Agglomerative Community Detection

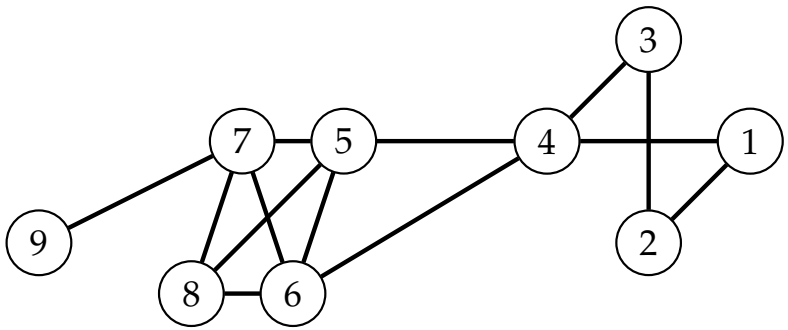
The Agglomerative procedure merges communities as follows:

- Start with the vertices as individual communities.
- At each step, merge communities successively into larger communities following a certain criterion, e.g., based on modularity increase.

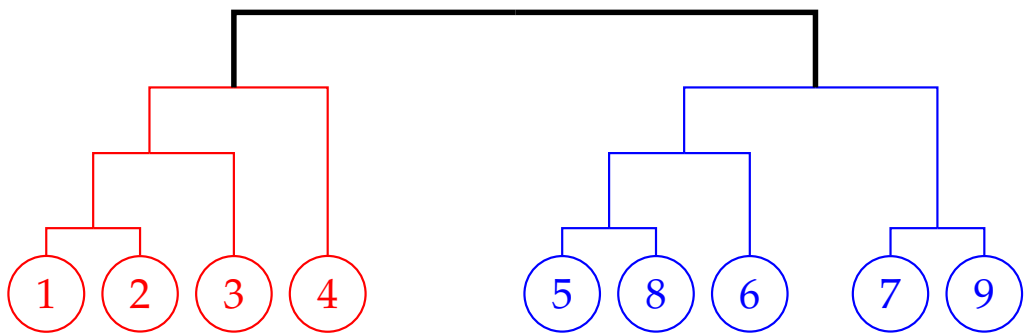
Agglomerative Community Detection

The Agglomerative procedure merges communities as follows:

- Start with the vertices as individual communities.
- At each step, merge communities successively into larger communities following a certain criterion, e.g., based on modularity increase.



Input Graph



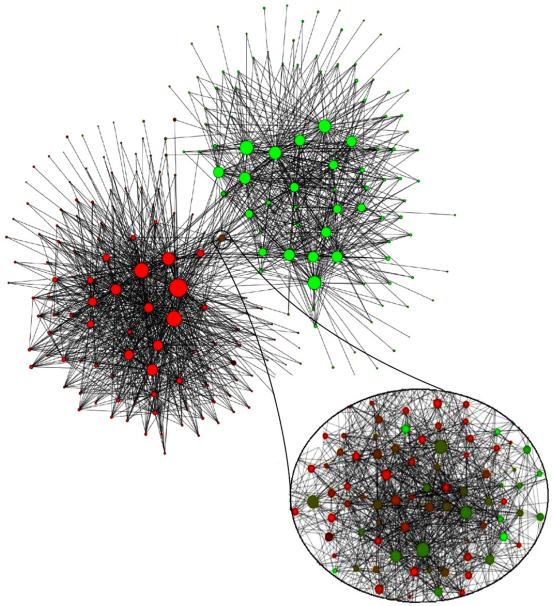
Communities Dendrogram

Louvain Method: Idea

- The Louvain method is an agglomerative community detection technique that uses modularity as an optimization function.
- The Louvain method is a fast solution: complexity is linear with the number of edges.
- The application of Louvain on a typical network of 2 million nodes takes few minutes.

Louvain Method: Application

- Telecom network in Belgium (calls between people) where each color represents a different language spoken by people.
- Each community has more than 1000 people.



Modularity: Idea

- We assume that real-world networks should be far from random. Therefore, the more distant they are from this randomly generated network, the more structural they are.
- Modularity defines this distance and modularity maximization tries to maximize this distance.
- The modularity can be considered as a measure of how well a network is partitioned into communities.

MODULARITY: DEFINITION/1

- The modularity is defined as:

$$Q = \texttt{(number of edges within groups)} - \texttt{(expected number within groups)}$$

- Formally:

$$Q = \frac{1}{2m} \sum_{i,j \in V} (a_{ij} - \frac{d_i d_j}{2m}) \delta_{C_i, C_j}$$

where:

- $a_{ij} \in A$ (the adjacency matrix);
- d_i and d_j are the degrees of v_i and v_j respectively;
- δ_{C_i, C_j}: Kronecker delta symbol (1 if $C_i = C_j$, 0 otherwise)

Modularity: Definition/2

- To simply the calculations, the modularity can be computed using this (equivalent) formula:

$$Q = \sum_{c=1}^{n_c} [\frac{L_c}{m} - (\frac{k_c}{2m})^2]$$

where:

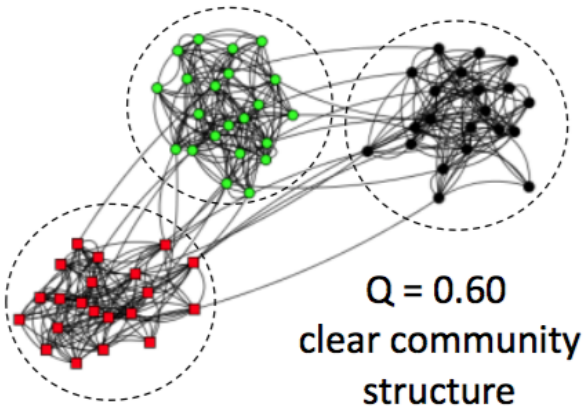
- n_c: number of communities;
- L_c: the total number of links within the community c;
- k_c: the total degree of the nodes in this community.

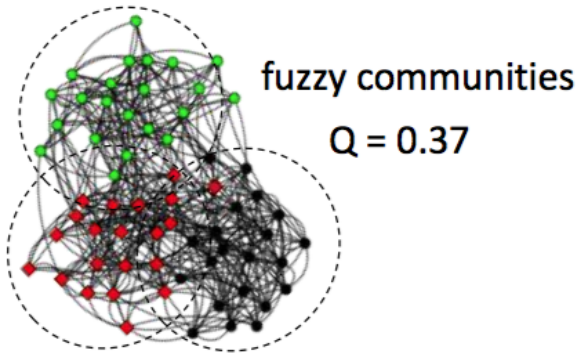
Modularity: Range

- Modularity values take range in the interval [-1, 1].
- The modularity is positive if the number of edges within groups exceeds the number of expected edges.
- High values of Q indicate a strong community structure.

