Università degli Studi di Padova

Dipartimento di Fisica e Astronomia Galileo Galilei

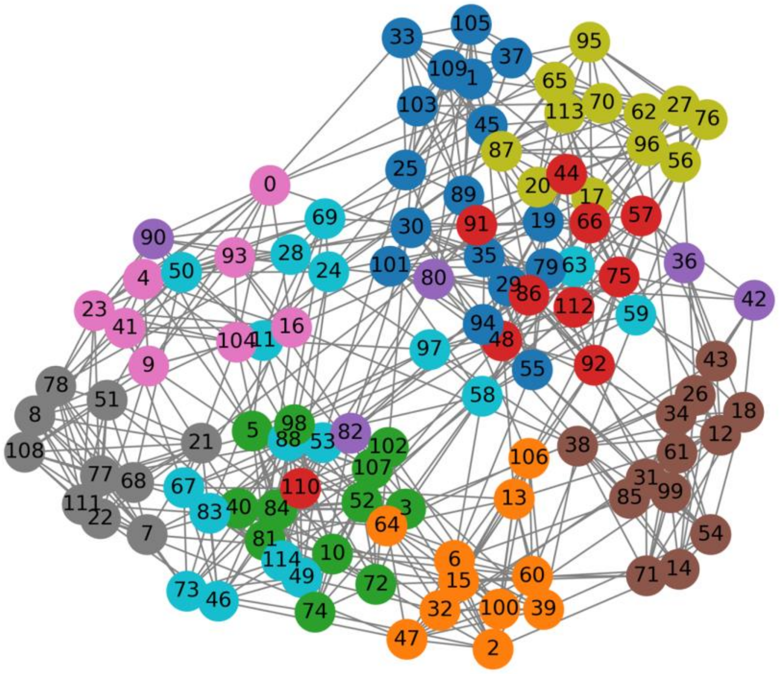
Community structures in complex networks

February 23, 2026

Information theory and Inference

Prof. Michele Allegra

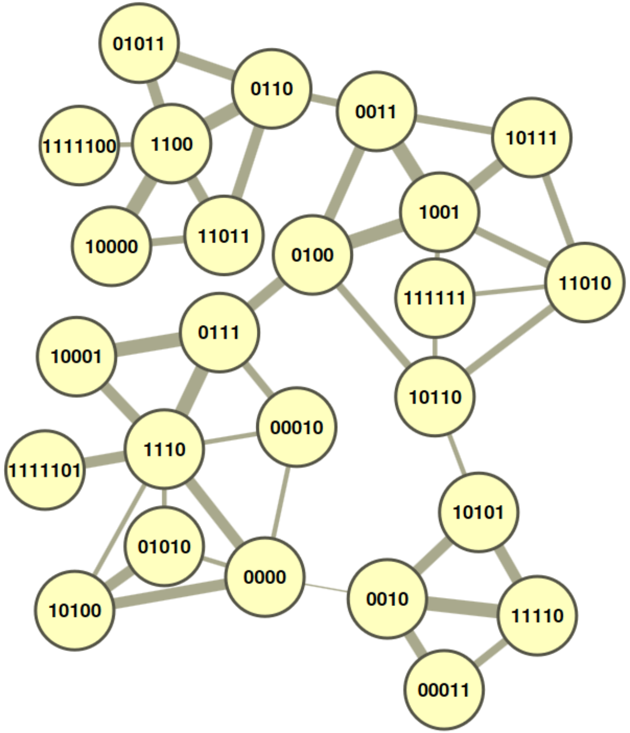
Riccardo Corte
Alessandro Miotto
Lorenzo Rizzi



Goal: decompose the network into modules to reveals community structure

- **Information flow easily between nodes inside a well-connected module**
  - **compression problem for a random walker in the network**

M. Rosvall, C. T. Bergstrom (2007)



$L = 4.50$ bits/step

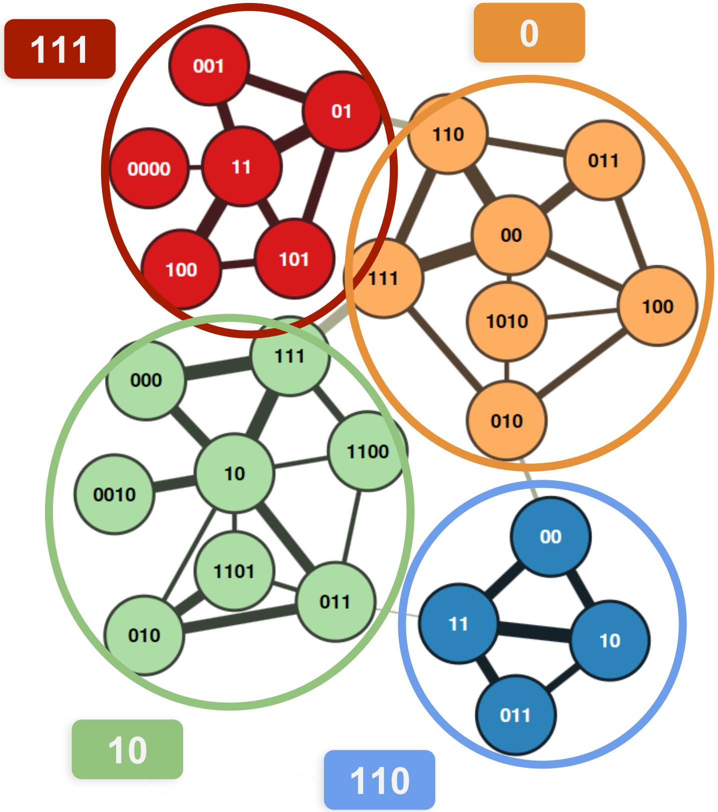
Shannon source code theorem

for symbolic coding

$$H[X] \leq L(E,X) \leq H[X] + 1$$

- ✓ **Huffman code optimally assign prefix-free codewords to nodes**
- ✗ **Does not highlight aspect of the underlying structure**

