Graph Learning
4. Hierarchical Clustering

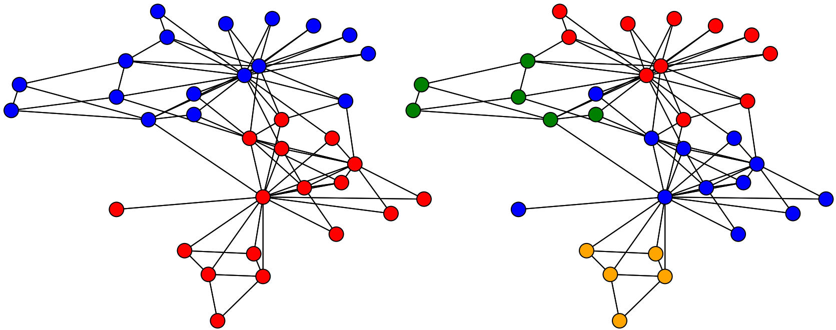
Thomas Bonald

2024 – 2025

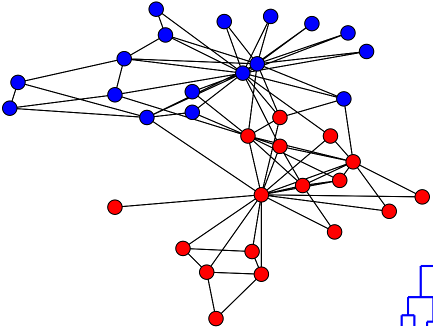
INSTITUT
POLYTECHNIQUE
DE PARIS

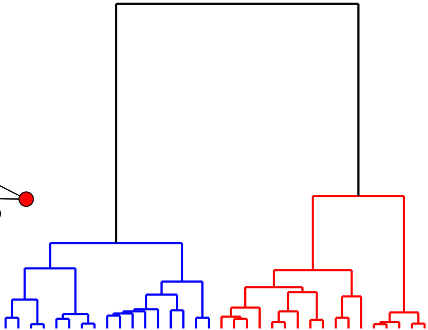
Motivation

- What is a **good** clustering?
- Which **resolution**?



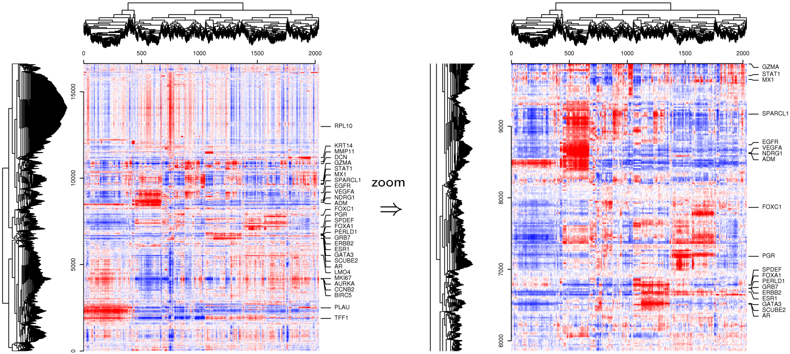
Hierarchical clustering





Example in biology

2,035 tumors, 16,634 non-redundant genes



Wirapati 2009

Hierarchical clustering: vector data

Divisive algorithms

- e.g., through successive k-means

Agglomerative algorithms

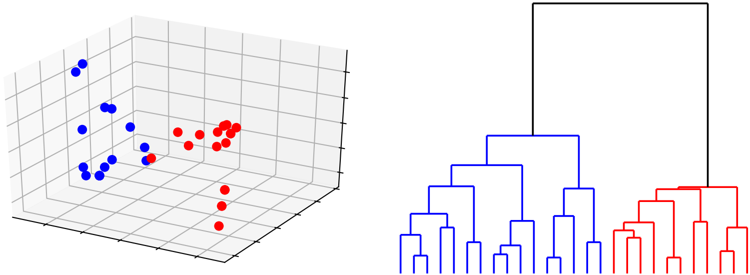
- Successive merges of the closest clusters $a, b \subset \{1, \ldots, n\}$

