Graph community detection

Lin Guosheng
School of Computer Science and Engineering
Nanyang Technological University

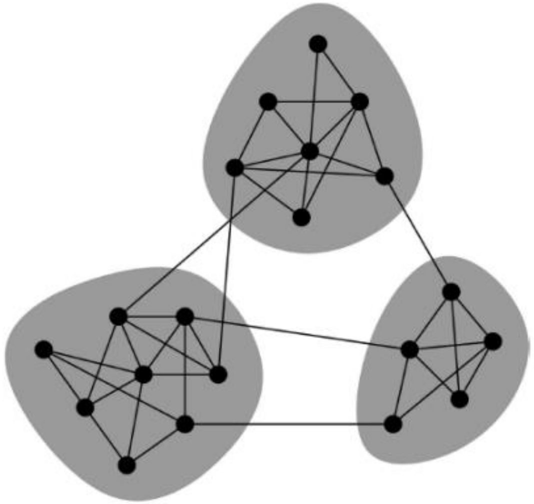
Outline

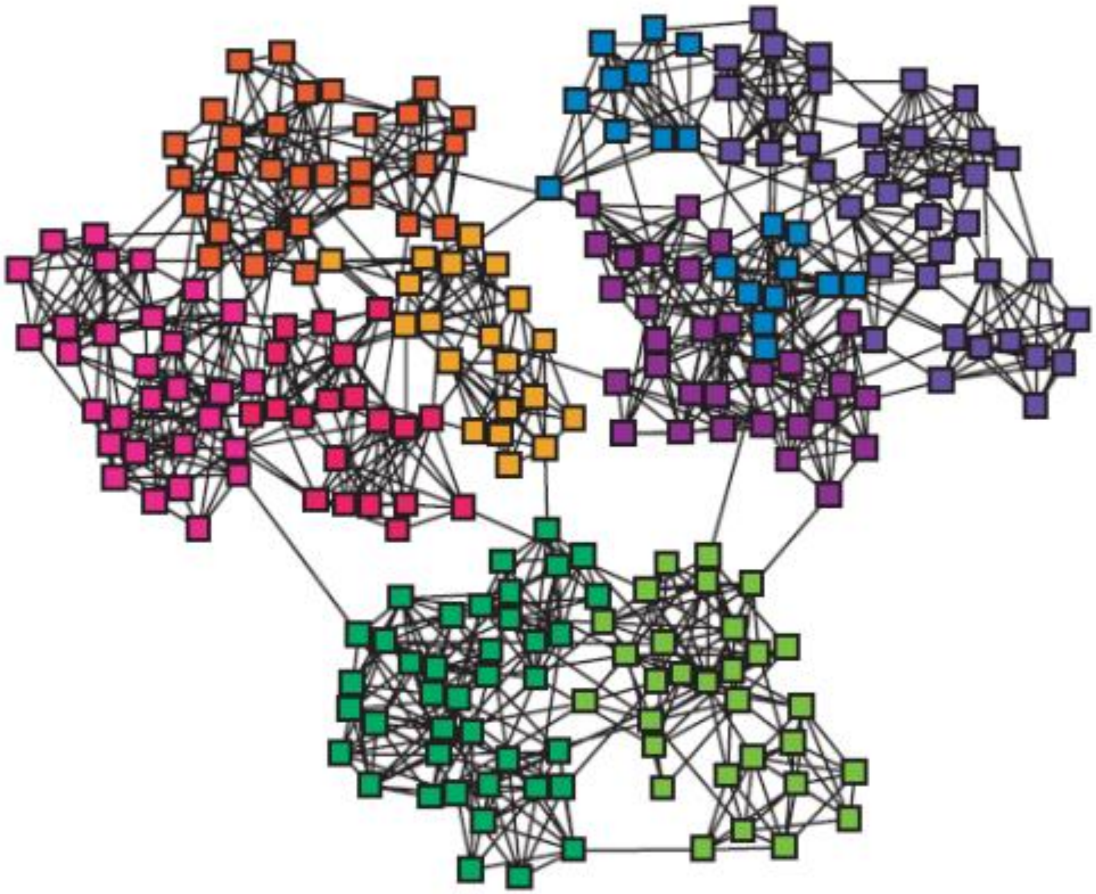
- Louvain Algorithm
  - Single pass
- Multi-pass Louvain Algorithm
  - **(non-examinable!)**

Many slides are from:
http://snap.stanford.edu/class/cs224w-2020/slides/13-communities.pdf

