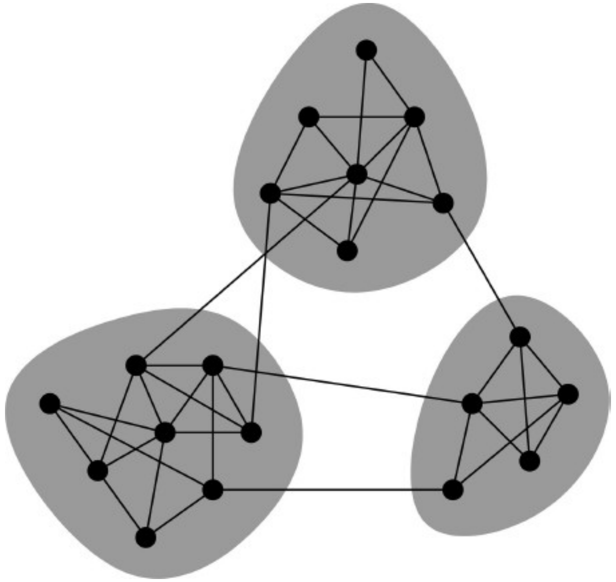
Louvain Algorithm

Network Communities

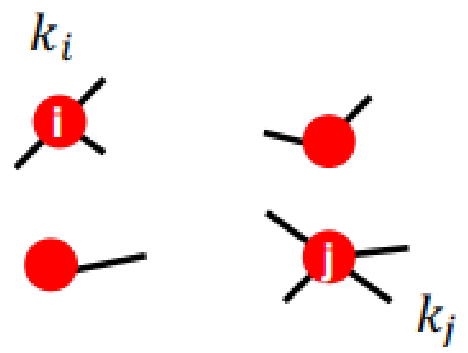
- **Communities:** sets of **tightly connected nodes**
- Define: **Modularity $\boldsymbol{Q}$**
  - A measure of how well a network is partitioned into communities
  - Given a **partitioning** of the network into groups disjoint $\boldsymbol{s} \in \boldsymbol{S}$



$$Q \propto \sum_{s \in S} [\ (\#\ \text{edges within group}\ s) - \underbrace{(\text{expected}\ \#\ \text{edges within group}\ s)\ ]}_{\text{Need a null model}}$$

Null Model: Configuration Model

- **Given real G on n nodes and m edges, construct rewired network G'**
  - **Same degree distribution** but uniformly random connections
  - Consider G' as a multigraph (multiple edges exist between nodes)
  - **The expected number of edges between nodes i and j** of degrees k_i and k_j equals: $k_i \cdot \frac{k_j}{2m} = \frac{k_i k_j}{2m}$
    - There are $2m$ **half** edges in total.
    - For each of k_i half edges from node i, the chance of it landing to node j is $k_j/2m$, hence $k_i k_j/2m$.



Modularity

- **Modularity of partitioning S of graph G:**
  - $\boldsymbol{Q \propto \sum_{s \in S} [(\#\text{edges within group } s) - (\text{expected \# edges within group } s)]}$
  - $\boldsymbol{Q(G,S) = \underbrace{\frac{1}{2m}}_{\text{Normalizing}} \sum_{s \in S} \sum_{i \in s} \sum_{j \in s} \left(A_{ij} - \frac{k_i k_j}{2m}\right)}$

A_{ij} = 1 if $i \to j$,
0 otherwise
(if G is weighted then A_{ij} is the edge weight)

