Network Community Detection

Network Application Diagnostics
B2M32DSA

Radek Mařík

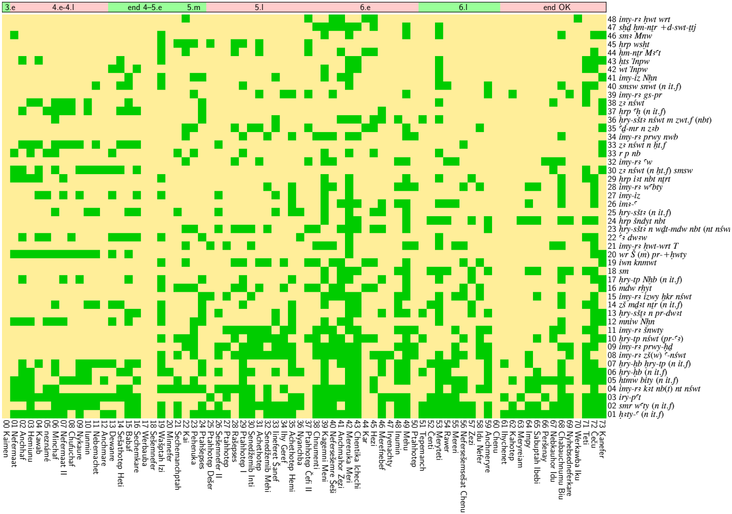
Czech Technical University
Faculty of Electrical Engineering
Department of Telecommunication Engineering
Prague CZ

October 31, 2017

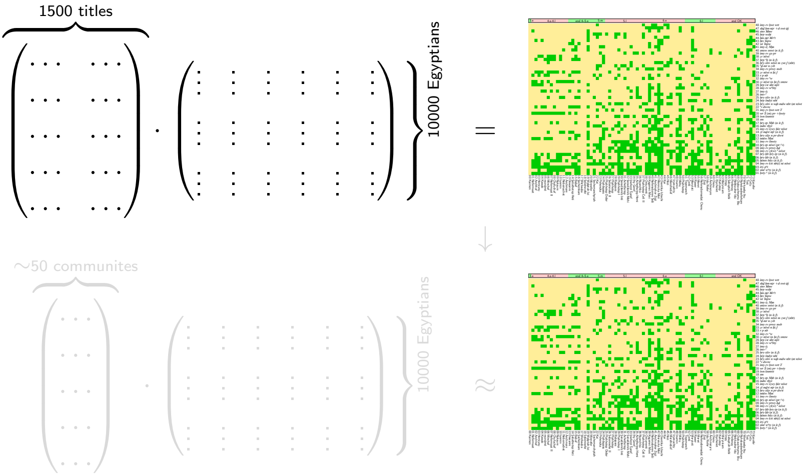
Outline

Outline

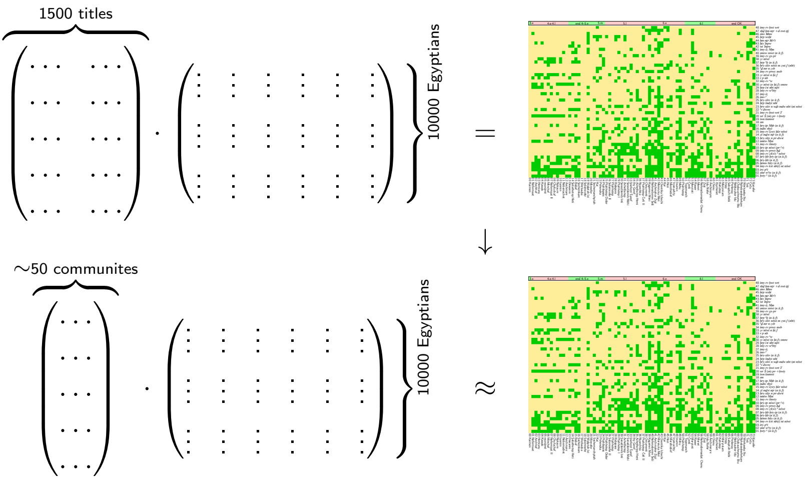
Network of Ancient Egypt Officials [Dul08]



Goal [DM16]

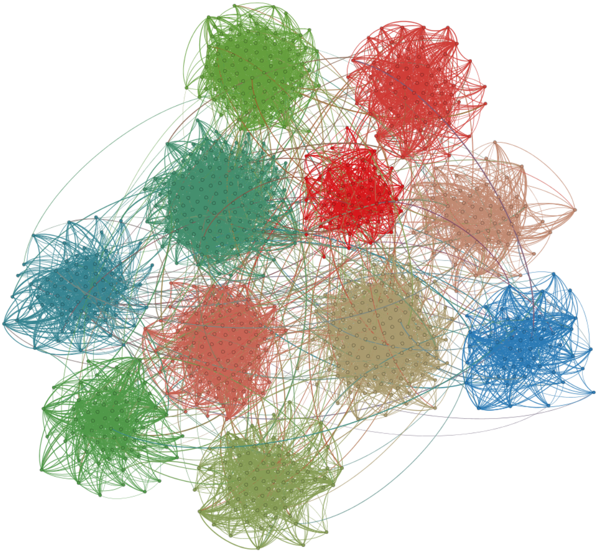

1500 titles
10000 Egyptians
=
~50 communites
10000 Egyptians
≈

Goal [DM16]


1500 titles
10000 Egyptians
~50 communites
10000 Egyptians

Outline

A Network with Communities - Example [BAV13]

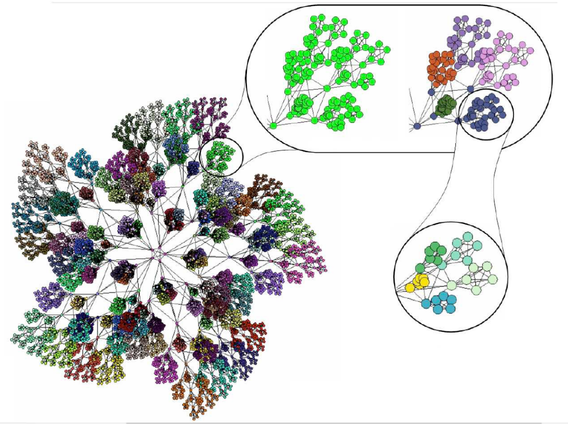


Community Concept [New06, Weh13, FH16]

- To reduce complexity to understand the intermediate structure.
- **Communities**, also called *clusters* or *modules*, are groups of vertices which probably share common properties and/or play similar roles within the graph.
- Communities are dense subgraphs of a network.
  - There must be more edges "inside" the community than edges linking vertices of the community with the rest of the graph.
- Subgroup composition of the network
- Common *local* subgroup definitions:
  - Mutuality (cliques),
  - Reachability (n-cliques),
  - Tie frequency (k-cores),
  - Relative tie frequency (lambda sets, communities)
- *Global* definitions
  - A graph has community structure if it is different from a random graph.
  - A **null model** is a graph which matches the original in some of its structural features, but which is otherwise a random graph.

Outline

Community Structure Extraction [BGLL08]



Overview of Methods

Basic Methods of Data Structure Analysis

- Cluster analysis
- Bi-clustering
- Matrix Factorization
- Community Detection (graphs/networks)

Community Detection

- Nonoverlapping community detection
- Overlapping community detection
- Community detection in bipartite graphs
- Community detection based on stochastic block models

Overview of Methods

Basic Methods of Data Structure Analysis

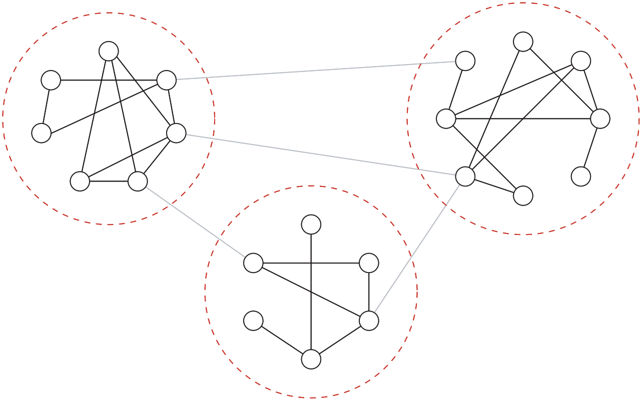
- Cluster analysis
- Bi-clustering
- Matrix Factorization
- Community Detection (graphs/networks)

Community Detection

- Nonoverlapping community detection
- Overlapping community detection
- Community detection in bipartite graphs
- Community detection based on stochastic block models