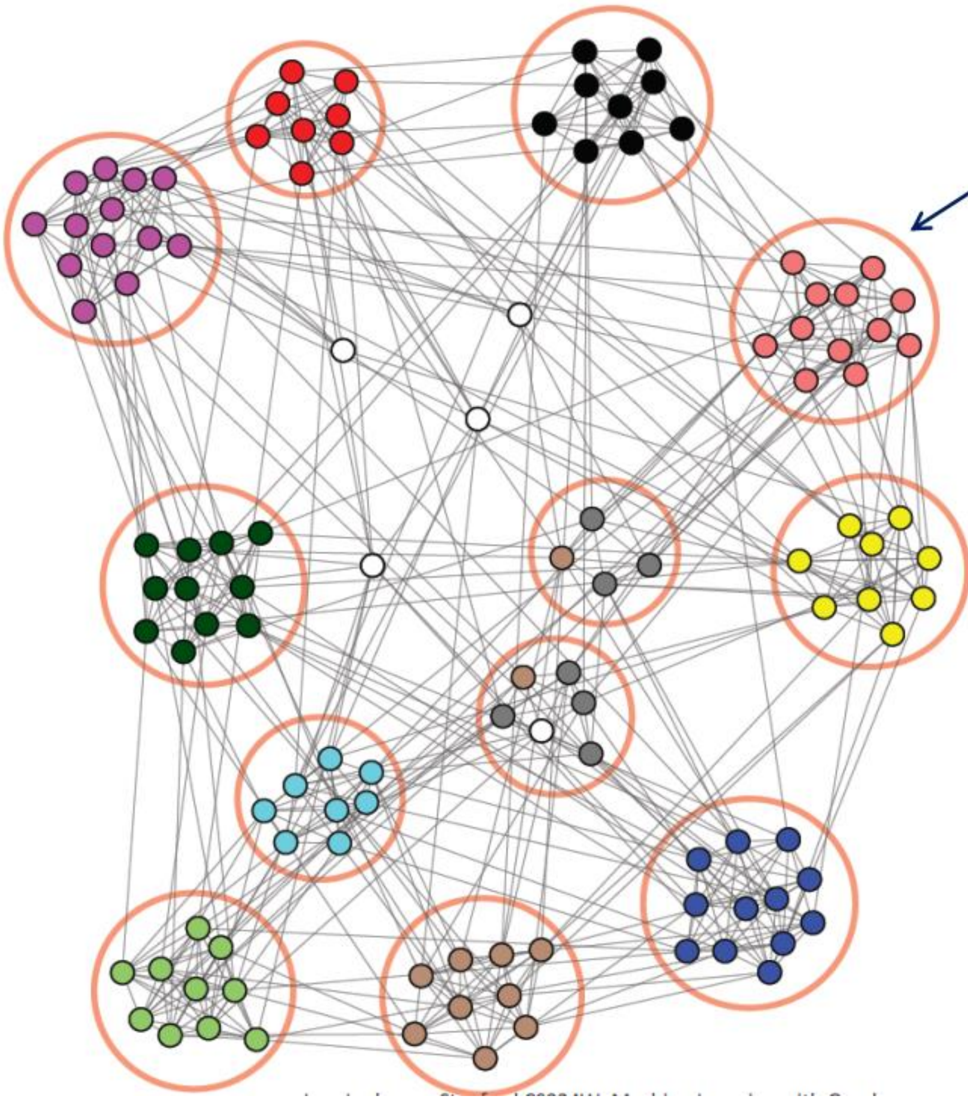
- One community in a graph:
  - A cluster of nodes
  - a group of tightly (densely) connected nodes
  - **many internal connections** and **few external connections** (to the rest of the network)
  - A community is also called: A cluster, A group, A module

- Community detection
  - A graph clustering task
  - Automatically find densely connected groups/clusters





NCAA football network



NCAA conferences

- Mid American
- Big East
- Atlantic Coast
- SEC
- Conference USA
- Big 12
- Western Athletic
- Pacific 10
- Mountain West
- Big 10
- Sun Belt
- Independents

Nodes: Teams
Edges: Games played

Louvain Algorithm

- Community detection on graphs
  - Greedy algorithm
    - Make locally optimal decision at each step
  - Supports weighted graphs
  - Provides hierarchical communities
  - Widely utilized to study large networks
    - Fast, rapid convergence
    - Produce high-quality results

Modularity score

Use this equation to calculate the modularity score for a community:

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

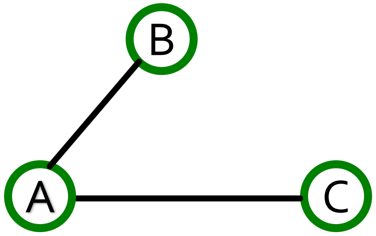
1. Modularity score: measure the quality of a community
2. We aim to maximize the total Modularity score of all communities.

$\boldsymbol{\Sigma_{in}} \equiv \sum_{i,j \in C} A_{ij}$: Sum of the weights of internal edges in the community C. (double count each edge) (high value indicates strong internal connections)

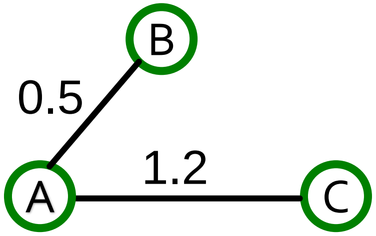
$\boldsymbol{\Sigma_{tot}} \equiv \sum_{i \in C} k_i$: Sum of the weighted degrees of all nodes in the community C

$\boldsymbol{m}$: the sum of all edge weights in the undirected graph

- Node degree
  - Un-weighted graph:
    - Node degree: the number of connected edges.
    - all edge weights equal to 1
  - Weighed graph:
    - Node degree (weighted):
      the sum of the weights of connected edges.



