https://sites.google.com/site/antonioscalaphys/

Multivariate analysis and complex networks

4: Communities & Partitions

Antonio Scala

CNR-ISC, the Institute for Complex Systems

INFN School of Statistics
Ischia, May 2017

Divide and conquer ?

- Spin Models
  - find communities like magnetic domains
- Flow trapping
  - define dynamics and observe stagnation
- Minimum-cut methods
  - find communities with minimum inter-linkage
- Hierarchical clustering
  - join recursively less and less connected communities
- Girvan–Newman algorithms
  - remove links among communities
- Modularity maximization
  - maximize target function over communities

Spin Models

Potts model with q states$\sigma_i \in \{1 \dots q\}$; each state correspond to a possible community. Order parameter are the fraction of nodes in the state s i.e. $n_s = n^{-1} \sum \delta_{\sigma_i s}$.

$$\mathcal{H} = -J \sum_{ij} A_{ij} \delta_{\sigma_i \sigma_j} + \lambda \sum_{s=1}^{q} \frac{n_s \left(n_s - 1\right)}{2}$$

To find the communities, minimise $\mathcal{H}$

- classical ferromagnetic Potts model energy favouring spin alignment
- second term favours homogeneously distributed spins
- the ratio γ/J allows to tune the fragmentation of the system

Walktrap

- Random walk is described by probability flows
- The transition matrix $M = D^{-1}A$ is Markov
- Strong flows in clusters, weak flows in-between

Walktrap uses short random walks of length t to detect communities:

- Calculate the matrix P^t giving the probability of going from a node to another in t random walk steps
- Define a distance among nodes based on P^t
- Use clustering algorithms (Ward's method) to find communities

Markov Clustering

- Random walk is described by probability flows
- The transition matrix $M = D^{-1}A$ is Markov
- Strong flows in clusters, weak flows in-between

MCL enhances strong flows, by repeating the following step:

- Expansion: $M \to \Phi_q M = M^q$
  - corresponds to perform q random walk steps
- Inflation: $M_{ij} \to (M_{ij})^r / \sum_j (M_{ij})^r$
  - non-linear operator that sends to 0 smaller values of M_{ij}

until M has converged to its fixpoint

$$M^* = \Gamma_r \Phi_q M^*$$

Laplacian

A adjacency matrix

sparse matrix with $A_{ij} = 1$ iff nodes i and j are linked

D degree matrix

diagonal matrix with $D_{ii} = \sum_j A_{ij}$ degree d_i of node i

$\mathcal{L}$ Laplacian matrix

$$\mathcal{L} = D - A$$

Laplacian and Diffusion

- Diffusion in the network is dictated by the Laplacian matrix

$$\partial_t \rho = -\mathcal{L}\rho$$

- The eigenvalues of $\mathcal{L}$ are $\lambda_1 = 0 \leq \lambda_2 \leq \ldots \leq \lambda_N$

The first non-zero eigenvalue λ_2 is the inverse timescale of slowest mode of diffusion (the most extended mode). In general, we can think of λ_2^{-1} as the timescale after which a perturbation (like the infection of a site) that spreads diffusively will settle a new state (like an epidemics) *in the whole network*.

Laplacian and vibration

- Network vibrations are dictated by the Laplacian matrix

$$\partial_t^2 \rho = -\mathcal{L}\rho$$

- Synchronizability is linked to the spectrum of $\mathcal{L}$
- Controllability is linked to the spectrum of $\mathcal{L}$

Minimal Cut

A partition of the nodes into two sets can be represented by a vector $\vec{x}$ with $x_i \in \{-1, 1\}$

- V_+ sites with $x_i > 0$
- E_+ links between nodes in V_+
- ∂V sites at the border of the partitions
- ∂E links among V_+ and V_-

Min-Cut

find $\min \mathcal{H}[\vec{x}]$ s.t. $x_i \in \{-1, 1\}$

$$\mathcal{H}[\vec{x}] = \sum_{ij}^{n.n.} \left(\frac{x_i - x_j}{2} \right)^2 = \frac{\vec{x}\mathcal{L}\vec{x}}{4}$$

Min-Cut & Laplacian

Relax the $min - cut$ conditions and let $\vec{x} \in \mathbb{R}^N$ s.t $\|\vec{x}\| = 1.$; then I can look at the *eigenvectors* $\vec{u}_\alpha$ *of* $\mathcal{L}$
by expressing $\vec{x} = \sum_\alpha a_\alpha \vec{u}_\alpha$ we get the relation

$$\vec{x}\mathcal{L}\vec{x} = \sum_\alpha a_\alpha^2 \lambda_i \geq \lambda_2 \|\vec{x}\|^2 = \lambda_2$$

