Online Social Networks and Media

Community detection

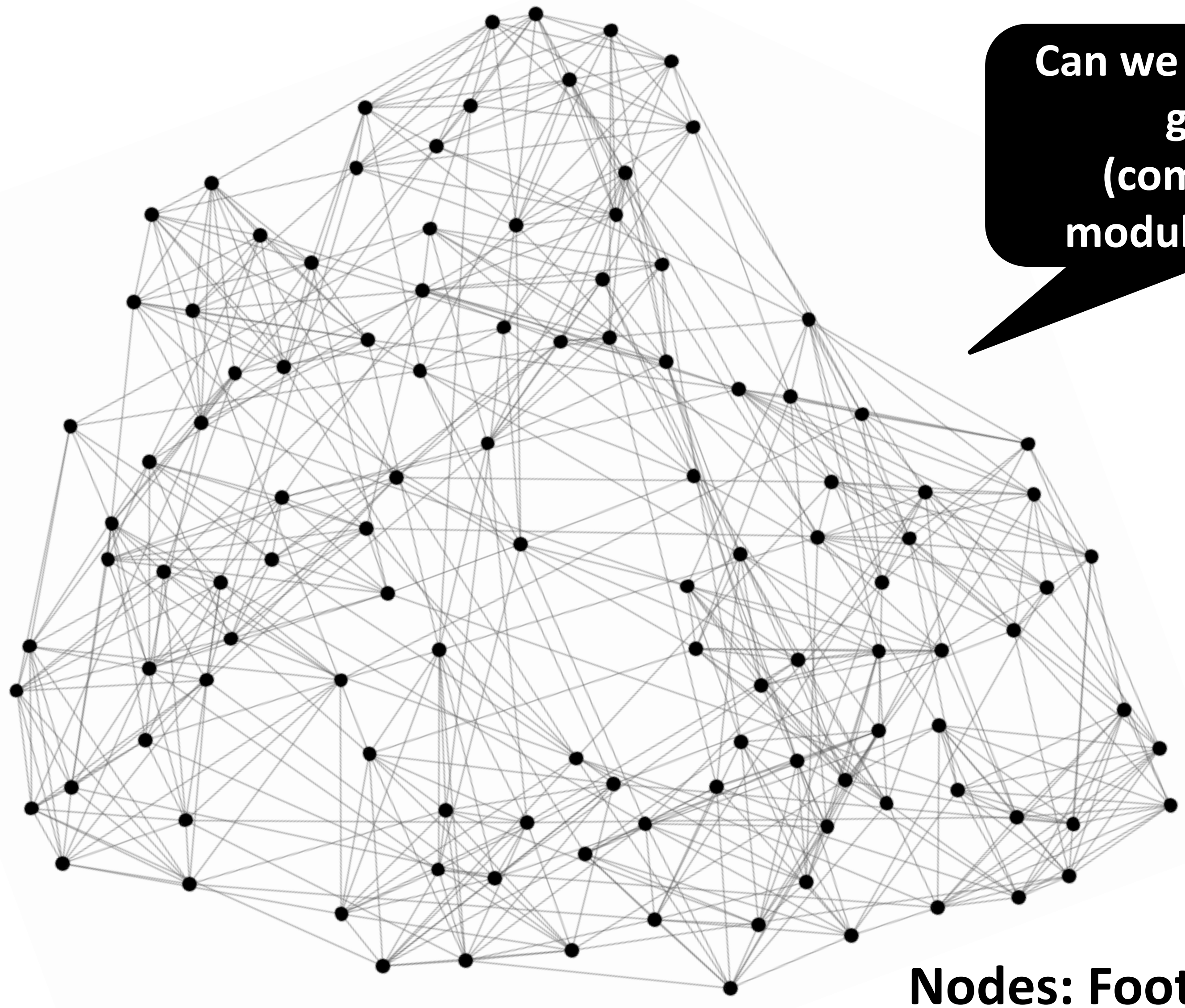
Introduction

Real networks are *not random graphs*

Communities
aka: groups, clusters, cohesive subgroups, modules

(informal) Definition: groups of vertices which probably share *common properties* and/or play *similar roles* within the graph

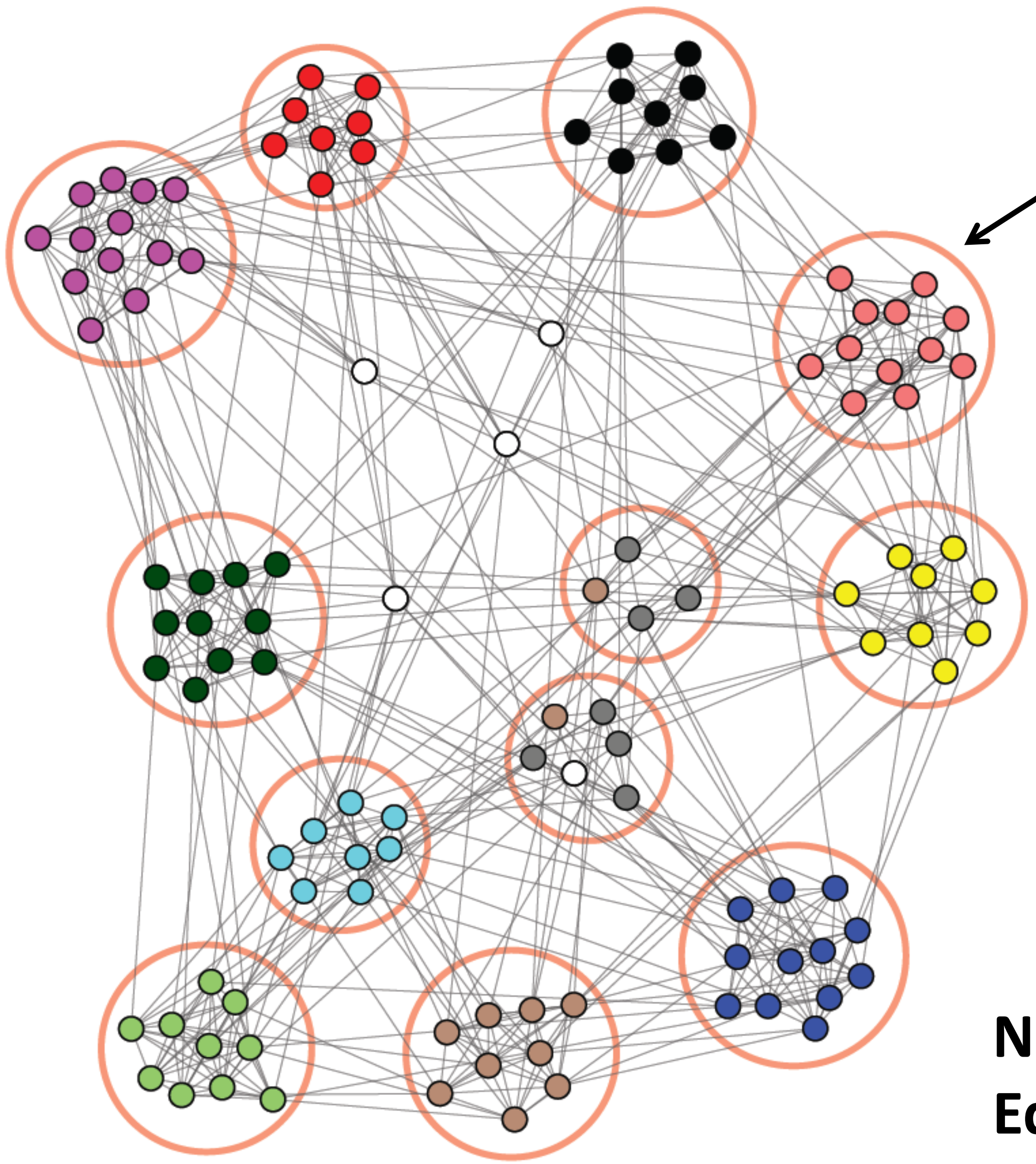
Some are *explicit (emic)* (e.g., Facebook (groups), LinkedIn (groups, associations), etc), we are interested in *implicit (etic)* ones



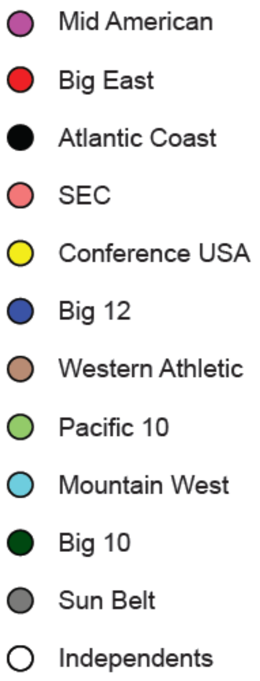
Can we identify node groups?
(communities, modules, clusters)

Nodes: Football Teams
Edges: Games played

NCAA Football Network

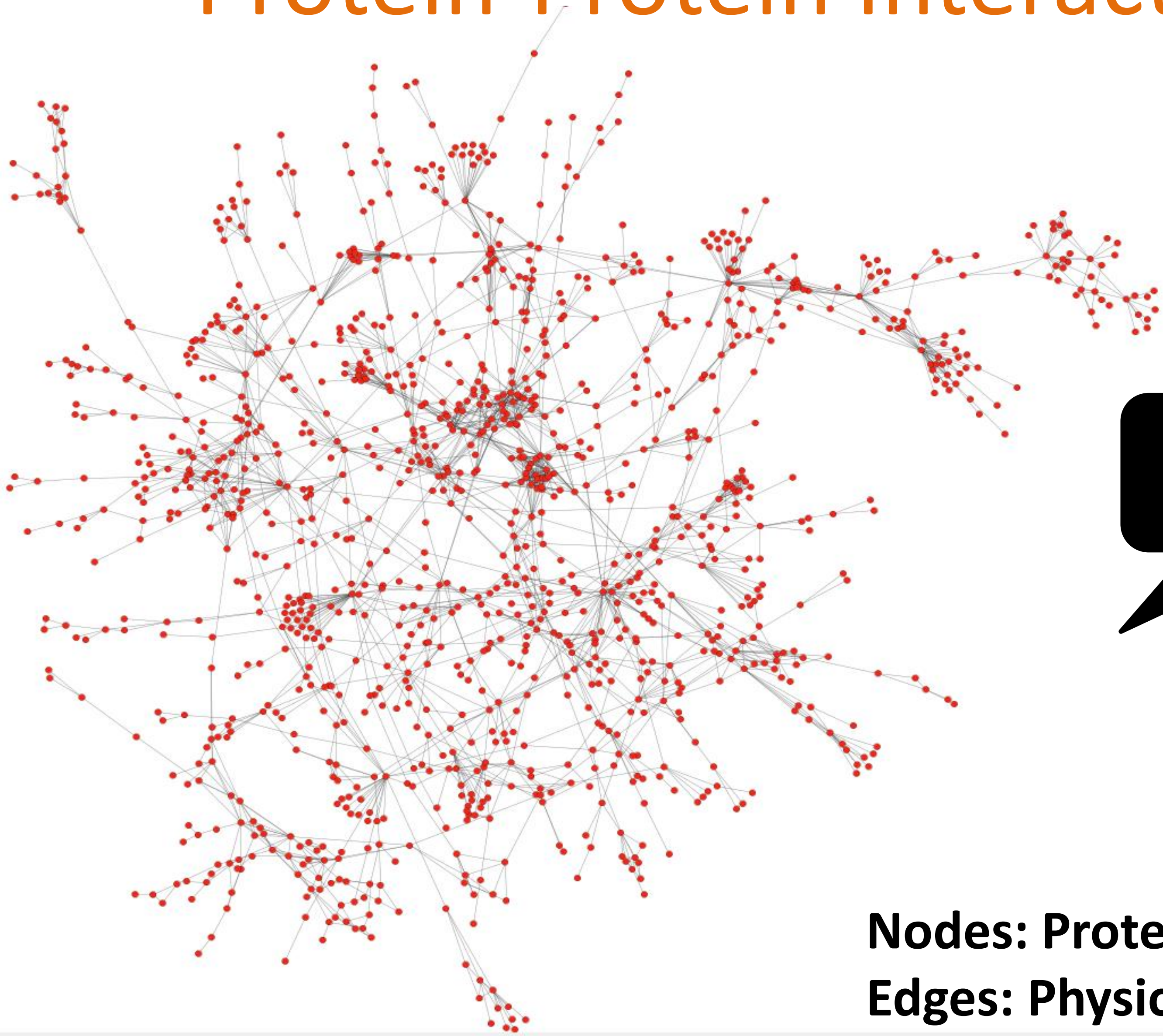


NCAA conferences



Nodes: Football Teams
Edges: Games played

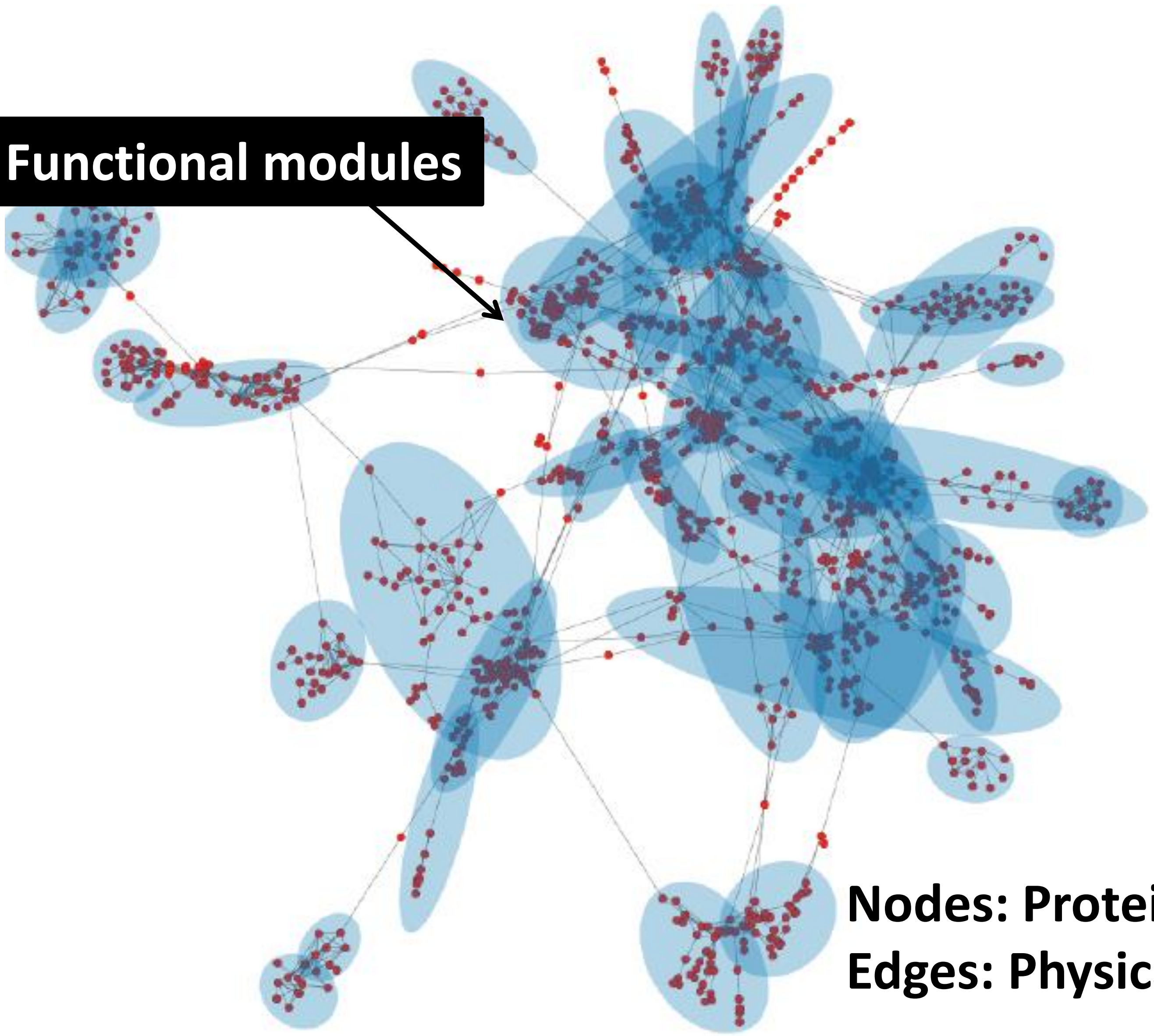
Protein-Protein Interactions



Can we identify functional modules?

Nodes: Proteins
Edges: Physical interactions

Protein-Protein Interactions

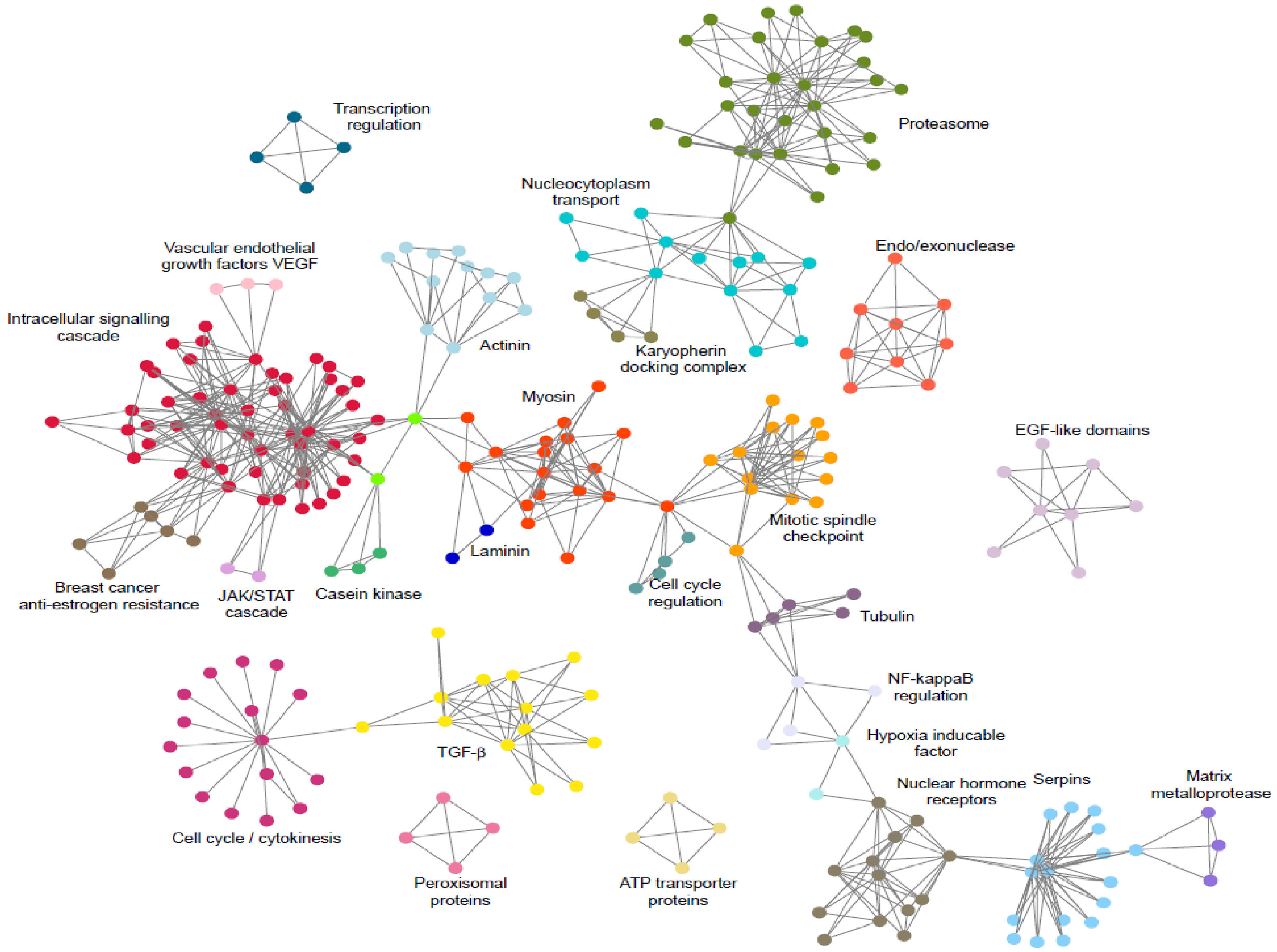


Functional modules

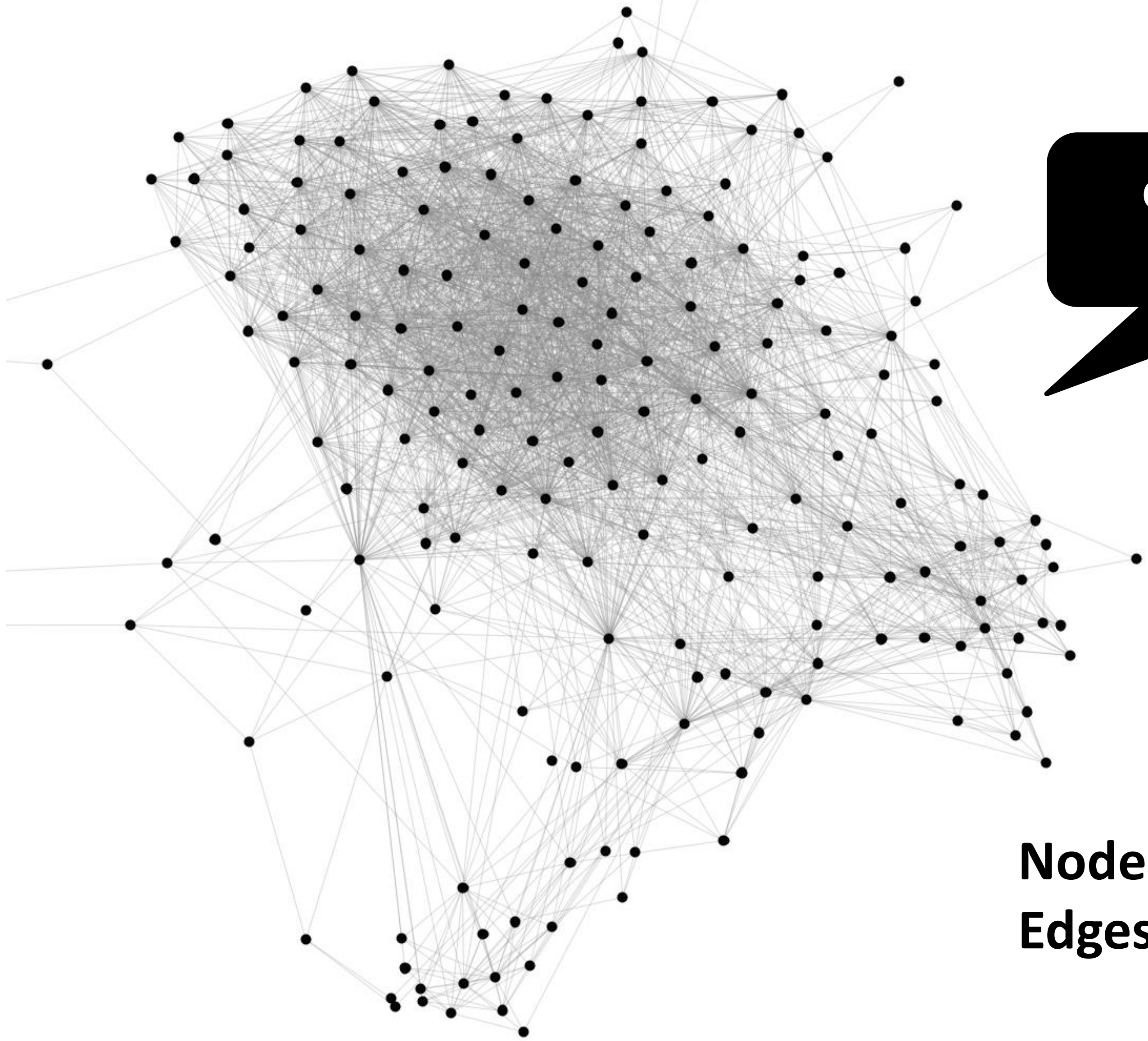
Nodes: Proteins

Edges: Physical interactions

Protein-Protein Interactions



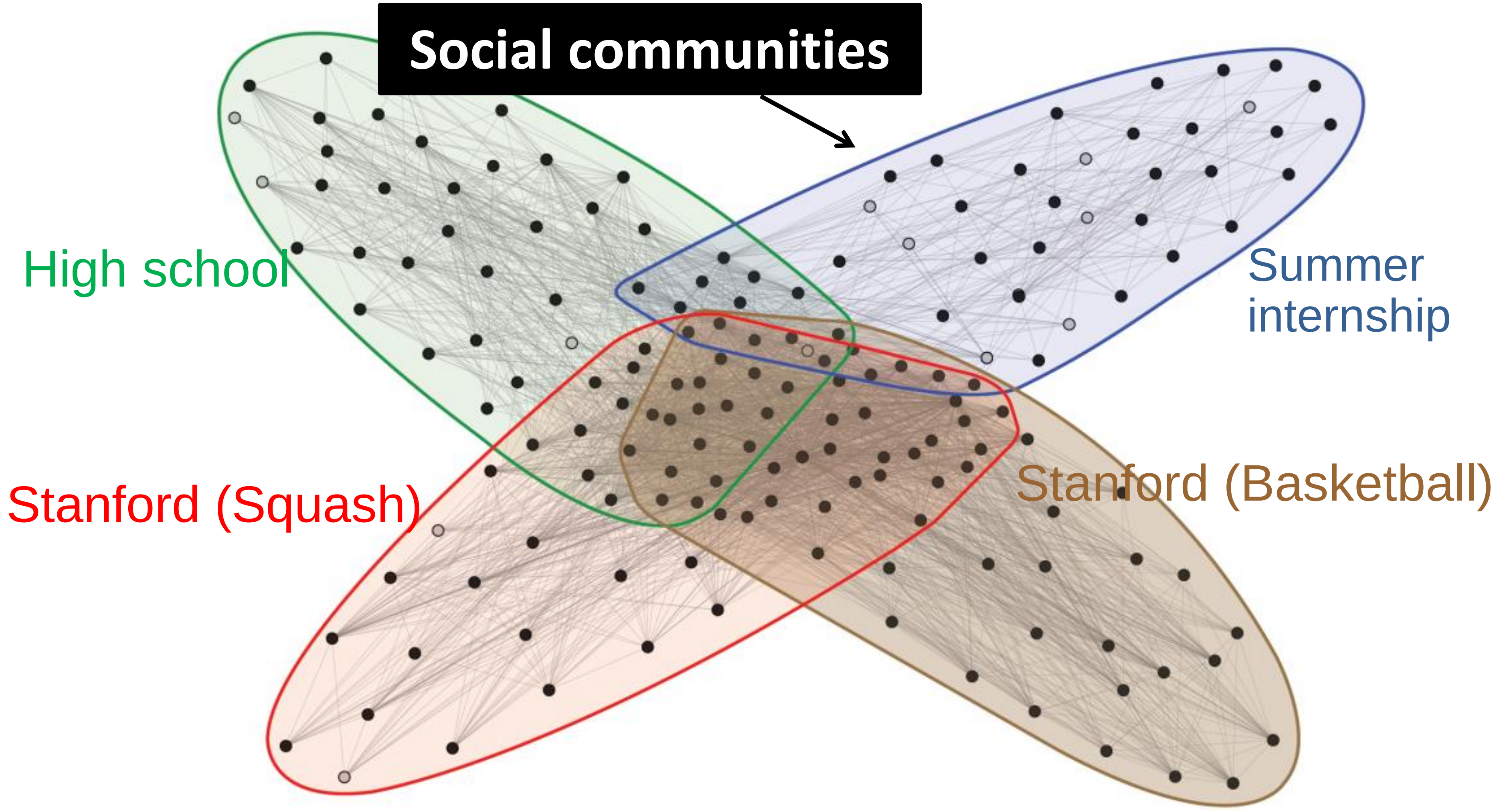
Facebook Network



Can we identify social communities?

