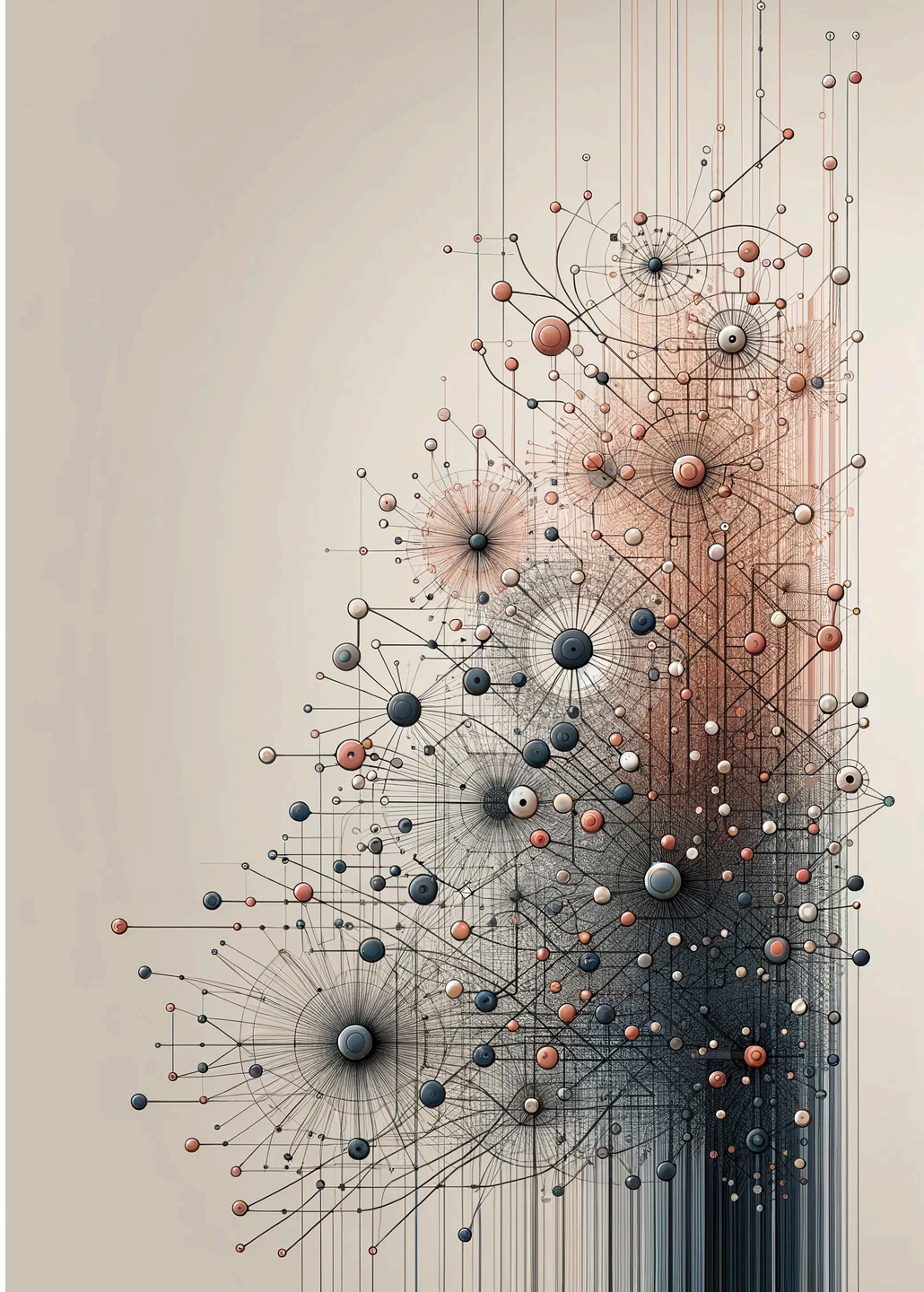
UNIVERSITÀ DI TORINO

Analisi e Visualizzazione delle Reti Complesse

NS12 - Communities

Prof. Rossano Schifanella



UNIVERSITÀ DI TORINO

Outline

- Basic definitions
  - **community variables**
  - **community definitions**
  - **partitions**
- Related problems
  - **network partitioning**
  - **data clustering and dendrograms**