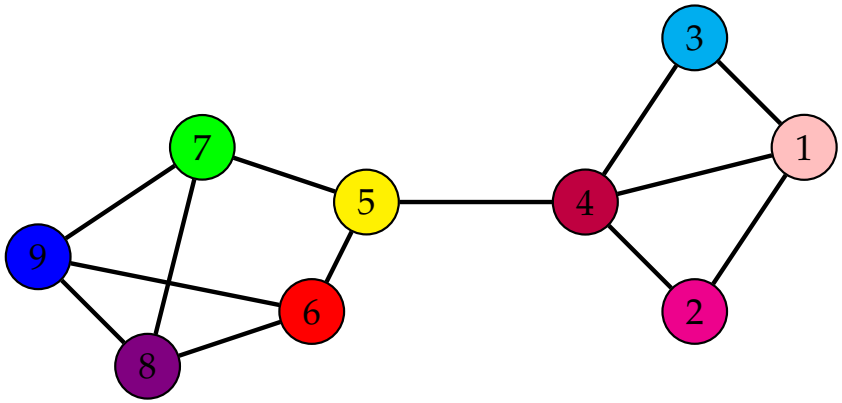
MODULARITY: EXAMPLE/1

- In a random graph, we expect that any possible partition would lead to $Q = 0$.
- Typically, in non-random graphs modularity takes values between 0.3 and 0.7.

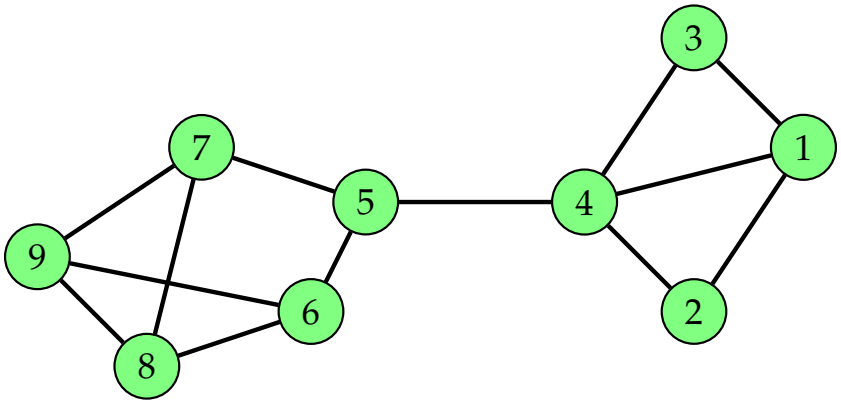




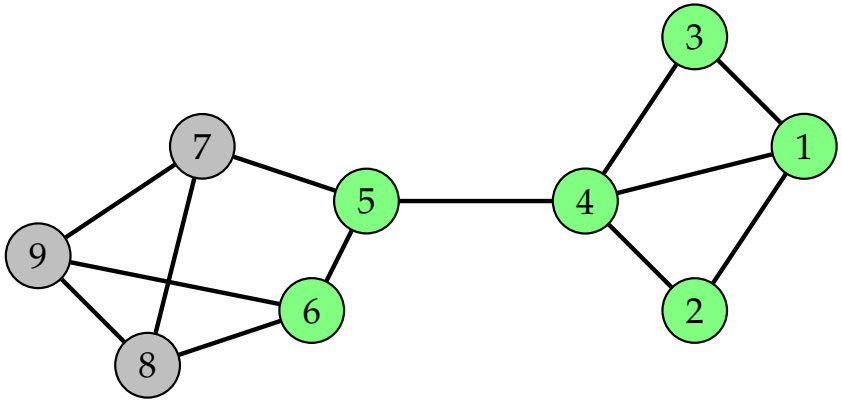
MODULARITY: EXAMPLE/2



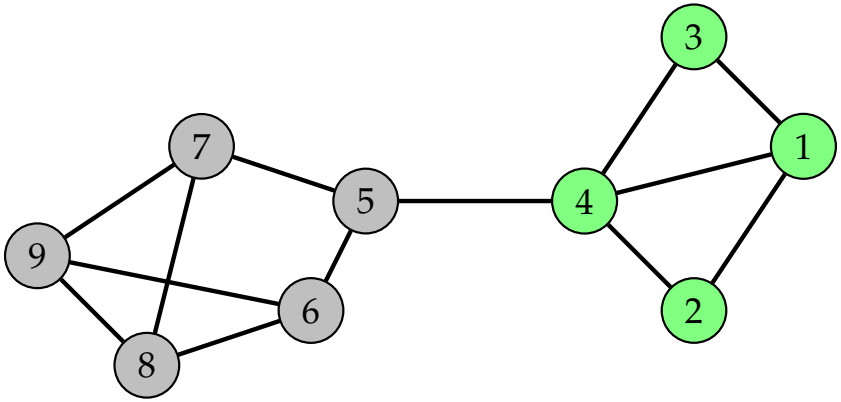
(a) Negative modularity: Q=-0.12



(b) Single community: Q=0



(c) Suboptimal communities: Q=0.22



(d) Optimal communities: Q=0.41

Modularity gain

- A core part of the Louvain technique is to use modularity gain ΔQ.
- Modularity gain captures the difference in modularity when merging two nodes i and j into the same community:

$$\Delta Q_{ij} = \frac{1}{2m}(d_{ij} - \frac{d_i d_j}{m})$$

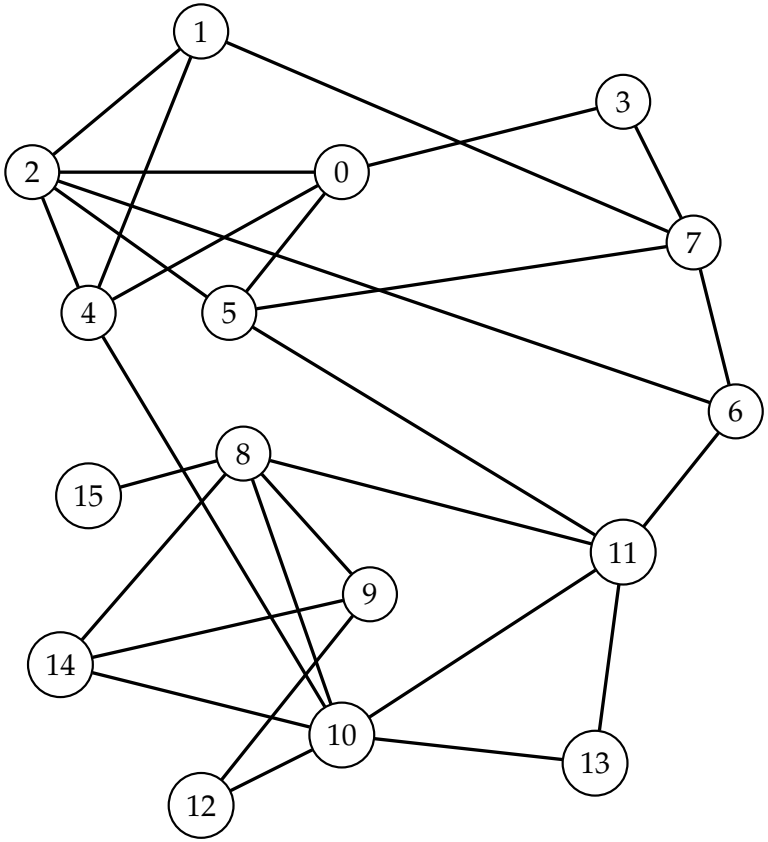
Louvain Method: Algorithm

For each passage:

- Step 1: Initialization: node = community
- Step 2: Remove node v_i from its community
- Step 3: Insert v_i in a neighboring community that maximizes ΔQ
- Step 4: Repeat Step 1 until the partition does not evolve
- Step 5: Transform the communities into (hyper)nodes and Repeat Step 1

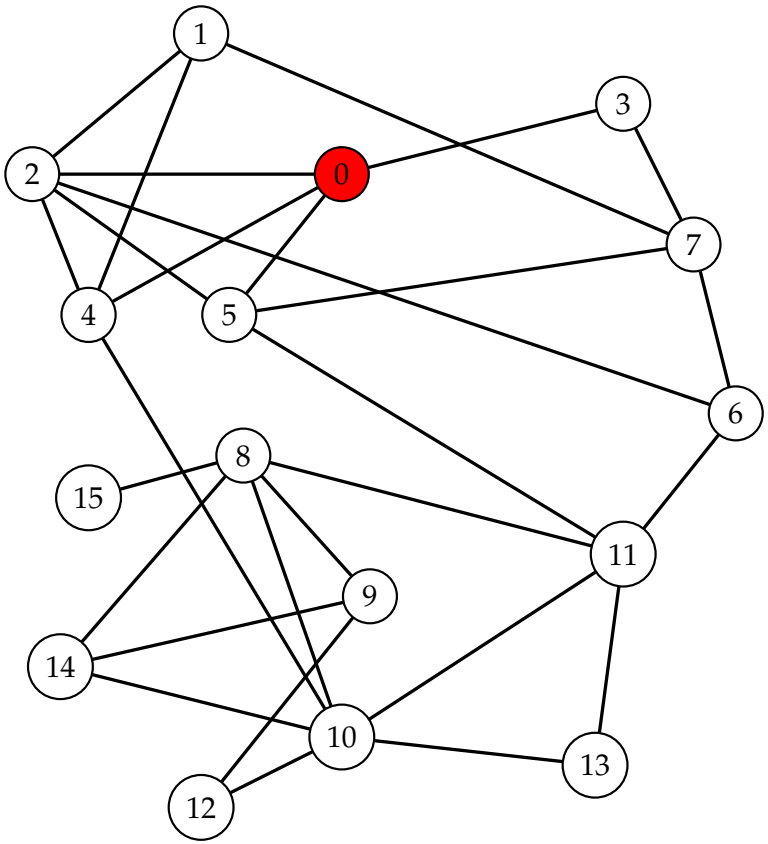
We do passages until convergence

Louvain Method: Example



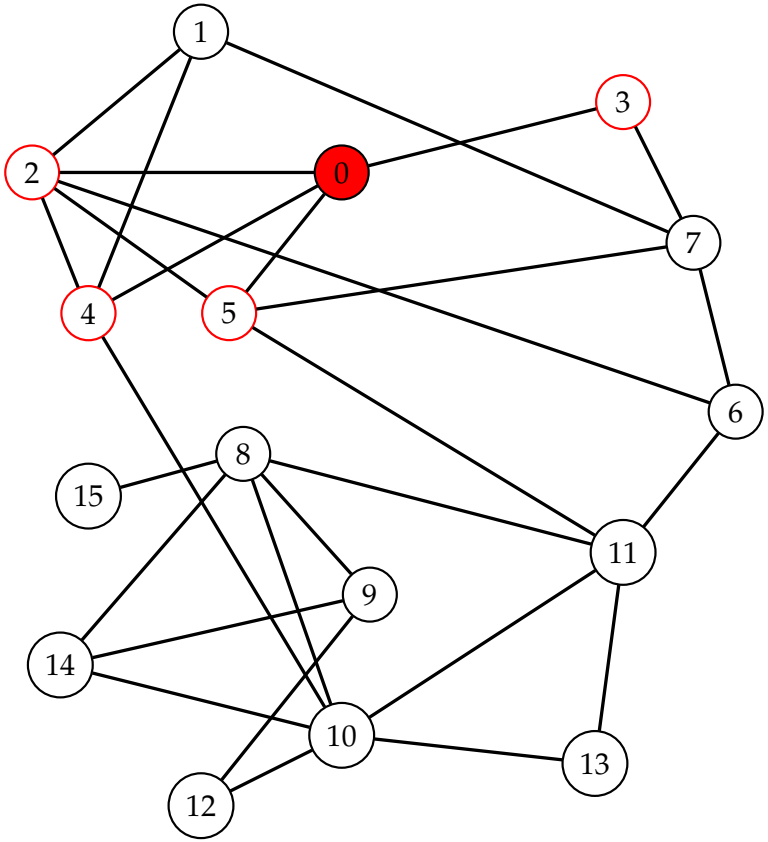
- We take as input a network of 16 nodes
- Each node represents a community
- Passage 1; iteration 1

Louvain Method: Example



- We start with node 0

Louvain Method: Example



- The neighboring communities of 0 are 2, 3, 4 and 5
- $m = 27$

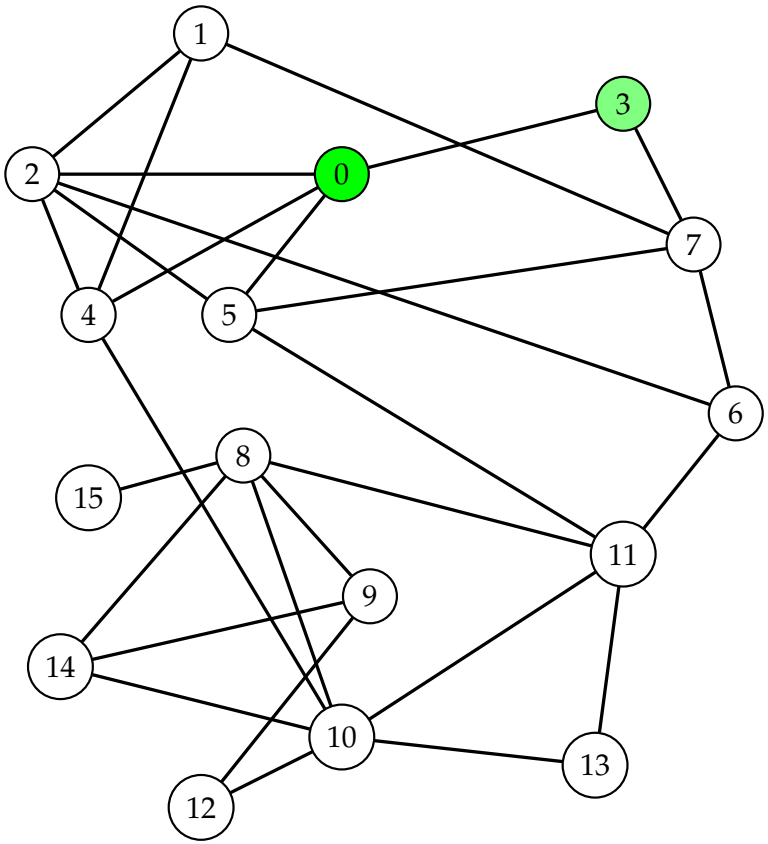
$$\Delta Q_{0,2} = \frac{1}{54}(2 - \frac{4 \times 5}{27}) = 0.023$$

