Coordinated Behaviour

Lorenzo Cima

lorenzo.cima@phd.unipi.it; lorenzo.cima@iit.cnr.it

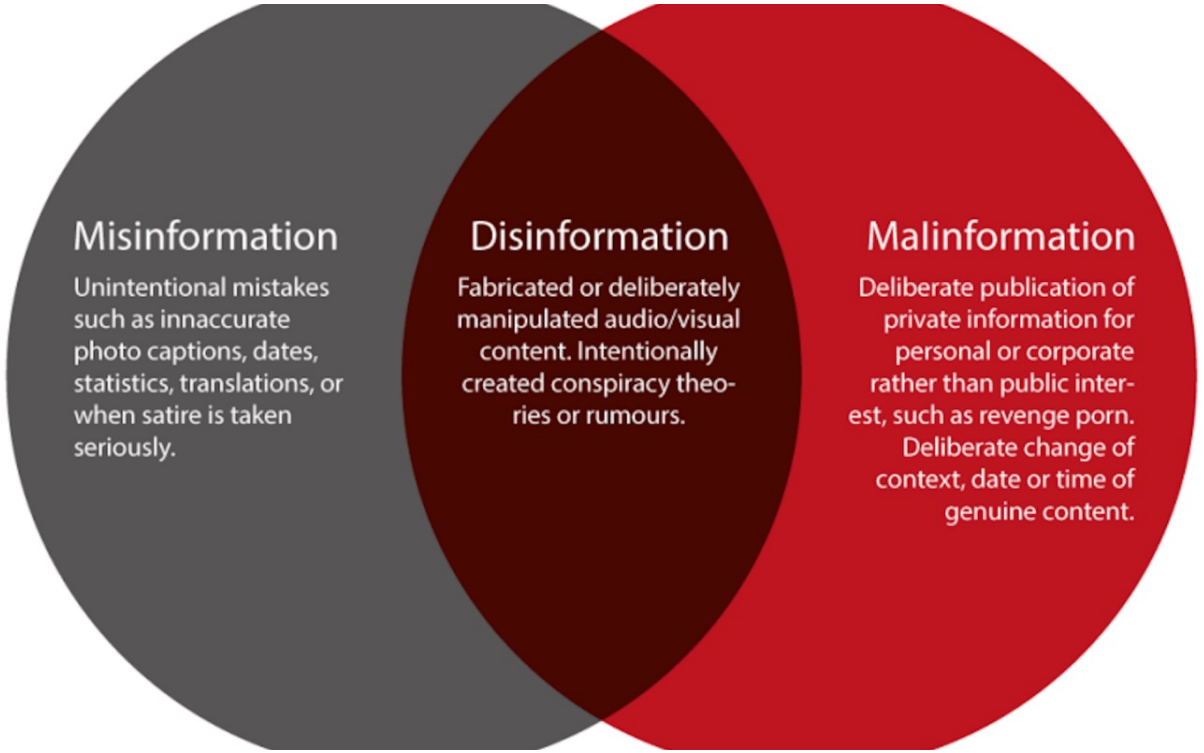
INTRODUCTION

Coordinated Online Behaviours

> *A group of (two or more) accounts,*
> *who perform synergic actions on one or more social media platforms,*
> *in order to pursue an intent.*

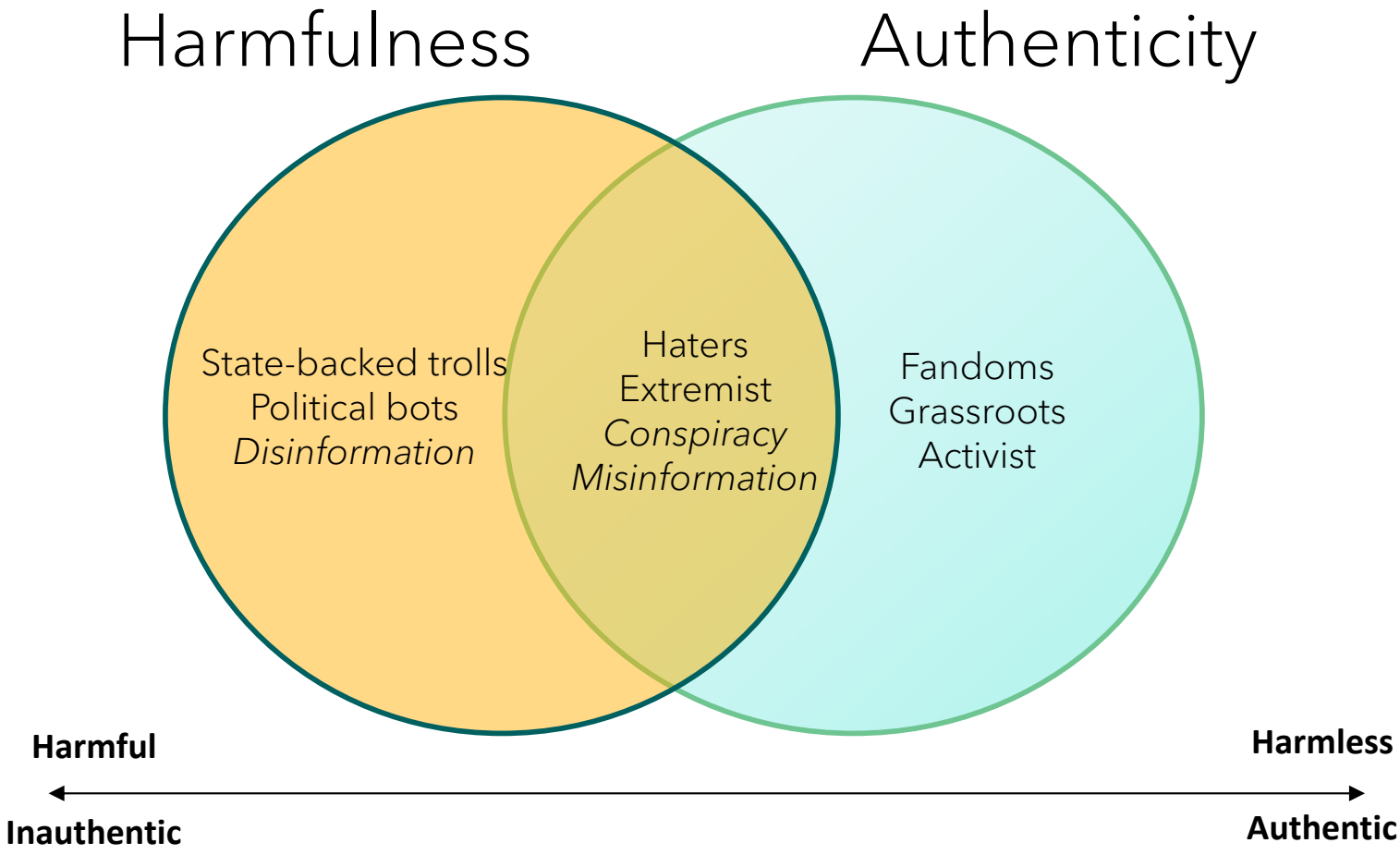
Information Disorder

- misinformation (involuntarily fake or bad)
- malinformation (possibly true, but with malicious intent)
- disinformation (both fake and maliciously so)
- fake news (information at least partially untrue)



Coordinated Behaviours

- Not all the forms of online coordination are bad!



Inauthenticity

Inauthentic behaviour

- Involves the use of **fake accounts**, such as social bots and state-backed troll
- The purpose is to **mislead** others

Authentic behaviour

- Is carried out by **legitimate accounts** and usually **arises naturally** from a community of users
- The purpose is to **inform** the greatest number of people about **common interests or beliefs**

Harmfulness

Harmful behaviour

- Can have **serious consequences** for individuals, groups, and society
- The concept of harmfulness depends largely on the viewpoint of the observer

Harmless behaviour

- Can be **beneficial** to online communities
- Users collaborate to share information, resources, or social support

Organization

Organized behaviour

- Typically **planned and directed** by a centralized authority interested in making the operation successful, such as a political party or a state-sponsored agency

Organic behaviour

- emerges **spontaneously** from multiple actors working toward a shared goal

Coordinated Inauthentic Behaviours (CIB)

"Groups of pages or people working together to mislead others about who they are or what they are doing"

6th December 2018

Coordinated Inauthentic Behaviours (CIB)

"Groups of pages or people working together
to mislead others
about who they are or what they are doing"

coordination ≠ **inauthenticity**

coordination ≠ **harmfulness**

coordination ≠ **automation** (social bots)

coordination ≠ **binary** concept

Coordinated Inauthentic Behaviours (CIB)

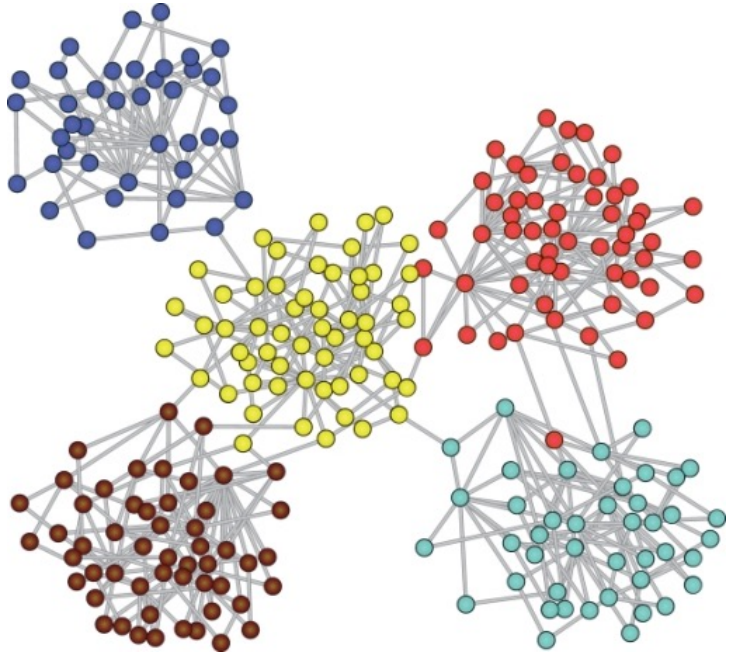
"Groups of pages or people working together
to mislead others
about who they are or what they are doing"

- There is a **shift** of focus **from content to behaviour**. It is not important if the content shared by the users is false or real
- **Inauthenticity** and **coordination** are shown as two **linked** concepts
- CIB does not coincide with botnets. **Inauthentic users are not only automatic accounts** (bots)
- Coordinated behaviours **evolve and vary over time**


COMMUNITY
DISCOVERY

Community Discovery

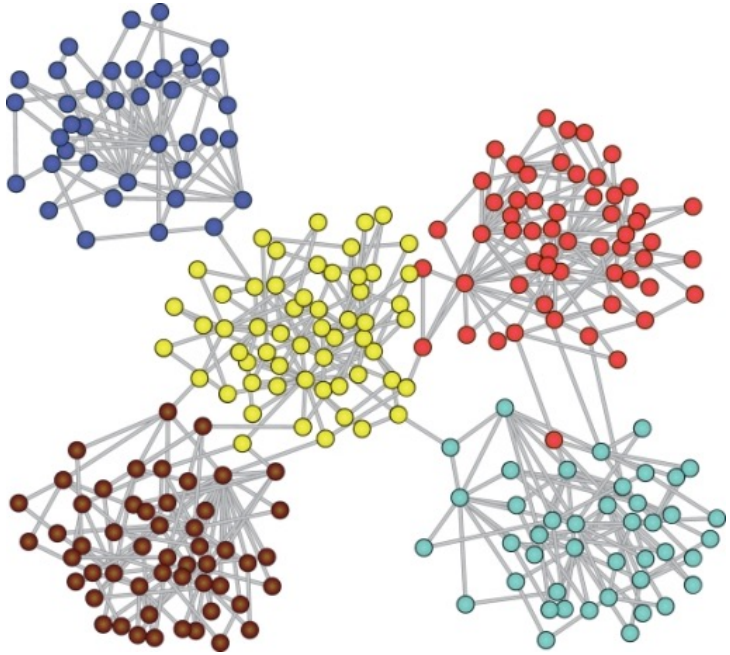
- Community Discovery algorithms **identify mesoscale topologies** hidden within complex network structures
- Each algorithm models **different properties** of communities, providing different results
- A universal definition of community does not exist



Community Discovery

"A set of entities where each entity is closer, in the network sense, to the other entities within the community than to the entities outside it"

"A set of nodes more tightly connected within each other than with nodes belonging to other sets"