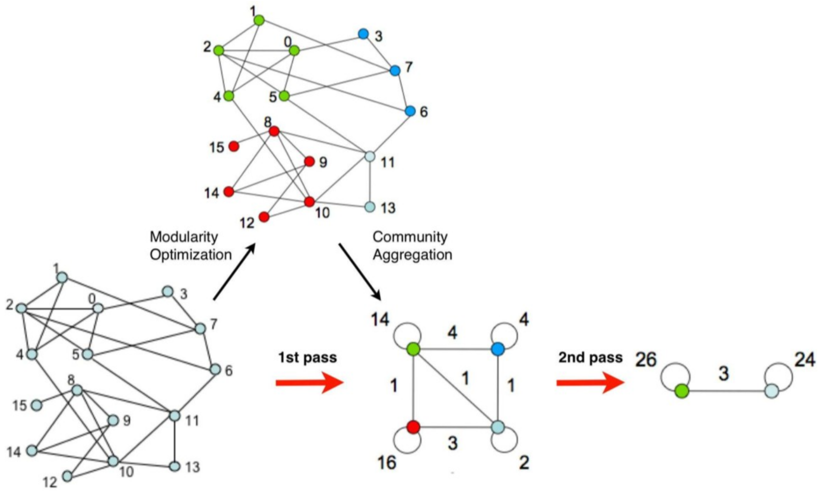
- **Modularity values take range [−1/2,1]**
  - It is positive if the number of edges within groups exceeds the expected number
  - $\boldsymbol{Q}$ greater than **0.3-0.7** means **significant community structure**
  - **Notice Modularity applies to weighted and unweighted networks.**

Brandes, U.; Delling, D.; Gaertler, M.; Gorke, R.; Hoefer, M.; Nikoloski, Z.; Wagner, D. "On Modularity Clustering". IEEE Transactions on Knowledge and Data Engineering. 20 (2): 172–188.

Louvain Algorithm: At High Level

- Louvain algorithm greedily **maximizes** modularity
- **Each pass is made of 2 phases:**
  - **Phase 1:** Modularity is optimized by allowing only local changes to node-communities memberships
  - **Phase 2:** The identified communities are aggregated into super-nodes to build a new network
  - **Goto Phase 1**

The passes are repeated **iteratively** until no increase of modularity is possible.



Louvain: 1st phase (Partitioning)