Linkage	$d(a,b)$
Single	$\min_{i \in a, j \in b} \lvert\lvert x_i - x_j \rvert\rvert$
Complete	$\max_{i \in a, j \in b} \lvert\lvert x_i - x_j \rvert\rvert$
Average	$\frac{1}{\lvert a\rvert\lvert b\rvert} \sum_{i \in a, j \in b} \lvert\lvert x_i - x_j \rvert\rvert$
Ward	$\frac{\lvert a\rvert\lvert b\rvert}{\lvert a\rvert+\lvert b\rvert} \lvert\lvert g_a - g_b \rvert\rvert^2$

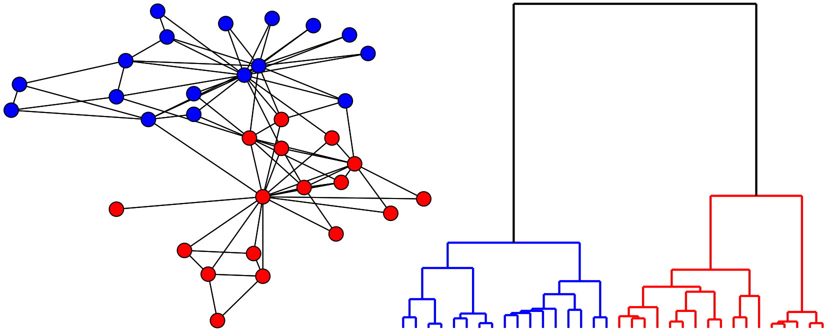
  Lance & Williams 1967

- Local search by the **nearest-neighbor chain**
  Murtagh 1983

Hierarchical clustering: vector data



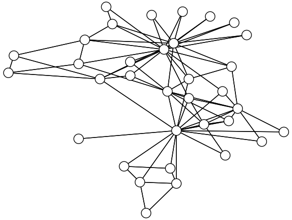
Hierarchical clustering: graph data



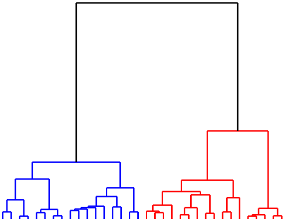
Outline

1. Notion of dendrogram
2. Divisive algorithm
3. Agglomerative algorithm
4. Extensions

if the solution is unsatisfying, we can go back/down on the dendrogram to obtain other solutions



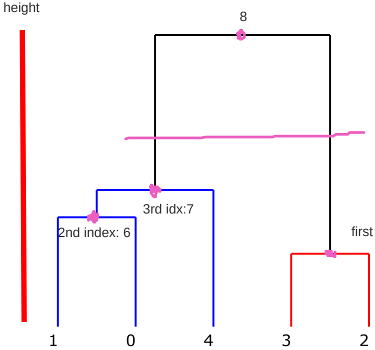
$\rightarrow$



Dendrogram

the depth of a dendrogram is the number of levels from the leaves to the root (entire dataset in one cluster)

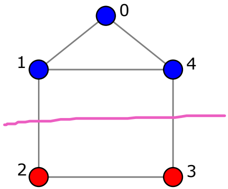
last column= number of points at that stage



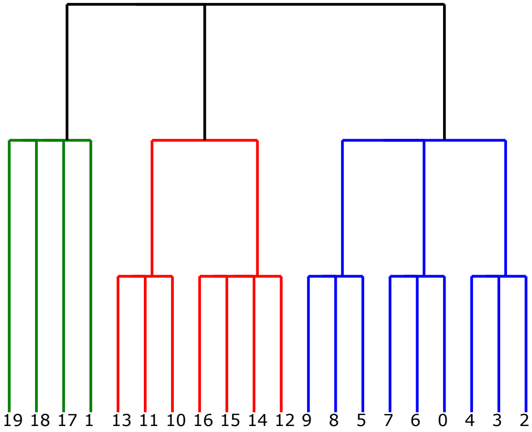
first available index: 5

		height	n of clusters in this stage	
2	3	x	2	1st cluster
0	1	x	2	
4	6	x	3	
5	7	x	5	

n - 1 rows : 4 rows for 5 points (n)