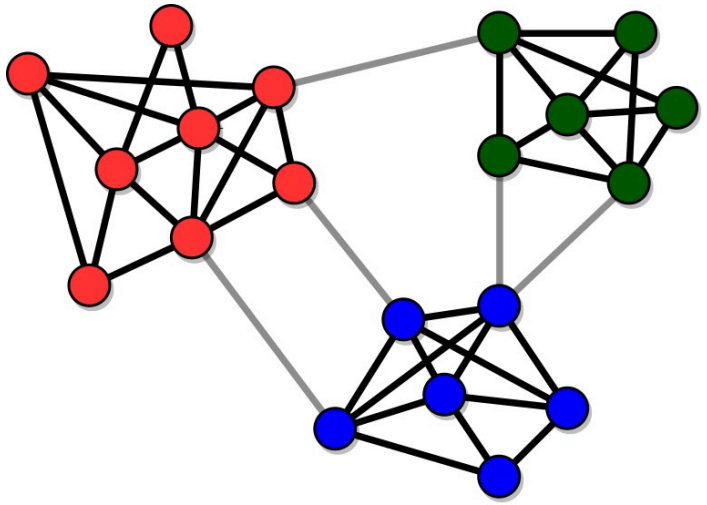
UNIVERSITÀ DI TORINO

Basic definitions

UNIVERSITÀ DI TORINO

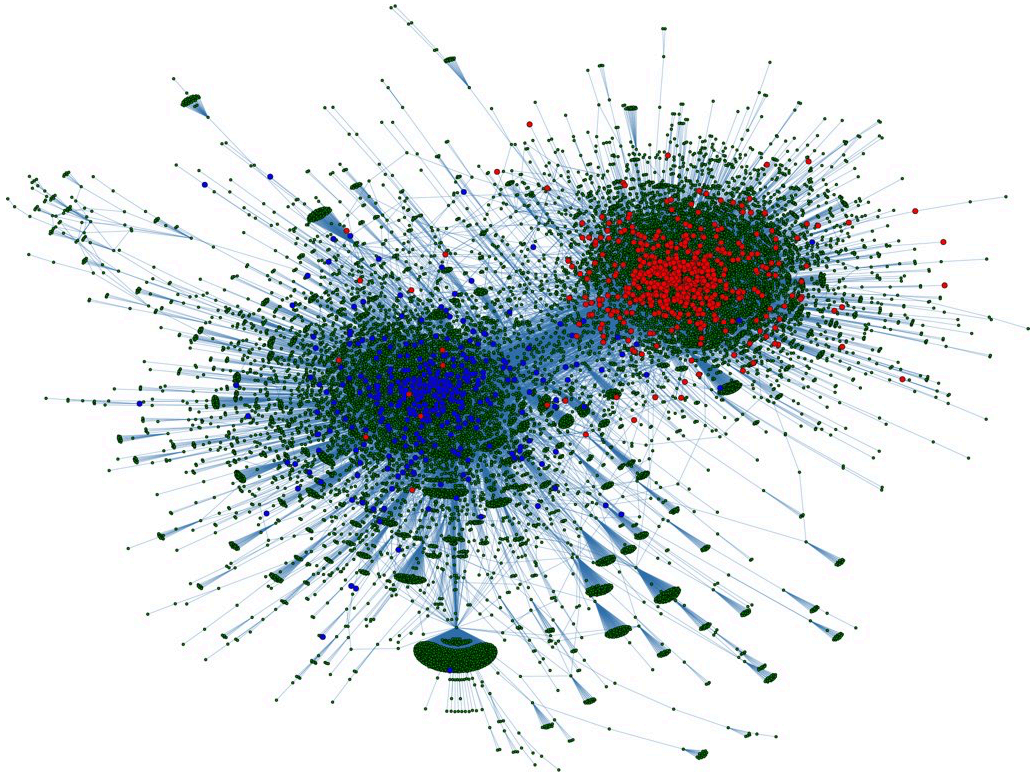
Community structure

Communities (or clusters): sets of tightly connected nodes



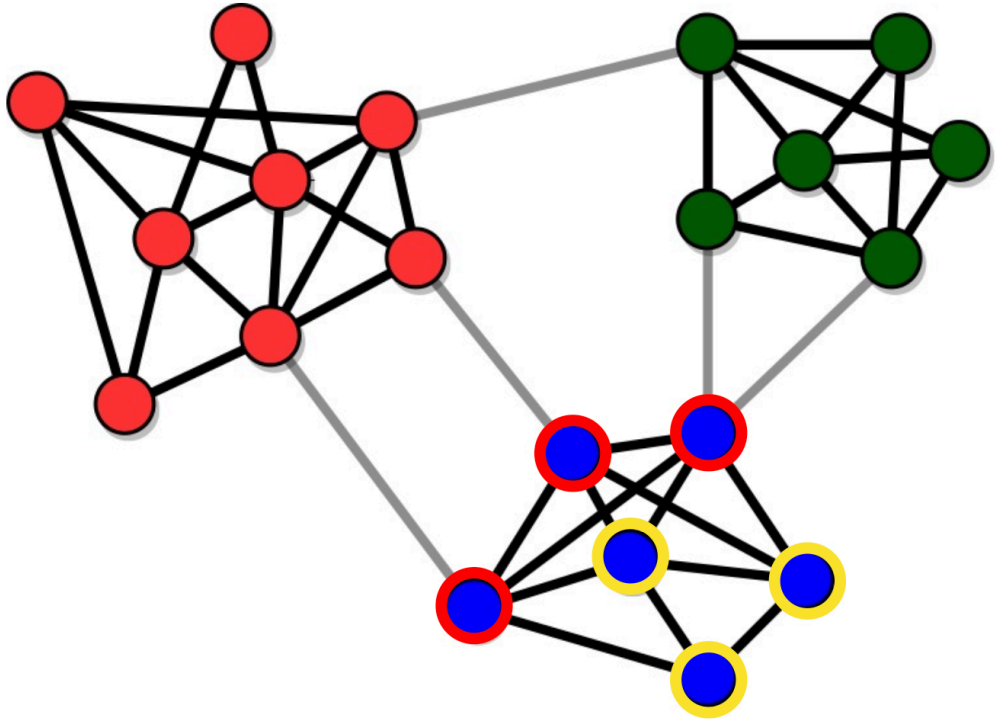
Community structure

- Example: Twitter users with strong political preferences tend to follow those aligned with them and not to follow users with different political orientation
- Other examples: social circles in social networks, functional modules in protein interaction networks, groups of pages about the same topic on the Web, etc.



Why study communities?

- Uncover the organization of the network
- Identify features of the nodes
- Classify the nodes based on their position in the clusters
- Find missing links

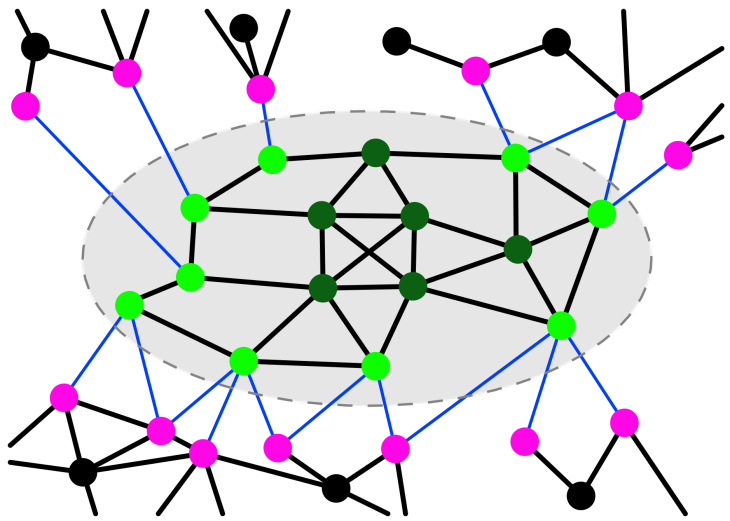


Basic definitions: variables

- **Internal degree of a node:** number of neighbors of the node in its community
- **External degree of a node:** number of neighbors of the node outside of its community
- **Community degree:** sum of the degrees of the nodes in the community
- **Internal link density:** ratio between the number of links L_C inside a community C and the maximal possible number of links that can lie inside C:

$$\delta_C^{int} = \frac{L_C}{L_c^{max}} = \frac{L_C}{\binom{N_C}{2}} = \frac{2L_C}{N_C(N_C-1)}$$