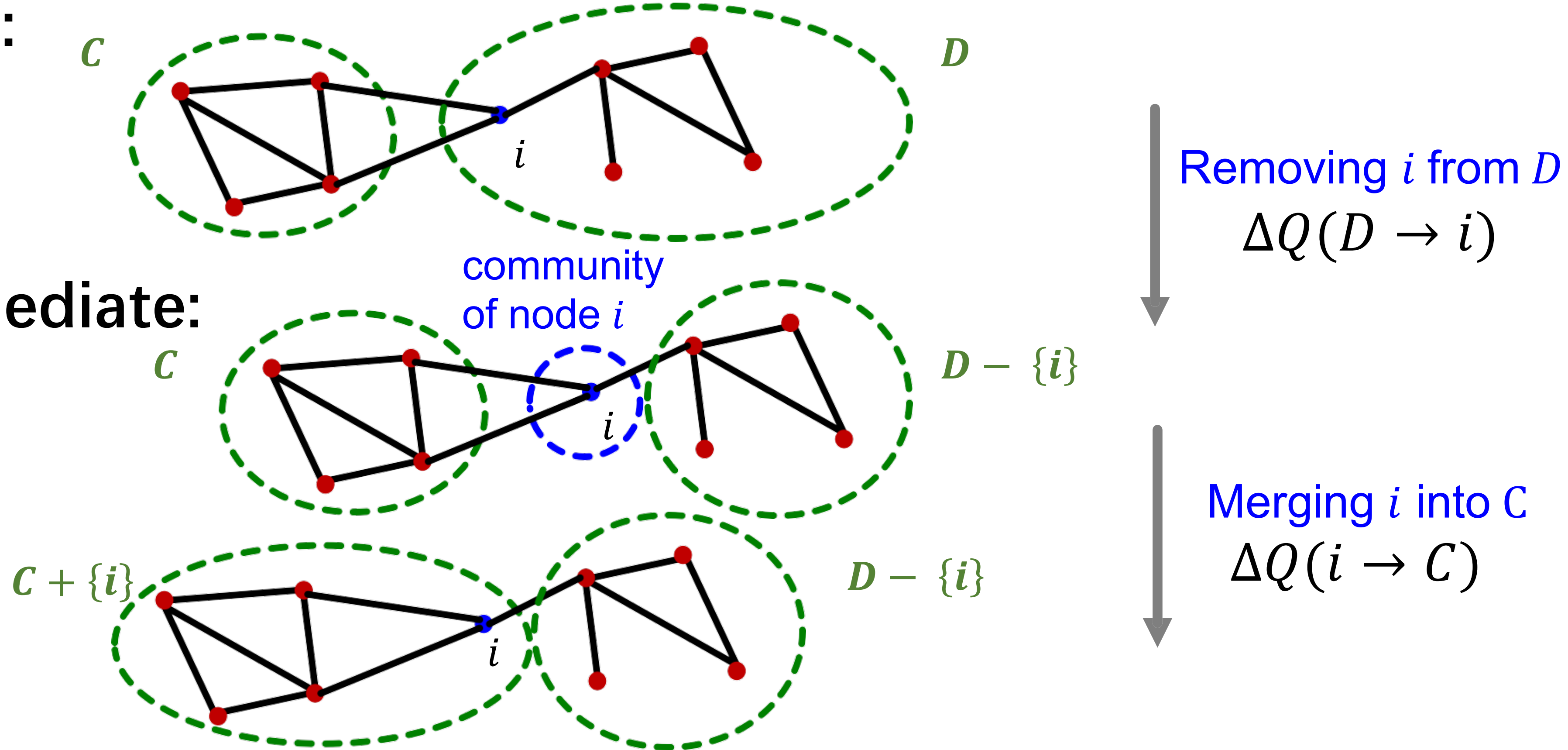
- Put each node in a graph into a distinct community
- For each node i, the algorithm performs two calculations:
  - Compute the modularity delta (ΔQ) when putting node i into the community of some neighbor j
  - Move i to a community of node j that yields the largest gain in ΔQ
- **Phase 1 runs until no movement yields a gain**

Louvain: Modularity Gain

- **What is ΔQ if we move node i to community D to C?**

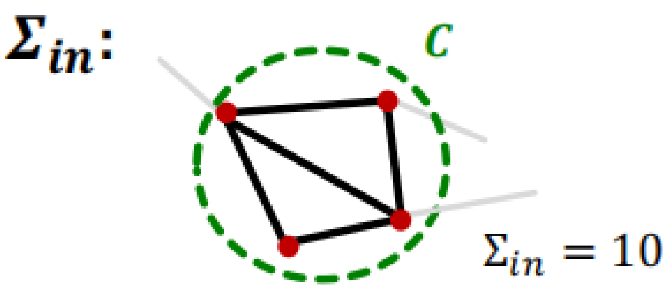
$$\Delta Q(D \rightarrow i \rightarrow C) = \Delta Q(D \rightarrow i) + \Delta Q(i \rightarrow C)$$

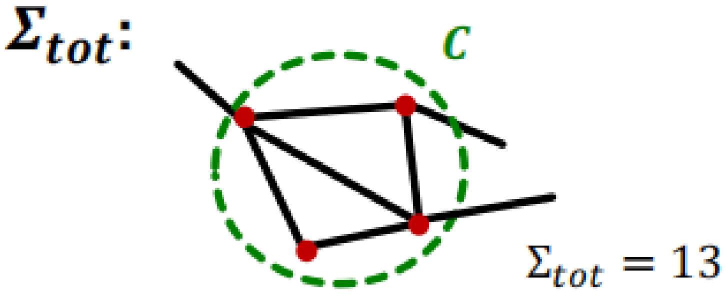
- **Before:**
- **Intermediate:**
- **After:**