M. Rosvall, C. T. Bergstrom (2007)



Two-levels description

Modules *or clusters* $H(\mathcal{Q})$ — **unique names** for large-scale objects

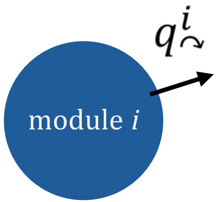
Nodes *within module* $H(\mathcal{P}_i)$ — **reuse names** for *fine-grade* details

Huffman Code $L = 4.50$ bits/step
Two-level description $L = 3.05$ bits/step

M. Rosvall, C. T. Bergstrom (2007)

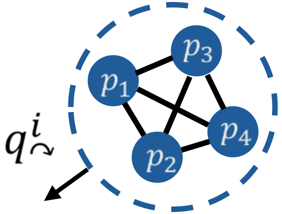
The map equation

$$L(M) = q_{\curvearrowright} H(\mathcal{Q}) + \sum_{i=1}^{m} p_{\circlearrowright}^{i} H(\mathcal{P}^{i}) \longrightarrow M^{*} = \arg\min_{M} L(M)$$



probability the walker switch module

$q_{\curvearrowright} = \sum_{i=1}^{m} q_{\curvearrowright}^{i}$



fraction of within-module movements

$p_{\circlearrowright}^{i} = q_{\curvearrowright}^{i} + \sum_{\alpha \in i} p_{\alpha}$

$$L(M) = \underbrace{q_{\curvearrowright} \log q_{\curvearrowright} - 2\sum_{i=1}^{m} q_{\curvearrowright}^{i} \log q_{\curvearrowright}^{i}}_{\text{movement between modules}} \underbrace{- \sum_{\alpha=1}^{n} p_{\alpha} \log p_{\alpha} + \sum_{\alpha=1}^{n} p_{\circlearrowright}^{i} \log p_{\circlearrowright}^{i}}_{\text{movement within modules}}$$

M. Rosvall, C. T. Bergstrom (2007)

The map equation

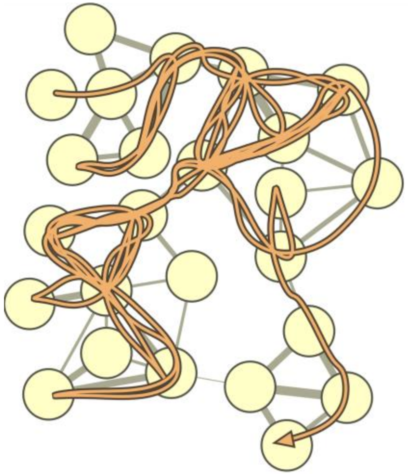
- **Nodes visit frequency** p_α

Random surfer: Markov Chain on the network with teleportation τ

ensure ergodicity (irreducible and aperiodic MC)

$$\mathcal{P} = \frac{\tau}{n}\mathbf{1}\mathbf{1}^T + (1-\tau)A$$

$$\mathcal{P}\pi_{\text{stat}} = \pi_{\text{stat}}$$



➔ **Power method**

$$\lim_{n\to\infty} \mathcal{P}^n \pi_0 = \pi_{\text{stat}}$$

➔ **Eigen-equation**

$$\mathcal{P}\pi_{\text{stat}} = \pi_{\text{stat}}$$

➔ **Pagerank algorithm**

Larry Page, Sergey Brin (1998)

$\tau = 0.15$

- **Module exit probability**

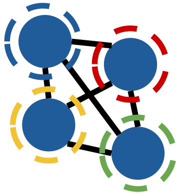
All the possible ways to leave the module *i*

$$q_{i\curvearrowright} = \underbrace{\tau \frac{n-n_i}{n-1} \sum_{\alpha\in i} p_\alpha}_{\text{by teleportation}} + \underbrace{(1-\tau) \sum_{\alpha\in i} \sum_{\beta\notin i} p_\alpha w_{\alpha\beta}}_{\text{by node connections}}$$

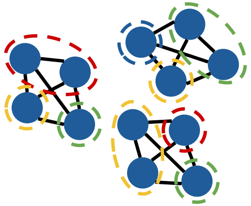
M. Rosvall, C. T. Bergstrom (2007)

Finding the optimal partition

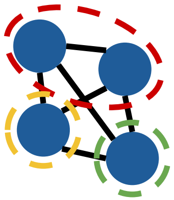
1. Greedy search



Start with each node in its own module



Consider all possible pairs of modules by merging them



Select the merge that yields the largest decrease in $L(M)$

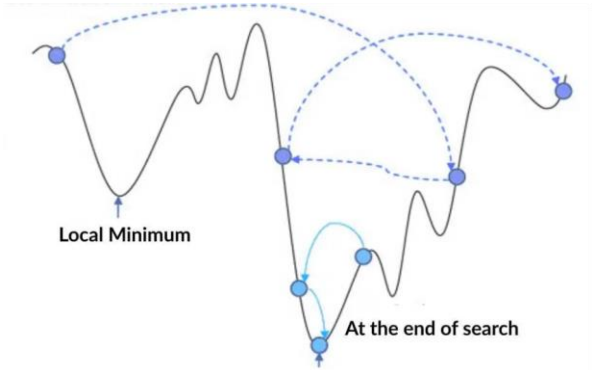
1. Simulated annealing

To escape local minima found by the greedy search

T_{ij} **Proposal move: a node is reassigned to a different module**

$$A_{ij} = \frac{1}{1 + e^{\Delta L_{ij}/T}}$$

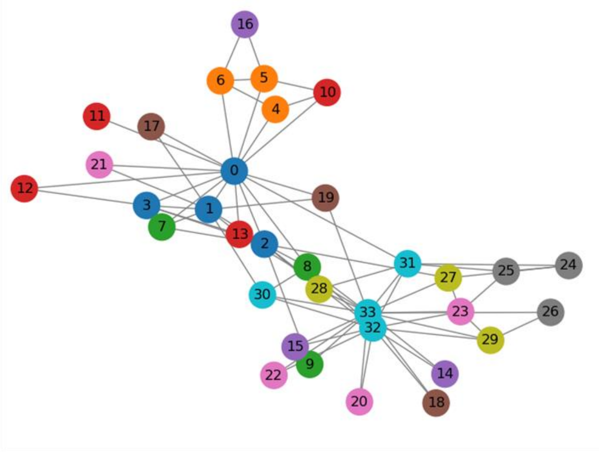
Acceptance rate: heat bath at temperature T