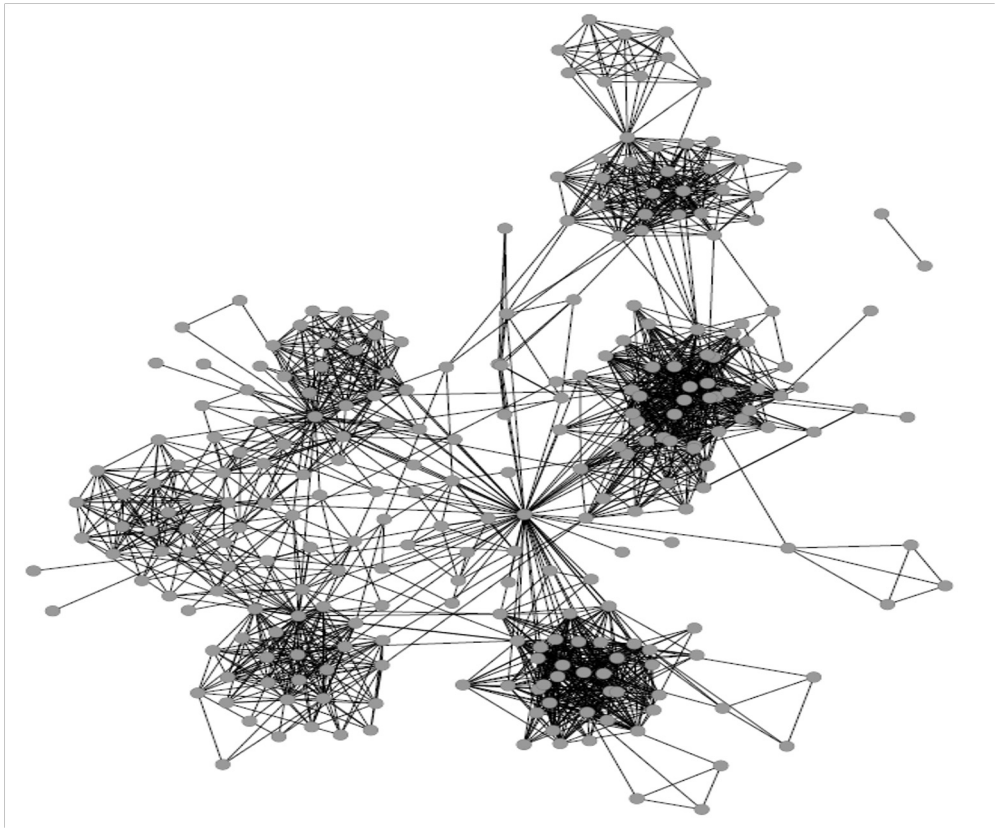


Community Discovery

- In simple, small, networks it is easy to identify communities by looking at the structure…

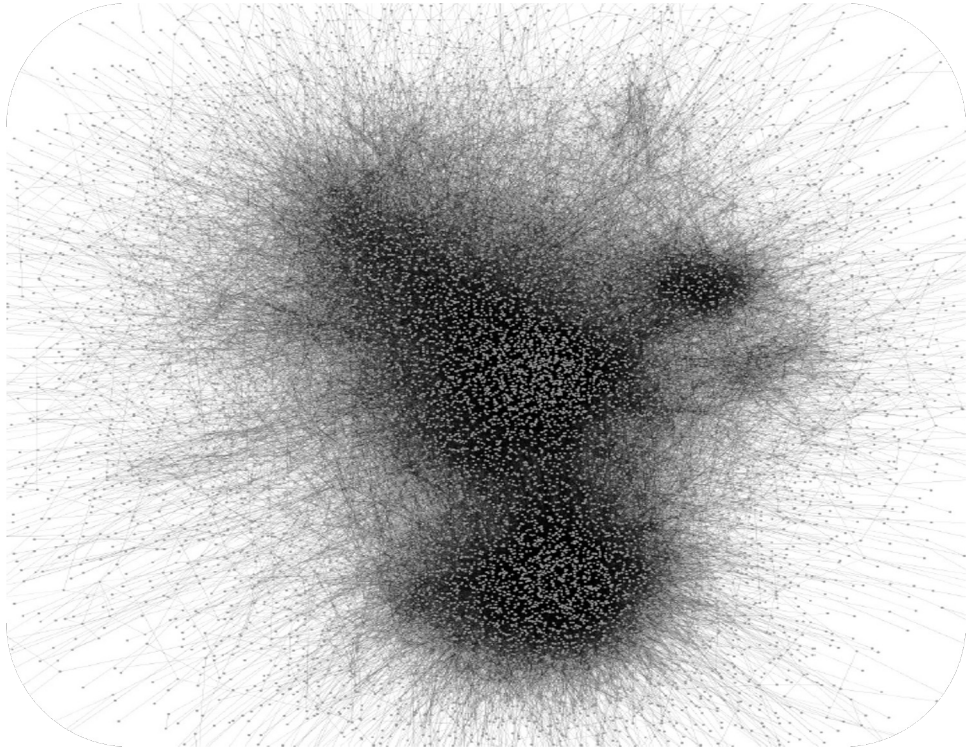
- Visual discovery using a Force-directed layout

Gephi



Community Discovery

- In real-world networks is impossible to visually identify the different communities

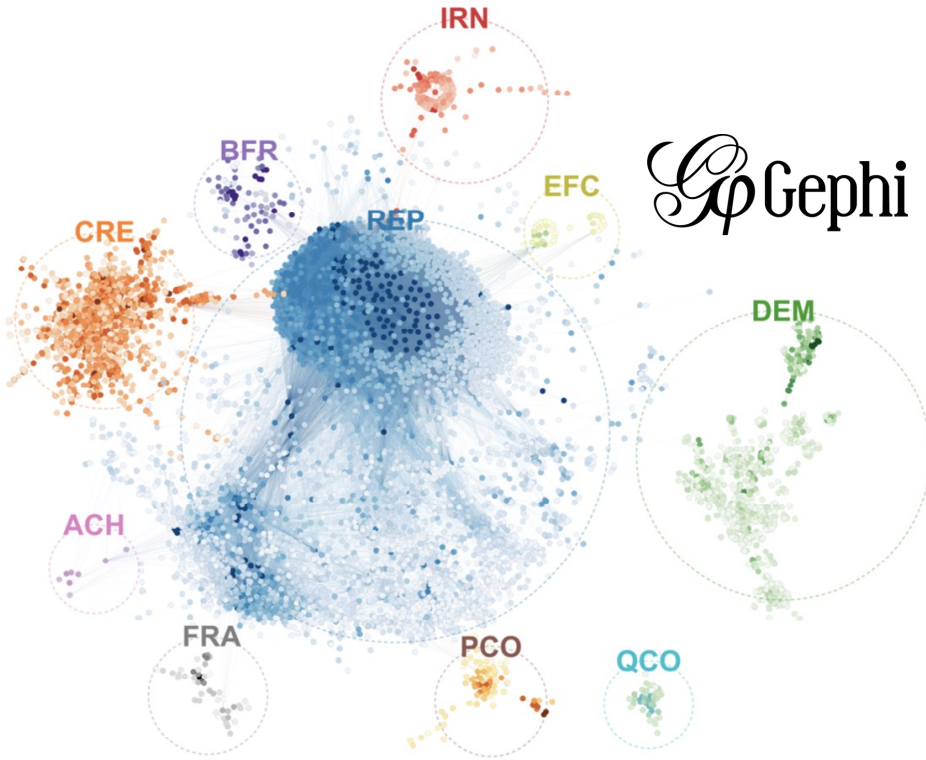


- We need automated procedures!

Community Discovery

- The most powerful way to analyze communities is to combine visual and automated techniques
- *Ex. 2020 USA Presidential Election*

  - ***DEM*** *and* ***REP****: main communities of democrats and republicans*
  - ***CRE****,* ***PCO****,* ***QCO*** *and* ***EFC****: supporters of multiple conspiracy theories (pandemic, QAnon, election fraud)*
  - ***IRN****,* ***BFR****,* ***FRA*** *and* ***ACH****: foreign Trump supporters*



Gephi

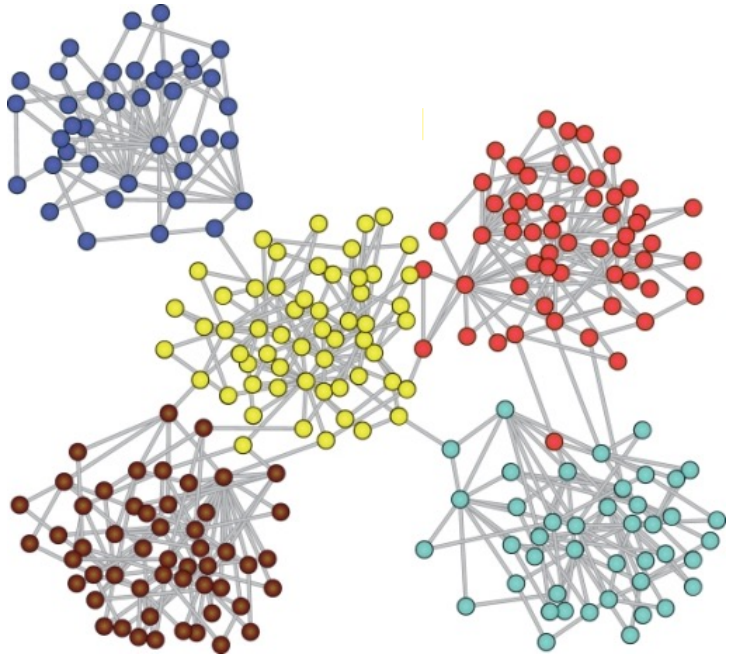
python

Algorithms Taxonomy

Community discovery algorithms can be classified according to:

- the constraints they impose on the mesoscale structures they are searching for
- the way they approach the community retrieval

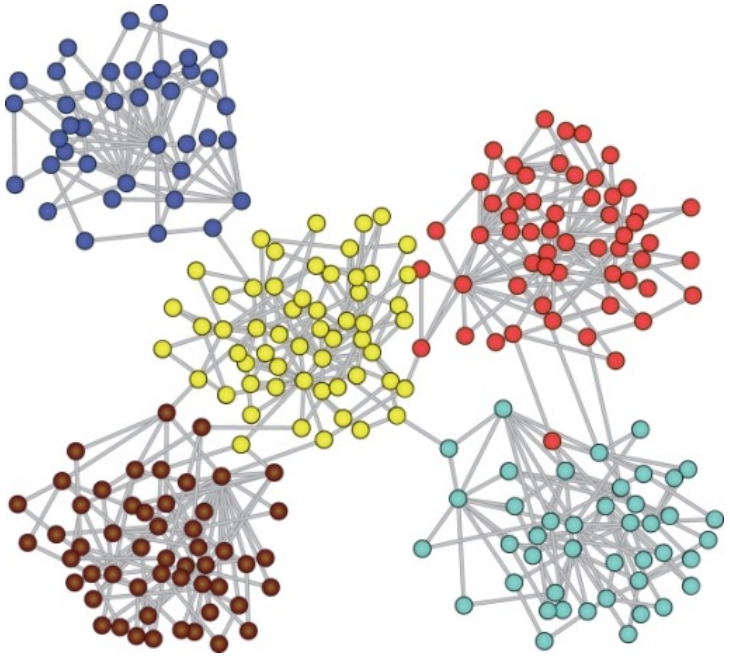
We will explore only internal density algorithms



Internal Density	Bridge Detection
Feature Distance	Percolation
Entity Closeness	Structure Definition
Link Communities	No a priori definition

Internal Density Algorithms

- **Communities**: sets of densely connected entities
- Each community must have a **number of edges** significantly **higher** than what is expected in a random graph
- **In weighted graphs, computations should account for weights**
- Greedy Modularity, Louvain, Leiden…



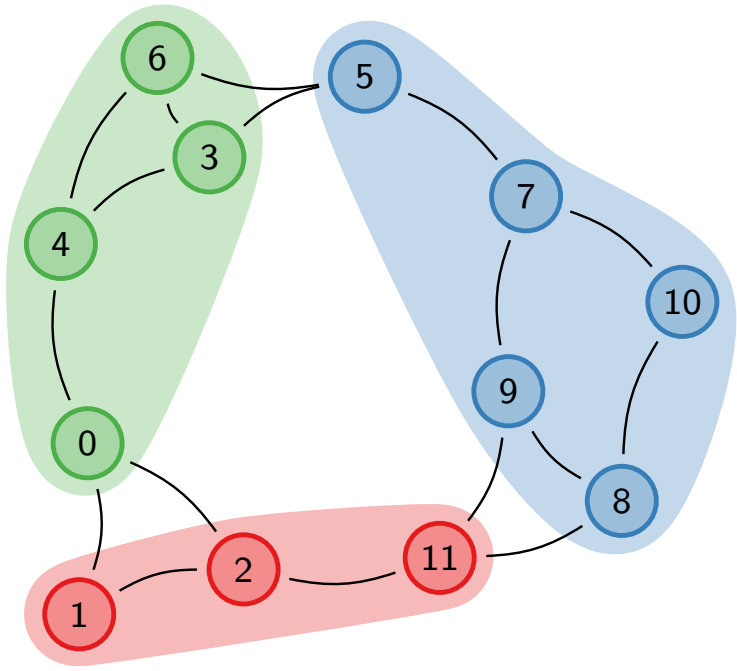
Internal Density	Bridge Detection
Feature Distance	Percolation
Entity Closeness	Structure Definition
Link Communities	No a priori definition

Simple Reward Function

- Get $1 - \gamma$ for every edge inthe cluster
- Pay γ for every missed edge
- The optimal solution has the highest reward

H: reward **e_c: edges in the cluster** **n_c: nodes in the cluster**

$$\mathcal{H} = \sum_c \left(e_c - \gamma \binom{n_c}{2} \right)$$



Suppose γ = 0,5
What is the reward of this cluster?
And if γ is 0,3 or 0,8?