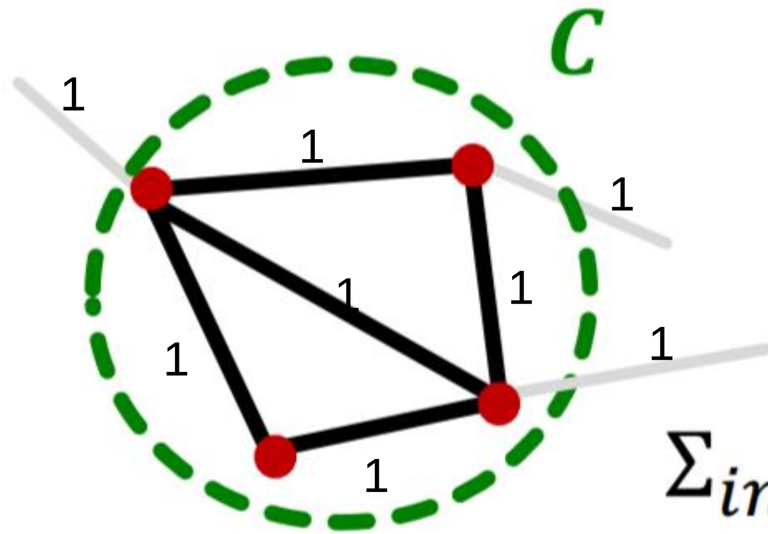
Un-weighted graph
D(A)=2



Weighted Graph
D(A)=1.7

For un-weighted graph: the weight for each edge is 1

Σ_{in}:



$$\Sigma_{in} = 10$$

The index i and j indicate the nodes in the community C

$\Sigma_{in} \equiv \sum_{i,j \in C} A_{ij}$: Sum of all internal edge weights of the community (each internal edge will be double counted in the summation)

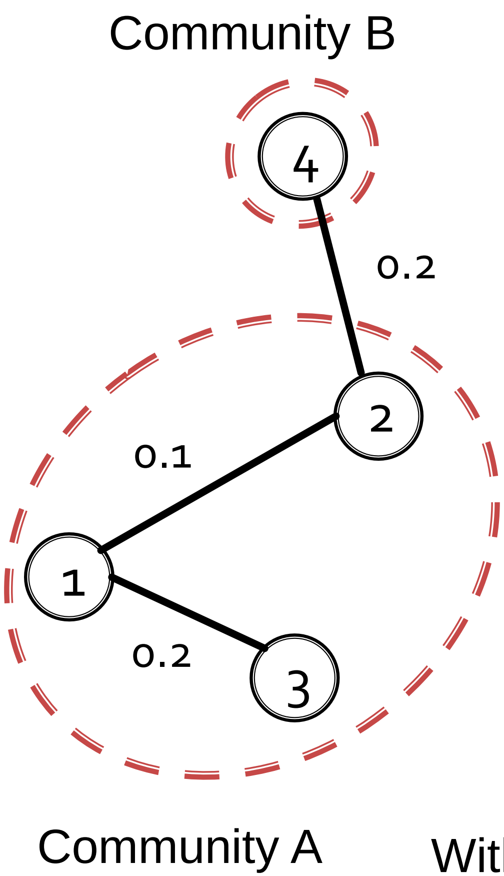
A_{ij} is the edge weight for the edge connecting the nodes (i, j)

In the undirected graph, the edge weight: $A_{i,j} = A_{j,i}$

For the example here, the internal edges will be double counted in the summation:

$$(1+1+1+1+1) \text{ x } 2=10$$

Another example



For the community A, there are three internal nodes {1,2,3}

We calculate Σ_{in} for community A:

$$\Sigma_{in} = \sum_{i,j \in C} A_{ij}$$

(i,j indicate the internal nodes)

