

# Hierarchical clustering

Introduction to Network Science

Carlos Castillo

Topic 14

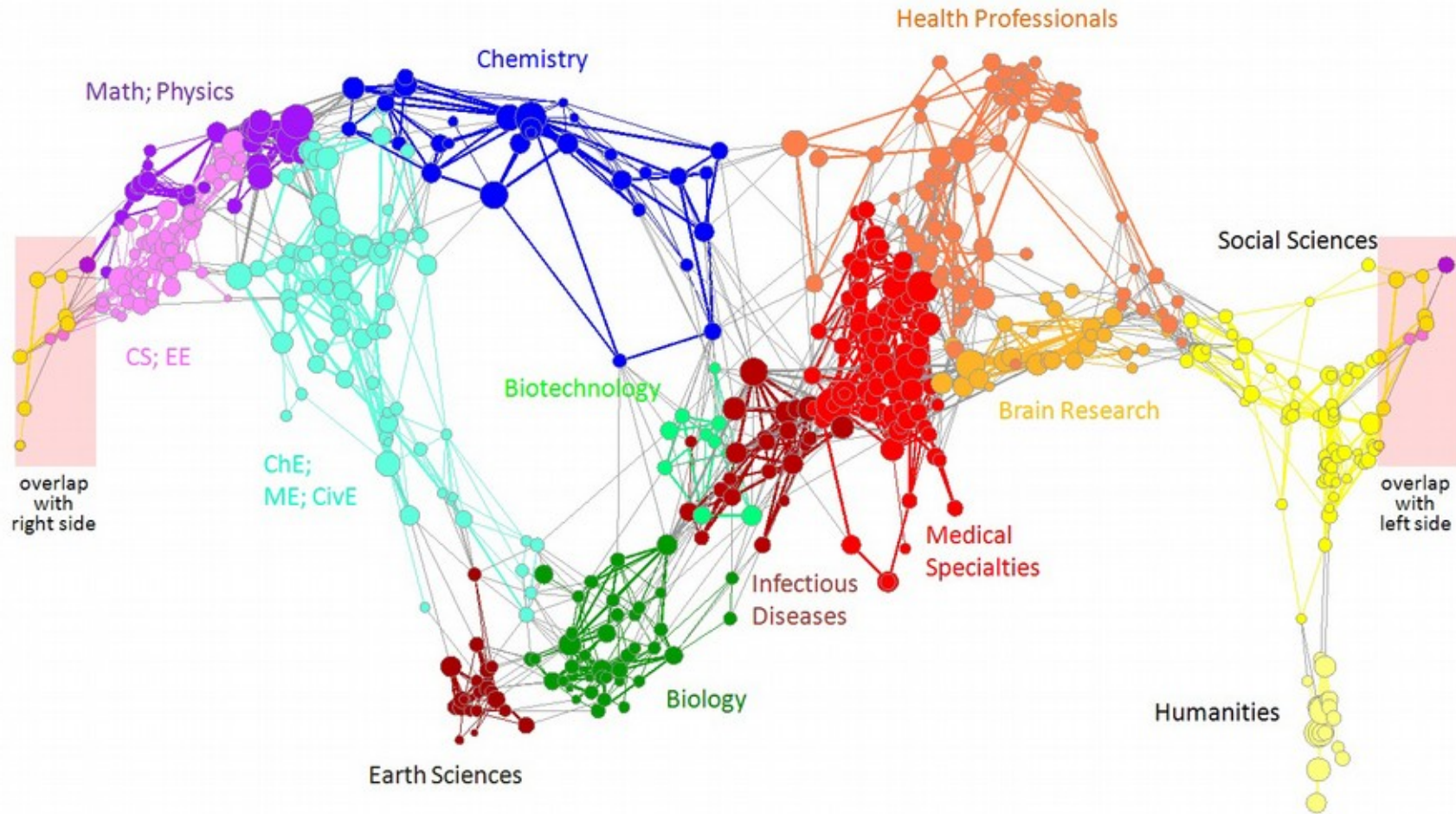
# Sources

- Barabási 2016 Chapter 9
- [Networks, Crowds, and Markets](#) Ch 3
- C. Castillo (2017) [Dense Sub-Graphs and Graph partitioning](#)

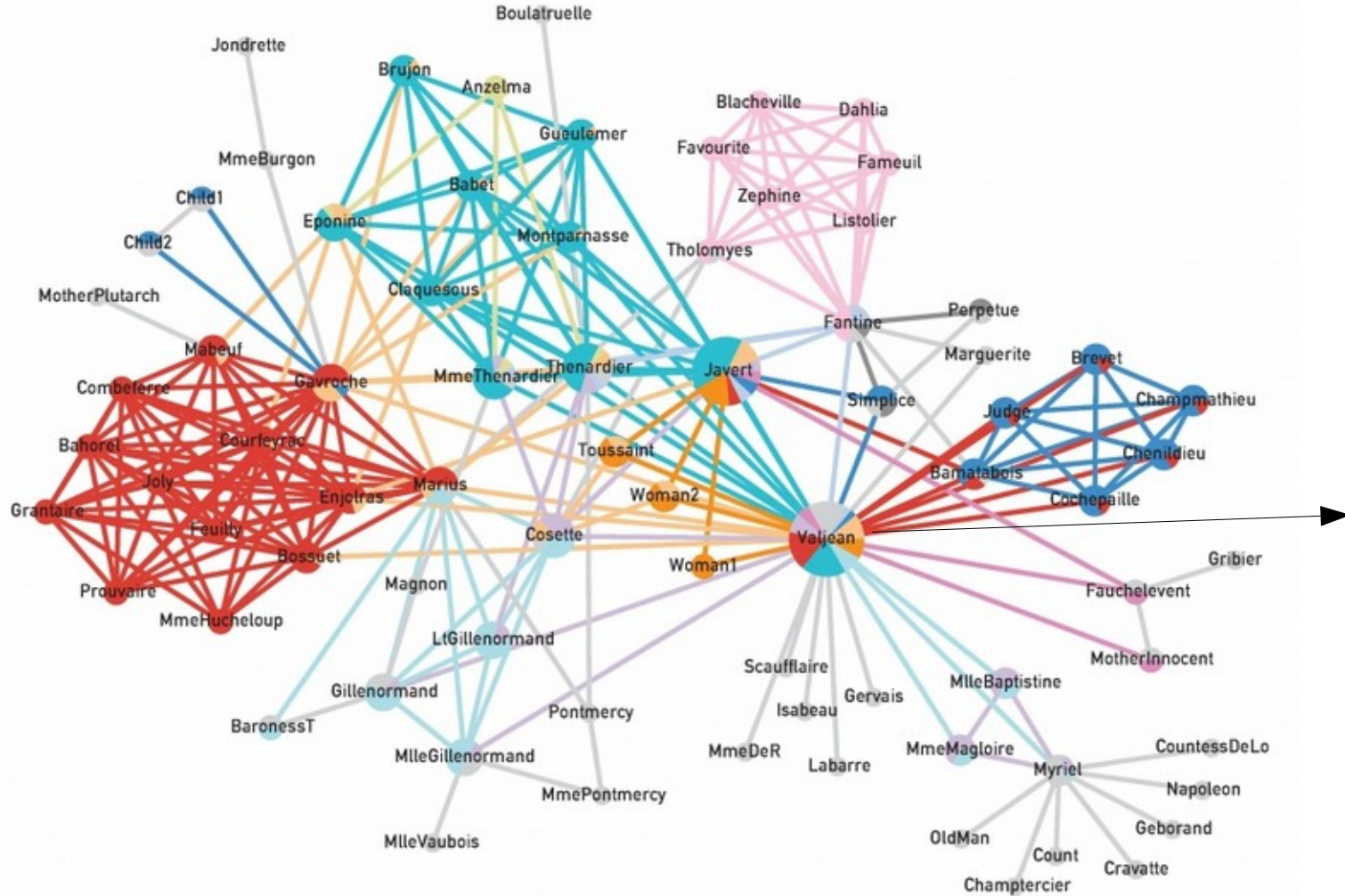
# Communities

2 communities	[previous topic]
1 community	[previous topic]
3+ communities	<b>[this topic]</b>