Simple Reward Function

- Get $1 - \gamma$ for every edge in cluster
- Pay γ for every missed edge

$$\mathcal{H} = \sum_c \left(e_c - \gamma \binom{n_c}{2} \right)$$

$$\gamma = 0{,}5$$

$n_c = 4$

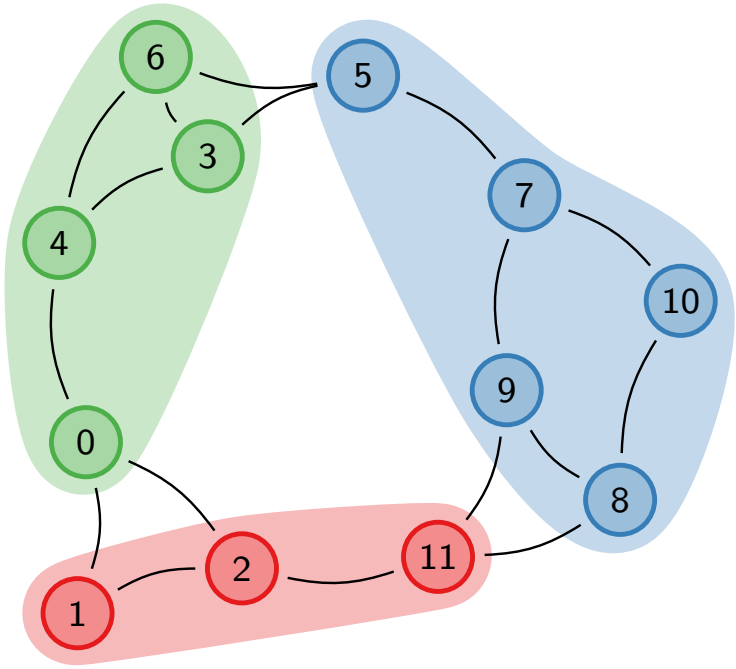
Expected edges = 4x3 /2 = 6

$n_c = 3$

Expected edges = 3x2 /2 = 3

$n_c = 5$

Expected edges = 5x4 /2 = 10

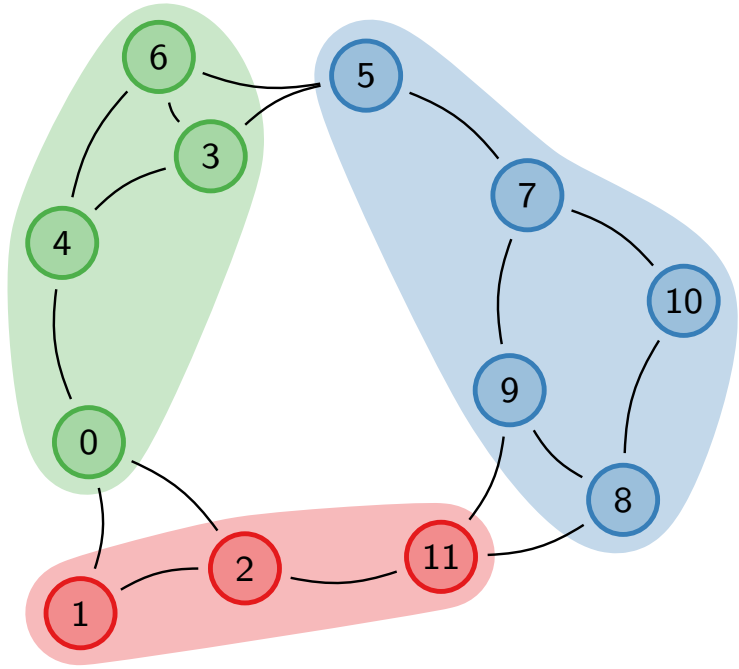


Simple Reward Function

- Get $1 - \gamma$ for every edge in cluster
- Pay γ for every missed edge

$$\mathcal{H} = \sum_c \left(e_c - \gamma \binom{n_c}{2} \right)$$

$$\gamma = 0{,}5$$



4 edges, 2 missing

4x0,5 - 2x0,5 = 1

2 edges, 1 missing

2x0,5 - 1x0,5 = 0,5

5 edges, 5 missing

5x0,5 - 5x0,5 = 0

H = 1 + 0,5 + 0 = 1,5

Simple Reward Function

$$\mathcal{H} = \sum_{c} \left(e_c - \gamma \binom{n_c}{2} \right)$$

$\gamma = 0{,}3 \mid 0{,}8$

$\gamma = 0{,}3$: *4x0,7 - 2x0,3 = 2,2*

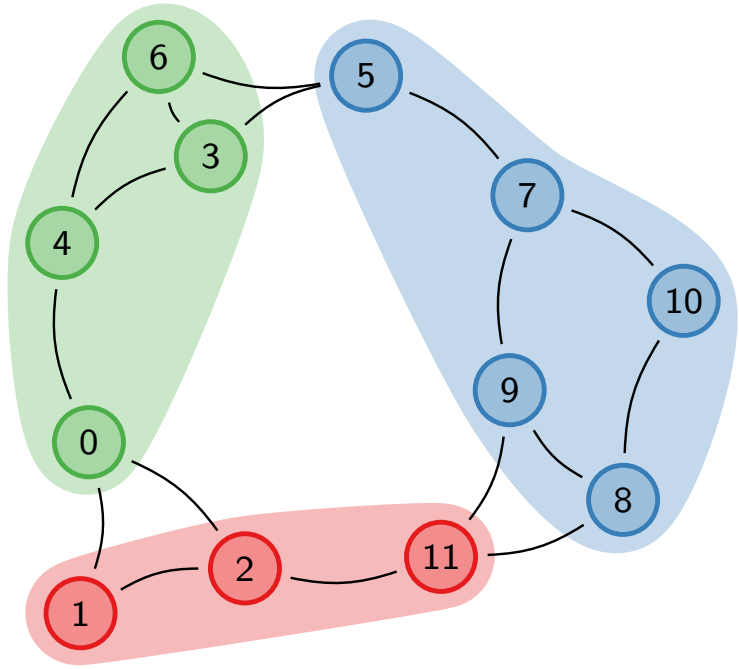
$\gamma = 0{,}8$: *4x0,2 - 2x0,8 = -0,8*

$\gamma = 0{,}3$: *2x0,7 - 1x0,3 = 1,1*

$\gamma = 0{,}8$: *2x0,2 - 1x0,8 = -0,4*

$\gamma = 0{,}3$: *5x0,7 - 5x0,3 = 2*

$\gamma = 0{,}8$: *5x0,2 - 5x0,8 = -3*

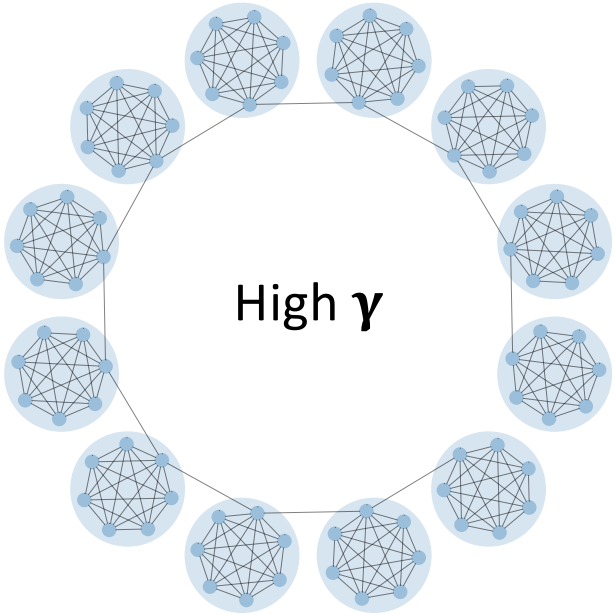


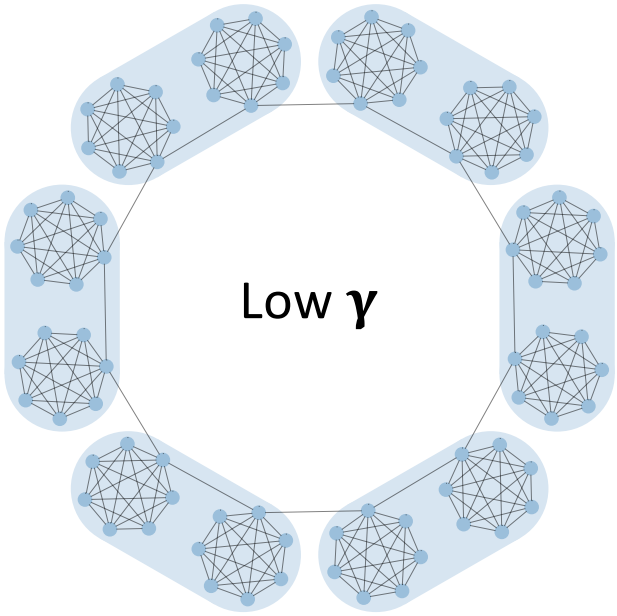
$\gamma = 0{,}3$: *H = 2,2 + 1,1 + 2 = 5,3*

$\gamma = 0{,}8$: *H = -0,8 -0,4 -3 = -4,2*

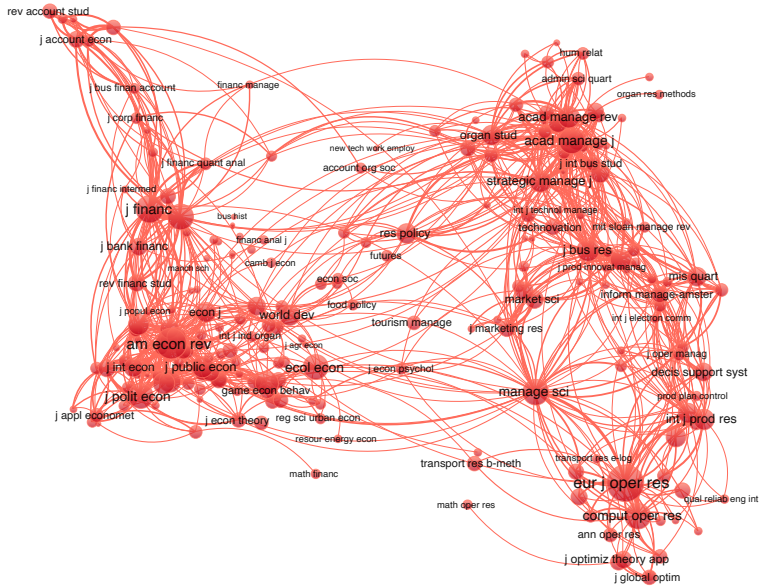
Resolution Limit

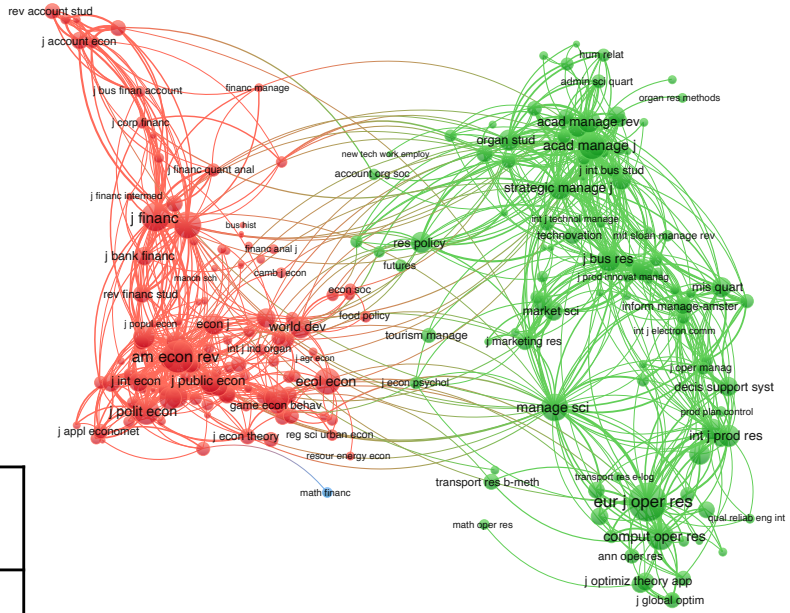
- Algorithms depend on the resolution limit γ
- Only local methods do not suffer resolution limits
- **High γ** => The optimal solution has **many small communities**
- **Low γ** => The optimal solution has **few large communities**