Deriving $\Delta Q(i \to C)$

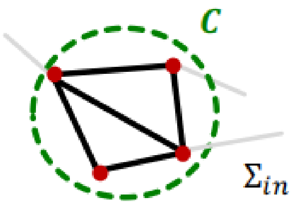
- **Let's derive $\boldsymbol{\Delta Q(i \to C)}$**
- First, we derive modularity **within $\boldsymbol{C}$**, i.e., $Q(C)$.
- **Define:**
  - $\Sigma_{in} \equiv \sum_{i,j \in C} A_{ij}$… sum of link weights <u>between</u> nodes in C (edge ij and ji count twice)
  - $\Sigma_{tot} \equiv \sum_{i \in C} k_i$… sum of <u>all</u> link weights of nodes in C (edges inside twice, outside only once)





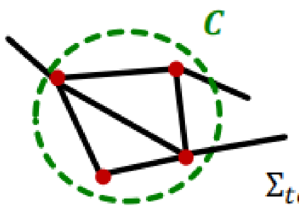
Deriving $\Delta Q(i \to C)$

$\boldsymbol{\Sigma_{in}}$:



$\Sigma_{in} = 10$

$\boldsymbol{\Sigma_{tot}}$:



$\Sigma_{tot} = 13$

- **Define:**
  - $\Sigma_{in} \equiv \sum_{i,j \in C} A_{ij} \cdots$ sum of link weights between nodes in C
  - $\Sigma_{tot} \equiv \sum_{i \in C} k_i \cdots$ sum of all link weights of nodes in C
- **Then, we have**

$$Q(C) \equiv \frac{1}{2m} \sum_{i,j \in C} \left[A_{ij} - \frac{k_i k_j}{2m} \right] = \frac{\sum_{i,j \in C} A_{ij}}{2m} - \frac{\left(\sum_{i \in C} k_i\right)\left(\sum_{j \in C} k_j\right)}{(2m)^2}$$

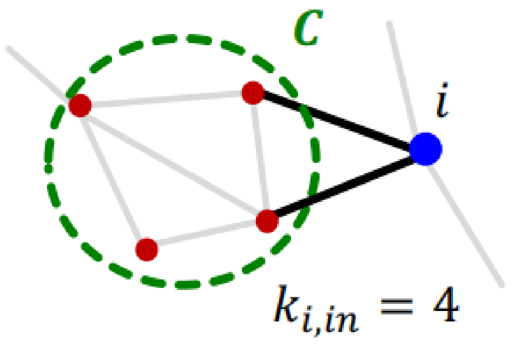
Links within the community $= \boxed{\frac{\Sigma_{in}}{2m}} - \boxed{\left(\frac{\Sigma_{tot}}{2m}\right)^2}$ **Total links**

- $Q(C)$ is large when most of the total links are within-community links

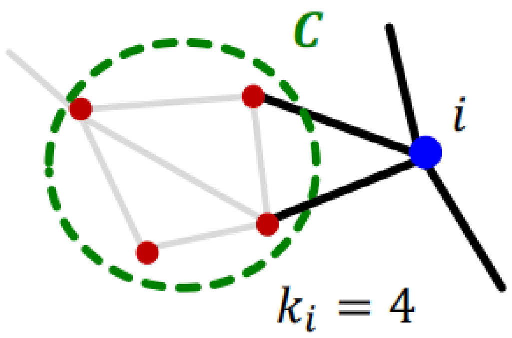
Deriving $\Delta Q(i \to C)$

- **Further Define:**
  - $\boldsymbol{k_{i,in}} \equiv \sum_{j \in C} A_{ij} + \sum_{j \in C} A_{ji} \cdots$ sum of link weights between node i and C
  - $\boldsymbol{k_i} \cdots$ sum of all link weights (i.e., degree) of node i