Nodes: Facebook Users
Edges: Friendships

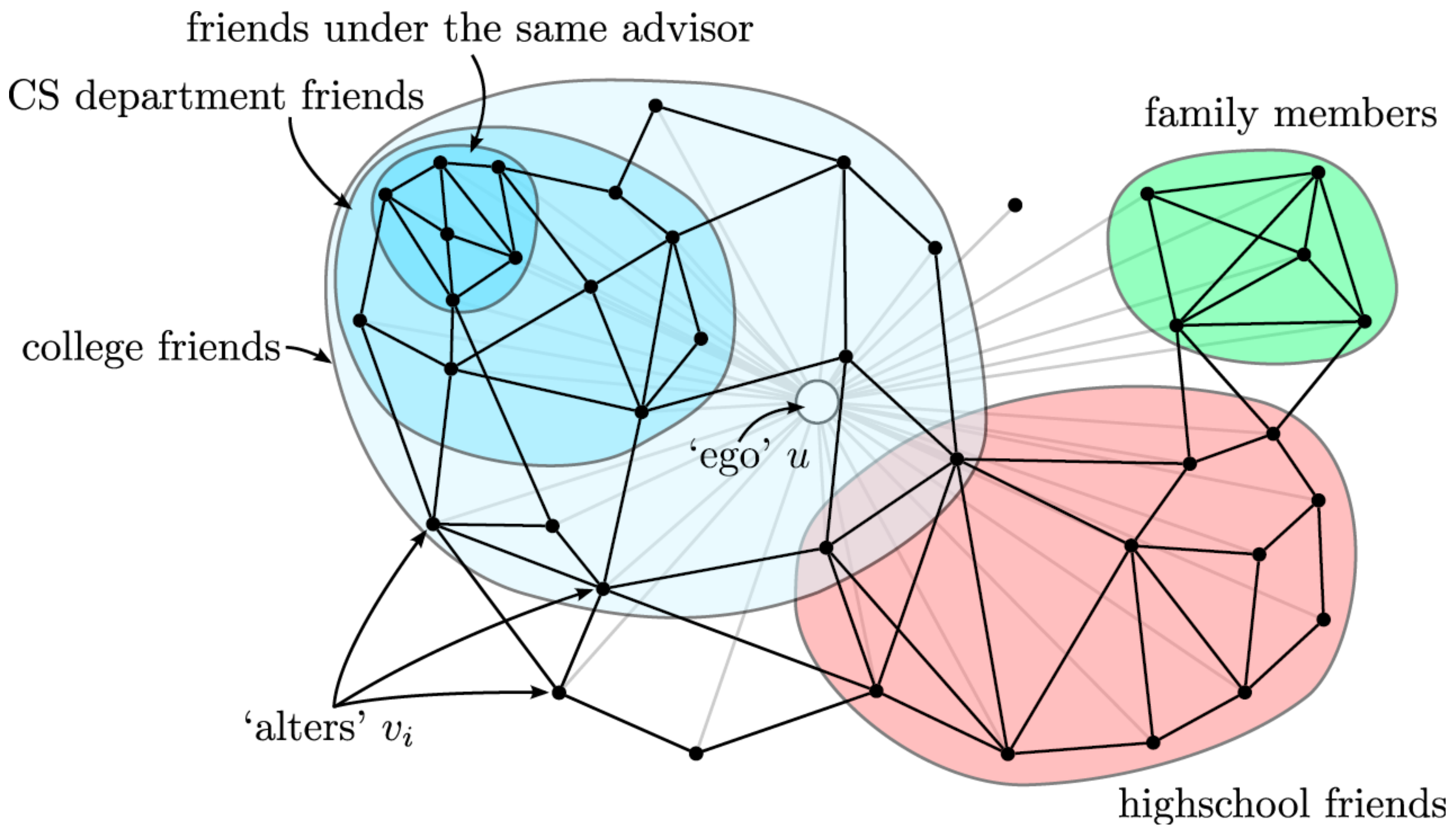
Facebook Network



Nodes: Facebook Users

Edges: Friendships

Twitter & Facebook



social circles, circles of trust

Outline

PART I

1. Introduction: what, why, types?
2. Cliques
3. Background: How it relates to “cluster analysis” (node/edge similarity)
4. Betweeness centrality
5. Modularity
6. *How to evaluate*

PART II

Cuts, Spectral clustering, Denser subgraphs, community evolution

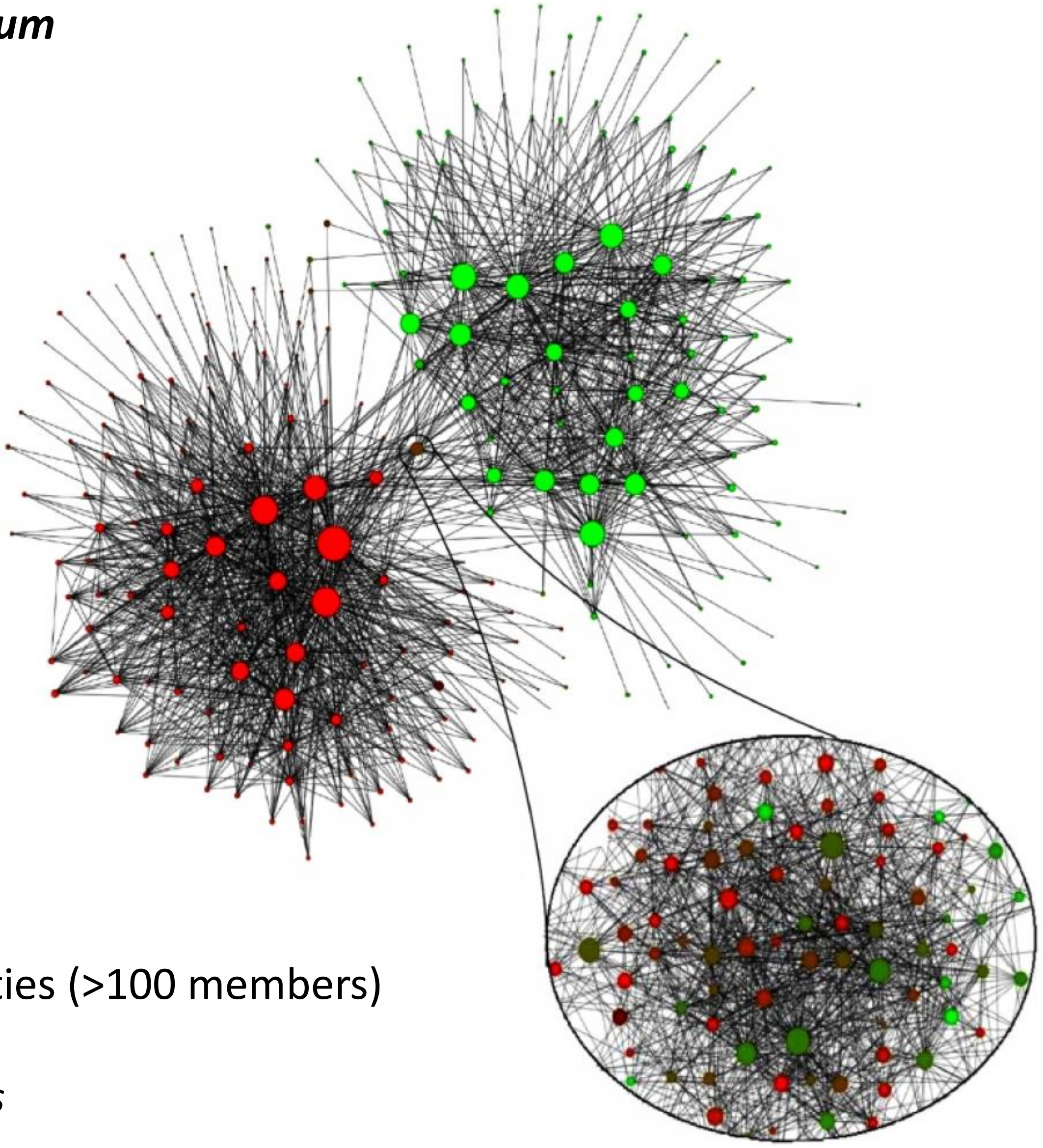
Why? (some applications)

- *Knowledge discovery*
- Groups based on common interests, behavior, etc (e.g., Canadians who call USA, readings tastes, etc)
  - *Recommendations, marketing*
- *Collective behavior* (observable at the group, not the individual level, local view is noisy and ad hoc)
- *Performance-wise* (partition a large graph into many machines, assigning web clients to web servers, routing in ad hoc networks, etc)
- *Classification of the nodes* (by identifying modules and their boundaries)
- Summary, visual *representation* of the graph

Example: communities in Belgium

59% Flemish, speaking Dutch 40% Walloons speaking French

Community structure in Belgium



2 million phone users

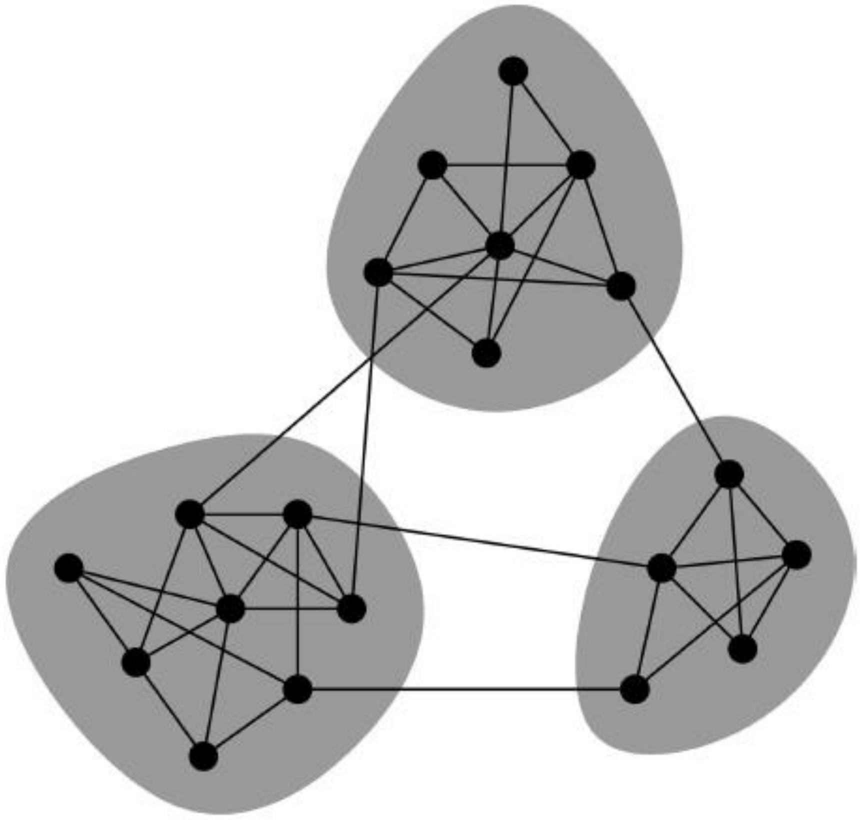
Nodes correspond to communities (>100 members)

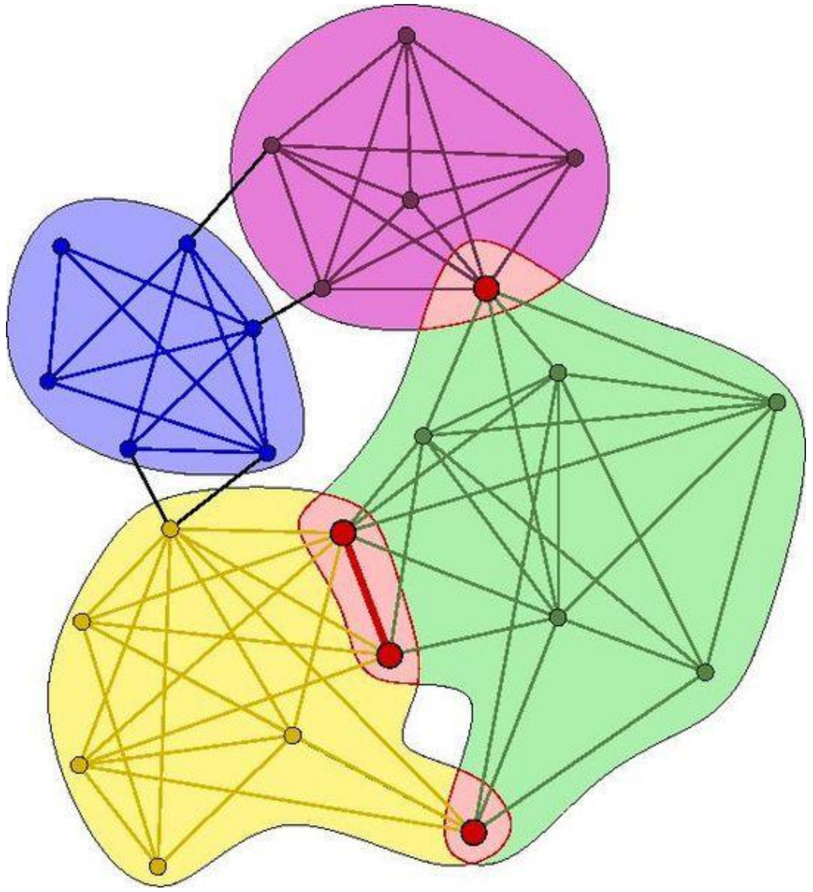
Red French, Green Dutch

Connecting community *Brussels*

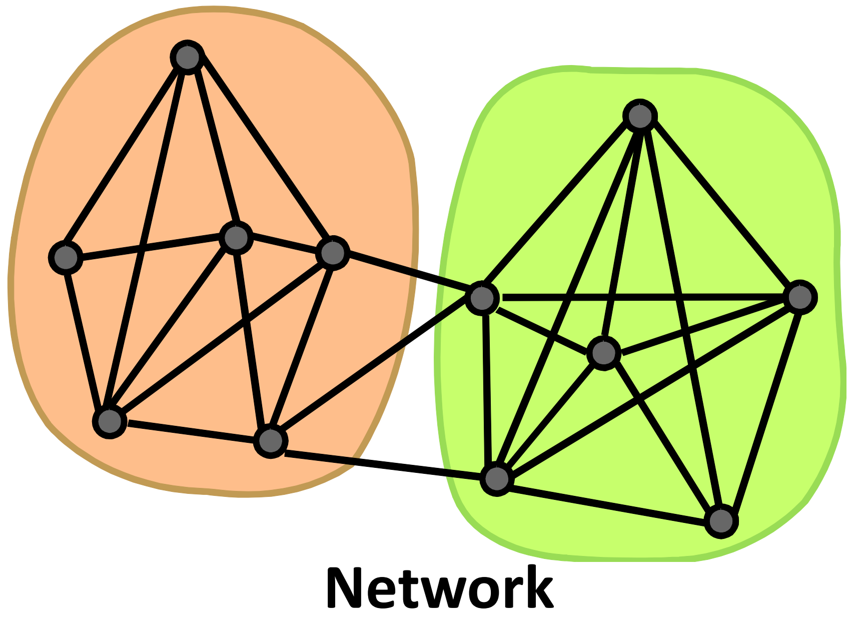
Community Types

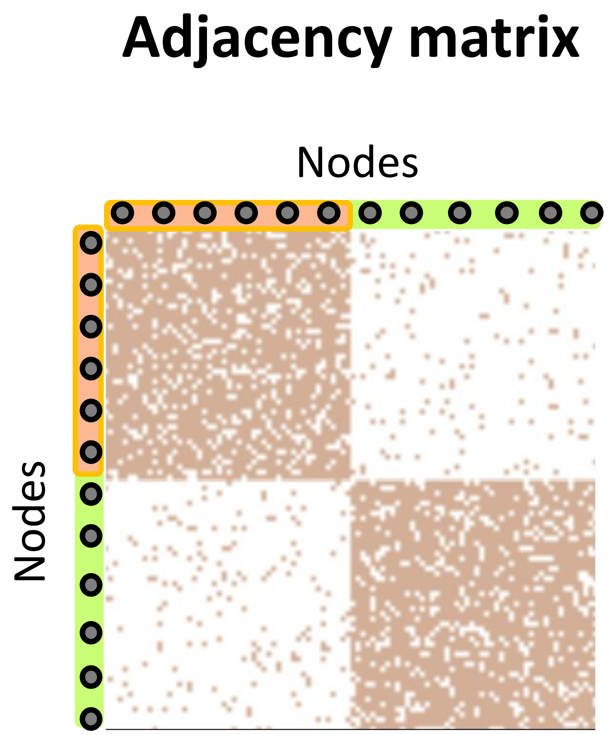
Non-overlapping vs. overlapping communities





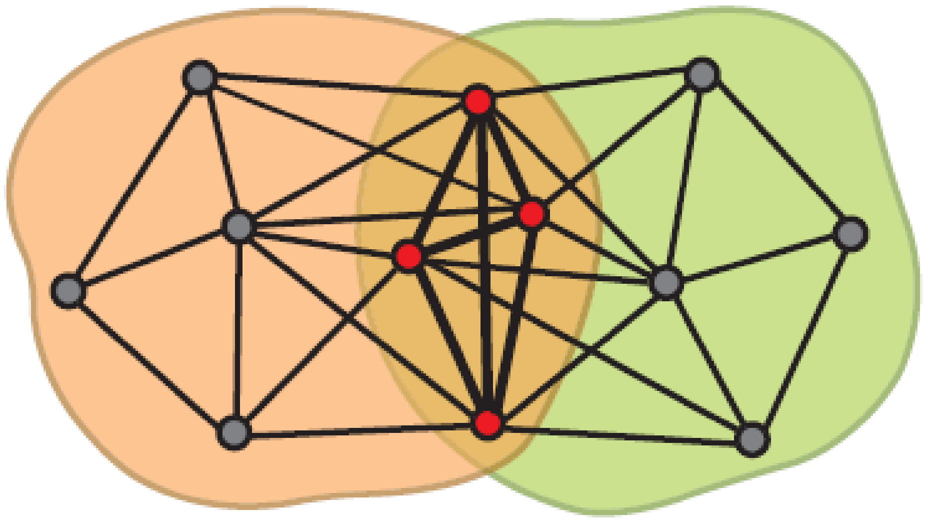
Non-overlapping Communities

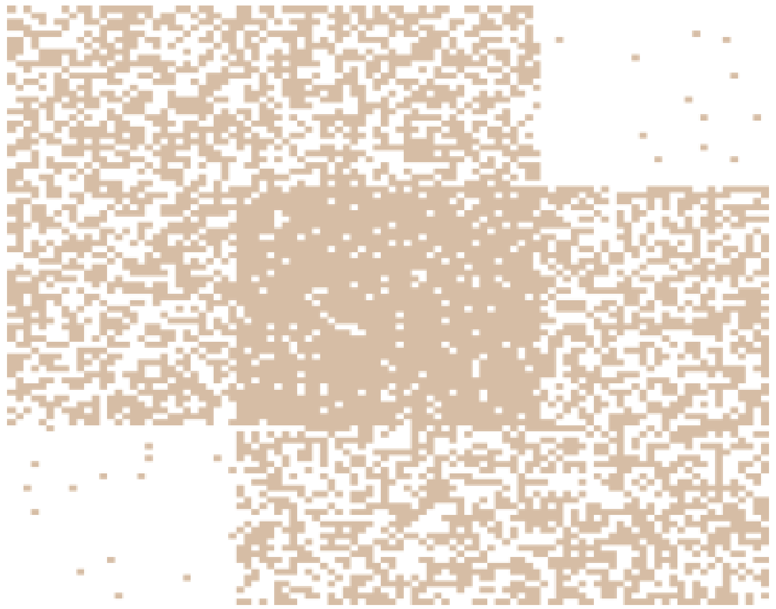




Overlapping Communities

What is the structure of community overlaps:
Edge density in the overlaps is higher!

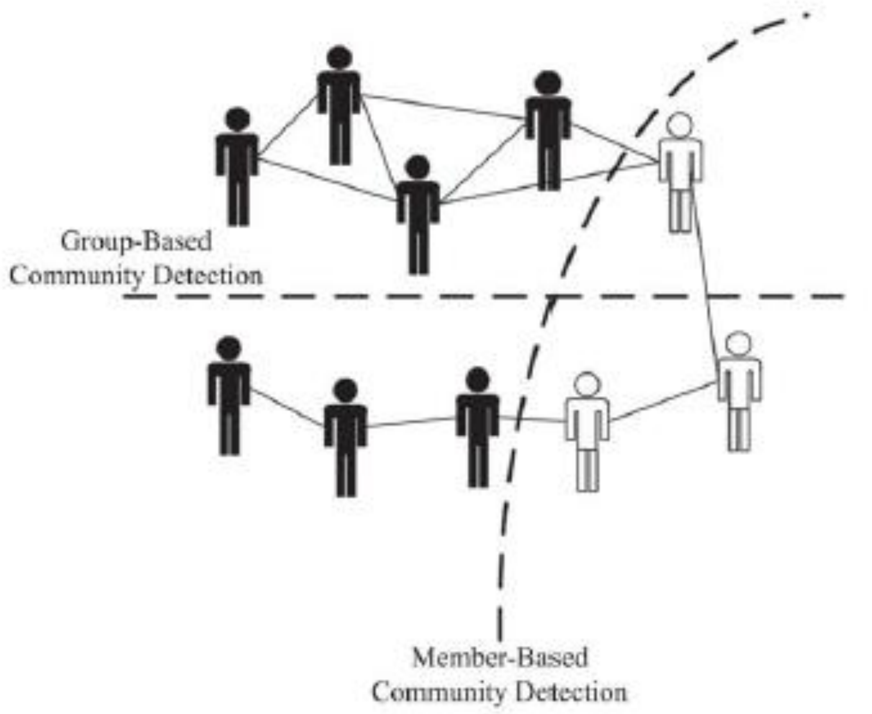




Communities as “tiles”

Community Types

Member-based (local) vs. group-based



Community Detection

Given a graph G(V, E), find subsets C_i of V, such that $\bigcup_i C_i \subseteq V$

Assumptions

- Undirected graphs
- Edges may have
  - weights, (easily extended)
  - labels
  - content or attributes shared by individuals (in the same location, of the same gender, etc)
- Nodes may have labels, attributed, or labeled graphs

Multipartite graphs – e.g., affiliation networks, citation networks, customers-products: reduced to unipartited projections of each vertex class

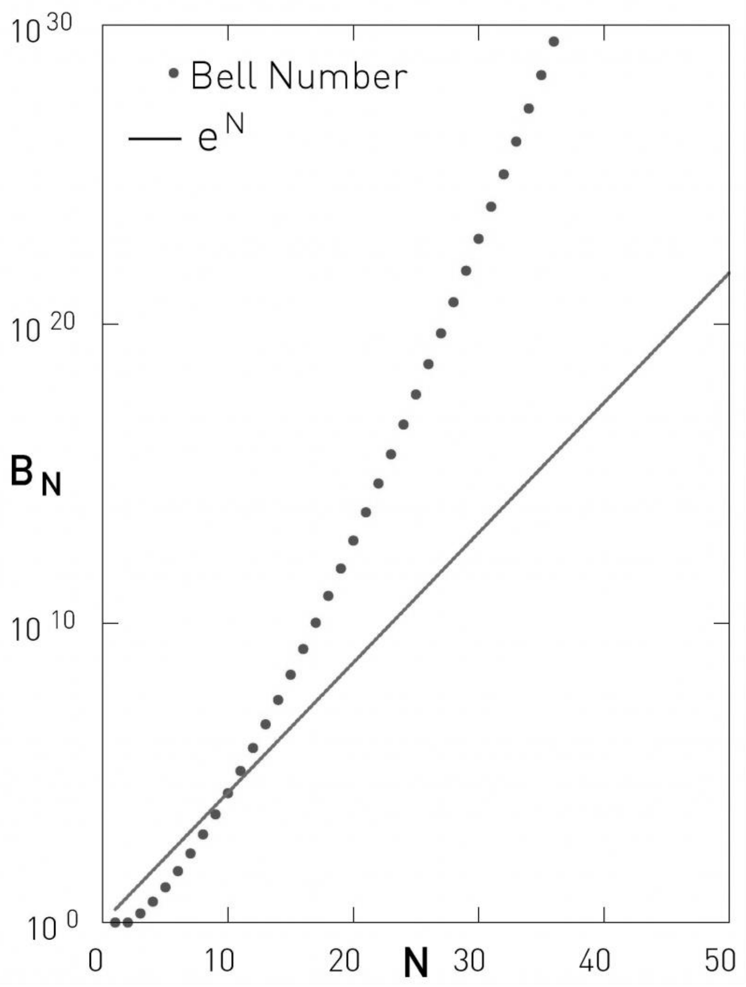
Hardness

Bell Number

Number of all possible partitions of *N* nodes

$$B_N = \frac{1}{e}\sum_{j=0}^{\infty}\frac{j^N}{j!}$$

1040 partitions for a network of size *N*=50



Community Detection

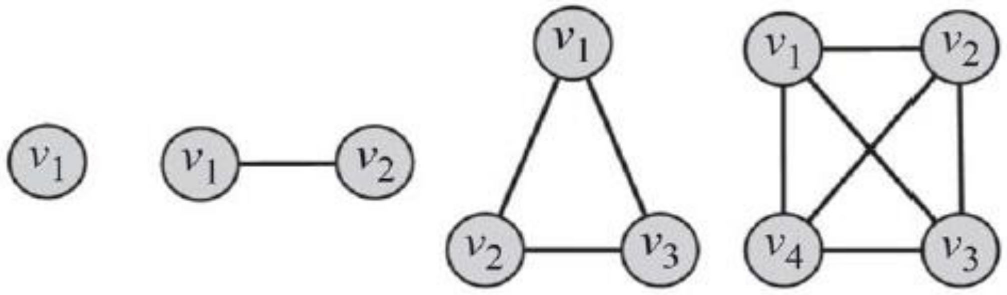
We will see three approaches

- Node *degree* (familiarity)
  - cliques
- Node *similarity*
  - cluster
- Node *reachability*
  - betweeness

Cliques (degree similarity)

Clique: a maximum *complete subgraph* in which all pairs of vertices are connected by an edge.

A *clique of size k* is a subgraph of *k* vertices where the degree of all vertices in the induced subgraph is *k -1* .



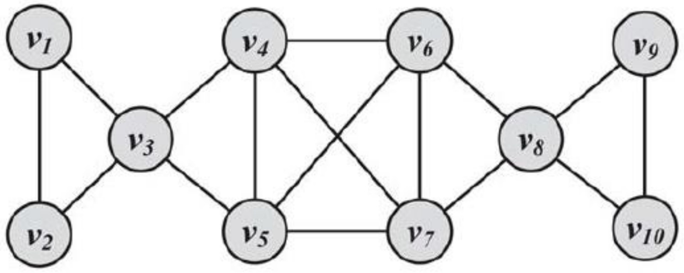
✓ Cliques vs complete graphs

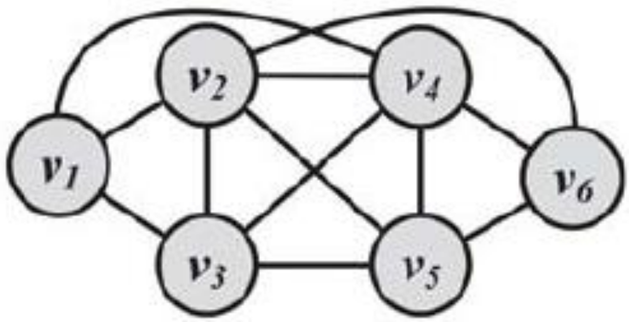
Cliques (degree similarity)

Search for:

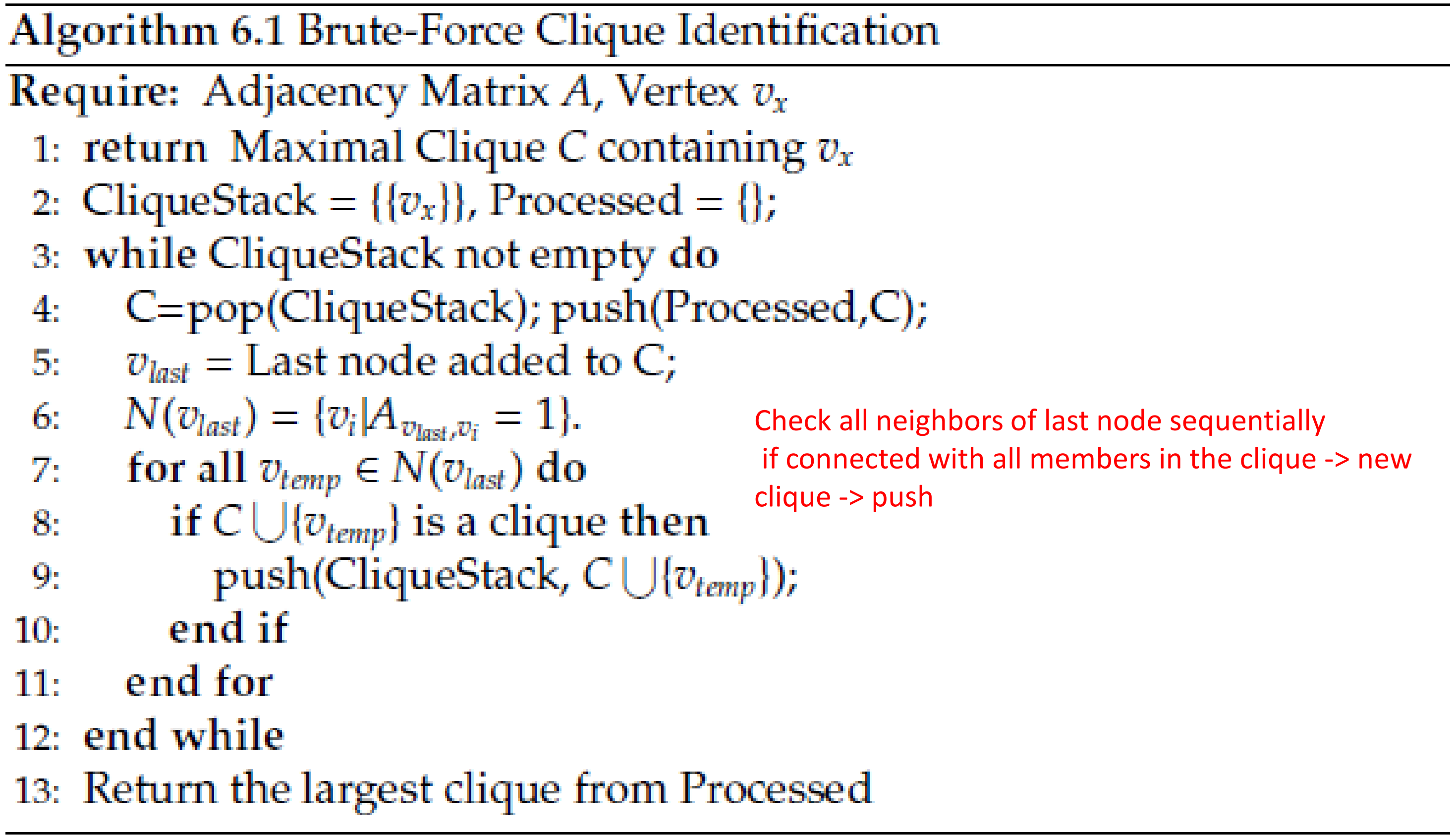
- the *maximum clique* (the one with the largest number of vertices) or
- all *maximal cliques* (cliques that are not subgraphs of a larger clique; i.e., cannot be expanded further).

Both problems are NP-hard, as is verifying whether a graph contains a clique larger than size *k*.





Cliques



Algorithm 6.1 Brute-Force Clique Identification

```
Require: Adjacency Matrix $A$, Vertex $v_x$
 1: return Maximal Clique C containing $v_x$
 2: CliqueStack = $\{\{v_x\}\}$, Processed = $\{\}$;
 3: while CliqueStack not empty do
 4:     C=pop(CliqueStack); push(Processed,C);
 5:     $v_{last}$ = Last node added to C;
 6:     $N(v_{last}) = \{v_i | A_{v_{last}, v_i} = 1\}$.
 7:     for all $v_{temp} \in N(v_{last})$ do
 8:         if $C \bigcup \{v_{temp}\}$ is a clique then
 9:             push(CliqueStack, $C \bigcup \{v_{temp}\}$);
10:         end if
11:     end for
12: end while
13: Return the largest clique from Processed
```

Check all neighbors of last node sequentially
if connected with all members in the clique -> new clique -> push

Enumerate all cliques (in alphabetical order)

Checks all permutations!

For (complete graph) 100 vertices, 2^{99}- 1 different cliques

Cliques

Pruning

- Prune all vertices (and incident edges) with degrees less than k - 1.
- Effective due to the power-law distribution of vertex degrees

"Exact cliques" are *rarely observed* in real networks.

E.g., a clique of 1,000 vertices has (999x1000)/2 = 499,500 edges.

- A single edge removal results in a subgraph that is no longer a clique.
- That represents less than 0.0002% of the edges

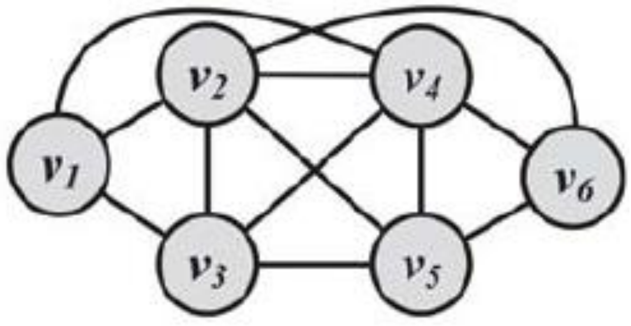
Relaxing Cliques

All vertices have *a minimum degree* but not necessarily *k* -1

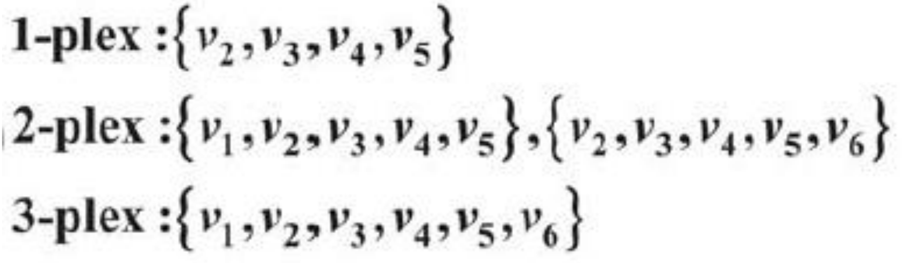
k-plex

For a set of vertices V_0, for all u, $d_u \geq |V_0| - k$
where d_u is the degree of v in the induced subgraph

What is k for a clique?