$$= [A_{1,1} + A_{1,2} + A_{1,3}] + [A_{2,2} + A_{2,1} + A_{2,3}]$$
$$+ [A_{3,3} + A_{3,1} + A_{3,2}]$$

There is no self links: $A_{1,1} = 0; A_{2,2} = 0; A_{3,3} = 0$

There is no edge between node 2 and 3: $A_{2,3} = A_{3,2} = 0$

In the undirected graph, the edge weight: $A_{i,j} = A_{j,i}$

$$A_{1,2} = A_{2,1} = 0.1; \quad A_{1,3} = A_{3,1} = 0.2;$$

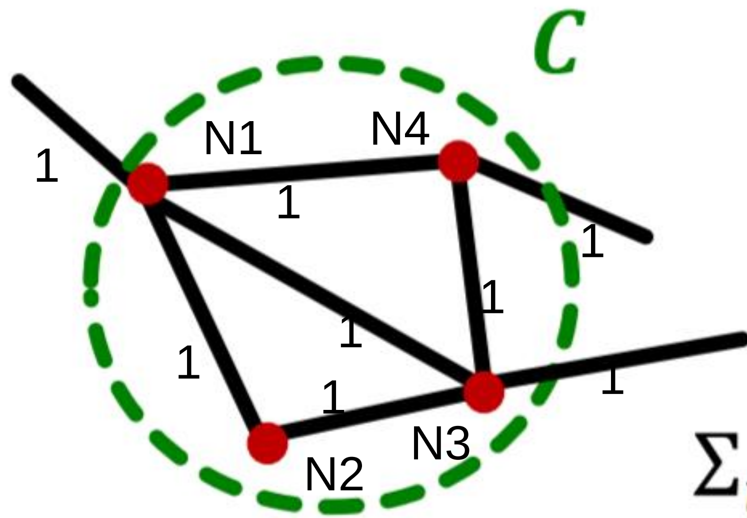
With the above, we can calculate Σ_{in} for community A as:

$$\Sigma_{in} = \sum_{i,j \in C} A_{ij} = (A_{1,2} + A_{1,3}) \times 2 = (0.1 + 0.2) \times 2 = 0.6$$

(sum the internal edge weights and multiply 2)

For un-weighted graph: the weight for each edge is 1

Σ_{tot}:



$$\Sigma_{tot} = 13$$

$\Sigma_{tot} \equiv \sum_{i \in C} k_i$: Sum of the degrees of all internal nodes in the community C.
Here k_i is the node degree of node i.

Sum of the node degrees of all internal nodes in C:

4 + 2 + 4 + 3 = 13

(for node N1, N2, N3, N4, respectively)

Example: an extreme case

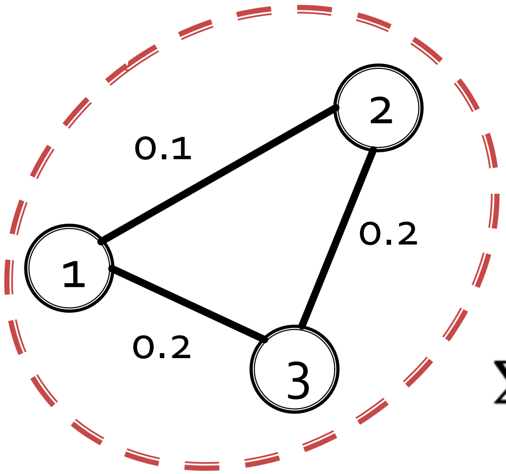
- An extreme case: all nodes belong to 1 community
  - This trivial clustering strategy provides a baseline score

$$Q(C) = \frac{\Sigma_{in}}{2m} - (\frac{\Sigma_{tot}}{2m})^2 = \frac{2m}{2m} - (\frac{2m}{2m})^2 = 0$$

We have: $\Sigma_{in} = 2m$

$$\Sigma_{tot} = 2m$$

Example for the left graph:



1 community

m : the sum of all edge weights in the undirected graph

$$m = 0.1 + 0.2 + 0.2 = 0.5$$

$$\Sigma_{in} = (0.1 + 0.2 + 0.2) \times 2 = 1$$

$$\Sigma_{tot} = (0.1 + 0.2) + (0.1 + 0.2) + (0.2 + 0.2) = 1$$

$$Q(C) = 0$$

Example: another extreme case

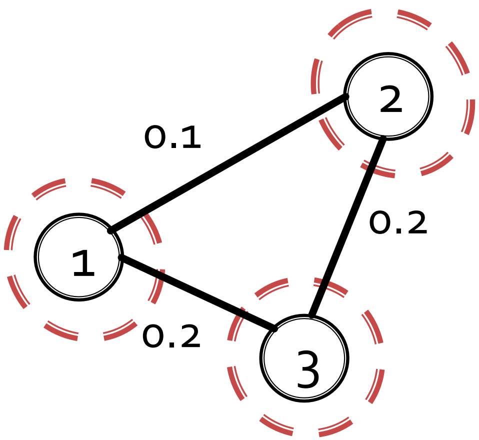
- Another extreme case: one node forms one community

$$Q(C_i) = \frac{\Sigma_{in}}{2m} - (\frac{\Sigma_{tot}}{2m})^2 = \frac{0}{2m} - (\frac{k_i}{2m})^2 = -(\frac{k_i}{2m})^2$$

We have: $\Sigma_{in} = 0$ $\qquad k_i$: the node degree of node i

$\Sigma_{tot} = k_i$

Example for the left graph:



3 communities

$$m = 0.1 + 0.2 + 0.2 = 0.5$$

$$Q(\{1\}) = -(\frac{k_1}{2m})^2 = -(\frac{0.1 + 0.2}{1})^2 = -0.09$$

$$Q(\{2\}) = -(\frac{k_2}{2m})^2 = -(\frac{0.1 + 0.2}{1})^2 = -0.09$$

$$Q(\{3\}) = -(\frac{k_3}{2m})^2 = -(\frac{0.2 + 0.2}{1})^2 = -0.16$$

Example: another extreme case

- Another extreme case: one node forms one community

The modularity score for the community formed by node i:

$$Q(C_i) = \frac{\Sigma_{in}}{2m} - (\frac{\Sigma_{tot}}{2m})^2 = \frac{0}{2m} - (\frac{k_i}{2m})^2 = -(\frac{k_i}{2m})^2$$

