

# Networks Structure and Dynamics

## 5. Community detection

Maximilien Danisch, Marwan Ghanem, Lionel Tabourier

LIP6 – CNRS and Université Pierre et Marie Curie

first\_name.last\_name@lip6.fr

October 16<sup>th</sup> 2018

## Community detection

**Goal:** Identify automatically **relevant groups** of nodes.

**Applications:**

- Understand the structure of a network
- Detect specific communities (web pages, proteins, ...)
- Help visualization
- Improvement information retrieval (search engines, recommendation, ...)

**Challenges:**

- Unknown number of communities
- Unknown sizes of communities
- Scalability

## Community detection

**Goal:** Identify automatically **relevant groups** of nodes.

**Applications:**

- Understand the structure of a network
- Detect specific communities (web pages, proteins, ...)
- Help visualization
- Improvement information retrieval (search engines, recommendation, ...)

**Challenges:**

- Unknown number of communities
- Unknown sizes of communities
- Scalability

## What is a community?

Set of nodes that **share something**:

- Affiliation (friends, colleagues, club, ...)
- Similar interests (tagging systems, ...)
- Similar contents (movies, books, products, web pages, ...)
- ...

What is the connexion with the network structure?

## What is a community?

Set of nodes that **share something**:

- Affiliation (friends, colleagues, club, ...)
- Similar interests (tagging systems, ...)
- Similar contents (movies, books, products, web pages, ...)
- ...

**What is the connexion with the network structure?**

## What is a community?

Set of nodes that **share something**:

- Affiliation (friends, colleagues, club, ...)
- Similar interests (tagging systems, ...)
- Similar contents (movies, books, products, web pages, ...)
- ...

**What is the connexion with the network structure?**

**More densely connected inside than outside**

## Outline

- 1 Find a single community
  - Structural approach
  - Optimization approach
- 2 Partition the graph into communities
  - Label propagation
  - Modularity and the Louvain algorithm
  - Divisive and agglomerative approaches
- 3 Overlapping communities

## Structural approaches: cohesive subgraphs

**clique**: complete subgraph

**$k$ -plex**: maximal subgraph  $H$  such that each node is connected to at least  $|H| - k$  other nodes (if  $k = 1$ : clique)

**$\alpha$ -set**: maximal subgraph such that any node has  $\alpha$  times more internal than external links

**Exercise**: suggest other relevant definitions.

## Optimization approach: quality function

**Quality function:** quantitatively evaluate the quality of a set of nodes as a community.

Local optimum of  $f(n, m, s, l_2, l_1)$ , with

- $s$ : number of nodes in the set
- $l_2$ : number of links with both end nodes in the set
- $l_1$ : number of links with exactly one node in the set

Examples:

- $\frac{l_2}{l_1 + l_2}$
- edit distance:  $\frac{s(s-1)}{2} - l_2 + l_1$

**Exercise:** suggest other relevant quality functions.

## Optimization approach: quality function

Local optimum of  $f(n, m, t, s, l_2, l_1, t_1, t_2, t_3)$ , with

- $t$ : number of triangles in the graph
- $t_3$ : number of triangles with three nodes in the set
- $t_2$ : number of triangles with exactly two nodes in the set
- $t_1$ : number of triangles with exactly one node in the set

Examples:

- triangle density:  $\frac{t_3}{s}$
- $C = \frac{t_3}{\binom{s}{3}} \times \frac{t_3}{t_3 + t_2}$  Triangles to Capture Social Cohesion - Friggeri et al.

**Exercise:** suggest other relevant quality functions.

## Optimization approach: Optimization

Use a simple greedy or stochastic approach:

- **Initialization:** start from a set containing only one node
- **Optimization:** at each iteration, add a randomly chosen node, neighbor of the set, that increases the quality  $f$
- **Stop:** when the quality function can no longer be increased

Can be done efficiently by updating  $s$ ,  $l_1$  and  $l_2$  locally:

<https://github.com/maxdan94/mocda>

## Outline

- 1 Find a single community
  - Structural approach
  - Optimization approach
- 2 Partition the graph into communities
  - Label propagation
  - Modularity and the Louvain algorithm
  - Divisive and agglomerative approaches
- 3 Overlapping communities

## A simple and fast algorithm: Label propagation

Near linear time algorithm to detect community structures in large-scale networks -  
*Raghavan et al.*

- **Step 1:** give a unique label to each node in the network
- **Step 2:** Arrange the nodes in the network in a random order
- **Step 3:** for each node in the network (in this random order) set its label to a label occurring with the highest frequency among its neighbours
- **Step 4:** go to 2 as long as there exists a node with a label that does not have the highest frequency among its neighbours.