$\boldsymbol{k_{i,in}}$ **:**

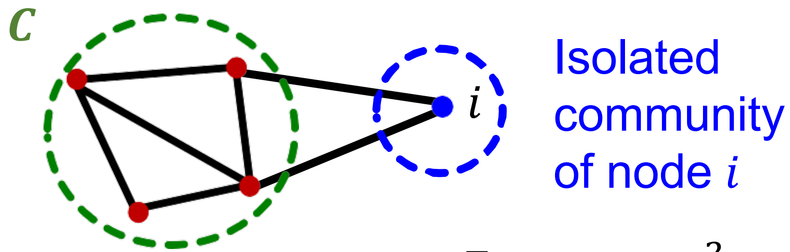


$\boldsymbol{k_i}$ **:**



Deriving $\Delta Q(i \rightarrow C)$

Before Merging

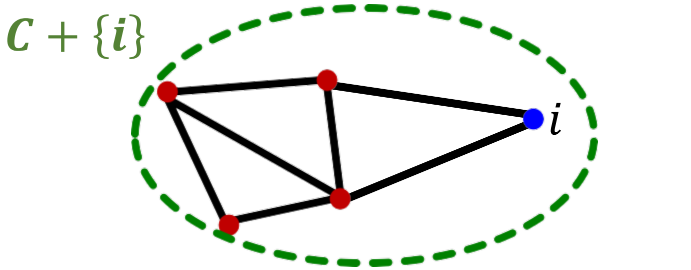


We have $Q(C) = \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2$

$$Q_{before} = Q(C) + Q(\{i\})$$

$$= \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2\right] + \left[0 - \left(\frac{k_i}{2m}\right)^2\right]$$

After Merging



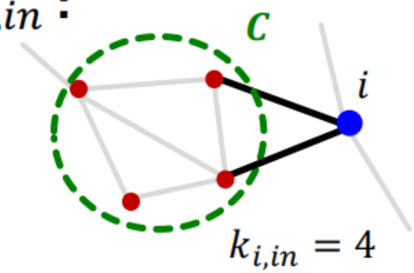
$$Q_{\text{after}} = Q(C + \{i\})$$

"Σ_{in}" of $C + \{i\}$ "Σ_{tot}" of $C + \{i\}$

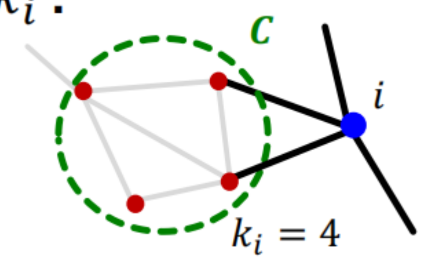
$$= \frac{\boxed{\Sigma_{in} + k_{i,in}}}{2m} - \left(\frac{\boxed{\Sigma_{tot} + k_i}}{2m}\right)^2$$

Recall:

$k_{i,in}$:



k_i :



Louvain: Modularity Gain

- $\Delta Q(i \rightarrow C) = Q_{after} - Q_{before}$

$$= \left[\frac{\Sigma_{in}+k_{i,in}}{2m} - \left(\frac{\Sigma_{tot}+k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

- $\Delta Q(D \rightarrow i)$ can be derived similarly.

- **In summary, we can compute:**

$$\Delta Q(D \rightarrow i \rightarrow C) = \Delta Q(D \rightarrow i) + \Delta Q(i \rightarrow C)$$