Maximal



1-plex : $\{v_2, v_3, v_4, v_5\}$

2-plex : $\{v_1, v_2, v_3, v_4, v_5\}, \{v_2, v_3, v_4, v_5, v_6\}$

3-plex : $\{v_1, v_2, v_3, v_4, v_5, v_6\}$

Relaxing Cliques

Clique $$\forall i \in C, k_i^{\text{int}} = N_C$$

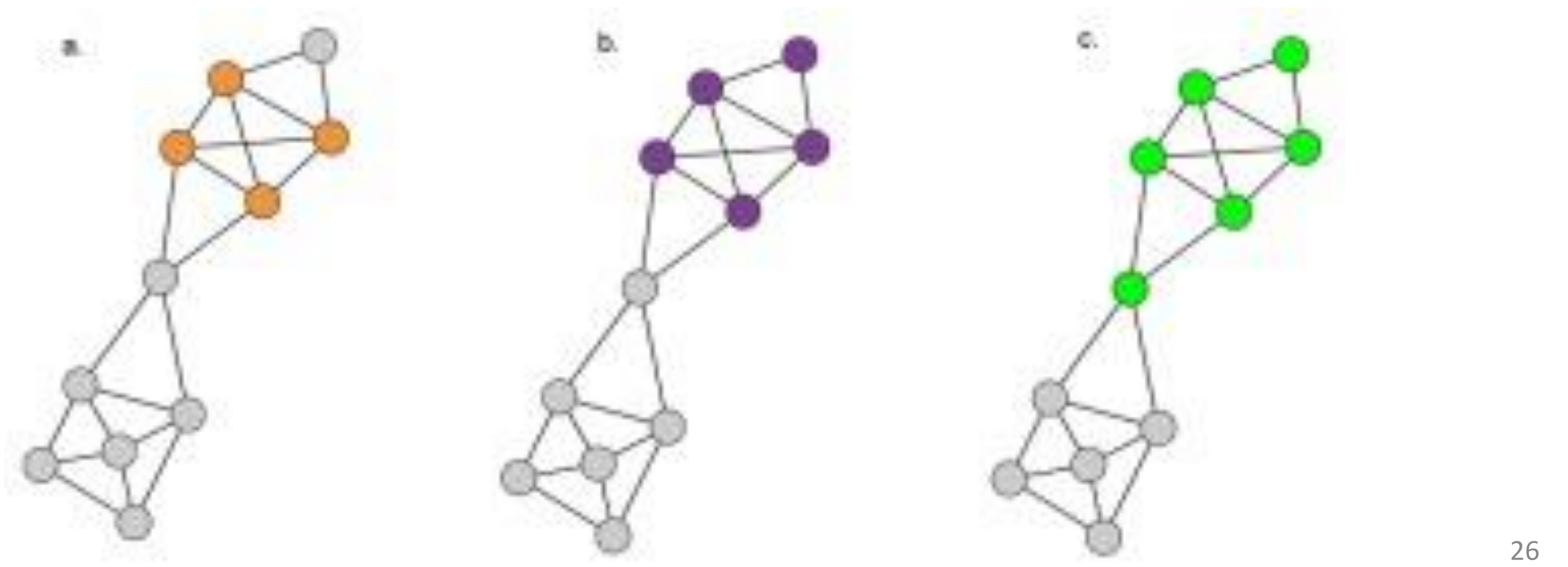
Strong community $$\forall i \in C, k_i^{\text{int}} > k_i^{\text{ext}}$$

Weak community $$\sum_{i \in C} k_i^{\text{int}} > \sum_{i \in C} k_i^{\text{ext}}$$

k_i^{int} Inside C

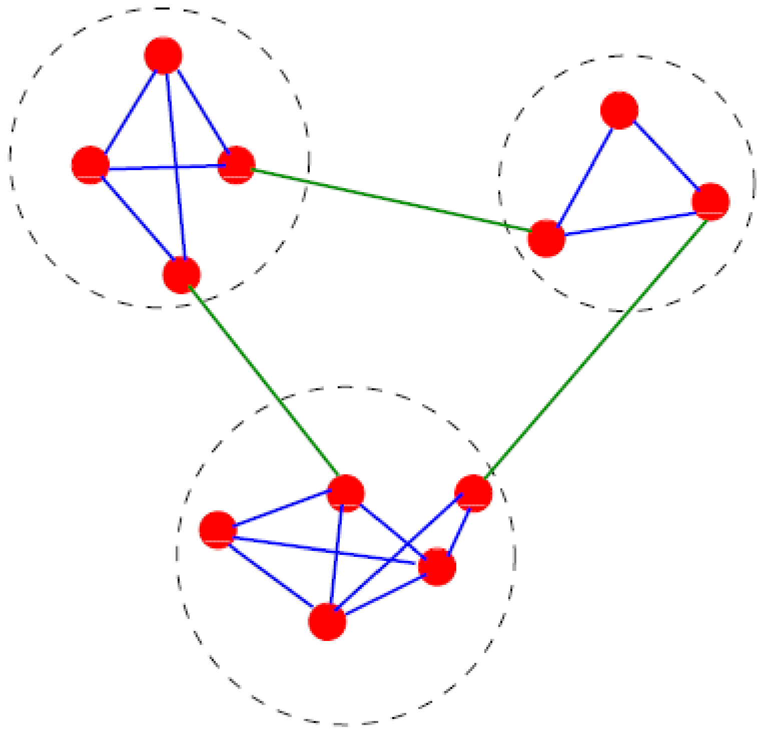
k_i^{ext} Outside C

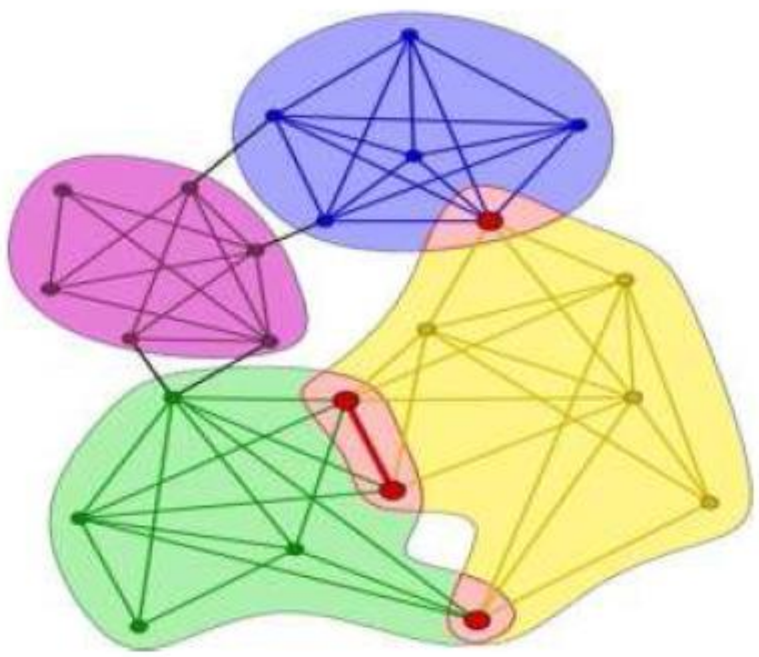
Where *k* stands for degree



Clique Percolation Method (CPM): Using cliques as seeds

Assumption: communities are formed from a set of cliques and edges that connect these cliques.





$k = 4$

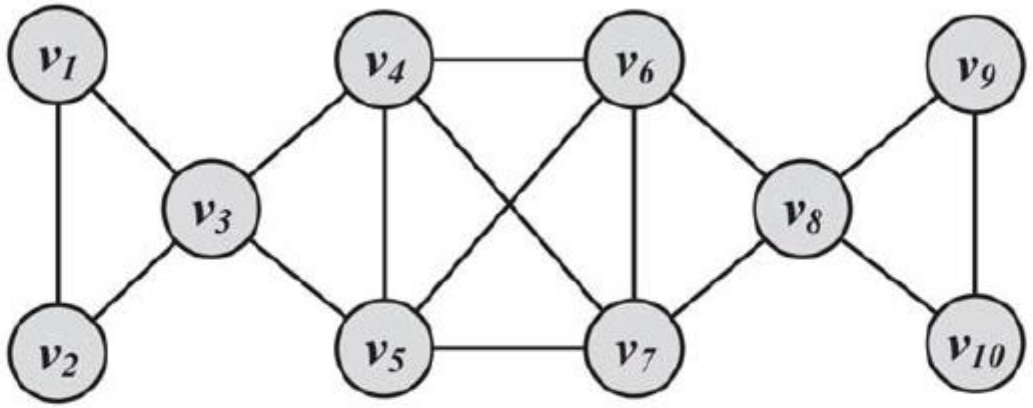
Clique Percolation Method (CPM): Using cliques as seeds

```
Algorithm 6.2 Clique Percolation Method (CPM)
Require: parameter $k$
1: return Overlapping Communities
2: $Cliques_k$ = find all cliques of size $k$
3: Construct clique graph $G(V, E)$, where $|V| = |Cliques_k|$
4: $E = \{e_{ij} \mid$ clique $i$ and clique $j$ share $k - 1$ nodes$\}$
5: Return all connected components of $G$
```

1. Given *k*, find all cliques of size *k*.
2. Create graph (clique graph) where all cliques are vertices, and two cliques that share *k* - 1 vertices are connected via an edge.
3. Communities are the connected components of this graph.

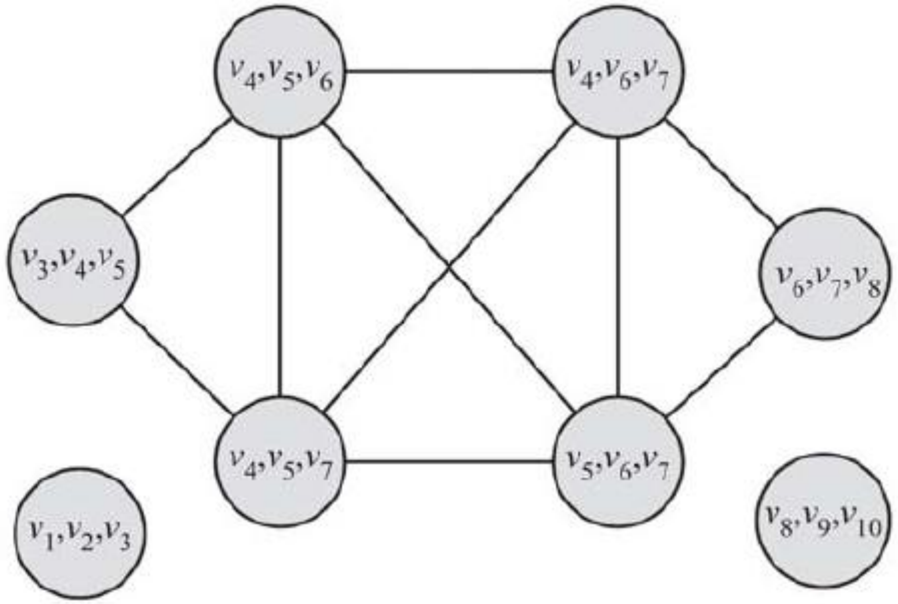
Clique Percolation Method (CPM): Using cliques as seeds

Input graph, let k = 3



Clique Percolation Method (CPM): Using cliques as seeds

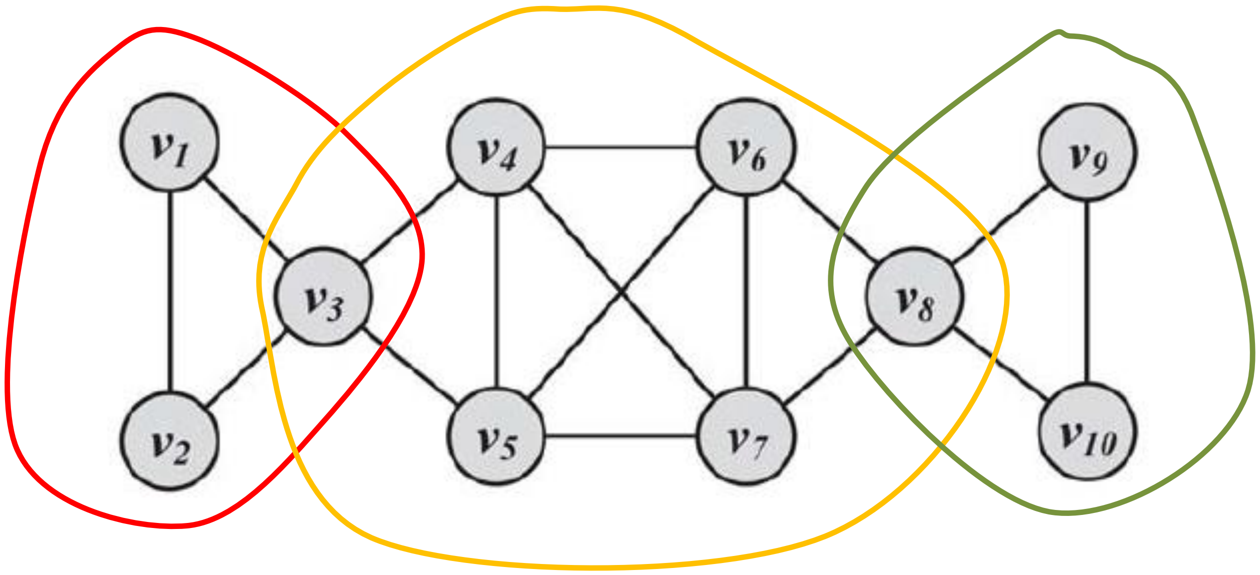
Clique graph for *k* = 3



(v1, v2, ,v3), (v8, v9, v10), and (v3, v4, v5, v6, v7, v8)

Clique Percolation Method (CPM): Using cliques as seeds

Result



(v1, v2, ,v3), (v8, v9, v10), and (v3, v4, v5, v6, v7, v8)

Note: the example protein network was detected using a CPM algorithm

Clique Percolation Method (CPM): Using cliques as seeds (terminology)

Two *k*-cliques are adjacent, if they share *k* - 1 vertices.

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.

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.

Clique Percolation Method (CPM): Using cliques as seeds

- A *k*-clique community is identified by making a *k*-clique "roll" over adjacent *k*-cliques, where rolling means rotating a *k*-clique about the *k*-1 vertices it shares with any adjacent *k*-clique.

- By construction, *overlapping* communities

- There may be vertices belonging to nonadjacent *k*-cliques, which could be reached by different paths and end up in different clusters. There are also vertices that cannot be reached by any *k*-clique

- Instead of $k = 3$, maximal cliques?

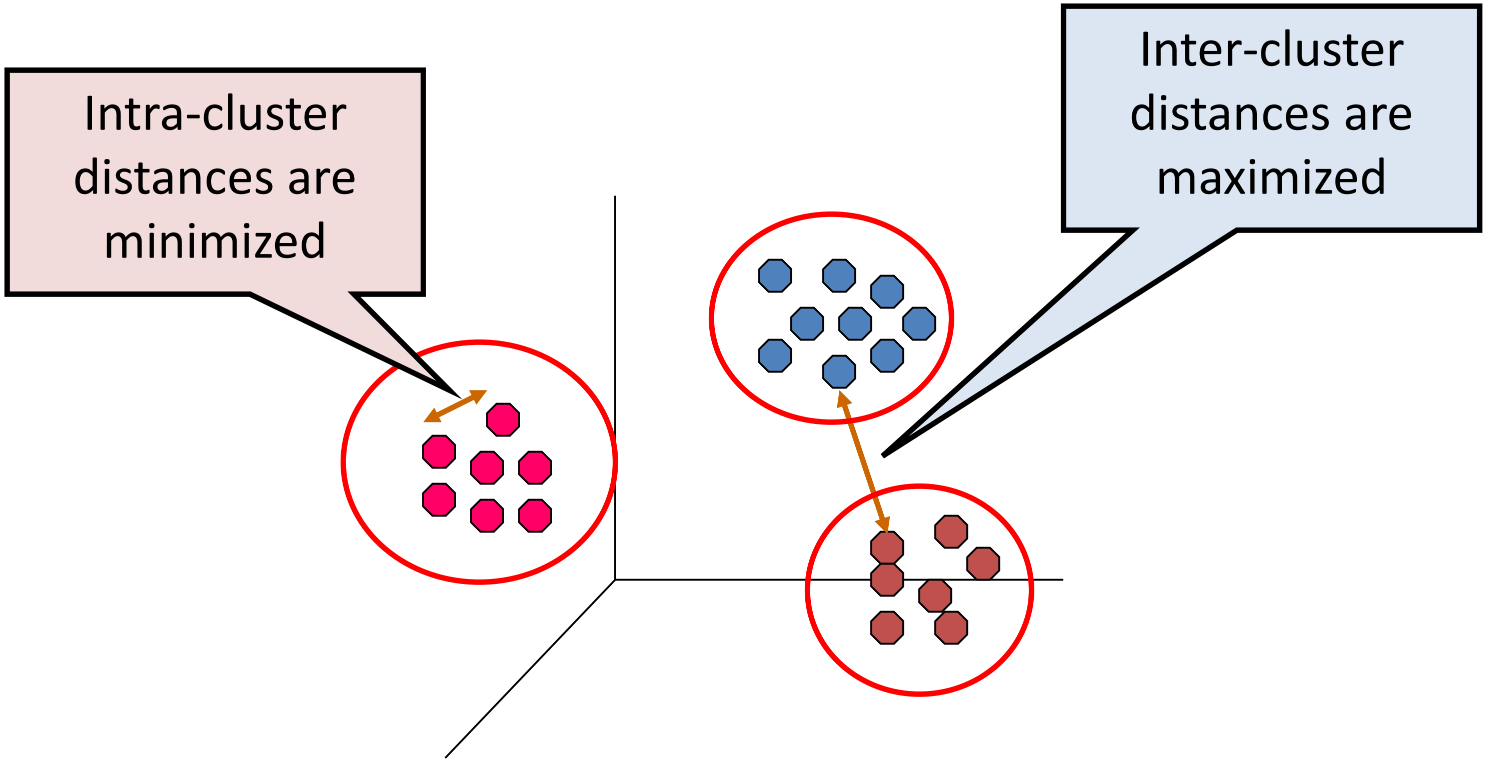
- Theoretical complexity grows exponential with size, but *efficient on sparse graphs*

Outline

PART I

What is Cluster Analysis?

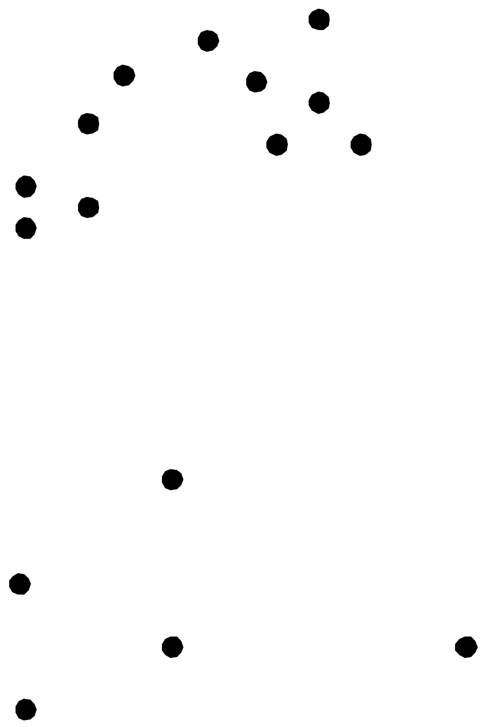
Finding groups of objects such that the objects in a group are similar (or related) to one another and different from (or unrelated to) the objects in other groups

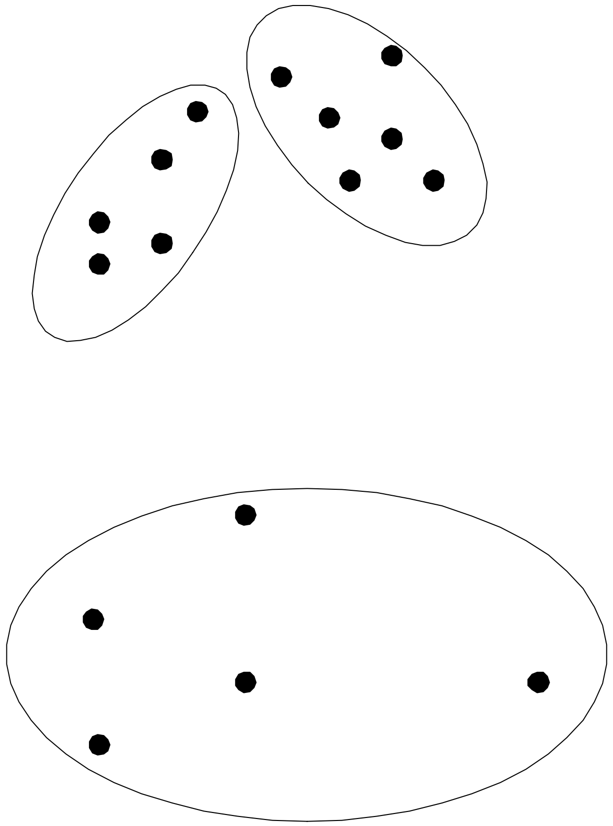

Intra-cluster distances are minimized
Inter-cluster distances are maximized

Types of Clustering

- Important distinction between hierarchical and partitional sets of clusters
- Partitional Clustering
  - Division of data objects into subsets (clusters)
  - Assumes that the *number of clusters is given*
- Hierarchical clustering
  - A set of *nested clusters* organized as a hierarchical tree

Partitional Clustering


Original Points

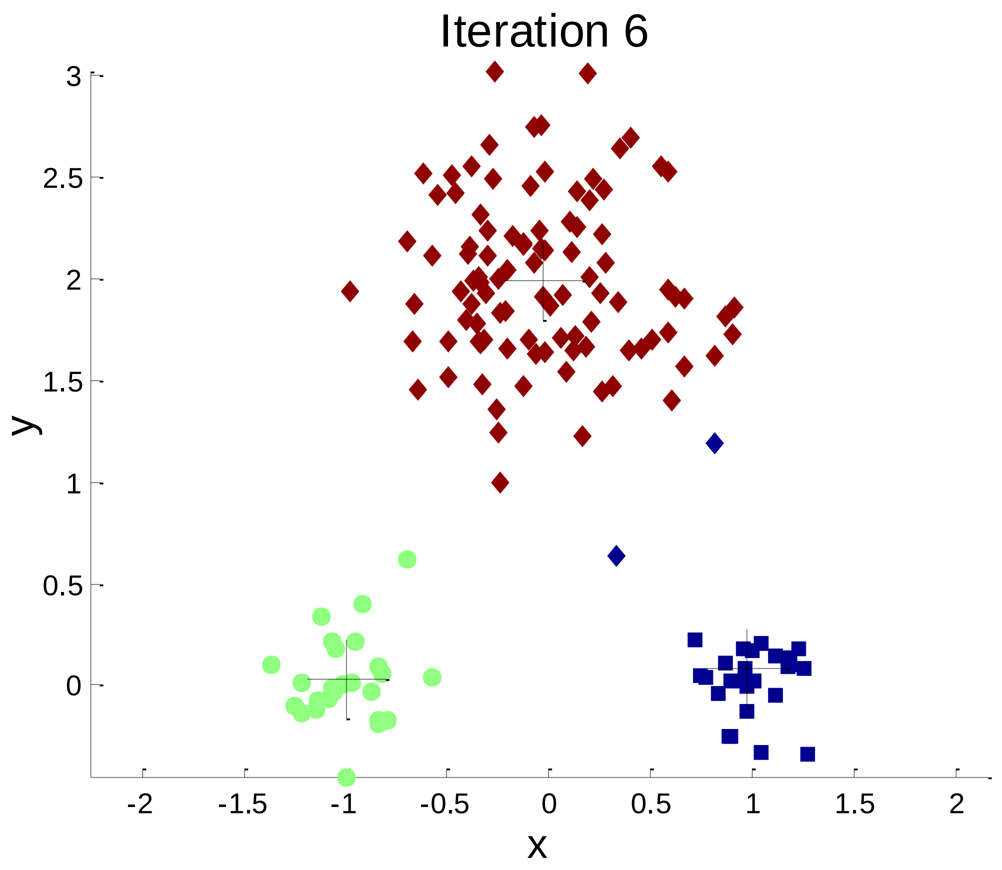

A Partitional Clustering

Example Partitioning: K-means Clustering

```
1: Select $K$ points as the initial centroids.
2: **repeat**
3:     Form $K$ clusters by assigning all points to the closest centroid.
4:     Recompute the centroid of each cluster.
5: **until** The centroids don't change
```

- Input: Number of clusters, K
- Each cluster is associated with a *centroid* (center point)
- Each point is assigned to the cluster with the closest centroid

Example



K-means Clustering

- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- 'Closeness' is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means *will converge* for common similarity measures mentioned above.
  - Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to 'Until relatively few points change clusters'
- Complexity is O(n * K * I * d)
  - n = number of points, K = number of clusters,
    I = number of iterations, d = number of attributes

K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

  - *x* is a data point in cluster C_i and m_i is the representative point for cluster C_i
    - can show that m_i corresponds to the center (mean) of the cluster
  - Given two clusters, we can choose the one with the smallest error
  - One easy way to reduce SSE is to increase K, the number of clusters
    - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

Vertex similarity

- Define similarity between two vertices
- Place similar vertices in the same cluster

- Use traditional *cluster analysis*

Vertex similarity

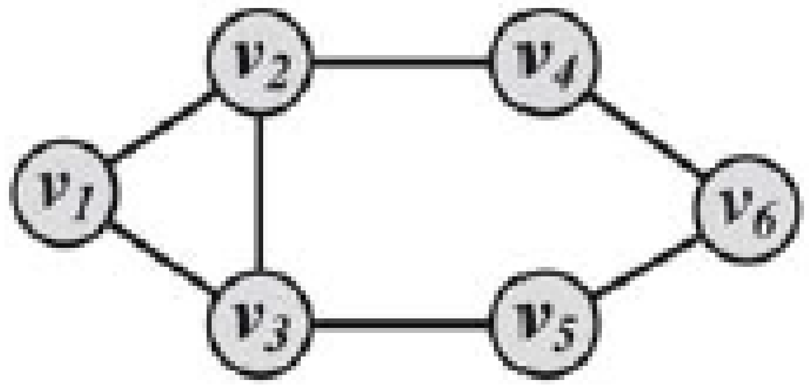
- Structural equivalence: based on the overlap between their neighborhoods

$$\sigma(v_i, v_j) = |N(v_i) \cap N(v_j)|$$

- Normalized to [0, 1], e.g.,

$$\sigma_{\text{Jaccard}}(v_i, v_j) = \frac{|N(v_i) \cap N(v_j)|}{|N(v_i) \cup N(v_j)|}$$

Vertex similarity



$$\sigma_{\text{Jaccard}}(v_2, v_5) = \frac{|\{v_1, v_3, v_4\} \cap \{v_3, v_6\}|}{|\{v_1, v_3, v_4, v_6\}|} = 0.25$$

Other definitions of vertex similarity

Use the adjacency matrix A,

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

Other definitions of vertex similarity

If we map vertices u, v to n-dimensional points A, B in the Euclidean space,

$$d^E_{AB} = \sum_{k=1}^{n} \sqrt{(a_k - b_k)^2}$$

$$d^M_{AB} = \sum_{k=1}^{n} |a_k - b_k|$$

$$d^\infty_{AB} = \max_{k \in [1,n]} |a_k - b_k|$$

$$\rho_{AB} = \arccos \frac{\mathbf{a} \cdot \mathbf{b}}{\sqrt{\sum_{k=1}^{n} a_k^2} \sqrt{\sum_{k=1}^{n} b_k^2}}$$

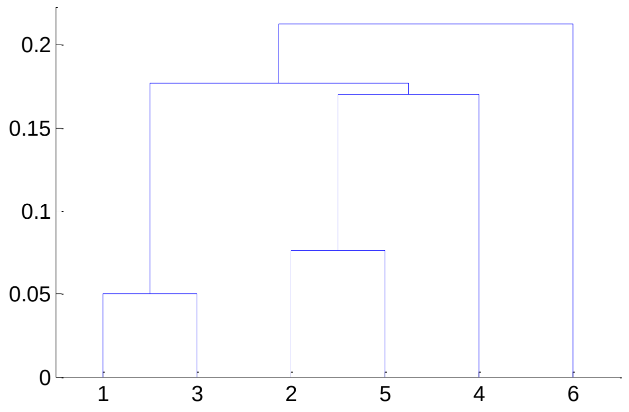
Other definitions of vertex similarity

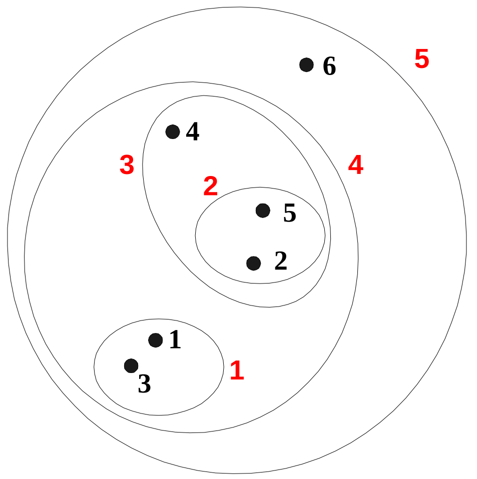
Many more – we shall revisit this issue when we talk *graph embeddings*

Also useful when there are *attributes* associated with nodes or edges to combine distances

Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits





Hierarchical Clustering

- Two main types of hierarchical clustering
  - Agglomerative:
    - Start with each node as an individual cluster (called singletons)
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) is left
  - Divisive:
    - Start with one, all-inclusive cluster = the whole graph
    - At each step, split a cluster until each cluster contains a single node (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