$$\Delta Q_{0,3} = \frac{1}{54}(2 - \frac{4 \times 2}{27}) = 0.032$$

$$\Delta Q_{0,4} = \frac{1}{54}(2 - \frac{4 \times 4}{27}) = 0.026$$

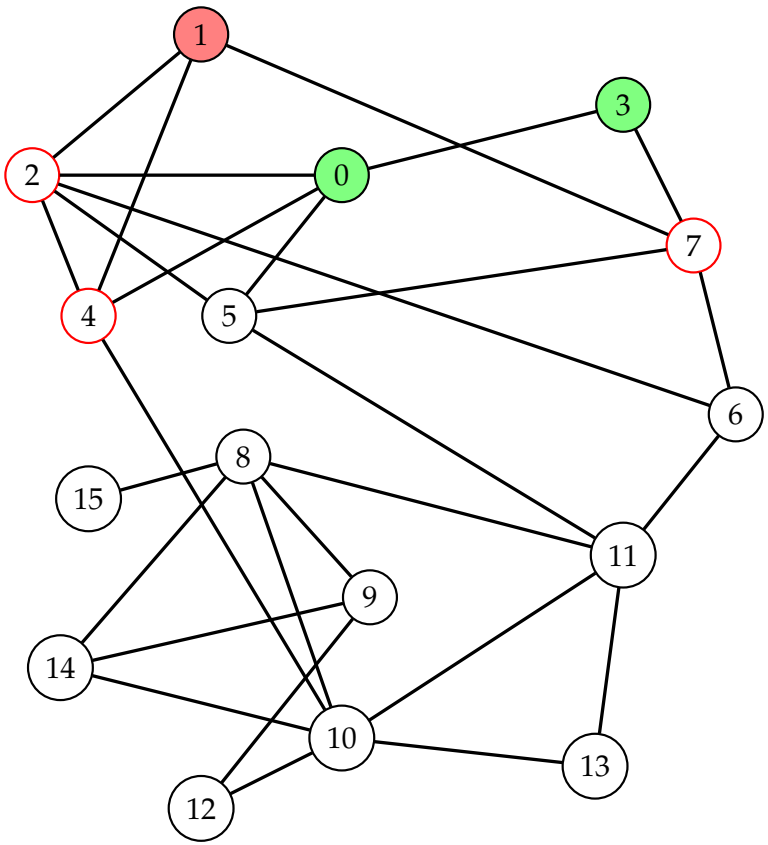
$$\Delta Q_{0,5} = \frac{1}{54}(2 - \frac{4 \times 4}{27}) = 0.026$$

Louvain Method: Example



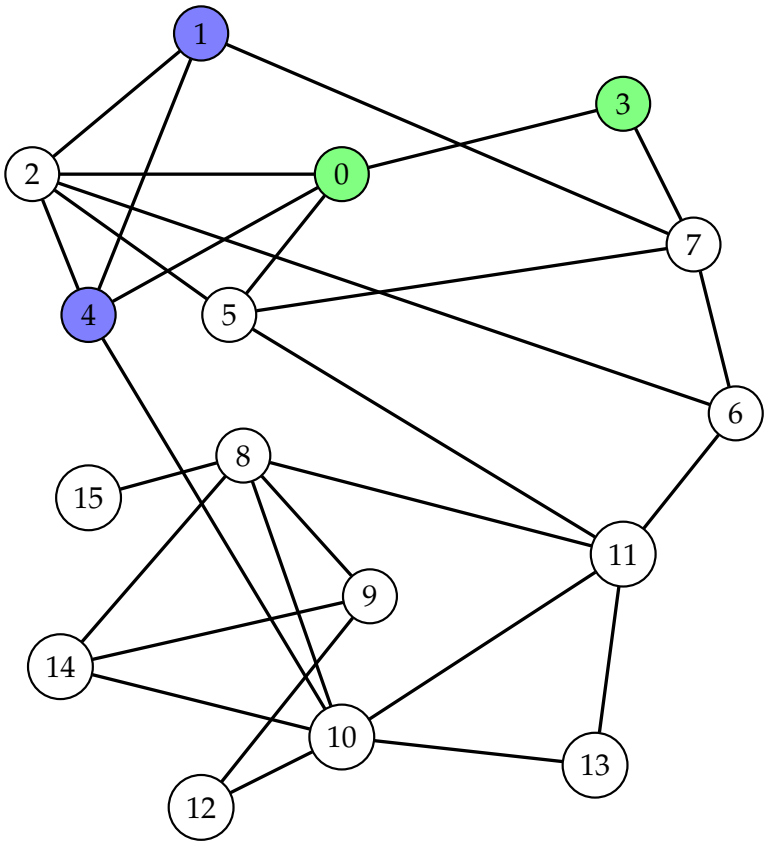
- 0 is put in C(3) with best Q increase

Louvain Method: Example



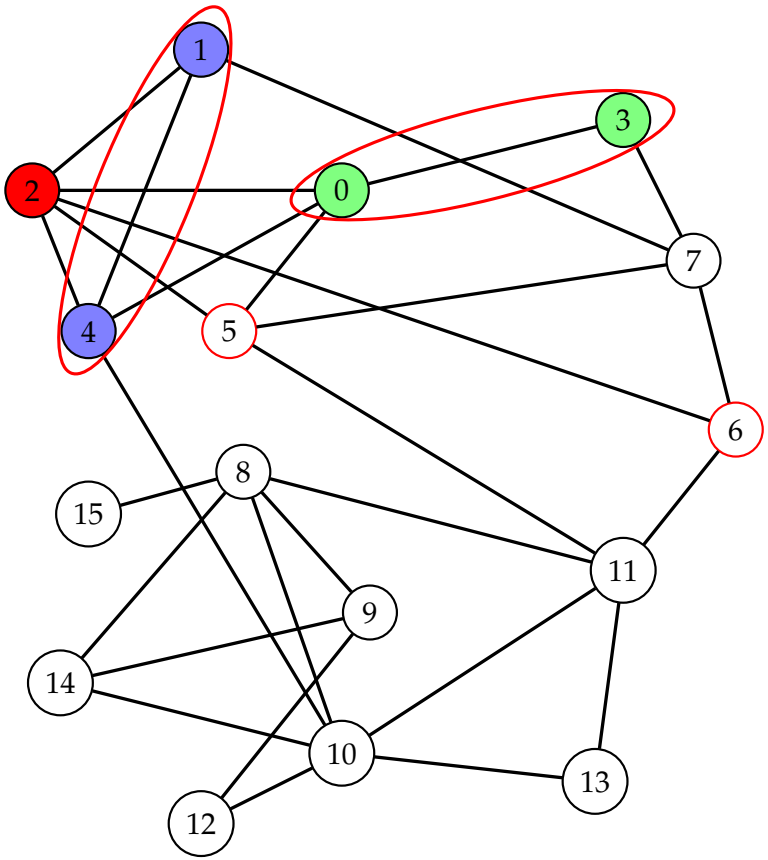
- Considering 1, its neighboring communities are 2, 4 and 7