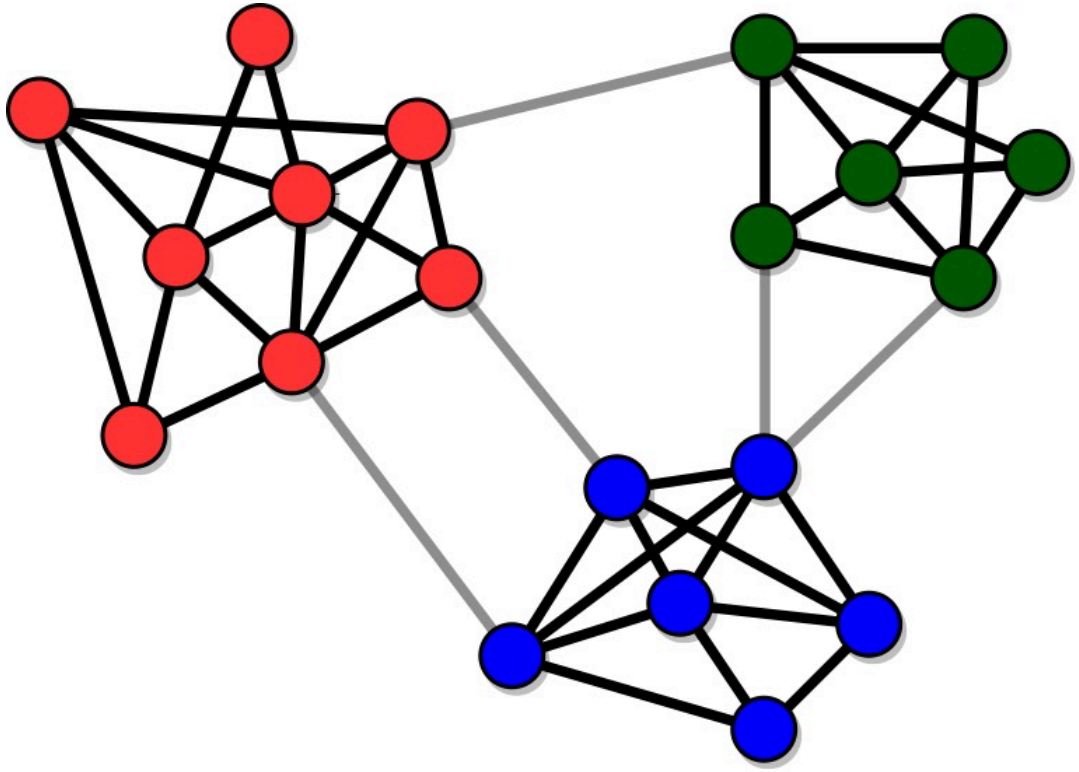
- where N_C is the number of nodes in C



Basic definitions: community

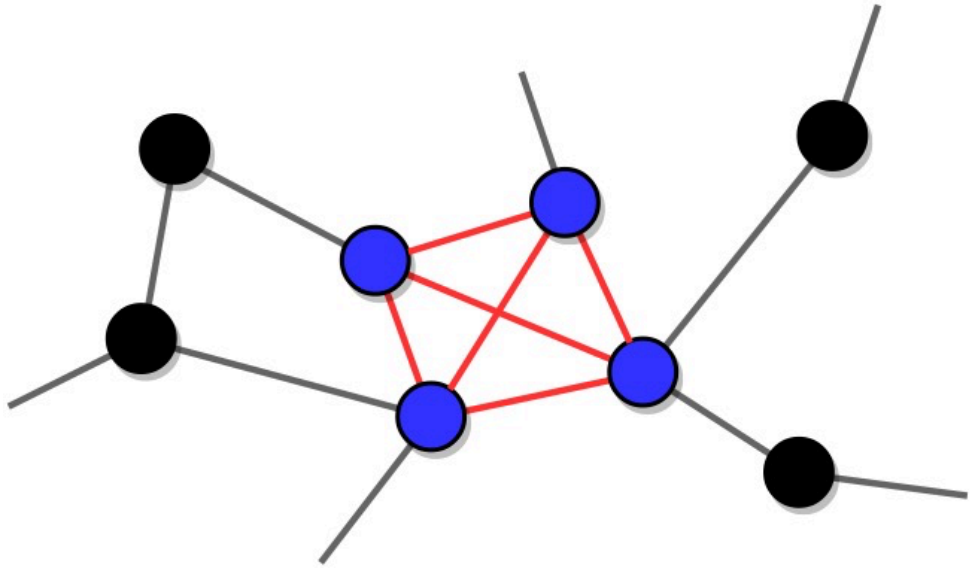
Two main features:

- **High cohesion:** communities have many internal links, so their nodes stick together
- **High separation:** communities are connected by a few links



Community definitions based on cohesion

- **Principle:** focus on the cluster's properties, disregarding the rest of the network
- **Example:** clique — all internal links are there (maximal cohesion)
- **Problem:** nodes are connected to all others in the cluster, whereas in real communities, they have different roles, which is reflected in heterogenous linking patterns



Community definitions based on cohesion versus separation

Principle:

Definition tends to achieve high cohesion and high separation

Popular idea:

The number of internal links exceeds the number of external links

Two concepts:

- **Strong community:** subnetwork such that the internal degree of each node is greater than its external degree
- **Weak community:** subnetwork such that the sum of the internal degrees of its nodes is greater than the sum of their external degrees

Community definitions based on cohesion versus separation

A strong community is also a weak community:

If the inequality between internal and external degree holds for each node, then it must hold for the sum over all nodes

A weak community is not a strong community, in general:

If the inequality between internal and external degree holds for the sum, it may be violated for one or more nodes

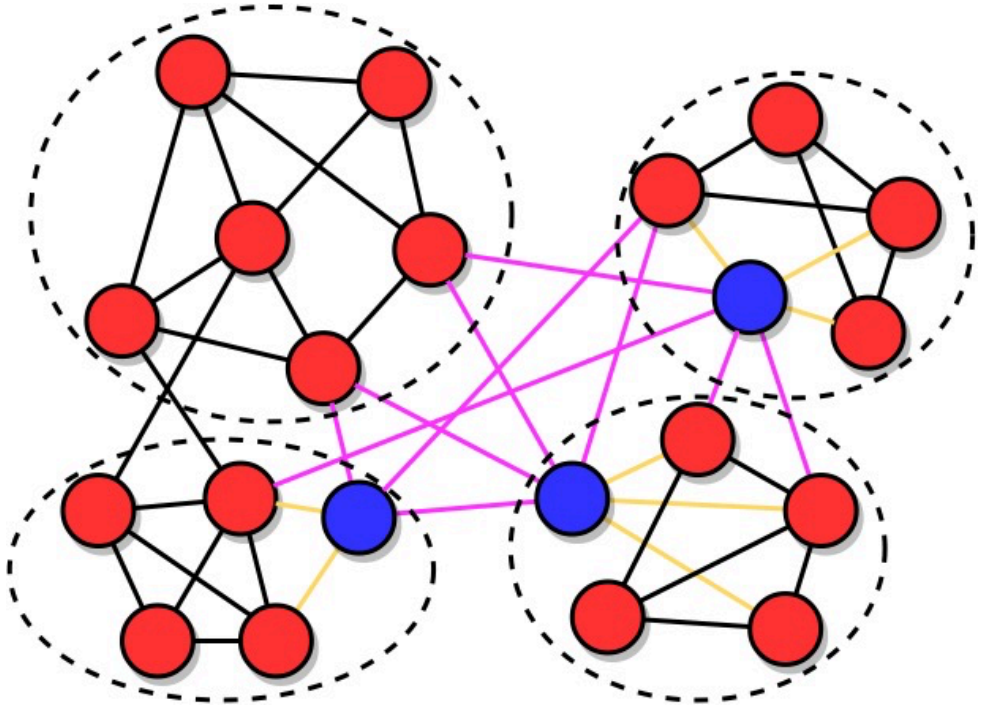
Problem:

In the definition of strong and weak community, one compares a subnetwork with the rest of the network. It makes more sense to **compare subnetworks to each other!**

Community definitions based on cohesion versus separation

Less stringent definitions of strong and weak community:

- **Strong community:** subnetwork such that each node has more neighbors inside it than in any other community
- **Weak community:** subnetwork such that the sum of the internal degrees of its nodes exceeds the total number of neighbors that the nodes have in any other community



Partitions

- A **partition** is a division of nodes into non-empty, non-overlapping subsets (communities)
- Each node belongs to exactly one community
- The Bell number B_n counts the number of possible ways to partition a set of n elements

n	B_n
1	1
2	2
3	5
4	15
5	52
6	203
7	877
8	4140
9	21147
10	115975
11	678570
12	4213597
13	27644437
14	190899322
15	1382958545

Partitions and Bell numbers

- The Bell number B_n grows **faster than exponentially with** n:
  - $B_1 = 1$
  - $B_2 = 2$
  - $B_5 = 52$
  - $B_{10} = 115,975$
  - $B_{15} \approx 1.4 \times 10^{10}$
  - $B_{20} \approx 5.2 \times 10^{13}$
- For networks with just 50 nodes, the number of possible partitions exceeds 10^{47}

Implications of Bell number for community detection

Why the Bell number?

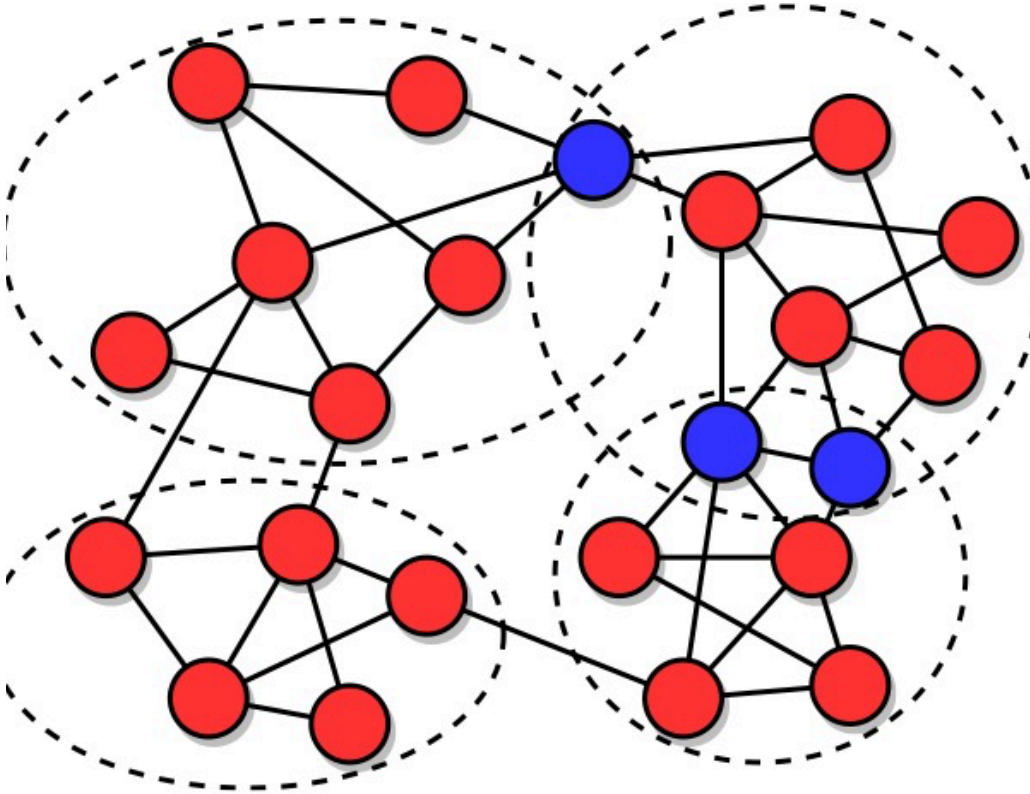
- When dividing nodes into communities, we're creating partitions of a set
- Each possible community assignment corresponds to a unique partition