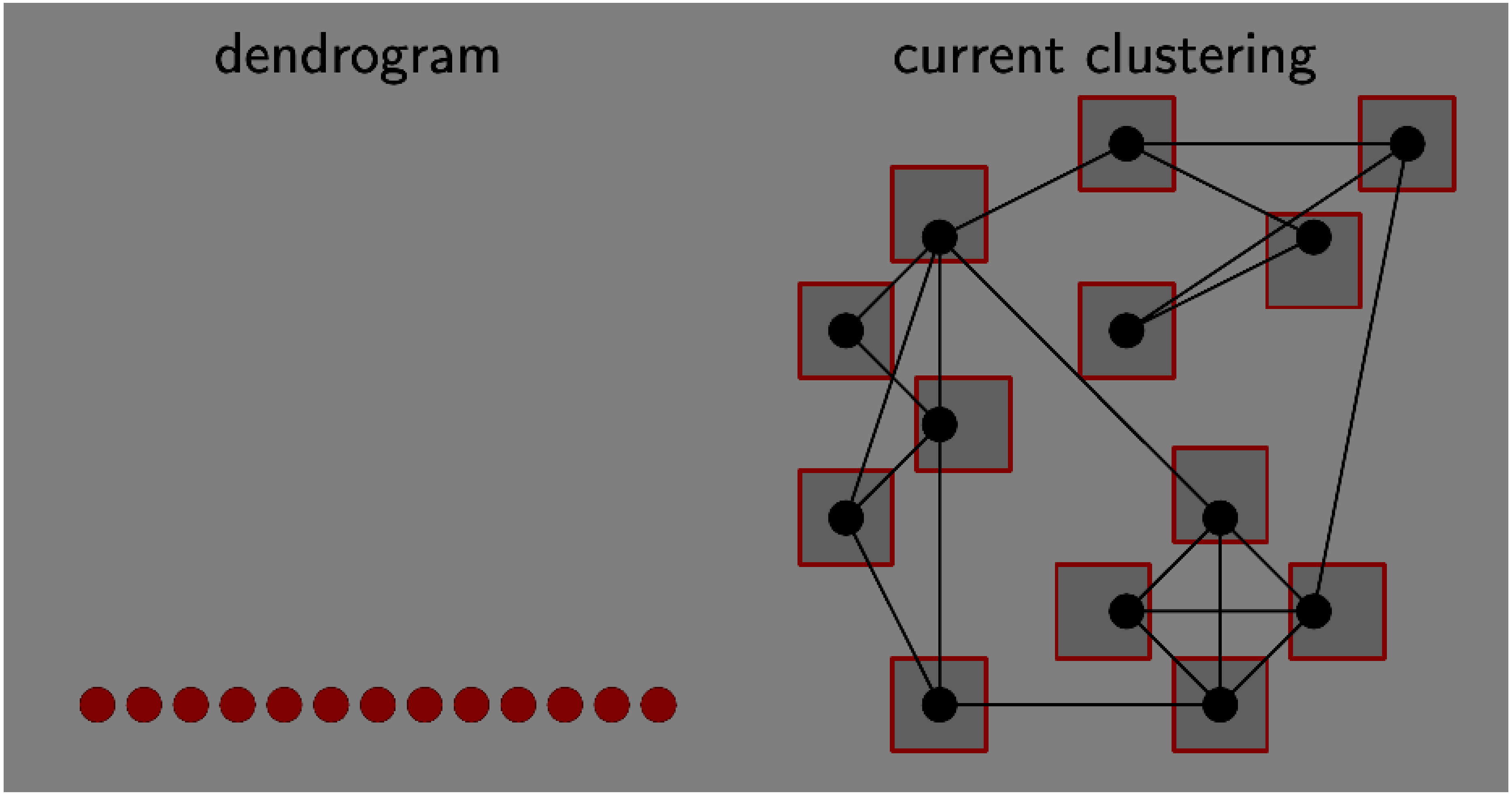
Agglomerative Clustering Algorithm

Popular hierarchical clustering technique

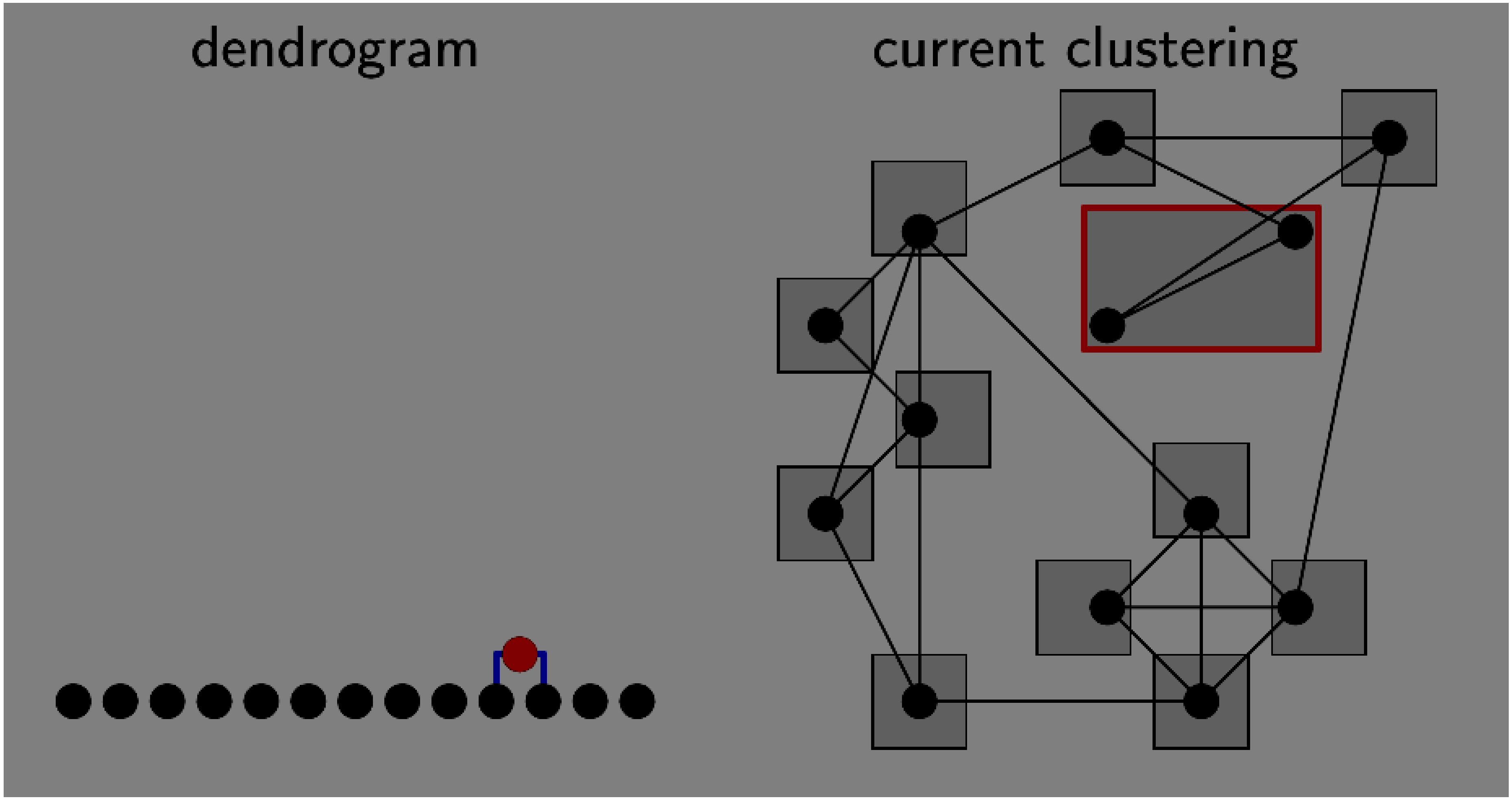
Basic algorithm is straightforward

```
1. [Compute the proximity matrix]
2. Let each node be a cluster
3. Repeat
4.         Merge the two closest clusters
5.         [Update the proximity matrix]
6. Until only a single cluster remains
```