therefore the minimal solution is $\vec{u}_2$

Eigenvectors & Partitions

Laplacian eigenvectors $\vec{u}_\alpha$ are zero sum and orthogonal

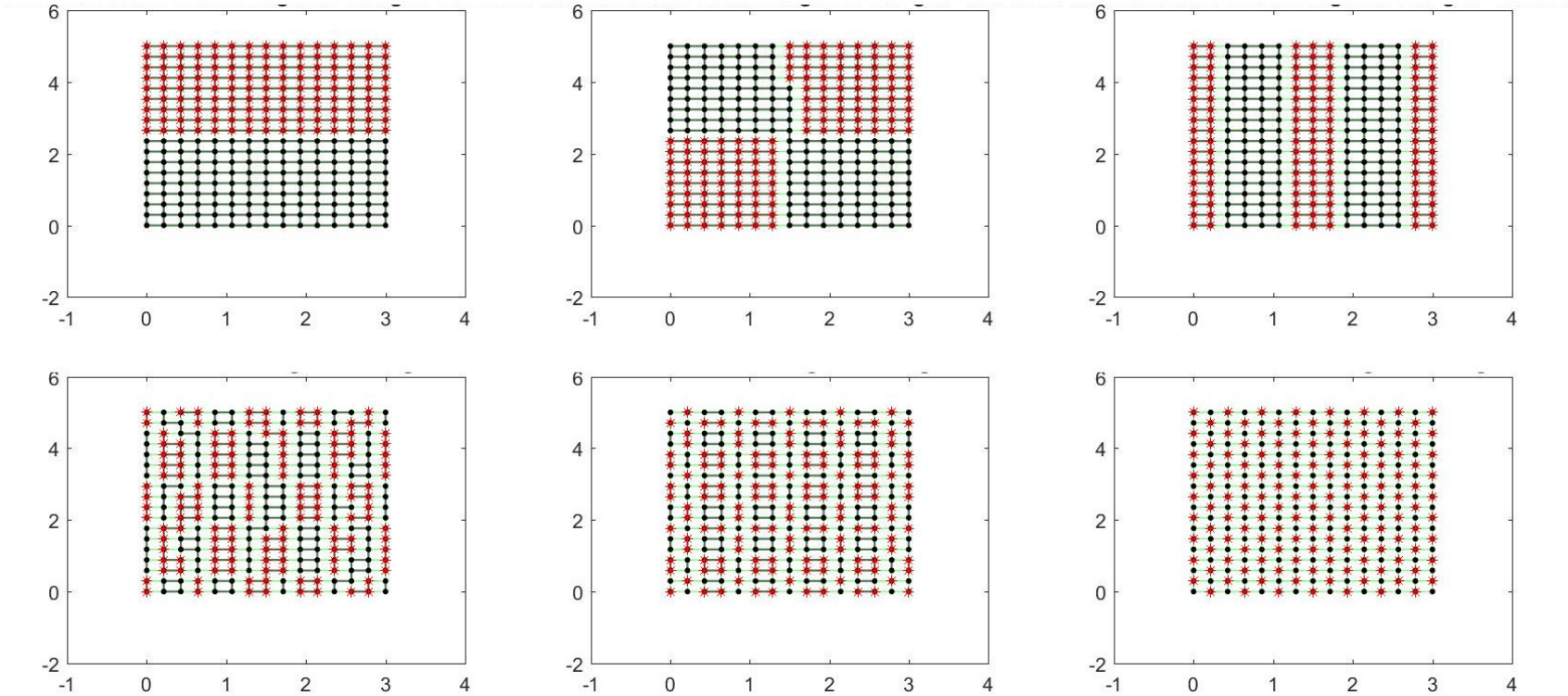
$$\sum_i (\vec{u}_\alpha)_i = 0 \ , \ \ \vec{u}_\alpha * \vec{u}_\beta \propto \delta_{\alpha\beta}$$

hence for each node i and for each $\alpha < n$ I have the signature

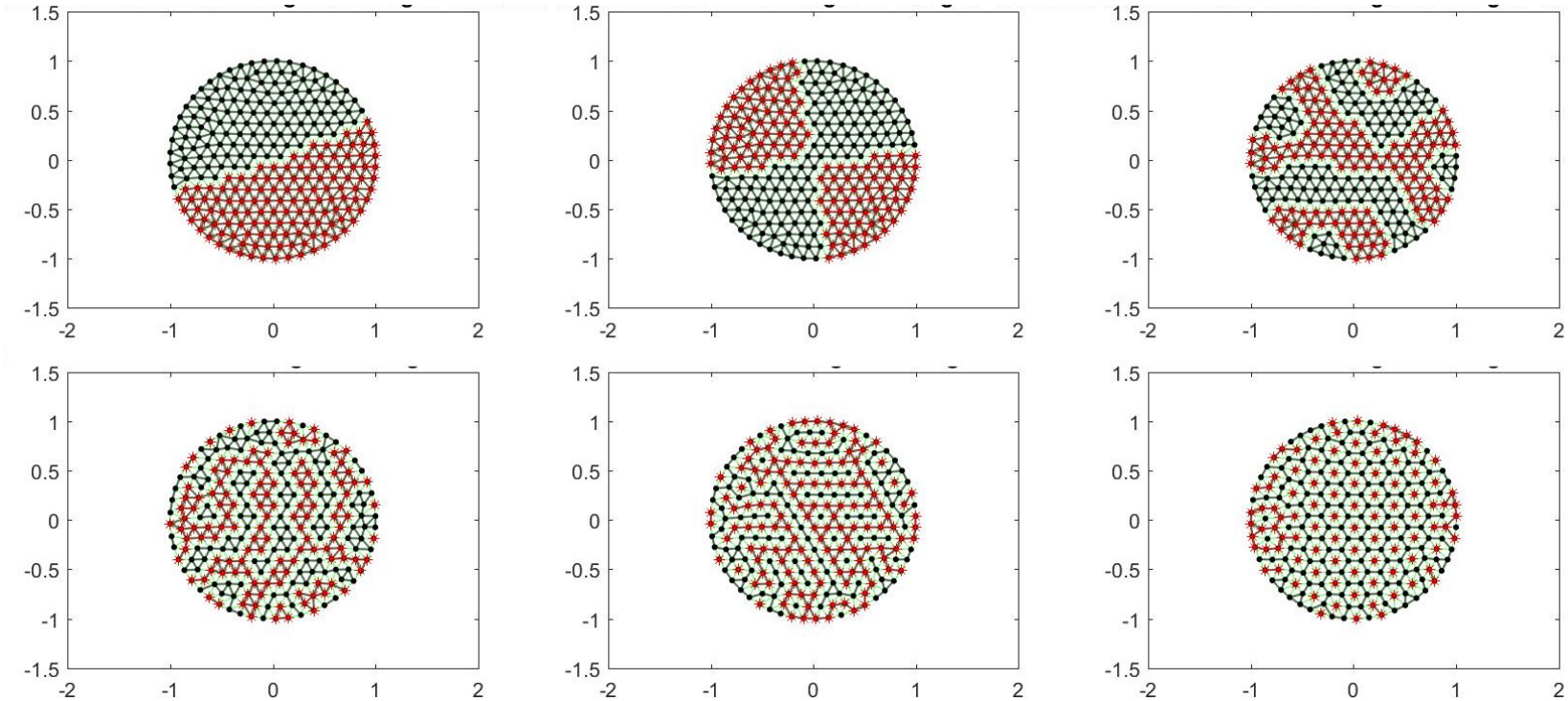
$$s_i(\alpha) = [sign\,(\vec{u}_1)_i\,, sign\,(\vec{u}_2)_i\,, \ldots, sign\,(\vec{u}_\alpha)_i]$$