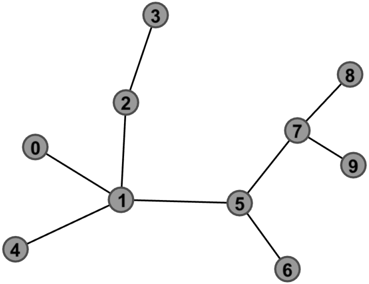
Outline

Nonoverlapping Communities [New04]



- Searching for dense connected subgraphs
  - there are less edges between subgraphs than inside them
- Fundamental approaches
  - Search for partitions
  - Search for hierarchy

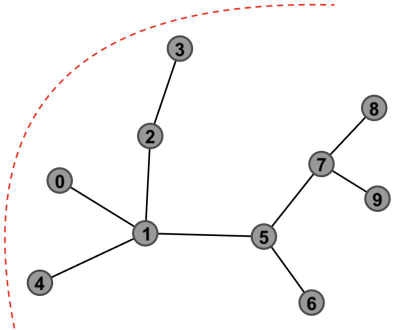
Nonoverlapping Communities - Graph Partitioning



$$e_{\text{inside}} - e_{\text{between}}$$

$$\frac{e_{\text{inside}}}{e_{\text{total}}}$$

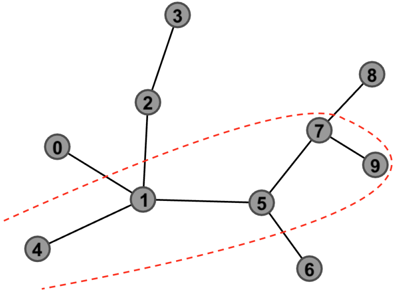
Nonoverlapping Communities - Graph Partitioning



$$e_{\text{inside}} - e_{\text{between}}$$

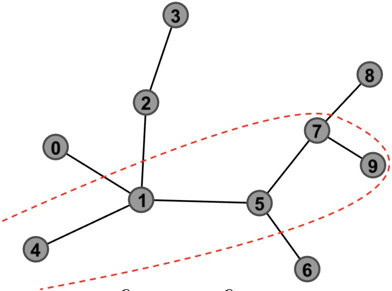
$$\frac{e_{\text{inside}}}{e_{\text{total}}}$$

Kernighan-Lin Algorithm: Goal [KL70]



- The goal to partition a given graph into subgraphs of known orders so that there is the minimum of edges between them.

Kernighan-Lin Algorithm: Node Move Gain [KL70]



$e_{\text{inside}} - e_{\text{between}}$

- Initial partitions: $A = \{0, 2, 3, 6, 8\}, \quad B = \{1, 4, 5, 7, 9\}$
- Node move gain: $D_i = |e(i)_{\text{between}}| - |e(i)_{\text{inside}}|$

vertex	0	1	2	3	4	5	6	7	8	9
D_i	1	0	0	-1	-1	-1	1	-1	1	-1

Kernighan-Lin Algorithm: Node Swap Gain [KL70]

- Partitions: $A = \{0, 2, 3, 6, 8\}, \quad B = \{1, 4, 5, 7, 9\}$
- Node move gain: $D_i = e(i)_{\text{between}} - e(i)_{\text{inside}}]$

vertex	0	1	2	3	4	5	6	7	8	9
D_i	1	0	0	-1	-1	-1	1	-1	1	-1

- 2 neighboring nodes swap gain

$$g_{ij} = (D_i - A_{ij}) + (D_j - A_{ij}) = D_i + D_j - 2A_{ij}, \quad i \in A, j \in B$$

$g_{ij} =$

i \ j	1	4	5	7	9
0	-1	0	0	0	0
2	-2	-1	-1	-1	-1
3	-1	-2	-2	-2	-2
6	1	0	-2	0	0
8	1	0	0	-2	0

Kernighan-Lin Algorithm: Node Swap Gain [KL70]

- Partitions: $A = \{0, 2, 3, 6, 8\}, \quad B = \{1, 4, 5, 7, 9\}$
- Node move gain: $D_i = e(i)_{\text{between}} - e(i)_{\text{inside}}]$

vertex	0	1	2	3	4	5	6	7	8	9
D_i	1	0	0	-1	-1	-1	1	-1	1	-1

- 2 neighboring nodes swap gain

$$g_{ij} = (D_i - A_{ij}) + (D_j - A_{ij}) = D_i + D_j - 2A_{ij}, \quad i \in A, j \in B$$

$g_{ij} =$

i ╲ j	1	4	5	7	9
0	-1	0	0	0	0
2	-2	-1	-1	-1	-1
3	-1	-2	-2	-2	-2
6	1	0	-2	0	0
8	1	0	0	-2	0

Kernighan-Lin Algorithm: Node Swap Gain [KL70]

- Partitions: $A = \{0, 2, 3, 6, 8\}, \quad B = \{1, 4, 5, 7, 9\}$
- Node move gain: $D_i = e(i)_{\text{between}} - e(i)_{\text{inside}}]$

vertex	0	1	2	3	4	5	6	7	8	9
D_i	1	0	0	-1	-1	-1	1	-1	1	-1

- 2 neighboring nodes swap gain

$$g_{ij} = (D_i - A_{ij}) + (D_j - A_{ij}) = D_i + D_j - 2A_{ij}, \quad i \in A, j \in B$$

$g_{ij} =$

i ＼ j	1	4	5	7	9
0	-1	0	0	0	0
2	-2	-1	-1	-1	-1
3	-1	-2	-2	-2	-2
6	1	0	-2	0	0
8	1	0	0	-2	0

If we swap 6 and 1 then we get the maximum gain $+1$.

Kernighan-Lin Algorithm: Update

The tuple 6 and 1 is eliminated in the rest of steps:

$$A = \{0, 2, 3, \not{6}, 8\}, \quad B = \{\not{1}, 4, 5, 7, 9\}$$

and D_i is updated:

$$D_a^{(1)} = D_a^{(0)} + 2A_{a,a_i} - 2A_{a,b_j}, \quad a \in A - \{a_i\}$$

$$D_b^{(1)} = D_b^{(0)} + 2A_{b,b_j} - 2A_{b,a_i}, \quad b \in B - \{b_j\}$$

vertex	0	1	2	3	4	5	6	7	8	9
D_i	-1	0	-2	-1	1	-1	1	-1	1	-1

Possible gains are updated: $g_{ij} =$

i \ j	4	5	7	9
0	0	-2	-2	-2
2	-1	-3	-3	-3
3	0	-2	-2	-2
8	2	0	-2	0

Kernighan-Lin Algorithm: Update

The tuple 6 and 1 is eliminated in the rest of steps:

$$A = \{0, 2, 3, \not{6}, 8\}, \quad B = \{\not{1}, 4, 5, 7, 9\}$$

and D_i is updated:

$$D_a^{(1)} = D_a^{(0)} + 2A_{a,a_i} - 2A_{a,b_j}, \quad a \in A - \{a_i\}$$

$$D_b^{(1)} = D_b^{(0)} + 2A_{b,b_j} - 2A_{b,a_i}, \quad b \in B - \{b_j\}$$

vertex	0	1	2	3	4	5	6	7	8	9
D_i	-1	0	-2	-1	1	-1	1	-1	1	-1

Possible gains are updated:

$g_{ij} =$

i \ j	4	5	7	9
0	0	-2	-2	-2
2	-1	-3	-3	-3
3	0	-2	-2	-2
8	2	0	-2	0

The next maximum gain is 2 if 8 and 4 are swapped.

Kernighan-Lin Algorithm: Following Steps

- Similarly, possible gains are calculated for all remaining pairs.

k	A	B	g_{max}	(a, b)	$\sum_0^k g_{max,i}$
0	$\{0, 2, 3, 6, 8\}$	$\{1, 4, 5, 7, 9\}$	1	(6,1)	1
1	$\{0, 2, 3, \not{6}, 8\}$	$\{\not{1}, 4, 5, 7, 9\}$	2	(8,4)	3
2	$\{0, 2, 3, \not{6}, \not{8}\}$	$\{\not{1}, \not{4}, 5, 7, 9\}$	-2	(0,5)	1
3	$\{\not{0}, 2, 3, \not{6}, \not{8}\}$	$\{\not{1}, \not{4}, \not{5}, 7, 9\}$	-2	(3,7)	-1
4	$\{\not{0}, 2, \not{3}, \not{6}, \not{8}\}$	$\{\not{1}, \not{4}, \not{5}, \not{7}, 9\}$	1	(2,9)	1

- We choose so many steps as reach the maximum total gain $\text{argmax}_k \sum_0^k g_{max,i}$.
- In this case just two steps are performed: we swap $\{6, 1\}$ and $\{8, 4\}$.
- The new partition is obtained $A = \{0, 1, 2, 3, 4\}, \quad B = \{8, 9, 5, 6, 7\}$
- The algorithm ends with the next iteration.