To shuffle in a clean way:

[https://en.wikipedia.org/wiki/Fisher-Yates\\_shuffle](https://en.wikipedia.org/wiki/Fisher-Yates_shuffle)

## A simple and fast algorithm: Label propagation

**Exercise:** why such an algorithm should lead to relevant groups?

**Exercise:** Which data structure should be used to implement this algorithm efficiently?

## A simple and fast algorithm: Label propagation

**Exercise:** why such an algorithm should lead to relevant groups?

- Densely connected groups should reach a common label.
- When such a consensus group is created it should expand until being stopped by other equivalent consensus groups.

**Exercise:** Which data structure should be used to implement this algorithm efficiently?

## Community structure

### Structural definitions

- A **community** is a set of nodes that are more connected among themselves than to the rest of the network
- **Modularity** is a measure to evaluate the quality of a community partitioning of a graph (**one among others**)

### What is modularity (intuitively)?

The difference between:

- the **number of links in a group**
- and the **expected number of links in the same group of a comparable random graph**

## Community structure

### Structural definitions

- A **community** is a set of nodes that are more connected among themselves than to the rest of the network
- **Modularity** is a measure to evaluate the quality of a community partitioning of a graph (**one among others**)

### What is modularity (intuitively)?

The difference between:

- the **number of links in a group**
- and the **expected number of links in the same group of a comparable random graph**

## Modularity definition: preliminary remarks

**Graph  $G$**  :  $m$  links,  $n$  nodes

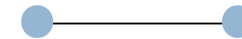
**Group  $S$**  : sum of degree  $d_s$  , number of internal links  $m_s$

**In a random graph with fixed degree distribution:**

probability for one end of a link to be in  $S$ :

⇒ probability for a link to be in  $S$ :

⇒ expected number of links in  $S$ :



## Modularity definition: preliminary remarks

**Graph  $G$**  :  $m$  links,  $n$  nodes

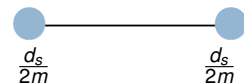
**Group  $S$**  : sum of degree  $d_s$  , number of internal links  $m_s$

**In a random graph with fixed degree distribution:**

probability for one end of a link to be in  $S$ :  $\frac{d_s}{2m}$

⇒ probability for a link to be in  $S$ :  $\frac{d_s}{2m} \cdot \frac{d_s}{2m}$

⇒ expected number of links in  $S$ :  $m \cdot \frac{d_s}{2m} \cdot \frac{d_s}{2m} = \frac{d_s^2}{4m}$



## Modularity definition

$$Q = \frac{1}{m} \sum_{s=1}^K \left( m_s - \frac{d_s^2}{4m} \right)$$

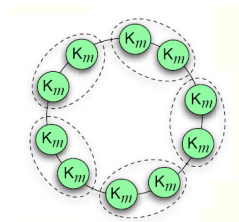
- $m_s$ : number of internal links in group  $s$
- $m$ : number of links in the graph

## Known problem: resolution limit

Ring of cliques:  $\alpha$  cliques of size  $\beta$

$$Q_{single} = 1 - \frac{2}{\beta(\beta - 1) + 2} - \frac{1}{\alpha}$$

$$Q_{pairs} = 1 - \frac{1}{\beta(\beta - 1) + 2} - \frac{2}{\alpha}$$



## Known problem: resolution limit

**Ring of cliques:**  $\alpha$  cliques of size  $\beta$

$$Q_{single} > Q_{pairs} \iff \beta(\beta - 1) + 2 > \alpha$$

Suppose 30 cliques of size 5 then:

- $\alpha = 30$  and  $\beta(\beta - 1) + 2 = 22 \Rightarrow Q_{single} < Q_{pairs}$
- $Q_{single} = 0.876$ ,  $Q_{pairs} = 0.888$