that assumes 2^α different values; grouping the node $i \in V$ according to their signature $s_i(\alpha)$, we can divide the graph in at most 2^α clusters

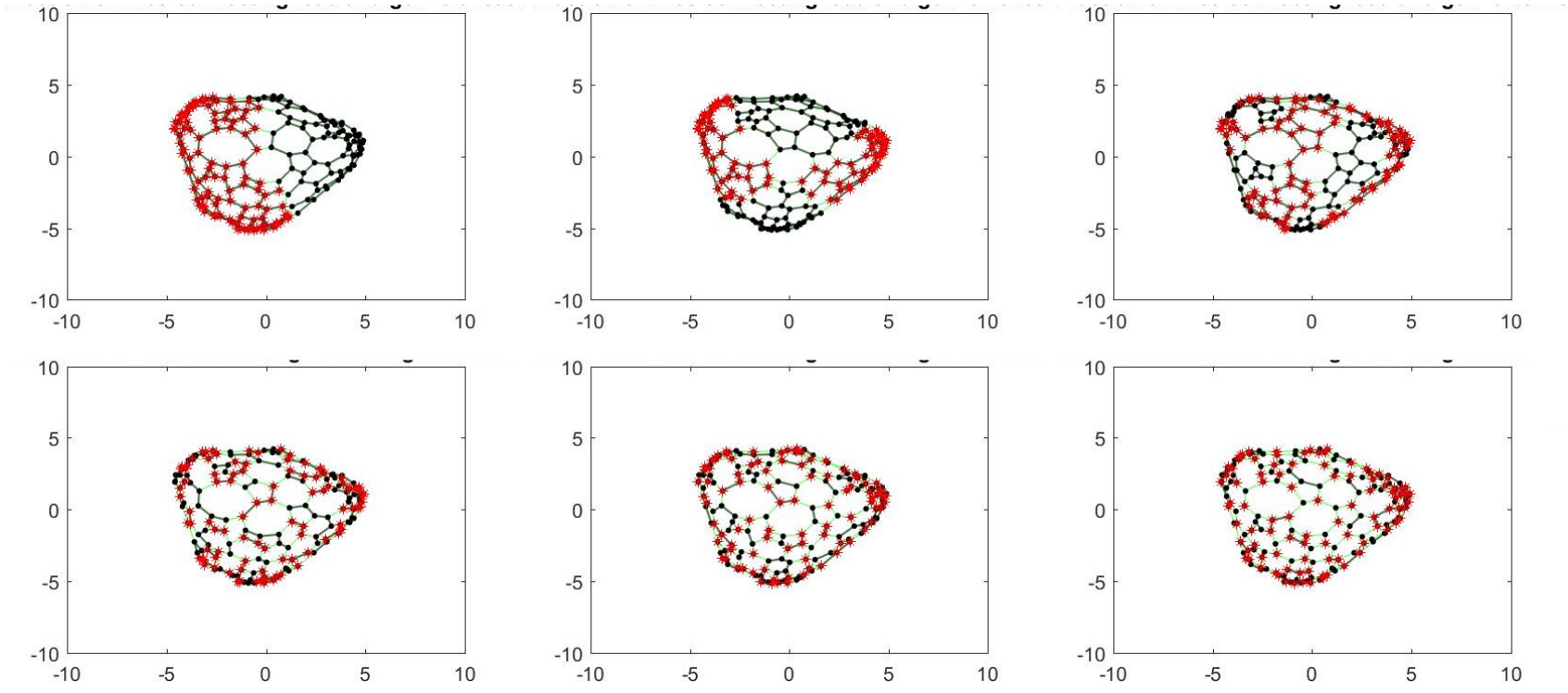
Square Lattice



Triangular Lattice



Random Planar Lattice

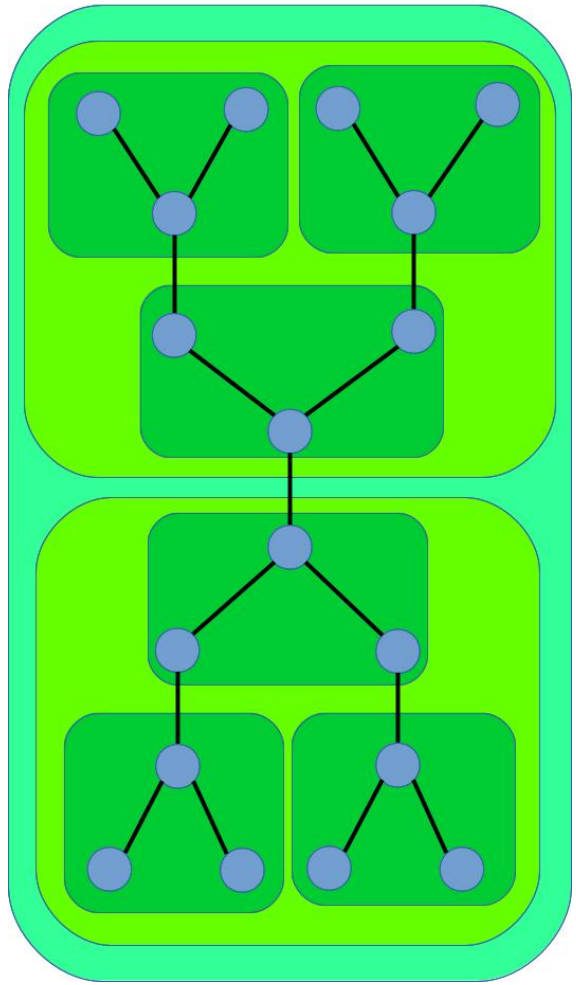


Girman-Newman method

The idea is that edges with high *edge betweeness* are kind of bridges that are mostly likely "between" the communities