Louvain Method: Example



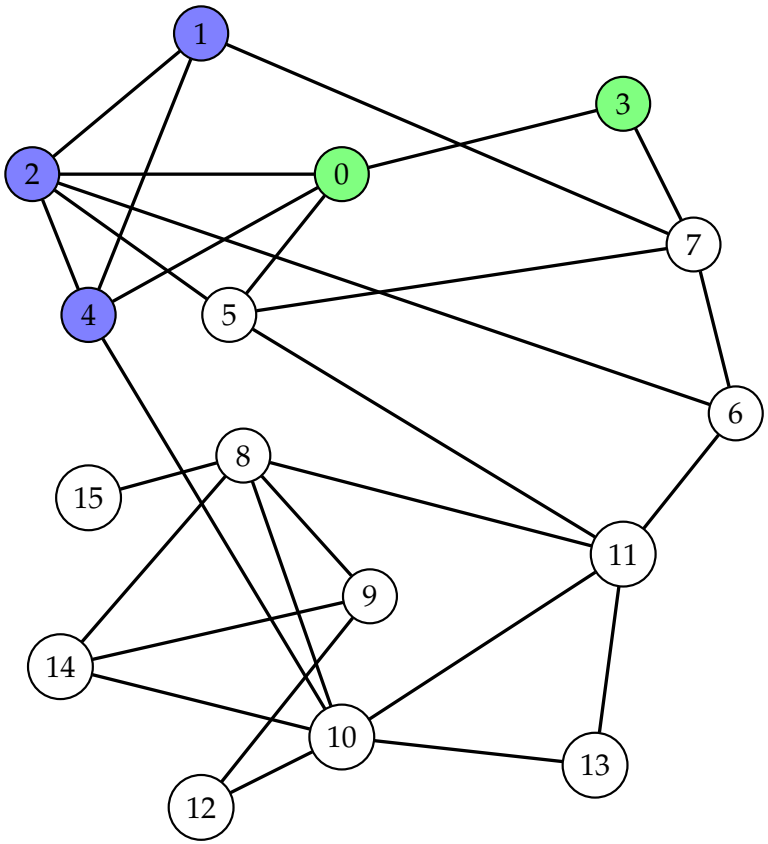
- 1 is put in C(4) with best Q increase

Louvain Method: Example



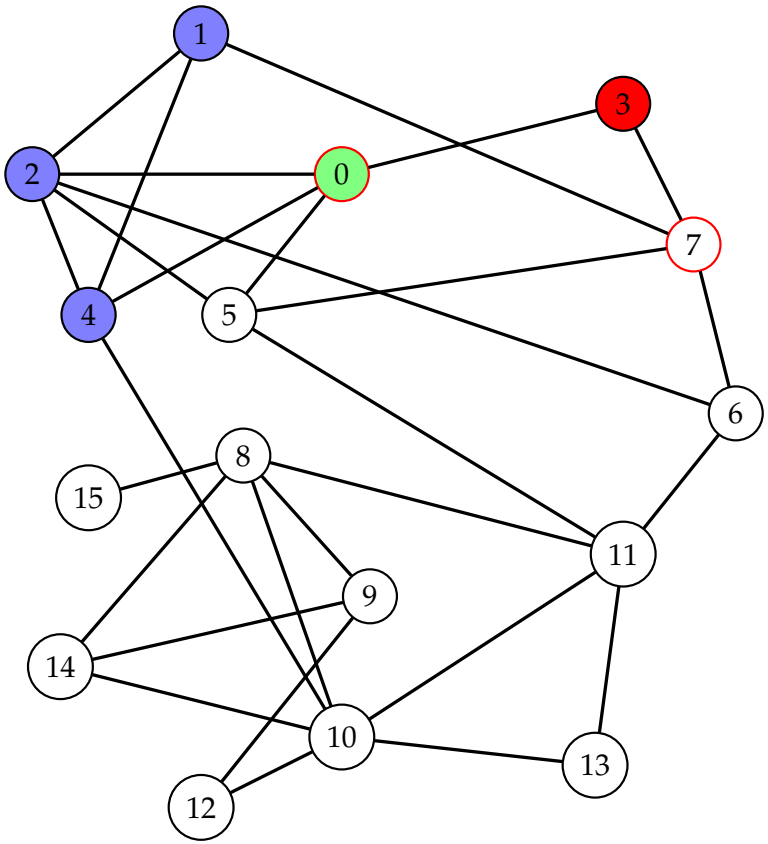
- Considering 2, its neighboring communities are {0,3}, {1, 4}, 5 and 6

Louvain Method: Example



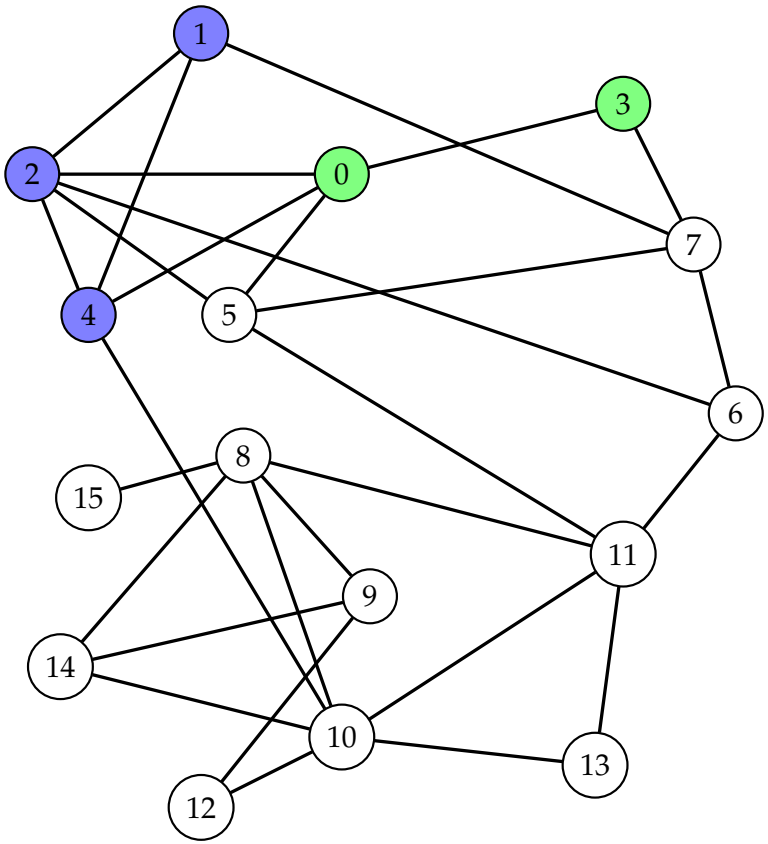
- 2 is put in C(1,4) with best Q increase