**counter-intuitive**

Tendency to favour large communities...  
... may appear at any length scale

## Greedy and efficient optimization of Modularity

- **Step 1.** Initialization: node = community
- **Step 2.** Remove node  $u$  from its community
- **Step 3.** Insert node  $u$  in a neighboring community that maximizes  $Q$
- **Step 4.** Iterate from step 1 until the partition does not evolve

## Greedy and efficient optimization of Modularity

- **Step 1.** Initialization: node = community
- **Step 2.** Remove node  $u$  from its community
- **Step 3.** Insert node  $u$  in a neighboring community that maximizes  $Q$
- **Step 4.** Iterate from step 1 until the partition does not evolve

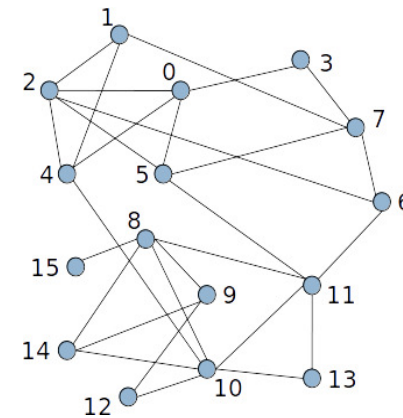
**Can be trapped in bad local minima**

## The Louvain algorithm

- **Step 1.** Initialization: node = community
- **Step 2.** Remove node  $u$  from its community
- **Step 3.** Insert node  $u$  in a neighboring community that maximizes  $Q$
- **Step 4.** Iterate from step 1 until the partition does not evolve
- **Step 5.** Transform the communities into (hyper-)nodes and go back to step 1 with the new graph

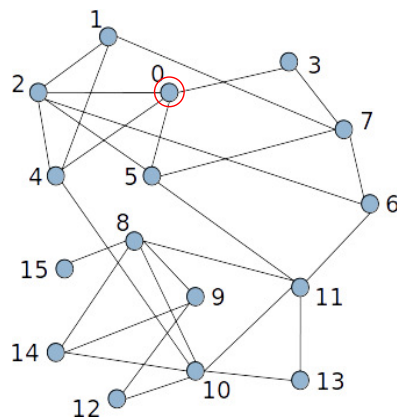
Leads to better local optima

## Example



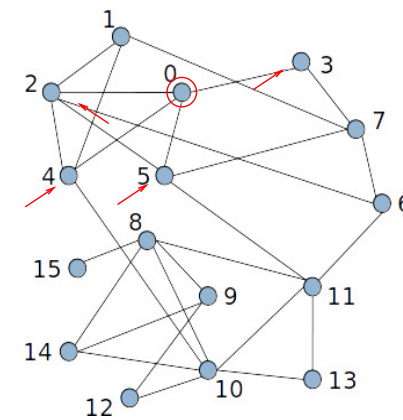
First passage, first iteration: isolated nodes

## Example



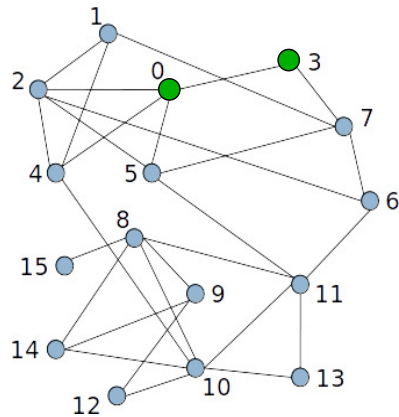
considering 0...

## Example



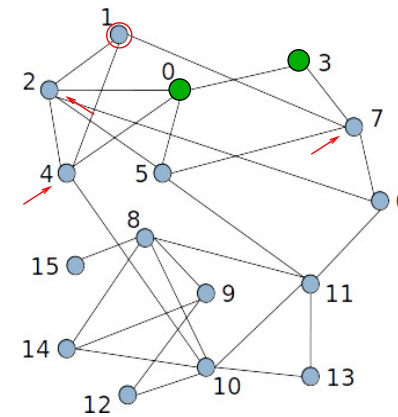
its neighboring communities are...