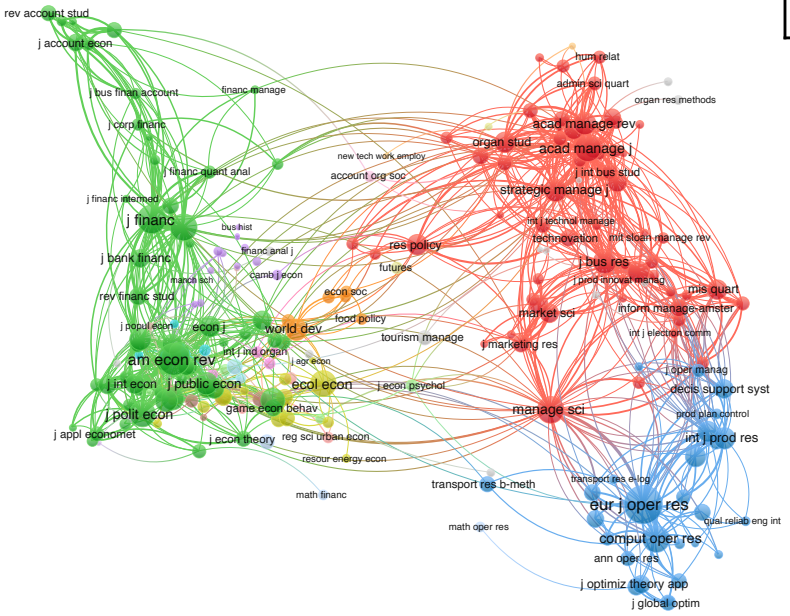


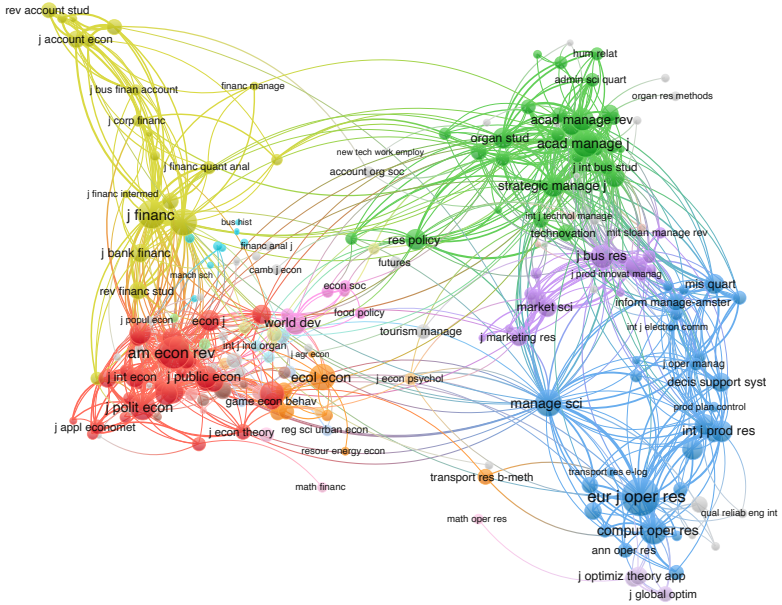
Resolution Limit





γ = 0,01	**γ = 0,1**
γ = 0,5	**γ = 1**



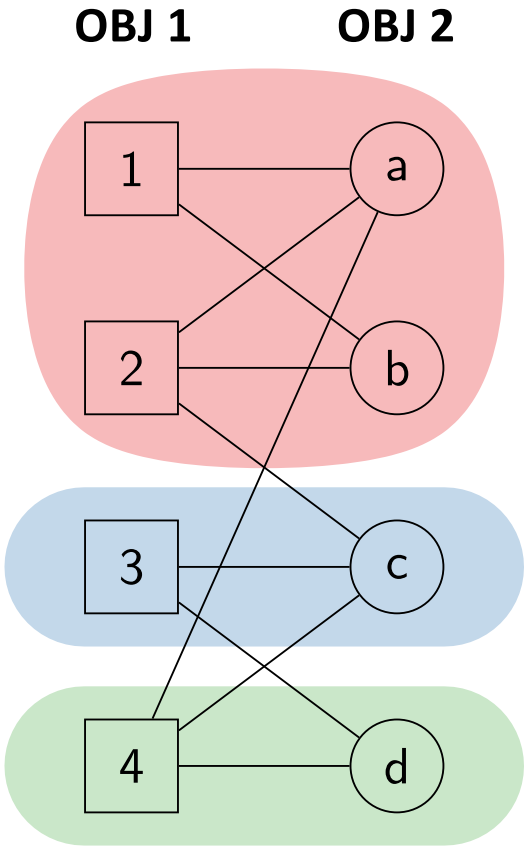


Simple Reward Function

- What about **bipartite** networks?
- The binomial coefficient is substituted by the product of cluster nodes

$$\mathcal{H} = \sum_c \left(e_c - \gamma \binom{n_c}{2} \right)$$

$n_c^{obj1} \times n_c^{obj2}$



Modularity Function

- A quality function that measures the density of a community
- The optimal solution has the highest modularity
- Modularity algorithm has a **greedy** approach
- Q ∈ [-1; 1]

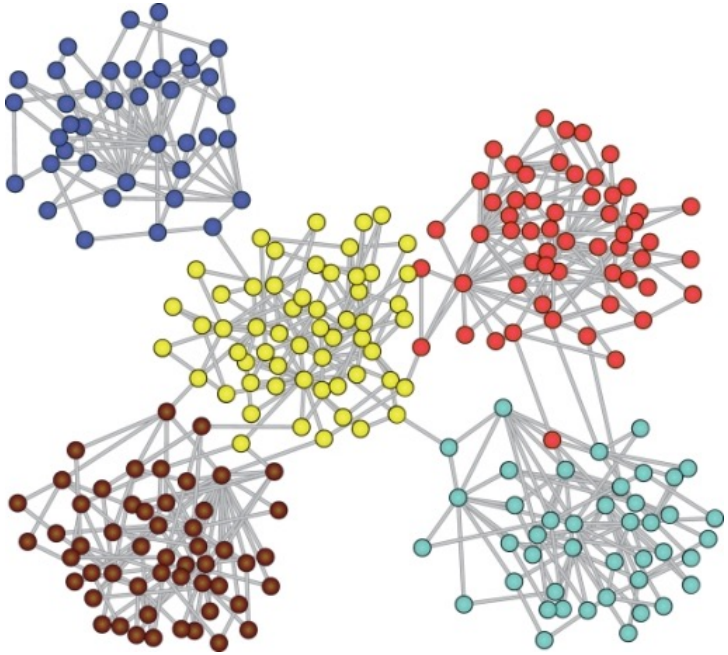
DIRECTED GRAPHS

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

Normalization Factor. m: number of edges

Null Model expected density ($k_i = k_j$ if undirected)

1 if i,j in same community, 0 otherwise

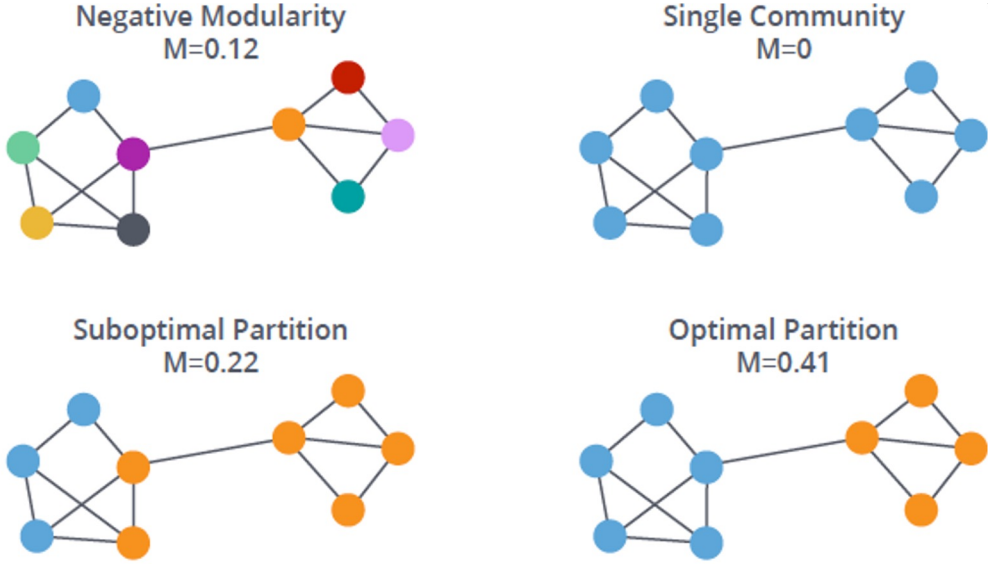


Modularity Function

- A quality function that measures the density of a community
- The optimal solution has the highest modularity
- The modularity algorithm has a **greedy** approach
- Q ∈ [-1; 1]

UNDIRECTED GRAPHS

$$\mathcal{Q} = \frac{1}{m}\sum_{c}\left(e_c - \frac{1}{2}\frac{K_c^2}{2m}\right)$$



Louvain Algorithm

- **Initialization**: Each node in the network is assigned to its own community

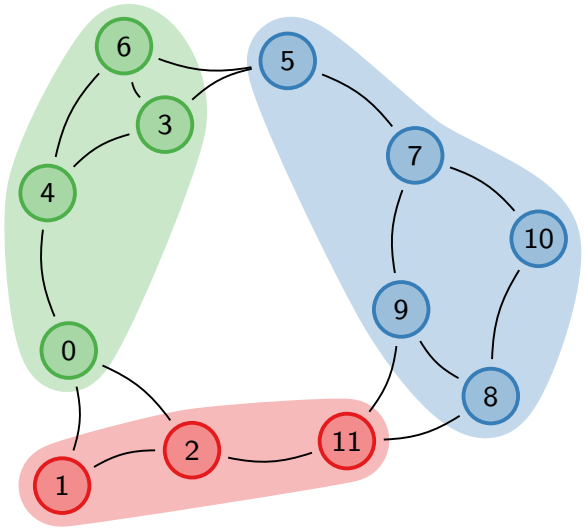
- **Phase 1 – Moving nodes**: Each node is **moved** one by one into the **adjacent community** which guarantees the **greatest modularity increment**, **if exists**
- Repeat phase 1 until no node is moved
- **Phase 2 - Aggregation**: A new graph is created: its nodes are the updated communities and weighted links connect them accounting for bridges in the original graph

- Phases 1 and 2 are repeated until modularity is maximized

Louvain Algorithm

- **Phase 1**: Each node is **moved** one by one into the **adjacent community** which guarantees the **greatest modularity increment**, **if exists**
- Repeat phase 1 until no node is moved
- **Phase 2 - Aggregation**: A new graph is created: its nodes are the updated communities and weighted links connect them accounting for bridges in the original graph

Supposing to use as reward the simple reward function with $\gamma = 0,5$ instead of modularity, what are the Louvain communities on this graph?