1. Calculate edge betweeness
2. Remove edge with highest betweeness
3. If there are still edges repeat from step

One can also fix in advance the number of communities as a stopping criterion

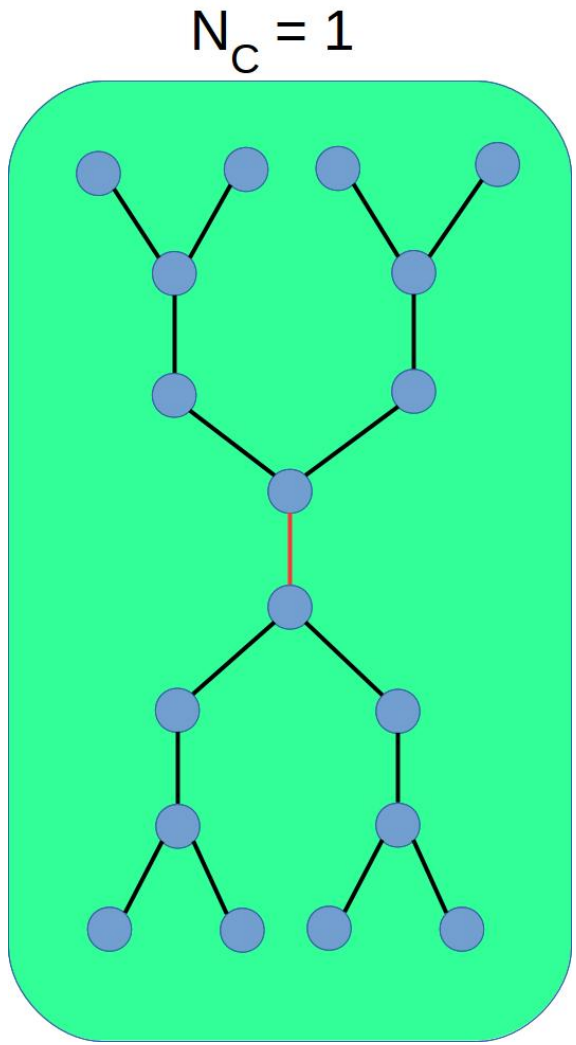


Girman-Newman method

The idea is that edges with high *edge betweeness* are kind of bridges that are mostly likely "between" the communities

1. Calculate edge betweeness
2. Remove edge with highest betweeness
3. If there are still edges repeat from step

One can also fix in advance the number of communities as a stopping criterion

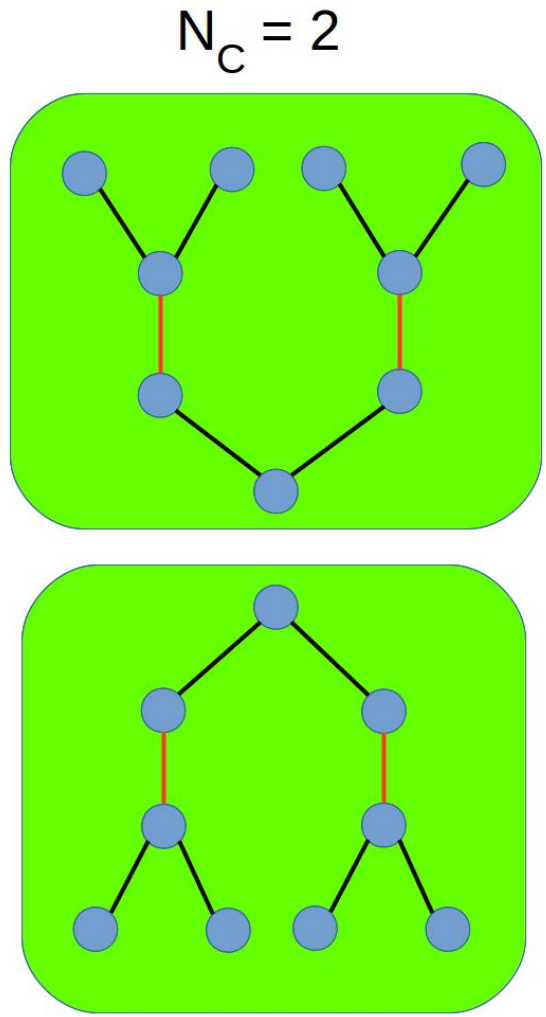


Girman-Newman method

The idea is that edges with high *edge betweeness* are kind of bridges that are mostly likely "between" the communities