Conclusion:

- Exhaustive search through all possible partitions is computationally impossible
- We need efficient algorithms that can find good community structures without exploring the entire partition space

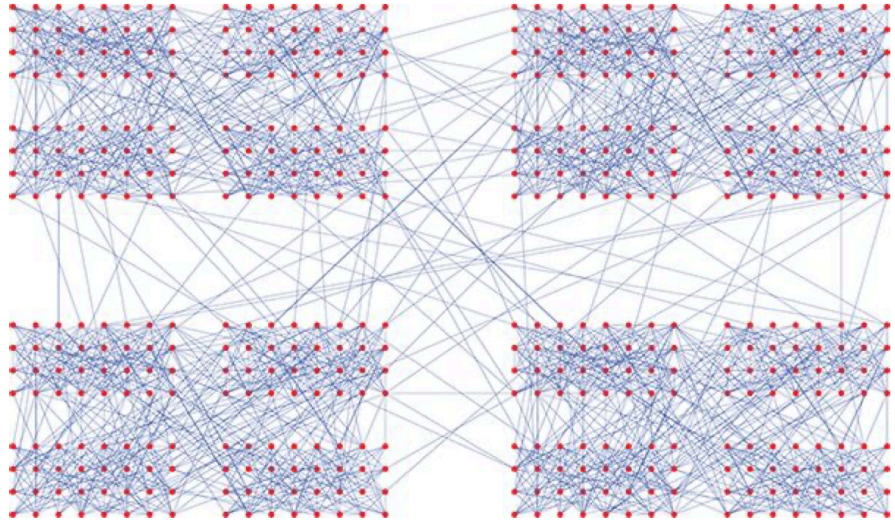
Overlapping communities

- Communities in many real networks **overlap**
- A division of a network into overlapping communities is called **cover**
- The number of possible covers of a network is **far higher** than the number of partitions due to the many ways clusters can overlap



Hierarchical communities

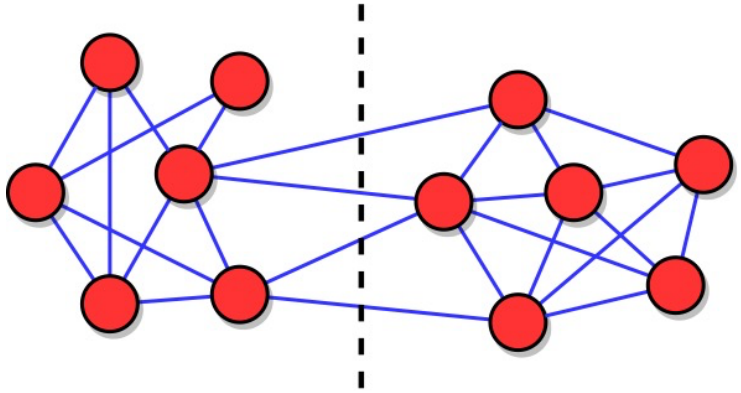
- If the network has multiple levels of organization, its communities could form a **hierarchy**, with small communities within larger ones
- **Example:** branches in a company, in turn divided into departments
- **All hierarchical partitions are meaningful:** a good clustering algorithm should detect all of them



Related problems

Network Partitioning

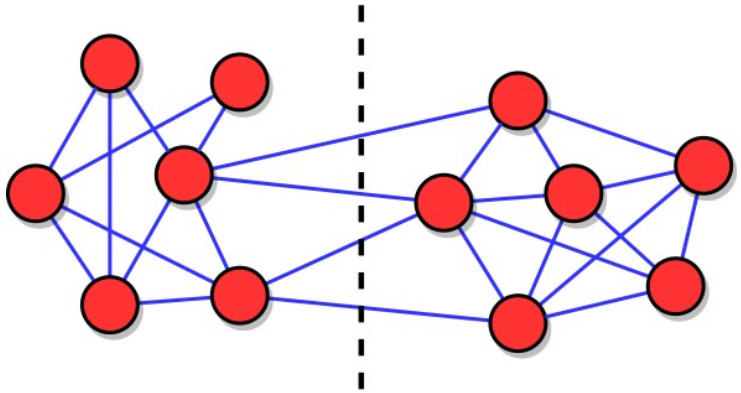
- **The problem:** dividing the nodes of a network into a number of groups of predefined size, such that the number of links between the groups is minimal
- The number of links between groups is called **cut size**
- **Key characteristics:**
  - Focused on **minimizing inter-group connections** rather than maximizing intra-group density
  - Often used in **engineering applications** where balanced partitions are required
  - The number and size of partitions are typically **specified in advance**



A particular case: Network Bisection

The problem:

Dividing the nodes of a network into **two groups of equal size**, such that the number of **links between the groups is minimal**



Example: Electronic circuit design

- **Problem:** Place components on circuit boards or chips to minimize wire length
- **Goal:** Divide circuit components into two equal groups across two boards/regions
- **Bisection benefits:**
  - Minimizes signal delay between components
  - Reduces manufacturing costs
  - Simplifies layout and routing problems
- **Challenge:** Finding optimal partition with minimal connections between groups

Example: Parallel computing task allocation

- **Problem:** Distribute computational tasks across processors while minimizing communication
- **Bisection approach:**
  i. Model tasks as nodes and dependencies as edges
  ii. Partition tasks into two equal groups (processors/cores)
  iii. Minimize communication overhead between processors
- **Real impact:**
  - Up to 30-40% reduction in communication overhead
  - Significant speedup for parallel algorithms
  - Critical for high-performance scientific computing applications

Kernighan-Lin Algorithm

Objective: Minimize the **cut size** between two partitions

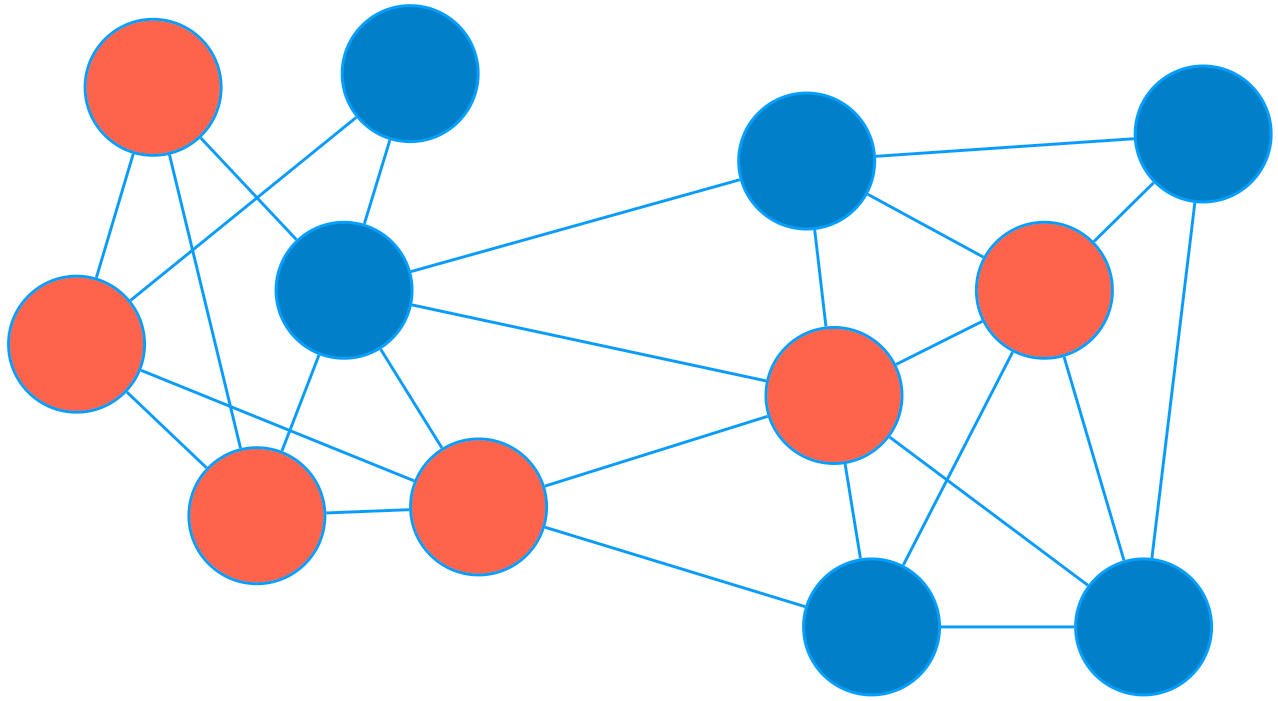
- **Cut size:** the number of edges connecting nodes in different partitions
- **Input:** A graph and an initial bisection into two sets A and B of equal size
- **Output:** An improved bisection with smaller cut size