We have: $\Sigma_{in} = 0$

$$\Sigma_{tot} = k_i$$

k_i : the node degree of node i

The total modularity score of all communities:
(N indicates the total number of communities)

$$\sum_{i=1}^{N} Q(C_i) = -\sum_{i=1}^{N}(\frac{k_i}{2m})^2$$

The total score is less than 0

Modularity score (discussion)

What is a good community:

The nodes in the community have lots of internal connections and very few external connections (to the rest of the network)

Modularity score: measure the quality of a community

A large value indicates a lot of internal connections

$$\frac{\Sigma_{in}}{2m} \leq 1$$

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

$$\frac{\Sigma_{tot}}{2m} \leq 1$$

1. A large value indicates a large community or large node degrees.
2. This term will penalize large community and the external connections.
   - The node degree includes the internal and external links of the node.
   - A large number of external links will lead to a large node degree.
3. Intuitively, the square operation is to downgrade the impact of the second term: e.g $(0.1)^2 < 0.1$

- Louvain Algorithm
  - aims to greedily maximize the Modularity score.
  - Modularity score: a metric to measure the quality of a community

  - Single pass Louvain Algorithm
    - Community generation
      - Also called modularity optimization

Louvain Algorithm: community generation

- Initialization: each node forms a distinct community
- A: generate a random list of nodes (start one scan)
- B: (one scan) sequentially process each node in the list for community update
- Repeat A, B until converge
  - Converge:
    there is no update of the community in the last scan

- B: sequentially process each node for community update
  - 1) Node movement step.
    - Identify possible movements by finding neighboring communities (directly connected external communities)
    - For each movement: compute the modularity gain (ΔQ) as the movement score.

$$\Delta Q = Q_{\text{after}} - Q_{\text{before}}$$

  - 2) Community update step.
    - Move the node to a community that yields the largest positive score (ΔQ). If the largest score is not positive, there is no movement of the node.

Modularity gain (movement score)

Directly use this equation to calculate the modularity score for each community:

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

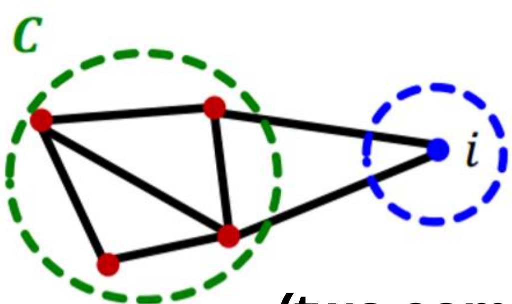
Modularity gain:

$$\Delta Q = Q_{\text{after}} - Q_{\text{before}}$$

Merging i into C

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

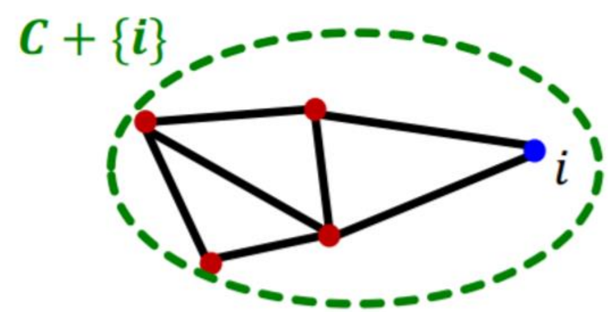
Before merging



Isolated community of node i

(two communities)