# Citations across scientific disciplines

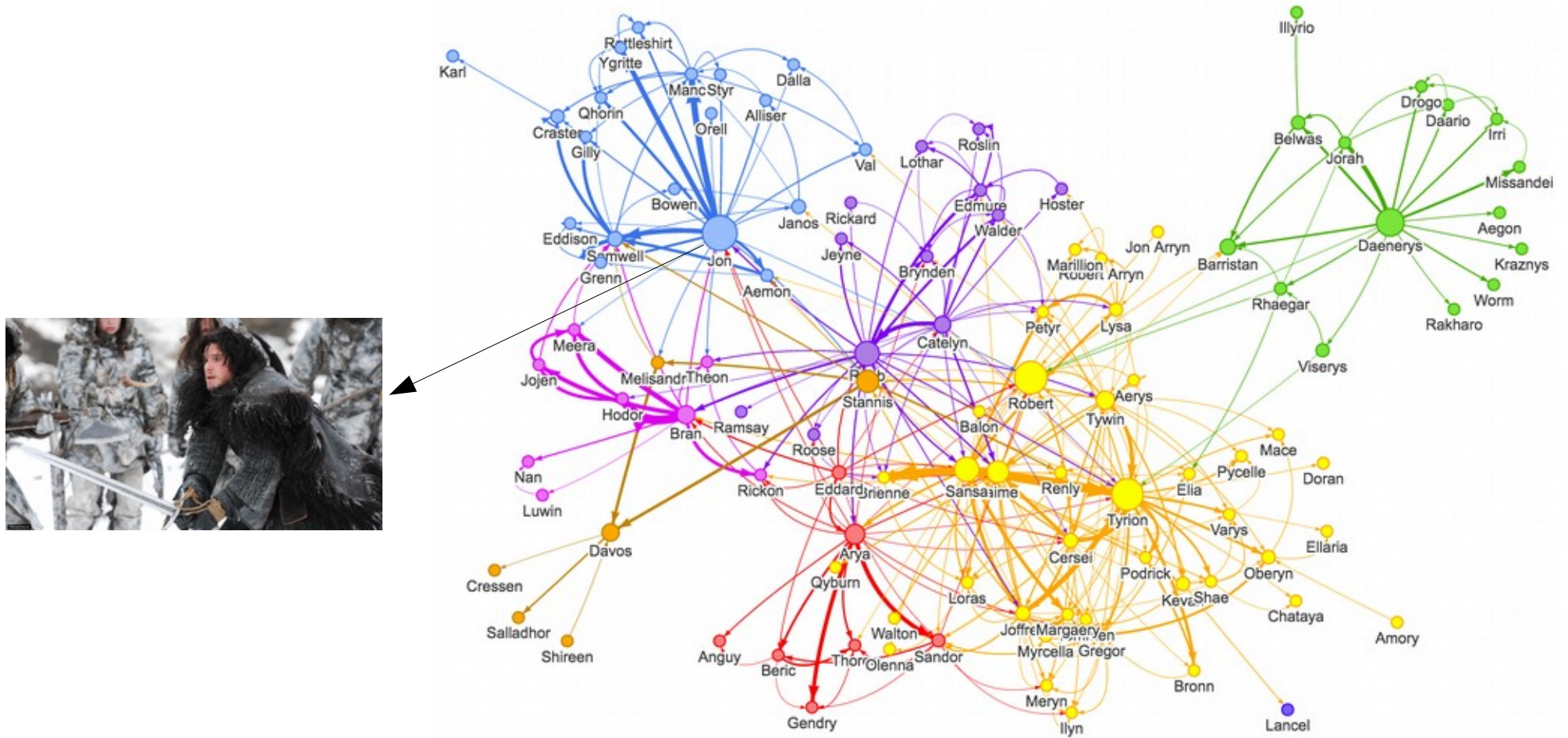


# Characters in *Les Misérables* (1862)

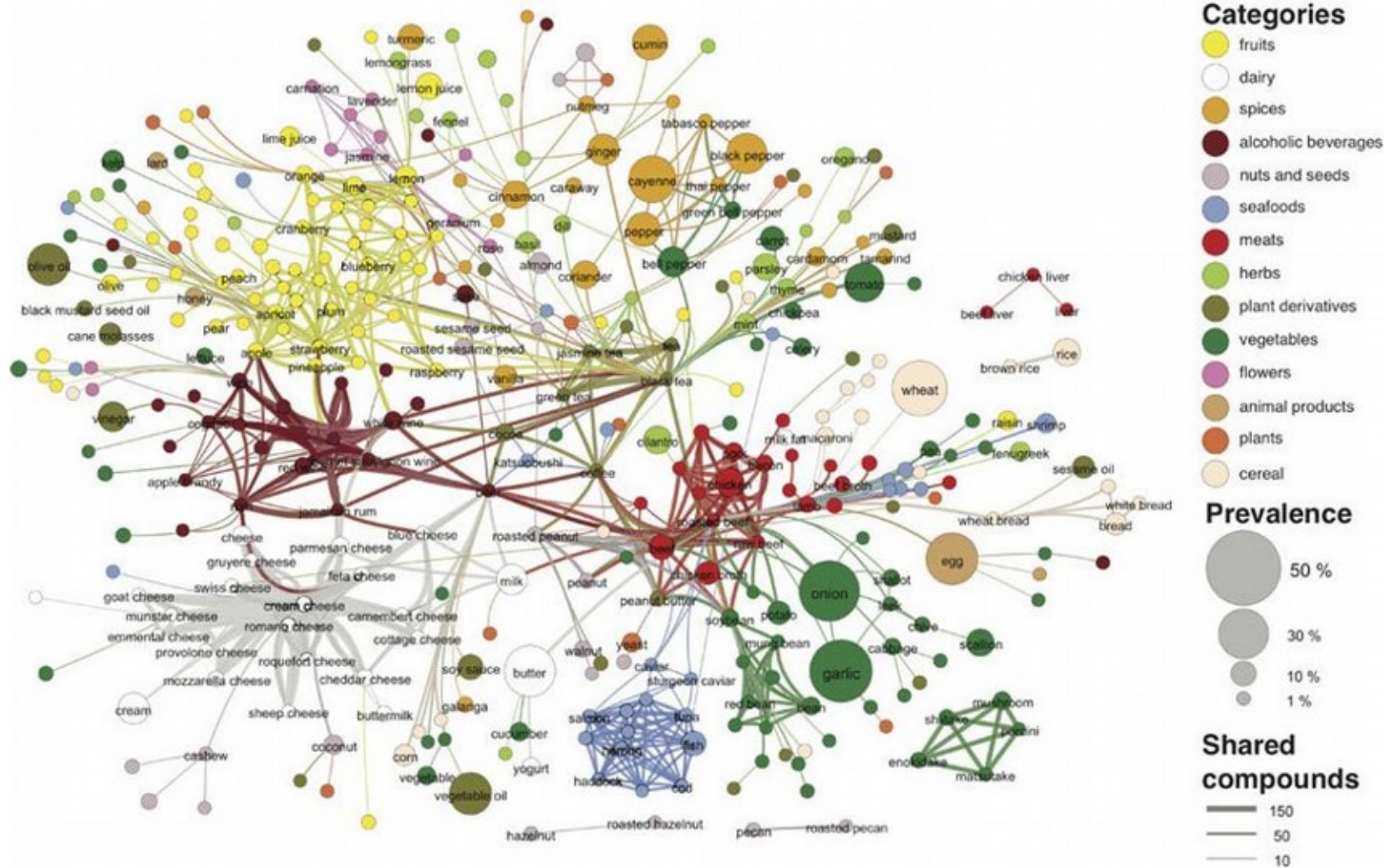




# Characters in *A Storm of Swords* (2000)

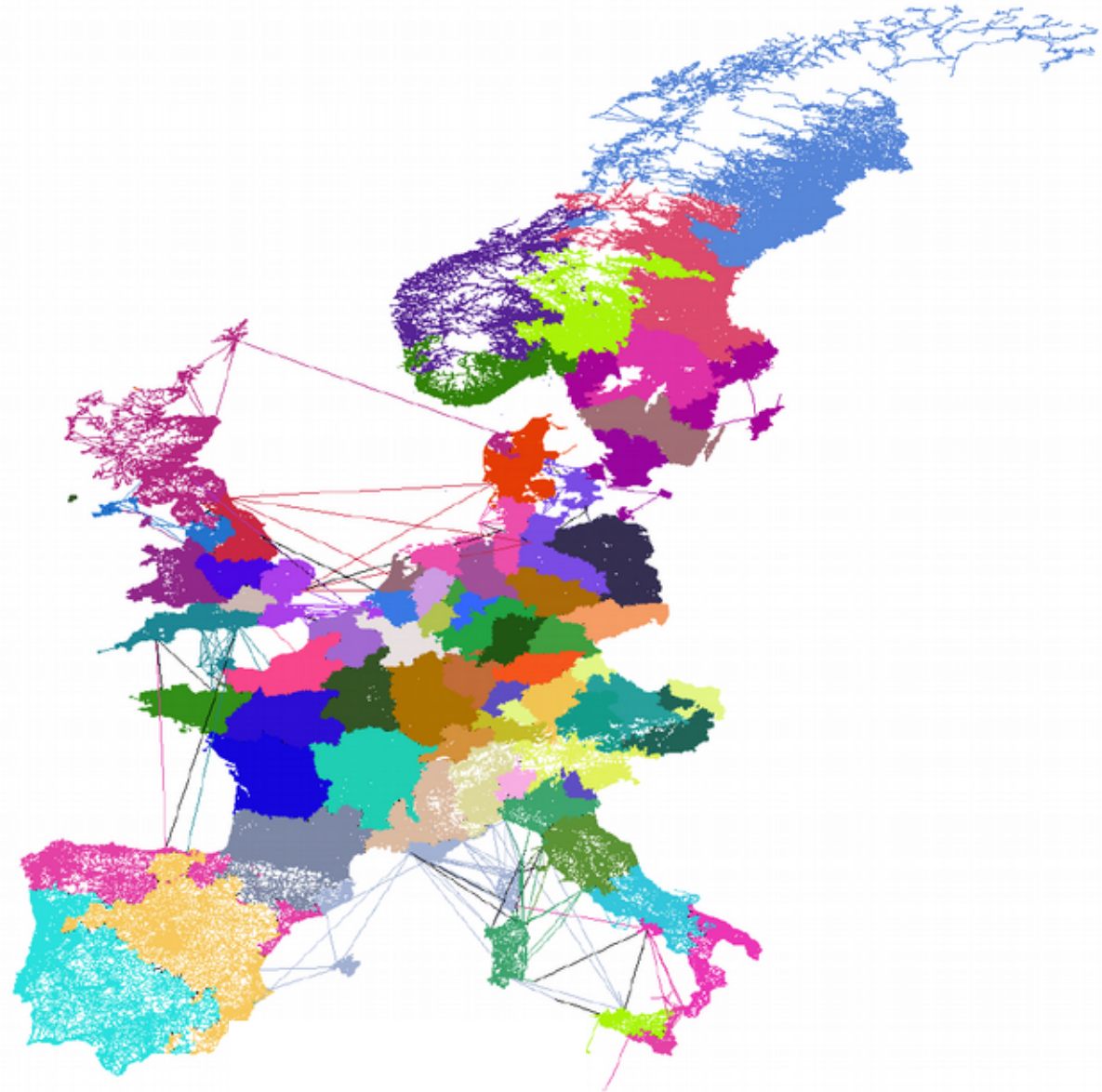


# Food pairing





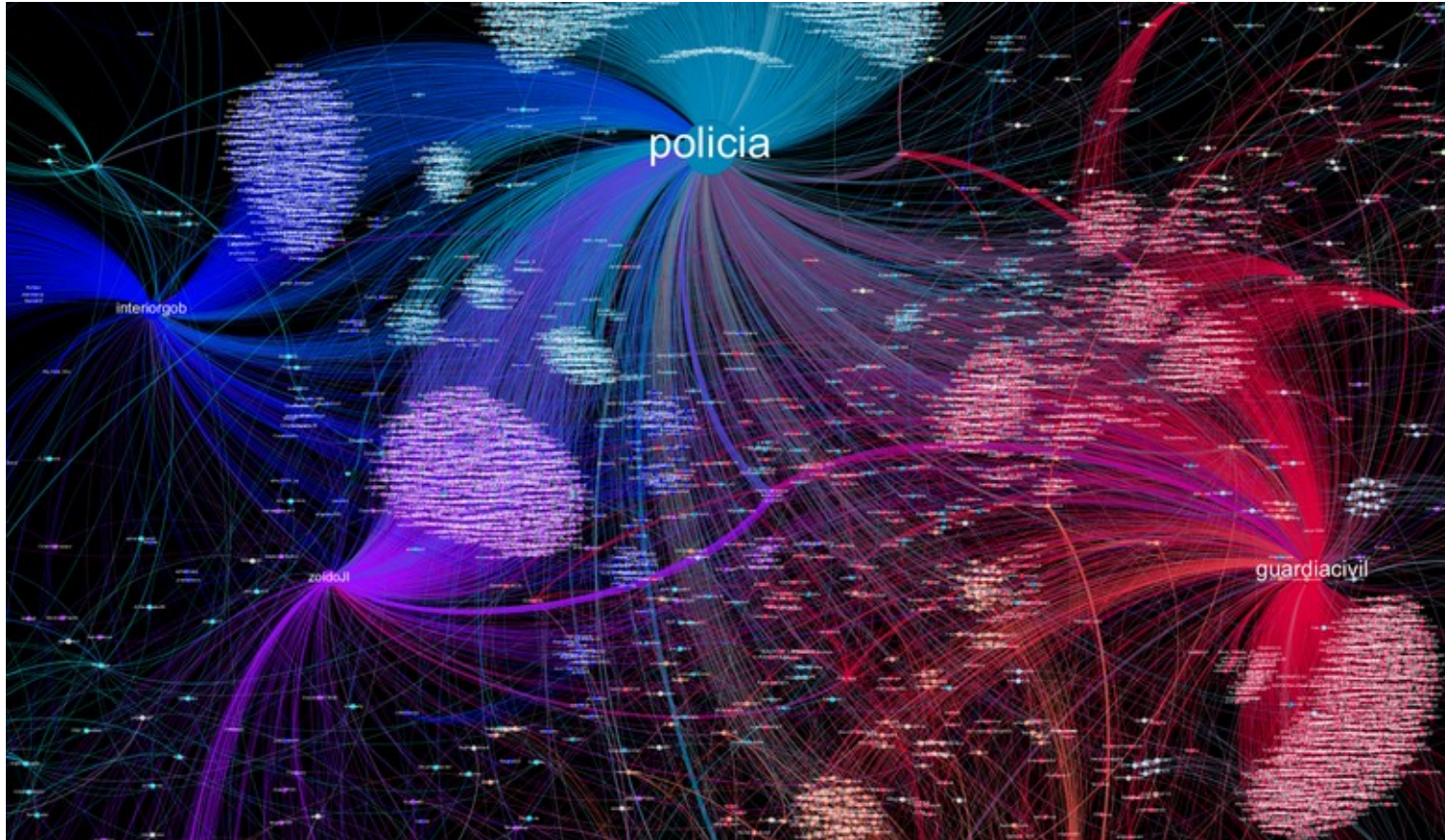
# Street network in Europe





# #estamosporti (Oct 1-3 2017)

## A state-sponsored hashtag



*Because there will be no referendum*  
*Because no one is above the law*  
*Because the unity of Spain is indissoluble*  
*#EstamosporTI is already a trend*

Source: Erin Gallagher

# Community hypotheses

H1. The community structure of a network is uniquely encoded by its links [discoverable]

H2. A community is a locally dense connected subgraph [dense, connected]

# Weak < Strong < Clique

- Let  $C$  be a community with  $N_C$  nodes
- Given a community  $C$ ,
  - Let  $k_i^{\text{int}}$  count links towards  $C$ ,  $k_i^{\text{ext}}$  towards  $V \setminus C$
- **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:**  $\sum_{i \in C} k_i^{\text{int}} > \sum_{i \in C} k_i^{\text{ext}}$

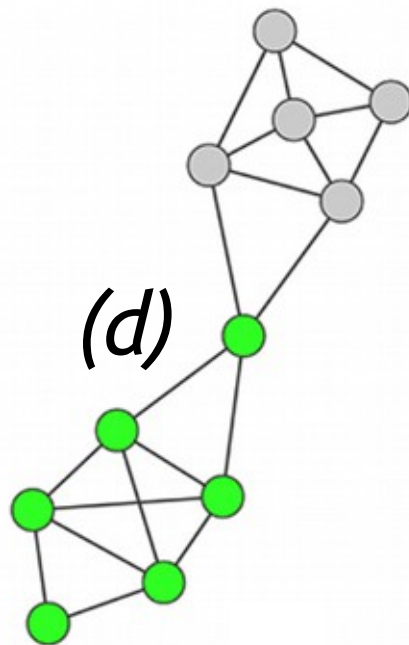
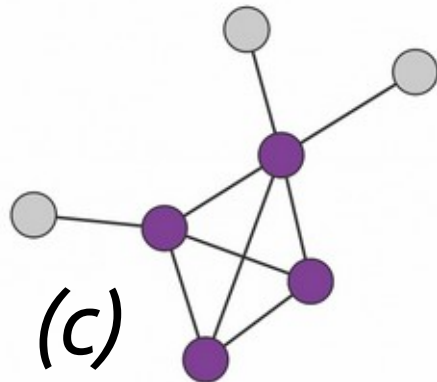
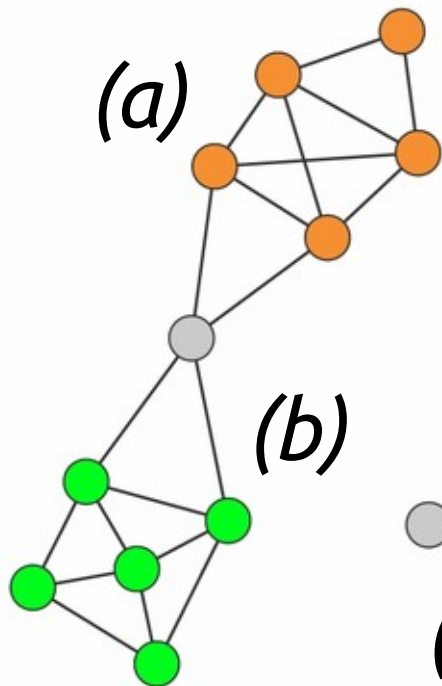
# Try it!