Begin with a high T (explore high L states), then reduce T according to a cooling schedule

M. Rosvall, C. T. Bergstrom (2007)

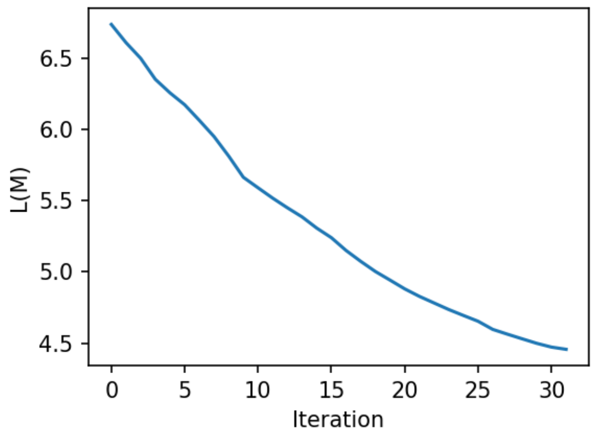
Example: Zachary's karate club

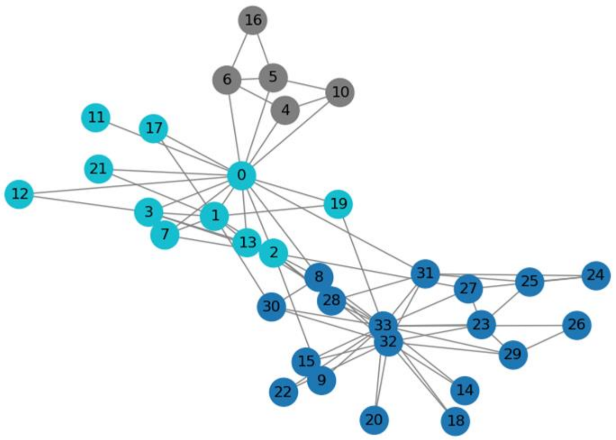


Initial partition

```
L = 6.74 bits/step
m = 36 modules
```

Greedy search
simulated annealing

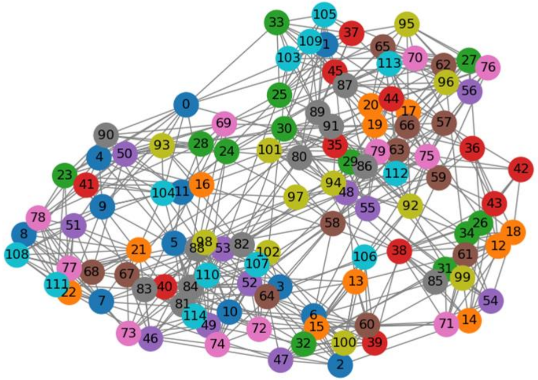


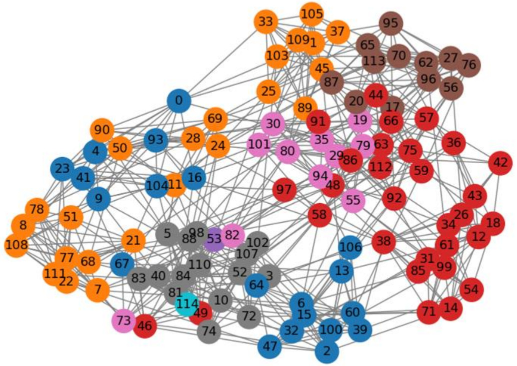


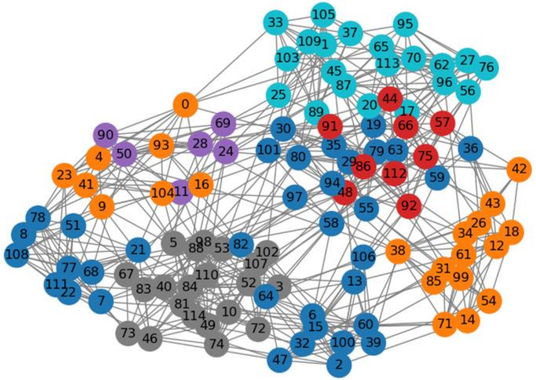
Final partition

```
L = 4.46 bits/step
m = 3 modules
```

Example: football network







Initial partition → Greedy merge → Simulated Annealing

Initial partition

`L = 8.842 bits/step`
`m = 115 modules`

Greedy merge

`L = 6.285 bits/step`
`m = 15 modules`

Simulated Annealing

`L = 5.954 bits/step`
`m = 12 modules`

ground truth

`L = 6.154 bits/step`
`m = 12 modules`

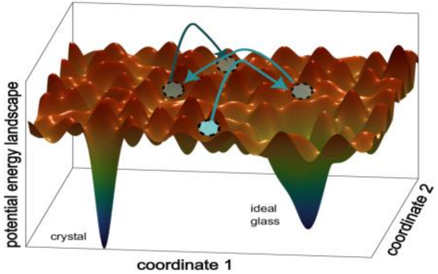
A slightly faster algorithm is the Louvain procedure