1. Calculate edge betweeness
2. Remove edge with highest betweeness
3. If there are still edges repeat from step

One can also fix in advance the number of communities as a stopping criterion

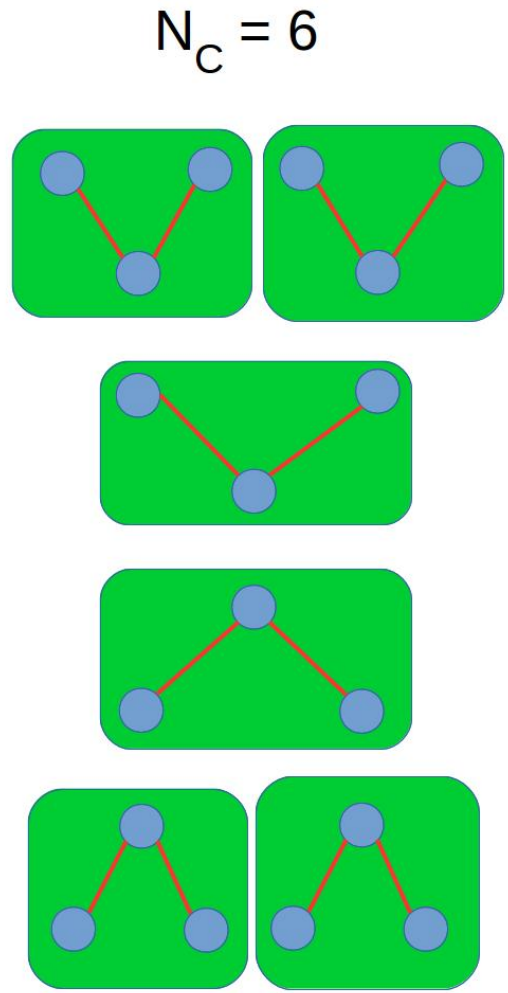


Girman-Newman method

The idea is that edges with high *edge betweeness* are kind of bridges that are mostly likely "between" the communities

1. Calculate edge betweeness
2. Remove edge with highest betweeness
3. If there are still edges repeat from step

One can also fix in advance the number of communities as a stopping criterion

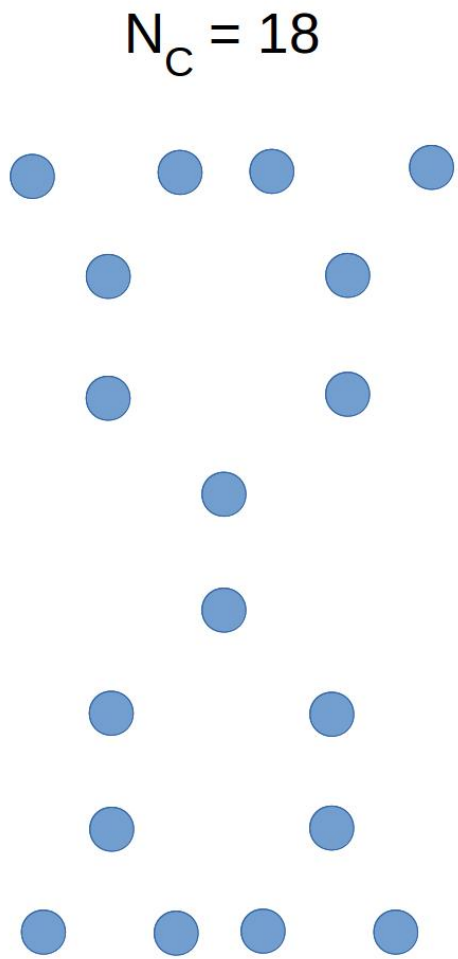


Girman-Newman method

The idea is that edges with high *edge betweeness* are kind of bridges that are mostly likely "between" the communities

1. Calculate edge betweeness
2. Remove edge with highest betweeness
3. If there are still edges repeat from step

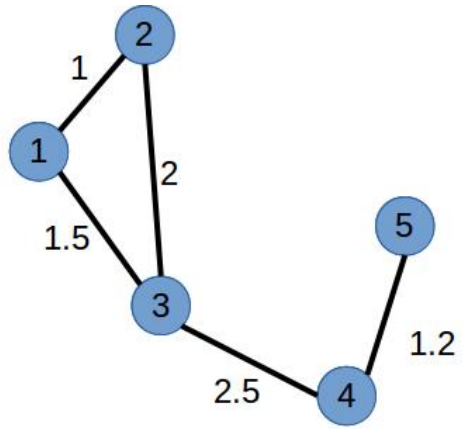
One can also fix in advance the number of communities as a stopping criterion

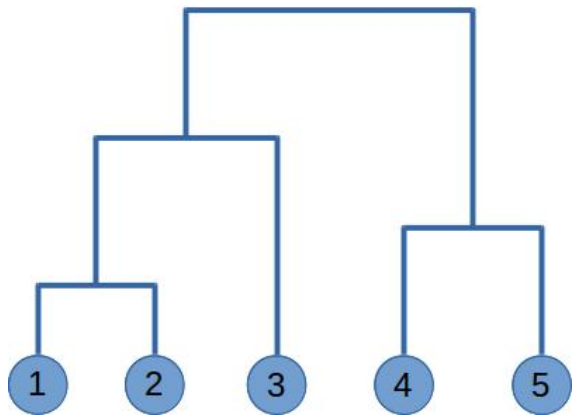


Ward's method: hierarchical clustering

The idea is to start from clusters formed by singletons (single nodes) as the lowest level of the hierarchy and join "nearby" clusters

- Define a distance among sets of nodes
- $h = 0$: each node is a single cluster
- $h > 0$: join the two clusters of level $h - 1$ with minimum distance in a new cluster of level h