## Example



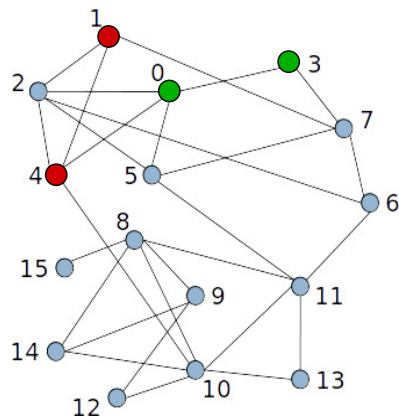
0 is put in  $C(3)$ , best  $Q$  increase

## Example



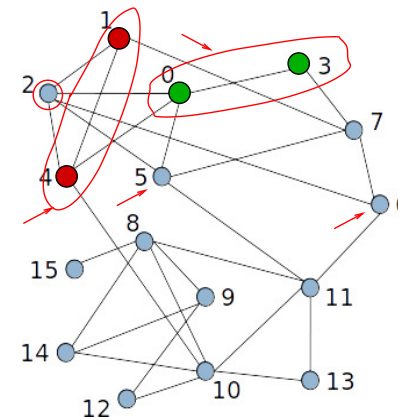
considering 1, its neighboring communities are...

## Example



1 is put in  $C(4)$ , best  $Q$  increase

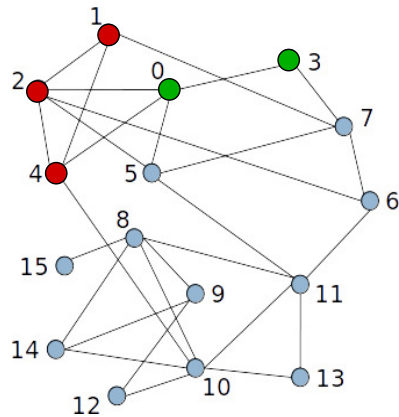
## Example



considering 2, its neighboring communities are...

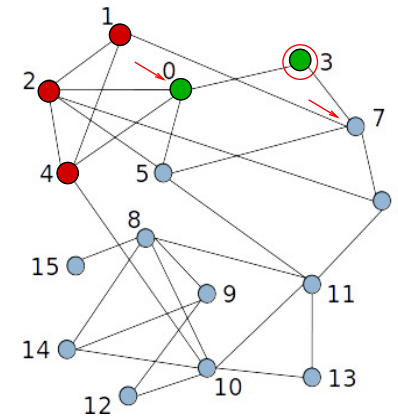


## Example



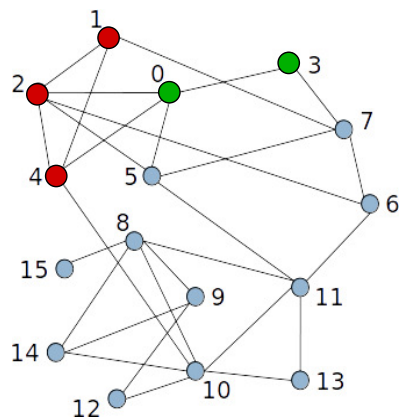
2 is put in  $C(1,4)$ , best Q increase

## Example



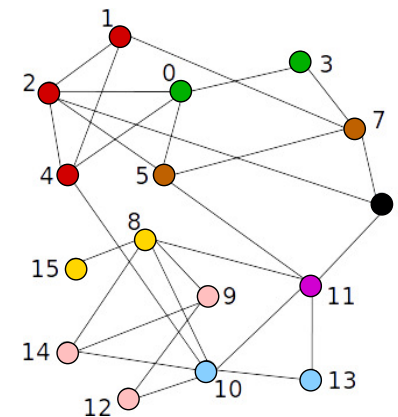
considering 3, its neighboring communities are...

## Example



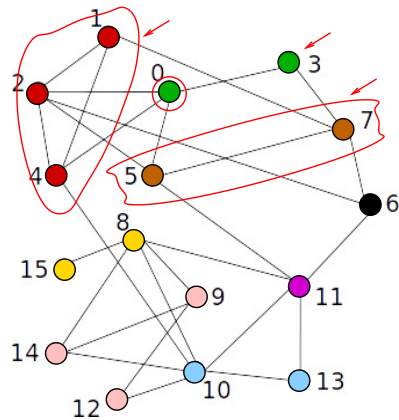
3 stays in the same community  $C(0,3)$ , otherwise Q decreases

## Example



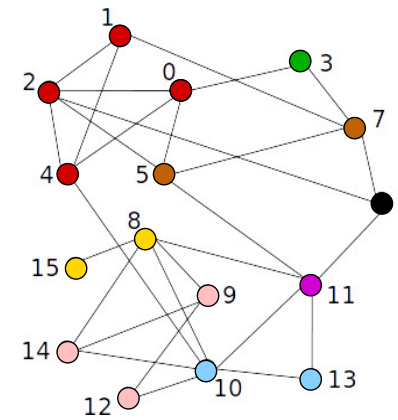
and so on...