A tree

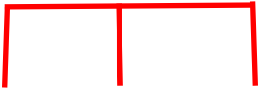


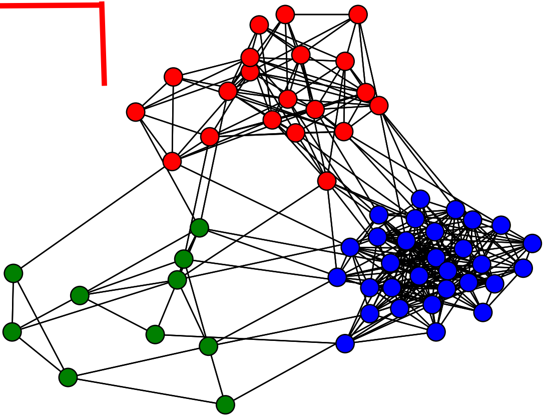
Outline

1. Notion of dendrogram
2. **Divisive algorithm**
3. Agglomerative algorithm
4. Extensions

Clustering by Louvain

the beginning of the dendrogram will look
like this (bc there are 3 clusters)





Hierarchical clustering by Louvain

Input: Graph G

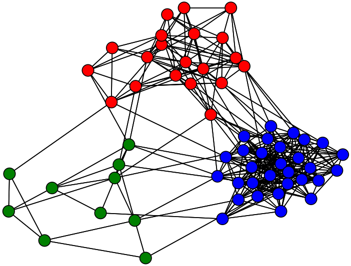
LouvainIteration(G):
  clusters $\leftarrow$ Louvain(G)
  if $|$clusters$| > 1$:
- subgraphs $\leftarrow$ GetSubgraphs(G, clusters)
- **return** [LouvainIteration(S) for S in subgraphs]

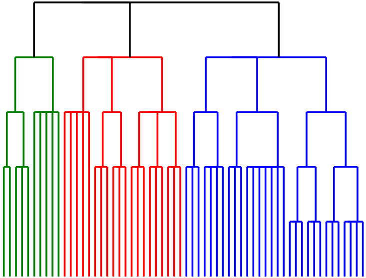
  else:
- **return** [nodes(G)]

Hierachical clustering by Louvain

k-means vs Louvain: k-means is for vectors, specify how many clusters you want, but not in louvain

louvain needs to be run at each step -> prefer automatic

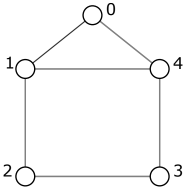




Exercise

What is the hierarchical structure of the house graph?

with louvain: the 1st cluster would be {0,1,4},{2,3}



Outline

1. Notion of dendrogram
2. Divisive algorithm
3. **Agglomerative algorithm**
4. Extensions

how to find the strongest connection between 2 nodes? sampling

Sampling

- Edge sampling:

$$p(i,j) = \frac{A_{ij}}{v}$$

v = sum of all entries in the adjacency matrix
v = 1^TA1

- Marginal distribution:

$$p(i) = \sum_{j \in V} p(i,j) = \frac{d_i}{v}$$

- Conditional distribution:

prob of sampling i coming from j,
prob of seeing this conection coming from j

$$p(i|j) = \frac{p(i,j)}{p(j)} = \frac{A_{ij}}{d_j}$$

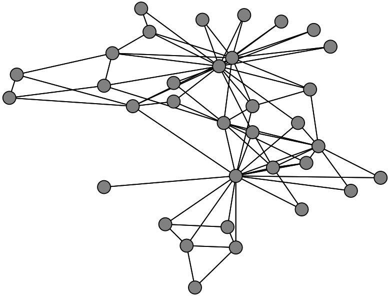
p(i, j)/p(i) = (Aij / di dj) * v = p(j, i)/p(j) due to simetry