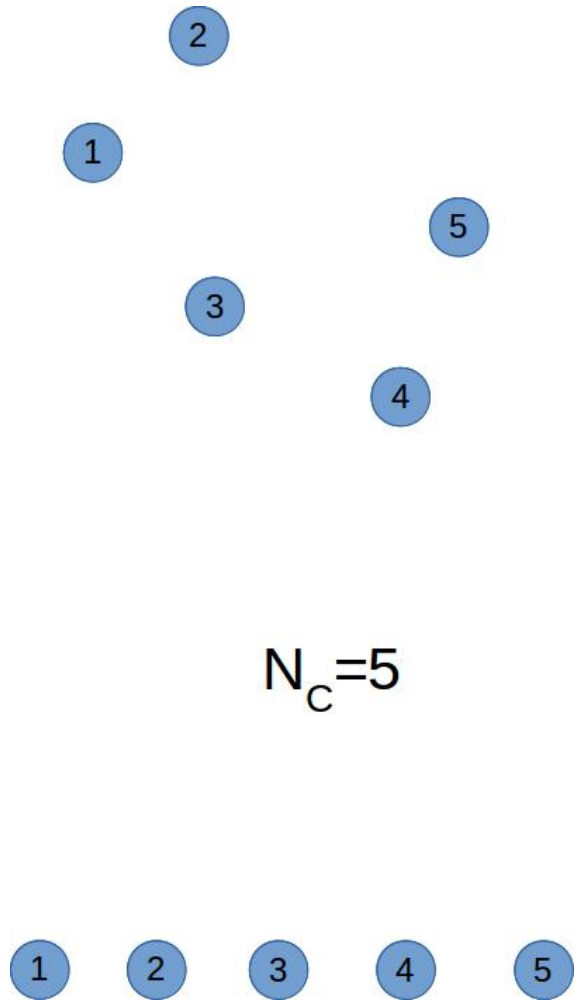




Ward's method: hierarchical clustering

The idea is to start from clusters formed by singletons (single nodes) as the lowest level of the hierarchy and join "nearby" clusters

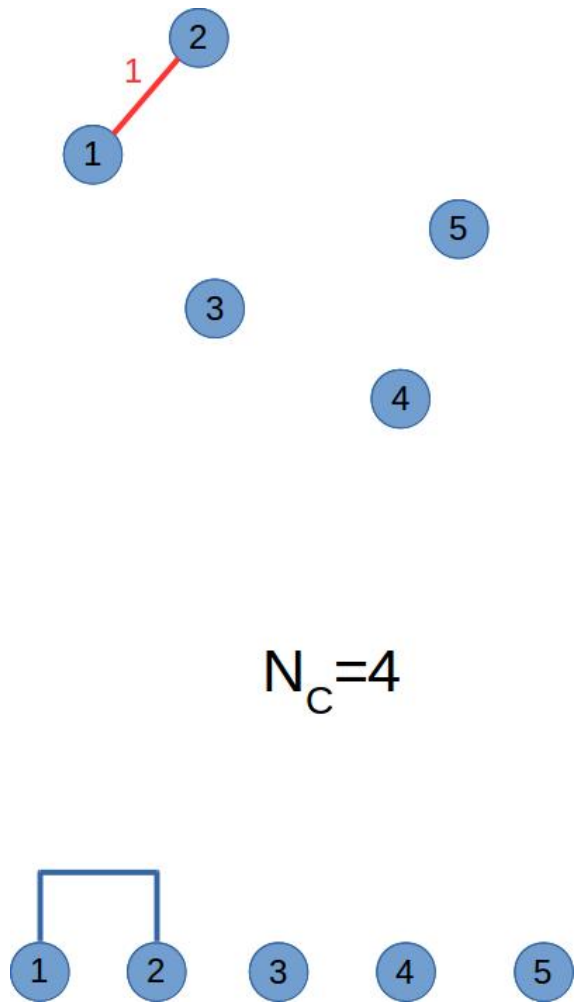
- Define a distance among sets of nodes
- $h = 0$: each node is a single cluster
- $h > 0$: join the two clusters of level $h - 1$ with minimum distance in a new cluster of level h



Ward's method: hierarchical clustering

The idea is to start from clusters formed by singletons (single nodes) as the lowest level of the hierarchy and join "nearby" clusters

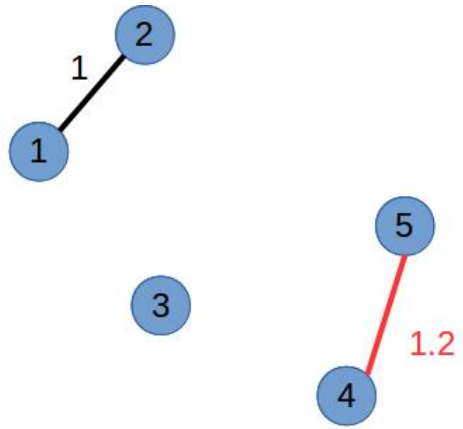
- Define a distance among sets of nodes
- $h = 0$: each node is a single cluster
- $h > 0$: join the two clusters of level $h - 1$ with minimum distance in a new cluster of level h

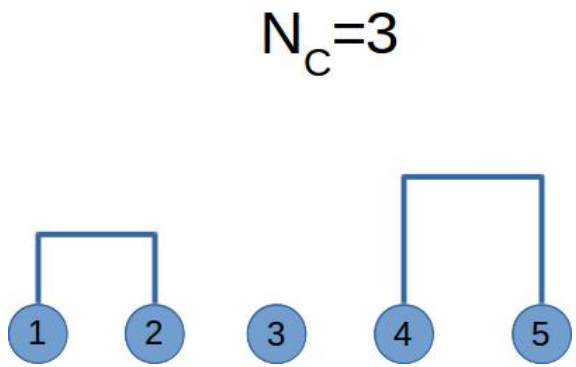


Ward's method: hierarchical clustering

The idea is to start from clusters formed by singletons (single nodes) as the lowest level of the hierarchy and join "nearby" clusters

- Define a distance among sets of nodes
- $h = 0$: each node is a single cluster
- $h > 0$: join the two clusters of level $h - 1$ with minimum distance in a new cluster of level h