*Which types of communities are these?*

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

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

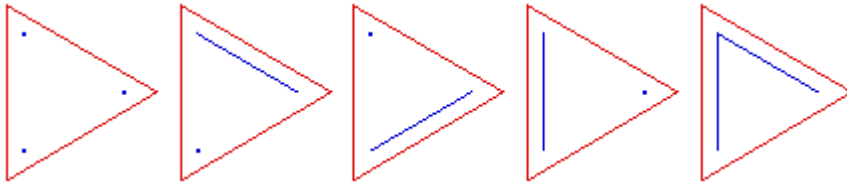
**Weak community:**  $\sum_{i \in C} k_i^{\text{int}} > \sum_{i \in C} k_i^{\text{ext}}$



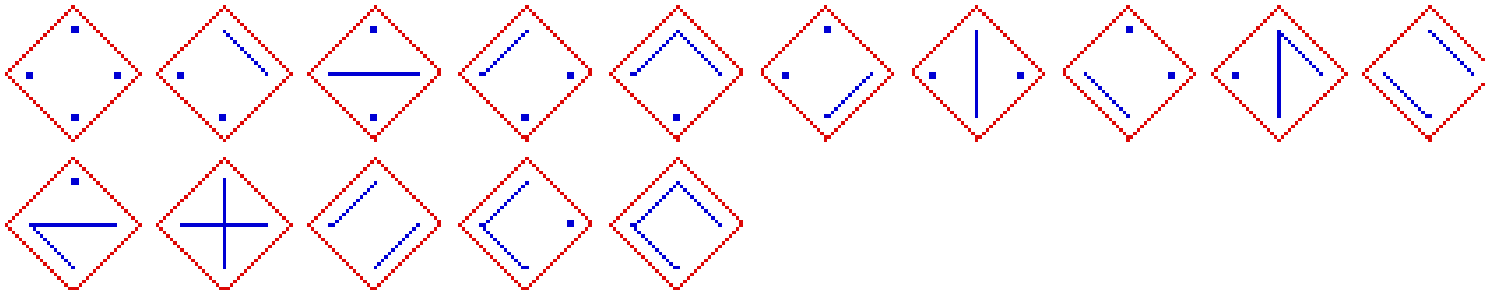


# How many possible partitions?

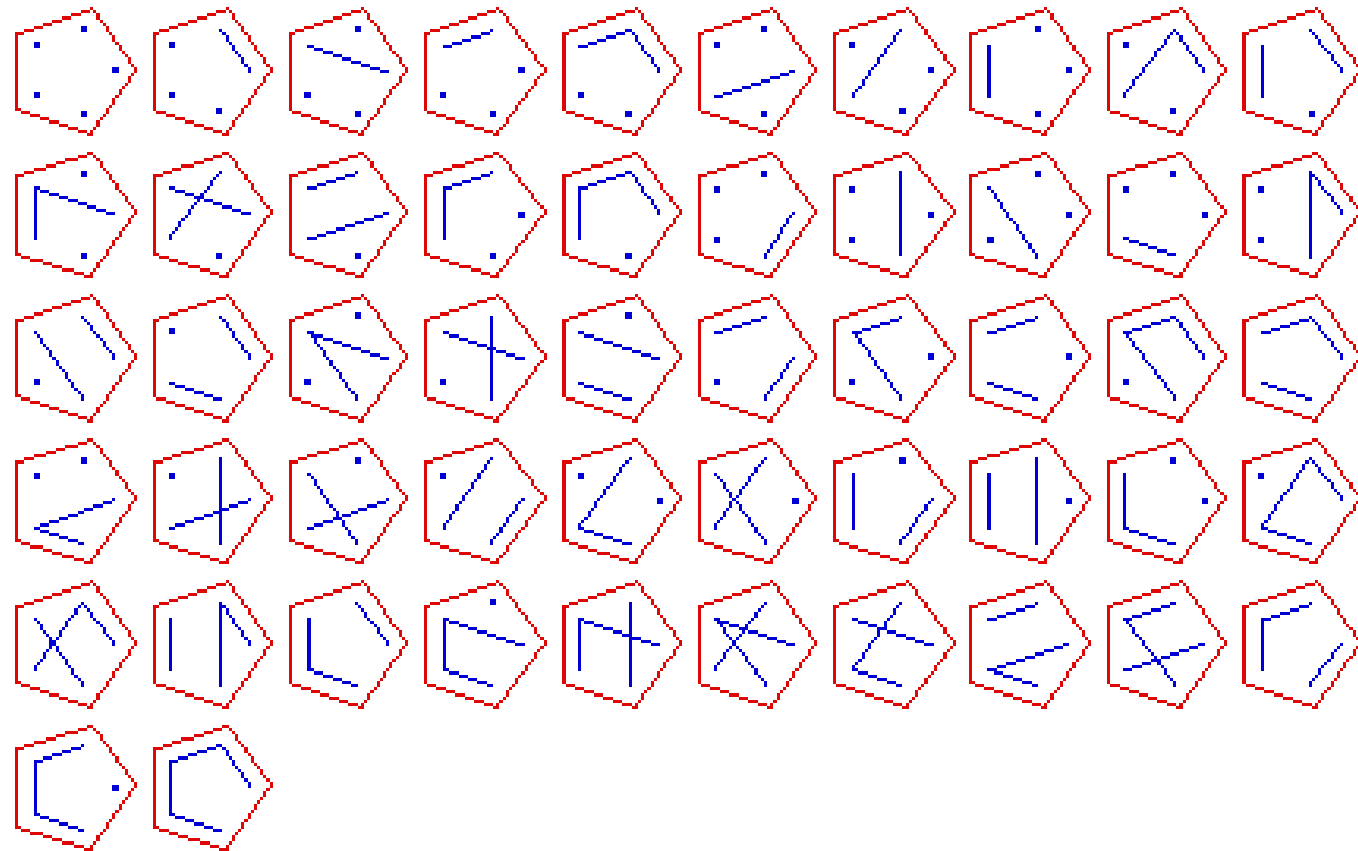
- With 3 nodes: 5 partitions

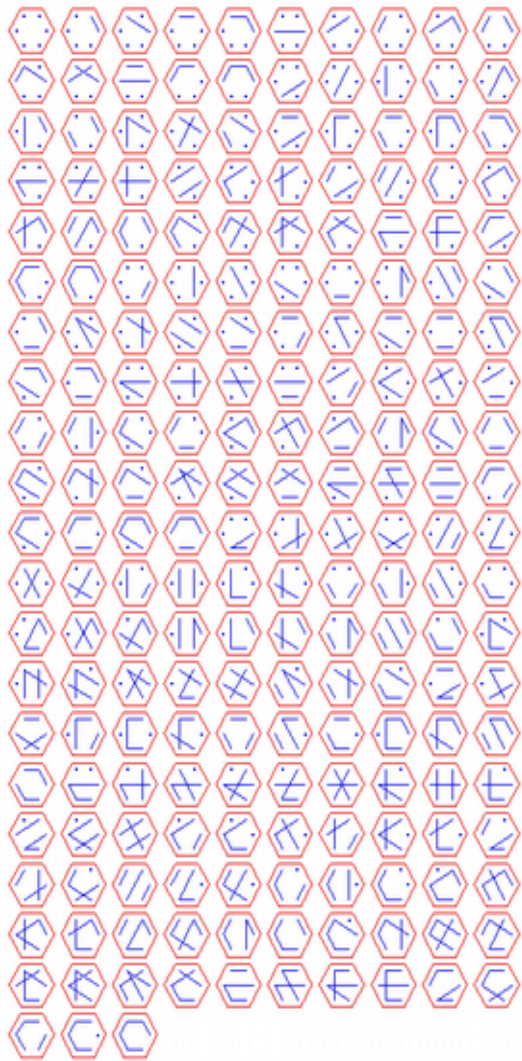


- With 4 nodes: 15 partitions



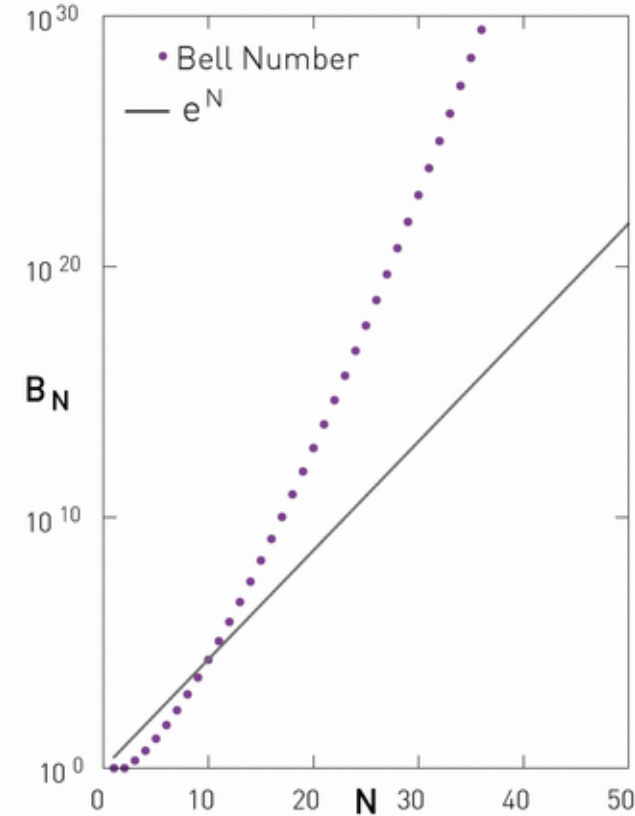
- With 5 nodes: 52 partitions





- With 6 nodes: 203 partitions
- With 7 nodes: 877 partitions
- With  $n$  nodes: Bell number

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



# Hierarchical algorithms

Recursively:

- Merge nodes that are similar (agglomerative)  
— OR —
- Split groups that are dissimilar (divisive)

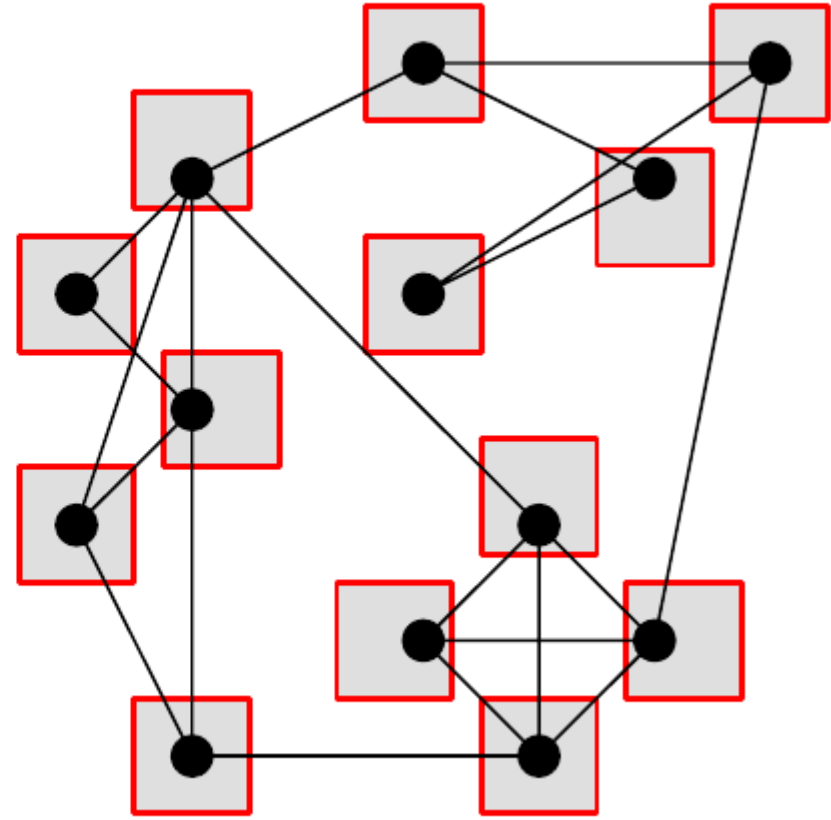


# Agglomerative clustering

- Start with every node in a separate community (these are called “singletons”)
- Repeat until all nodes are together:
  - Compute community distances between all pairs
  - Merge the two closest communities