Kernighan-Lin Algorithm: Following Steps

- Similarly, possible gains are calculated for all remaining pairs.

k	A	B	g_{max}	(a,b)	$\sum_0^k g_{max,i}$
0	$\{0,2,3,6,8\}$	$\{1,4,5,7,9\}$	1	(6,1)	1
1	$\{0,2,3,\not{6},8\}$	$\{\not{1},4,5,7,9\}$	2	(8,4)	3
2	$\{0,2,3,\not{6},\not{8}\}$	$\{\not{1},\not{4},5,7,9\}$	-2	(0,5)	1
3	$\{\not{0},2,3,\not{6},\not{8}\}$	$\{\not{1},\not{4},\not{5},7,9\}$	-2	(3,7)	-1
4	$\{\not{0},2,\not{3},\not{6},\not{8}\}$	$\{\not{1},\not{4},\not{5},\not{7},9\}$	1	(2,9)	1

- We choose so many steps as reach the maximum total gain $\text{argmax}_k \sum_0^k g_{max,i}$.
- In this case just two steps are performed: we swap $\{6,1\}$ and $\{8,4\}$.
- The new partition is obtained $A = \{0,1,2,3,4\}, \quad B = \{8,9,5,6,7\}$
- The algorithm ends with the next iteration.

Kernighan-Lin Algorithm: Following Steps

- Similarly, possible gains are calculated for all remaining pairs.

k	A	B	g_{max}	(a,b)	$\sum_0^k g_{max,i}$
0	$\{0,2,3,6,8\}$	$\{1,4,5,7,9\}$	1	(6,1)	1
1	$\{0,2,3,\not6,8\}$	$\{\not1,4,5,7,9\}$	2	(8,4)	3
2	$\{0,2,3,\not6,\not8\}$	$\{\not1,\not4,5,7,9\}$	-2	(0,5)	1
3	$\{\not0,2,3,\not6,\not8\}$	$\{\not1,\not4,\not5,7,9\}$	-2	(3,7)	-1
4	$\{\not0,2,\not3,\not6,\not8\}$	$\{\not1,\not4,\not5,\not7,9\}$	1	(2,9)	1

- We choose so many steps as reach the maximum total gain $\text{argmax}_k \sum_0^k g_{max,i}$.
- In this case just two steps are performed: we swap $\{6,1\}$ and $\{8,4\}$.
- The new partition is obtained $A = \{0,1,2,3,4\}, \quad B = \{8,9,5,6,7\}$
- The algorithm ends with the next iteration.

Kernighan-Lin Algorithm: Following Steps

- Similarly, possible gains are calculated for all remaining pairs.

k	A	B	g_{max}	(a,b)	$\sum_0^k g_{max,i}$
0	$\{0,2,3,6,8\}$	$\{1,4,5,7,9\}$	1	(6,1)	1
1	$\{0,2,3,\cancel{6},8\}$	$\{\cancel{1},4,5,7,9\}$	2	(8,4)	3
2	$\{0,2,3,\cancel{6},\cancel{8}\}$	$\{\cancel{1},\cancel{4},5,7,9\}$	-2	(0,5)	1
3	$\{\cancel{0},2,3,\cancel{6},\cancel{8}\}$	$\{\cancel{1},\cancel{4},\cancel{5},7,9\}$	-2	(3,7)	-1
4	$\{\cancel{0},2,\cancel{3},\cancel{6},\cancel{8}\}$	$\{\cancel{1},\cancel{4},\cancel{5},\cancel{7},9\}$	1	(2,9)	1

- We choose so many steps as reach the maximum total gain $\text{argmax}_k \sum_0^k g_{max,i}$.
- In this case just two steps are performed: we swap $\{6,1\}$ and $\{8,4\}$.
- The new partition is obtained $A = \{0,1,2,3,4\}, \quad B = \{8,9,5,6,7\}$
- The algorithm ends with the next iteration.

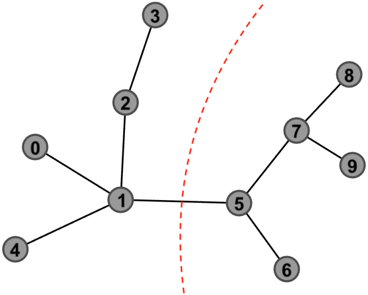
Kernighan-Lin Algorithm: Following Steps

- Similarly, possible gains are calculated for all remaining pairs.

k	A	B	g_{max}	(a,b)	$\sum_0^k g_{max,i}$
0	$\{0,2,3,6,8\}$	$\{1,4,5,7,9\}$	1	(6,1)	1
1	$\{0,2,3,\not 6,8\}$	$\{\not 1,4,5,7,9\}$	2	(8,4)	3
2	$\{0,2,3,\not 6,\not 8\}$	$\{\not 1,\not 4,5,7,9\}$	-2	(0,5)	1
3	$\{\not 0,2,3,\not 6,\not 8\}$	$\{\not 1,\not 4,\not 5,7,9\}$	-2	(3,7)	-1
4	$\{\not 0,2,\not 3,\not 6,\not 8\}$	$\{\not 1,\not 4,\not 5,\not 7,9\}$	1	(2,9)	1

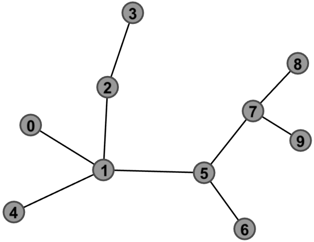
- We choose so many steps as reach the maximum total gain $\text{argmax}_k \sum_0^k g_{max,i}$.
- In this case just two steps are performed: we swap $\{6,1\}$ and $\{8,4\}$.
- The new partition is obtained $A = \{0,1,2,3,4\}, \quad B = \{8,9,5,6,7\}$
- The algorithm ends with the next iteration.

Kernighan-Lin Algorithm: The Result



- The new partition $A = \{0, 1, 2, 3, 4\}, \quad B = \{8, 9, 5, 6, 7\}$
- **Drawbacks:**
  - The number of partitions must be given in advance.
  - The size of partitions must be given in advance.

Spectral Bisection: Input Data [New10]



$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

- Spectral partitioning method of Fiedler
- It makes use of the matrix properties of the graph Laplacian
- The graph bisection . . . the problem of divident a graph into two parts of specified sizes N_1 and N_2.
- N vertices, M edges
- The cut size for the division
  - i.e. the number of edges running between the two groups

$$R = \frac{1}{2} \sum_{\substack{i,j \text{ in} \\ \text{different} \\ \text{groups}}} A_{ij}$$

Spectral Bisection: Graph Laplacian

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

$$\mathbf{L} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 3 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

Spectral Bisection [New10]

- A division vector s as a set of quantities s_i for each vertex i.

$$s_i = \begin{cases} +1 & \text{if vertex } i \text{ belongs to group 1,} \\ -1 & \text{if vertex } i \text{ belongs to group 2} \end{cases}$$

- Then

$$\frac{1}{2}(1 - s_i s_j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ belong to different groups} \\ 0 & \text{if } i \text{ and } j \text{ belong to the same group} \end{cases}$$

- Since $\sum_{ij} A_{ij} = \sum_i k_i = \sum_i k_i s_i^2 = \sum_{ij} k_i \delta_{ij} s_i s_j$
- we can find that (considering graph Laplacian L)

$$R = \frac{1}{4} \sum_{ij} A_{ij}(1 - s_i s_j) = \frac{1}{4} \sum_{ij} (A_{ij} - A_{ij} s_i s_j) \quad (1)$$

$$= \frac{1}{4} \sum_{ij} (k_i \delta_{ij} s_i s_j - A_{ij} s_i s_j) = \frac{1}{4} \sum_{ij} (k_i \delta_{ij} - A_{ij}) s_i s_j \quad (2)$$

$$= \frac{1}{4} \sum_{ij} L_{ij} s_i s_j = \frac{1}{4} \mathbf{s}^T \mathbf{L} \mathbf{s} \quad (3)$$

Spectral Bisection - Minimization Problem [New10]

- The goal is to find the vector $\mathbf{s}$ that minimizes the cut size R for given $\mathbf{L}$.
- Using the *relaxation method* ... an approximate solution of vector optimization problem.
  - Two constraints $\sum_i s_i^2 = N$ and $\sum_i s_i = N_1 - N_2$
- The solution

$$\mathbf{L}\mathbf{s} = \lambda\mathbf{s} + \mu\mathbf{1} \qquad \ldots \mathbf{1}^T \times$$