This a generic approach to find the partition minimizing/maximizing a generic graph functional H(G)

Initially proposed for Newman Q modularity

$$Q = \frac{1}{2m} \sum_{i,j=1}^{N} (A_{ij} - P_{ij}) \delta_{b_i, b_j}$$

null hypothesis: $P_{ij} = \frac{k_i k_j}{2m}$

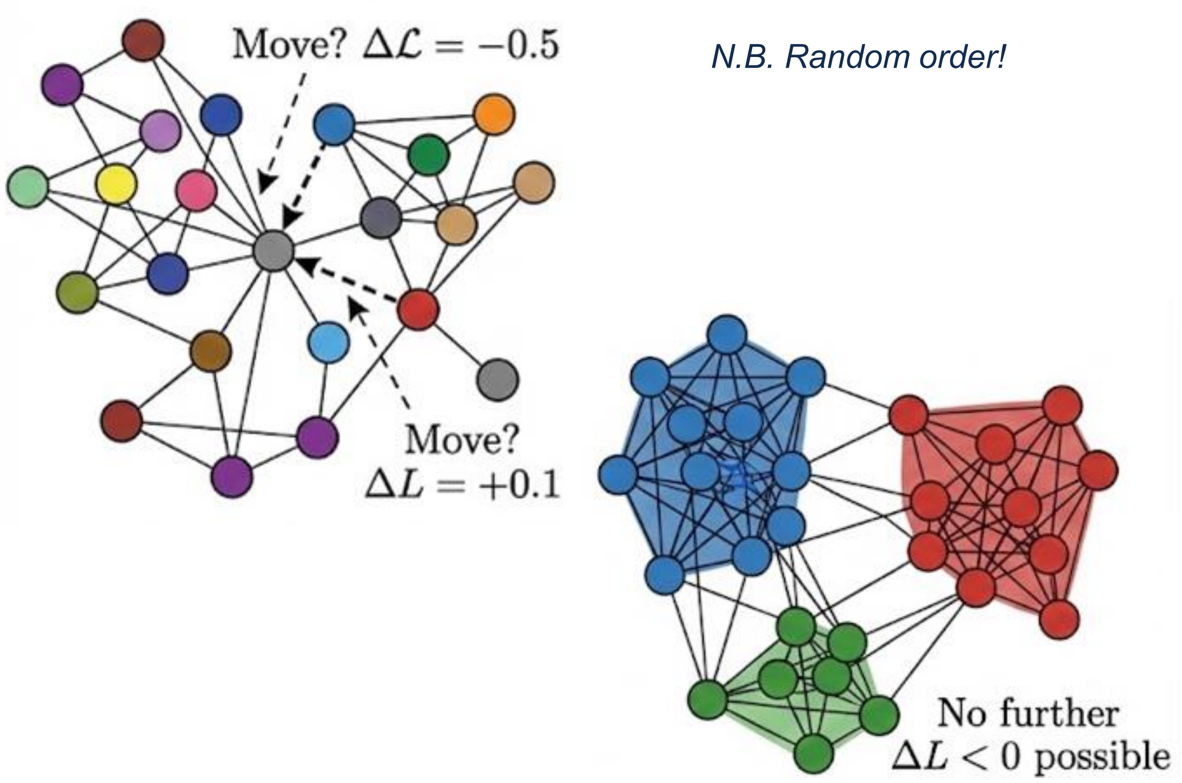


Blondel et al. (2008)

Local Optimization

Greedy phase

1. **Start with each node in its own module**
2. **Try to move a node to its neighbor module**
3. **Select the move that yields the largest decrease in $L(M)$**



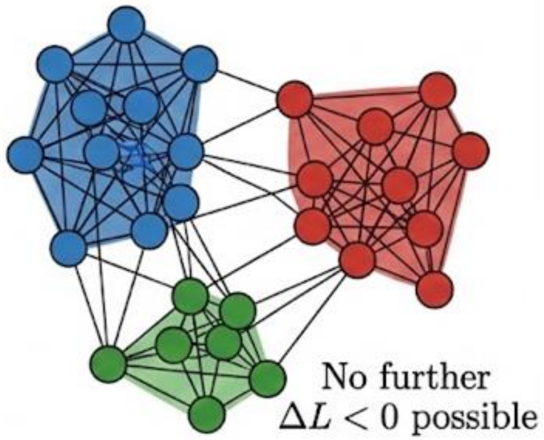
N.B. Random order!

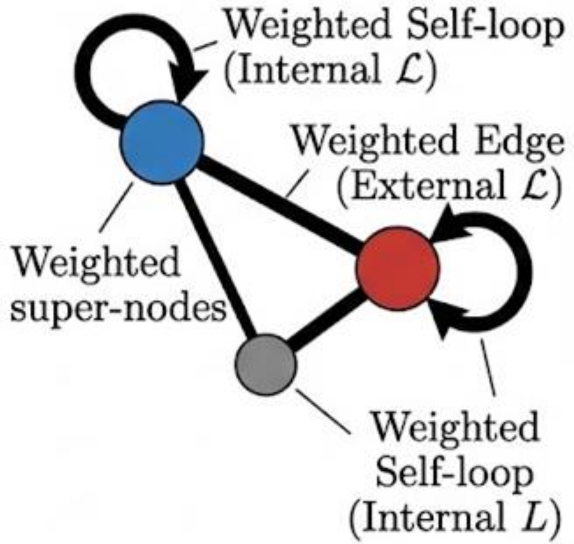
Blondel et al. (2008)