Louvain 1st Phase: Summary

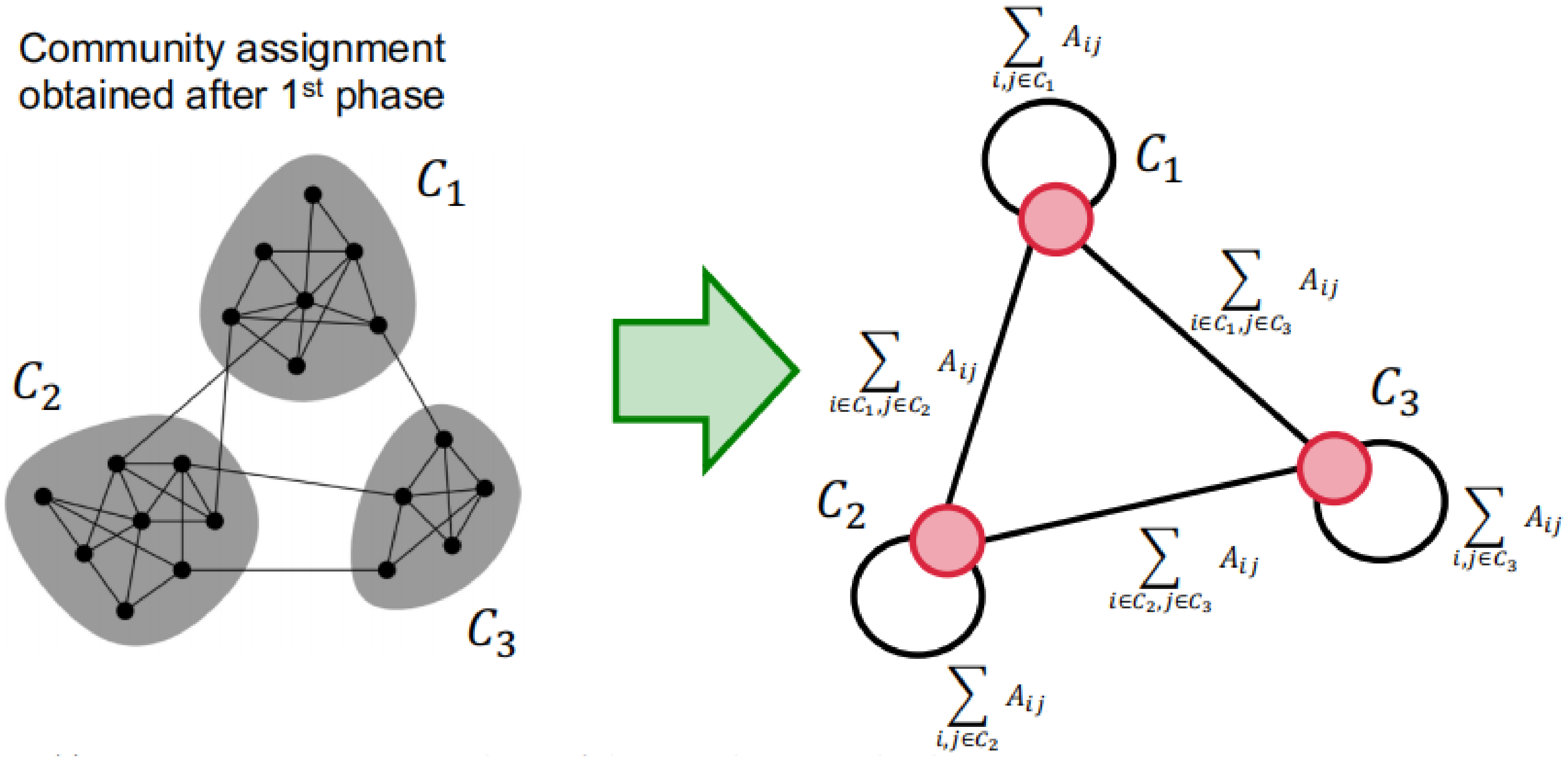
- **Iterate until no node moves to a new community:**
  - For each node $i \in V$ currently in community C, compute the best community C':
  - $C' = \text{argmax}_{C'} \Delta Q(C \rightarrow i \rightarrow C')$
  - If $\Delta Q(C \rightarrow i \rightarrow C') > 0$, then **update the community:**
    - $C \leftarrow C - \{i\}$
    - $C' \leftarrow C' + \{i\}$

Louvain: 2nd phase (Restructuring)