Procedure:

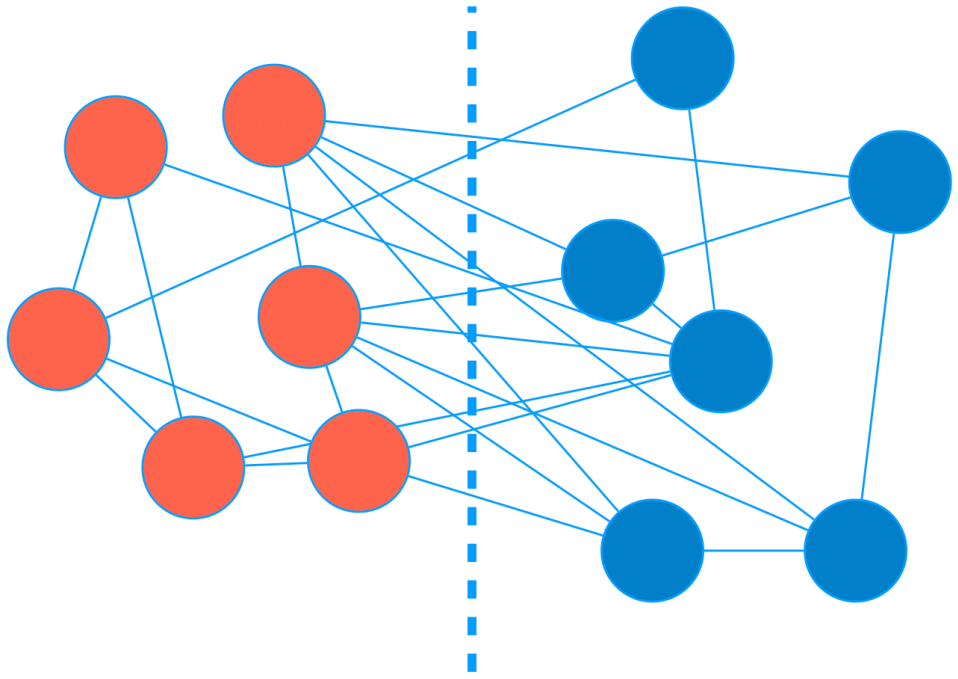
1. **Initialization:** Split nodes into two groups A and B of equal size (for bisection)
    - Start with any initial division (often random)
    - Calculate initial cut size between partitions
2. **Iterative Improvement:**
    - For each node pair (a, b) where $a \in A$ and $b \in B$:
        - Calculate the gain g_{ab} if we swap a and b
        - $g_{ab} =$ reduction in cut size after swapping
    - Select the pair (a', b') with maximum gain $g_{a'b'}$
    - Swap a' and b' even if $g_{a'b'} < 0$
    - Mark these nodes as "locked" (cannot be moved again in this pass)
    - Record the cumulative gain after this swap

1. **Pass Completion:**
    - Continue step 2 until all nodes are locked
    - Find the point during the sequence where the cumulative gain was highest
    - If this maximum gain is positive:
        - Apply only the sequence of swaps up to this point
        - Unlock all nodes and begin a new pass
    - If maximum gain is not positive, keep the original partition
2. **Termination:** Stop when a complete pass produces no improvement