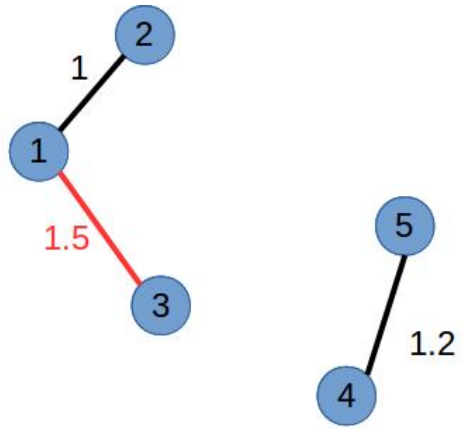


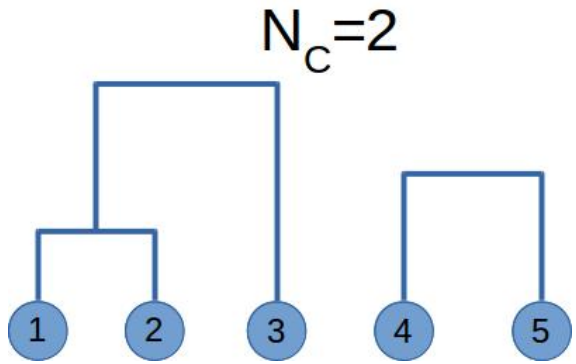


Ward's method: hierarchical clustering

The idea is to start from clusters formed by singletons (single nodes) as the lowest level of the hierarchy and join "nearby" clusters

- Define a distance among sets of nodes
- $h = 0$: each node is a single cluster
- $h > 0$: join the two clusters of level $h - 1$ with minimum distance in a new cluster of level h

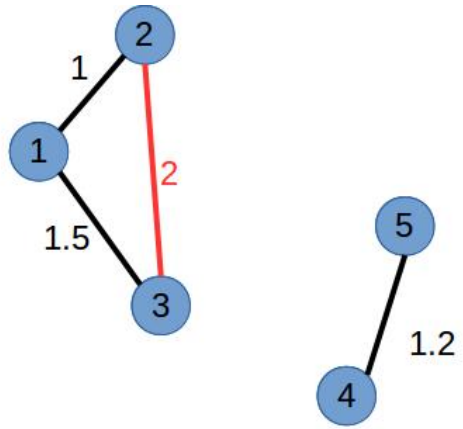


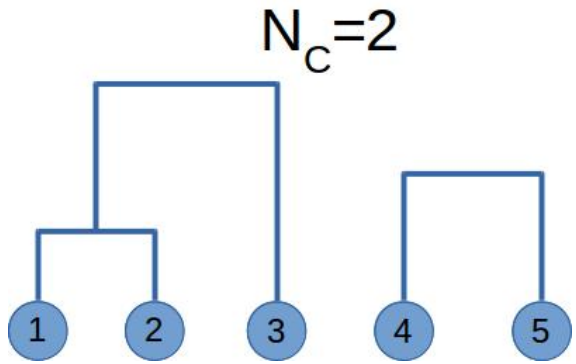


Ward's method: hierarchical clustering

The idea is to start from clusters formed by singletons (single nodes) as the lowest level of the hierarchy and join "nearby" clusters

- Define a distance among sets of nodes
- $h = 0$: each node is a single cluster
- $h > 0$: join the two clusters of level $h - 1$ with minimum distance in a new cluster of level h

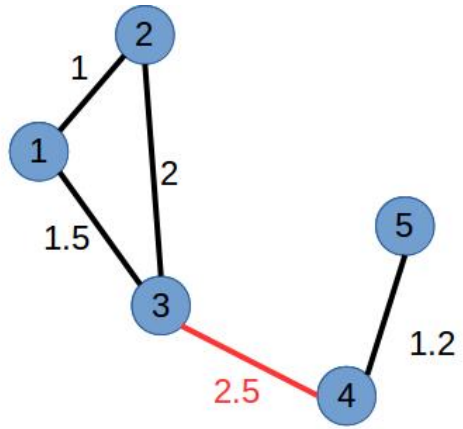


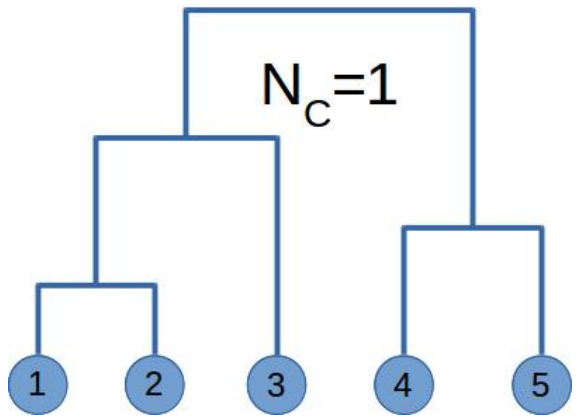


Ward's method: hierarchical clustering

The idea is to start from clusters formed by singletons (single nodes) as the lowest level of the hierarchy and join "nearby" clusters

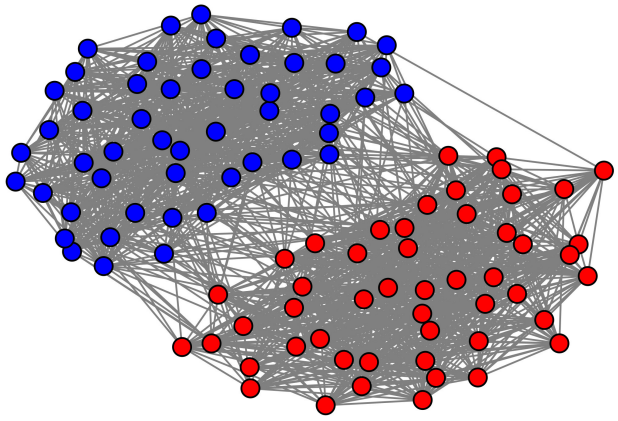
- Define a distance among sets of nodes
- $h = 0$: each node is a single cluster
- $h > 0$: join the two clusters of level $h - 1$ with minimum distance in a new cluster of level h