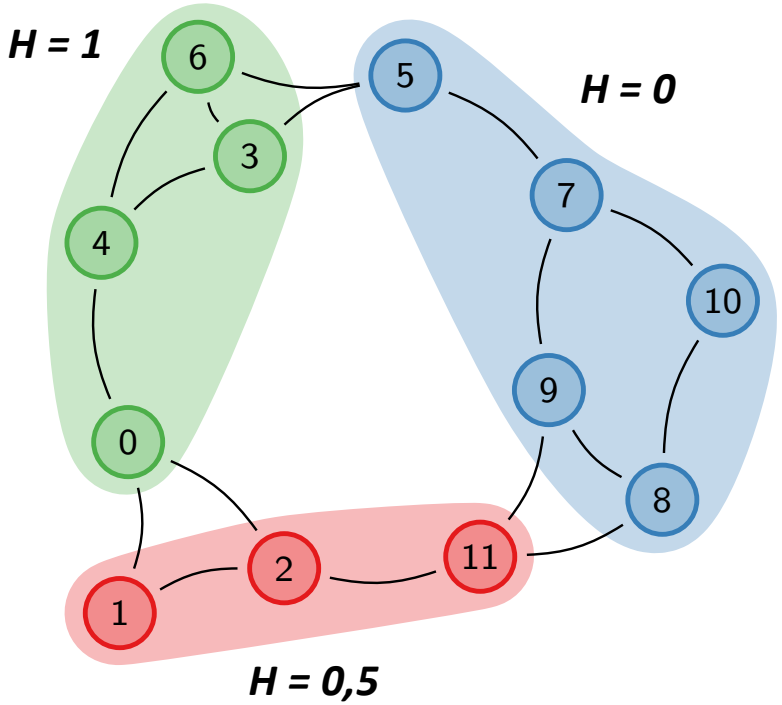
Louvain Algorithm

Supposing to use as reward the simple reward function with γ = 0,5 instead of modularity, what are the Louvain communities on this graph?

H = 1,5

Move 0? YES

two edges gained, one lost



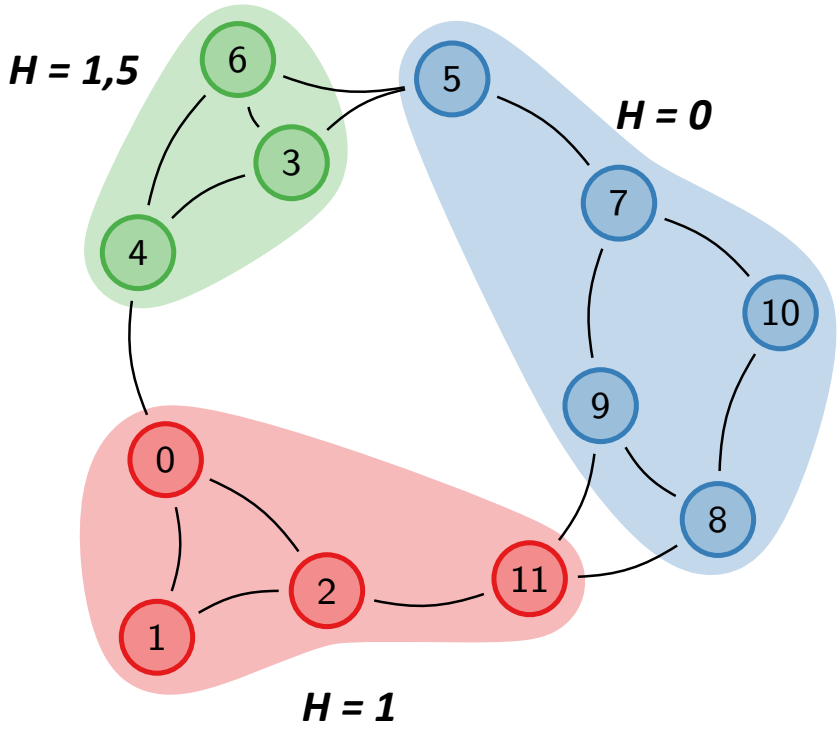
Louvain Algorithm

Supposing to use as reward the simple reward function with $\gamma = 0,5$ instead of modularity, what are the Louvain communities on this graph?

H = 2,5

Move 0? YES

two edges gained, one lost



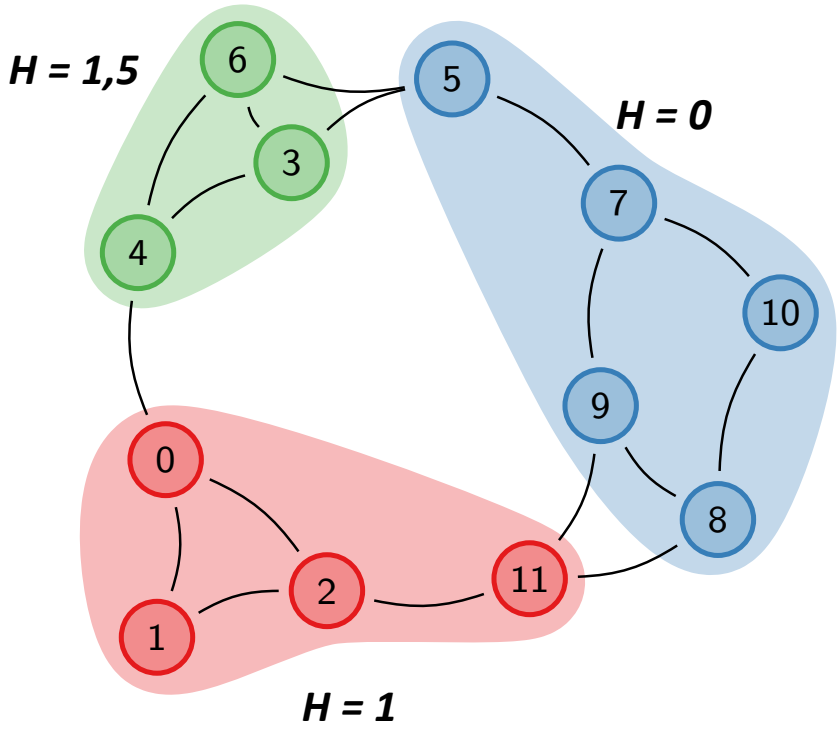
Louvain Algorithm

Supposing to use as reward the simple reward function with $\gamma = 0,5$ instead of modularity, what are the Louvain communities on this graph?

H = 2,5

Move 1? NO
Move 2? NO
Move 3? NO
Move 4? NO
Move 5? YES

two edges gained, one lost



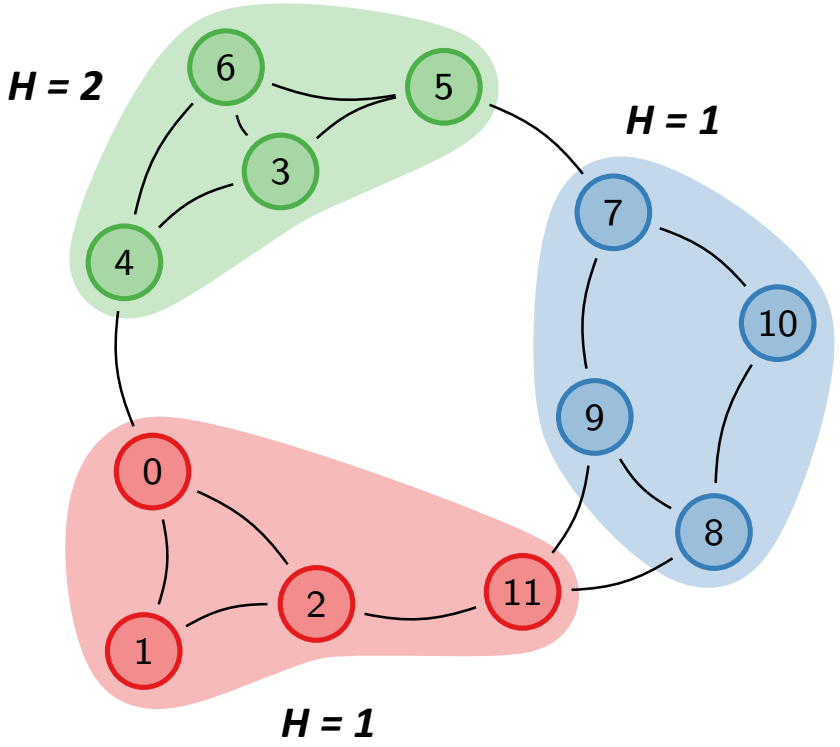
Louvain Algorithm

Supposing to use as reward the simple reward function with $\gamma = 0,5$ instead of modularity, what are the Louvain communities on this graph?

H = 4

Move 1? NO
Move 2? NO
Move 3? NO
Move 4? NO
Move 5? YES

two edges gained, one lost



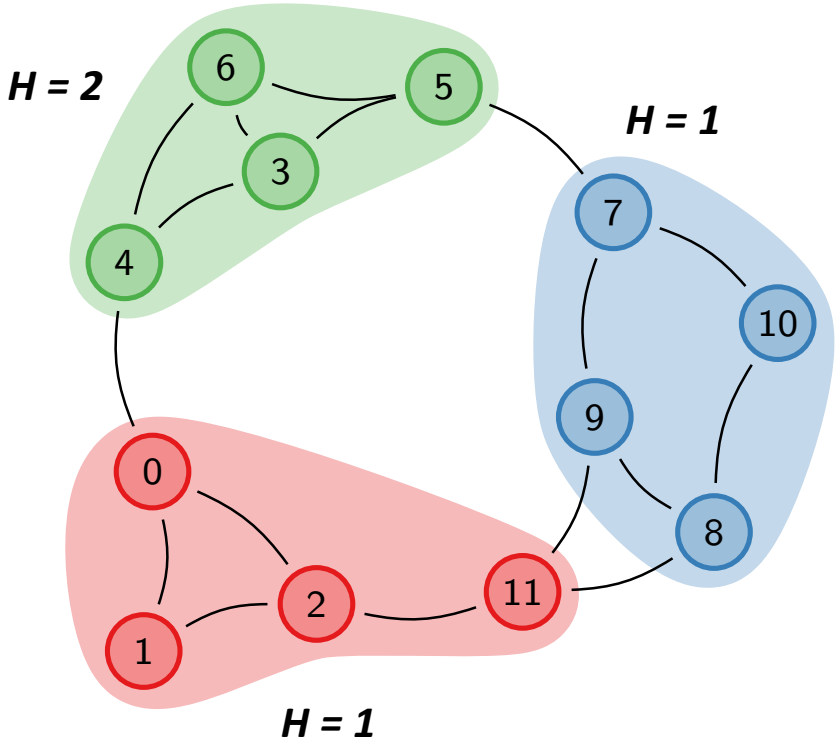
Louvain Algorithm

Supposing to use as reward the simple reward function with $\gamma = 0,5$ instead of modularity, what are the Louvain communities on this graph?

H = 4

Move 6? NO
Move 7? NO
Move 8? NO
Move 9? NO
Move 10? NO
Move 11? YES

two edges gained, one lost



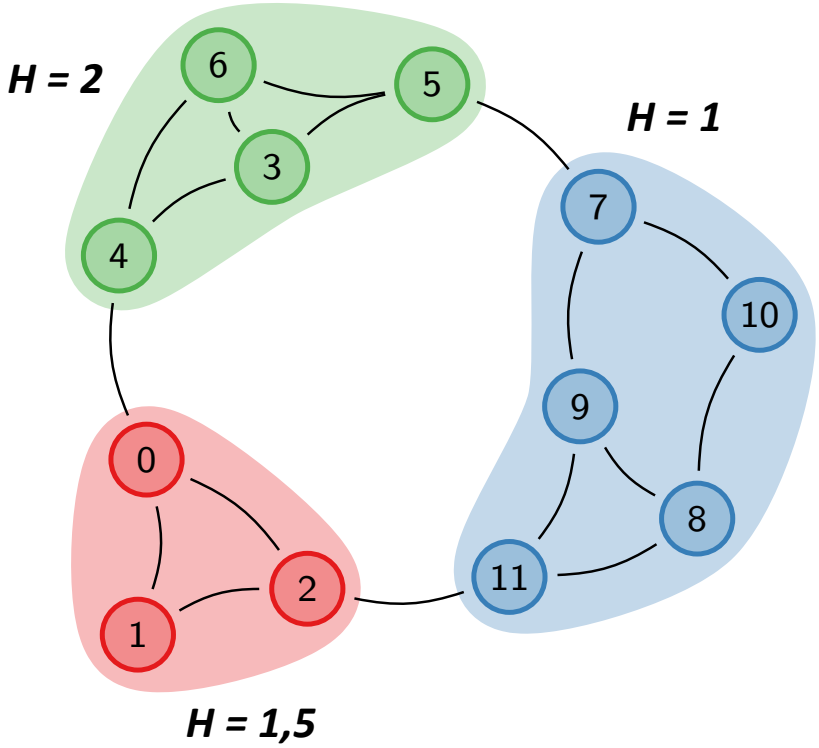
Louvain Algorithm

Supposing to use as reward the simple reward function with $\gamma = 0,5$ instead of modularity, what are the Louvain communities on this graph?