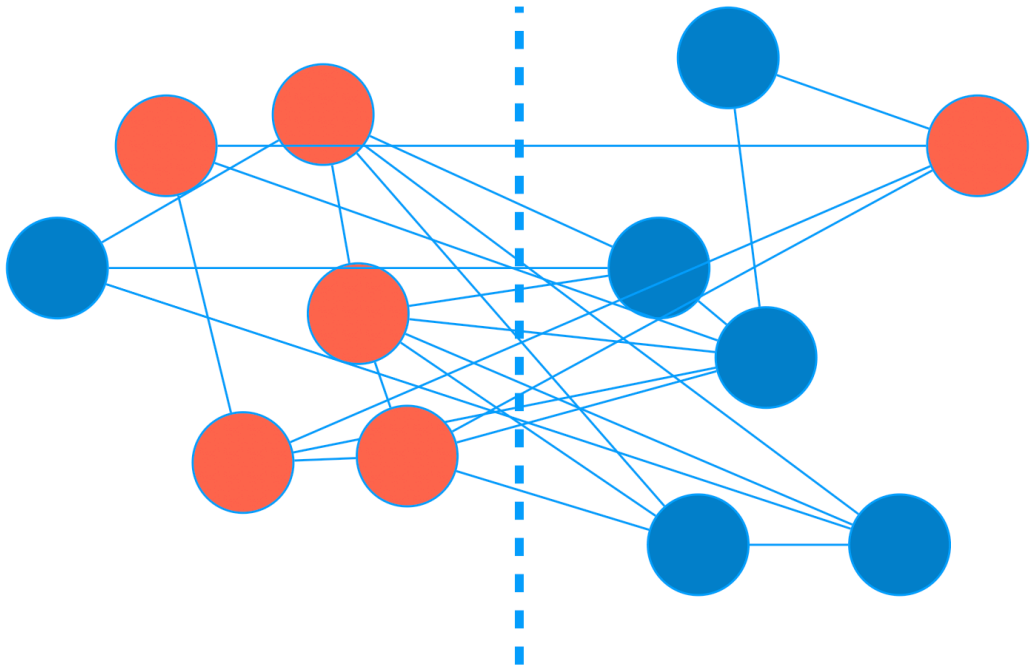
Kernighan-Lin algorithm



Kernighan-Lin algorithm

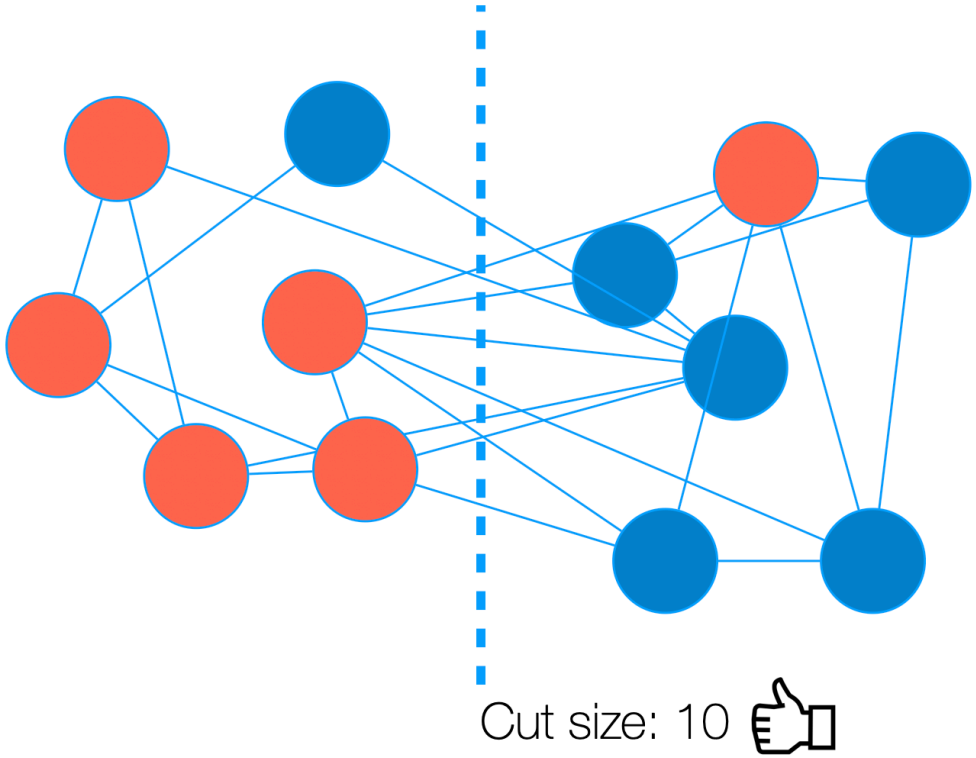


Kernighan-Lin algorithm

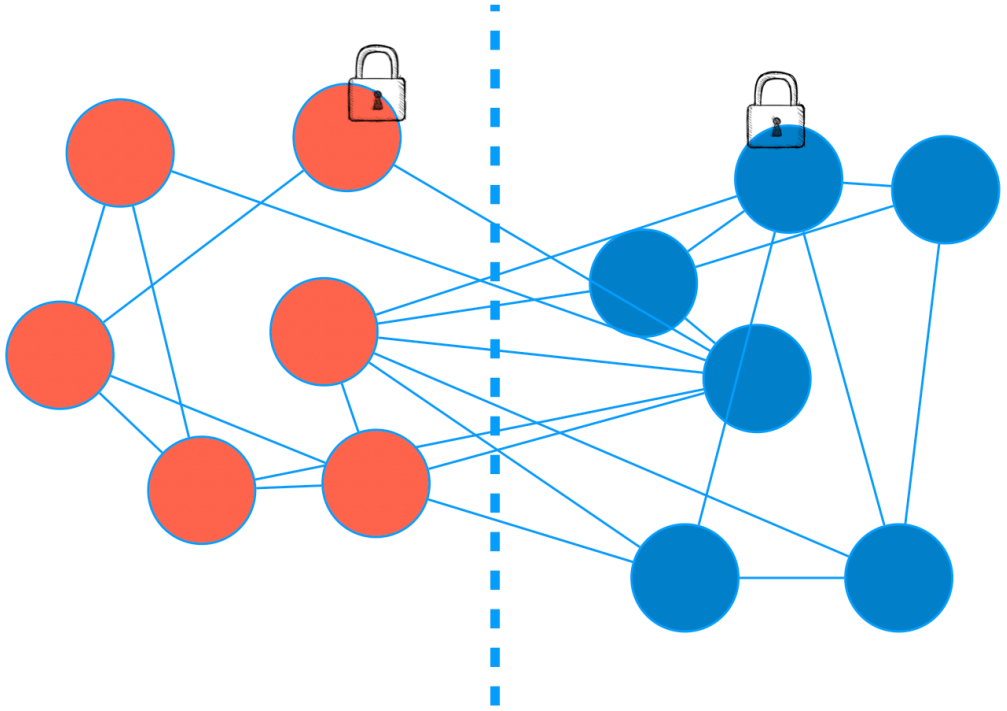


Cut size: 16 👎

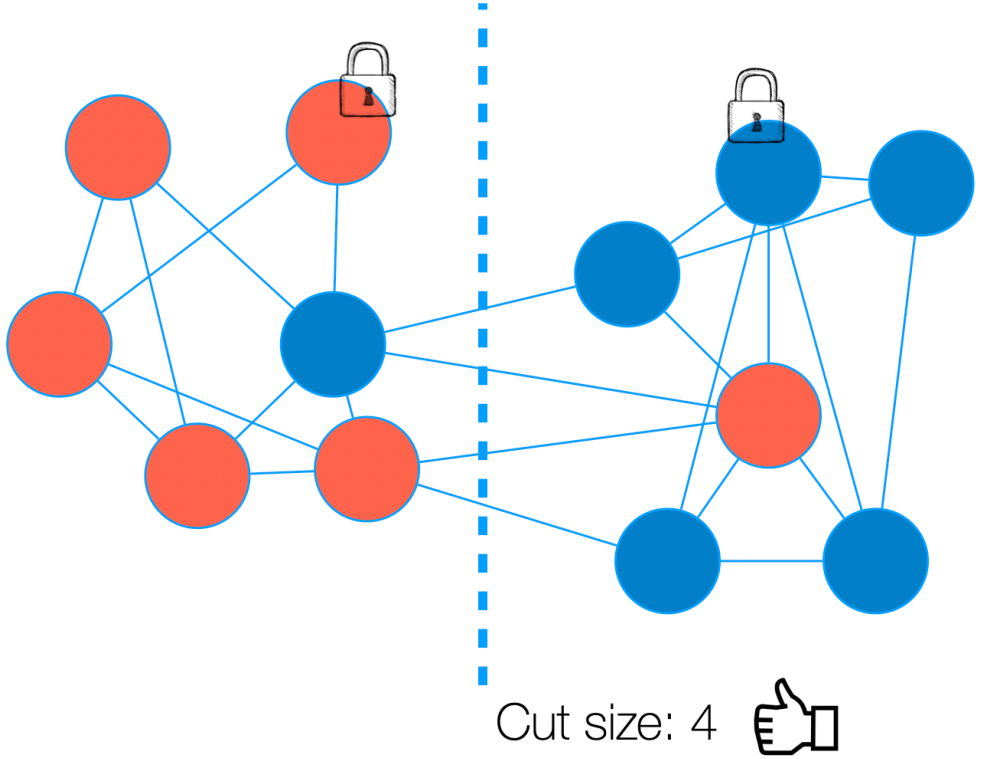
Kernighan-Lin algorithm



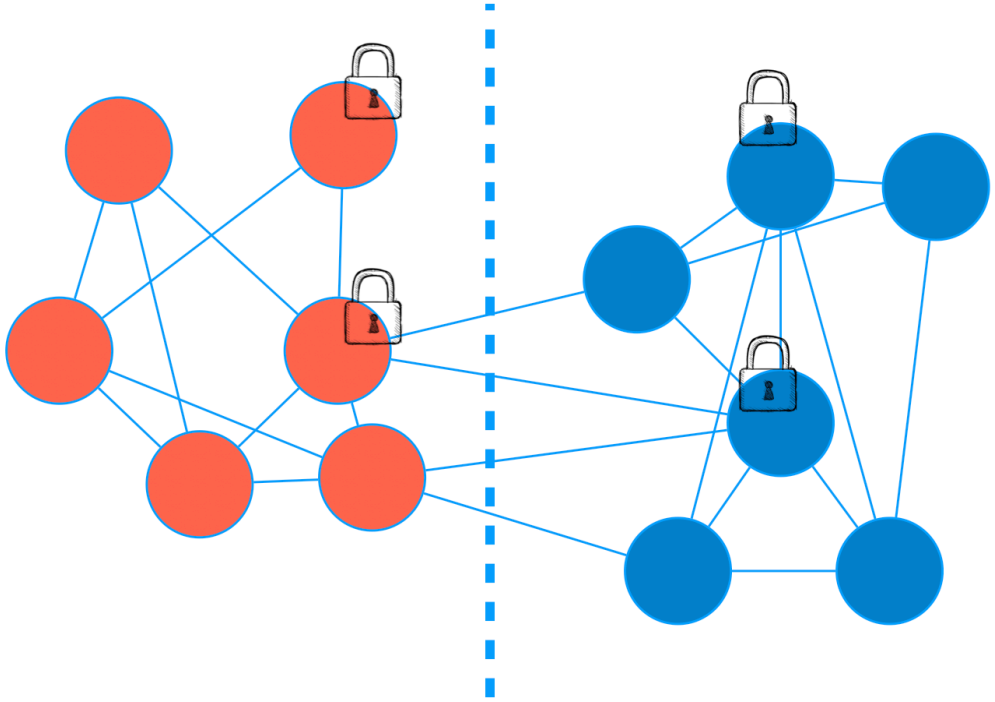
Kernighan-Lin algorithm



Kernighan-Lin algorithm



Kernighan-Lin algorithm



Kernighan-Lin algorithm

- **Convergence:** The procedure ends when the cut size of partitions obtained after consecutive iterations is the same, meaning that the algorithm is unable to improve the result
- The algorithm can be applied to minimize the cut size of partitions into more than two clusters, by swapping nodes between pairs of clusters
- **Problem: the choice of the initial partition heavily affects the final result.** The larger the cut size of the initial partition, the worse the final solution and the longer the time to reach convergence
- **Solution:** creating multiple random partitions and picking the one with the lowest cut size as initial partition

Kernighan-Lin algorithm

- The described procedure is **greedy**, in that at each step one looks for the partition with the smallest cut size. Because of that, the algorithm gets stuck in **local optima**, i.e., solutions whose cut size is not as low as it can be
- The problem can be mitigated by occasionally allowing swaps of nodes that increase the cut size
- The Kernighan-Lin algorithm is widely applied as a post-processing technique, to improve partitions delivered by other methods. Such partitions can be used as starting points for the method, which might return solutions with lower cut size

Network partitioning: limits

- Clusters have to be well-separated, but they do not need to have high internal link density
- Clusters found via graph partitioning are not communities, in general
- The number of clusters must be given as input, but it is usually unknown

In NetworkX:

```
# minimum cut bisection: returns a pair of sets of nodes

partition = nx.community.kernighan_lin_bisection(G)