dendrogram

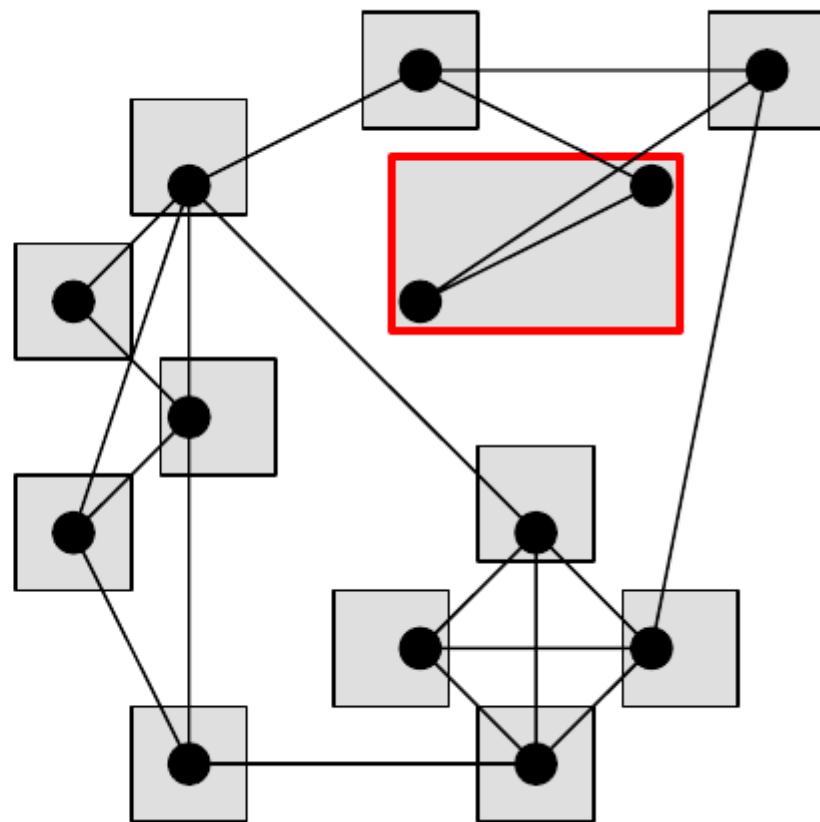
current clustering



dendrogram



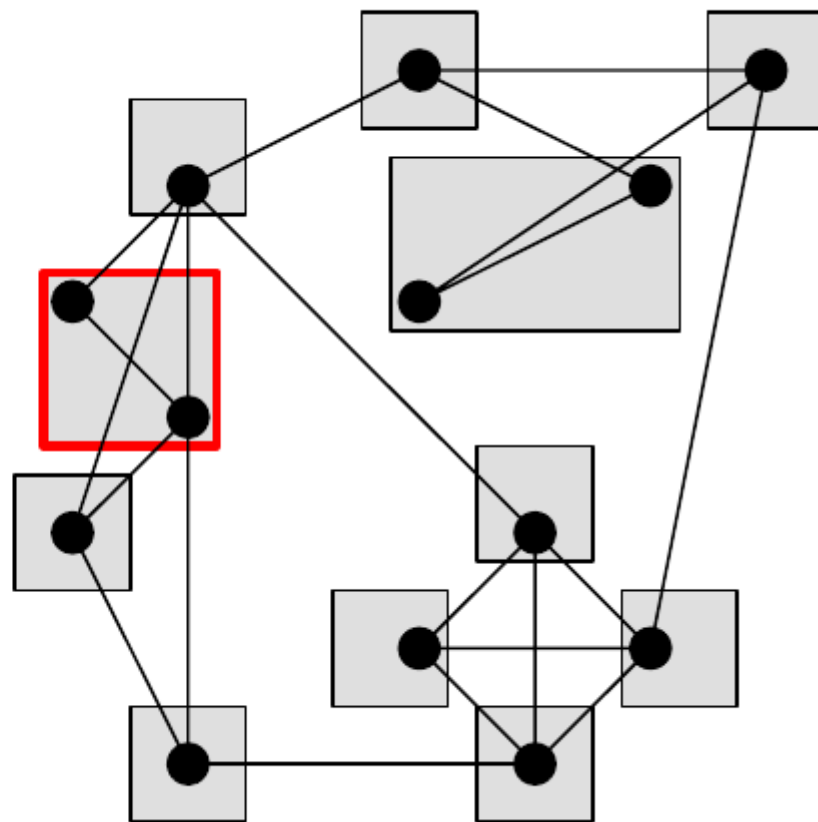
current clustering



dendrogram



current clustering

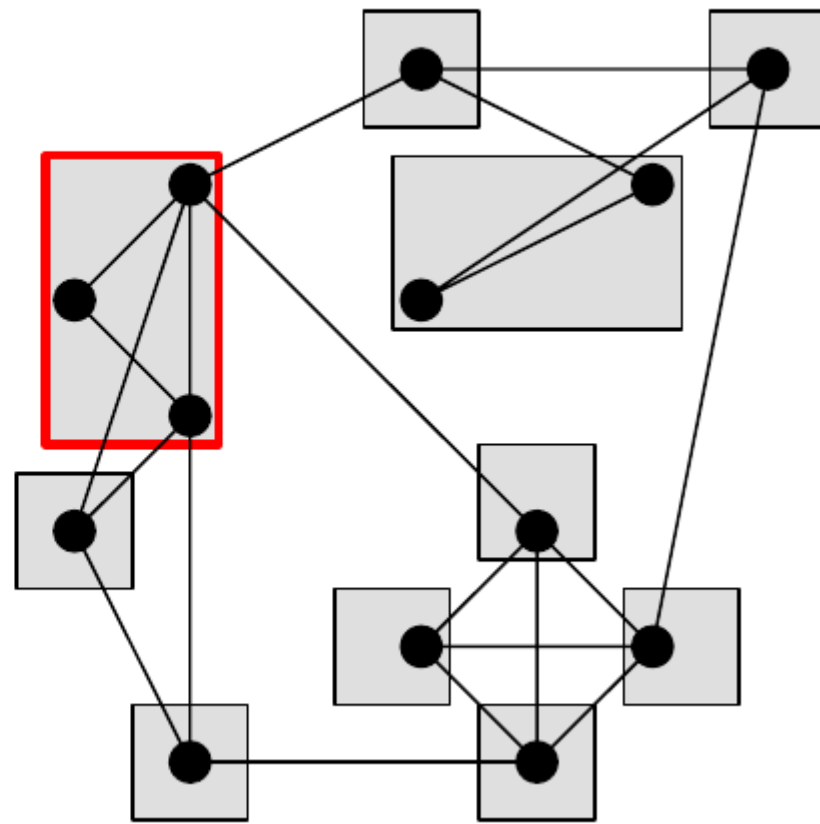




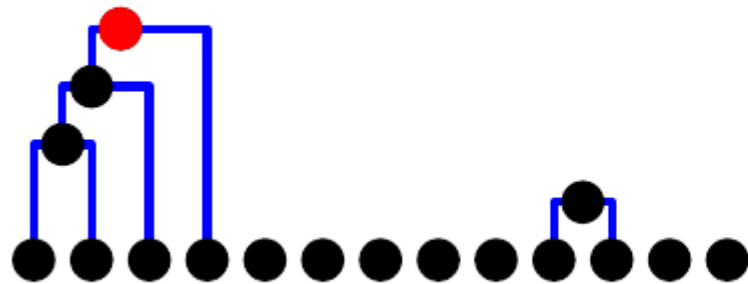
dendrogram



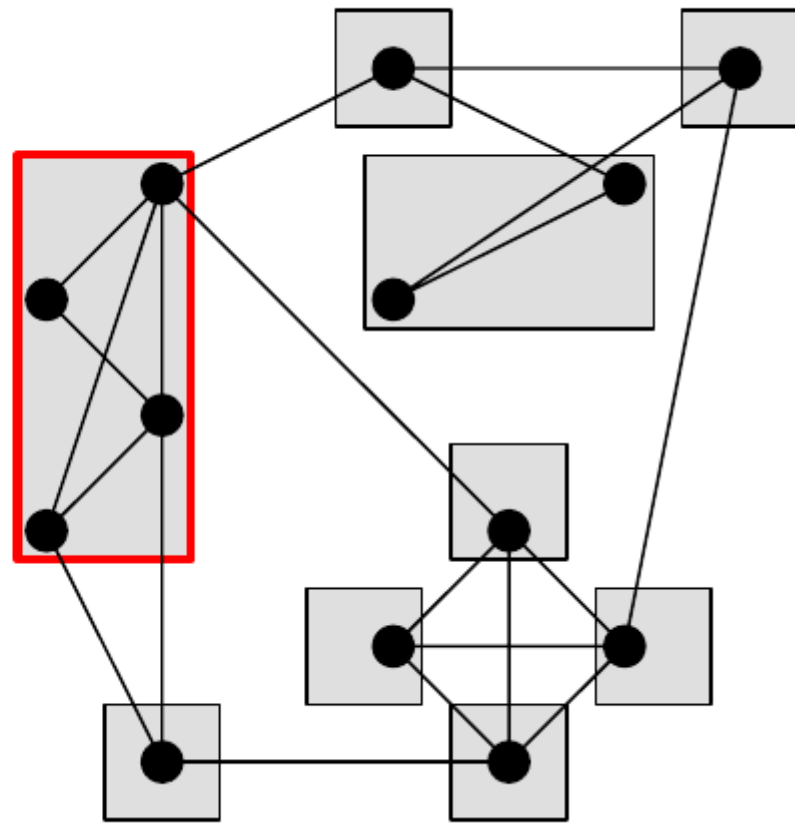
current clustering



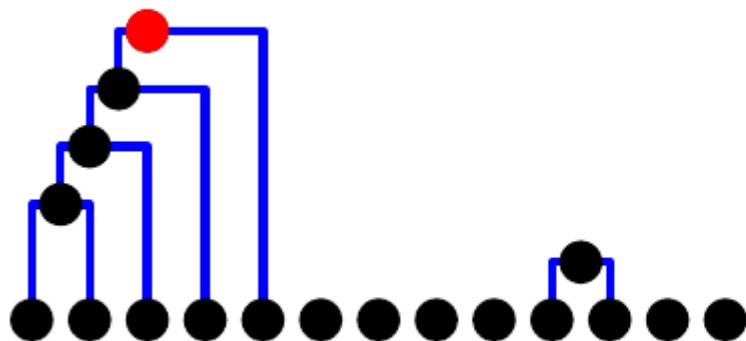
dendrogram



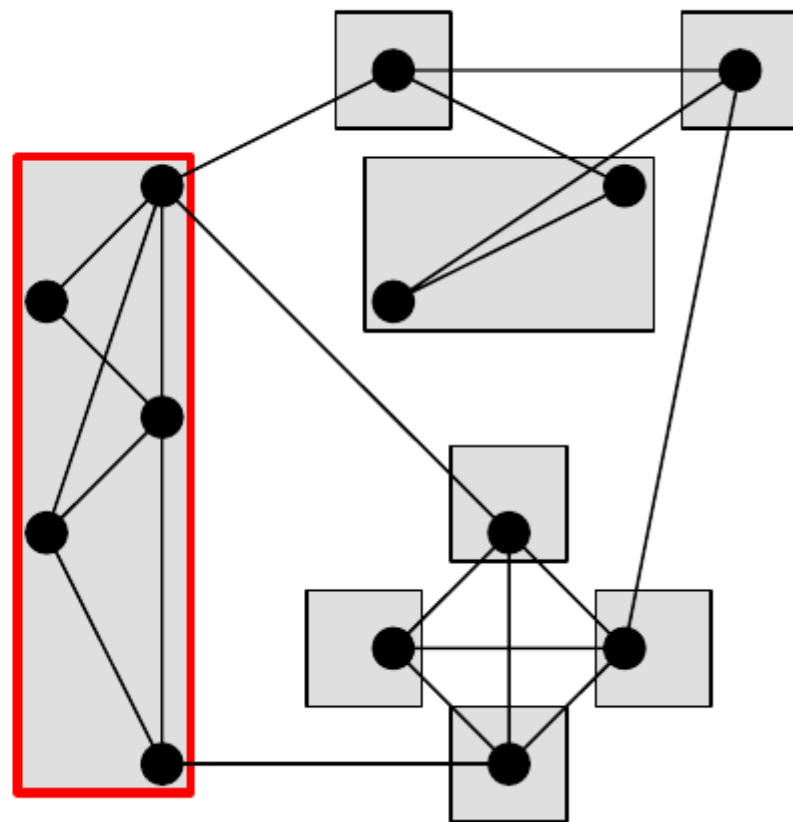
current clustering



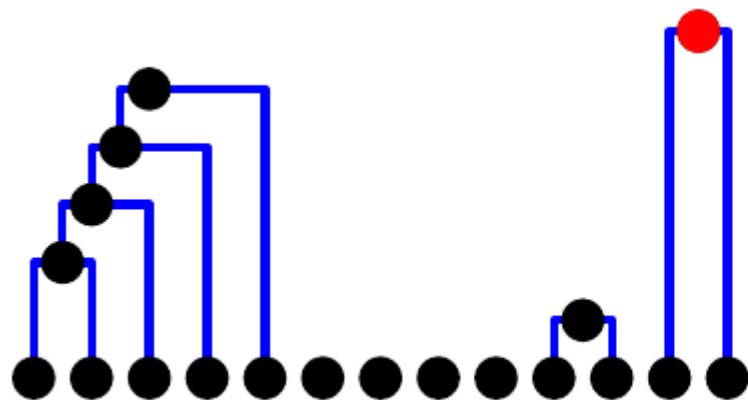
dendrogram



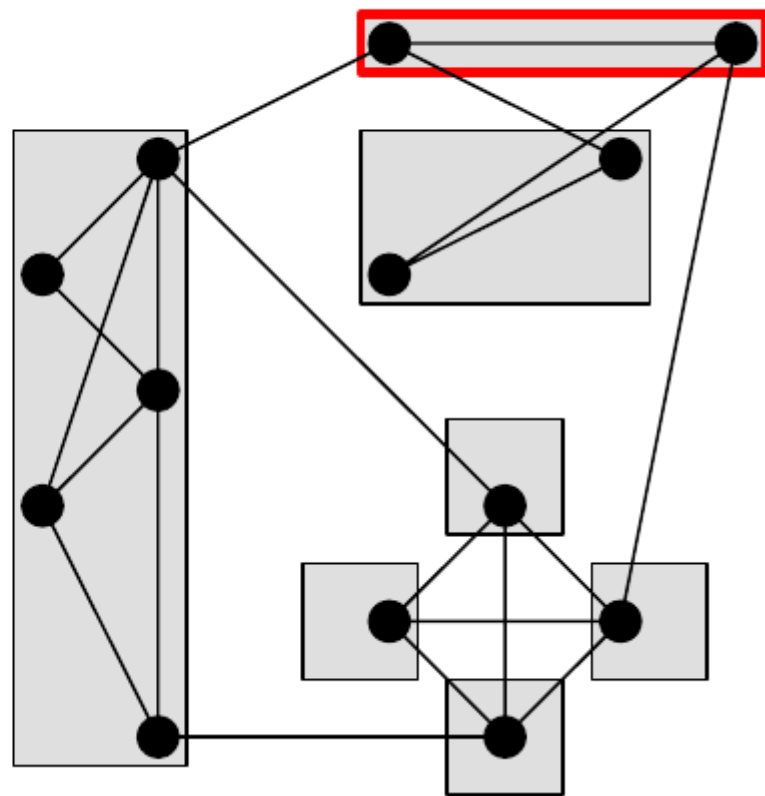
current clustering



dendrogram

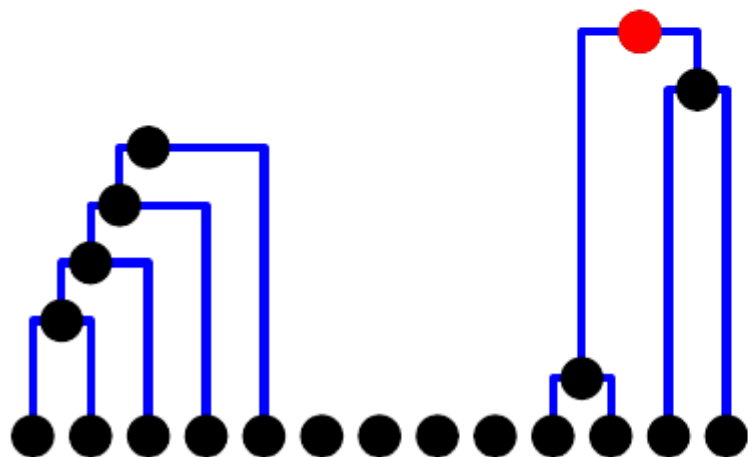


