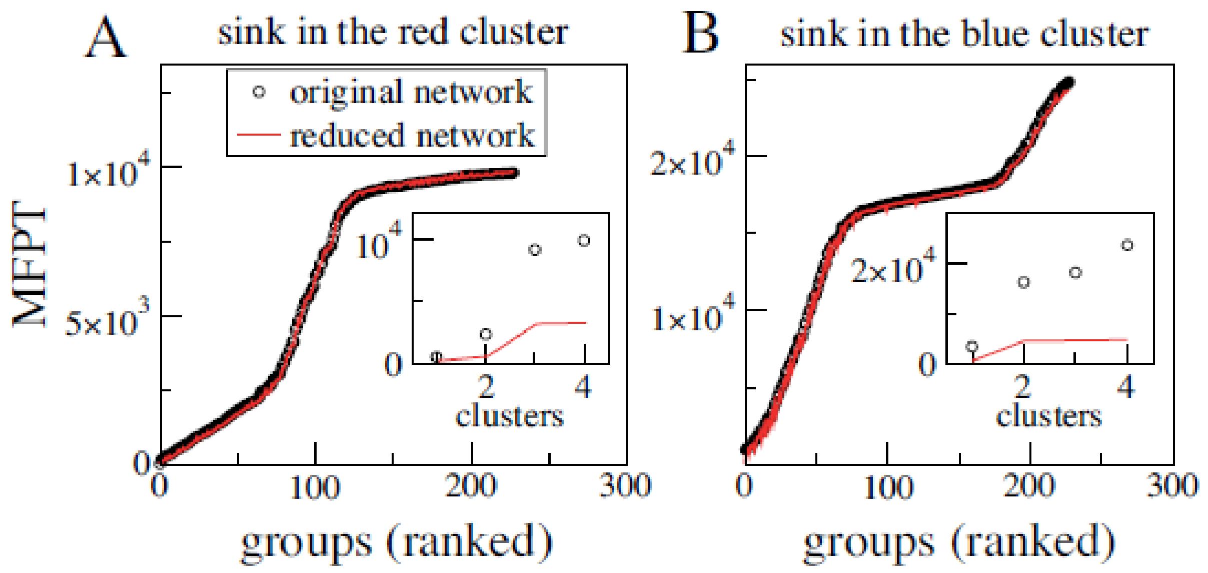
- W:stochastic matrix (W_{ij}: transition probability form j to i)
  - For connected and undirected networks: $1 = \lambda_1 \geq \lambda_2 \geq . . . \geq \lambda_N$
  - The right eigenvector p_1 corresponding to λ_1 is the stationary
  - large-scale behavior of the random walk
  - left eigenvectors are repetitive of dynamical behavior from the point of view of the random walk evolution.
- Nodes 1 and 2 have exactly the same neighbors
  - They cannot be distinguished from the point of view of a random walk starting anywhere else; i.e. $W_{i1} = W_{i2}$
  - For left eigenvector u^{α} of W, it means that $u^{\alpha}_1=u^{\alpha}_2$ for λ^{α} != 0
  - coalesce nodes satisfying $u^{\alpha}_1=u^{\alpha}_2$ with the resulting new node carrying the sum of the edges of the original ones.



Mean First Passing Time in Coarse-Graind Network

T_{ij}= mean first passing time from node j to node i

$$T_{ij} = \sum_{t=0}^{\infty} t(\hat{W}^t)_{ij} = \sum_{\alpha=1}^{N} \hat{u}_i^\alpha \hat{p}_j^\alpha \frac{\hat{\lambda}^\alpha}{(1-\hat{\lambda}^\alpha)^2}$$

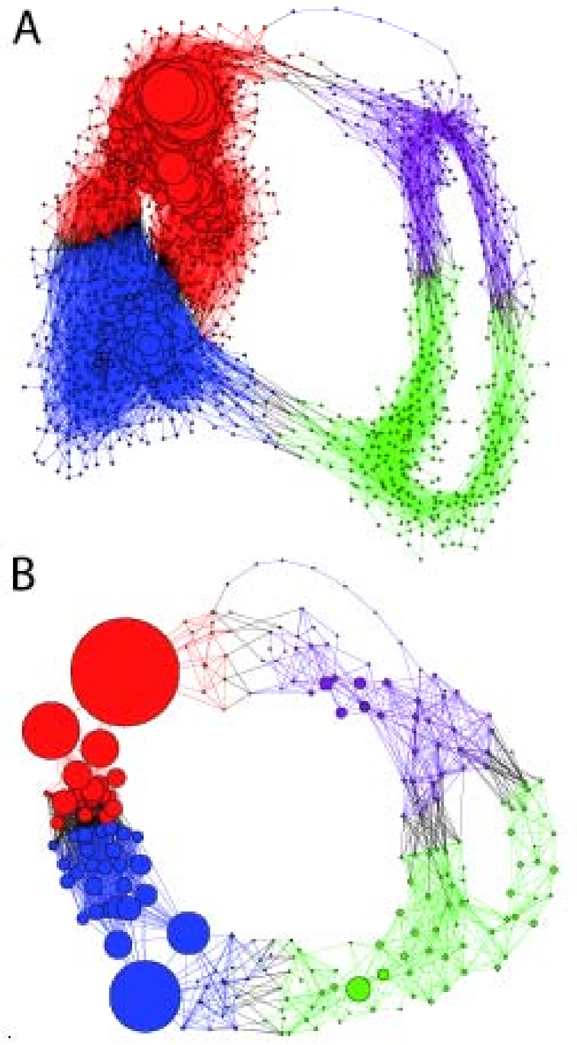


Coarse graining

Results of spectral coarse graining

(Gfeller and De Los Rios, PRL 2008)

A: Di-alanine folding network (N=1832). Node size is proportional to their weight (i.e. the number of times nodes have been visited in the simulation). The four different colors correspond to the clusters found in B: Coarse-grained network ($\tilde{N}$ = 227). Node size is proportional to the total weights of the groups. Colors correspond to the clusters in which the nodes of each group have been classified



Readings

- Santo Fortunato, Community detection in graphs, Physics Reports, 486, 75-174, 2010