```

Clustering

- **Grouping objects based on how similar to each other they are**
- Two main classes of algorithms:
  - **Hierarchical clustering**
  - **Partitional clustering**

Hierarchical clustering

- Hierarchical clustering delivers a nested set of partitions
- Main ingredient: a **similarity measure** that quantifies how alike two objects are
- Two main categories based on spatial context:
- **Spatial Embedding**
  - **Embedded in space:** Nodes have coordinates in a metric space
  - Similarity defined by distance metrics (Euclidean, Manhattan, etc.)
  - Example: Geographic locations of cellular towers in a telecommunications network
  - Interpretation: Physical or feature-space proximity indicates similarity
- **Network-Based Similarity**
  - **Not embedded in space:** Similarity derived from network structure
  - Based on connectivity patterns rather than coordinates
  - Example: Similarity between users in a social network based on mutual friends
  - Interpretation: Similar connection patterns suggest similar roles or communities
  - Can use measures like Jaccard index, structural equivalence, etc.

Similarity: structural equivalence

- **Concept:** nodes are similar if their neighbors are similar

$$S_{ij}^{SE} = \frac{\text{number of neighbors shared by i and j}}{\text{total number of nodes neighboring only i, only j or both}}$$

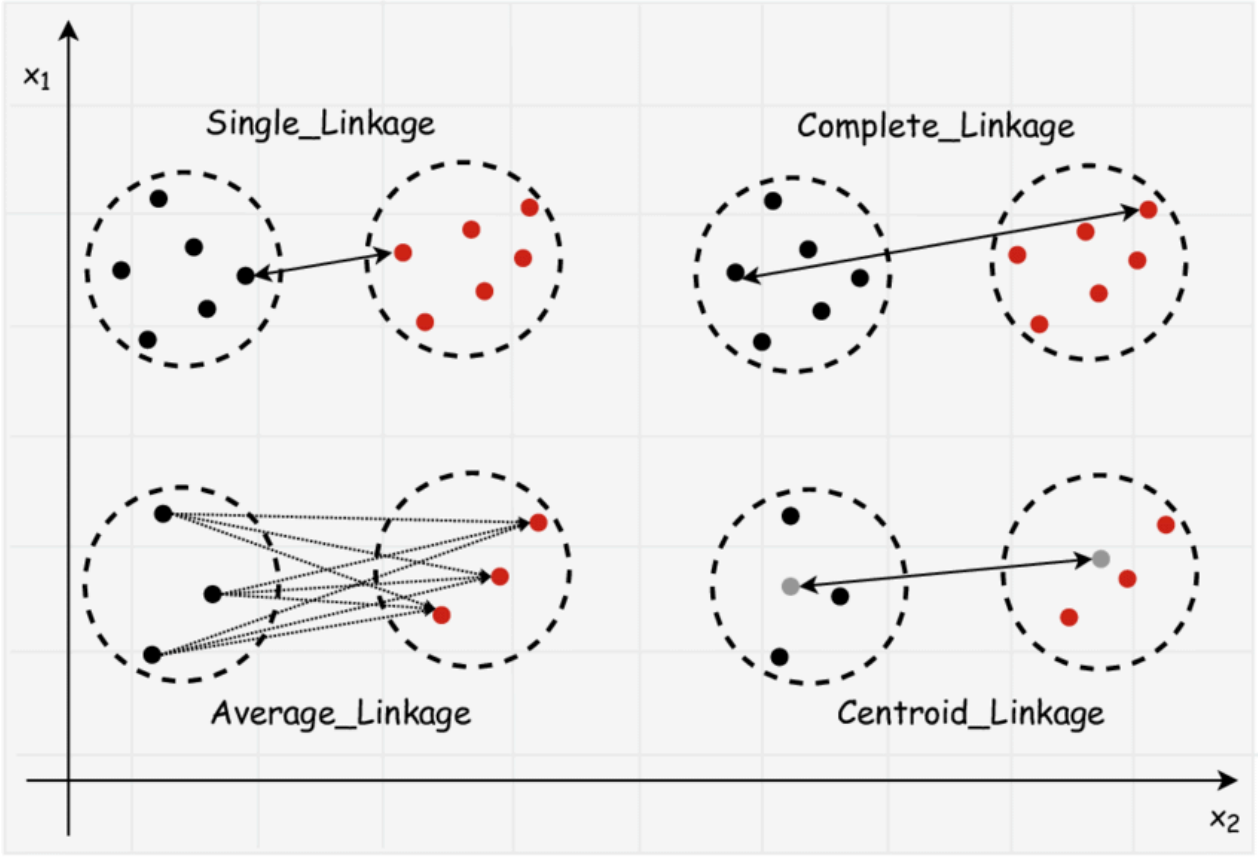
- **Examples:**
  - If the neighbors of i and j are (v_1, v_2, v_3) and (v_1, v_2, v_4, v_5), respectively, $S_{ij} = 2/5 = 0.4$, because there are two common neighbors (v_1 and v_2) out of five distinct neighbors in total $(v_1, v_2, v_3, v_4, v_5)$
  - If i and j have no neighbors in common, $S_{ij} = 0$
  - If i and j have the same neighbors, $S_{ij} = 1$

Similarity of node groups

- **Question:** how can we define the similarity S_{G_1,G_2} between two groups of nodes G_1 and G_2 via the similarity S between pairs of nodes?
- **Answer:** multiple approaches. The first step is to compute the pairwise similarity S_{ij} between each node i in G_1 and each node j in G_2.