After merging



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

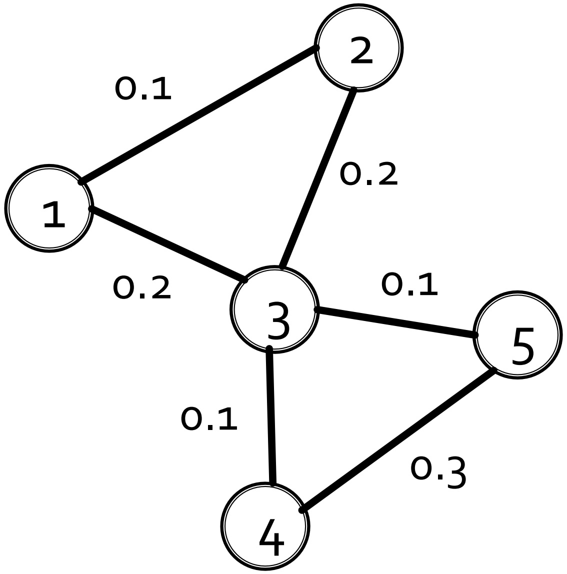
$$Q_{\text{after}} = Q(C + \{i\})$$

Modularity gain: $\Delta Q(i \to C) = Q_{\text{after}} - Q_{\text{before}}$

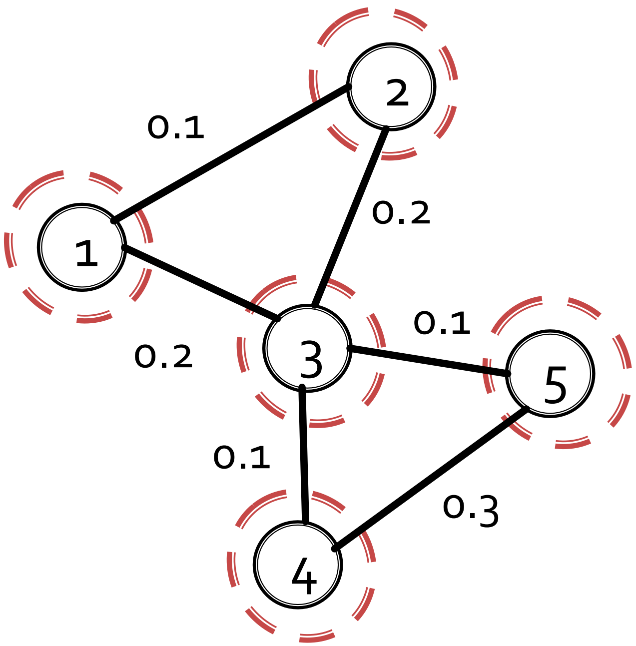
Example: community update

Q: A graph is given below.

In the initialization, each node forms a distinct community. Use Louvain algorithm to sequentially process the nodes in the order {3, 1, 2, 4, 5} to update the communities given in the initialization.



Result

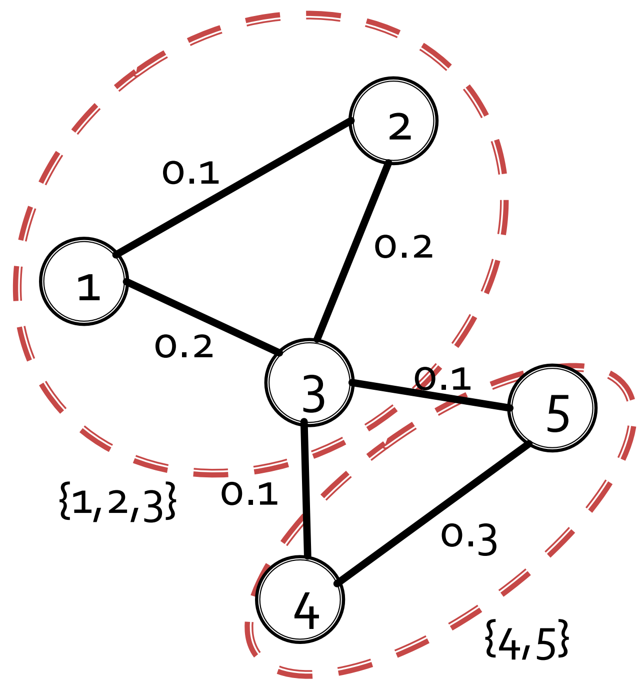


Process nodes:

{3, 1, 2, 4, 5}

to update

communities



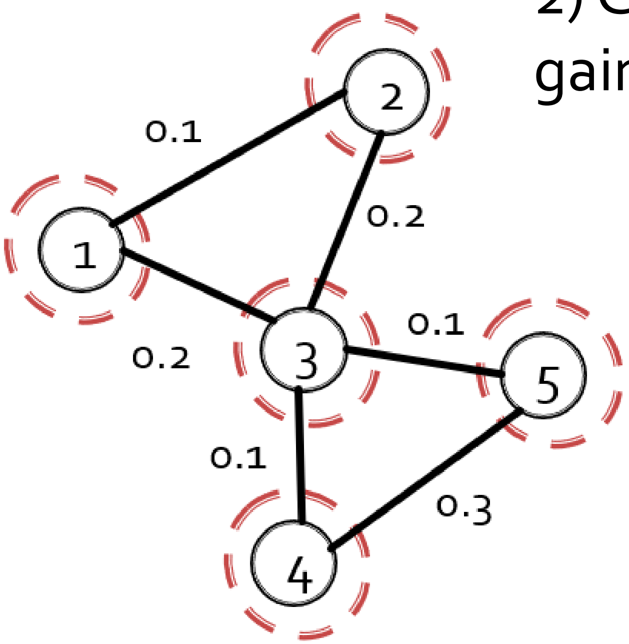
In the initialization,
each node forms a distinct community