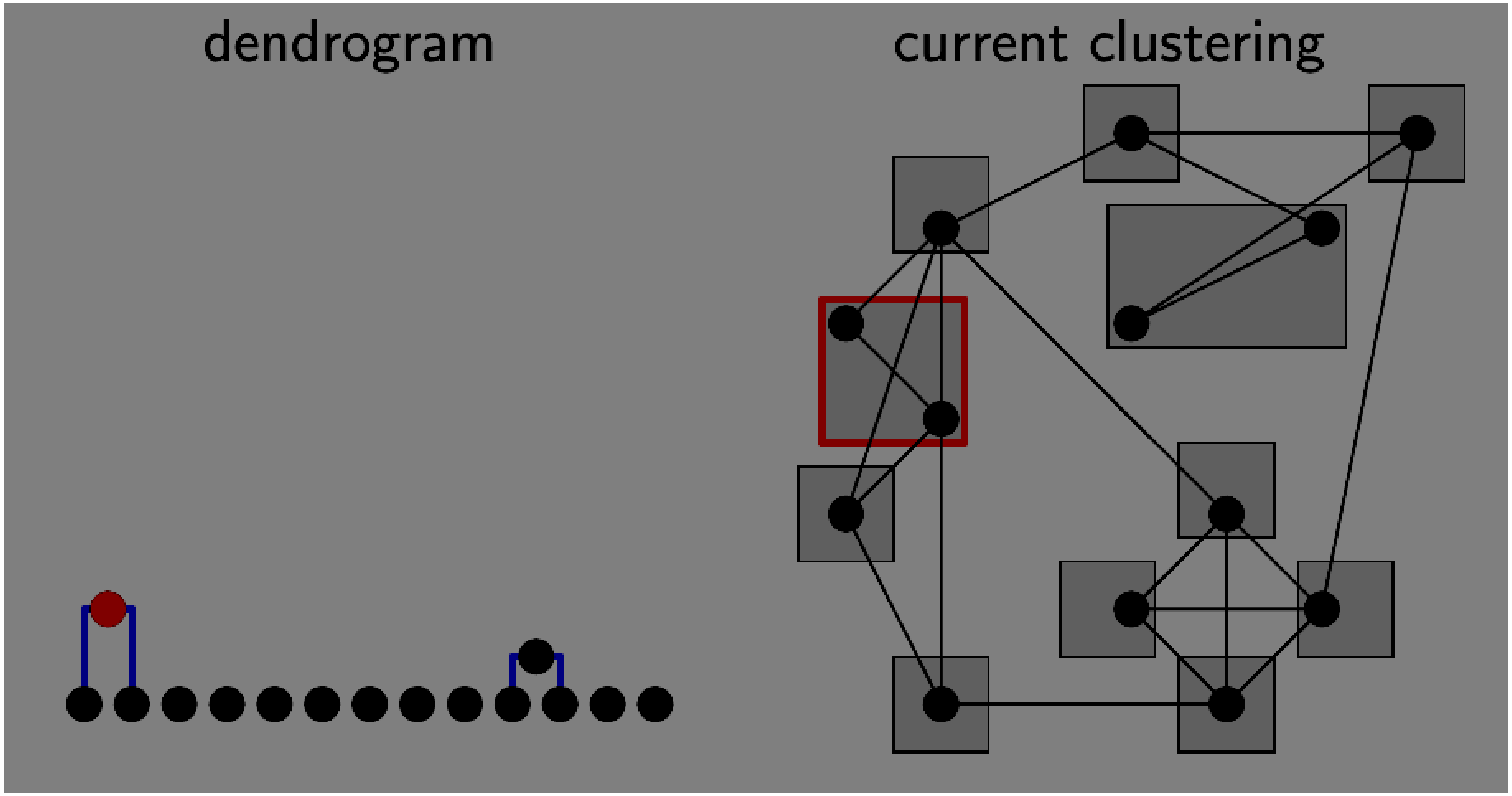
Agglomerative



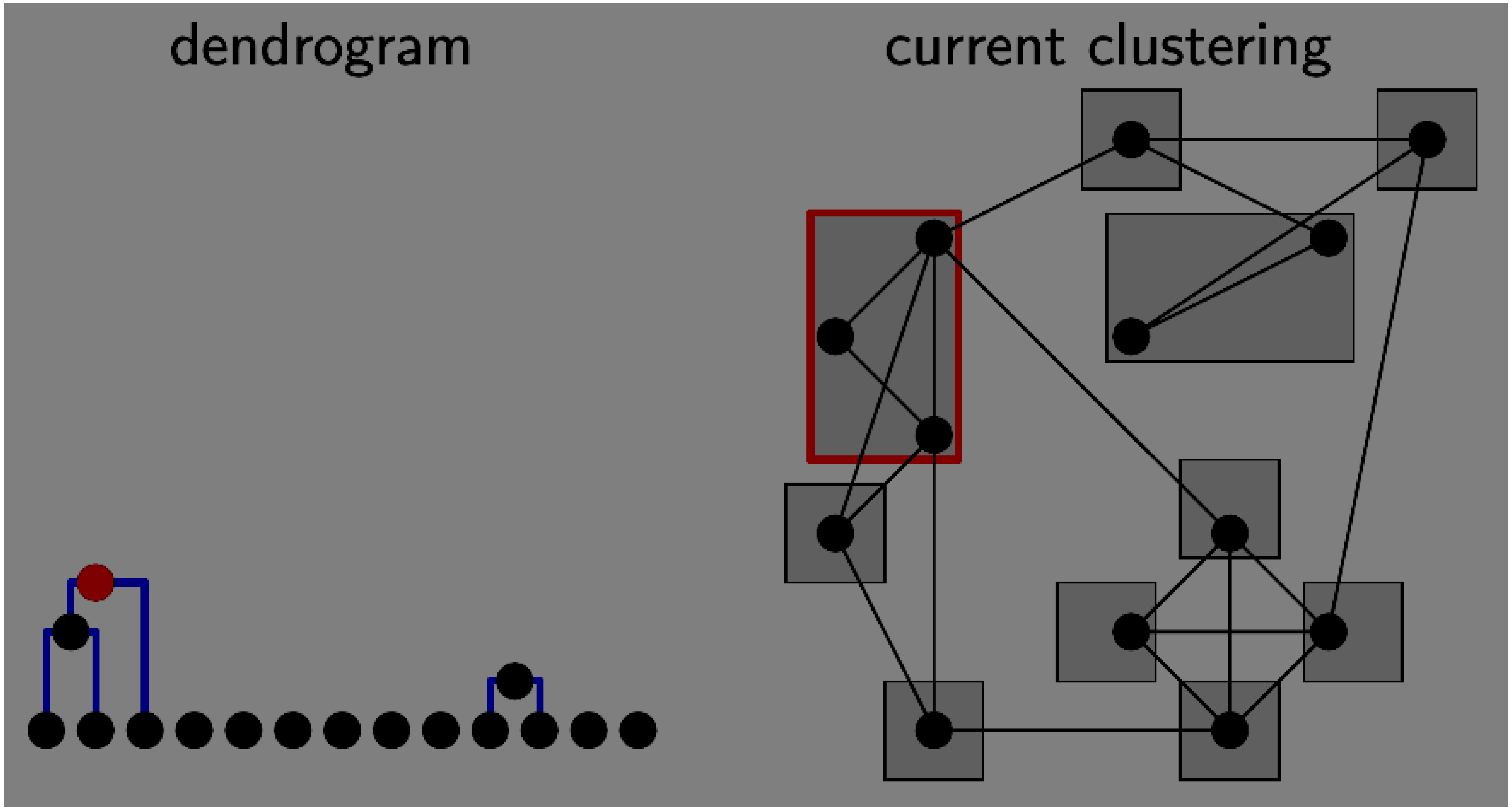
Agglomerative



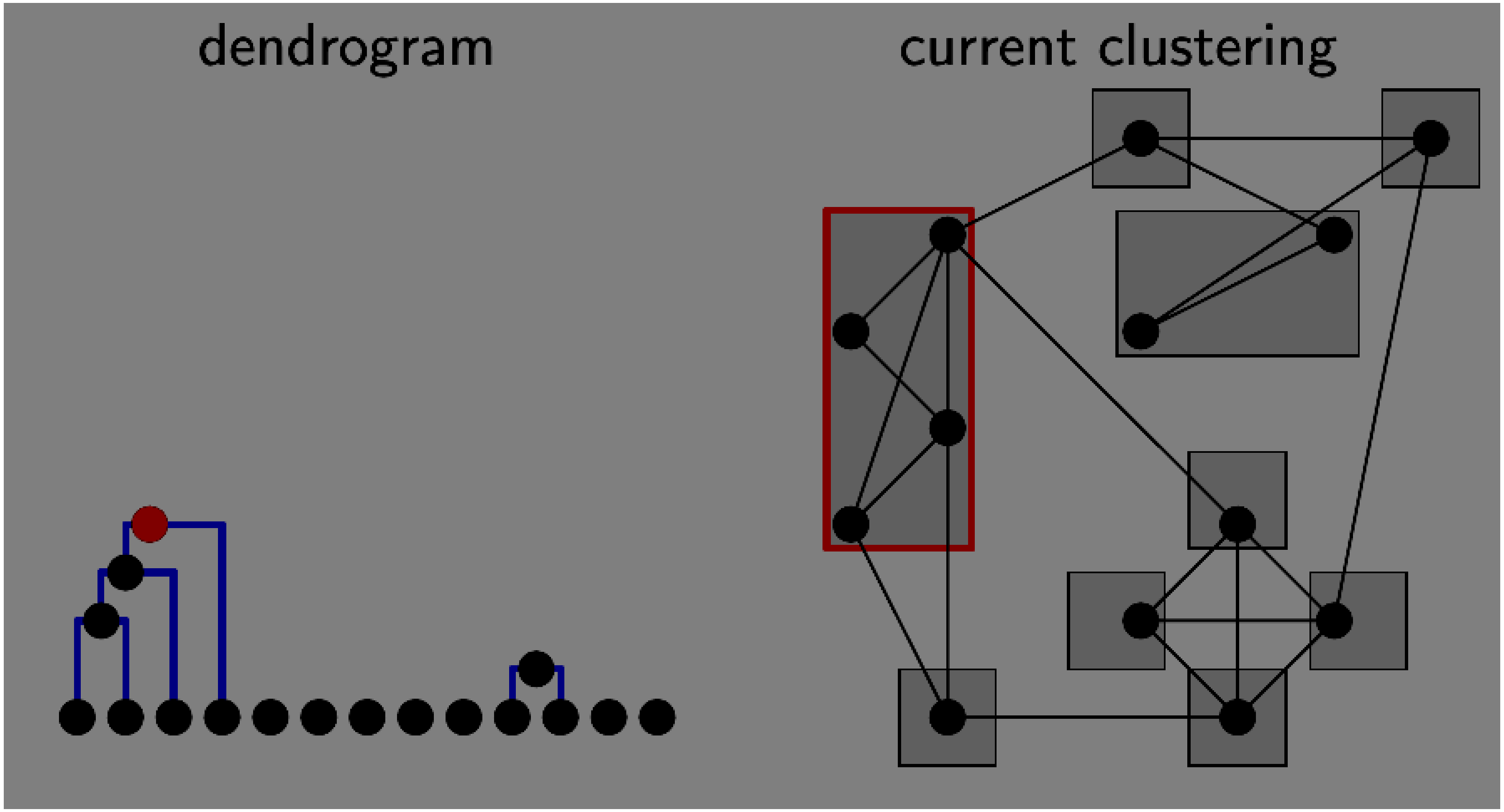
Agglomerative



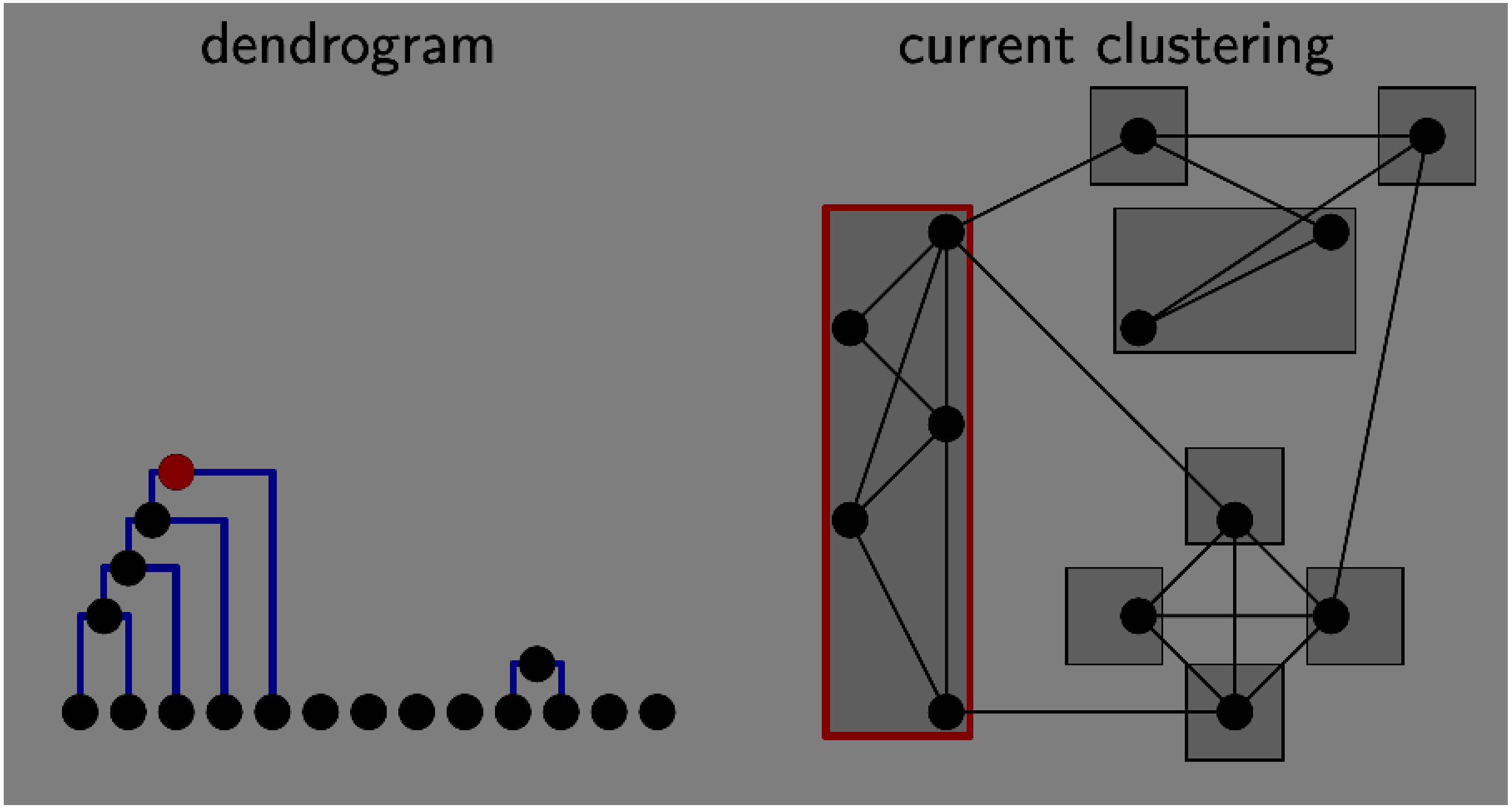
Agglomerative



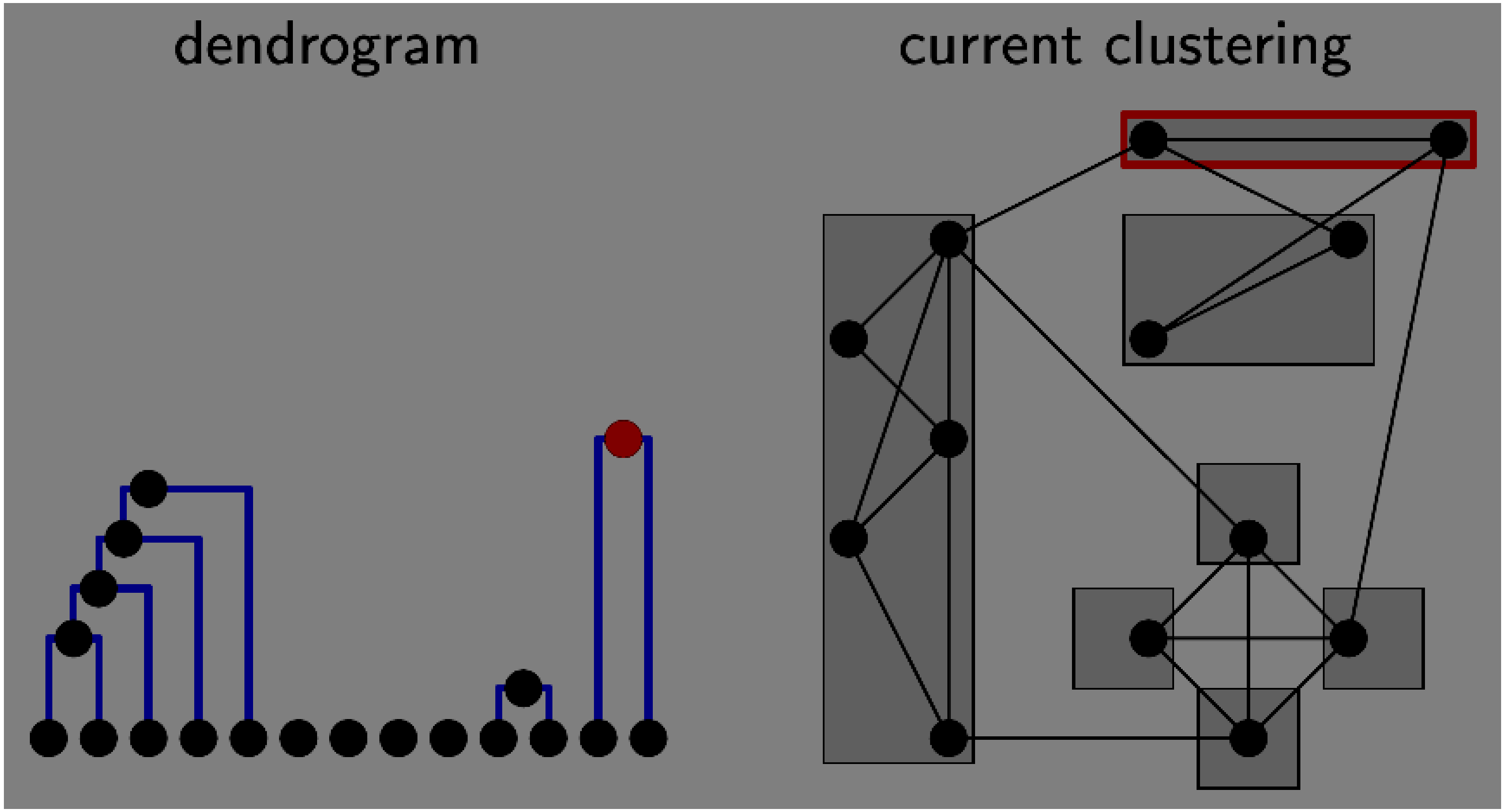
Agglomerative



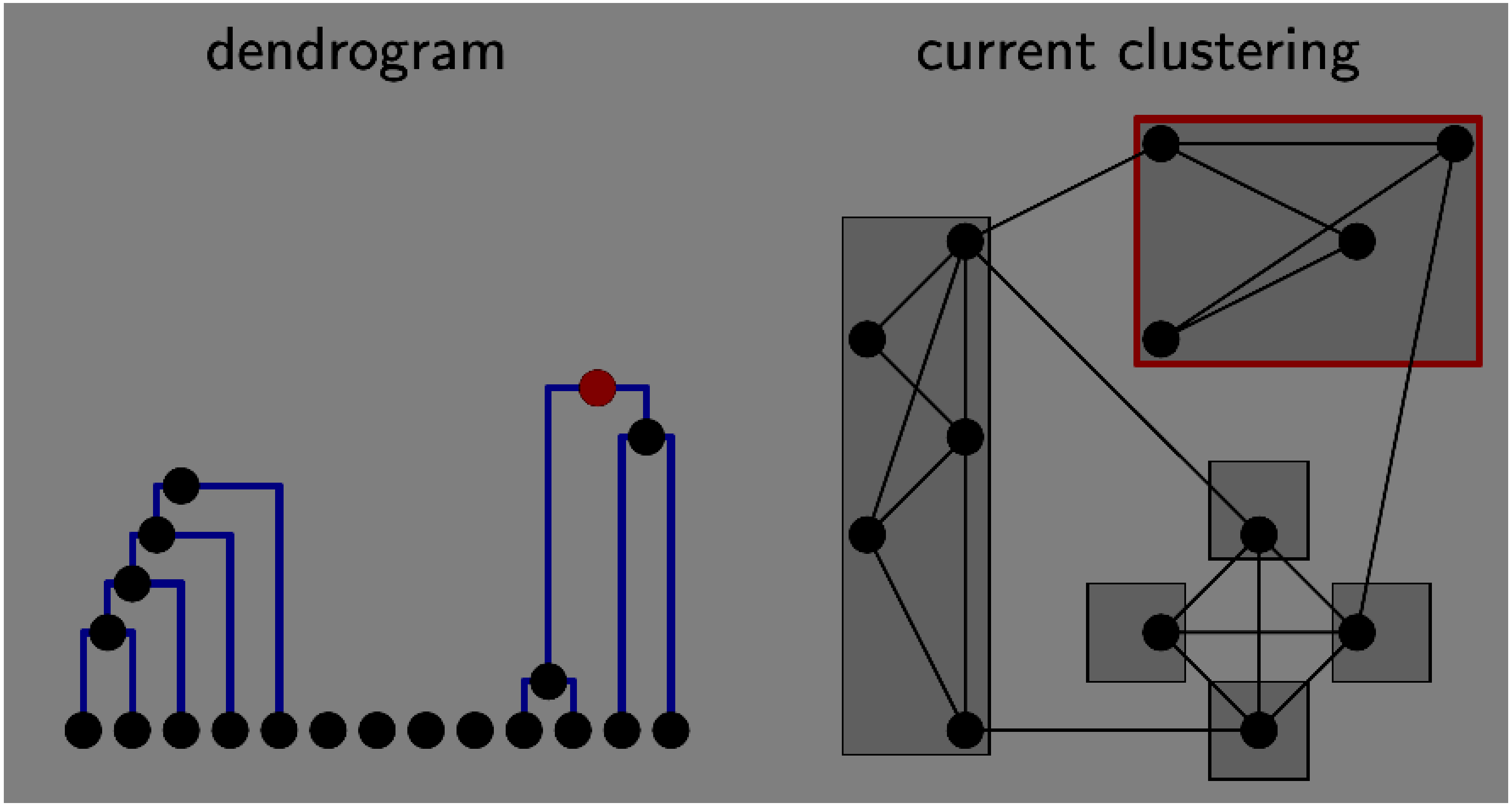
Agglomerative



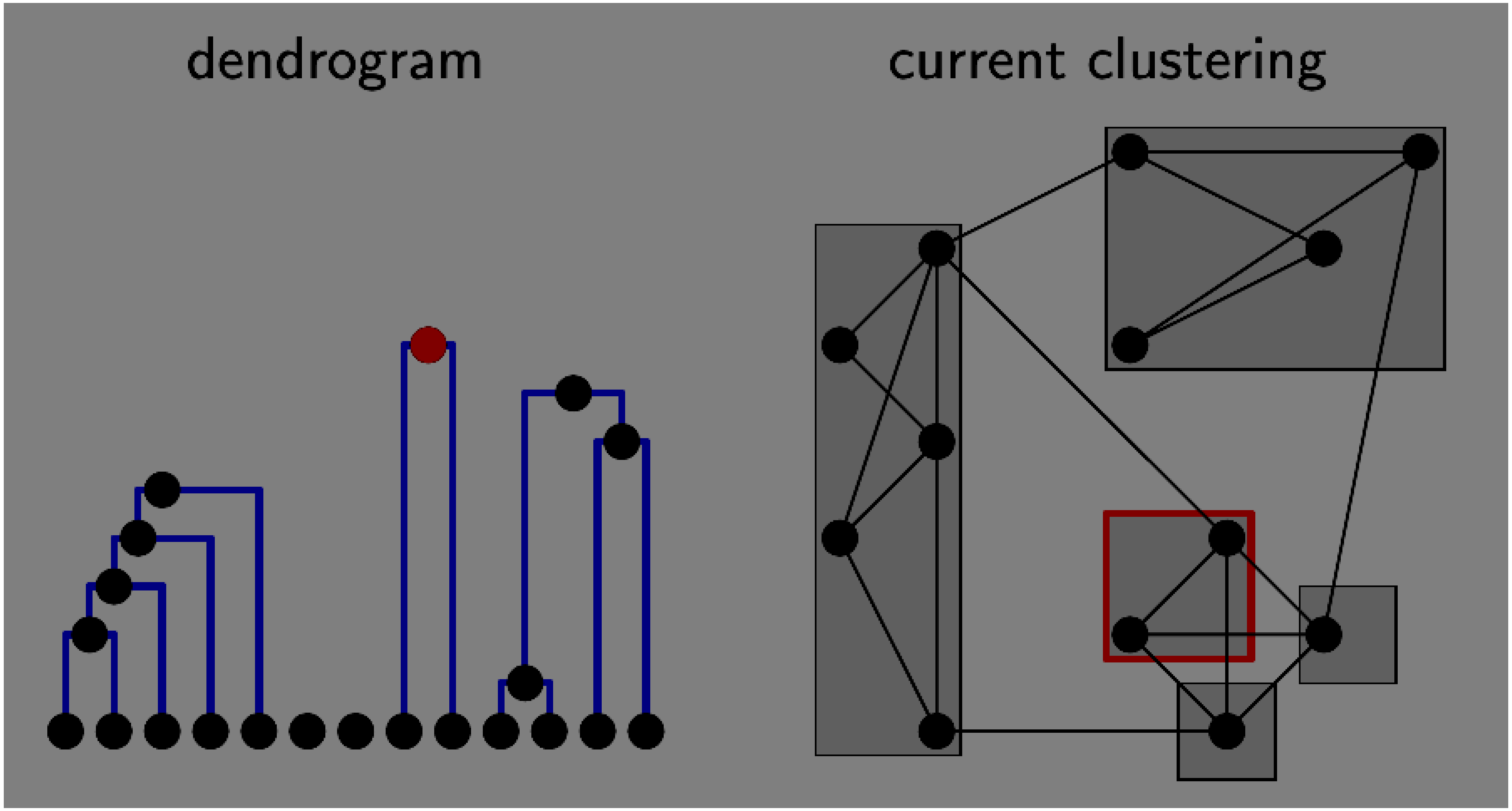
Agglomerative



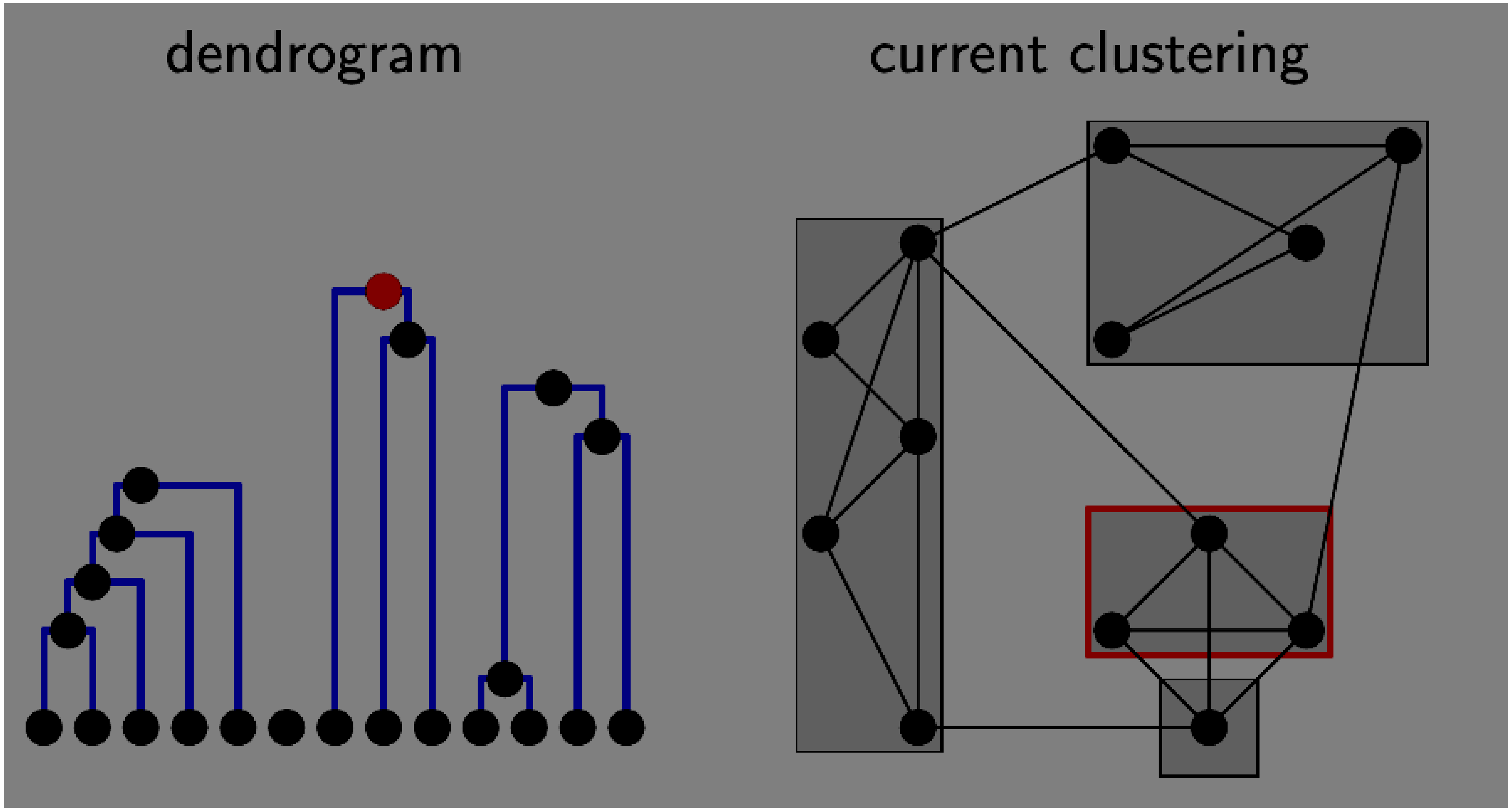
Agglomerative



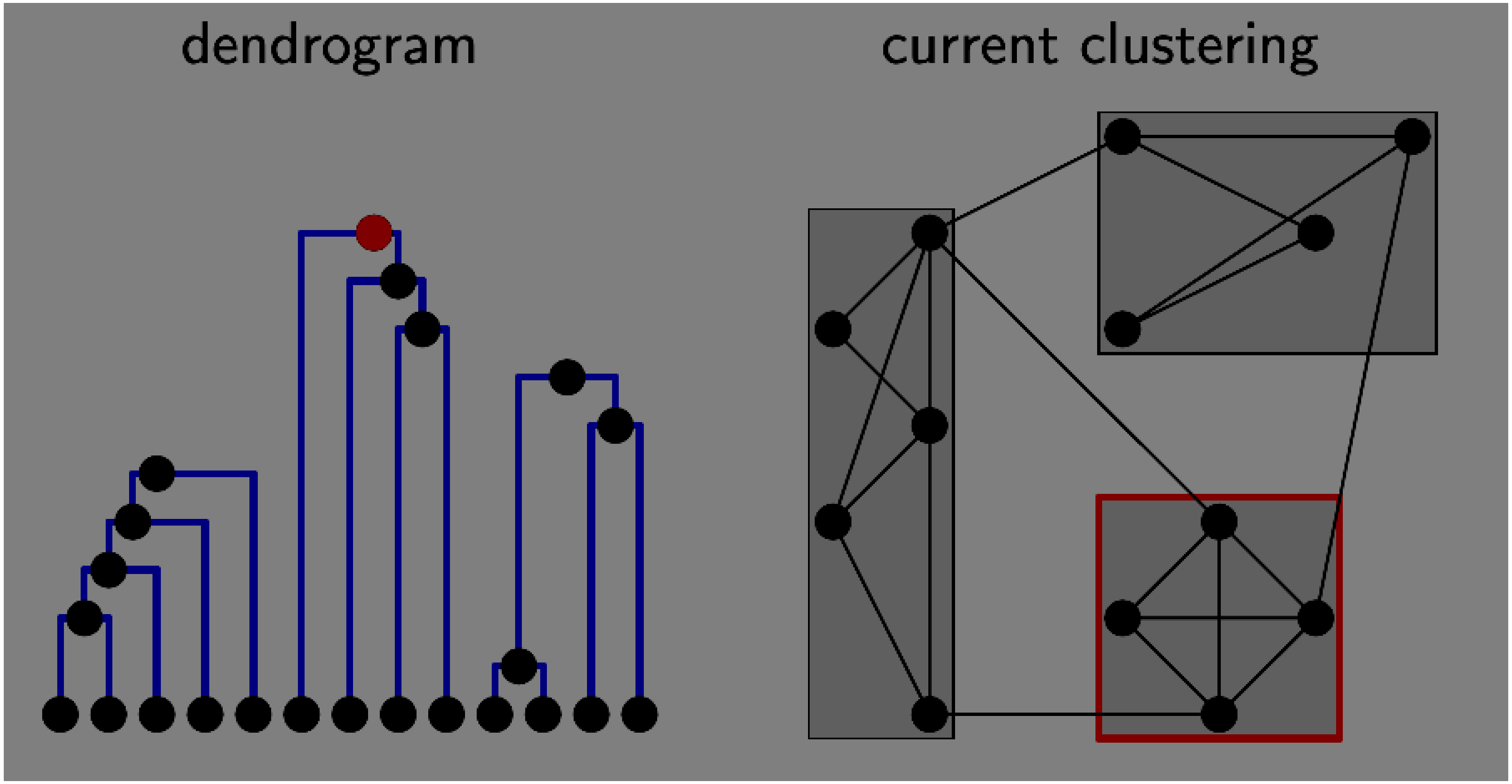
Agglomerative



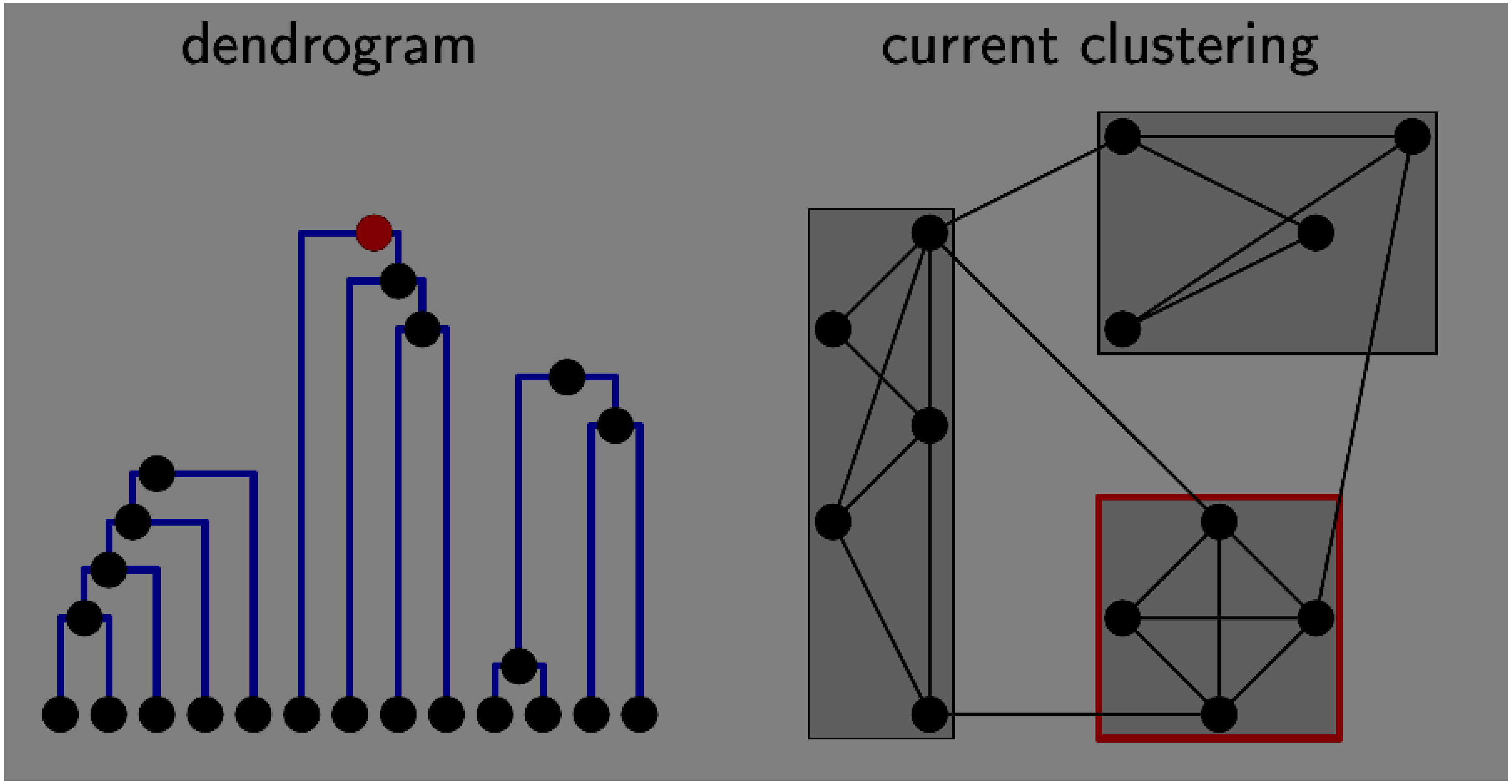
Agglomerative



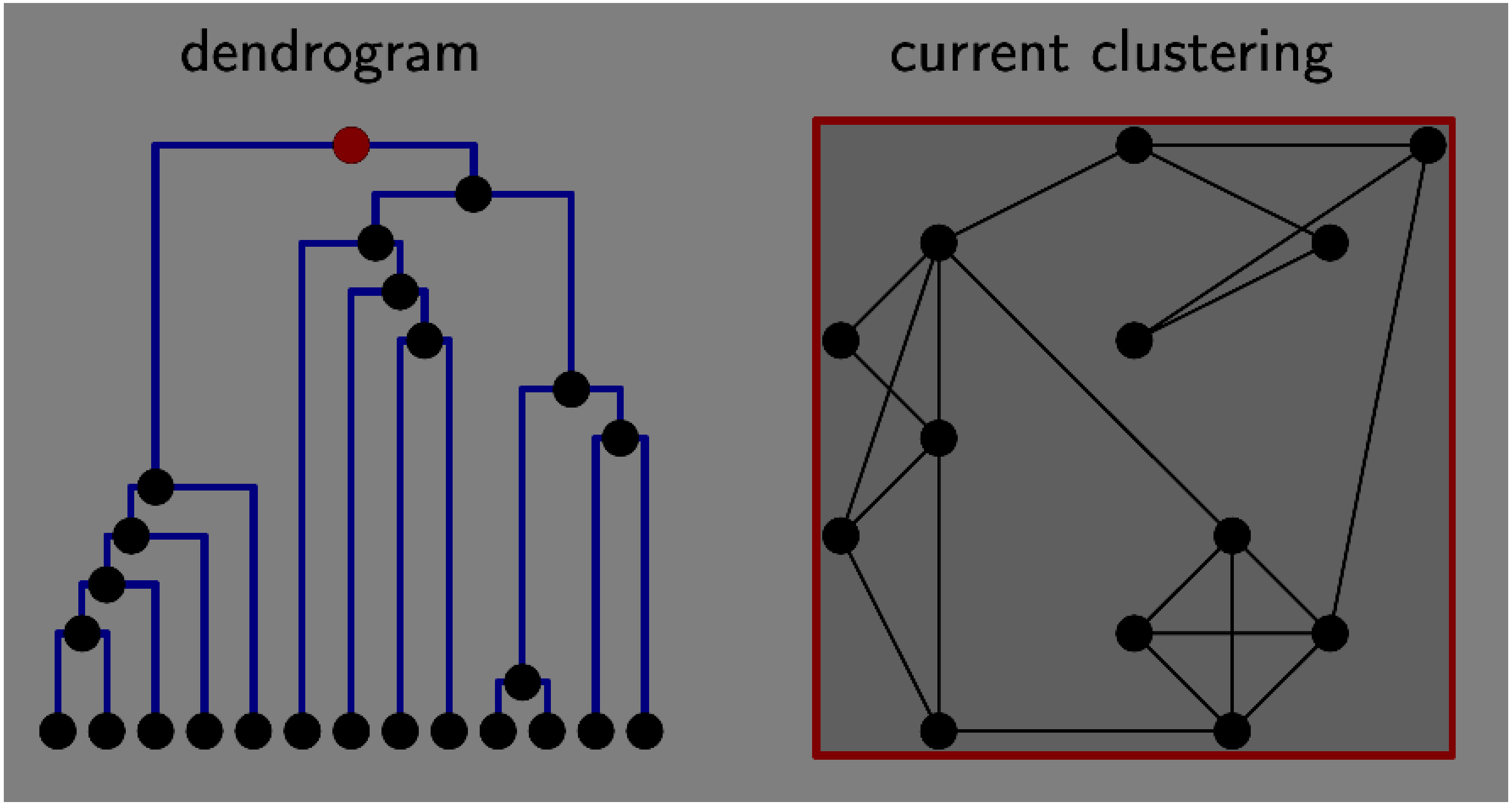
Agglomerative



Agglomerative



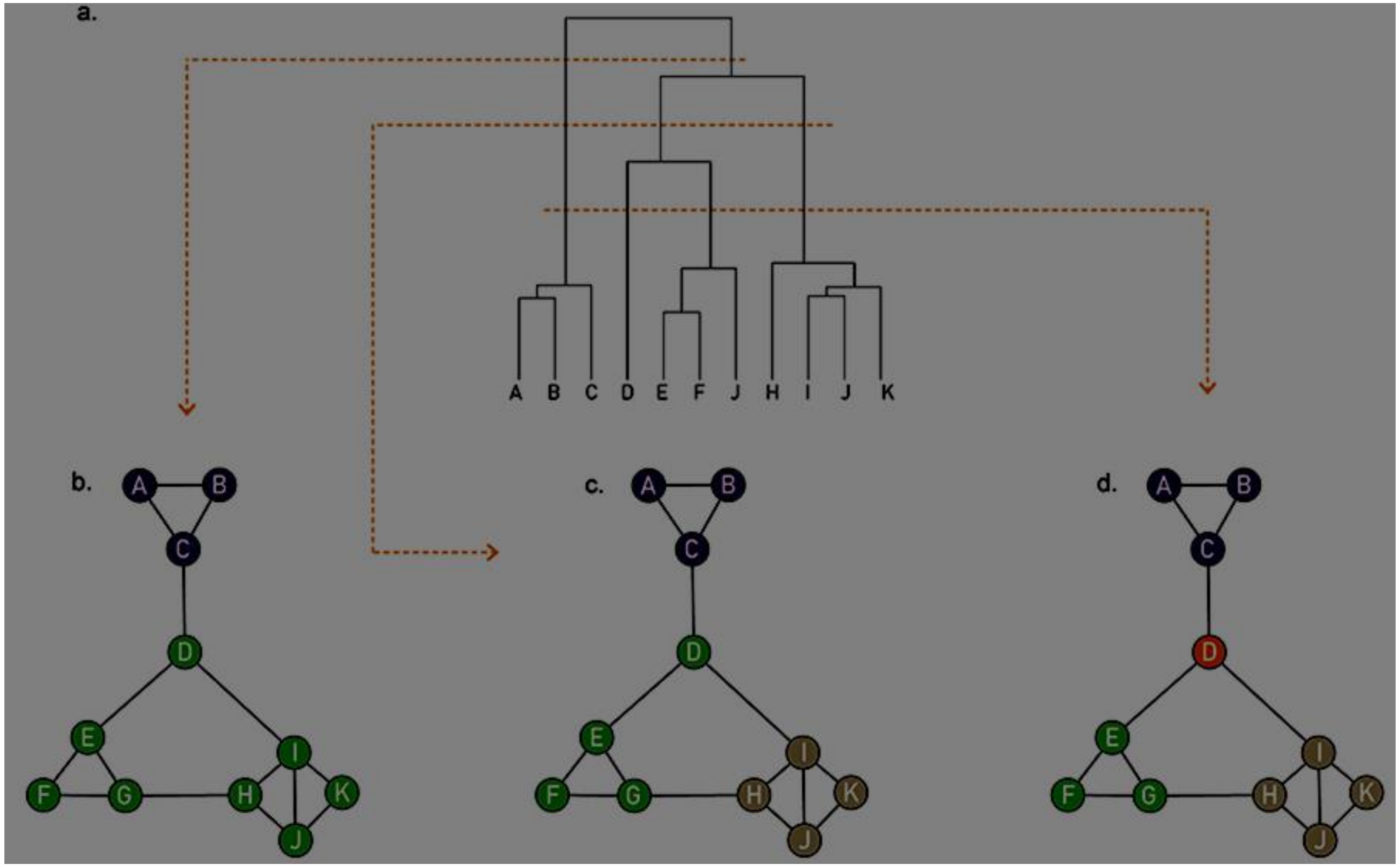
Agglomerative



Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendogram at the proper level
- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)

Where to cut?

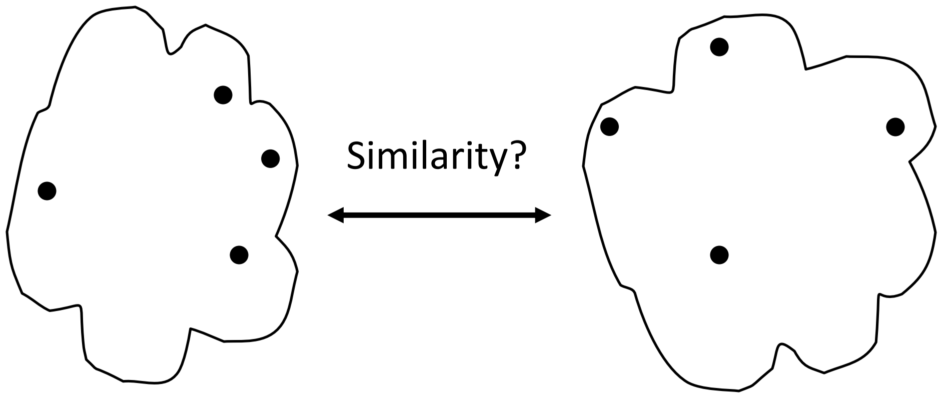


Agglomerative Clustering Algorithm

Key operation is the *computation of the proximity of two clusters*

- Different approaches to defining the distance between clusters distinguish the different algorithms

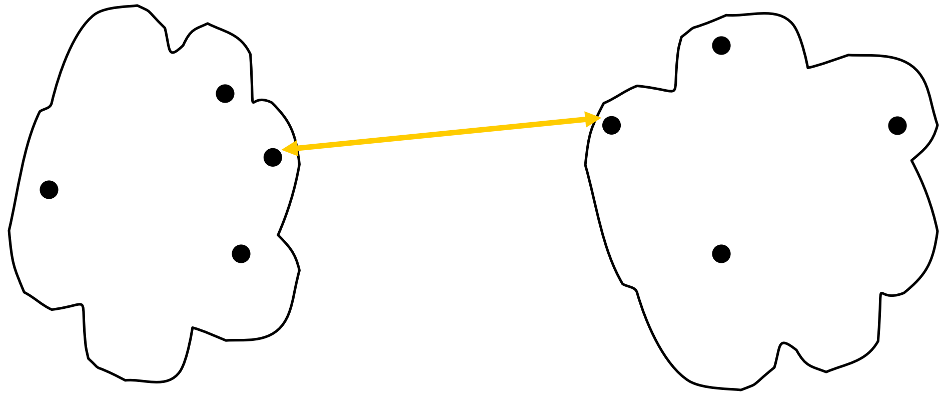
How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
⋮						

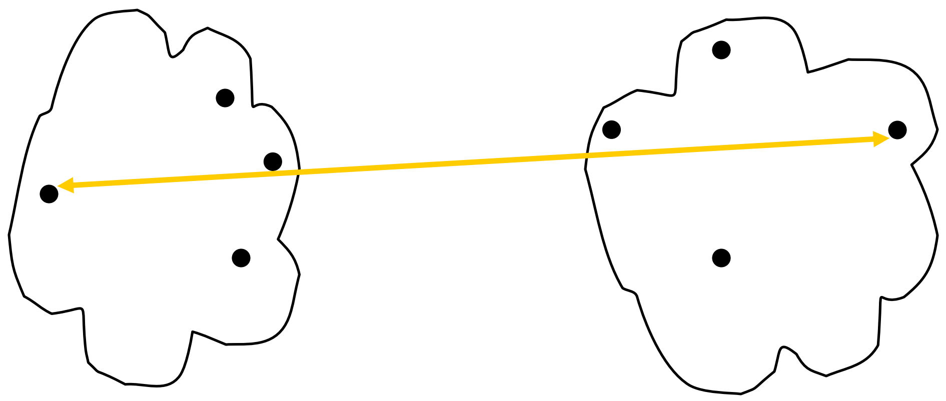
Proximity Matrix

MIN or single link

based on the two most similar (closest) points in the different clusters

(sensitive to outliers)

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

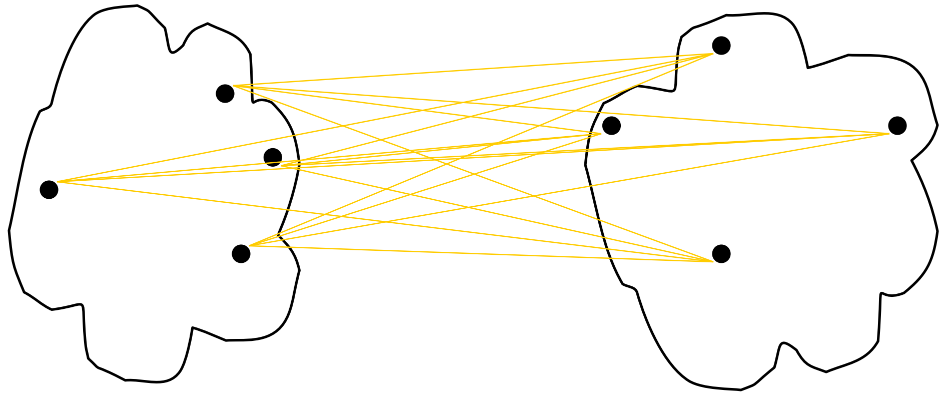
Proximity Matrix

MAX or complete linkage

Similarity of two clusters is based on the two least similar (most distant) points in the different clusters

(Tends to break large clusters
Biased towards globular clusters)

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

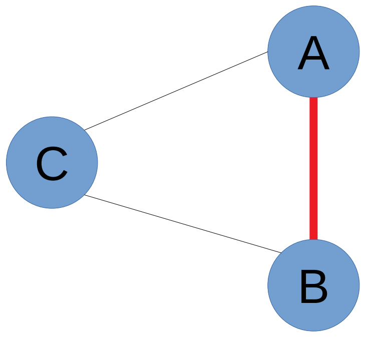
Group Average

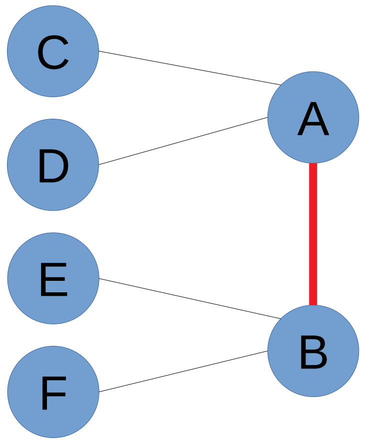
Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