Global Aggregation

Super-Nodes

Previous communities are collapsed into super-nodes

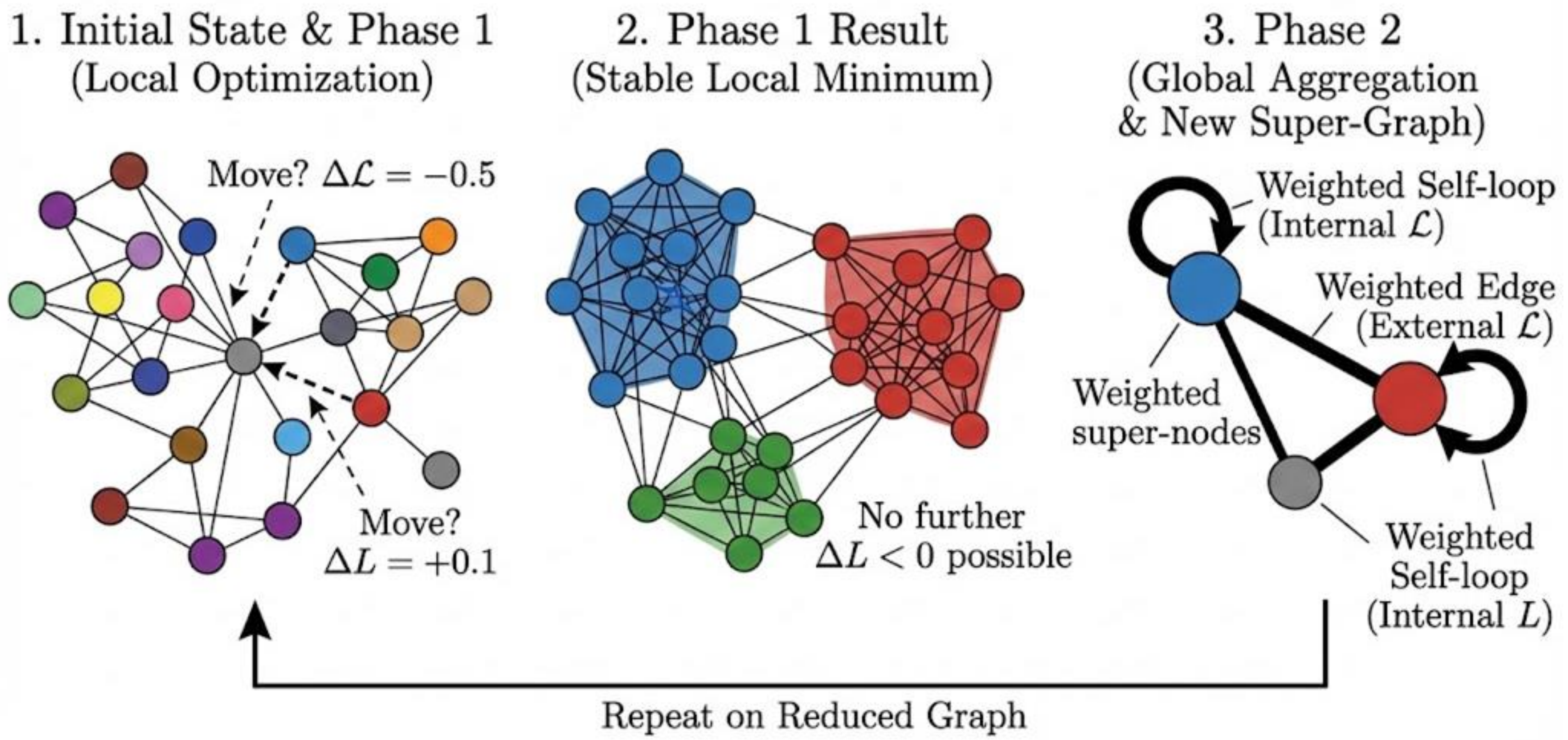




Internal module links	→	**weighted self-loop**
connection between different communities	→	**weighted edges between super-nodes**

Blondel et al. (2008)

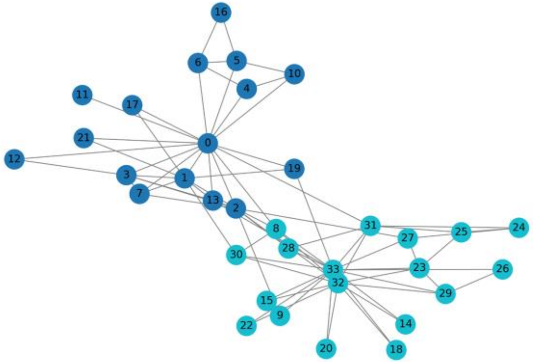
The Louvain Heuristic



Blondel et al. (2008)

Example: Zachary's karate club

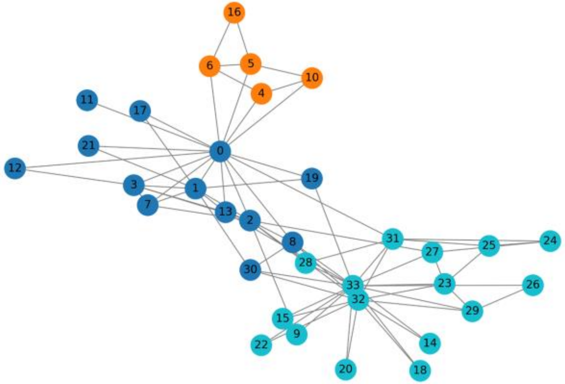
Louvain



```
L = 4.547 bits/step
m = 2 modules
```