- Since $\mathbf{L} \cdot \mathbf{1} = 0 = \mathbf{1}^T \cdot \mathbf{L}$, it is $\mu = -\frac{N_1 - N_2}{N}\lambda$
- We define a new vector $\mathbf{x} = \mathbf{s} + \frac{\mu}{\lambda}\mathbf{1} = \mathbf{s} - \frac{N_1 - N_2}{N}\mathbf{1}$
- Then $\mathbf{x}$ is the eigenvector of $\mathbf{L}$ with eigenvalue λ

$$\mathbf{L}\mathbf{x} = \mathbf{L}(\mathbf{s} + \tfrac{\mu}{\lambda}\mathbf{1}) = \mathbf{L}\mathbf{s} = \lambda\mathbf{s} + \mu\mathbf{1} = \lambda\mathbf{x}$$

- NOT $\mathbf{1}$:

$$\mathbf{1}^T\mathbf{x} = \mathbf{1}^T\mathbf{s} - \tfrac{\mu}{\lambda}\mathbf{1}^T\mathbf{1} = (N_1 - N_2) - \frac{N_1 - N_2}{N}N = 0$$

Spectral Bisection - Eigenvector Choice [New10]

- Since

$$\mathbf{x}^T\mathbf{x} = (\mathbf{s} + \tfrac{\mu}{\lambda}\mathbf{1})^T(\mathbf{s} + \tfrac{\mu}{\lambda}\mathbf{1}) = \mathbf{s}^T\mathbf{s} + \tfrac{\mu}{\lambda}(\mathbf{s}^T\mathbf{1} + \mathbf{1}^T\mathbf{s}) + \tfrac{\mu^2}{\lambda^2}\mathbf{1}^T\mathbf{1} \quad (4)$$

$$= N - 2\frac{N_1 - N_2}{N}(N_1 - N_2) + \frac{(N_1 - N_2)^2}{N^2}N = 4\frac{N_1 N_2}{N} \quad (5)$$

- Searching for the smallest value of the cut size R

$$R = \tfrac{1}{4}\mathbf{s}^T\mathbf{L}\mathbf{s} = \tfrac{1}{4}\mathbf{x}^T\mathbf{L}\mathbf{x} = \tfrac{1}{4}\lambda\mathbf{x}^T\mathbf{x} = \frac{N_1 N_2}{N}\lambda$$

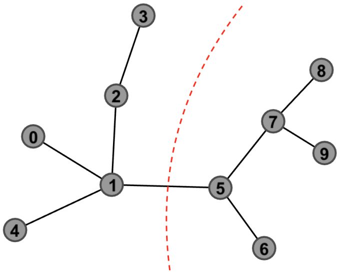
- $\Longrightarrow$ we search for the second smallest eigenvalue λ_2
  - λ_2 ...the Fiedler value, the corresponding eigenvector, the Fiedler vector [Fie73, Fie75]
  - $\lambda_1 = 0$ puts all vertices into one group.
- The most positive values $s_i = x_i + (N_1 - N_2)/N$ are also the most positive values of x_i.
- **Compute eigenvector v_2 and assign N_1 vertices according to the N_1 most/least positive elements of v_2 into group 1.**

Spectral Bisection

Eigenvectors:

$$\begin{pmatrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\end{pmatrix}, \begin{pmatrix}0.2393\\0.1911\\0.3500\\0.4384\\0.2393\\-0.1027\\-0.1287\\-0.3500\\-0.4384\\-0.4384\end{pmatrix} \Longrightarrow$$

$A = \{0, 1, 2, 3, 4\}$
$B = \{5, 6, 7, 8, 9\}$



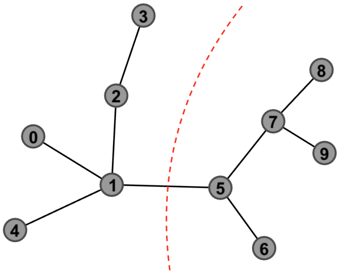
Eigenvalues: $\lambda_1 = 0, \quad \lambda_2 = 0.2015$

Spectral Bisection

Eigenvectors:

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0.2393 \\ 0.1911 \\ 0.3500 \\ 0.4384 \\ 0.2393 \\ -0.1027 \\ -0.1287 \\ -0.3500 \\ -0.4384 \\ -0.4384 \end{pmatrix} \Longrightarrow$$

$$A = \{0, 1, 2, 3, 4\}$$
$$B = \{5, 6, 7, 8, 9\}$$



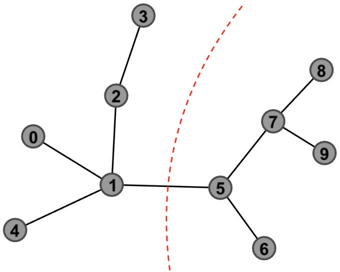
Eigenvalues: $\lambda_1 = 0, \quad \lambda_2 = 0.2015$

Spectral Bisection

Eigenvectors:

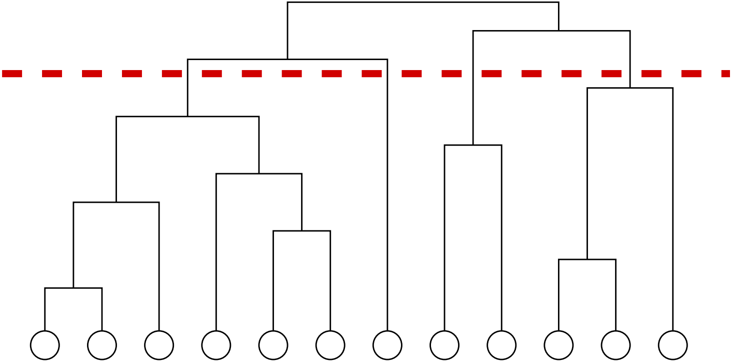
$$\begin{pmatrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\end{pmatrix}, \begin{pmatrix}0.2393\\0.1911\\0.3500\\0.4384\\0.2393\\-0.1027\\-0.1287\\-0.3500\\-0.4384\\-0.4384\end{pmatrix} \Longrightarrow$$

$$A = \{0, 1, 2, 3, 4\}$$
$$B = \{5, 6, 7, 8, 9\}$$



Eigenvalues: $\lambda_1 = 0, \quad \lambda_2 = 0.2015$

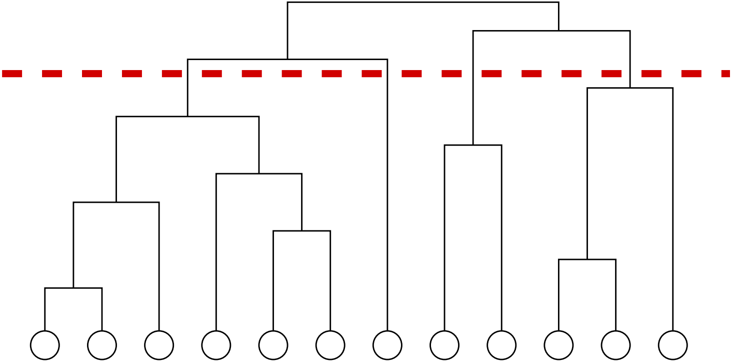
Hierarchical clustering [New04]



Modularity

$$Q = \frac{1}{2M} \sum_{i,j} \left(\mathbf{A}_{ij} - P_{ij} \right) \delta_{C_i C_j}$$

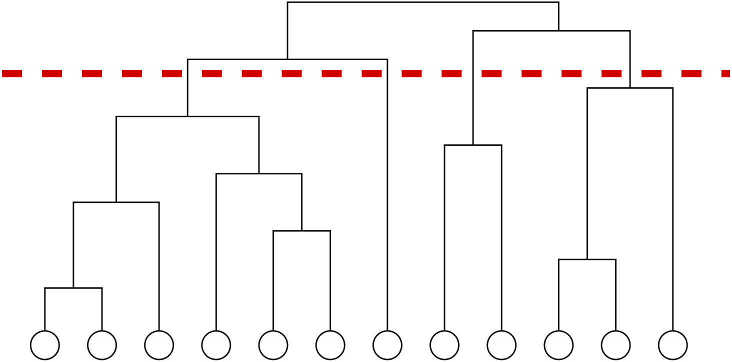
Hierarchical clustering [New04]



Modularity

$$Q = \frac{1}{2M} \sum_{i,j} \left(\mathbf{A}_{ij} - P_{ij} \right) \delta_{C_i C_j}$$

Hierarchical clustering [New04]



Modularity

$$Q = \frac{1}{2M} \sum_{i,j} \left(\mathbf{A}_{ij} - \frac{k_i k_j}{2M} \right) \delta_{C_i C_j}$$

Newman's Modularity [New06, Weh13]

Modularity: function which measures the quality of a partition

- **Communities** are dense subgraphs of a network.
- Reduce complexity to understand the intermediate structure.
- Subgroup composition of the network
- Common subgroup definitions:
  - Mutuality (cliques),
  - Reachability (n-cliques),
  - tie frequency (k-cores),
  - relative tie frequency (lambda sets, communities)
- *"A good division of a network into communities is not merely one in which there are few edges between communities; it is one in which there are fewer than expected edges between communities".*
- **Modularity** . . . is - up to a normalization constant - the number of edges within communities c minus those for **a null model**:

Modularity [New06]

$$Q = \frac{1}{2M} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2M} \right) \delta_{C_i C_j},$$

where

A_{ij}	. . . a weight of the edge between vertices i and j
$k_i = \sum_j A_{ij}$	. . . a (weighted) vertex degree i
$M = \frac{1}{2} \sum_{i,j} A_{ij}$	. . . the total edge weight (the total number of edges)
$k_i k_j / 2M$	. . . the expected weight (number) of edges between i and j
	. . . **null model**
C_i	. . . the attribute (community)of the vertex i
δ_{uv}	. . . Kronecker delta

- $Q \in [-1, 1]$ is normalized
- for edges with weights

Newman Spectral Method - Modularity matrix [New06]

$$Q = \frac{1}{2M} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2M} \right) \delta_{C_i C_j},$$

Definice 2.1 (Modularity matrix)

$$\mathbf{B}_{ij} = \mathbf{A}_{ij} - \frac{k_i k_j}{2M},$$

- Property of B_{ij}

$$\sum_j B_{ij} = \sum_j A_{ij} - \frac{k_i}{2M} \sum_j k_j = k_i - \frac{k_i}{2M} 2M = 0$$

- Just two communities:
  a division vector s as a set of quantities s_i for each vertex i.

$$s_i = \begin{cases} +1 & \text{if vertex } i \text{ belongs to group 1,} \\ -1 & \text{if vertex } i \text{ belongs to group 2} \end{cases}$$

$$\delta_{C_i C_j} = \frac{1}{2}(s_i s_j + 1) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ belong to the same group} \\ 0 & \text{if } i \text{ and } j \text{ belong to different groups} \end{cases}$$

Newman Spectral Method[New06]

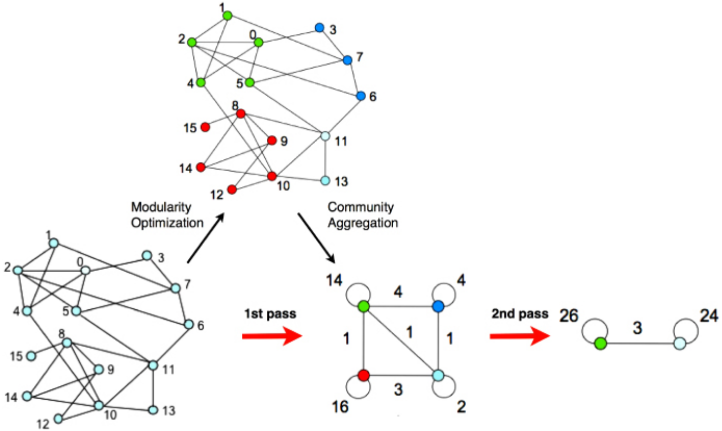
- Substituting

$$Q = \frac{1}{4}\sum_{ij} B_{ij}(s_i s_j + 1) = \frac{1}{4}\sum_{ij} B_{ij} s_i s_j = \frac{1}{4}\mathbf{s}^T\mathbf{B}\mathbf{s}$$

- A solution found similarly as for the spectral partitioning
  - The constraint $\mathbf{s}^T\mathbf{s} = \sum_i s_i^2 = N$
  - The solution $\mathbf{B}\mathbf{s} = \beta\mathbf{s}$
  - The modularity $Q = \frac{1}{4M}\beta\mathbf{s}^T\mathbf{s} = \frac{N}{4M}\beta$
  - For maximum modularity we should choose $\mathbf{s}$ to be the eigenvector $\mathbf{u}_1$ corresponding to the largest eigenvalue of the modularity matrix.
  - The constraint $s_i = \pm 1$.
- The best choice:
  - Select the $\mathbf{u}_1$ and maximize the product $\mathbf{s}^T\mathbf{u}_1 = \sum_i s_i[\mathbf{u}]_i$

$$s_i = \begin{cases} +1 & \text{if } [\mathbf{u}]_i > 0 \\ -1 & \text{if } [\mathbf{u}]_i < 0 \end{cases}$$

Community Structure Extraction - Louvain Method [BGLL08]



Repeated step

1. modularity is optimized by allowing only local changes of communities
2. the communities found are aggregated in order to build a new network of communities

Louvain Algorithm [BGLL08, Bar16]

$$Q = \frac{1}{2M}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2M}\right)\delta_{C_i C_j}$$

- The first term rewritten as a sum over communities

$$\frac{1}{2M}\sum_{i,j} A_{ij}\delta_{C_i C_j} = \sum_{c=1}^{n_c}\frac{1}{2M}\sum_{i,j\in C_c} A_{ij} = \sum_{c=1}^{n_c}\frac{M_c}{M}$$

where M_c is the number edges within community C_c

- The second term becomes

$$\frac{1}{2M}\sum_{i,j}\frac{k_i k_j}{2M}\delta_{C_i C_j} = \sum_{c=1}^{n_c}\frac{1}{(2M)^2}\sum_{i,j\in C_c} k_i k_j = \sum_{c=1}^{n_c}\frac{1}{4M^2}\sum_{i\in C_c} k_i \sum_{j\in C_c} k_j = \sum_{c=1}^{n_c}\frac{k_c^2}{4M^2}$$

where $k_c = \sum_{i\in C_c} k_i$ is the total degree of the nodes in community C_c

- Then

$$Q = \sum_{c=1}^{n_c}\left[\frac{M_c}{M} - \frac{k_c^2}{4M^2}\right]$$

Louvain Algorithm [BGLL08, Bar16]

$$Q = \frac{1}{2M} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2M} \right) \delta_{C_i C_j}$$

- The first term rewritten as a sum over communities

$$\frac{1}{2M} \sum_{i,j} A_{ij} \delta_{C_i C_j} = \sum_{c=1}^{n_c} \frac{1}{2M} \sum_{i,j \in C_c} A_{ij} = \sum_{c=1}^{n_c} \frac{M_c}{M}$$

  where M_c is the number edges within community C_c

- The second term becomes

$$\frac{1}{2M} \sum_{i,j} \frac{k_i k_j}{2M} \delta_{C_i C_j} = \sum_{c=1}^{n_c} \frac{1}{(2M)^2} \sum_{i,j \in C_c} k_i k_j = \sum_{c=1}^{n_c} \frac{1}{4M^2} \sum_{i \in C_c} k_i \sum_{j \in C_c} k_j = \sum_{c=1}^{n_c} \frac{k_c^2}{4M^2}$$

where $k_c = \sum_{i \in C_c} k_i$ is the total degree of the nodes in community C_c

- Then

$$Q = \sum_{c=1}^{n_c} \left[\frac{M_c}{M} - \frac{k_c^2}{4M^2} \right]$$

Louvain Algorithm - Merging Two Communities [BGLL08]

- Given two communities A and B with the total degrees k_A and k_B, respectively, in these communities.
  - The number M_A and M_B of edges in communities A and B, resp.
- The resulting (merged) community AB with the total degree k_{AB}
  - $k_{AB} = k_A + k_B$
  - The number of edges: $M_{AB} = M_A + M_B + m_{AB}$
  - where m_{AB} is the number of direct links between the nodes of communities A and B
- The change in modularity after merging of A with B and substitutions:

$$\Delta Q_{AB} = \left[\overbrace{\frac{M_{AB}}{M} - \frac{k_{AB}^2}{4M^2}}^{Q_{AB}} \right] - \left[\overbrace{\frac{M_A}{M} - \frac{k_A^2}{4M^2}}^{Q_A} + \overbrace{\frac{M_B}{M} - \frac{k_B^2}{4M^2}}^{Q_B} \right]$$

$$= \frac{m_{AB}}{M} - \frac{k_A k_B}{2M^2}$$

Louvain Algorithm - Moving One Node [BGLL08]

$$\Delta Q_{AB} = \frac{m_{AB}}{M} - \frac{k_A k_B}{2M^2}$$

- Merging a given isolated node i as the community $B = \{i\}$ [BGLL08]:

$$\begin{aligned}
\Delta Q_{Ai} &= \frac{m_{Ai}}{M} - \frac{k_A k_i}{2M^2} = \\
&= \frac{M_A}{2M} + \frac{2m_{Ai}}{2M} - \left(\frac{(k_A)^2}{(2M)^2} + \frac{2k_A k_i}{(2M)^2} + \frac{(k_i)^2}{(2M)^2} \right) - \\
&\quad - \frac{M_A}{2M} + \frac{(k_A)^2}{(2M)^2} + \frac{(k_i)^2}{(2M)^2} = \\
&= \left[\frac{M_A + 2m_{Ai}}{2M} - \left(\frac{k_A + k_i}{2M} \right)^2 \right] - \left[\frac{M_A}{2M} - \left(\frac{k_A}{2M} \right)^2 - \left(\frac{k_i}{2M} \right)^2 \right]
\end{aligned}$$

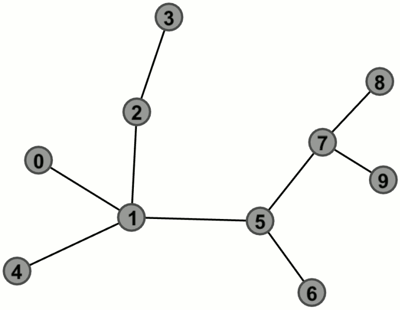
- If a single node i if removed from the community A then the change in modularity is $-\Delta Q_{Ai}$.