Agglomerative algorithm

Idea: Merge successively the nodes with the “strongest” link

Link strength

$$\sigma(i,j) = \frac{p(j|i)}{p(j)} = \frac{p(i|j)}{p(i)} = \frac{p(i,j)}{p(i)p(j)} = v\frac{A_{ij}}{d_i d_j}$$

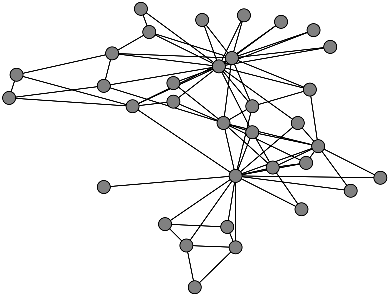


Agglomerative algorithm

Equivalently, merge successively the two "closest" nodes

Distance

$$d(i,j) = \frac{1}{\sigma(i,j)} = \frac{d_i d_j}{v A_{ij}}$$



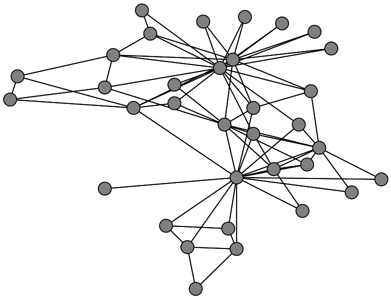
Merging two nodes

Idea: Aggregate external links, add a self-loop for internal links

New (weighted) adjacency matrix

$$A_{i \cup j, k} \leftarrow A_{i,k} + A_{j,k} \quad \forall k \in V \setminus \{i, j\}$$

$$A_{i \cup j, i \cup j} \leftarrow A_{i,i} + A_{j,j} + 2A_{i,j}$$



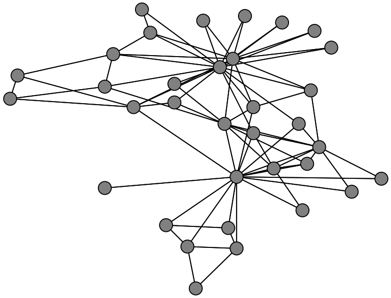
Merging two nodes

Equivalently, update the sampling

New sampling distribution

$$p(i \cup j, k) = p(i, k) + p(j, k), \quad \forall k \in V \setminus \{i, j\}$$

$$p(i \cup j, i \cup j) = p(i, i) + p(j, j) + 2p(i, j)$$

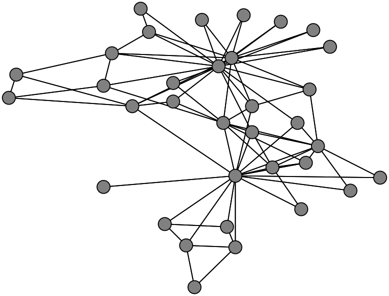


Merging two nodes

New link strengths

$$\forall k \neq i, j, \quad \sigma(i \cup j, k) = \frac{p(i)}{p(i) + p(j)} \sigma(i, k) + \frac{p(j)}{p(i) + p(j)} \sigma(j, k)$$

$$p(i \cup j) = p(i) + p(j)$$



Paris[1] algorithm

Paris

Input: Graph $G = (V, E)$ with $V = \{1, \ldots, n\}$

For $t = 1, \ldots, n-1$

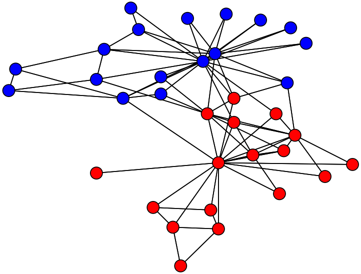
- $i, j \leftarrow \arg\max_{i,j \in V, i \neq j} \sigma(i, j)$
- merge i, j into node $n + t$
- update σ