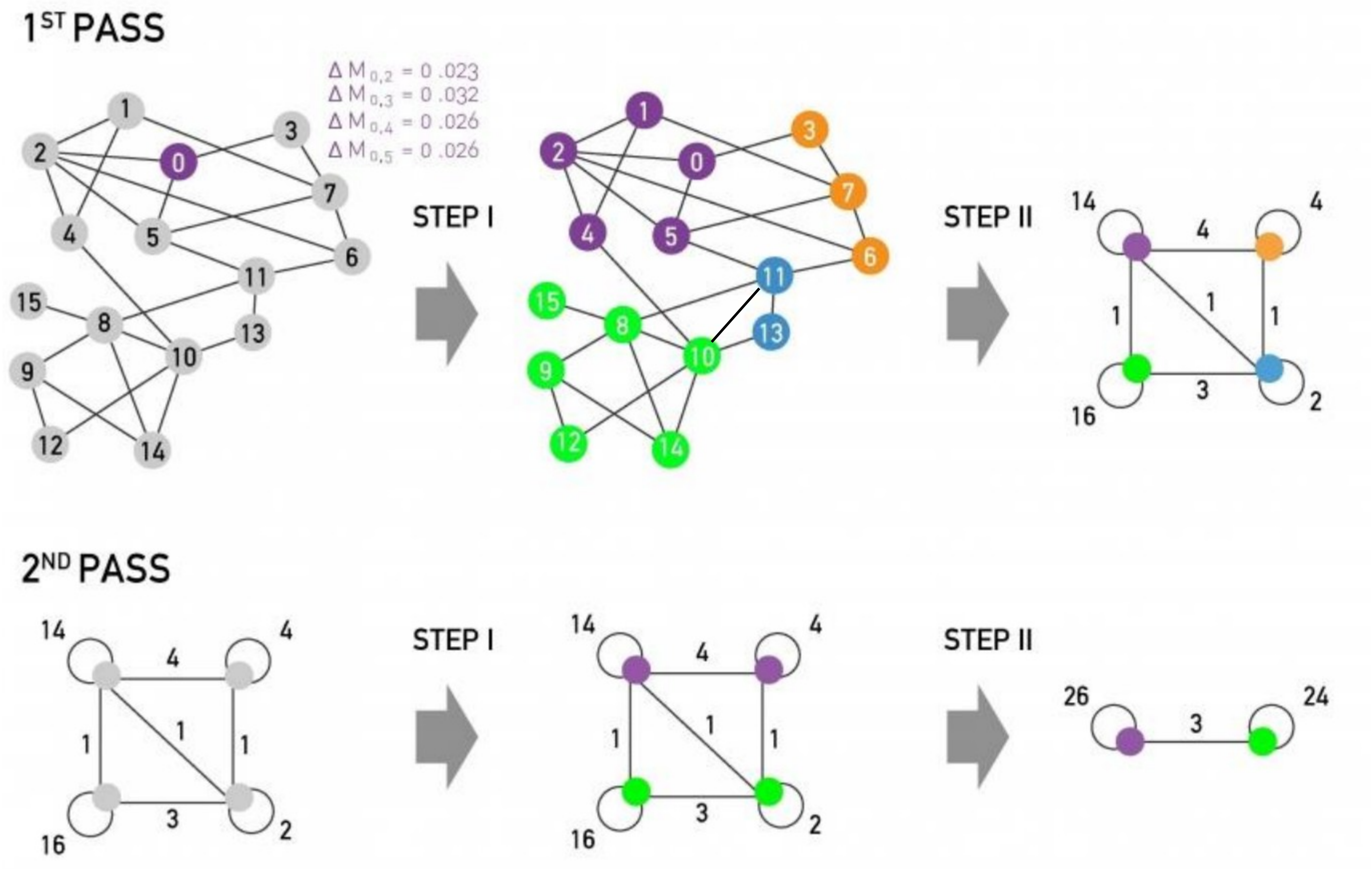
- The communities obtained in the first phase are contracted into **super-nodes**, and the network is created accordingly:
  - Super-nodes are connected if there is at least one edge between the nodes of the corresponding communities
  - The weight of the edge between the two super nodes is the sum of the weights from all edges between their corresponding communities
- **Phase 1 is then run on the super-node network**