Then the following strategies can be adopted:

- **Single linkage:** take the maximum pairwise similarity $\rightarrow S_{G_1,G_2} = max_{i,j} S_{ij}$
- **Complete linkage:** take the minimum pairwise similarity $\rightarrow S_{G_1,G_2} = min_{i,j} S_{ij}$
- **Average linkage:** take the average pairwise similarity $\rightarrow S_{G_1,G_2} = \langle S_{ij} \rangle_{i,j}$


x1
Single_Linkage
Complete_Linkage
Average_Linkage
Centroid_Linkage
x2

Choosing a linkage method

- **Single linkage:**
  - Creates **elongated**, chain-like structures where clusters can merge through single connections
  - **Good for:** detecting extended or chain-like structures
  - **Applications:** detecting connected components, identifying outlying communities
- **Complete linkage:**
  - Considers farthest elements between clusters
  - Creates **compact, spherical** clusters of similar sizes
  - **Good for:** finding tightly-knit communities with similar diameters
  - **Applications:** social circles, functional modules with strong internal cohesion
- **Average linkage:**
  - Balanced approach considering all pairwise relationships
  - Creates **intermediate** clusters, less affected by outliers
  - **Good for:** general-purpose community detection with unknown structure
  - **Applications:** most social and biological networks with moderate noise

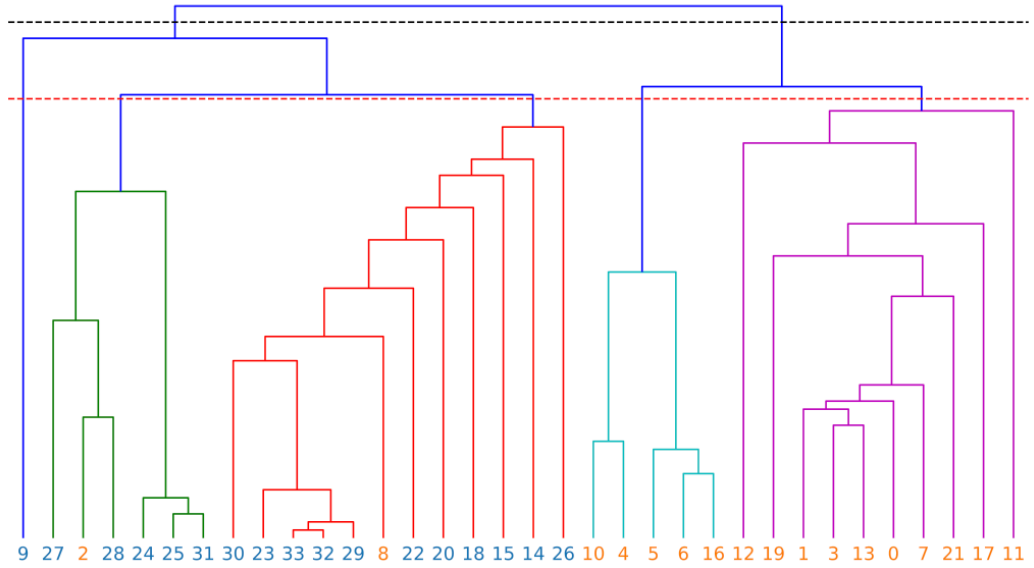
Hierarchical clustering

Two approaches

- **Agglomerative** partitions are generated by iteratively merging groups of nodes
- **Divisive** partitions are generated by iteratively splitting groups of nodes

Dendrograms

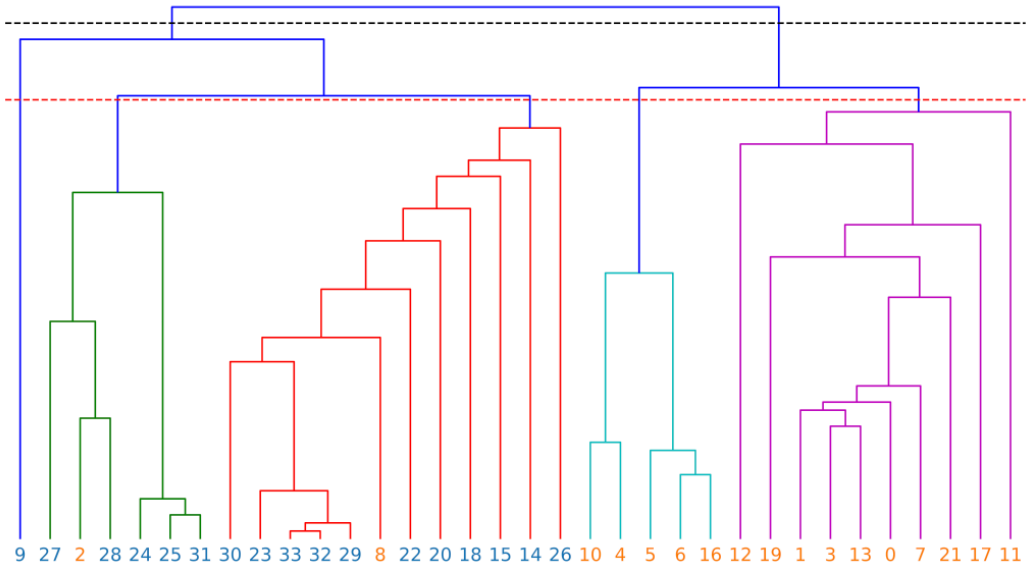
- Outcome: **dendrogram (hierarchical tree)**
- A dendrogram is a compact summary of all partitions created by hierarchical clustering
- Since each merger reduces the number of groups by one, **the total number of partitions is** N



Dendrograms

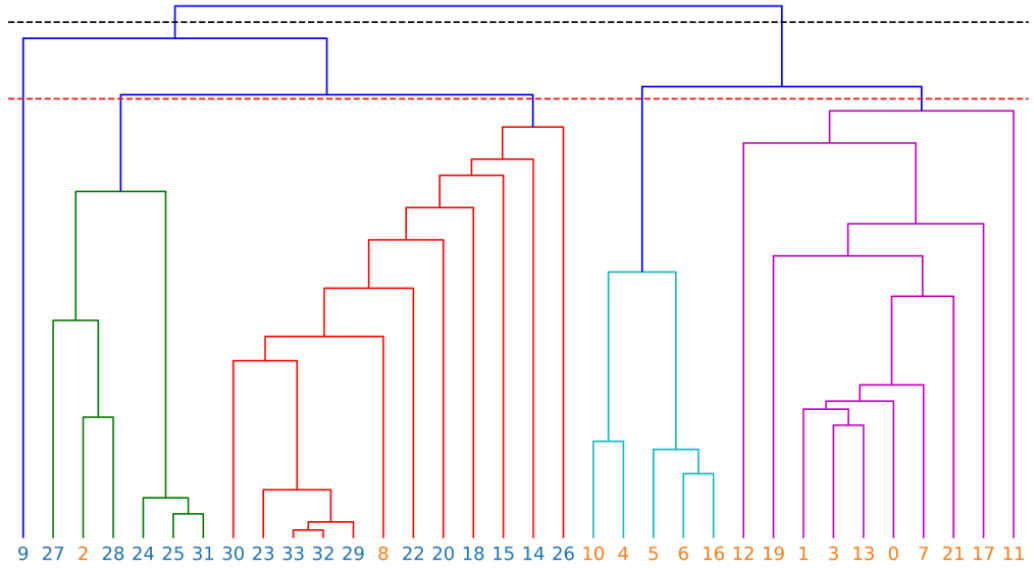
Features:

- **Bottom: leaves** of the tree, indicated by the labels of the nodes
- Going upwards, pairs of clusters are merged. Mergers are illustrated by horizontal lines joining two vertical lines, each representing a cluster
- The nodes of each cluster can be identified by following the vertical line representing the cluster all the way down



Dendrograms

- Partitions are selected via **horizontal cuts** of the dendrogram: the clusters are the ones corresponding to the vertical lines severed by the cut
- High cuts yield partitions into a few large clusters, low cuts yield partitions into many small clusters
- **Hierarchy:** each partition has clusters including clusters of all partitions lying lower in the dendrogram



Reading material

References

[ns1] **Chapter 6 - Communities**

This paper overviews multiple perspectives for community detection in complex systems

- M. T. Schaub et al., The many facets of community detection in complex networks, Applied Network Science 2:4 (2017) https://doi.org/10.1007/s41109-017-0023-6

UNIVERSITÀ DI TORINO

Q&A