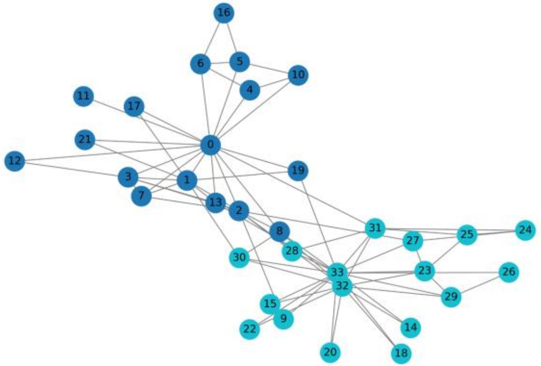
Louvain

Best over 100 runs



```
L = 4.507 bits/step
m = 3 modules
```

Ground truth



```
L = 4.589 bits/step
m = 2 modules
```

Comparing partitions

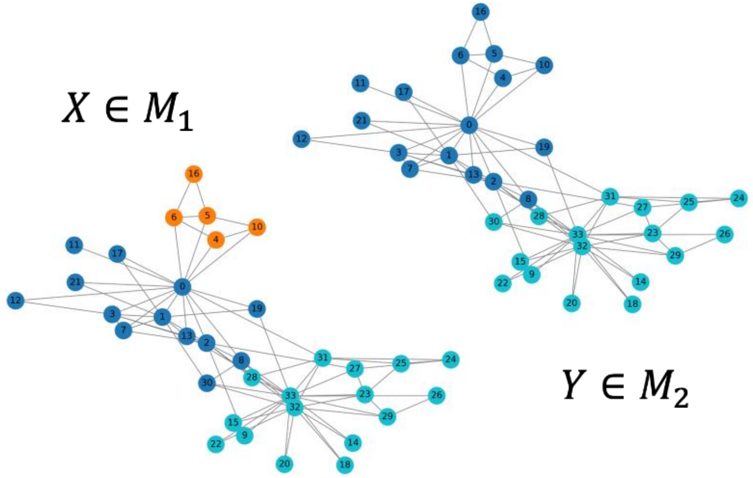
$$\mathrm{NMI}(X,Y) = \frac{2\,I(X:Y)}{H(X) + H(Y)}$$

Normalized mutual information

Metrics to quantify similarities between partitions

- → NMI = *1* **if the partitions are identical** $I(X:Y) = H(X) = H(Y)$
- → NMI = *0* **the partitions are independent** $I(X:Y) = 0$

$X \in M_1$

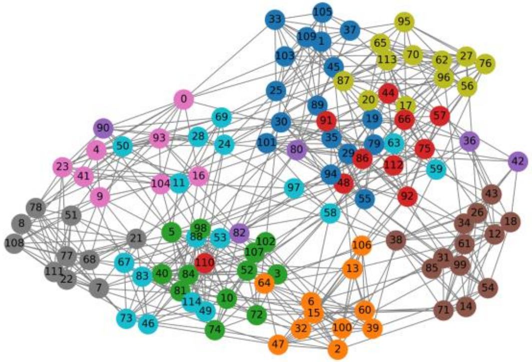


$Y \in M_2$

NMI works by asking how efficiently we can describe one labeling if we know the other. It measures how much less information it takes to communicate the first labeling if we know the second

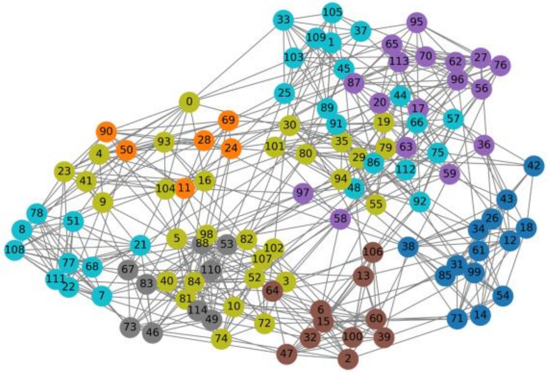
X, Y random variables of which partition a random node is selected

Example: football network



Ground truth

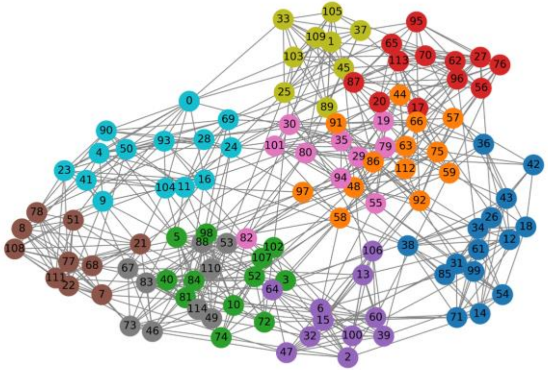
`L = 6.154 bits/step`



Louvain Algorithm

our implementation

`L = 5.954 bits/step`
`NMI = 0.9242`



InfoMap

optimize implementation by mapequation.org

`L = 5.465 bits/step`
`NMI = 0.9114`

Between the two implementations

`NMI = 0.9744`

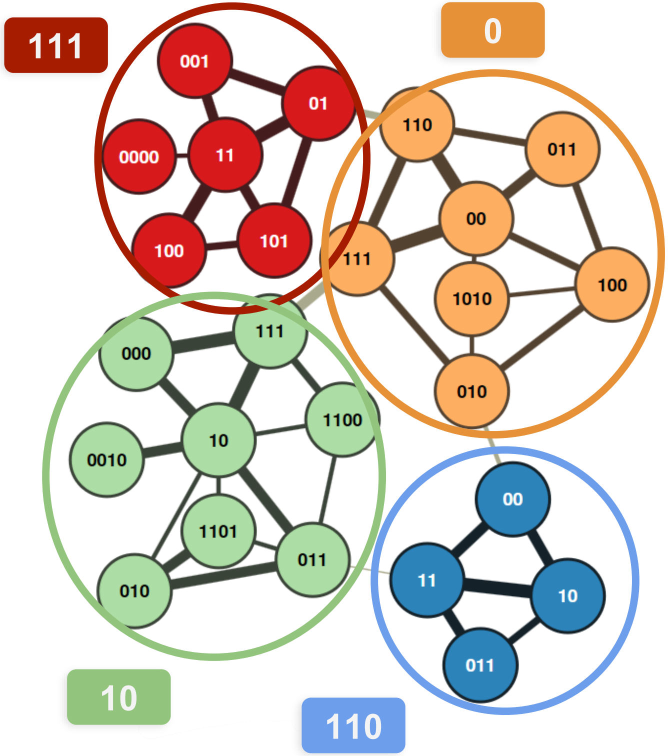
https://www.mapequation.org/

Example: even bigger networks

	Time (s)	NMI	L (bits)	m
Ground truth	-	-	10.145	7
Louvain	465.49	0.4136	7.2390	293
InfoMap	0.06	0.4082	7.2332	277