Divisive Algorithms

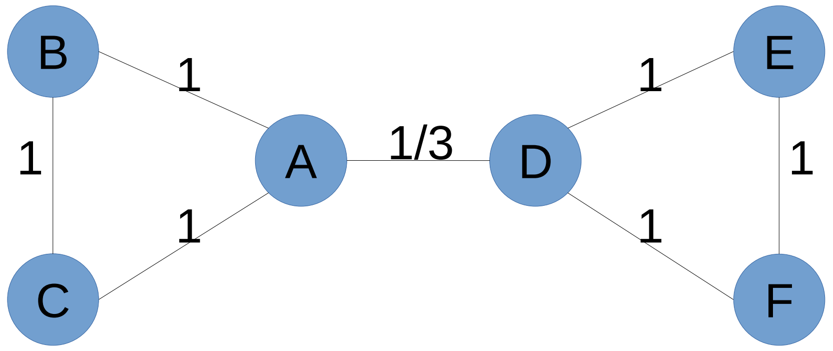
remove the “spanning links” between densely-connected regions

$$e_{ij} = \frac{|N(i) \cap N(j)| + A_{ij}}{\min\{d(i), d(j)\}}$$

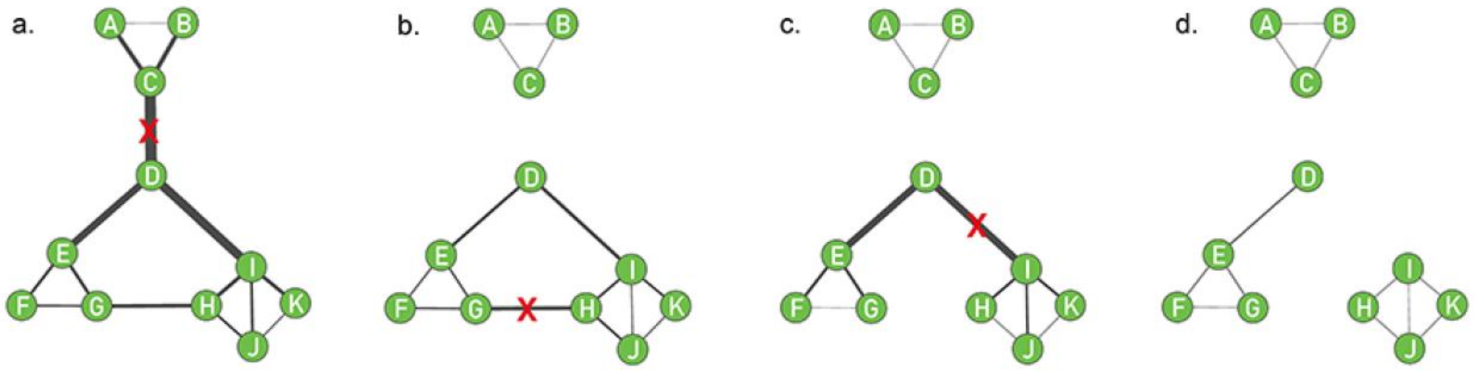


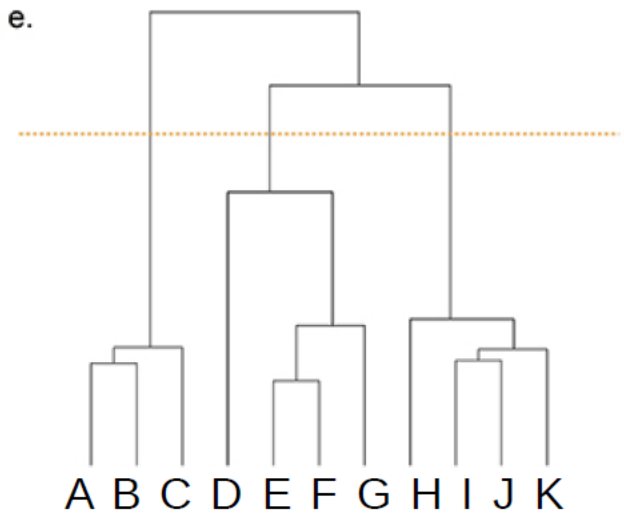


Divisive Algorithms

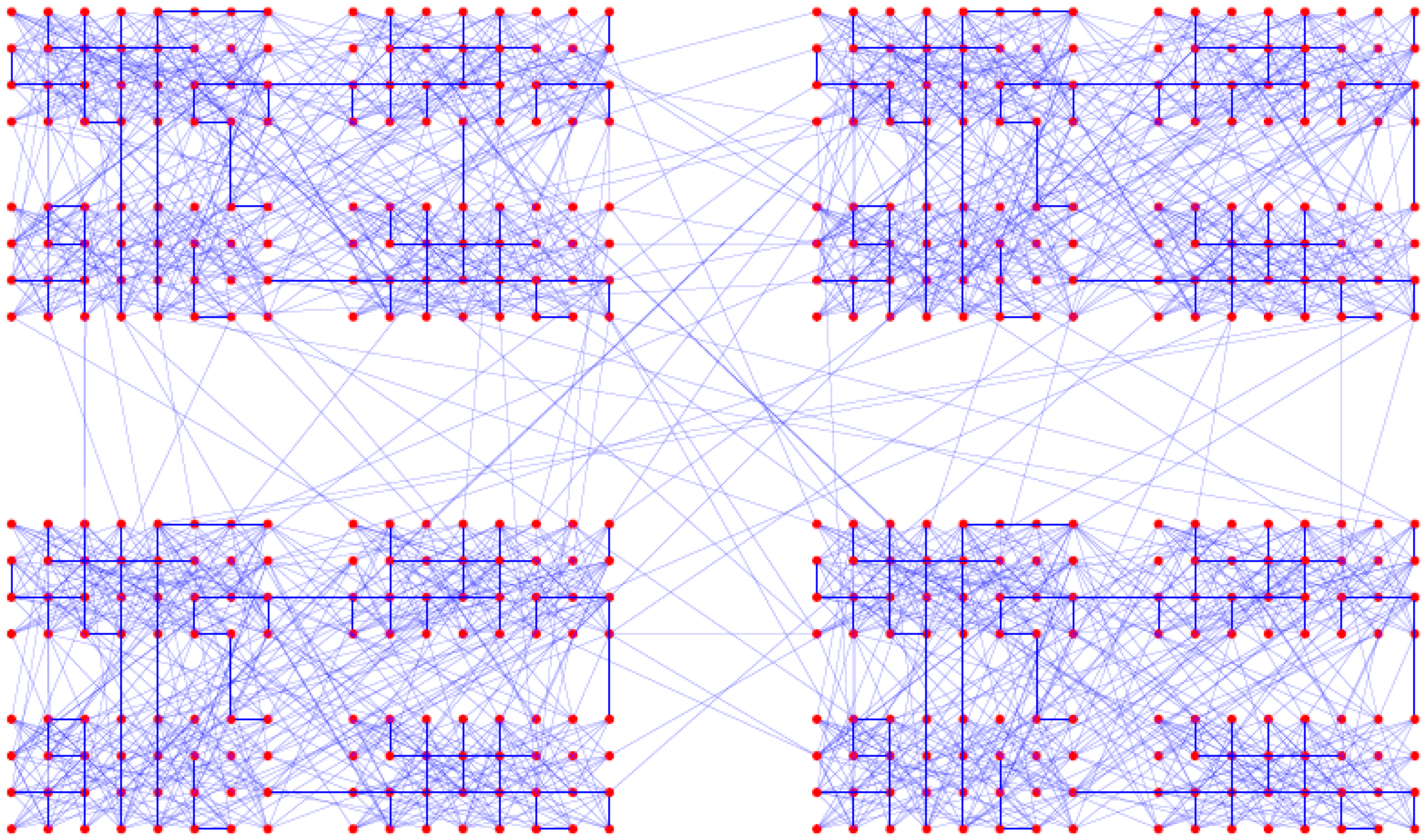


Divisive Algorithms





Example of a Hierarchically Structured Graph



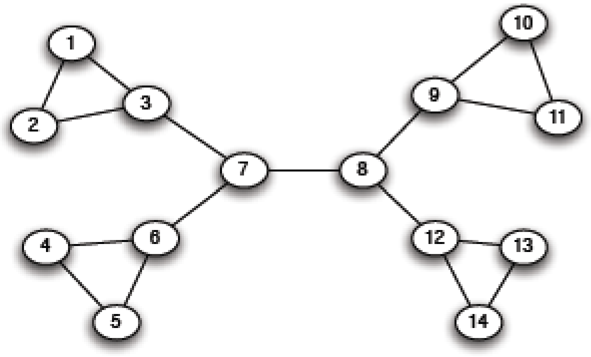
Outline

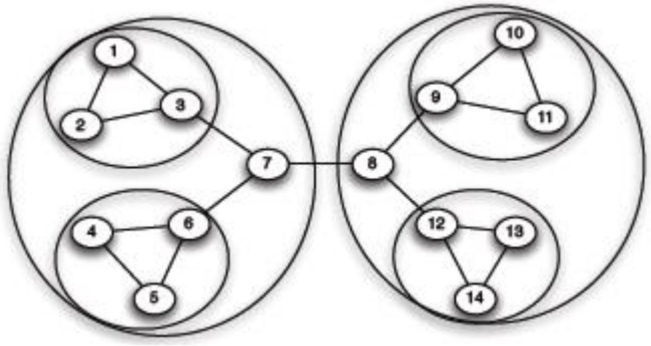
PART I

1. Introduction: what, why, types?
2. Cliques
3. Background: How it relates to “cluster analysis” (node/edge similarity)
4. Betweeness centrality
5. Modularity
6. How to evaluate

Graph Partitioning

- Divisive methods: try to identify and remove the "spanning links" between densely-connected regions
- Agglomerative methods: Find nodes that are likely to belong to the same region and merge them together (bottom-up)



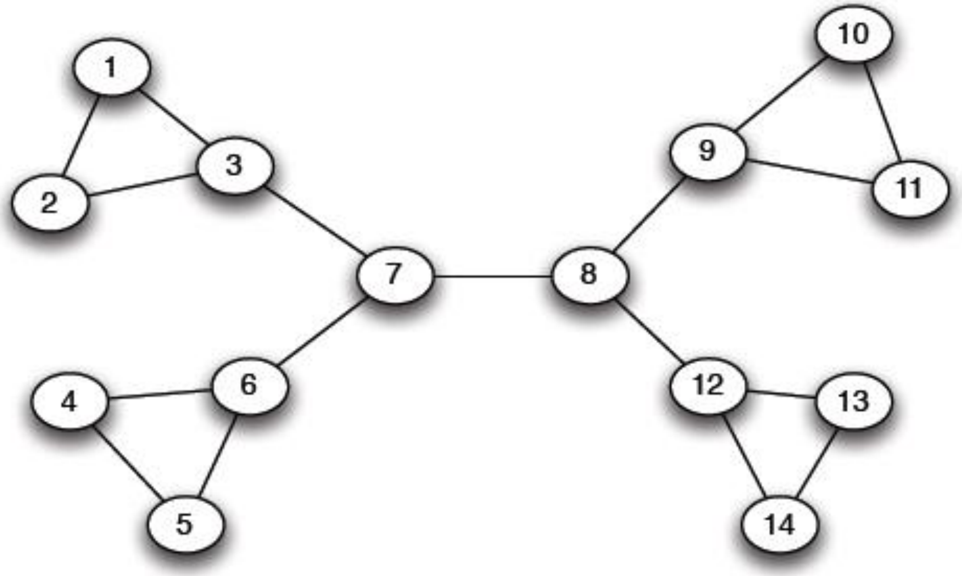


The Girvan Newman method

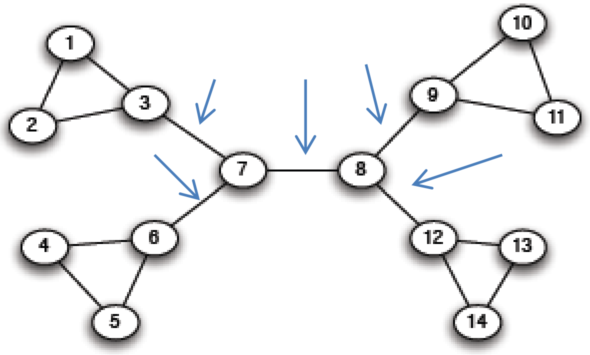
Hierarchical divisive method

- Start with the whole graph
- Find edges whose removal *"partitions"* the graph
- Repeat with each subgraph until single vertices

Which edge?



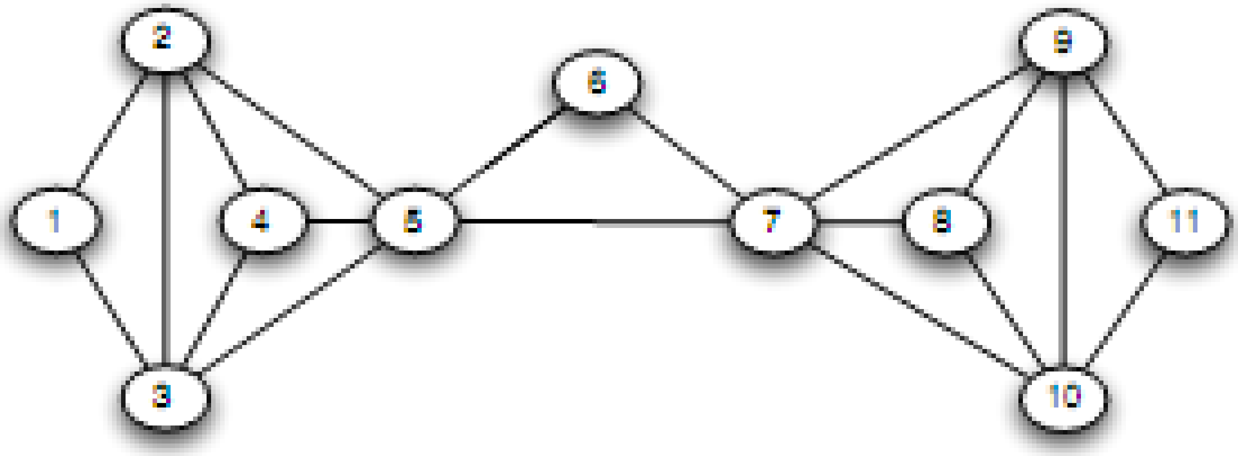
The Girvan Newman method

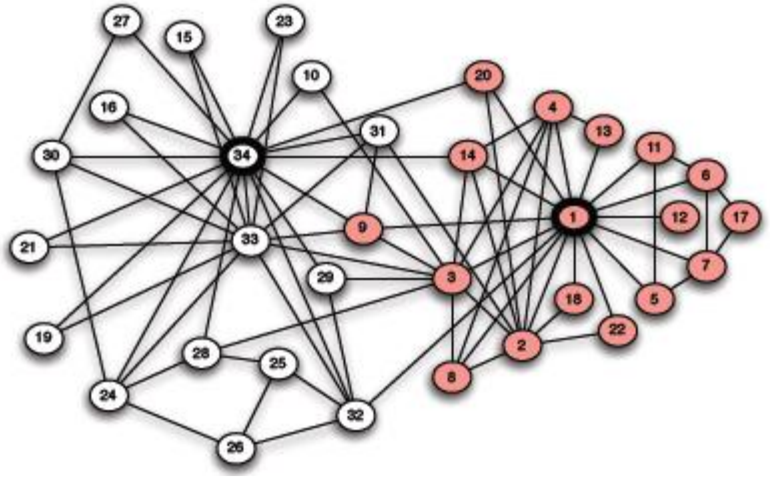


Use bridges or cut-edge (if removed, the nodes become disconnected)

Which one to choose?

The Girvan Newman method



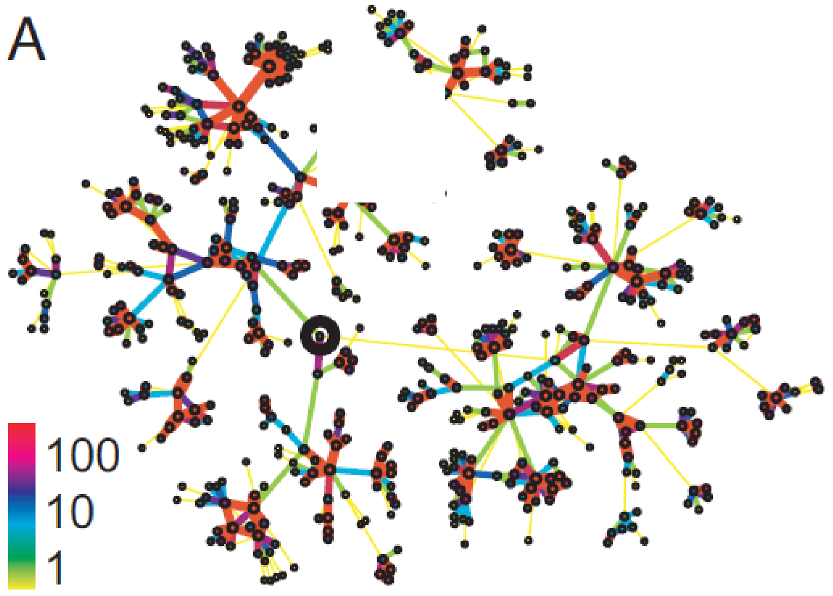


There may be none!

Strength of Weak Ties

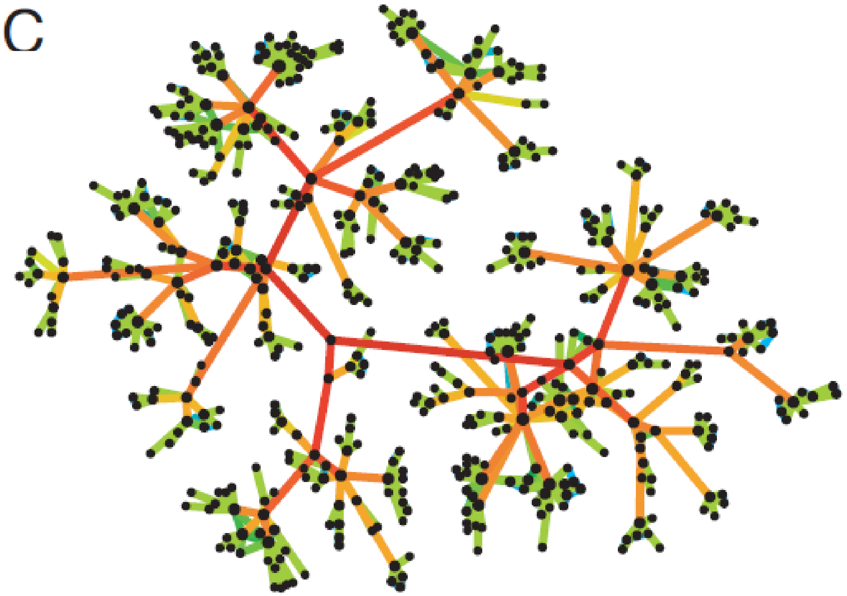
- Edge betweenness: Number of shortest paths passing over the edge
- Intuition:

A



Edge strengths (call volume) in a real network

C

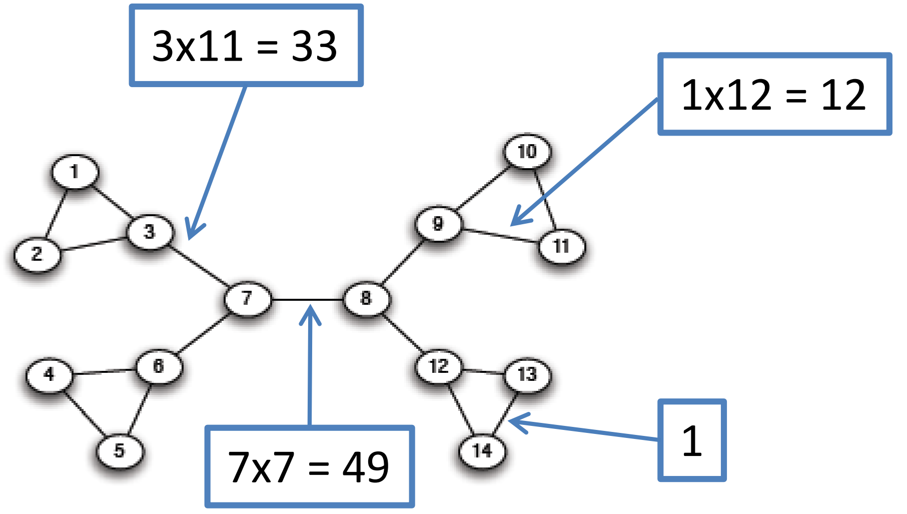


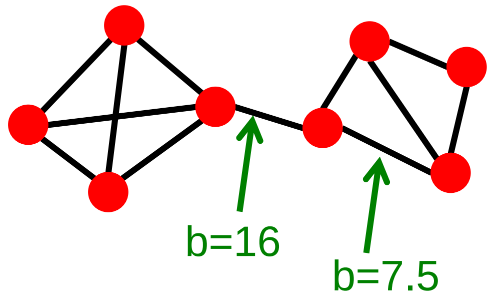
Edge betweenness in a real network

Edge Betweenness

Betweenness of an edge (a, b): number of pairs of nodes x and y such that the edge (a, b) lies on the shortest path between x and y - since there can be several such shortest paths edge (a, b) is credited with the fraction of those shortest paths that include (a, b).

$$\mathrm{bt}(a,b) = \sum_{x,y} \frac{\#shortest\_paths(x,y)through(a,b)}{\#shortest\_paths(x,y)}$$





Edges that have a high probability to occur on a randomly chosen shortest path between two randomly chosen nodes have a high betweenness.

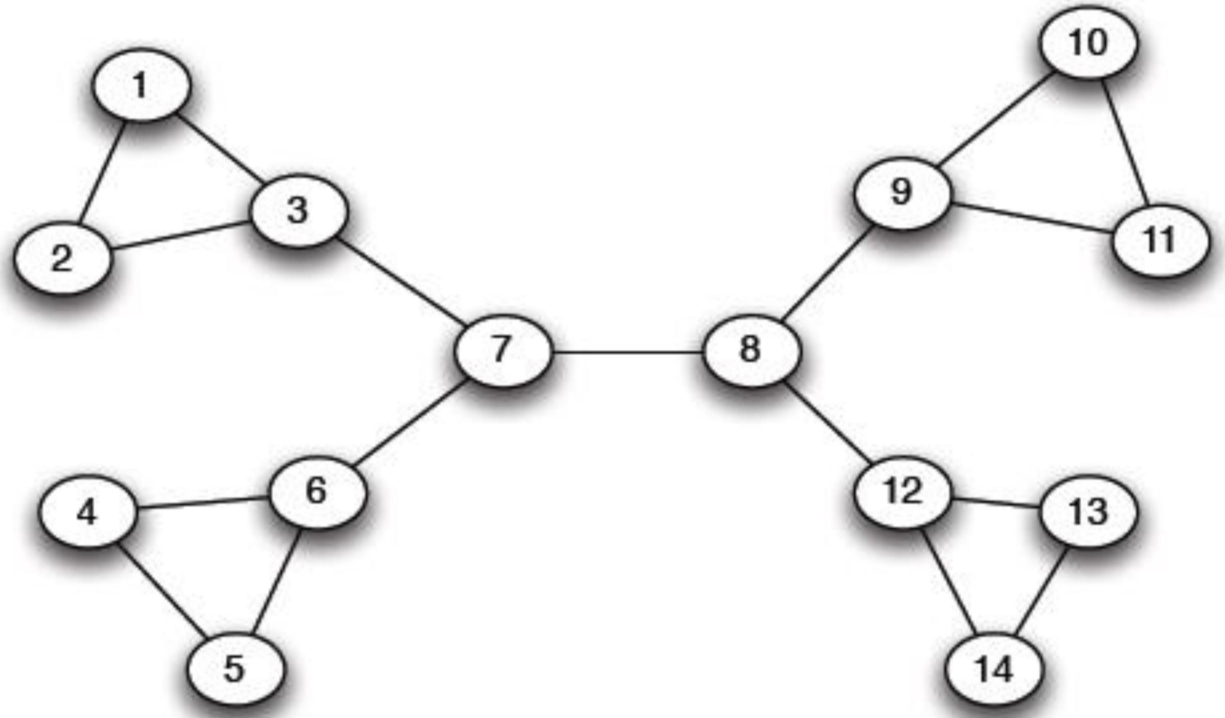
Traffic (unit of flow)

The Girvan Newman method

» **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 method: An example

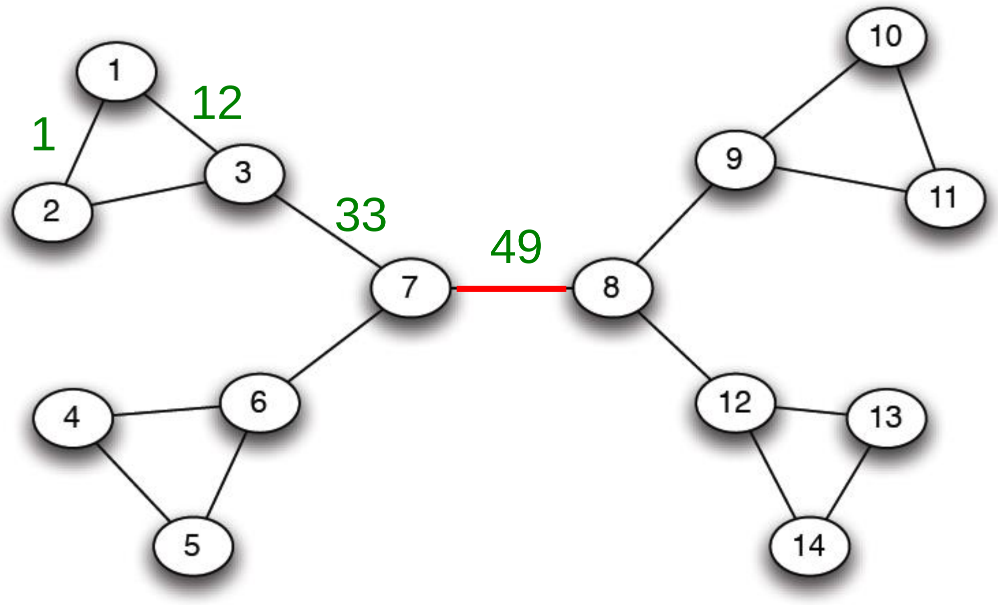


Betweenness(7, 8)= 7x7 = 49

Betweenness(3, 7)=Betweenness(6, 7)=Betweenness(8, 9) = Betweenness(8, 12)= 3x11=33

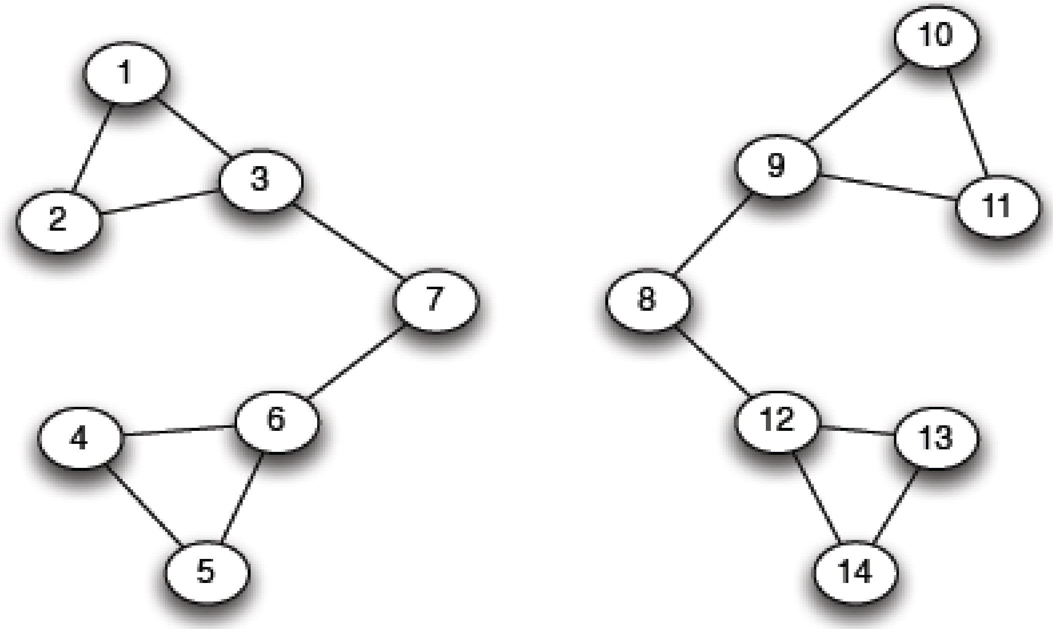
Betweenness(1, 3) = 1x12=12

Girvan-Newman: Example



Need to re-compute betweenness at every step

Girvan Newman method: An example

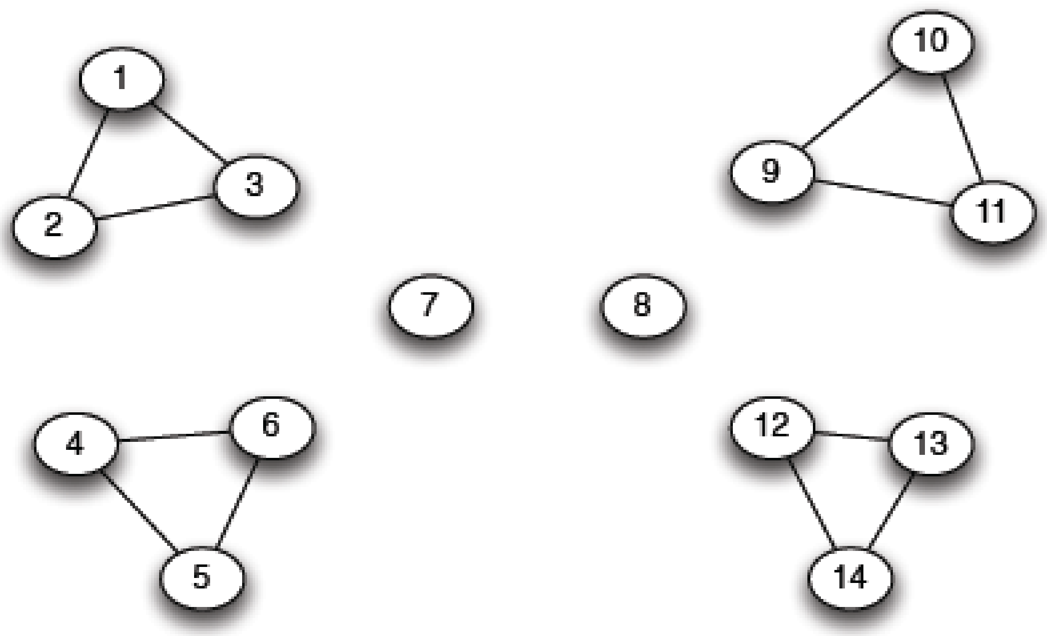


(a) *Step 1*

Betweenness(1, 3) = 1x5=5

Betweenness(3,7)=Betweenness(6,7)=Betweenness(8,9) = Betweenness(8,12)= 3x4=12

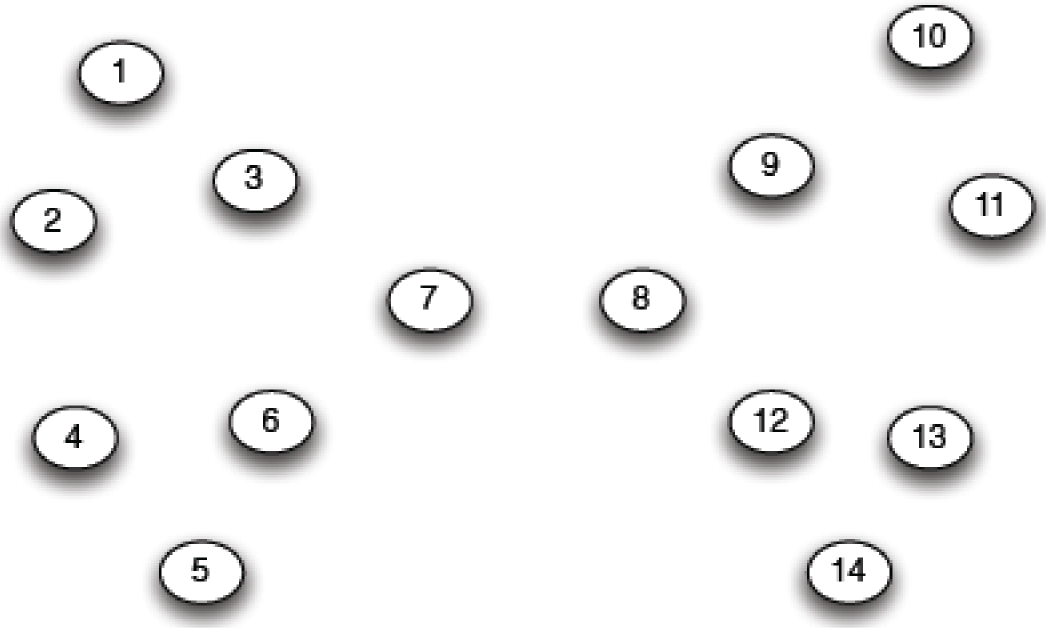
Girvan Newman method: An example



(b) *Step 2*

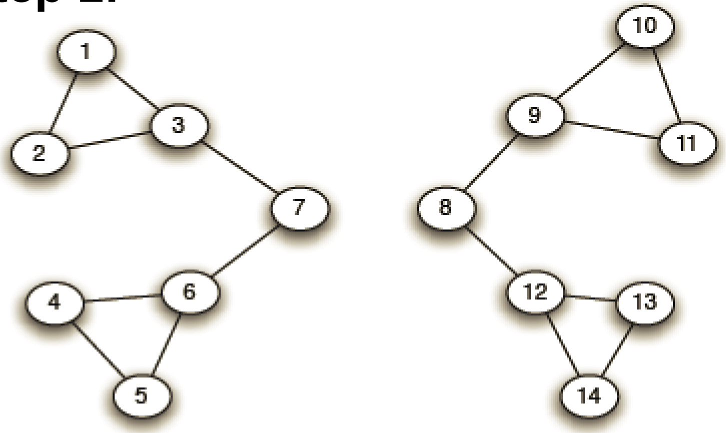
Betweenness of every edge = 1

Girvan Newman method: An example

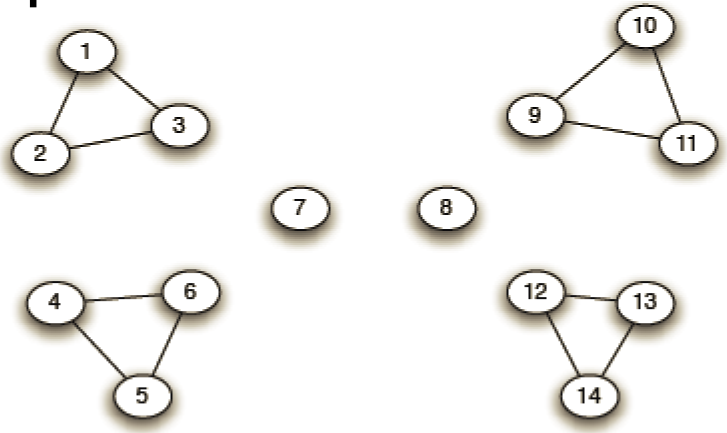


Girvan-Newman: Example

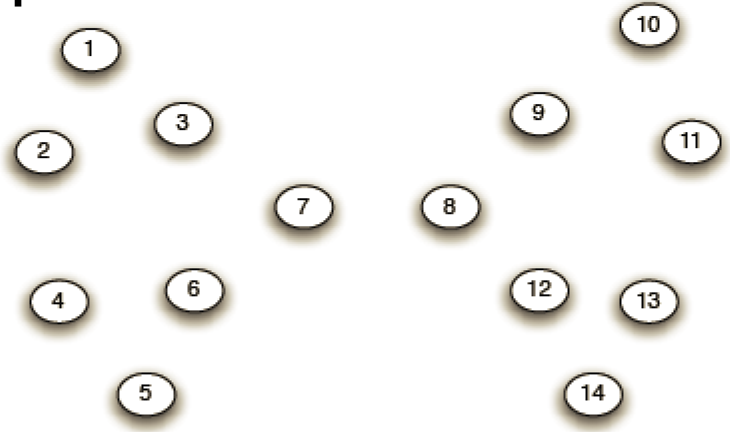
Step 1:



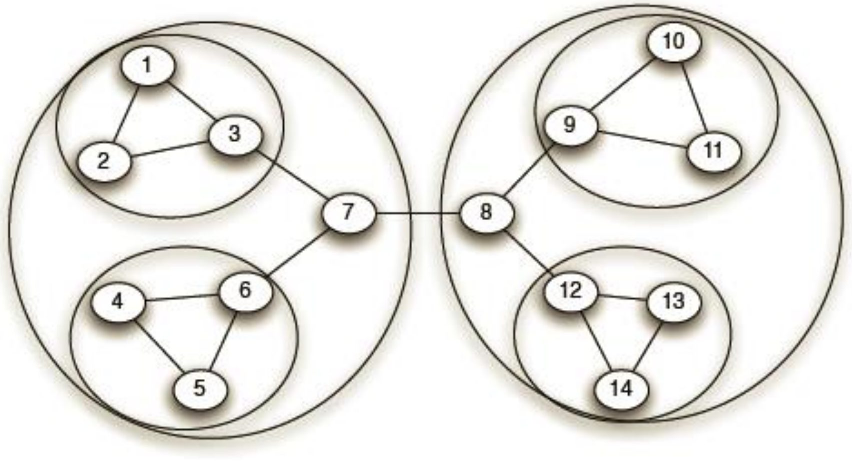
Step 2:



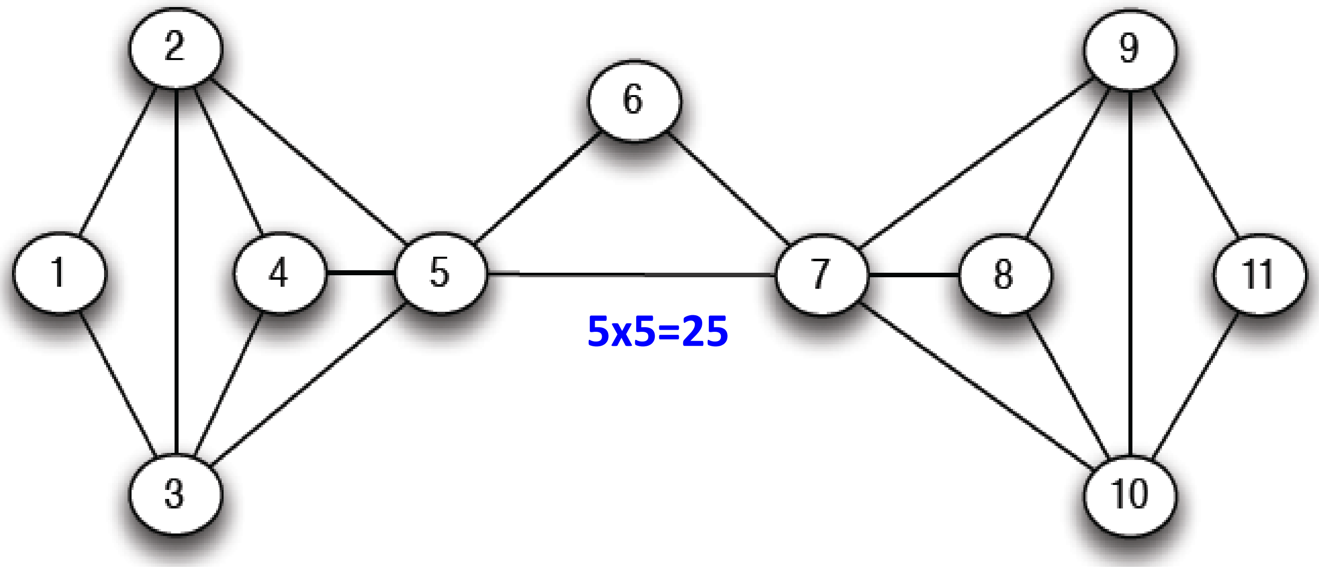
Step 3:



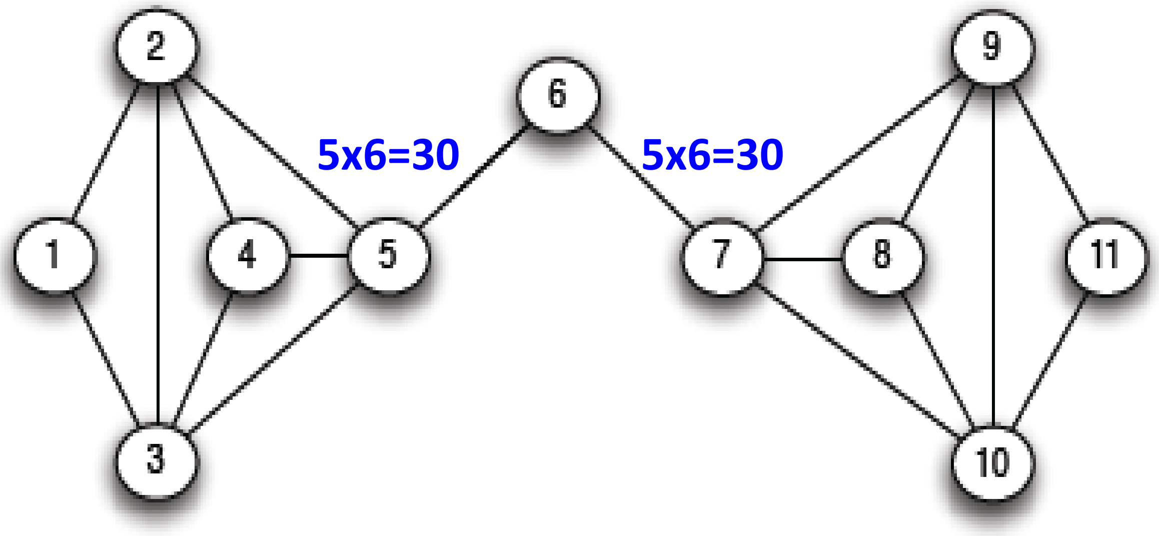
Hierarchical network decomposition:



Another example

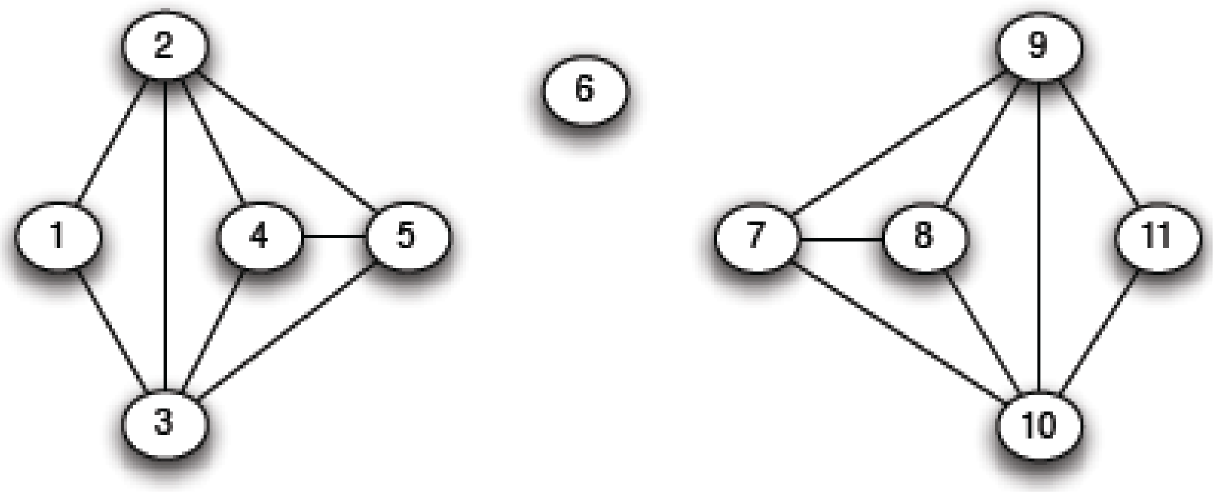


Another example



(a) *Step 1*

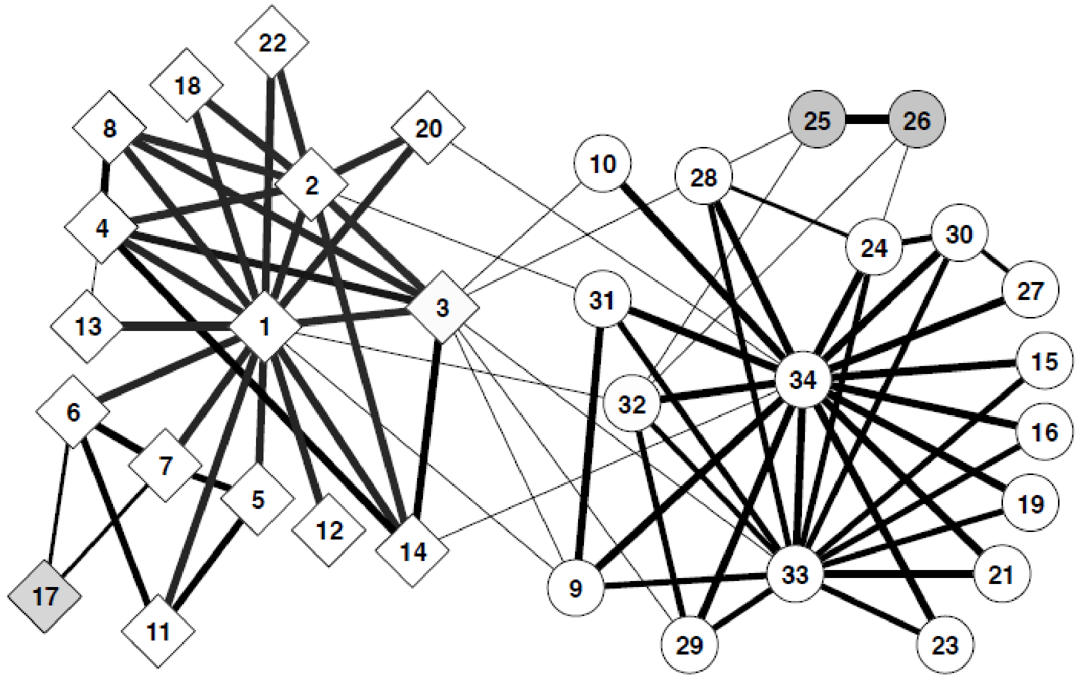
Another example

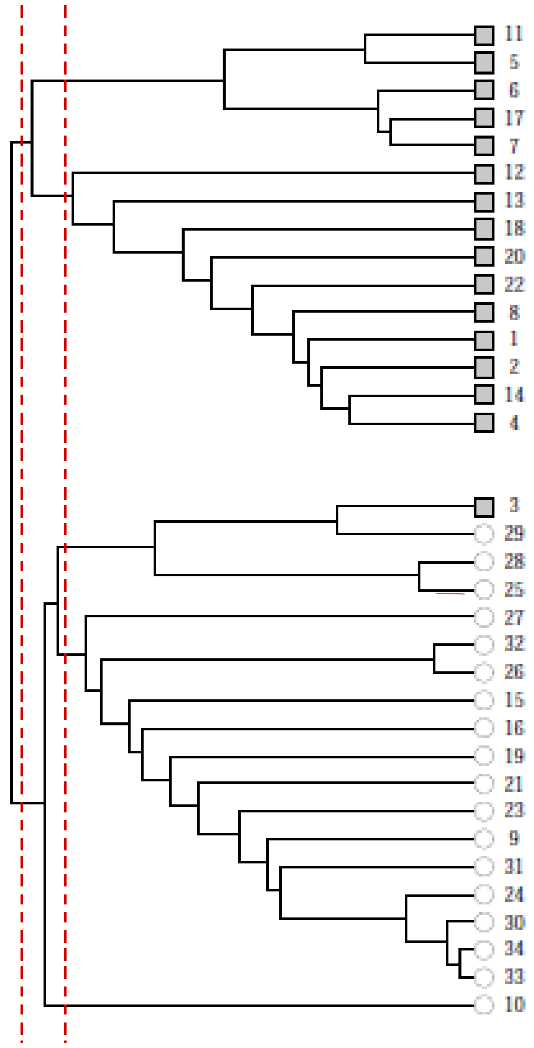


(b) *Step 2*

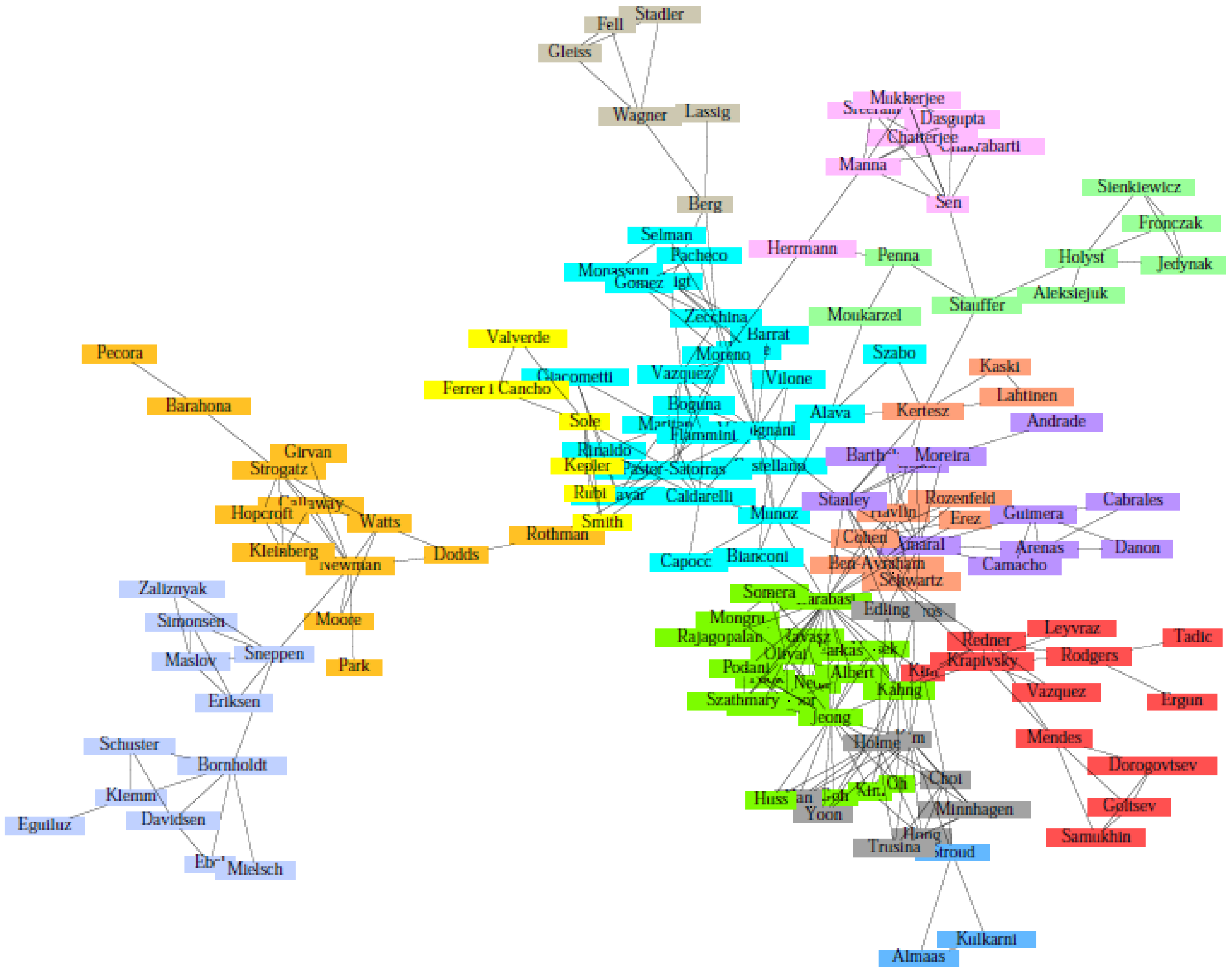
Girvan-Newman: Results

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





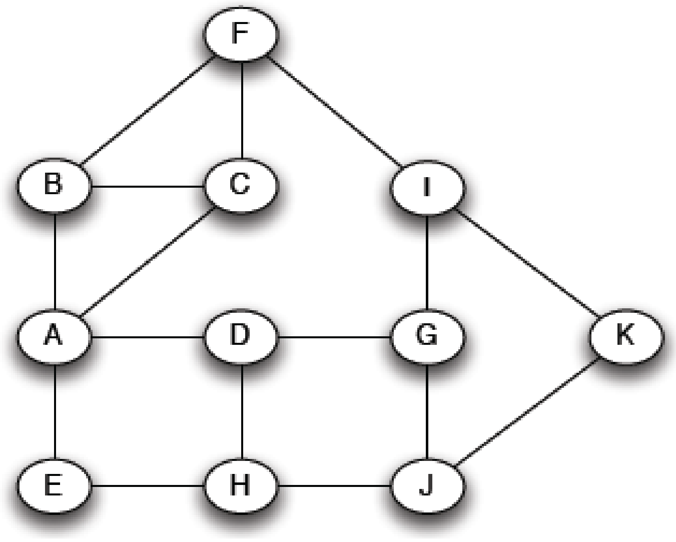
Girvan-Newman: Results



Communities in physics collaborations

How to Compute Betweenness?

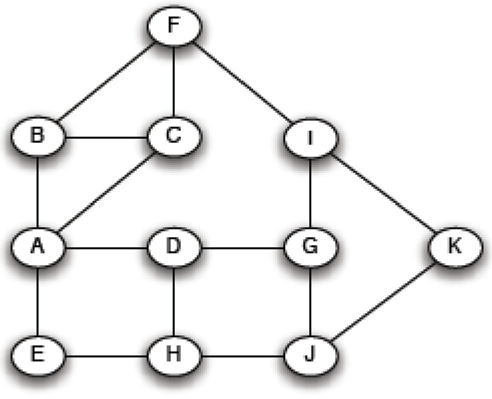
- Want to compute betweenness of paths starting at node A



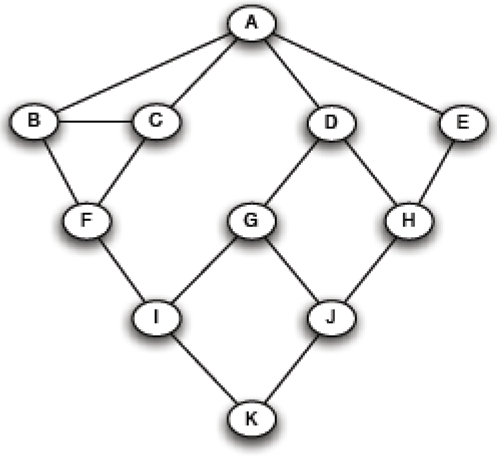
Computing Betweenness

1. Perform a *BFS* starting from A
2. Determine the *number of shortest path* from A to each other *node*
3. Based on these numbers, determine the amount of *flow* from A to all other nodes that uses each edge

Computing Betweenness: step 1



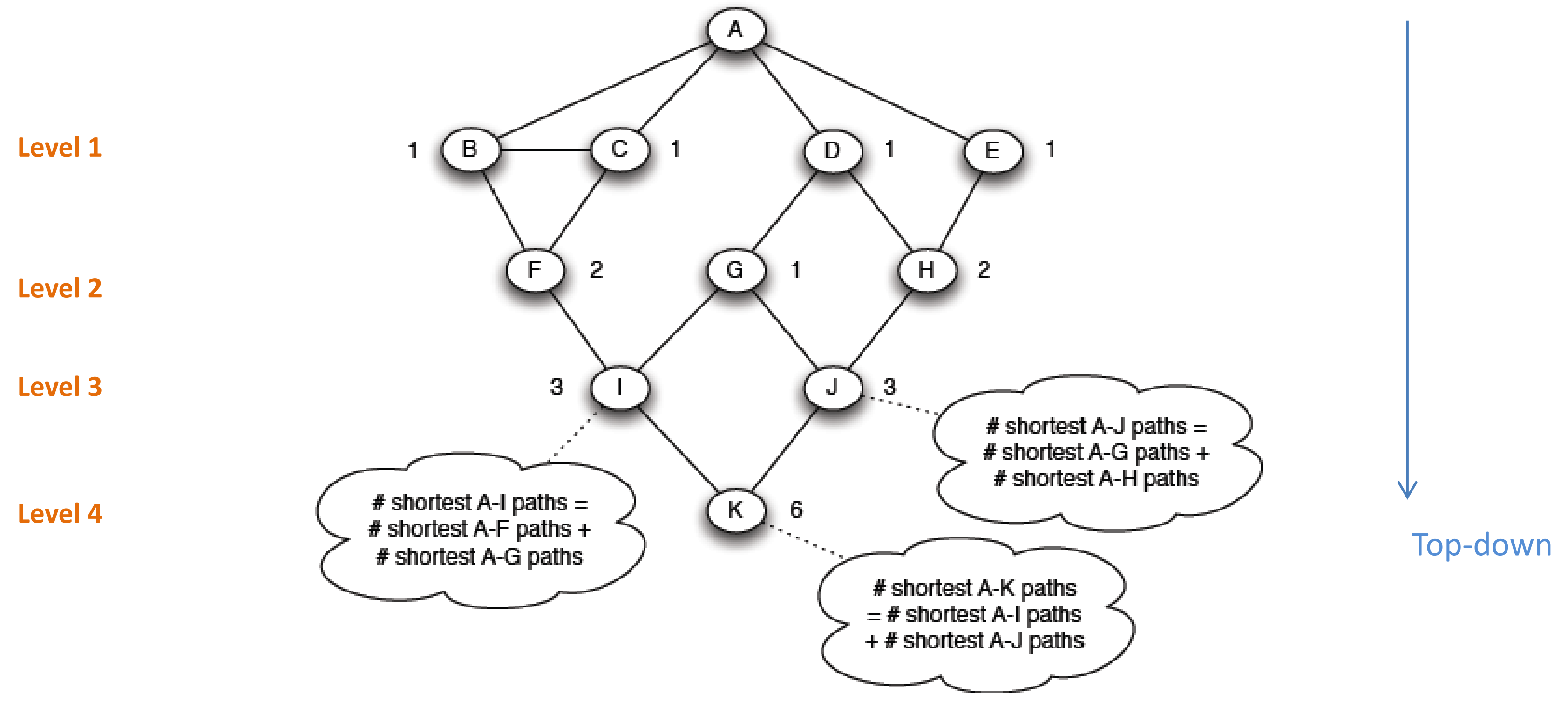
Initial network



BFS on A

Computing Betweenness: step 2

Count how many shortest paths from A to a specific node

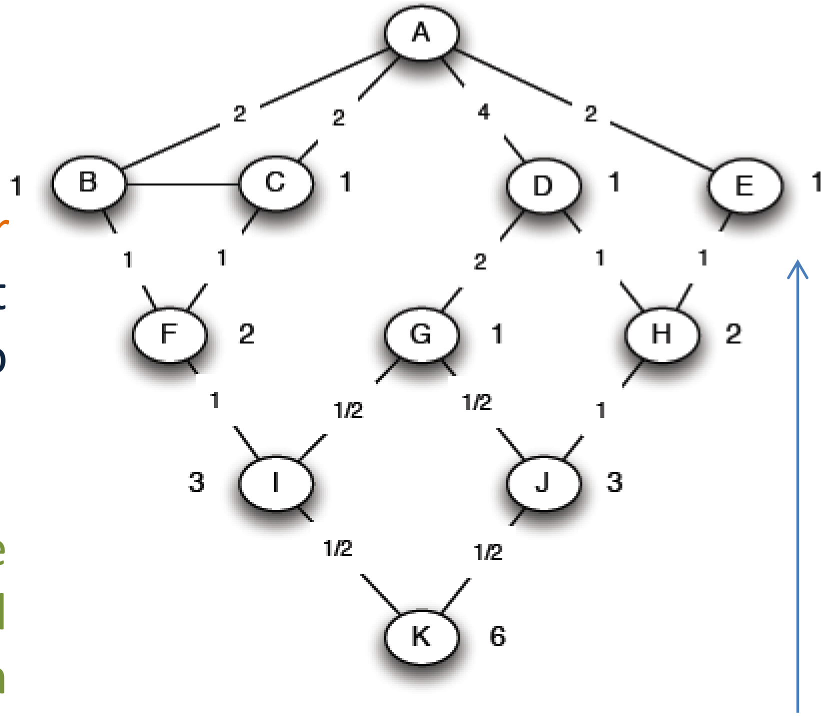

Level 1
Level 2
Level 3
Level 4
A
B 1
C 1
D 1
E 1
F 2
G 1
H 2
I 3
J 3
K 6
# shortest A-I paths = # shortest A-F paths + # shortest A-G paths
# shortest A-J paths = # shortest A-G paths + # shortest A-H paths
# shortest A-K paths = # shortest A-I paths + # shortest A-J paths
Top-down

Computing Betweenness: step 3

Compute betweenness by working up the tree: If there are multiple paths count them fractionally

For *each edge e*: calculate the sum *over all nodes Y* of the fraction of shortest paths *from the root A to Y* that go through e.