Louvain 2nd Phase: Summary

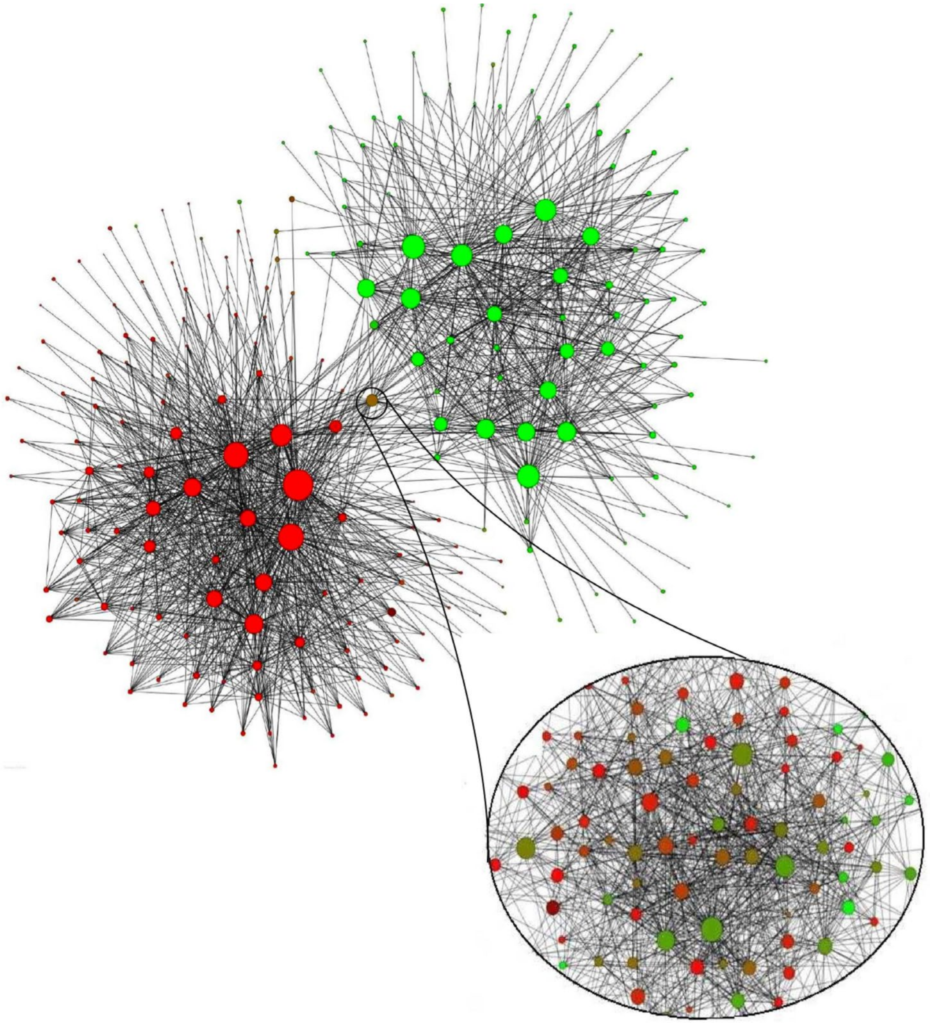
- Super nodes are constructed by merging nodes in the same community.



Louvain Algorithm



Belgian Mobile Phone Network



- 2M nodes
- **Red nodes:** French speakers
- **Green nodes:** Dutch speakers

Adopted from Blondel et al. 2008.

Summary: Modularity

- **Modularity:**
  - Overall quality of the partitioning of a graph into communities
  - Used to determine the number of communities

- **Louvain modularity maximization:**
  - Greedy strategy
  - Great performance, scales to large networks