## Example



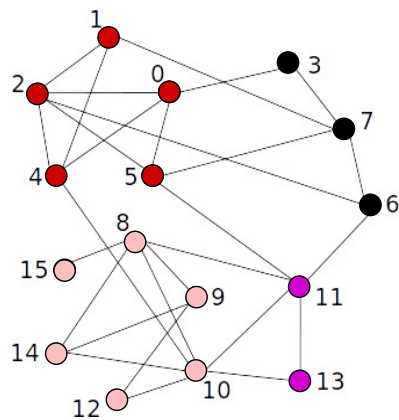
First passage, second iteration: considering 0...

## Example



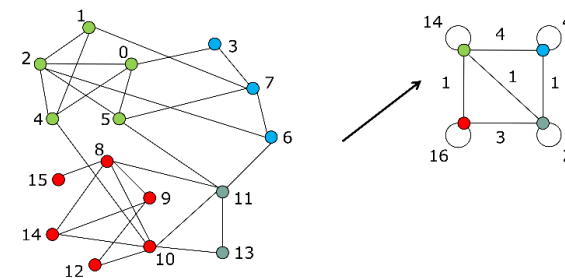
0 is put in  $C(1,2,4)$ , best Q increase

## Example



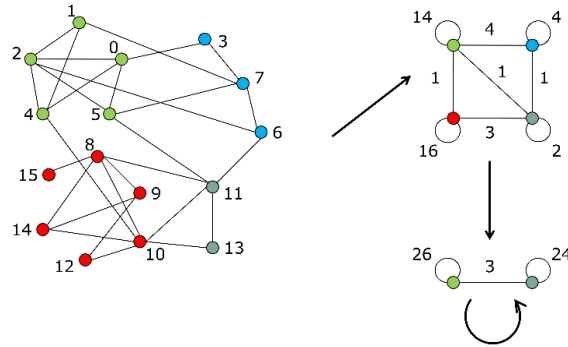
after 4 iterations, no change anymore

## Example



Second passage

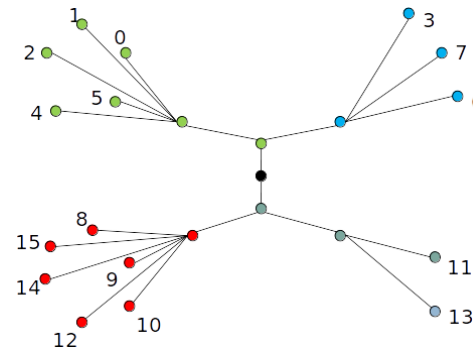
## Example



Third passage

## Example

### Outcome: non-binary dendrogram



## Other experimental observations

- Graphs on the highest levels of the dendrogram are small, **the first passages are the most expensive** practically, the first passage is more than 90% of the time
- **Few iterations per passage** (less than < 33 for all the networks tested)
- **Processing a node is simple** (cheap)

## Modularity

$$Q = \frac{1}{m} \sum_s m_s - \frac{d_s^2}{4m} = \sum_s \frac{m_s}{m} - \left( \frac{d_s}{2m} \right)^2$$

$m_s$ : links  $\in S$

$d_s$ : sum of the degrees of nodes in  $S$

Note that the contribution of an isolated node is then:

$$Q(i) = - \left( \frac{k_i}{2m} \right)^2$$

with  $k_i$ : degree of  $i$

$\Rightarrow$  always merged with a neighboring community

## Modularity

$$Q = \frac{1}{m} \sum_s m_s - \frac{d_s^2}{4m} = \sum_s \frac{m_s}{m} - \left( \frac{d_s}{2m} \right)^2$$