After processing,
the communities are updated

Intermediate steps

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 3



1) Identify neighbouring communities of node 3: {1}, {2}, {4} and {5}

2) Calculate the movement scores (modularity gains) for the following 4 movements:

$$\Delta Q(3 \rightarrow \{1\})$$

$$\Delta Q(3 \rightarrow \{2\})$$

$$\Delta Q(3 \rightarrow \{4\})$$

$$\Delta Q(3 \rightarrow \{5\})$$

Intermediate steps

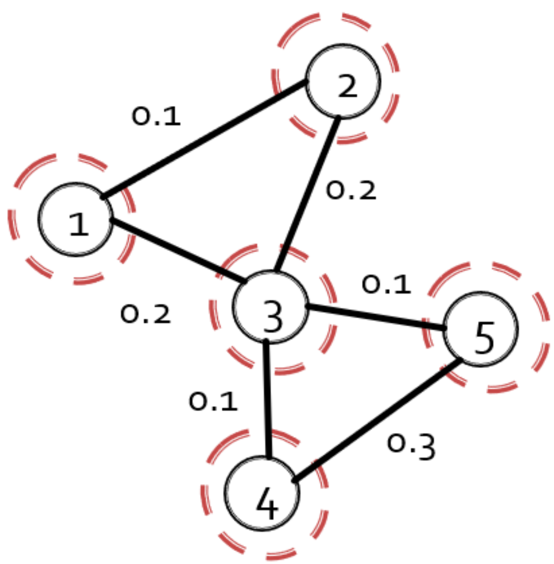
$$Q(C) = \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2$$

Sequentially process the node list {3, 1, 2, 4, 5}

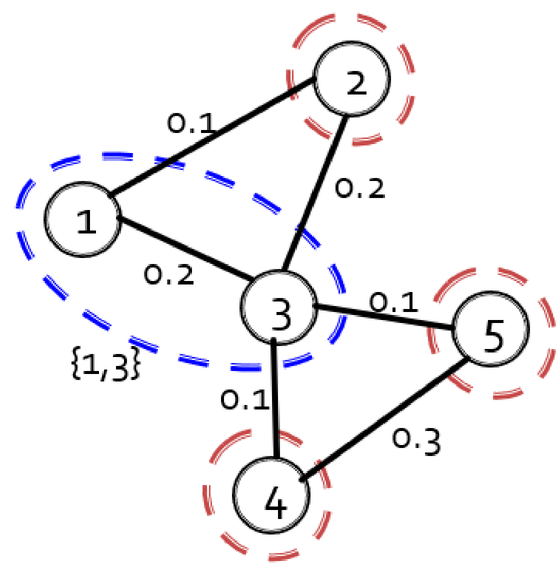
processing node 3

Movement score (Modularity gain) of moving node 3 to community {1}:

$$\begin{aligned}\Delta Q(3 \to \{1\}) &= Q_{after} - Q_{before} \\ &= Q(\{1\} + \{3\}) - [Q(\{1\}) + Q(\{3\})]\end{aligned}$$



Q_{before}



Q_{after}

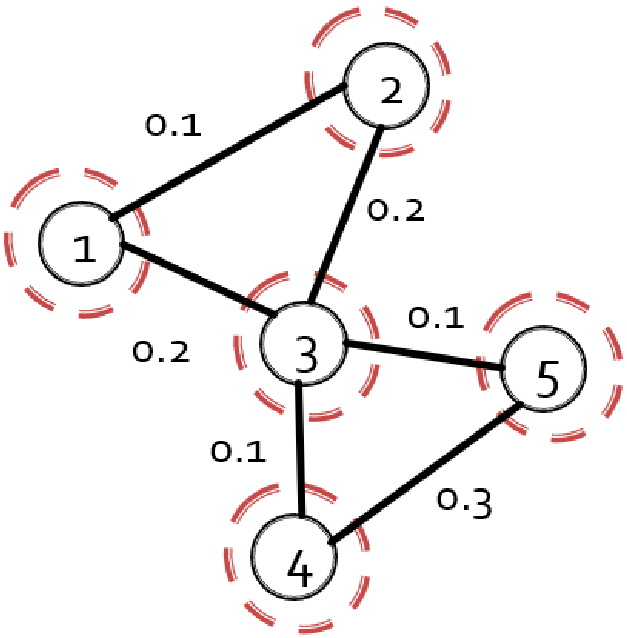
Intermediate steps

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

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 3

$$\begin{aligned} \Delta Q(3 \to \{1\}) &= Q_{after} - Q_{before} \\ &= Q(\{1\} + \{3\}) - [Q(\{1\}) + Q(\{3\})] \end{aligned}$$



Q_{before}

k_1= 0.1+0.2 =0.3; k_3= 0.2+0.2+0.1+0.1 =0.6;