H = 4,5

Move 6? NO
Move 7? NO
Move 8? NO
Move 9? NO
Move 10? NO
Move 11? YES

two edges gained, one lost



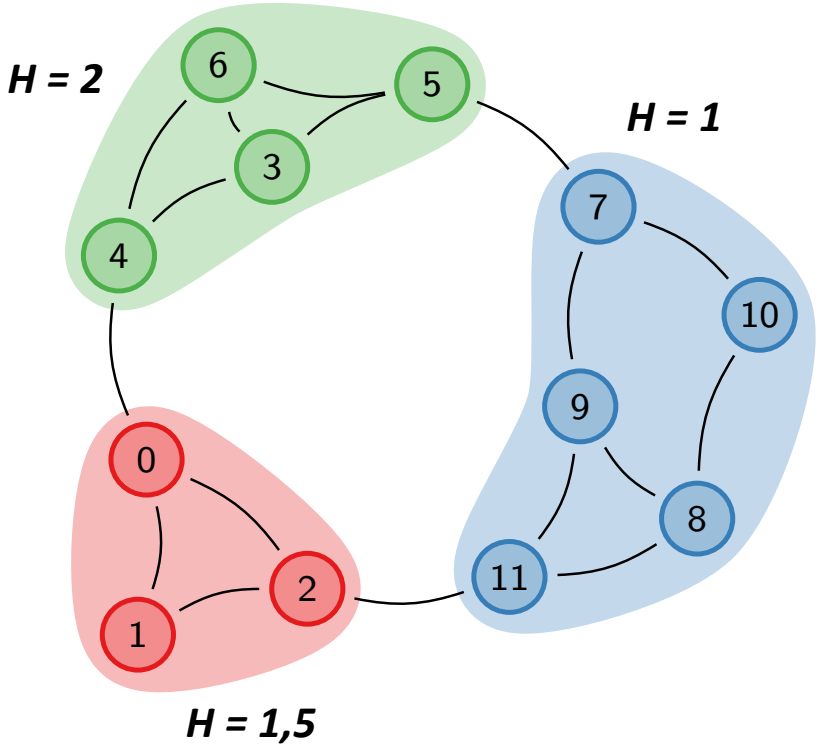
Louvain Algorithm

Supposing to use as reward the simple reward function with γ = 0,5 instead of modularity, what are the Louvain communities on this graph?

H = 4,5

Cycle ended, have some clusters been modified? **YES, restart cycle**

Move 0? NO
Move 1? NO
…
Move 11? NO
Cycle ended, have some clusters been modified? **NO, aggregation**

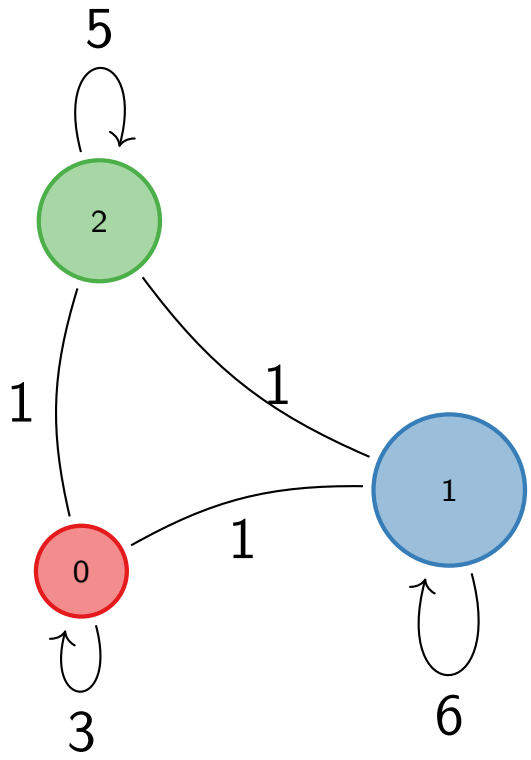


Louvain Algorithm

Supposing to use as reward the simple reward function with $\gamma = 0,5$ instead of modularity, what are the Louvain communities on this graph?

H = 4,5

Move 0? NO
Move 1? NO
Move 2? NO
Cycle ended, have some clusters been modified? **NO, aggregation**



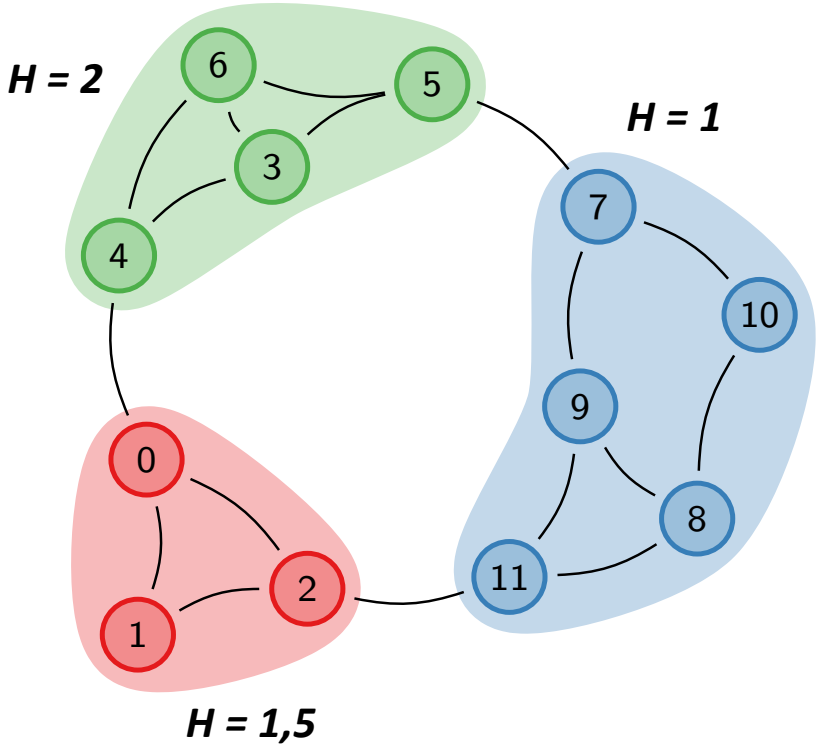
Louvain Algorithm

Supposing to use as reward the simple reward function with $\gamma = 0,5$ instead of modularity, what are the Louvain communities on this graph?

H = 4,5

No further aggregations available, so Louvain algorithm ended

These are the resulting communities



Leiden Algorithm

- **Initialization**: Each node in the network is assigned to its own community and marked as *unstable*

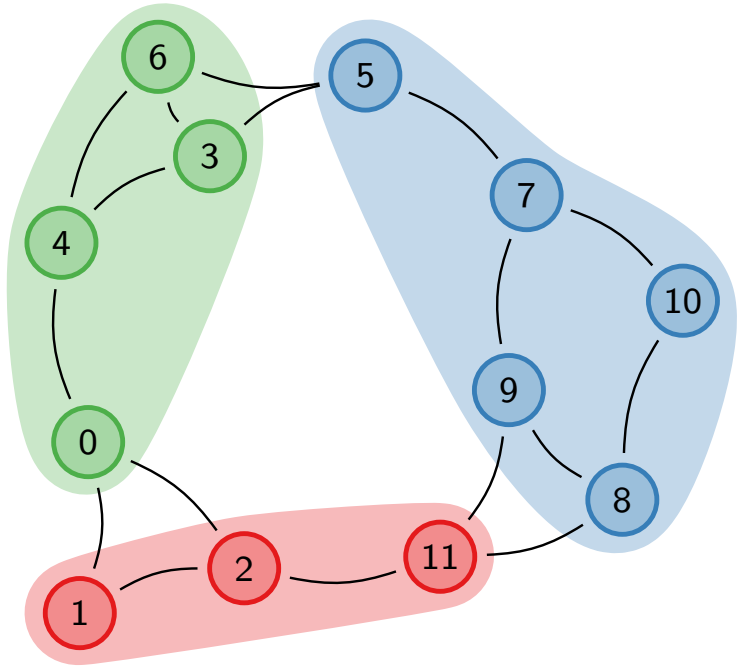
- **Phase 1 – Moving Nodes**:
  1. Visit each *unstable* node, one by one
  2. When the node is assigned to a cluster mark it as *stable*
  3. Mark neighbours as *unstable* when a node moves through clusters
- Repeat phase 1 until no node is moved
- **Phase 2 – Refinement:** takes decisions on disconnected clusters
- **Phase 3 - Aggregation**: Aggregate clusters like in the Louvain algorithm, taking into consideration the refinement output.

- Phases 1, 2, and 3 are repeated until modularity is maximized

Leiden Algorithm

- **Phase 1 – Moving Nodes**:
  1. Visit each *unstable* node, one by one
  2. When the node is assigned to a cluster mark it as *stable*
  3. Mark neighbours as *unstable* when a node moves through clusters

Supposing to use as reward the simple reward function with $\gamma = 0{,}5$ instead of modularity, what are the Leiden communities on this graph after the phase 1?

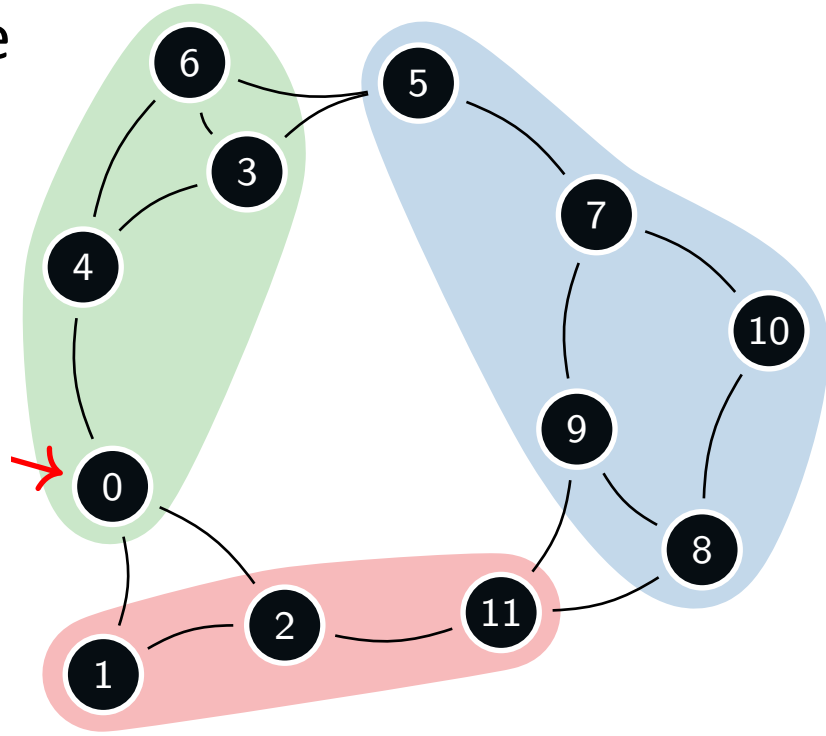


Leiden Algorithm

- **Phase 1 – Moving Nodes**:
  1. Visit each *unstable* node, one by one
  2. When the node is assigned to a cluster mark it as *stable*
  3. Mark neighbours as *unstable* when a node moves through clusters

Initialization: all the nodes are unstable

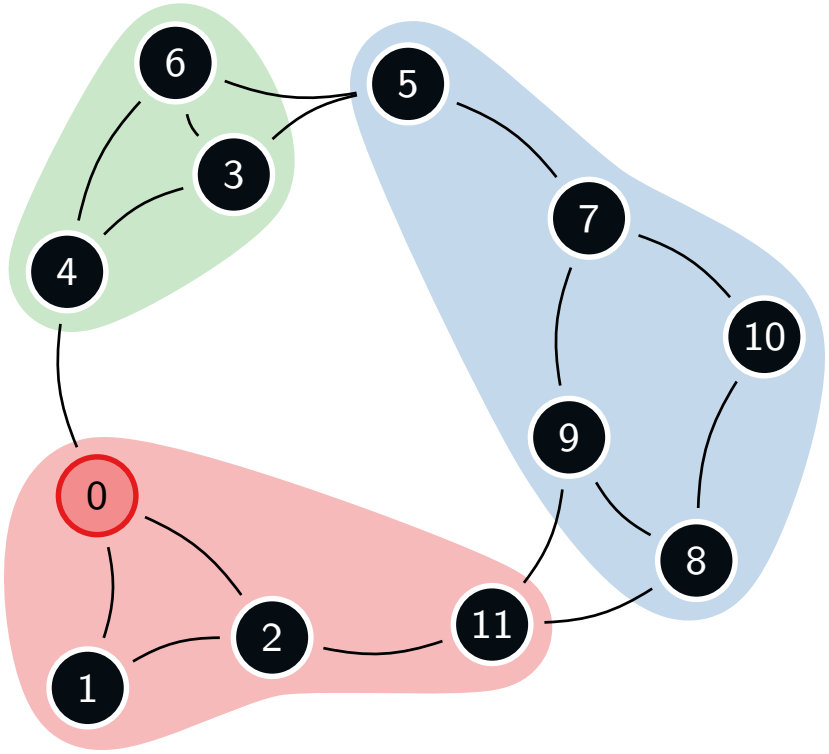
Move 0? YES
Mark 0 as stable
Mark the neighbours as unstable:
all the neighbours are still unstable