Louvain Method: Example



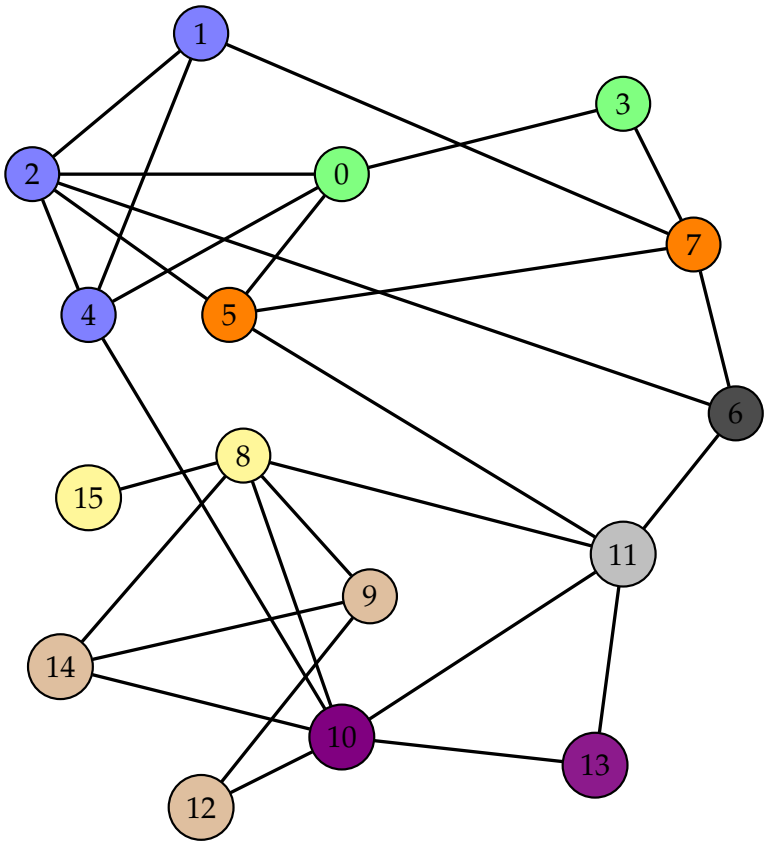
- Considering 3, its neighboring communities are 0 and 7

Louvain Method: Example



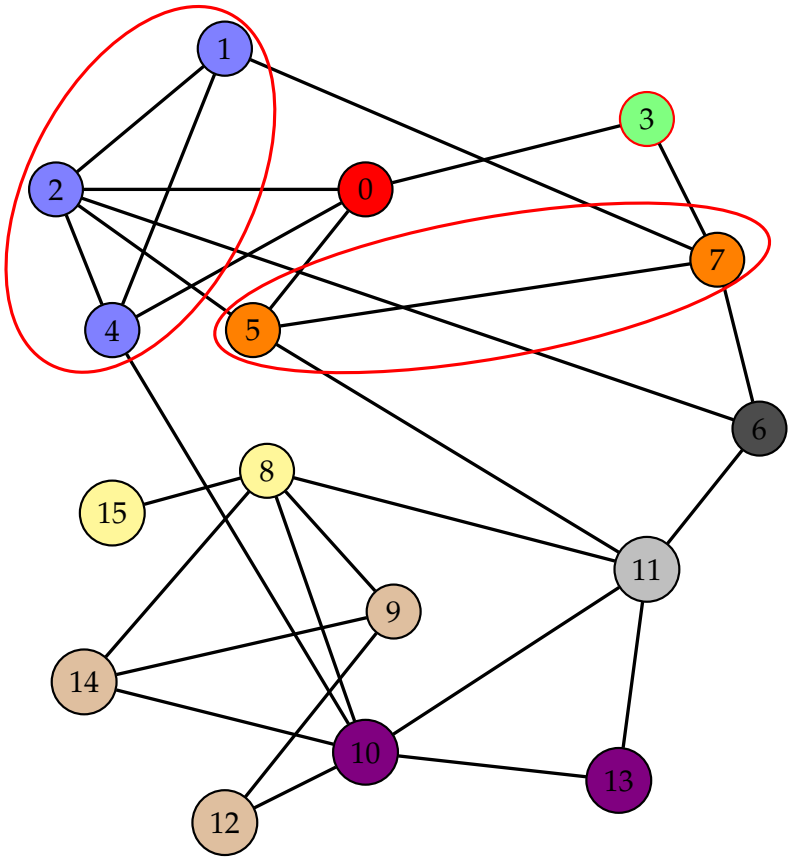
- 3 stays in the same community C(0,3), otherwise Q decreases