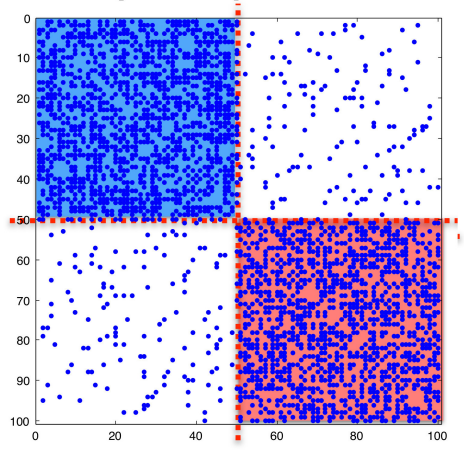


Communities vs Core

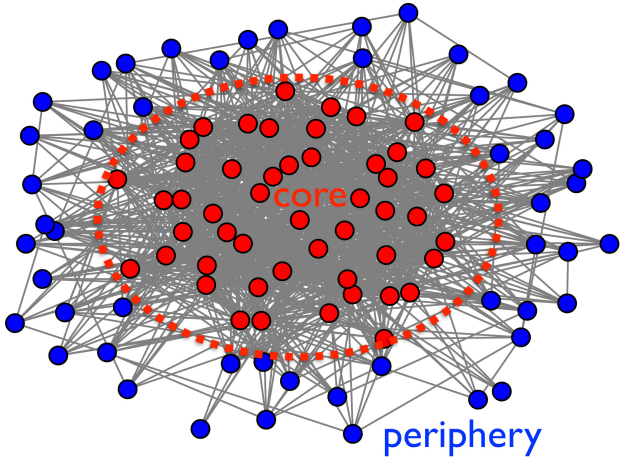
Community structure



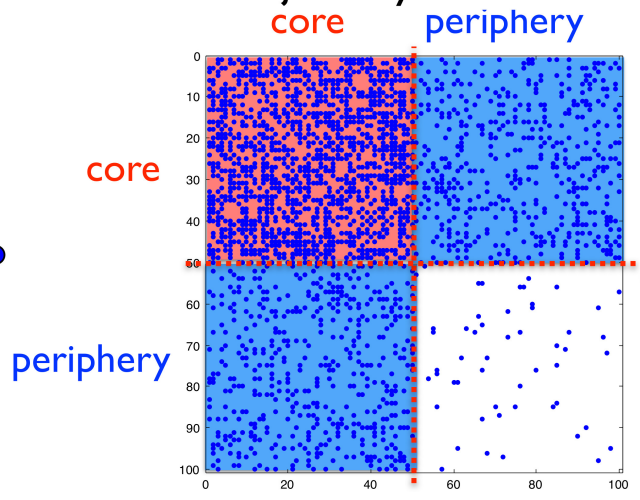
adjacency matrix



Core-periphery structure



adjacency matrix



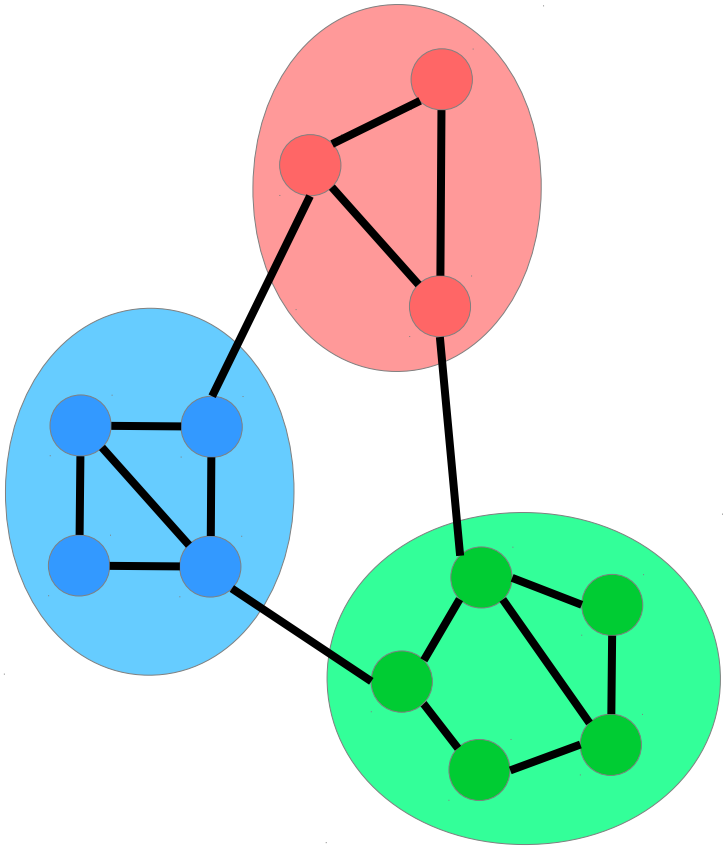
Modularity

- The maximum number q of possible community is fixed
- The variable $s_i \in \{1 \dots q\}$ indicates to which community the node i belongs
- Community are found by minimizing *MODULARITY*

$$Q = \frac{1}{2|E|} \sum_{ij} \left(A_{ij} - \frac{d_i d_j}{2|E|} \right) \delta_{s_i s_j}$$

- A_{ij} is the *observed* number of edges among i, j
- $d_i d_j / 2|E|$ is the *expected* number of edges among i, j if edges are taken at random

Community Structure



Fundamental open questions even for the most basic models of community detection:

- Are there really clusters or communities? Most algorithms will output some community structure; when are these meaningful or artefacts?
- Can we always extract the communities, fully or partially?
- What is a good benchmark to measure the performance of algorithms, and how good are the current algorithms?