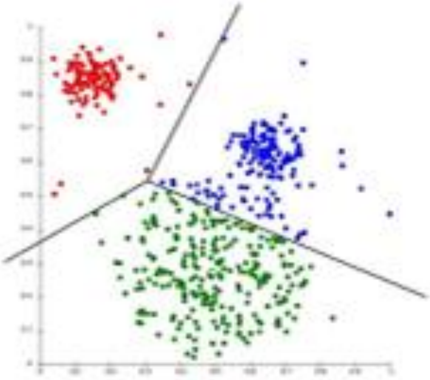
Network with 2708 nodes and 5278 edges

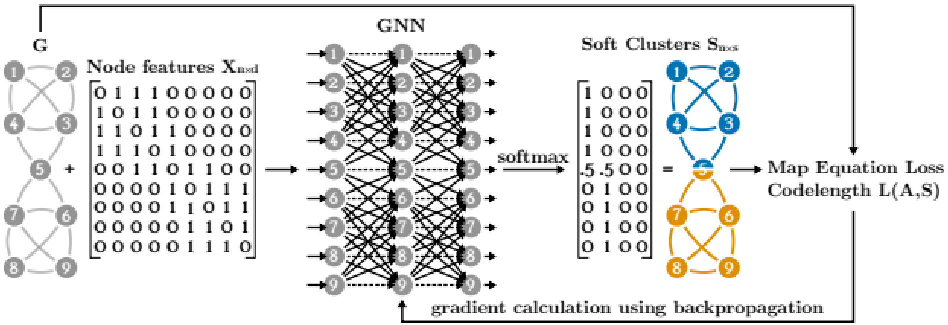
Neuromap



From Infomap

To Machine Learning Methods





Map Equation Formulation

***From an** hard partition **to a** soft cluster matrix representation*

$$L(\mathsf{M}) = qH(Q) + \sum_{\mathsf{m}\in\mathsf{M}} p_{\mathsf{m}} H(P_{\mathsf{m}})$$

$$\downarrow$$

$$L(\mathsf{M}) = \underbrace{q\log_2 q - \sum_{\mathsf{m}\in\mathsf{M}} q_{\mathsf{m}} \log_2 q_{\mathsf{m}}}_{\text{movement between modules}} - \underbrace{\sum_{\mathsf{m}\in\mathsf{M}} \mathsf{m}_{\mathrm{exit}} \log_2 \mathsf{m}_{\mathrm{exit}} - \sum_{u\in V} p_u \log_2 p_u + \sum_{\mathsf{m}\in\mathsf{M}} p_{\mathsf{m}} \log_2 p_{\mathsf{m}}}_{\text{movement within modules}}$$

From the Network to the Map equation

Adjacency Matrix A

Feature Matrix X

Logits definition

$$z = f_\theta(A, X)$$

$$S = \mathrm{softmax}(z/T)$$

q: total probability of **exiting modules** (inter-module flow)
q_m: exiting probability of each module
p_m: total probability mass **inside** module m
m_exit: module exit distribution derived from q_m

Visit rates:
Stationary distribution p
Transition matrix T

$$p^{(t+1)} = \alpha \frac{d_{in}}{w_{tot}} + (1 - \alpha) p^{(t)} T$$

Flow Matrix:

$$F = \alpha \frac{A}{w_{tot}} + (1 - \alpha)\, \mathrm{diag}(p)\, T$$

From nodes flow matrix to Module flow matrix:

C = S.T @ F @ S

q: total probability of **exiting modules** (inter-module flow)
q_m: exiting probability of each module
p_m: total probability mass **inside** module m
m_exit: module exit distribution derived from q_m

$$q = \sum_{m \neq n} C_{mn}$$

$$q_m = \sum_{n \neq m} C_{mn}$$

$$p_m = \sum_{n} C_{mn}$$

From the Network to the Map equation

From nodes flow matrix to Module flow matrix:

C = S.T @ F @ S

↓

q: total probability of **exiting modules** (inter-module flow)
q_m: exiting probability of each module
p_m: total probability mass **inside** module m
m_exit: module exit distribution derived from q_m

$$q = \sum_{m \neq n} C_{mn}$$

$$q_m = \sum_{n \neq m} C_{mn}$$

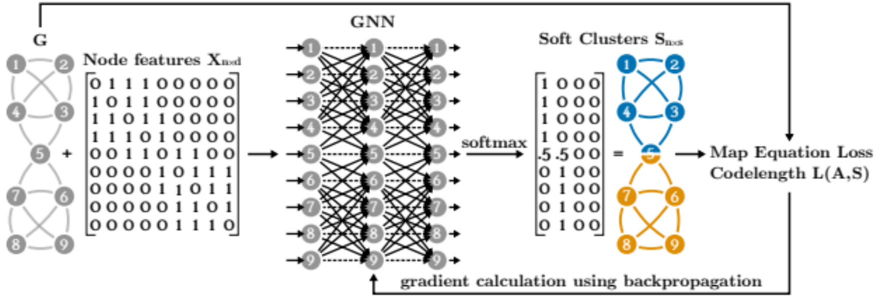
$$p_m = \sum_{n} C_{mn}$$

$$\theta^* = \arg\min_{\theta} L(S(\theta))$$

θ→z→S→L

Models used to parametrize Partitions

Forward(A,x) → **Logits** → **S**



Linear:

$$Z = XW$$

MLP:

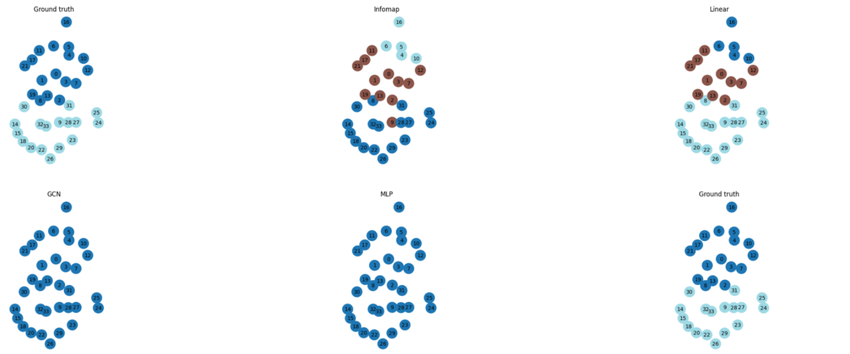
$$Z = \mathrm{MLP}(X)$$

GNN:

$$Z = \mathrm{GNN}(A, X)$$

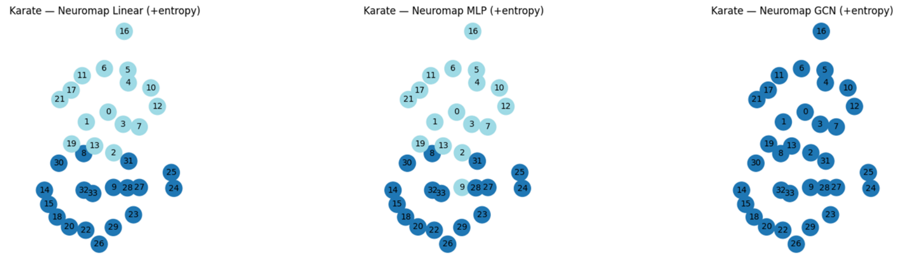
Example: Zachary's karate club

Simple environment Implementation:



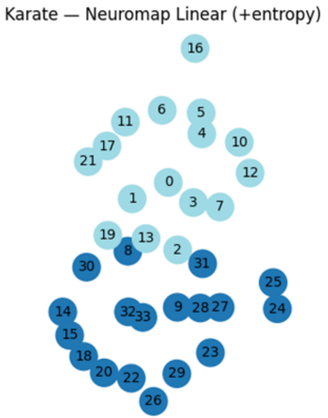
Entropy Implementation:

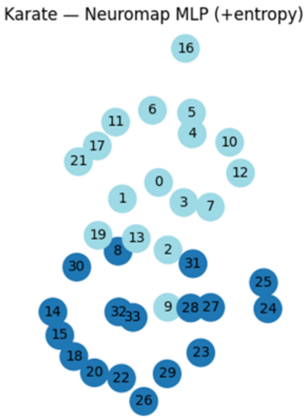
$$L_{train} = L_{Neuromap} - \lambda H(S)$$

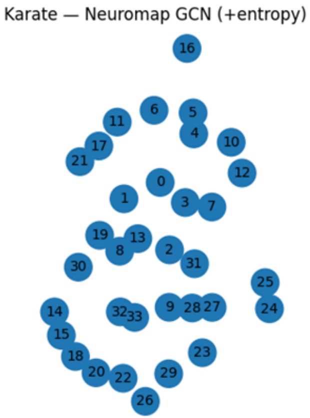


Example: Zachary's karate club

Example: Zachary's karate club

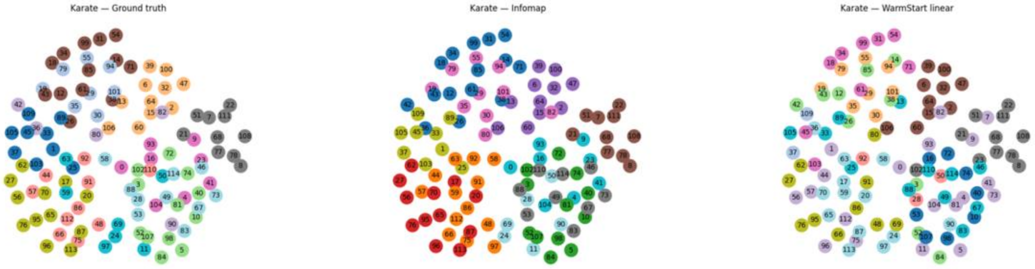






Example: football network

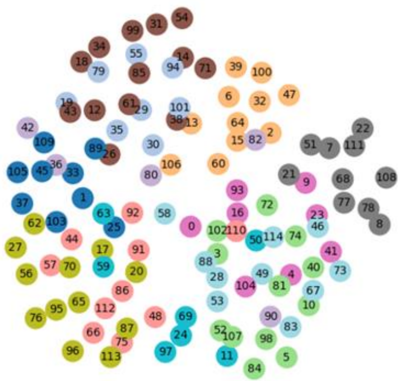
Simple environment Implementation:



Authors Implementation:

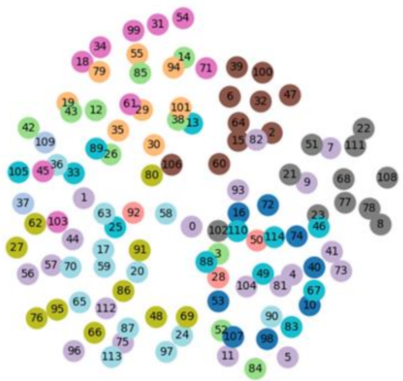
Example: football network

Karate — Ground truth



Karate — Infomap

Karate — WarmStart linear



Example: football network