current clustering

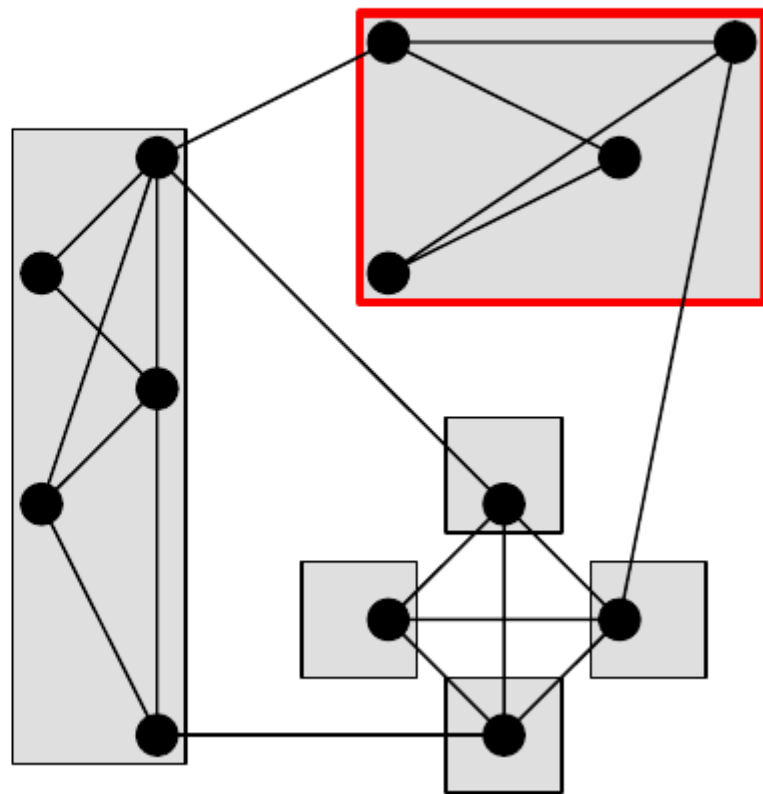




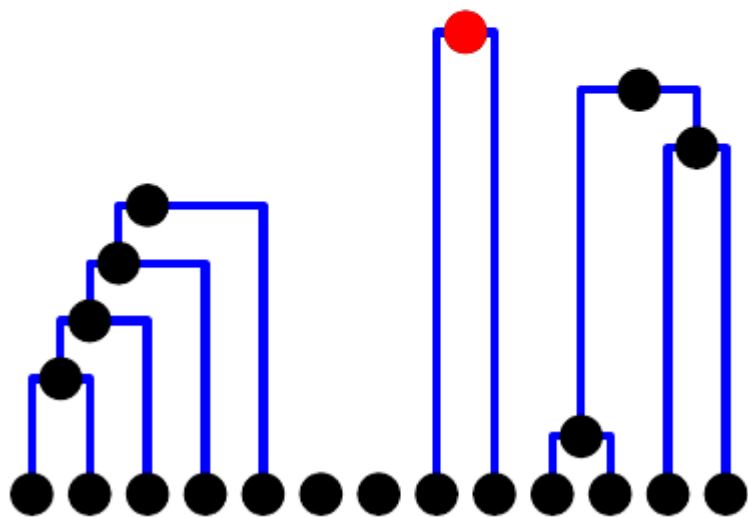
dendrogram



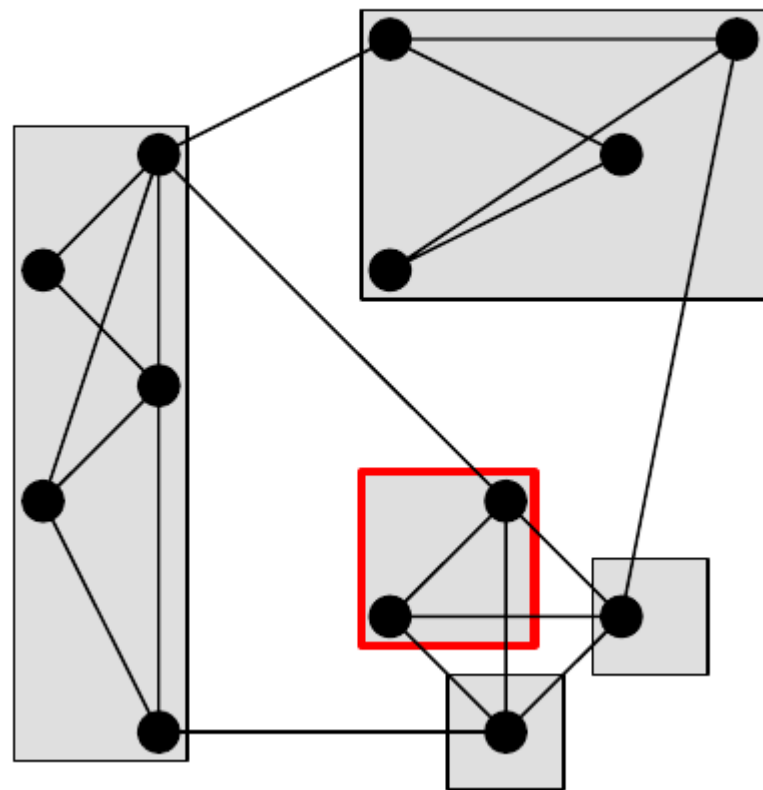
current clustering



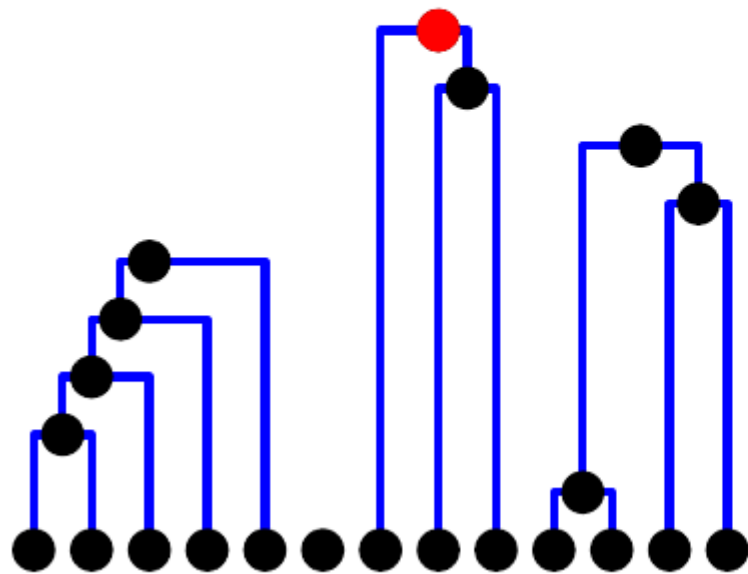
dendrogram



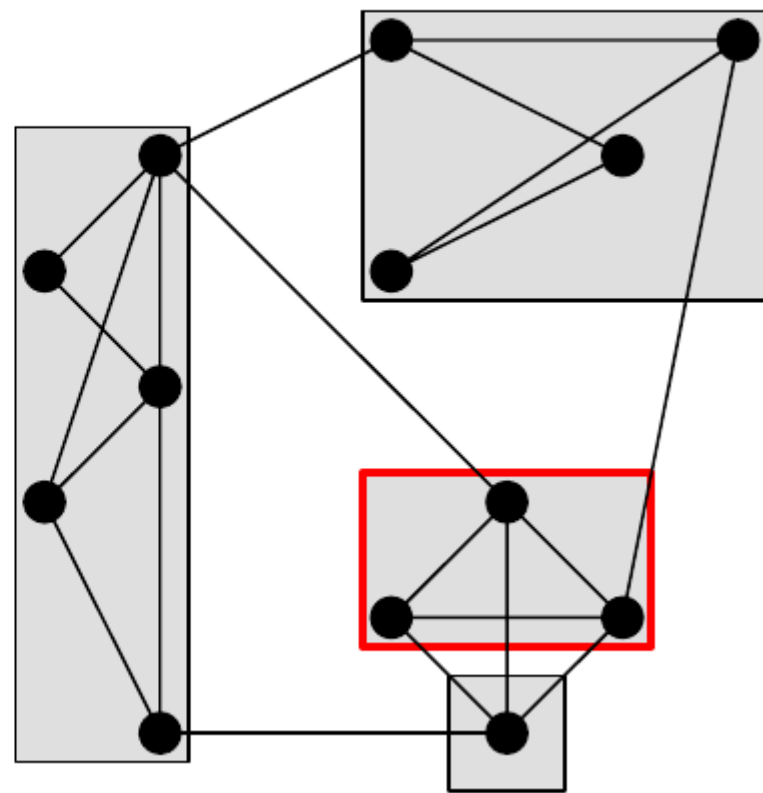
current clustering



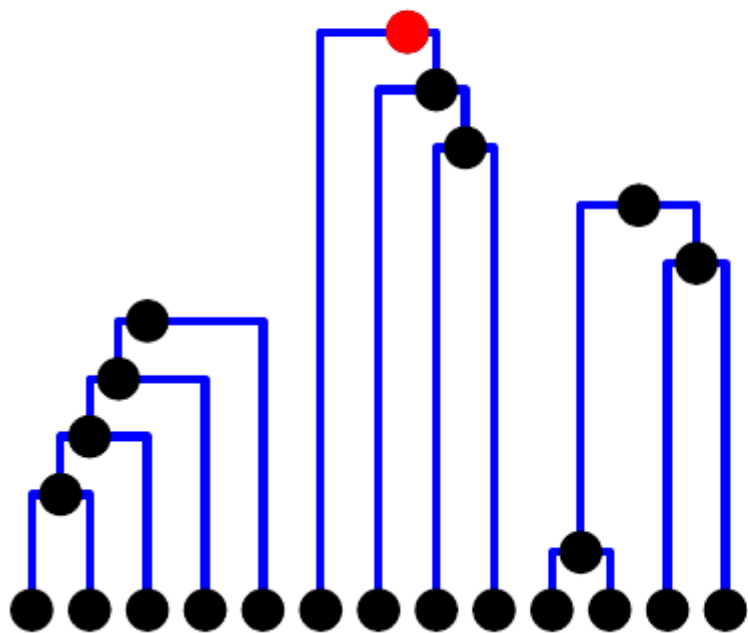
dendrogram



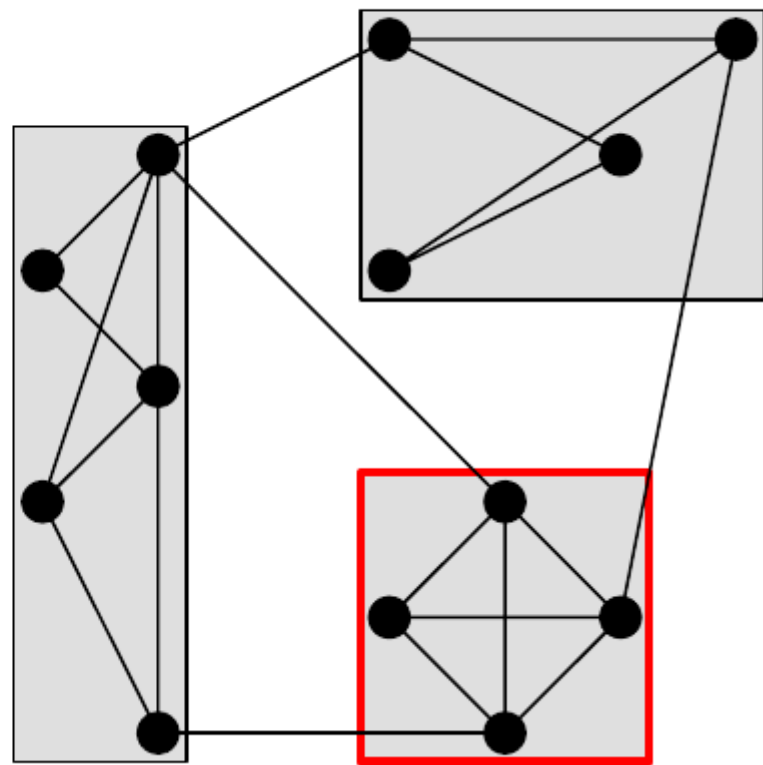
current clustering



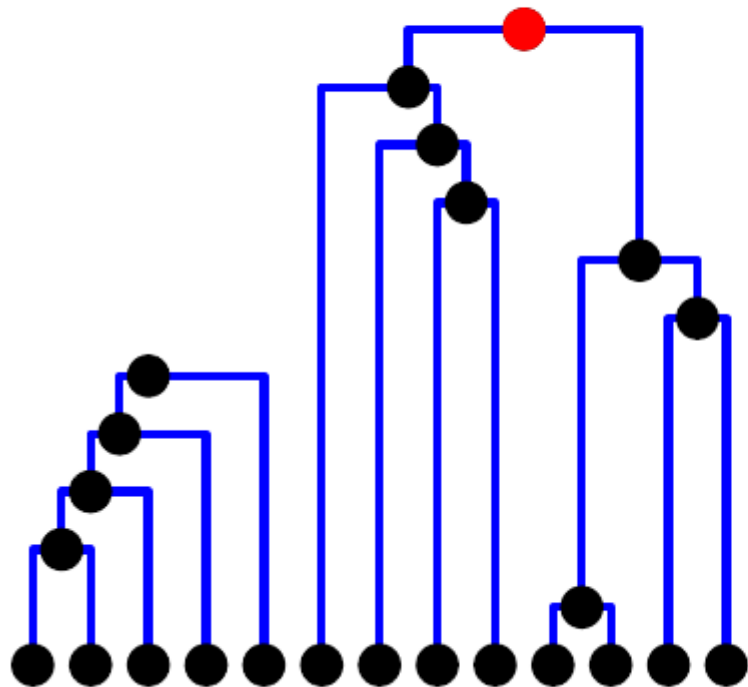
dendrogram



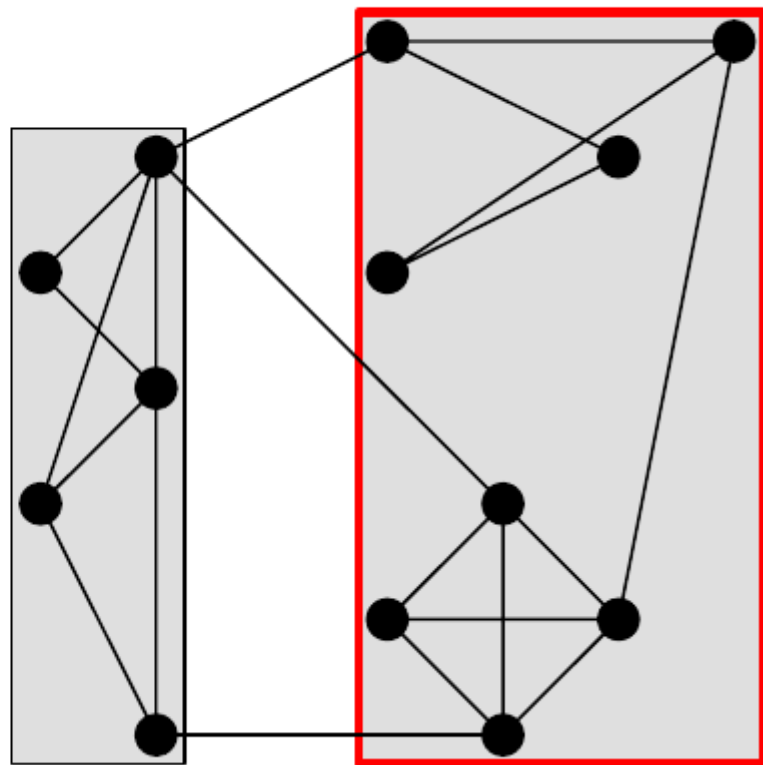
current clustering



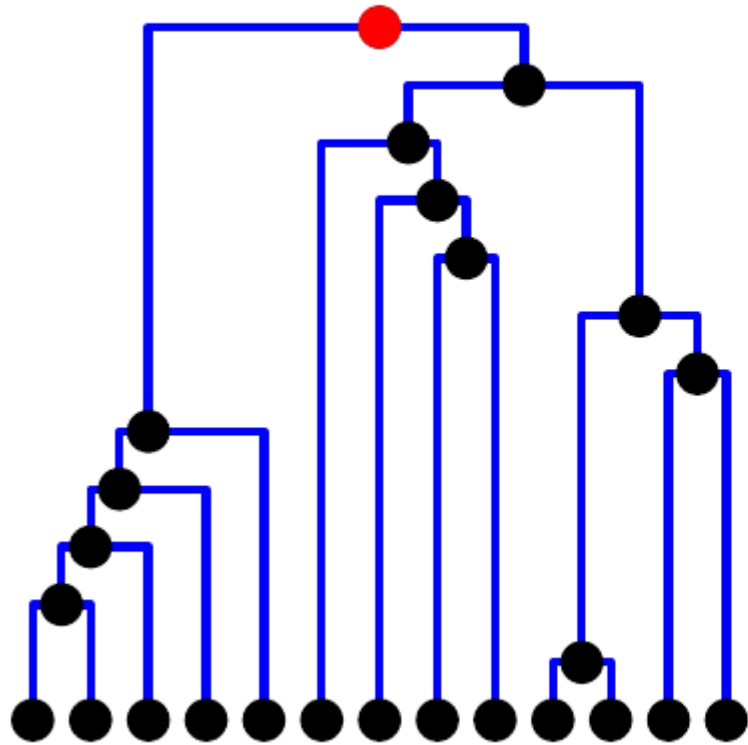
dendrogram



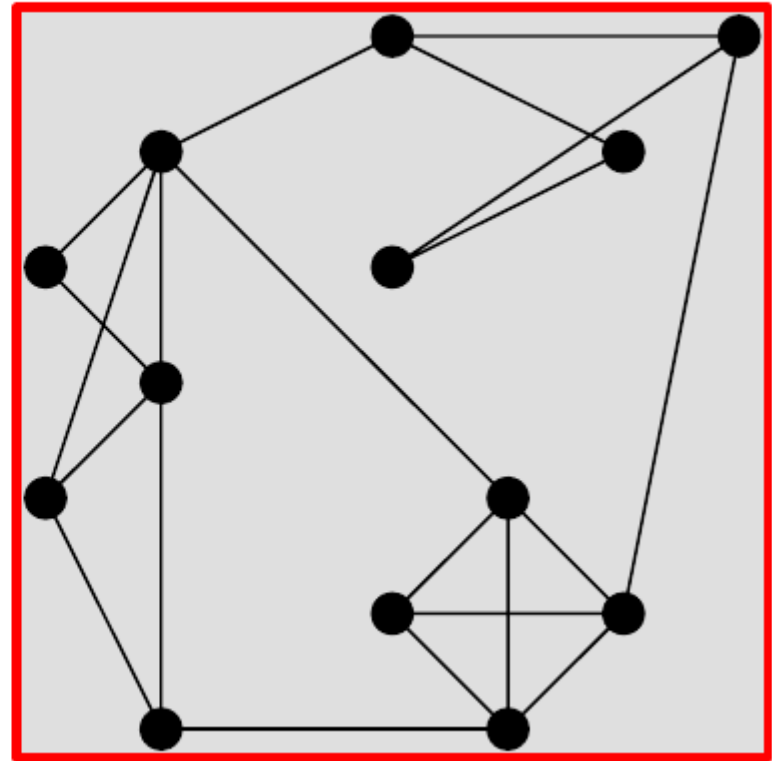