Louvain Method: Example



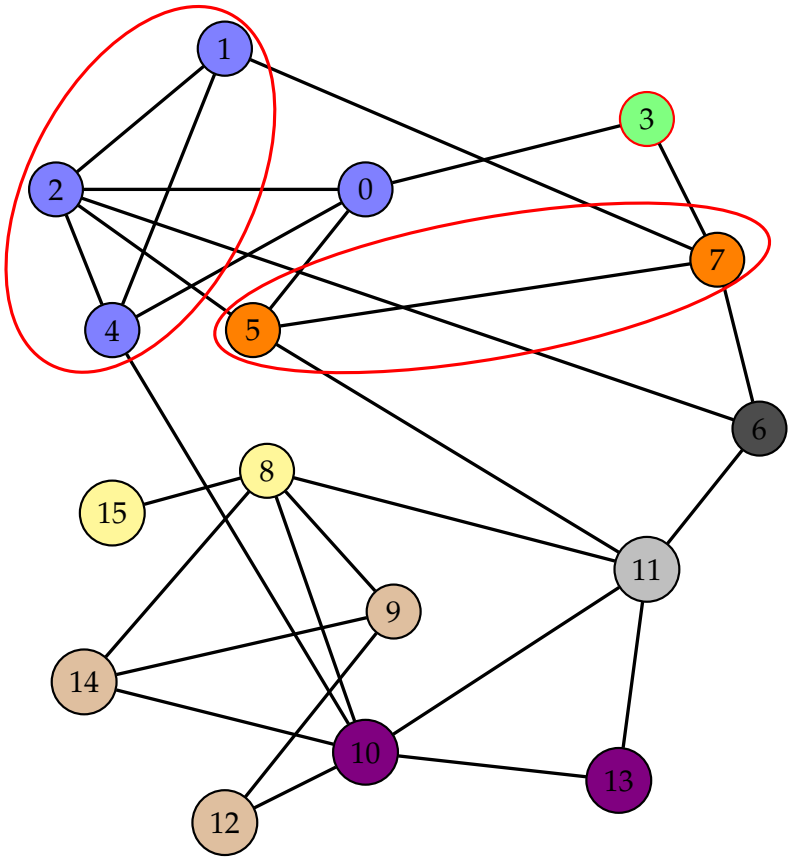
- At the end of iteration 1 we get

Louvain Method: Example



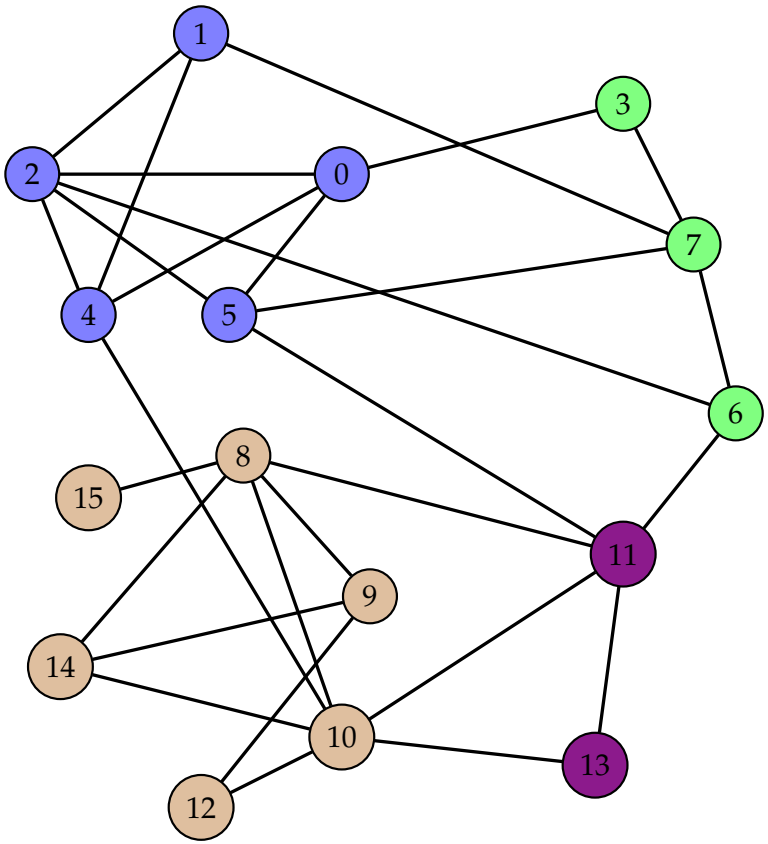
- Passage 1, iteration 2, consider node 0
- The neighboring communities of 0 are {1,2,4}, {5,7} and 3

Louvain Method: Example



- 0 is put in C(1,2,4), best Q increase

Louvain Method: Example



- After 4 iterations, no change anymore.
- Passage 2: $C_1 = \{0, 1, 2, 4, 5\}$, $C_2 = \{3, 6, 7\}$, $C_3 = \{11, 13\}$ and $C_4 = \{8, 9, 10, 12, 14, 15\}$.
- The same result remains after passage 2 and passage 3.

TABLE OF CONTENTS — SL04

Community Evaluation

- How to quantify how well a given method detects communities that exist in a graph?
- The evaluation can be performed on real-world graphs or synthetic graphs.
- The community detection can be evaluated depending on whether the graph is
  - with ground truth or
  - without ground truth

Evaluation with Ground Truth

- Networks with ground truth pertain to the networks with known community structure.
- In order to quantify the quality of the community detection, the following evaluation measures are used:
  - Precision/recall or F-Measure
  - Purity
  - Normalized Mutual Information (NMI)

Precision and Recall (1/2)

- *Correct decisions*:
  - **True Positive (TP)**: when similar members are assigned to the same community.
  - **True Negative (TN)**: when dissimilar members are assigned to different communities.
- *Errors*:
  - **False Positive (FP)**: when dissimilar members are assigned to the same community.
  - **False Negative (FN)**: when similar members are assigned to different communities.

Precision and Recall (2/2)

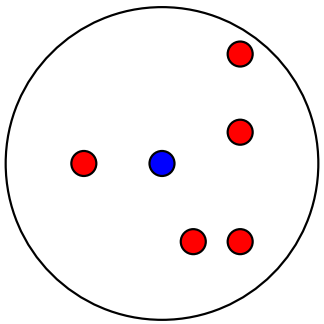
- The precision (P) is the ability to retrieve top-ranked communities that are mostly relevant.

$$P = \frac{Relevant\ and\ retrieved}{Retrieved} = \frac{TP}{TP + FP}$$

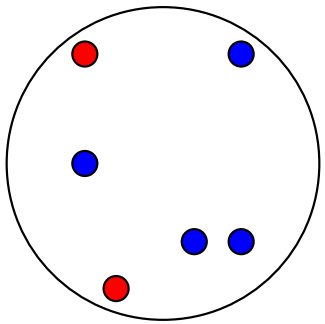
- The recall (R) is the ability of the search to find all of the relevant communities in the network

$$R = \frac{Relevant\ and\ retrieved}{Relevant} = \frac{TP}{TP + FN}$$

Precision and Recall: Example



Community 1



Community 2

- $TP = \underbrace{\binom{5}{2}}_{comm\ 1} + \underbrace{\binom{4}{2}}_{comm\ 2} = 16$
- $FP = \underbrace{(5 \times 1)}_{comm\ 1} + \underbrace{(4 \times 2)}_{comm\ 2} = 13$
- $TN = (5 \times 4) + (2 \times 1) = 22$
- $FN = (5 \times 2) + (4 \times 1) = 14$

- $P = \frac{TP}{TP+FP} = \frac{16}{16+13} = 0.55$
- $R = \frac{TP}{TP+FN} = \frac{16}{16+14} = 0.53$

F-Measure

- The precision and recall don't make sense in the isolation from each other: high level of P may be obtained by reducing R and vice versa.
- F-Measure combines P and R into one measure:

$$F = 2 \times \frac{P \times R}{P + R}$$

- Applied to our previous example we get:

$$F = 2 \times \frac{0.55 \times 0.53}{0.55 + 0.53} = 0.54$$

Purity: Definition

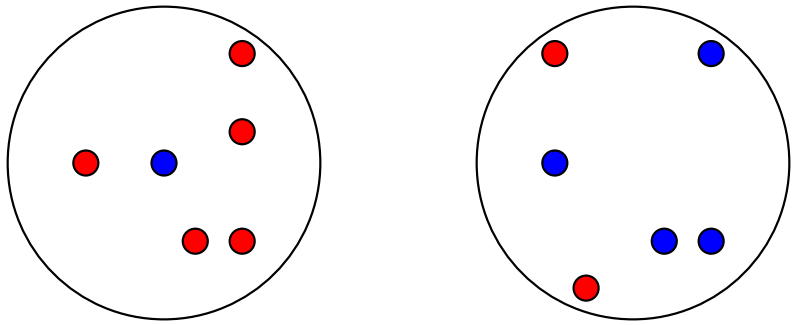
- Purity pertains to the fraction of instances that have labels equal to the label of the community's majority:

$$Purity = \frac{1}{n} \sum_{i=1}^{k} \max_{j} |C_i \cap L_j|$$

with

- k: the number of communities
- n: total number of nodes
- C_i: the set of members in community i
- L_j: the set of instances with label j in all communities

PURITY: EXAMPLE



Community 1 Community 2

$$
\begin{aligned}
Purity &= \frac{1}{12}\max(5,1) + \frac{1}{12}\max(4,2) \\
&= \frac{9}{12} \\
&= 0.75
\end{aligned}
$$

Mutual Information (MI)/1

- Purity is sensitive to singleton communities (of size 1) or very large communities.
- The Mutual Information (MI) measures the amount of information that two random variables share.
- MI is used to measure the amount of information one community carries regarding the ground truth.