Louvain Algorithm [BGLL08]

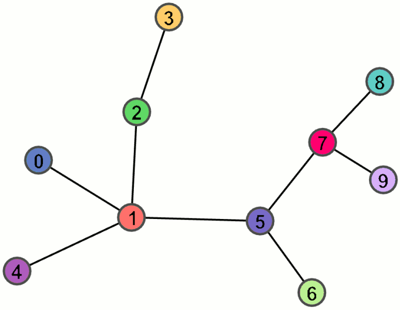
The Algorithm

1. A different community is assigned to each node of the network.
2. For each node i
   a. The neighbors j of i are considered
   b. The gain of modularity is evaluated for moving i from its community and placing it into the community of j.
   c. The node i is placed into the community for which the gain is maximum, but only if this gain is positive.
   d. Repeated for all nodes and
   e. Repeated until no further improvement can be achieved.
3. Build a new network whose nodes are the communities found during the first phase
4. The process is iterated from Step (2)

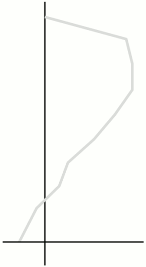
Louvain Algorithm [BGLL08]



Louvain Algorithm [BGLL08]

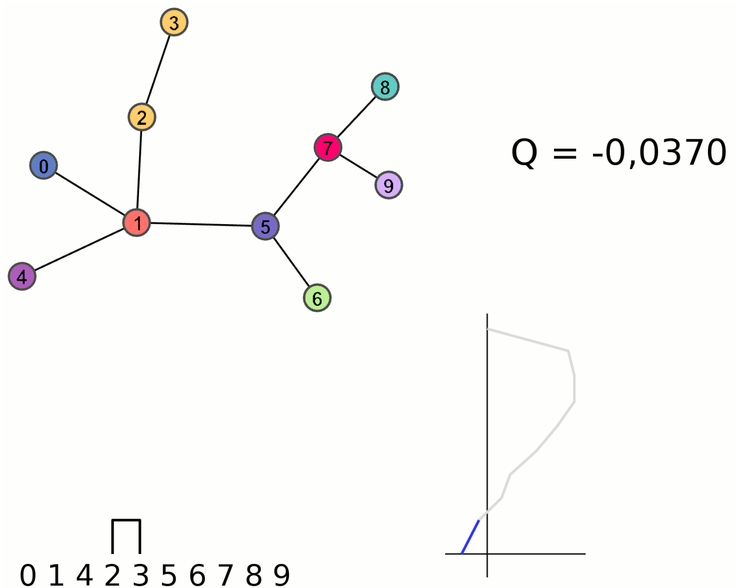


Q = -0,1358

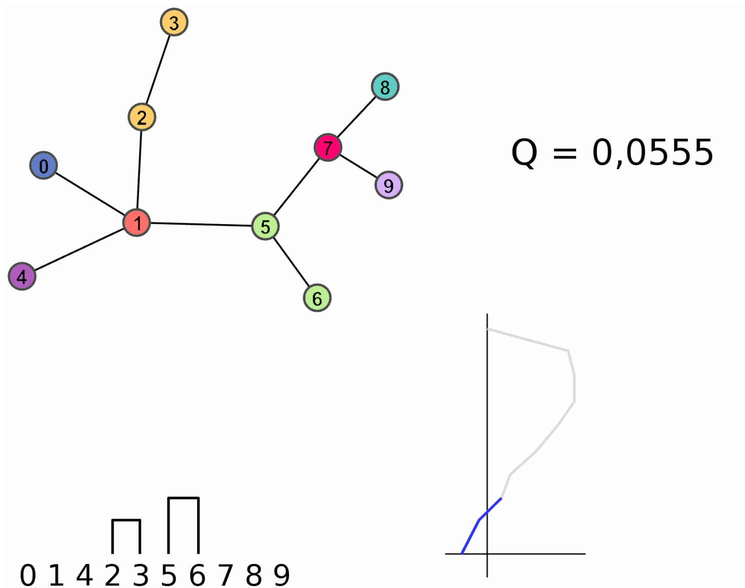


0 1 4 2 3 5 6 7 8 9

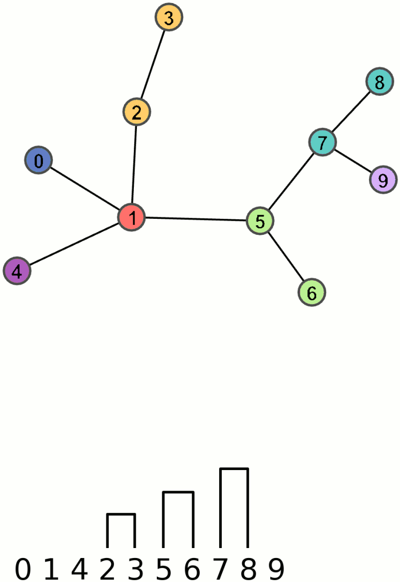
Louvain Algorithm [BGLL08]



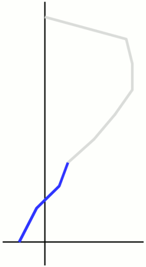
Louvain Algorithm [BGLL08]



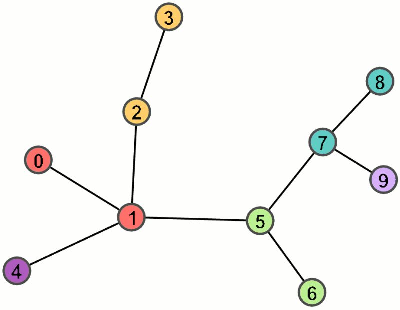
Louvain Algorithm [BGLL08]



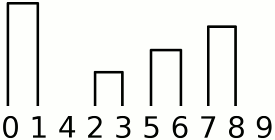
$Q = 0{,}0926$

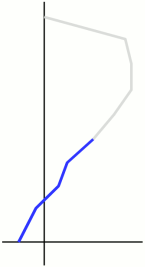


Louvain Algorithm [BGLL08]

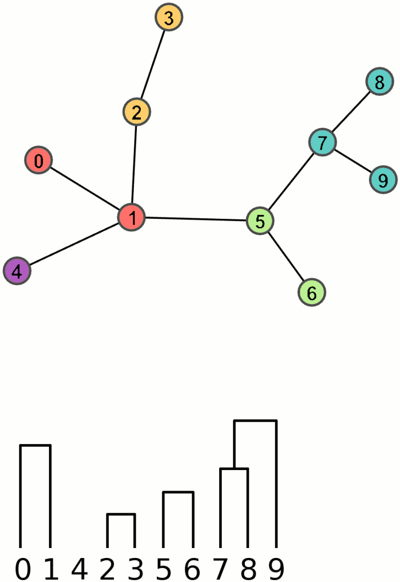


Q = 0,2345

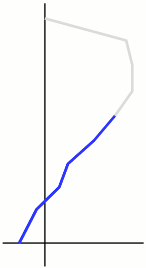




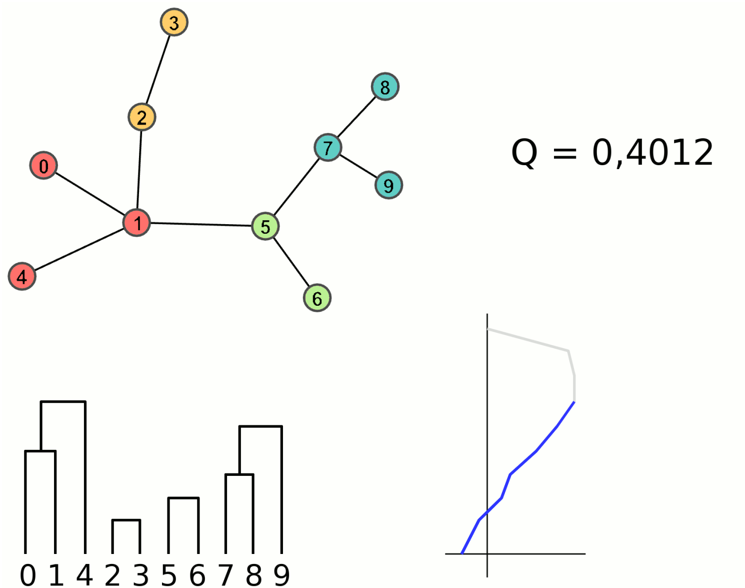
Louvain Algorithm [BGLL08]