current clustering



dendrogram

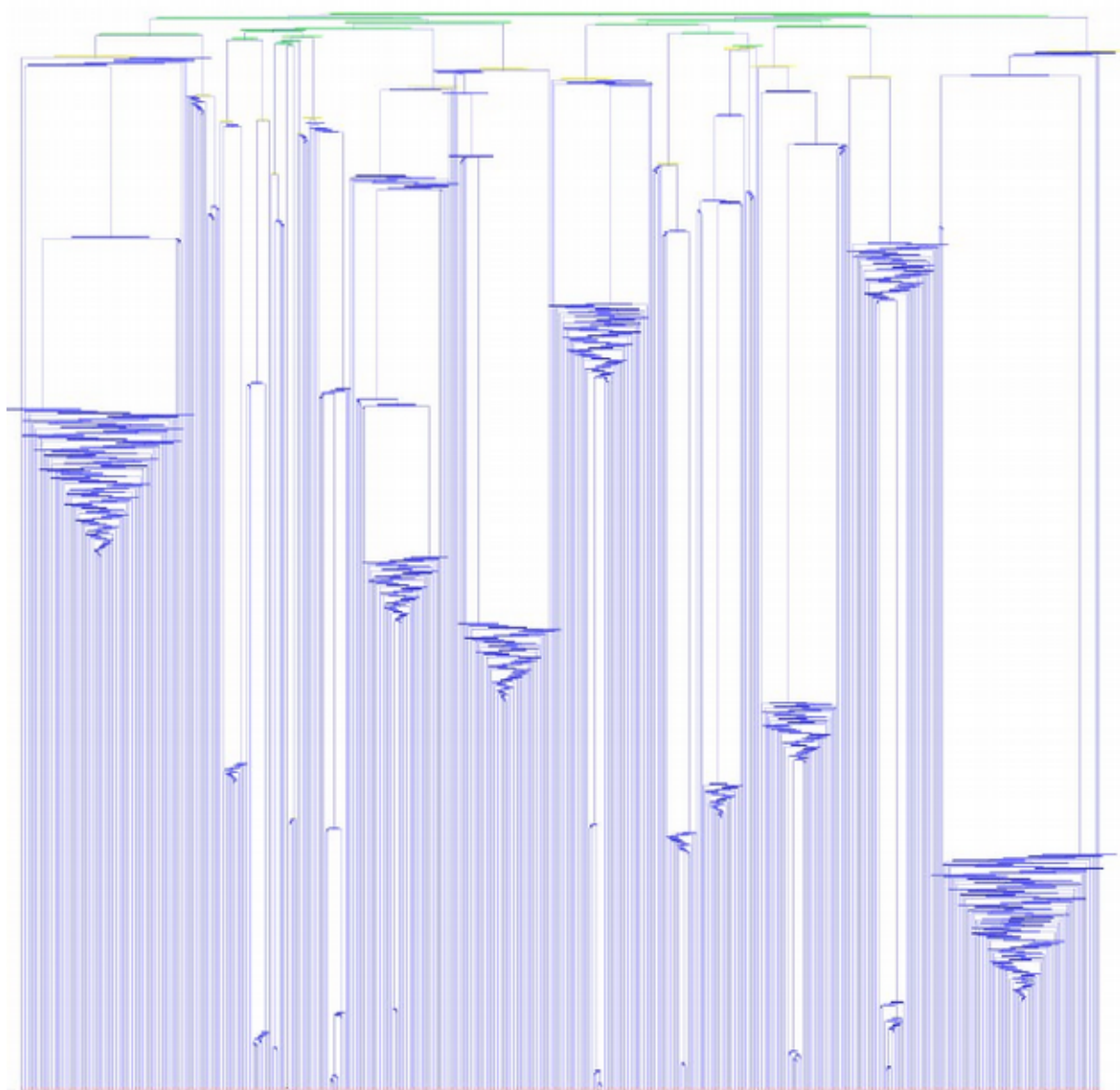


current clustering



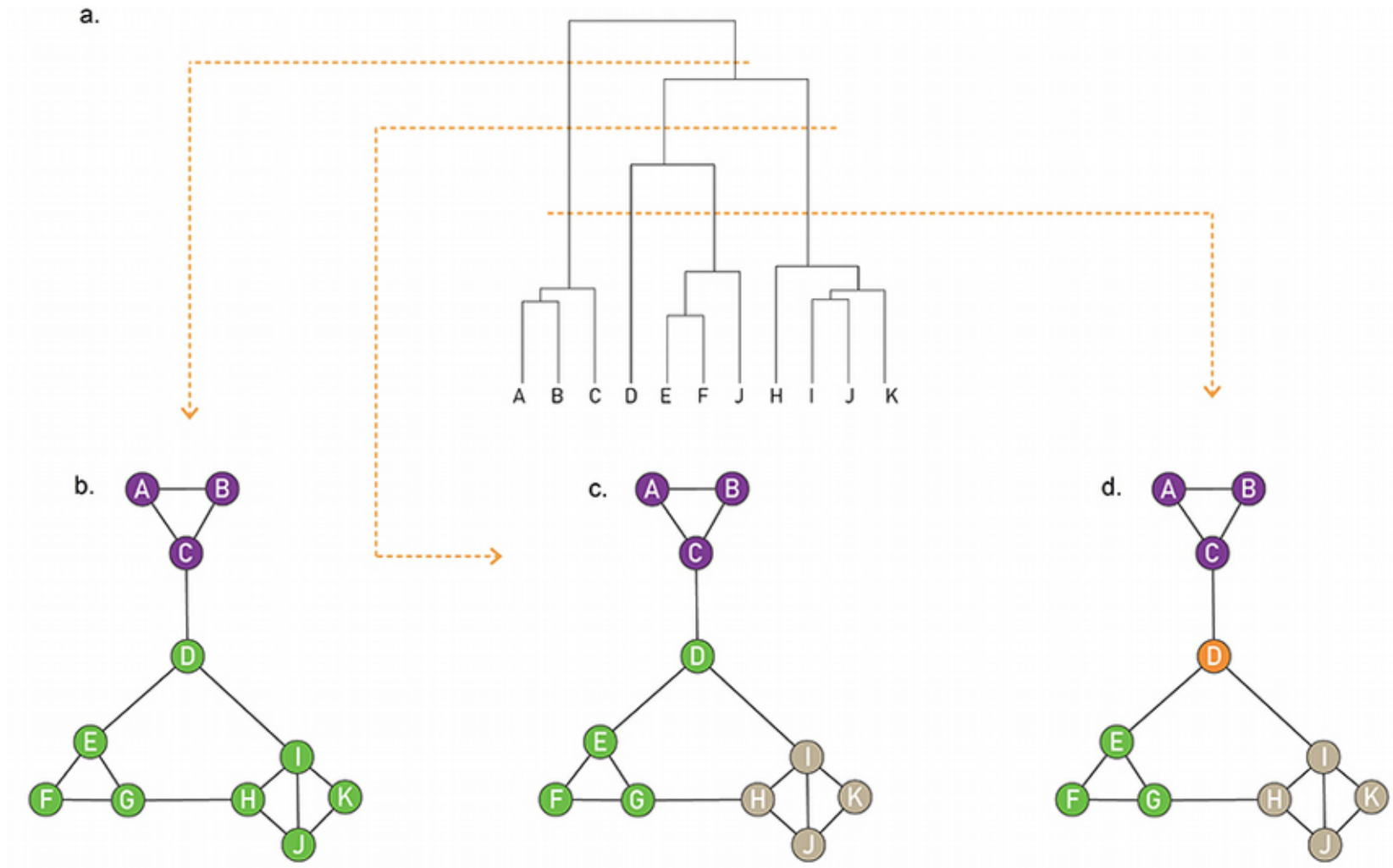
# Can handle large graphs

In this  
example,  
 $|V| \simeq 1000$





# Where to cut?

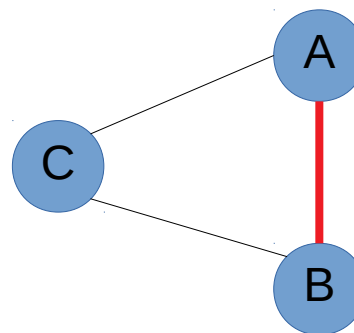


# We need to specify some details

- When are two nodes considered similar
- When are two groups of nodes considered similar

# Node similarity: topological overlap

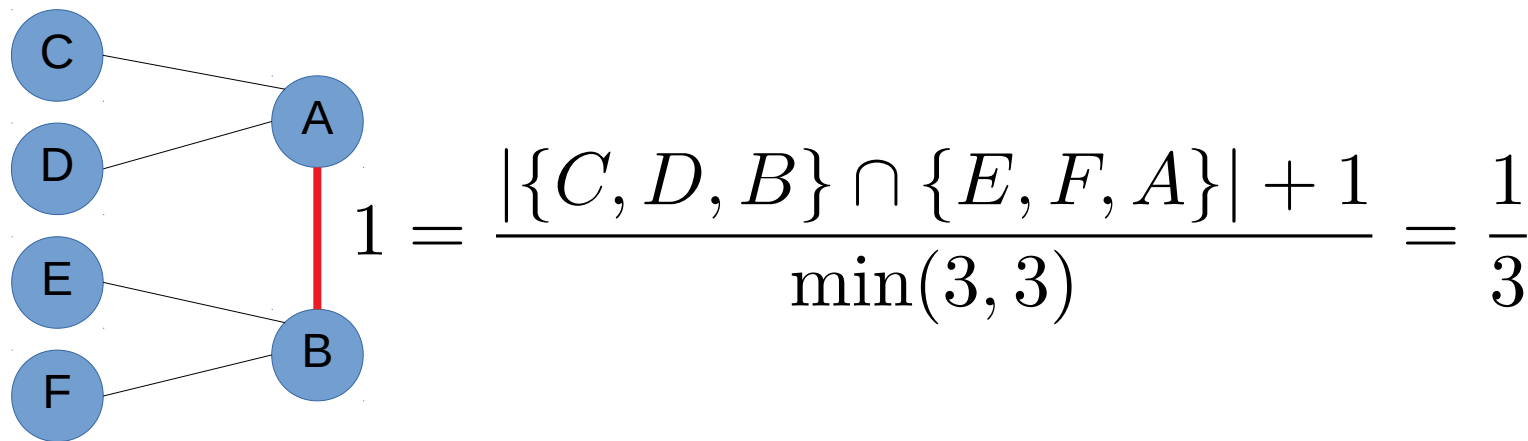
$$x_{ij}^0 = \frac{|\Gamma(i) \cap \Gamma(j)| + A_{ij}}{\min(k_i, k_j)}$$



1 =  $\frac{|\{C, B\} \cap \{C, A\}| + 1}{\min(2, 2)} = \frac{2}{2}$