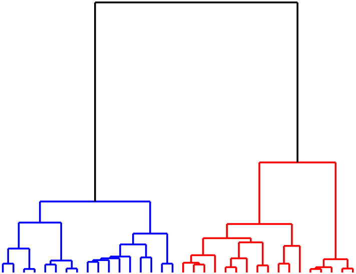
Output: List of merges

B, Charpentier, Galland, Hollocou 2018

[1] Paris = Pairwise AgglomeRation Induced by Sampling

Dendrogram





Distance

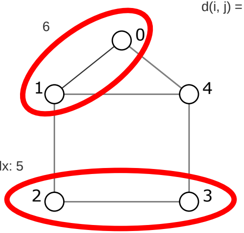
$$d(i,j) = \frac{1}{\sigma(i,j)} = \frac{d_i d_j}{v A_{ij}}$$

Exercise

Give the dendrogram returned by Paris on the house graph:

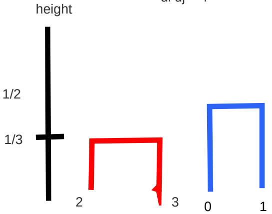
d(i, j) = (di dj) / (Aij v) = 4 / 12

new idx: 5



2	3	1/3	2
0	1	1/2	2
	...		

di dj = 4

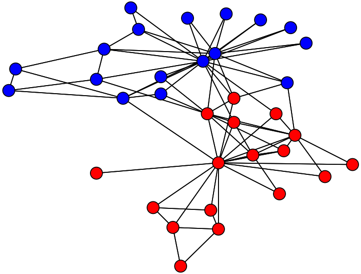


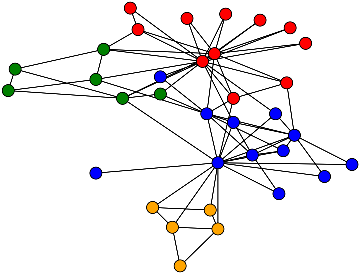
Modularity

Modularity at resolution γ:

$$Q_\gamma(C) = \frac{1}{v} \sum_{i,j \in V} \left(A_{ij} - \gamma \frac{d_i d_j}{v} \right) \delta_{C(i),C(j)}$$

The fit ($\gamma \to 0$) vs diversity ($\gamma \to +\infty$) trade-off





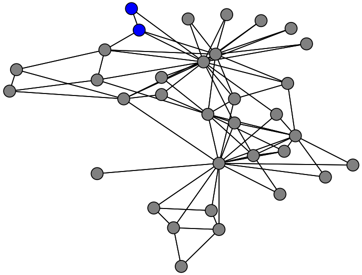
Resolution limit

Proposition

The resolution limit of Louvain, beyond which all clusters have size 1, is the maximum **link strength**:

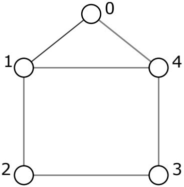
$$\gamma = \max_{i \neq j} \sigma(i, j)$$

Consequence: The first node pair i, j merged by Paris is that merged by Louvain at the resolution limit.



Exercise

Give the resolution limit of Louvain on the house graph:

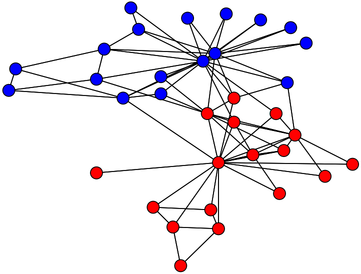


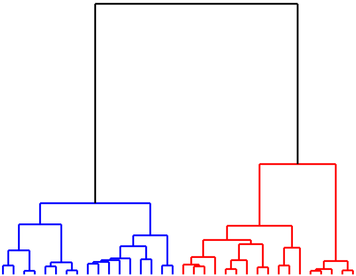
Reducibility

Proposition

$$d(i \cup j, k) \geq \min(d(i,k), d(j,k))$$

Consequence: The sequence of distances from any leaf to the root is **non-decreasing**.