$m_s$ : links  $\in S$

$d_s$ : sum of the degrees of nodes in  $S$

Note that the contribution of an isolated node is then:

$$Q(i) = - \left( \frac{k_i}{2m} \right)^2$$

with  $k_i$ : degree of  $i$

$\Rightarrow$  always merged with a neighboring community

## The cost of moving one node

An isolated node  $i$  may be moved to  $S$  with a gain of:

$$\Delta Q(S, i) = \left[ \frac{m_s}{m} + \frac{k_{i,s}}{m} - \left( \frac{d_s + k_i}{2m} \right)^2 \right] - \left[ \frac{m_s}{m} - \left( \frac{d_s}{2m} \right)^2 - \left( \frac{k_i}{2m} \right)^2 \right]$$

$k_{i,s}$ : links from  $i$  to  $S$

Only depends on  $S$  and  $i$ , linear complexity with  $k_i$

## The cost of moving one node

An isolated node  $i$  may be moved to  $S$  with a gain of:

$$\Delta Q(S, i) = \left[ \frac{m_s}{m} + \frac{k_{i,s}}{m} - \left( \frac{d_s + k_i}{2m} \right)^2 \right] - \left[ \frac{m_s}{m} - \left( \frac{d_s}{2m} \right)^2 - \left( \frac{k_i}{2m} \right)^2 \right]$$

$k_{i,s}$ : links from  $i$  to  $S$

Only depends on  $S$  and  $i$ , linear complexity with  $k_i$

## Data structures

We have to keep in memory:

- the adjacency lists: size  $(2m + n)$
- vectors  $m_s$ ,  $d_s$  and  $node2comm$  (stores  $k_{i,s}$ ): size  $n$  each

A total of  $2m + 4n$ , meaning a few GB for a billion links graph

## Conclusion

Fast unfolding of communities in large networks - *Blondel et al, 2008*

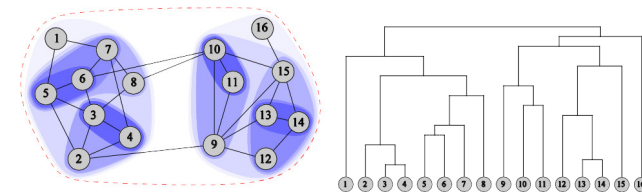
- What kind of approach is it?
  - **greedy local** approach
  - **modularity-based**
- **Good results** in terms of modularity
- **Quasi linear** complexity  
⇒ allow to process very large graphs
- **Non-deterministic** algorithm

## Divisive and agglomerative approaches

Many other algorithms can be found in the literature:

*Community detection in graphs - Fortunato, 2010*

A large amount of them can be seen as **divisive** or **agglomerative** approaches.



## Agglomerative approach

- **Step 1:** Each node is in a community (initialization)
- **Step 2:** Compute the **similarity** between each pair of communities
- **Step 3:** Merge the two closest communities
- **Step 4:** Iterate from step 1

**Exercise:** Suggest a relevant similarity metric between two communities

## Divisive approaches

- **Step 1:** All nodes are in a unique community (initialization)
- **Step 2:** Compute a **strength score** for each link
- **Step 3:** Delete the weakest link
- **Step 4:** Iterate from step 1

**Exercise:** Suggest a relevant strength score for a link

## Outline

- 1 Find a single community
  - Structural approach
  - Optimization approach
- 2 Partition the graph into communities
  - Label propagation
  - Modularity and the Louvain algorithm
  - Divisive and agglomerative approaches
- 3 Overlapping communities

## Many algorithms

Again there is a plethora of algorithms for finding overlapping communities:

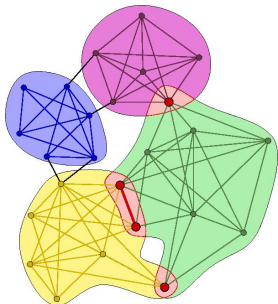
[Overlapping Community Detection in Networks: The State-of-the-art and Comparative Study - Xie et al. 2013](#)

We just show one which was among the first to do the job.

## k-clique percolation method

**Definition:** Two  $k$ -cliques are considered adjacent if they share  $k - 1$  nodes.

**Definition:** A community is defined as the maximal union of  $k$ -cliques that can be reached from each other through a series of adjacent  $k$ -cliques.



**Exercise:** how can we find all “communities” efficiently for  $k = 3$ ?