Leiden Algorithm

- **Phase 1 – Moving Nodes**:
  1. Visit each *unstable* node, one by one
  2. When the node is assigned to a cluster mark it as *stable*
  3. Mark neighbours as *unstable* when a node moves through clusters

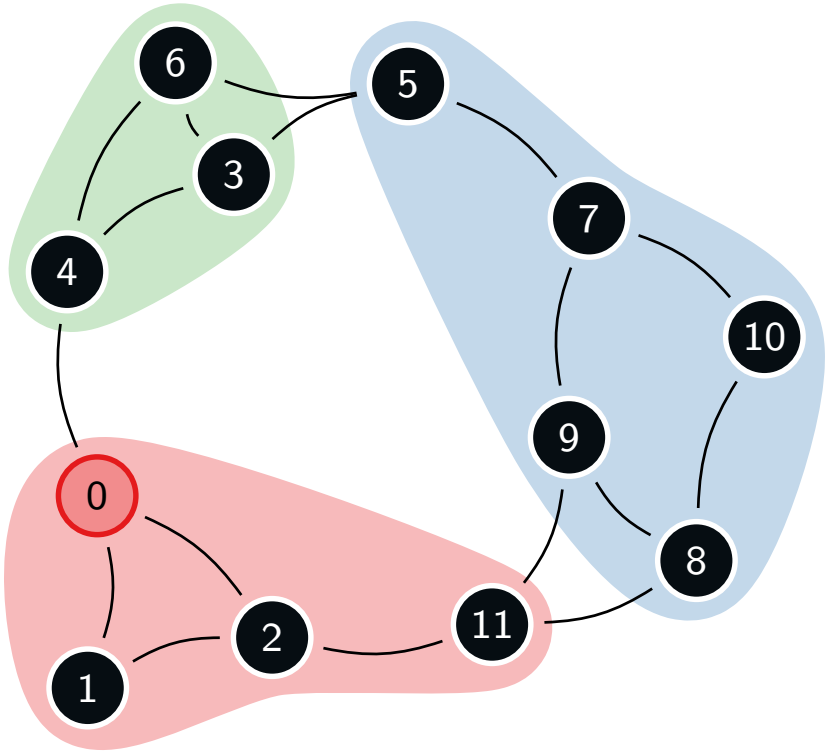
Move 0? YES
Mark 0 as stable
Mark the neighbours as unstable:
all the neighbours are still unstable



Leiden Algorithm

- **Phase 1 – Moving Nodes**:
  1. Visit each *unstable* node, one by one
  2. When the node is assigned to a cluster mark it as *stable*
  3. Mark neighbours as *unstable* when a node moves through clusters

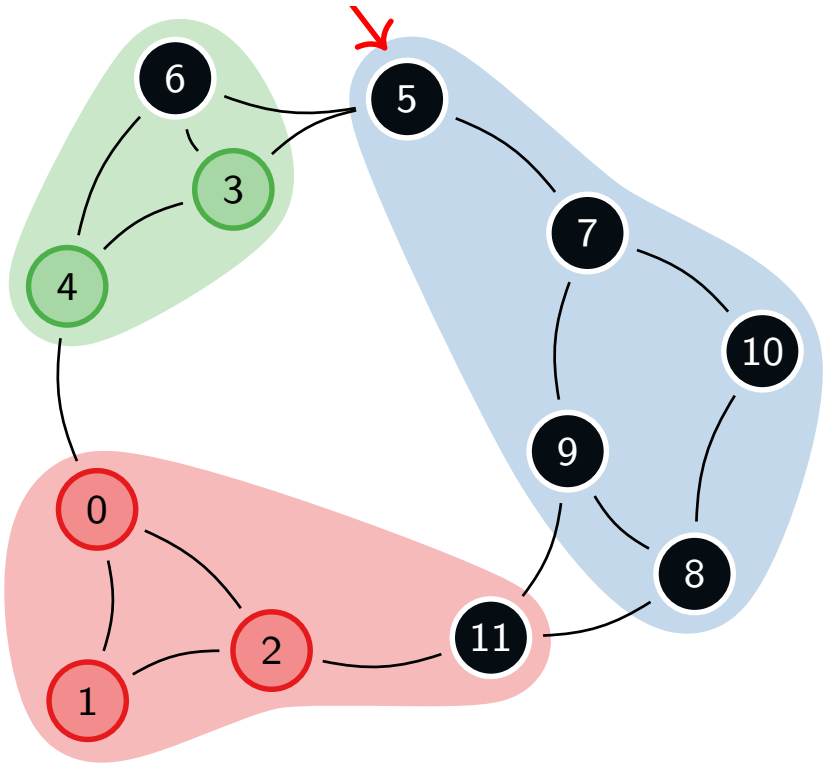
Move 1? NO => STABLE
Move 2? NO => STABLE
Move 3? NO => STABLE
Move 4? NO => STABLE
Move 5? YES



Leiden Algorithm

- **Phase 1 – Moving Nodes**:
  1. Visit each *unstable* node, one by one
  2. When the node is assigned to a cluster mark it as *stable*
  3. Mark neighbours as *unstable* when a node moves through clusters

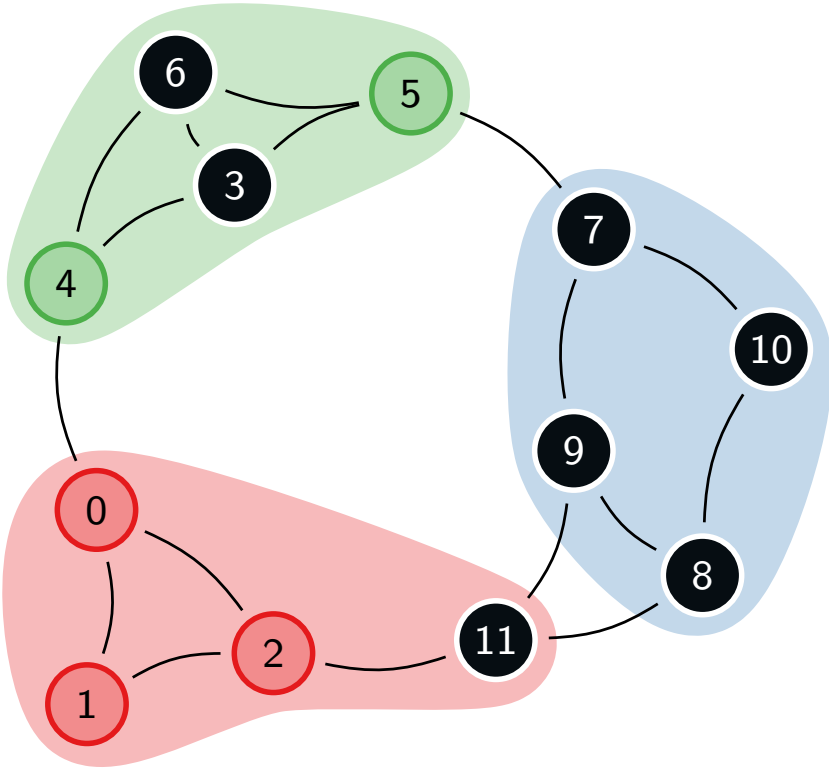
Move 5? YES
Mark 5 as stable
Mark the neighbours as unstable:
6-7 are still unstable
3 becomes unstable



Leiden Algorithm

- **Phase 1 – Moving Nodes**:
  1. Visit each *unstable* node, one by one
  2. When the node is assigned to a cluster mark it as *stable*
  3. Mark neighbours as *unstable* when a node moves through clusters

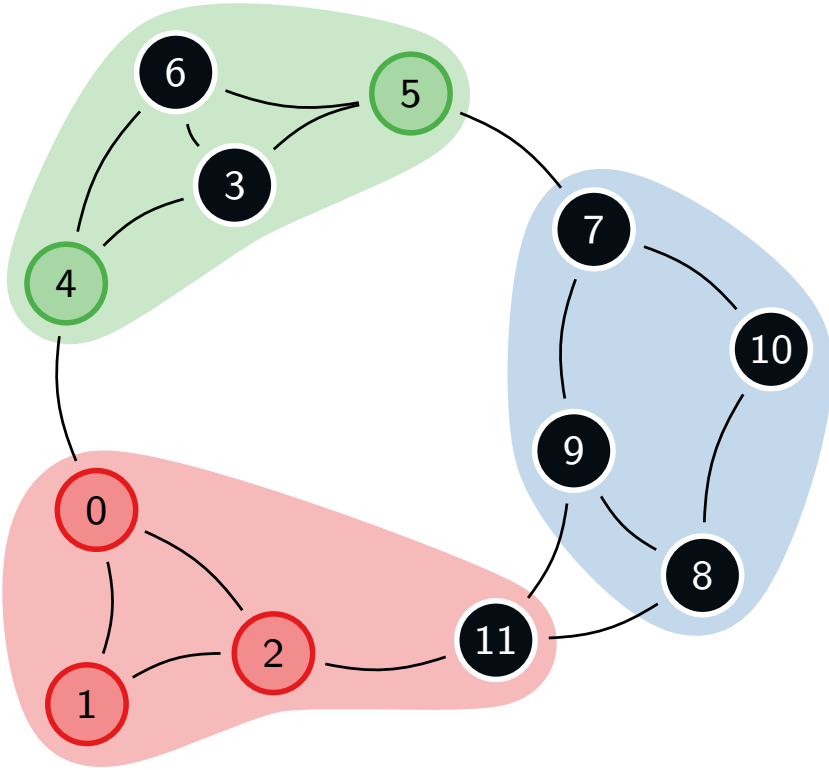
Move 5? YES
Mark 5 as stable
Mark the neighbours as unstable:
6-7 are still unstable
3 becomes unstable



Leiden Algorithm

- **Phase 1 – Moving Nodes**:
  1. Visit each *unstable* node, one by one
  2. When the node is assigned to a cluster mark it as *stable*
  3. Mark neighbours as *unstable* when a node moves through clusters

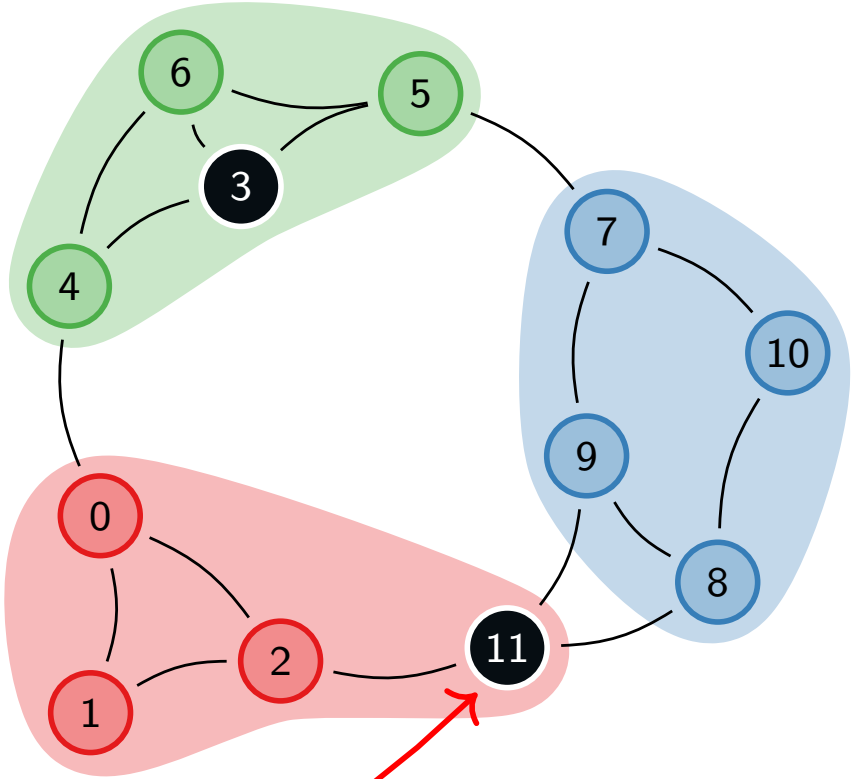
Move 6? NO => STABLE
Move 7? NO => STABLE
Move 8? NO => STABLE
Move 9? NO => STABLE
Move 10? NO => STABLE
Move 11? YES