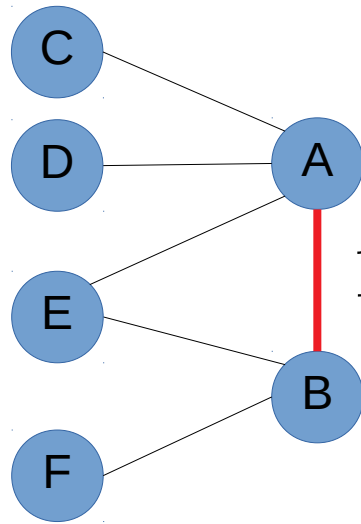
# Node similarity: topological overlap

$$x_{ij}^0 = \frac{|\Gamma(i) \cap \Gamma(j)| + A_{ij}}{\min(k_i, k_j)}$$



# Node similarity: topological overlap

$$x_{ij}^0 = \frac{|\Gamma(i) \cap \Gamma(j)| + A_{ij}}{\min(k_i, k_j)}$$



$$1 = \frac{|\{C, D, E, B\} \cap \{E, F, A\}| + 1}{\min(4, 3)} = \frac{2}{3}$$

# Group similarity of groups U and V

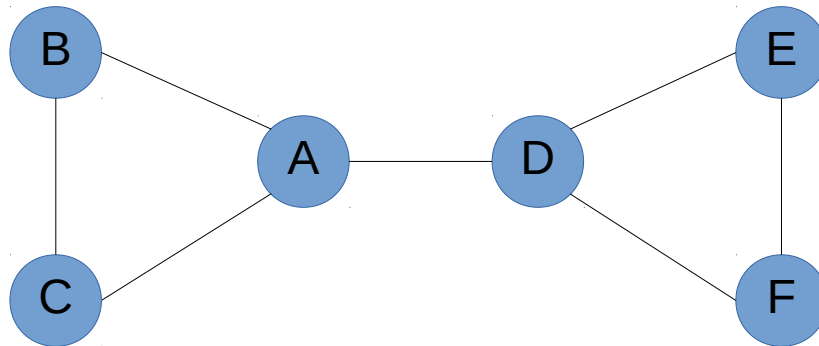
- Single-linkage:  $x_{U,V} = \min_{i \in U, j \in V} x_{ij}$
- Complete-linkage:  $x_{U,V} = \max_{i \in U, j \in V} x_{ij}$
- Average-linkage:  $x_{U,V} = \frac{1}{|U||V|} \sum_{i \in U, j \in V} x_{ij}$