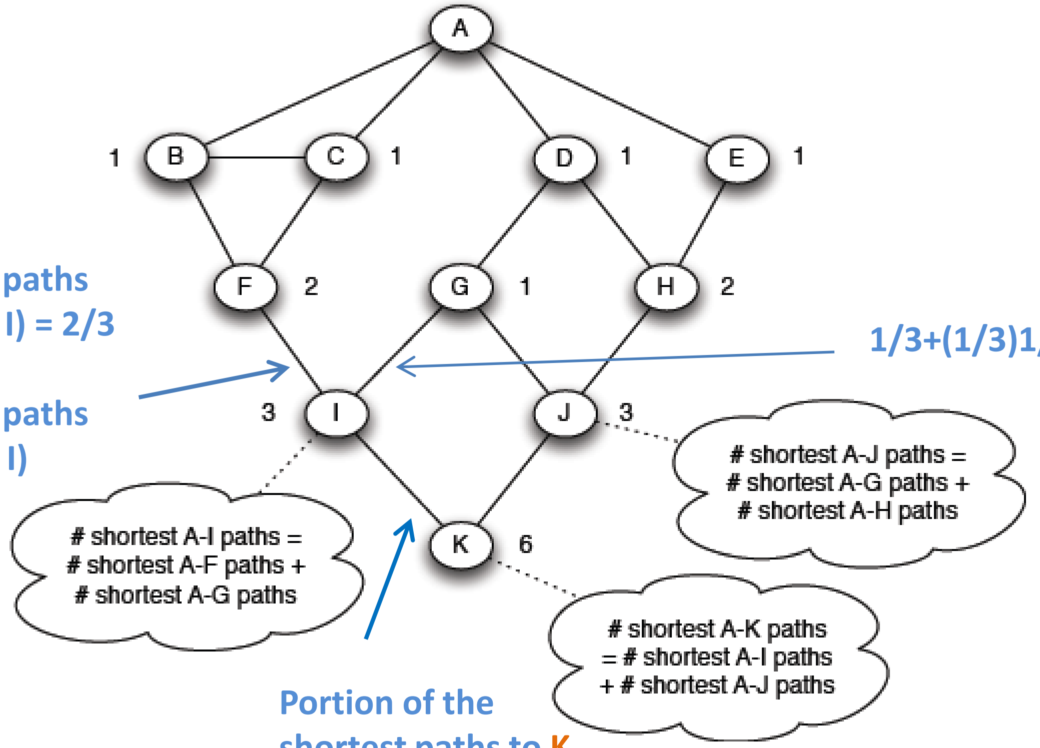
Each edge (X, Y) participates in the shortest-paths from the root to Y and to nodes (at levels) below Y -> Bottom up calculation



Computing Betweenness: step 3

Count the flow through each edge

$$credit(e) = \sum_{X,Y} \frac{|shortest-path(X,Y)\text{through } e|}{|shortest\_path(X,Y)\}|}$$



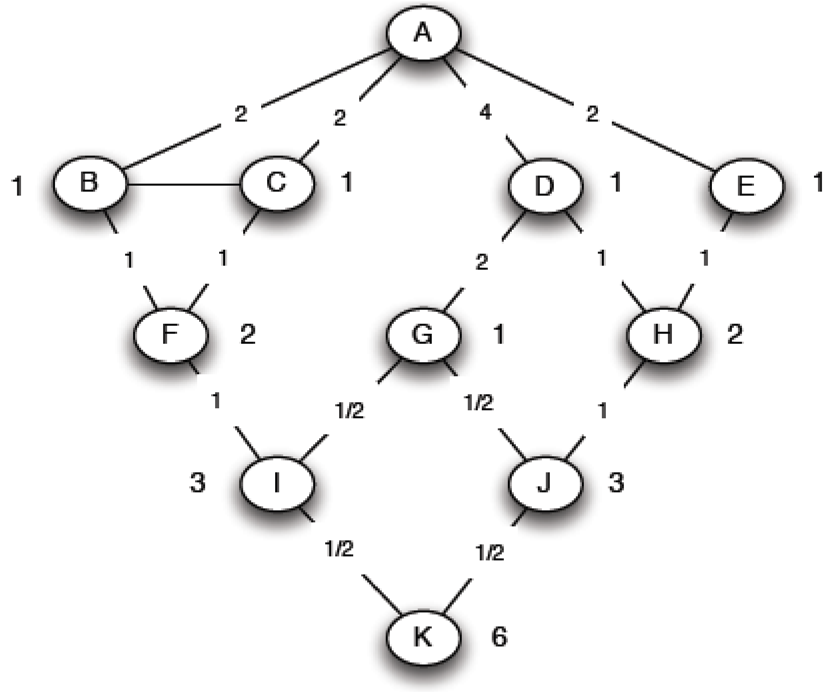
A
B
C
D
E
F
G
H
I
J
K
1
1
1
1
2
1
2
3
3
6
# shortest A-I paths = # shortest A-F paths + # shortest A-G paths
# shortest A-J paths = # shortest A-G paths + # shortest A-H paths
# shortest A-K paths = # shortest A-I paths + # shortest A-J paths

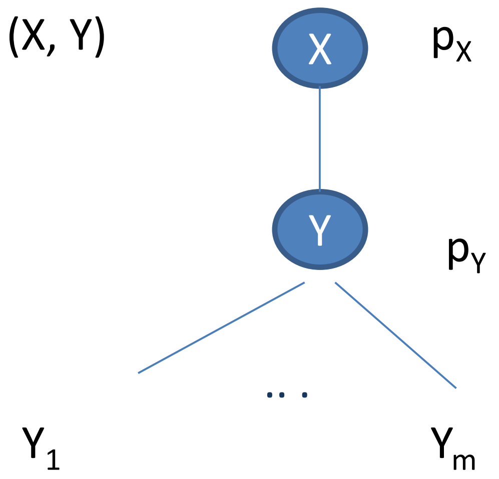
Portion of the shortest paths to I that go through (F, I) = 2/3
+
Portion of the shortest paths to K that go through (F, I)
(2/3) (1/2) = 1/3
= 1

1/3+(1/3)1/2 = 1/2

Portion of the shortest paths to K that go through (I, K) = 3/6 = 1/2

Computing Betweenness: step 3





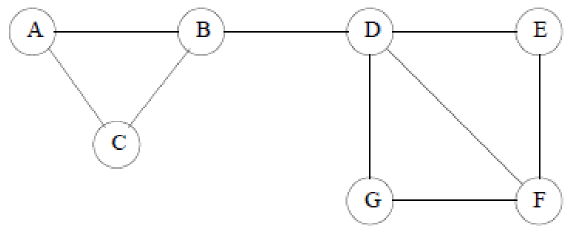
$$flow(X,Y) = p_X / p_Y + \sum_{Y_i child of Y} (p_X / p_Y)\, flow(Y, Y_i)$$

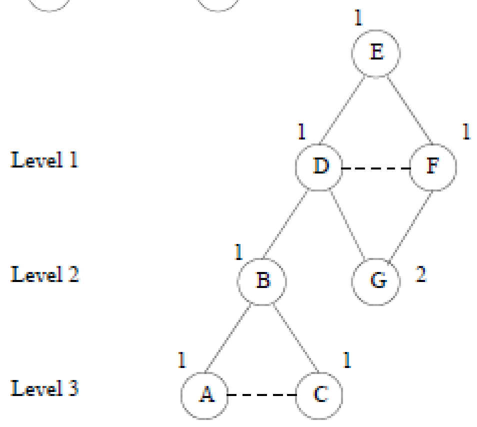
Computing Betweenness

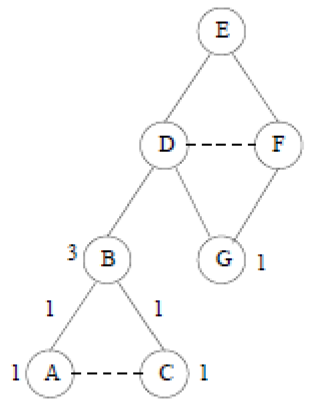
Repeat the process for all nodes

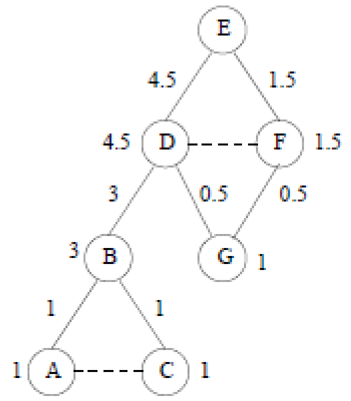
Sum over all BFSs

Example

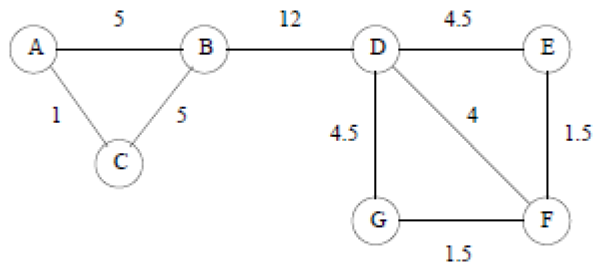

A
B
C
D
E
F
G

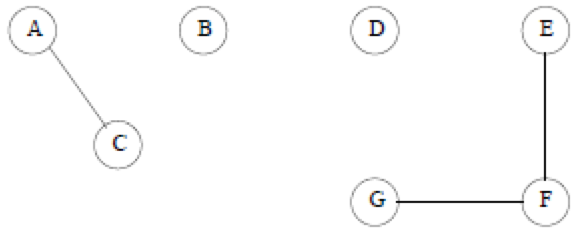

1
E
Level 1
1
D
F
1
Level 2
1
B
G
2
Level 3
1
A
C
1


E
D
F
3
B
G
1
1
1
1
A
C
1


E
4.5
1.5
4.5
D
F
1.5
3
0.5
0.5
3
B
G
1
1
1
1
A
C
1

Example





Computing Betweenness

Issues

- Test for connectivity?
- Re-compute all paths, or only those affected
- Parallel computation
- Sampling

Centrality measures

Degree centrality

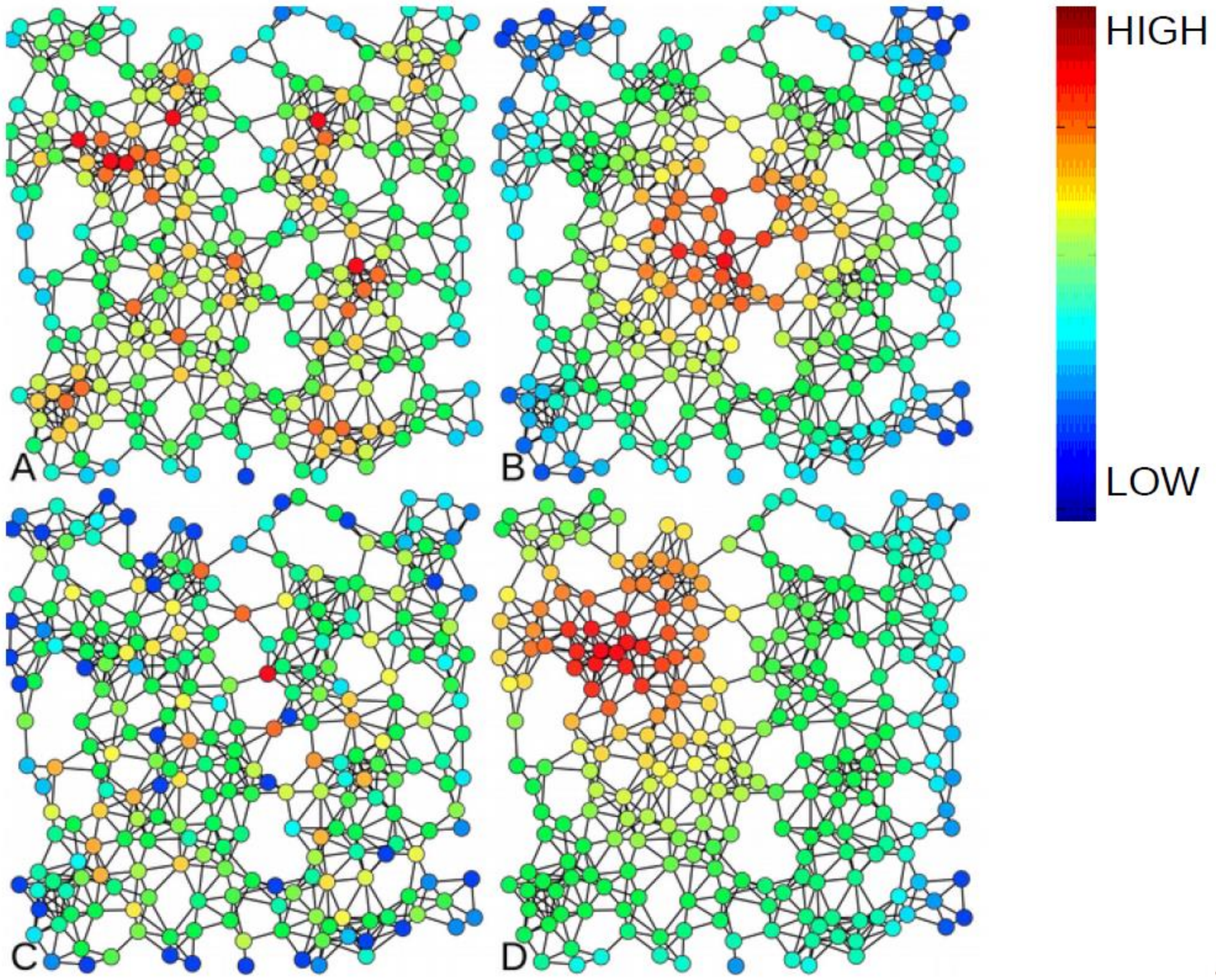
$$\text{closeness}(u) = \frac{1}{\sum_{v \in V, v \neq u} d(u, v)}$$

A: Degree

B: Closeness

C: Betweenness

D: PageRank

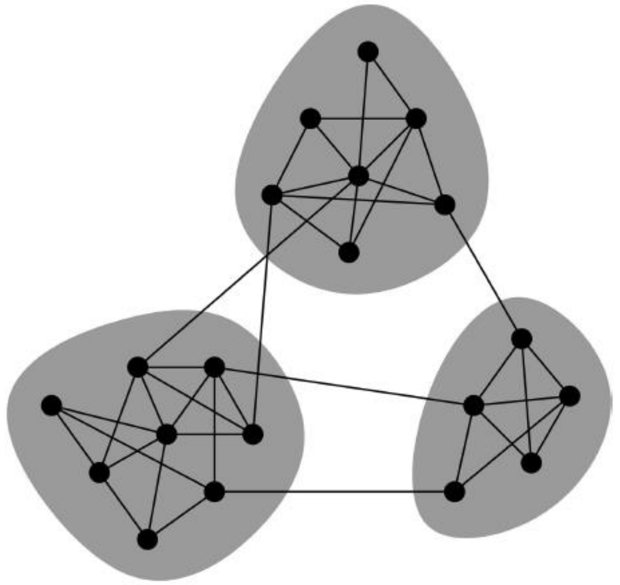


Outline

PART I

Modularity

- Communities: sets of **tightly connected nodes**
- Define: **Modularity *Q***
  - A measure of how well a network is partitioned into communities
  - Given a partitioning of the network into groups $s \in S$:



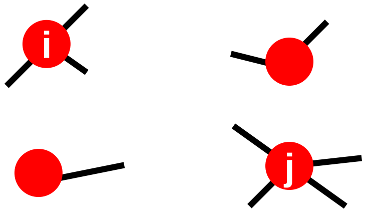
$$Q \propto \sum_{s \in S} \underbrace{[\,(\#\text{ edges within group } s) - (\text{expected }\#\text{ edges within group } s)\,]}_{\text{Need a null model!}}$$

Need a null model!

a copy of the original graph keeping some of its structural properties but without community structure

Null Model: Configuration Model

- Given real G on n nodes and m edges, construct rewired network G'
  - *Same degree distribution* but random connections
  - Consider $\boldsymbol{G'}$ as a multigraph
  - The expected number of edges between nodes i and j of degrees $\boldsymbol{d_i}$ and $\boldsymbol{d_j}$ equals to: $\boldsymbol{d_i} \cdot \frac{\boldsymbol{d_j}}{\boldsymbol{2m}} = \frac{\boldsymbol{d_i d_j}}{\boldsymbol{2m}}$



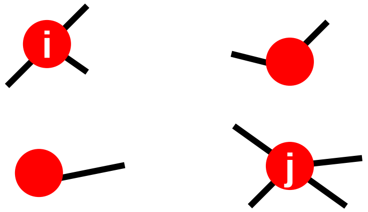
For any edge going out of i randomly, the probability of this edge getting connected to node j is $\frac{\boldsymbol{d_j}}{\boldsymbol{2m}}$

Because the degree for i is d$_i$*, we have d*$_i$ *number of such edges*

Note:

$$\sum_{u \in N} d_u = 2m$$

Null Model: Configuration Model



- The expected number of edges in (multigraph) **G'**:
  - $= \frac{1}{2}\sum_{i\in N}\sum_{j\in N}\frac{d_i d_j}{2m} = \frac{1}{2}\cdot\frac{1}{2m}\sum_{i\in N} d_i\left(\sum_{j\in N} d_j\right) =$
  - $= \frac{1}{4m}2m\cdot 2m = m$

Modularity

- Modularity of partitioning S of graph G:
  - $Q \propto \sum_{s \in S}$ [(# edges within group s) – (expected # edges within group s)]
  - $$Q(G,S) = \underbrace{\frac{1}{2m}}_{\text{Normalizing cost.: -1<Q<1}} \sum_{s \in S} \boxed{\sum_{i \in s} \sum_{j \in s} \left(A_{ij} - \frac{d_i d_j}{2m}\right)}$$

$A_{ij} = 1$ if $i \rightarrow j$, 0 else

- Modularity values take range [−1, 1]
  - It is positive if the number of edges within groups exceeds the expected number
  - 0.3-0.7 < Q means significant community structure

Modularity

Greedy method of Newman (one of the many ways to use modularity)

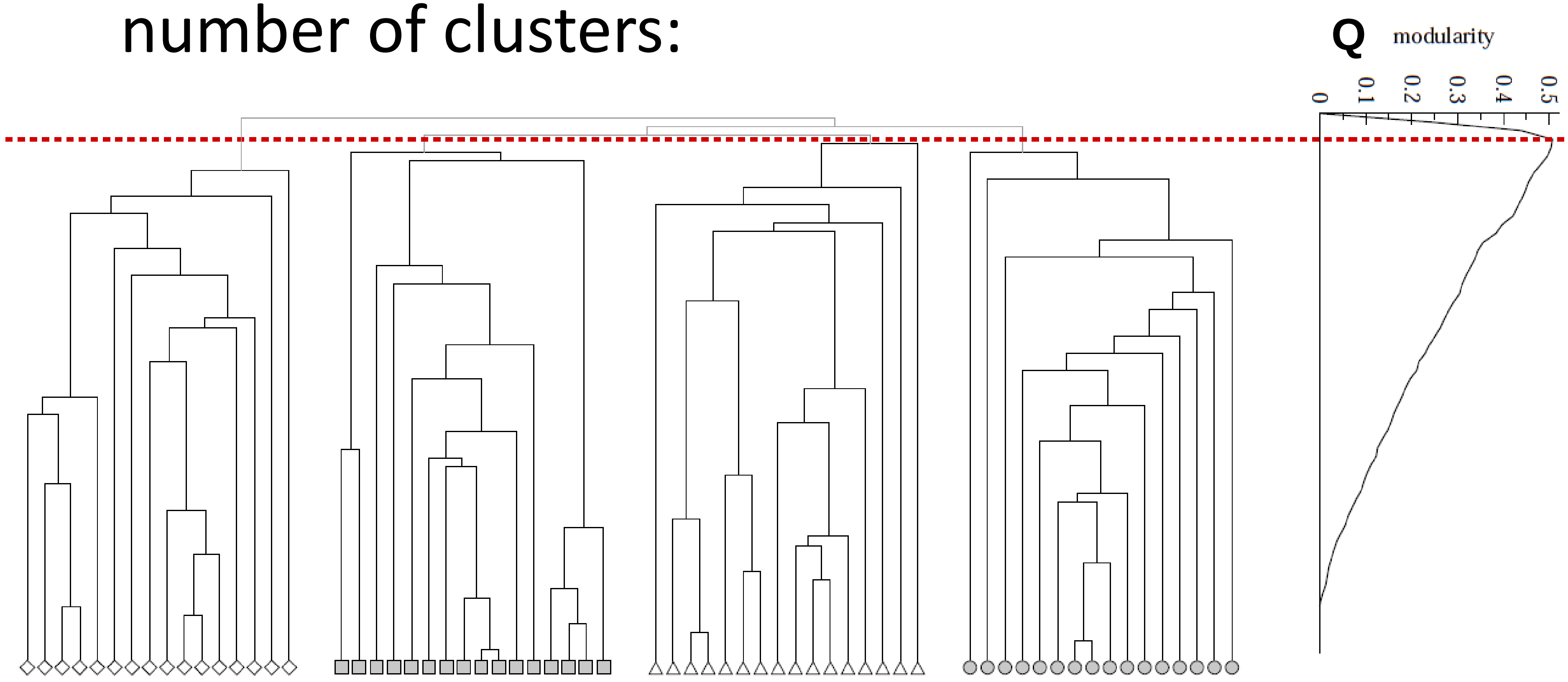
Agglomerative hierarchical clustering method

1. Start with a state in which each vertex is the sole member of one of *n* communities
2. Repeatedly join communities together in pairs, choosing at each step the join that results in the greatest increase (or smallest decrease) in Q.

Since the joining of a pair of communities between which there are no edges can never result in an increase in modularity, we ***need only consider those pairs between which there are edges***, of which there will at any time be at most m

Modularity: Number of clusters

- Modularity is useful for selecting the number of clusters:



Modularity: Cluster quality

When a given clustering is “good”?

Also, it is both a local (per individual cluster) and global measure

Outline

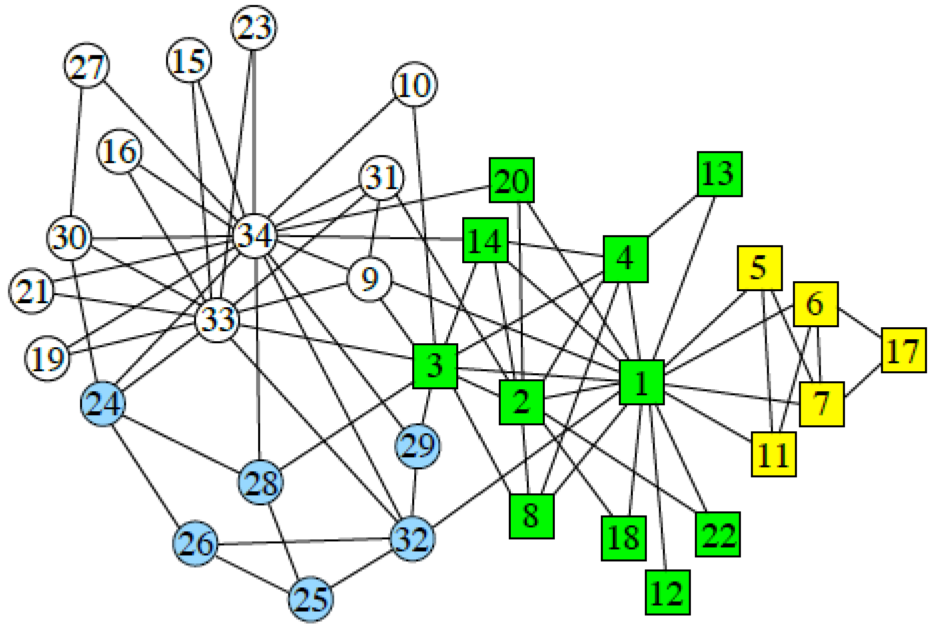
PART I

1. Introduction: what, why, types?
2. Cliques
3. Background: cluster analysis (node/edge similarity)
4. Hierarchical clustering (betweenness)
5. Modularity
6. How to evaluate

Community Evaluation

- With ground truth
- Without ground truth

Evaluation with ground truth



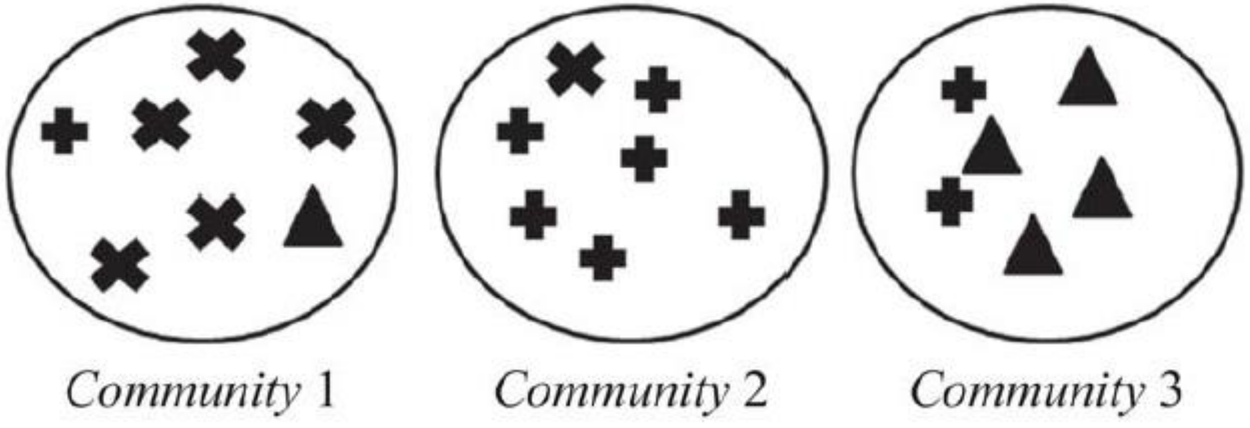
Zachary’s Karate Club

Club president (34) (circles) and instructor (1) (rectangles)

Metrics: purity

the fraction of instances that have labels equal to the label of the community's majority

$$Purity = \frac{1}{N} \sum_{i=1}^{k} \max_j |C_i \cap L_j|$$



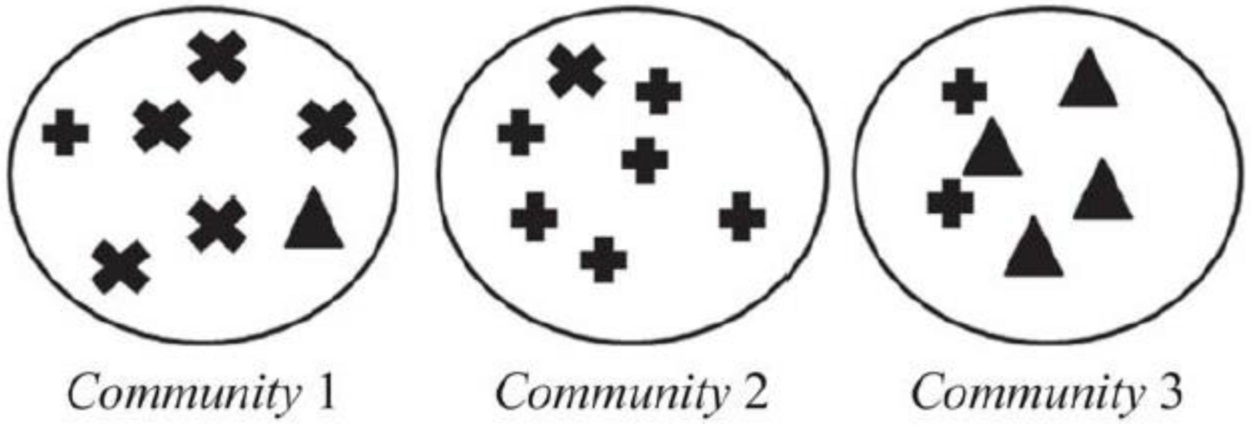
(5+6+4)/20 = 0.75

Metrics

Based on pair counting: the number of pairs of vertices which are classified in the same (different) clusters in the two partitions.

- True Positive (**TP**) Assignment: when similar members are assigned to the same community. This is a correct decision.
- True Negative (**TN**) Assignment: when dissimilar members are assigned to different communities. This is a correct decision.
- False Negative (**FN**) Assignment: when similar members are assigned to different communities. This is an incorrect decision.
- False Positive (**FP**) Assignment: when dissimilar members are assigned to the same community. This is an incorrect decision.

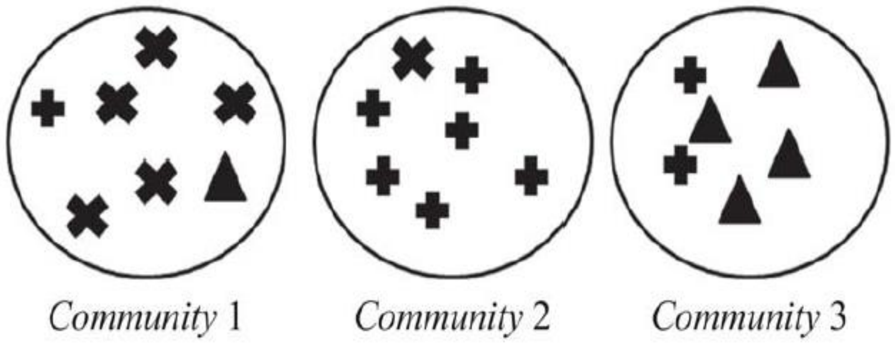
Metrics: pairs



For TP, we need to compute the number of pairs with the same label that are in the same community

$$TP = \underbrace{\binom{5}{2}}_{\textit{Community 1}} + \underbrace{\binom{6}{2}}_{\textit{Community 2}} + \underbrace{\left(\binom{4}{2} + \binom{2}{2}\right)}_{\textit{Community 3}} = 32$$

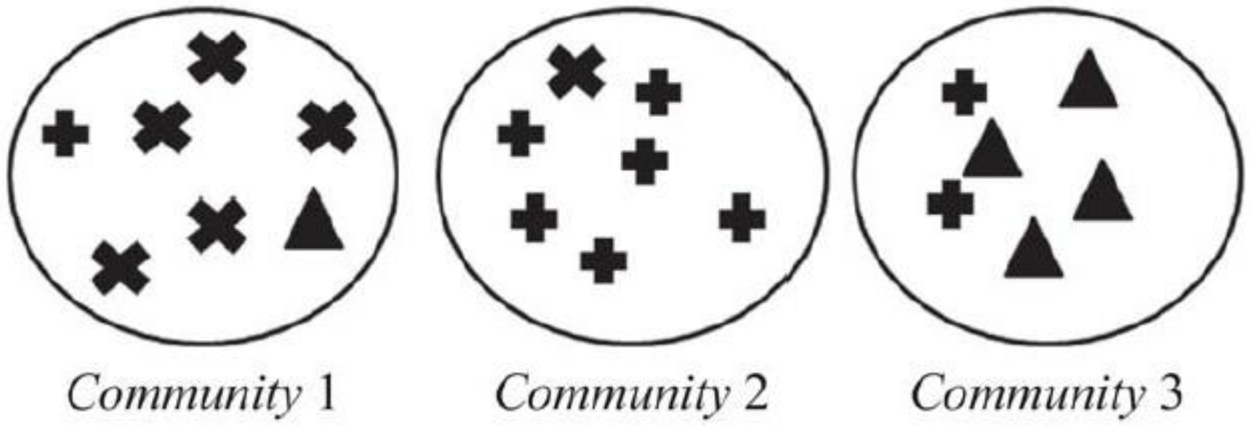
Metrics: pairs



For TN: compute the number of dissimilar pairs in dissimilar communities

$$
\begin{aligned}
TN \;=\; & \underbrace{(\overbrace{5\times 6}^{\times,+} + \overbrace{1\times 1}^{+,\times} + \overbrace{1\times 6}^{\Delta,+} + \overbrace{1\times 1}^{\Delta,\times})}_{\textit{Communities 1 and 2}} \\
& + \underbrace{(\overbrace{5\times 4}^{\times,\Delta} + \overbrace{5\times 2}^{\times,+} + \overbrace{1\times 4}^{+,\Delta} + \overbrace{1\times 2}^{\Delta,+})}_{\textit{Communities 1 and 3}} \\
& + \underbrace{(\overbrace{6\times 4}^{+,\Delta} + \overbrace{1\times 2}^{\times,+} + \overbrace{1\times 4}^{\times,\Delta}}_{\textit{Communities 2 and 3}} = 104.
\end{aligned}
$$

Metrics: pairs



For FP, compute dissimilar pairs that are in the same community.

$$FP = \underbrace{(5\times 1 + 5\times 1 + 1\times 1)}_{Community\ 1} + \underbrace{(6\times 1)}_{Community\ 2} + \underbrace{(4\times 2)}_{Community\ 3} = 25$$

For FN, compute similar members that are in different communities.

$$FN = \underbrace{(5\times 1)}_{\times} + \underbrace{(6\times 1 + 6\times 2 + 2\times 1)}_{+} + \underbrace{(4\times 1)}_{\triangle} = 29$$

Metrics: pairs

Precision (P): the fraction of pairs that have been correctly assigned to the same community.

TP/(TP+FP)

Recall (R): the fraction of pairs assigned to the same community of all the pairs that should have been in the same community.

TP/(TP+FN)

F-measure

2PR/(P+R)

Evaluation without ground truth

- Cluster Cohesion: Measures how closely related are objects in a cluster
- Cluster Separation: Measure how distinct or well-separated a cluster is from other clusters
- Example: Squared Error
  - Cohesion is measured by the within cluster sum of squares (SSE)

$$\boldsymbol{WSS} = \sum_{\boldsymbol{i}} \sum_{\boldsymbol{x} \in \boldsymbol{C_i}} (\boldsymbol{x} - \boldsymbol{m_i})^2$$

  - Separation is measured by the between cluster sum of squares

$$BSS = \sum_i |C_i| (m - m_i)^2$$

  - Where $|C_i|$ is the size of cluster i

Evaluation without ground truth

$$\delta_{int}(C) = \frac{\#\text{ internal edges of } C}{n_c(n_c - 1)/2}$$

$$\delta_{ext}(C) = \frac{\#\text{ inter-cluster edges of } C}{n_c(n - n_c)}$$

Evaluation without ground truth

Modularity

Both as a local (per individual community) and as a global measure

Evaluation without ground truth

With semantics:

- (ad hoc) analyze other attributes (e.g., profile, content generated) for coherence
- human subjects (user study) Mechanical Turk

Visual representation (similarity/adjacency matric, word clouds, etc)

States
President
United
Congress
State
Amendment
Representatives
Constitution
House
Senate

(a) U.S . Constitution

racing
sports
football
hockey
Sports
games
handball
Football
Rugby
tennis

(b) Sports

Basic References

- Jure Leskovec, Anand Rajaraman, Jeff Ullman, Mining of Massive Datasets, Chapter 10, http://www.mmds.org/
- Reza Zafarani, Mohammad Ali Abbasi, Huan Liu, Social Media Mining: An Introduction, Chapter 6, http://dmml.asu.edu/smm/
- Santo Fortunato: Community detection in graphs. CoRR abs/0906.0612v2 (2010)
- Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Introduction to Data Mining, Chapter 8, http://www.users.cs.umn.edu/~kumar/dmbook/index.php
- Albert-László Barabasi, Network Science, Chapter 9, http://networksciencebook.com/

Questions?