Leiden Algorithm

- **Phase 1 – Moving Nodes**:
  1. Visit each *unstable* node, one by one
  2. When the node is assigned to a cluster mark it as *stable*
  3. Mark neighbours as *unstable* when a node moves through clusters

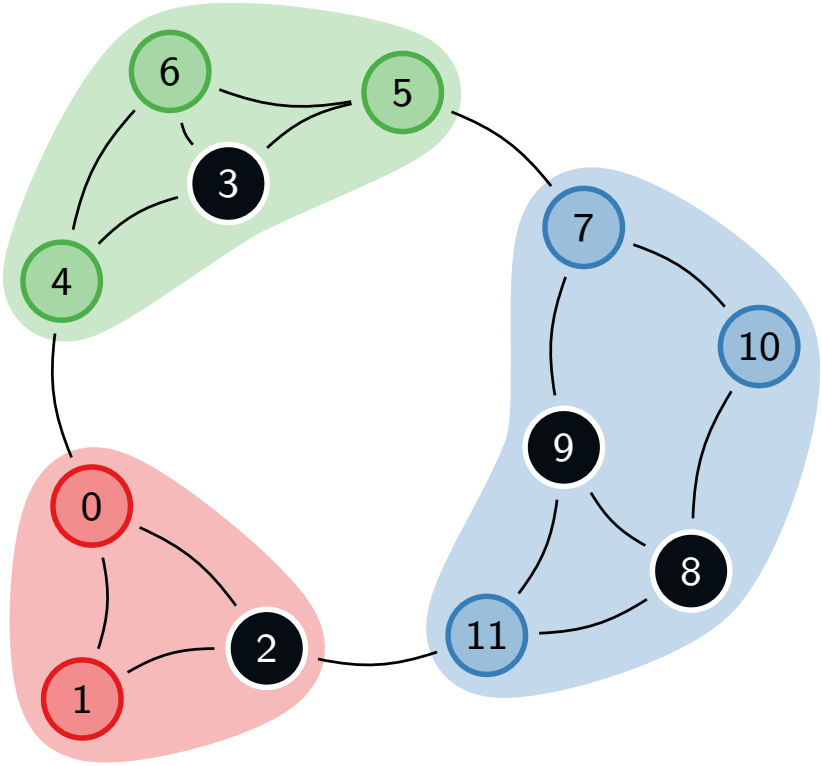
Move 11? YES
Mark 11 as stable
Mark the neighbours as unstable:
2-8-9 become unstable



Leiden Algorithm

- **Phase 1 – Moving Nodes**:
  1. Visit each *unstable* node, one by one
  2. When the node is assigned to a cluster mark it as *stable*
  3. Mark neighbours as *unstable* when a node moves through clusters

Move 11? YES
Mark 11 as stable
Mark the neighbours as unstable:
2-8-9 become unstable

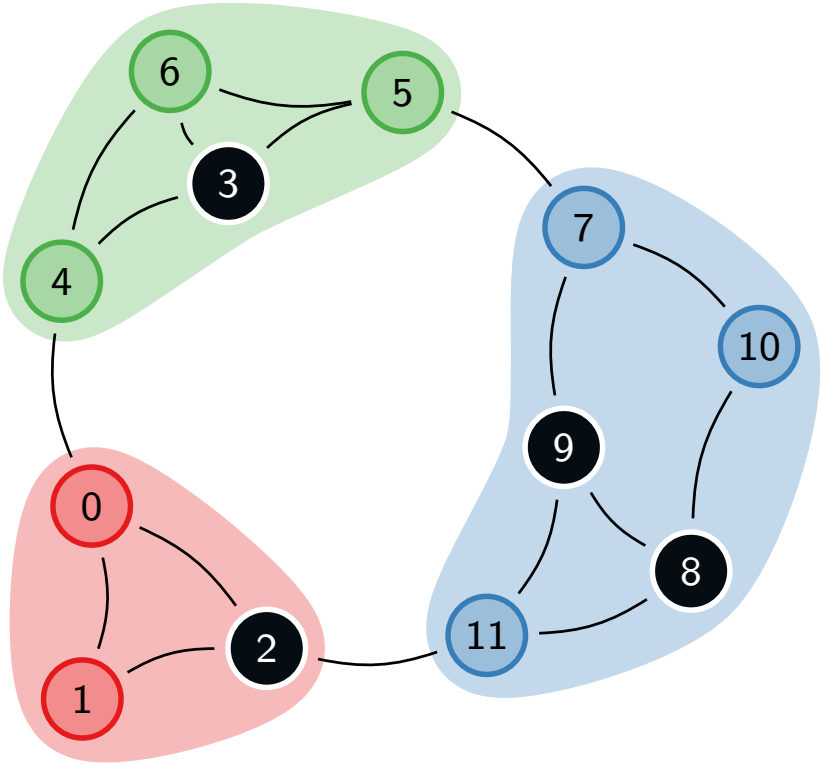


Leiden Algorithm

- **Phase 1 – Moving Nodes**:
  1. Visit each *unstable* node, one by one
  2. When the node is assigned to a cluster mark it as *stable*
  3. Mark neighbours as *unstable* when a node moves through clusters

Cycle ended, have some clusters been modified? **YES, restart cycle**

Move 2? NO => STABLE
Move 3? NO => STABLE
Move 8? NO => STABLE
Move 9? NO => STABLE

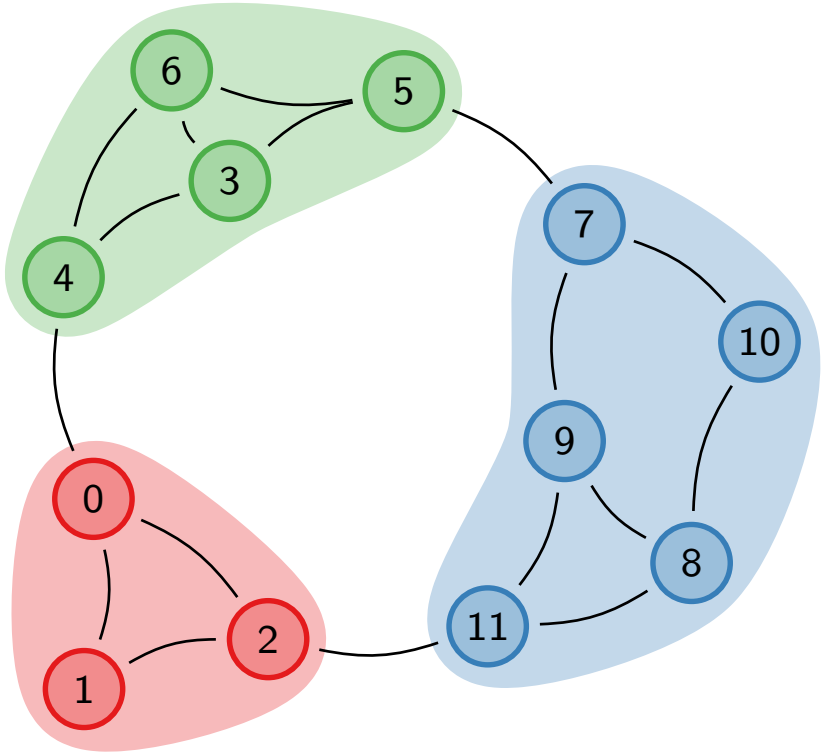


Leiden Algorithm

- **Phase 1 – Moving Nodes**:
  1. Visit each *unstable* node, one by one
  2. When the node is assigned to a cluster mark it as *stable*
  3. Mark neighbours as *unstable* when a node moves through clusters

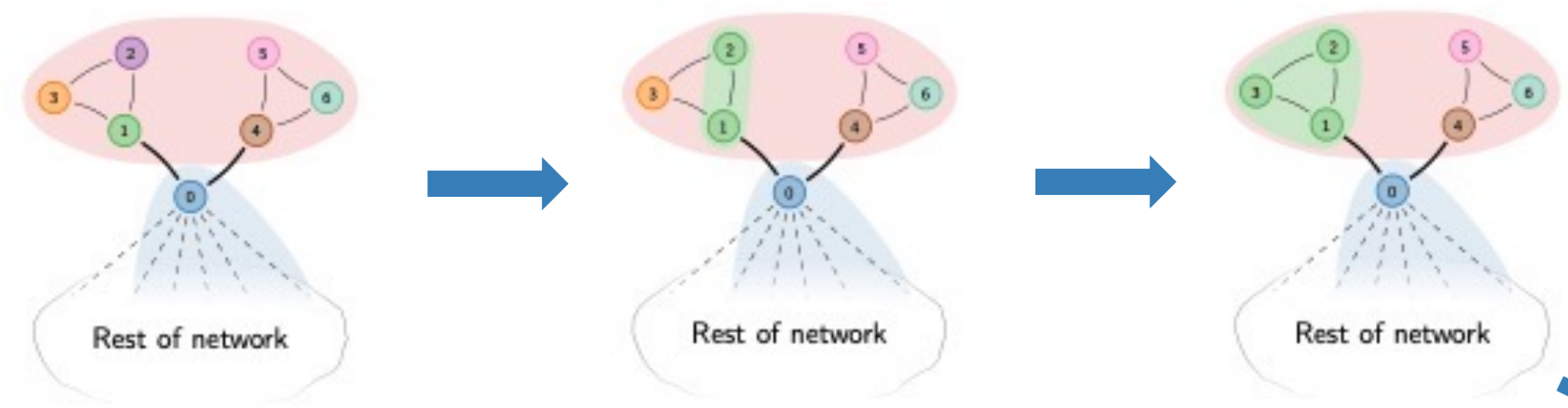
Cycle ended, have some clusters been modified? **NO**

Start refinement and aggregation



Leiden Algorithm

- **Phase 2,3 – Refinement and aggregation**

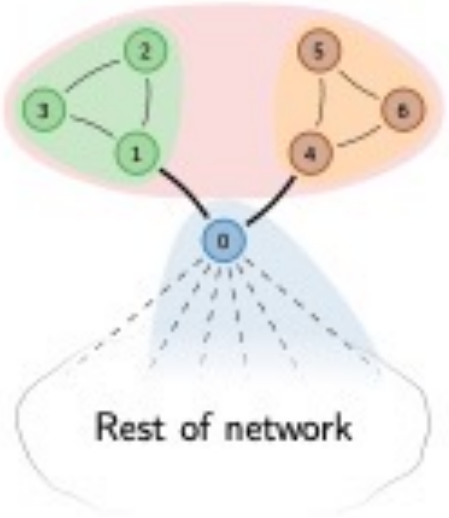


Refinement identifies and aggregates disconnected clusters

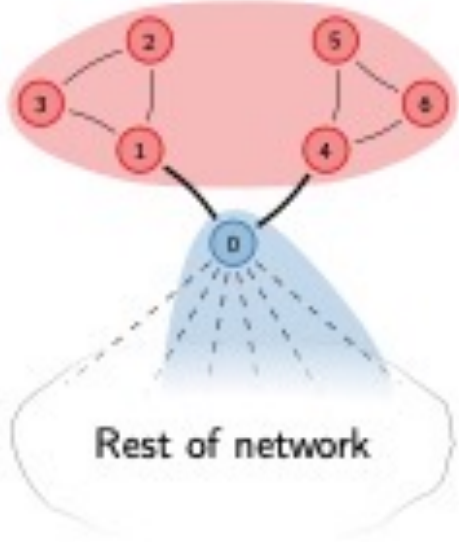
Aggregated network **Refined network**

Leiden vs Louvain

- **Leiden is faster** (checks only unstable nodes) but more complex
- The **refinement manages better disconnected nodes**, while Louvain puts all the disconnected nodes in a unique cluster



Leiden refinement



Louvain management

Leiden vs Louvain

- **Leiden is better on large networks** (could work fast on networks having billions of links)
- Leiden has a highly quicker first iteration and after the stable iteration obtains better quality with respect to Louvain