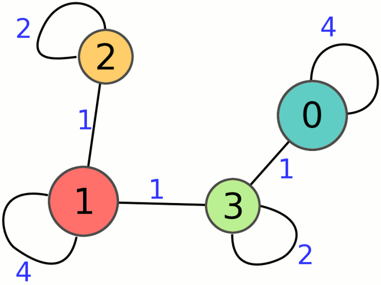
$Q = 0{,}3209$



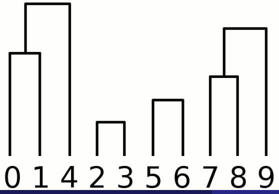
Louvain Algorithm [BGLL08]

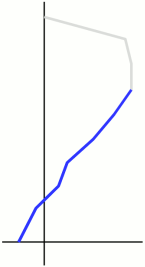


Louvain Algorithm [BGLL08]

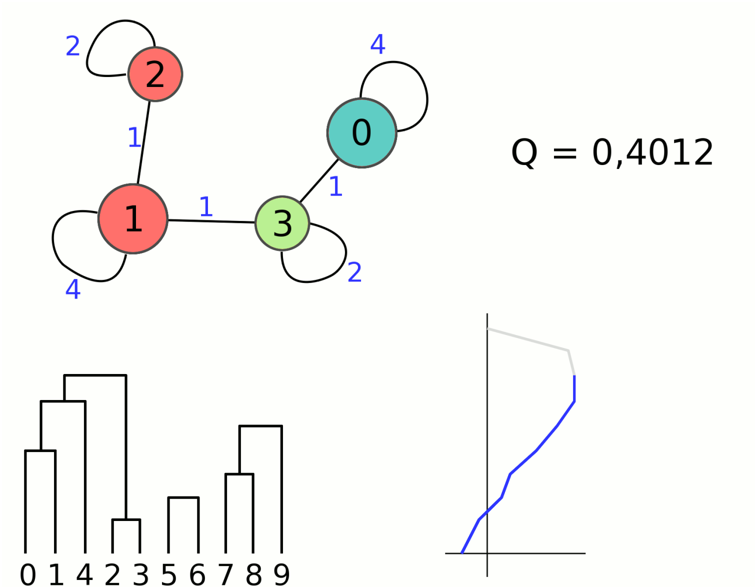


Q = 0,4012

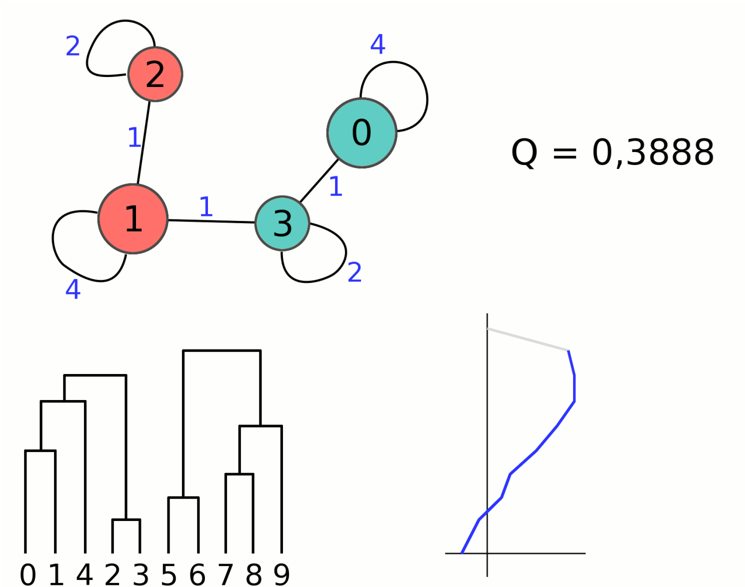




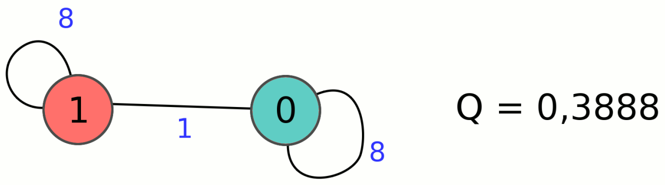
Louvain Algorithm [BGLL08]

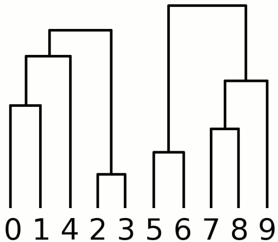


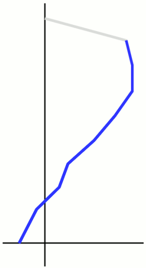
Louvain Algorithm [BGLL08]



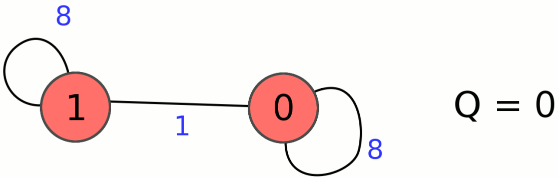
Louvain Algorithm [BGLL08]

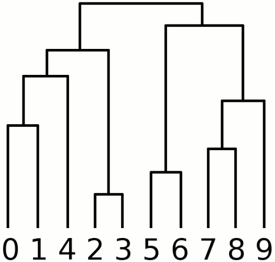


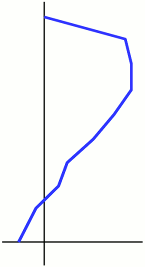




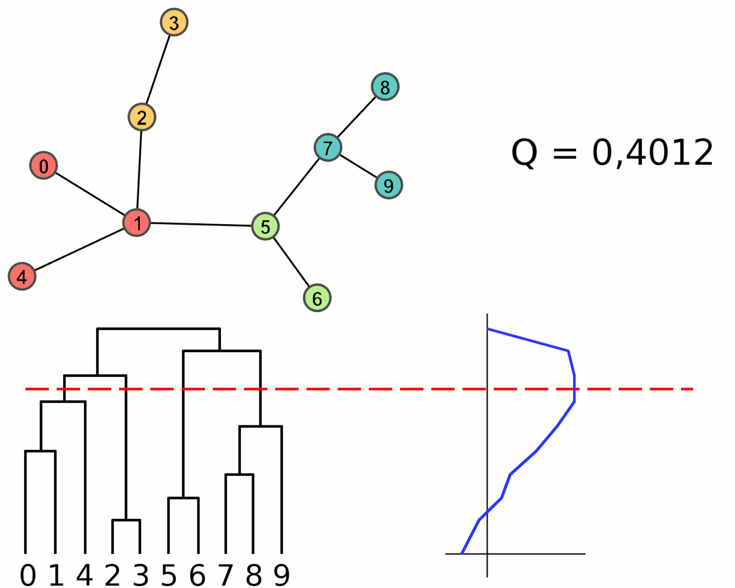
Louvain Algorithm [BGLL08]



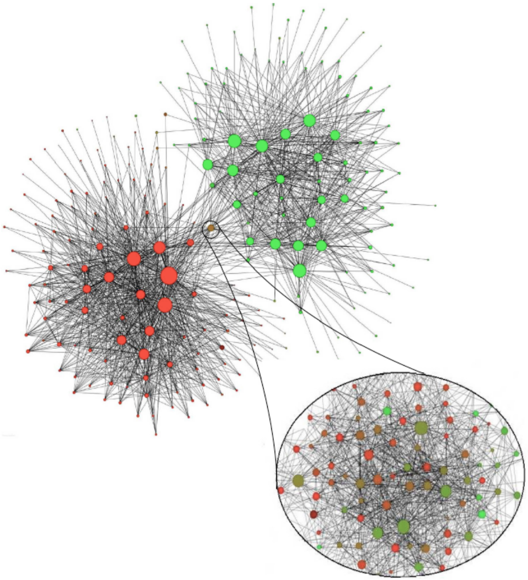




Louvain Algorithm [BGLL08]



Belgian Mobile Phone Network - Louvain Method [BGLL08]



- 2.6 millions customers
- Language: Dutch, English, French, German,
- 6.3 millions links
- Weights ...number of call + sms
- Red ...French,
- $> 93\%$ segregated,
- The center ...Brussels