Mutual Information (MI)/2

$$MI(X,Y) = \sum_{h \in H} \sum_{l \in L} \frac{n_{h,l}}{n} \log \frac{n \times n_{h,l}}{n_h \times n_l}$$

where:

- l and h are respectively the known (with labels) and the found communities;
- n_h and n_l are the number of members in the community h and l, respectively;
- $n_{h,l}$ is the number of members in community h and labeled l;
- n is the number of vertices in the network.

Normalized Mutual Information (NMI)/1

- MI is unbounded. NMI is a normalized value of the MI that ranges between 0 and 1.
- NMI values close to 1 indicate high similarity between communities found and labels.
- Values close to zero indicate high dissimilarity between them.

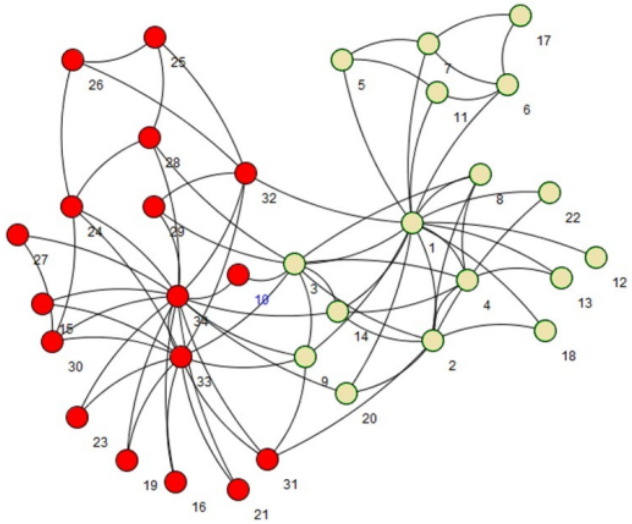
Normalized Mutual Information (NMI)/2

$$NMI = \frac{\sum_{h \in H} \sum_{l \in L} n_{h,l} \log \frac{n \times n_{h,l}}{n_h \times n_l}}{\sqrt{(\sum_{h \in H} n_h \times \log \frac{n_h}{n})(\sum_{l \in L} n_l \times \log \frac{n_l}{n})}}$$

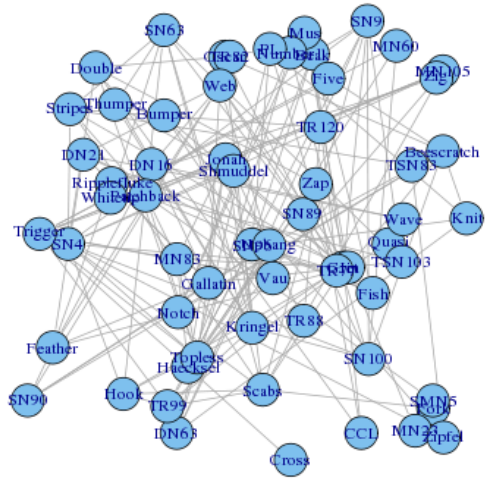
where:

- l and h are respectively the known (with labels) and the found communities;
- n_h and n_l are the number of members in the community h and l, respectively;
- $n_{h,l}$ is the number of members in community h and labeled l;
- n is the number of vertices in the network.

NETWORKS WITH GROUND TRUTH (1/2)



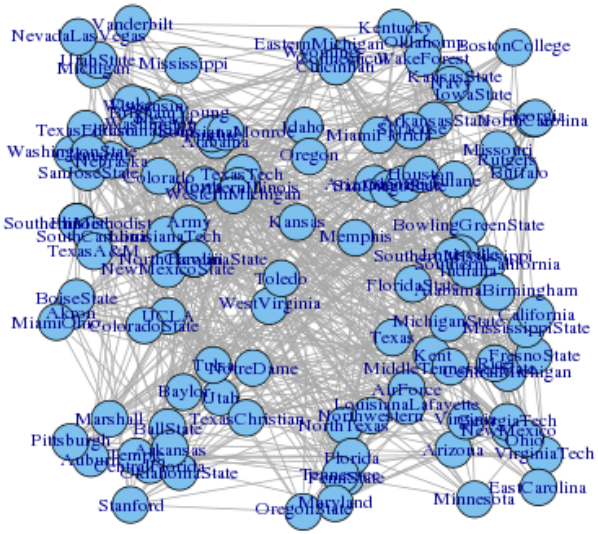
Zachary karate club



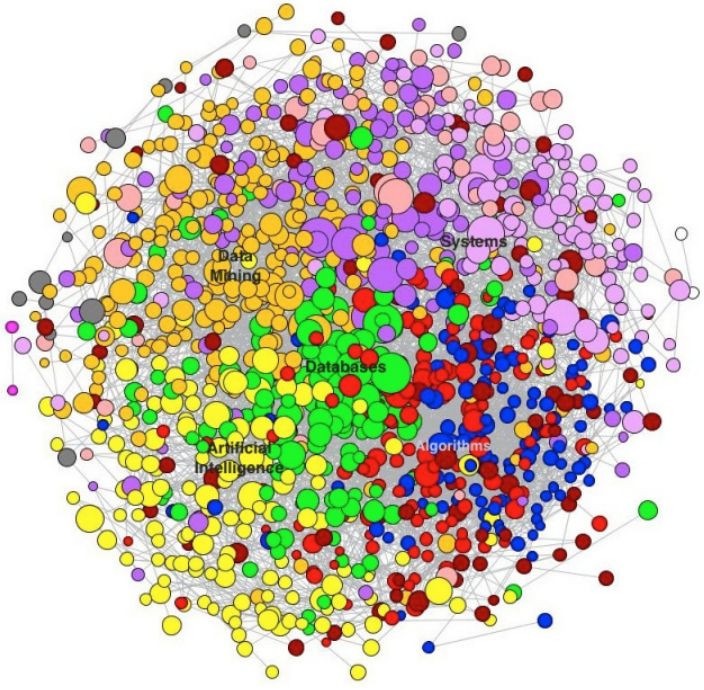
Dolphin social network

- **reference:** `http://www-personal.umich.edu/~mejn/netdata/`

NETWORKS WITH GROUND TRUTH (2/2)



American College Football



Researchers & conferences (DBLP)

- reference: `http://www-personal.umich.edu/~mejn/netdata/`

Evaluation without Ground Truth

Evaluation with Semantics:

- Attributes analysis: Labor markets such as Amazon Mechanical Turk platform (Chapter 9) can be used to analyze community members' attributes (posts, profile information, etc.).
- Word frequencies: by generating a list of frequent keywords for each community, human subjects determine whether these keywords represent a coherent topic.