The nearest-neighbor chain

The complexity of the basic algorithm is in $O(nm)$.
More efficient approach through the **nearest-neighbor chain**:

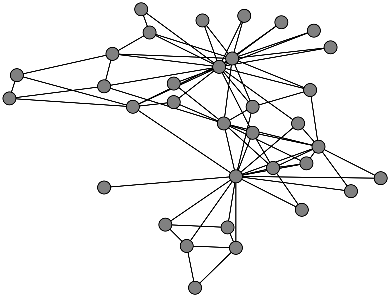
Paris with the NN chain

Input: Graph $G = (V, E)$ with $V = \{1, \ldots, n\}$
While $|V| > 1$:

- take a node at random
- build the chain of nearest-neighbors
- merge the two last nodes of this chain
- update σ
- restart the chain

Output: List of merges

Example



Outline

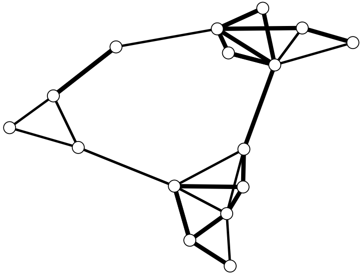
1. Notion of dendrogram
2. Divisive algorithm
3. Agglomerative algorithm
4. **Extensions**

Case of weighted graphs

Let $G = (V, E)$ be a **weighted** graph with adjacency matrix A
Let $w = A1$ be the vector of node weights

Link strength

$$\sigma(i,j) = \frac{p(j|i)}{p(j)} = \frac{p(i|j)}{p(i)} = \frac{p(i,j)}{p(i)p(j)} = v\frac{A_{ij}}{w_i w_j}$$

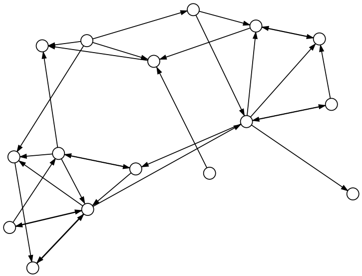


Case of directed graphs

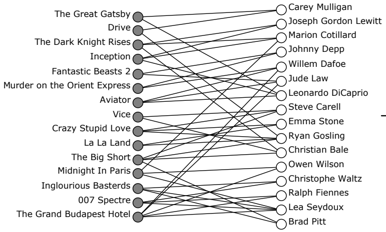
Let $G = (V, E)$ be a **directed** graph with adjacency matrix A
Let $d^+ = A1$ and $d^- = A^T 1$ be the vectors of out/in-degrees

Link strength

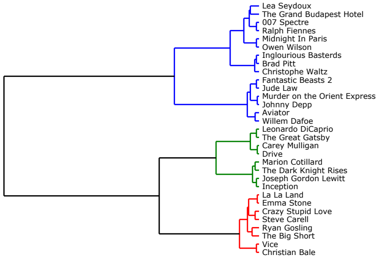
$$\sigma(i,j) = \frac{p(i,j) + p(j,i)}{p^+(i)p^-(j) + p^+(j)p^-(i)} = v\frac{A_{ij} + A_{ji}}{d_i^+ d_j^- + d_j^+ d_i^-}$$



Case of bipartite graphs

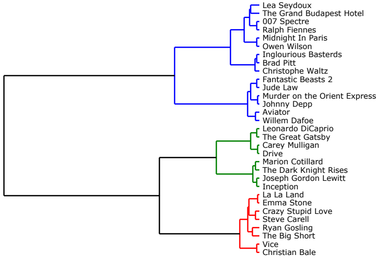


$\rightarrow$

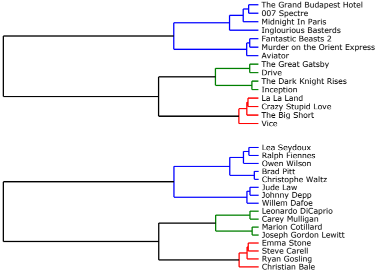


Co-clustering

Separate dendrograms



$\rightarrow$



Summary

Hierarchical clustering

- Useful to **cluster** graphs at different resolutions
- **Louvain Iteration** $\rightarrow$ divisive algorithm
- **Paris** $\rightarrow$ agglomerative algorithm
- Applicable to **weighted**, **directed** and **bipartite** graphs