Louvain Algorithm - Resolution Limit [Bar16]

$$\Delta Q_{AB} = \frac{m_{AB}}{M} - \frac{k_A k_B}{2M^2}$$

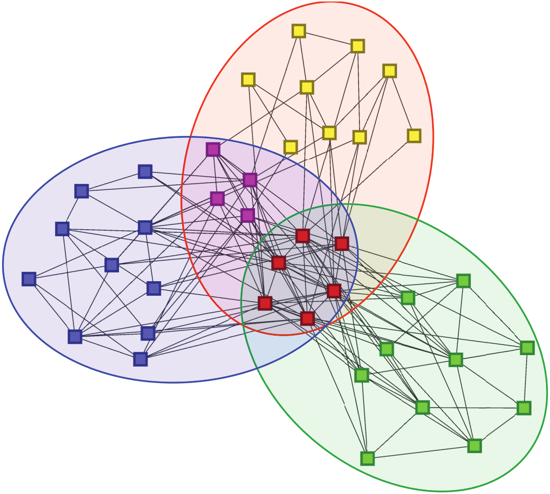
- If there is at least one link between the two communities
  - $m_{AB} \geq 1$
- and if $\frac{k_A k_B}{2M} < 1$
- then $\Delta Q_{AB} > 0$
- Therefore, if A and B are distinct communities linked with at least one edge, then they are merged if they are small enough.
- The **resolution limit**: assuming $k_A \approx k_B = k$ and if

$$k \leq \sqrt{2M}$$

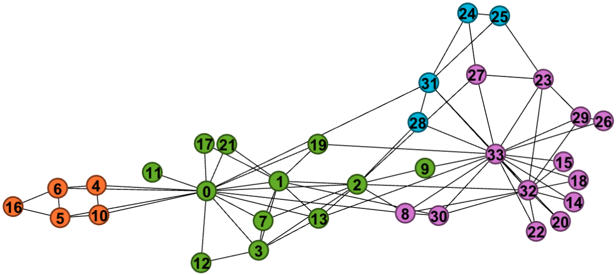
  then modularity increases by merging A and B.
- An artifact of modularity maximization:
  - If k_A and k_B are under the threshold, the *expected* number of links between them is smaller than one.
  - Proposed methods for resolution limit compensation.

Outline

Overlapping Communities [YL12]



Overlapping Communities

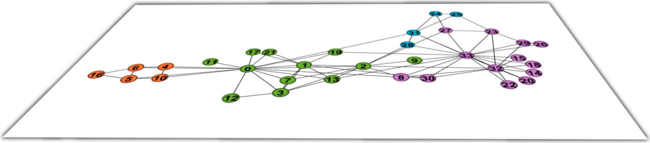


- An attempt to explaining the links of the observed network, "causes" of the graph creation.
- affiliation... "community membership"
- The probability that an edge between the nodes i and j is generated:

$$p(i,j) = 1 - \prod_{c \in C_{ij}} (1 - p_c)$$

C_{ij} ... a set of communities that i and j share

Overlapping Communities

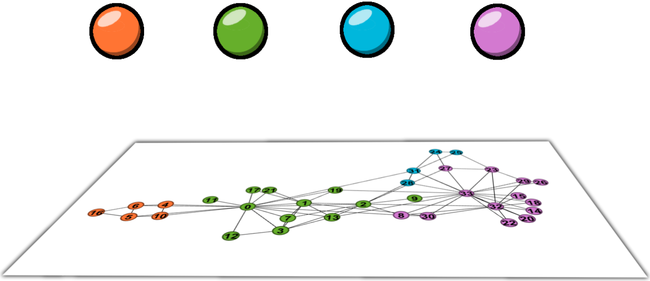


- An attempt to explaining the links of the observed network, "causes" of the graph creation.
- affiliation. . . "community membership"
- The probability that an edge between the nodes i and j is generated:

$$p(i,j) = 1 - \prod_{c \in C_{ij}} (1 - p_c)$$

C_{ij} . . . a set of communities that i and j share

Overlapping Communities

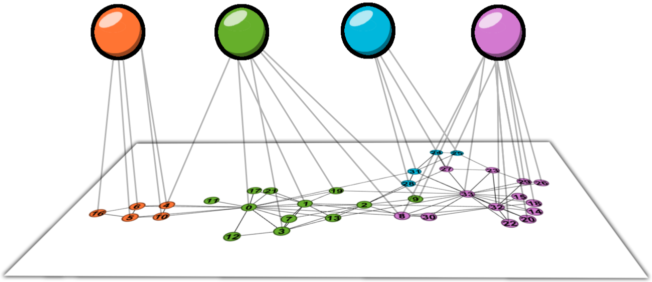


- An attempt to explaining the links of the observed network, “causes” of the graph creation.
- affiliation. . . “community membership”
- The probability that an edge between the nodes i and j is generated:

$$p(i,j) = 1 - \prod_{c \in C_{ij}} (1 - p_c)$$

C_{ij} . . . a set of communities that i and j share

Overlapping Communities

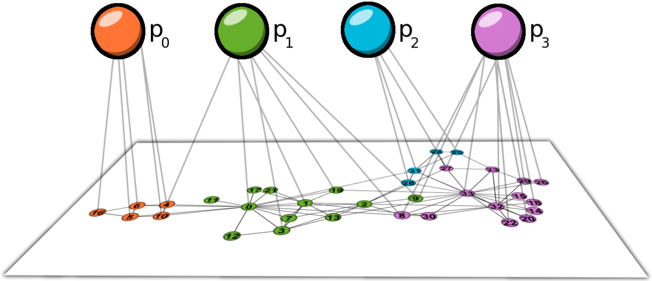


- An attempt to explaining the links of the observed network, "causes" of the graph creation.
- affiliation... "community membership"
- The probability that an edge between the nodes i and j is generated:

$$p(i,j) = 1 - \prod_{c \in C_{ij}} (1 - p_c)$$

C_{ij} ... a set of communities that i and j share

Overlapping Communities



- An attempt to explaining the links of the observed network, “causes” of the graph creation.
- affiliation. . . “community membership”
- The probability that an edge between the nodes i and j is generated:

$$p(i, j) = 1 - \prod_{c \in C_{ij}} (1 - p_c)$$

C_{ij} . . . a set of communities that i and j share

Affiliation Graph Model

Given

- an observed graph: $G(V, E)$,
- model afilací: $AGM(B(V, C, M), \mathcal{P} = \{p_c | c \in C\})$.
- C . . . a set of communities,
- M . . . affiliation (it assigns nodes to communities)

Then the probability that the model AGM generates the graph G is

$$P(G|_{B\mathcal{P}}) = \prod_{(i,j) \in E} p(i,j) \prod_{(i,j) \notin E} (1 - p(i,j))$$

Summary

- Community detection
- Community detection method taxonomy
- Kernighan-Lin algorithm
- Spectral bisection
- Hierarchical clustering
- Community detection based on modularity
- Overlapping communities

Competencies

- Describe the concept of community.
- What is null model of a graph?
- What types of community dection methods do you know?
- Describe Kernighan-Lin algorithm.
- Describe graph partitioning using the spectral bisection method.
- What is modularity of graph proposed by Newman?
- How can modularity be used for community detection?
- Describe principles of the Louvain algorithms.
- What is the resolution limi in community detection based on modularity?
- Describe principles of overlapping community detection.

Acknowledgements

A number of slides were originally prepared by Tomas Zikmund (a BSc. student at FNSPE CTU Prague) during his preparation for BSc. thesis defense.

References I

[Bar16] Albert-László Barabási. *Network Science*. Cambridge University Press, 1 edition, 2016.

[BAV13] A. Browet, P.-A. Absil, and P. Van Dooren. Fast community detection using local neighbourhood search. *ArXiv e-prints*, August 2013.

[BGLL08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[DM16] Veronika Dulíková and Radek Mařík. Data mining applied to ancient egypt data in the old kingdom. In *Sborník abstraktů z konference Počítačová podpora v archeologii 2016, Velké Pavlovice, 30.května - 1. června 2016*. Dept. of Archaeology and Museology, Faculty of Arts, Masaryk University, CZ, 2016.

[Dul08] Veronika Dulíková. Instituce vezirátu ve staré říši. Master's thesis, Praha: Univerzita Karlova v Praze (nepublikovaná magisterská diplomová práce), 2008.

[FH16] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1 – 44, 2016. Community detection in networks: A user guide.

[Fie73] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[Fie75] Miroslav Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(4):619–633, 1975.

[KL70] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, Feb 1970.

[New04] M. E. J. Newman. Detecting community structure in networks. *Eur. Phys. J. B 38*, pages 321–330, 2004.

[New06] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[New10] M. Newman. *Networks: an introduction*. Oxford University Press, Inc., 2010.

References II

[Weh13] Stefan Wehrli. Social network analysis, lecture notes, December 2013.

[YL12] Jaewon Yang and Jure Leskovec. Structure and overlaps of communities in networks. September 2012.