# Try it!

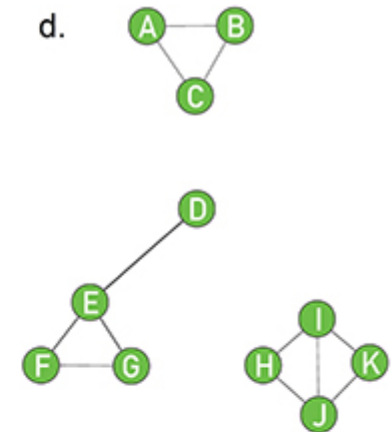
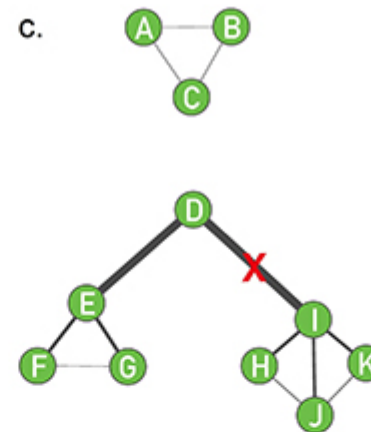
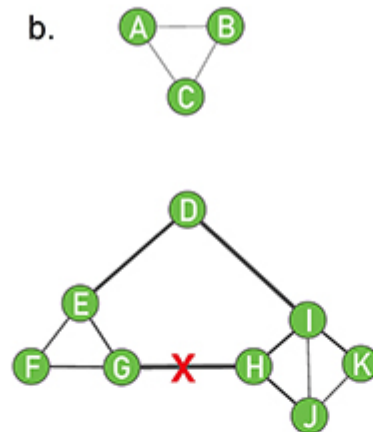
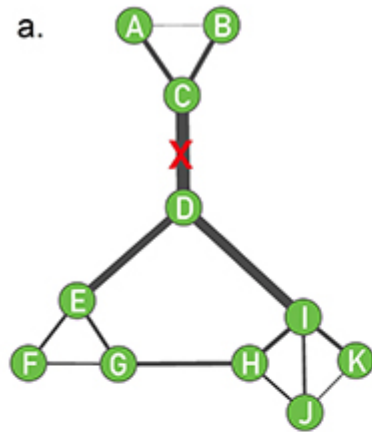
*Run Kavas's algorithm on this graph with complete (max) linkage; to save time, compute only over existing edges*

$$x_{ij}^0 = \frac{|\Gamma(i) \cap \Gamma(j)| + A_{ij}}{\min(k_i, k_j)}$$



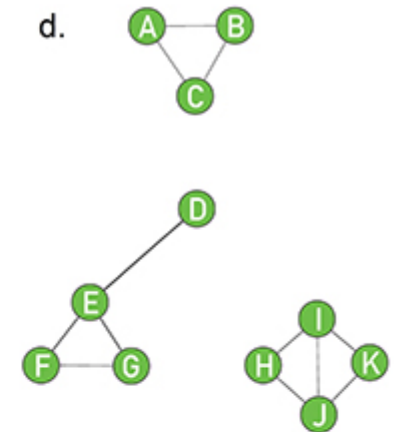
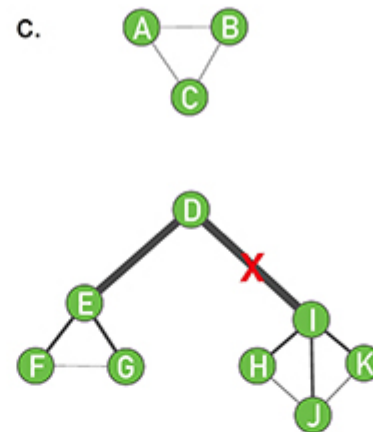
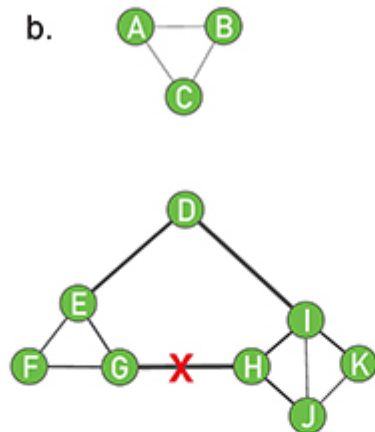
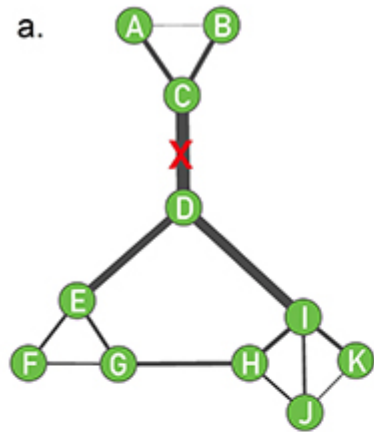
# Divisive clustering

- Start with the original graph
- Repeat until all nodes are separated:
  - Remove a link
- Which link to remove? The one with maximum edge betweenness!

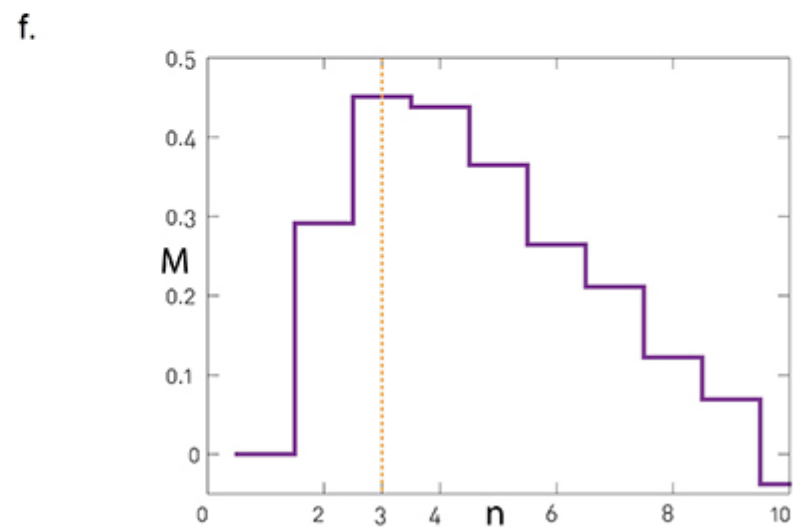
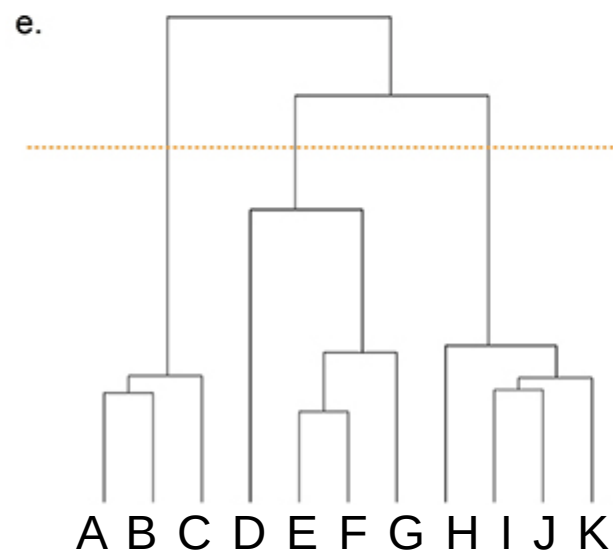
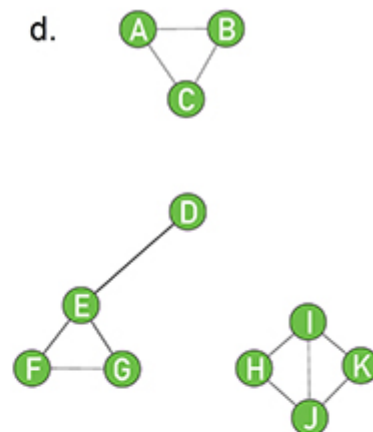
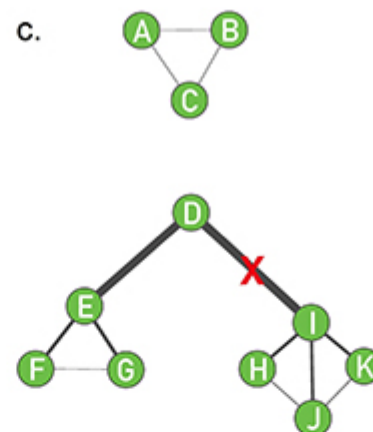
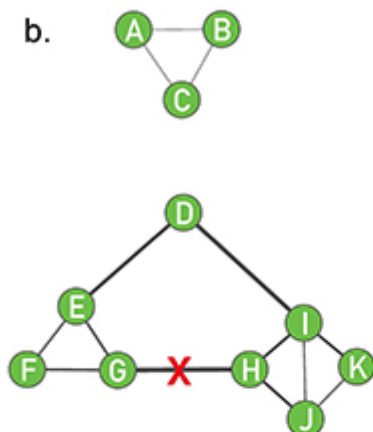
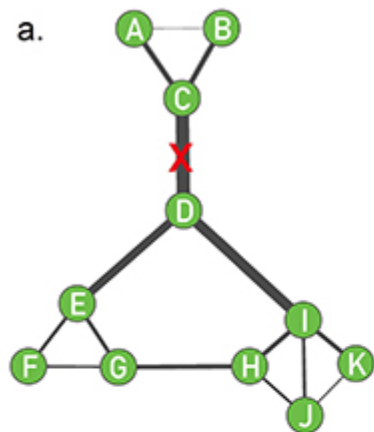


As in the case of  
agglomerative methods, we  
have a choice on where to  
cut the graph

The cut on the left has 3  
communities



What if we had a function that was high for a good clustering and low for a bad clustering?



Something like this?

# The modularity function

Let  $e_{ii}$  = probability that an edge connects a node in community  $i$  with another node in community  $i$

$$e_{ii} = \frac{|(u, v) \in E : u \in C_i \wedge v \in C_i|}{|E|}$$

Let  $a_i$  = probability that an edge has at least one end in community  $i$

$$a_i = \frac{\sum_{u \in i} k_u}{2|E|}$$



# Modularity with $n_c$ communities

$$Q = \sum_{i=1}^{n_c} (e_{ii} - a_i^2)$$

$$e_{ii} = \frac{|(u, v) \in E : u \in C_i \wedge v \in C_i|}{|E|}$$

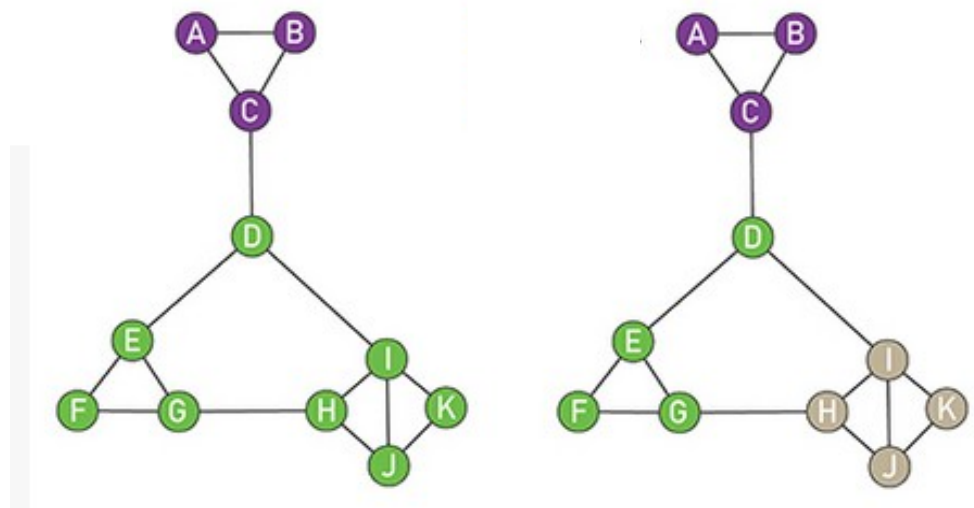
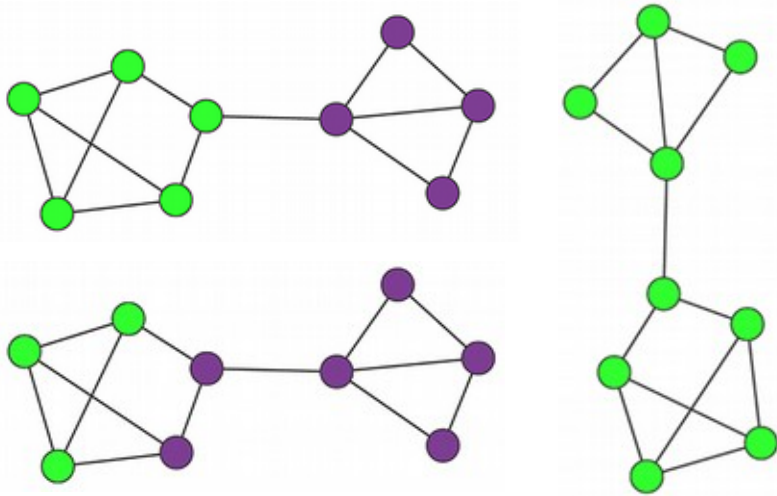
$$a_i = \frac{\sum_{u \in i} k_u}{2|E|}$$

# Try it!

*Compute the modularity of these partitions*

$$Q = \sum_{i=1}^{n_c} (e_{ii} - a_i^2)$$

$$e_{ii} = \frac{|(u, v) \in E : u \in C_i \wedge v \in C_i|}{|E|}$$
$$a_i = \frac{\sum_{u \in i} k_u}{2|E|}$$



# Greedy modularity optimization

- Start with each node in one community
- Repeat until all nodes are together:
  - For every pair of communities connected by a link:
    - Compute  $Q$  of the networks if those communities were merged
  - Merge the communities that give the larger increase (or the smaller decrease) in  $Q$
- Return the intermediate step at which  $Q$  was maximized

# Community sizes distribution