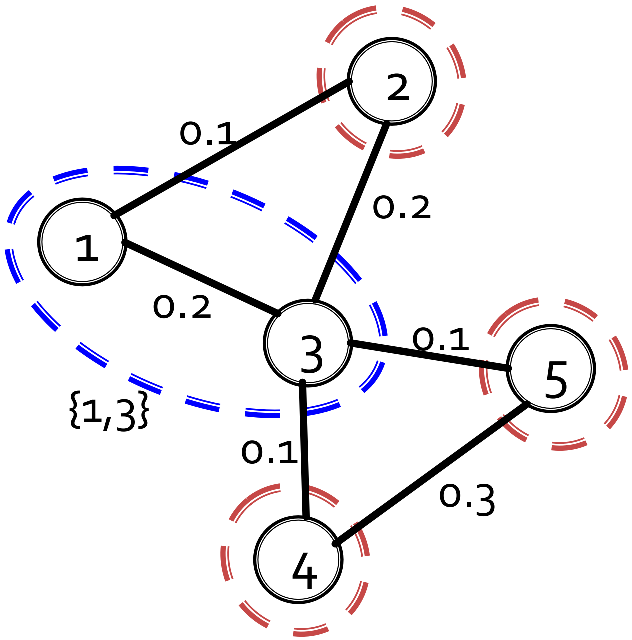
$$m = 0.1 + 0.2 + 0.2 + 0.1 + 0.1 + 0.3 = 1$$

$$\begin{aligned} Q(\{1\}) &= \left[0 - \left(\frac{k_1}{2m}\right)^2\right] \\ &= -0.0225 \end{aligned}$$

$$\begin{aligned} Q(\{3\}) &= \left[0 - \left(\frac{k_3}{2m}\right)^2\right] \\ &= -0.09 \end{aligned}$$

processing node 3

$$\Delta Q(3 \rightarrow \{1\}) = Q_{after} - Q_{before}$$
$$= Q(\{1\} + \{3\}) - [Q(\{1\}) + Q(\{3\})]$$

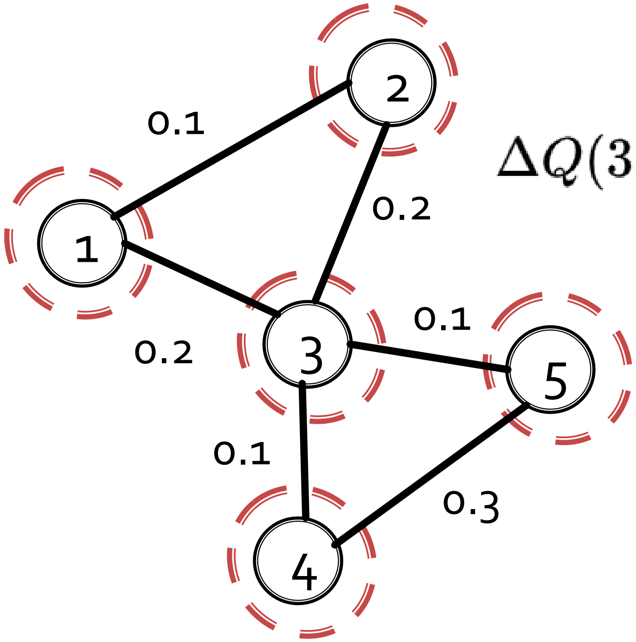


$$
\begin{aligned}
Q(\{1,3\}) &= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \\
&= \frac{2 \times e_{31}}{2m} - \left(\frac{k_1 + k_3}{2m}\right)^2 \\
&= \frac{2 \times 0.2}{2} - \left(\frac{0.3 + 0.6}{2}\right)^2 \\
&= -0.0025
\end{aligned}
$$

Q_{after}

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 3



Movement score (Modularity gain) of moving node 3 to community {1}

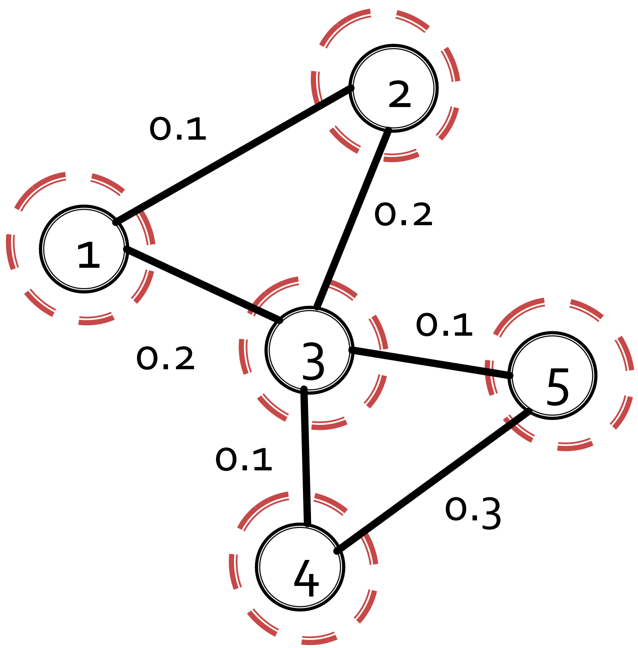
$$\begin{aligned}\Delta Q(3 \to \{1\}) &= Q_{after} - Q_{before} \\ &= Q(\{1\} + \{3\}) - \left[Q(\{1\}) + Q(\{3\})\right] \\ &= Q(\{1,3\}) - \left[Q(\{1\}) + Q(\{3\})\right] \\ &= -0.0025 - (-0.0225 - 0.09) \\ &= 0.11\end{aligned}$$

Sequentially process the node list {3, 1, 2, 4, 5}

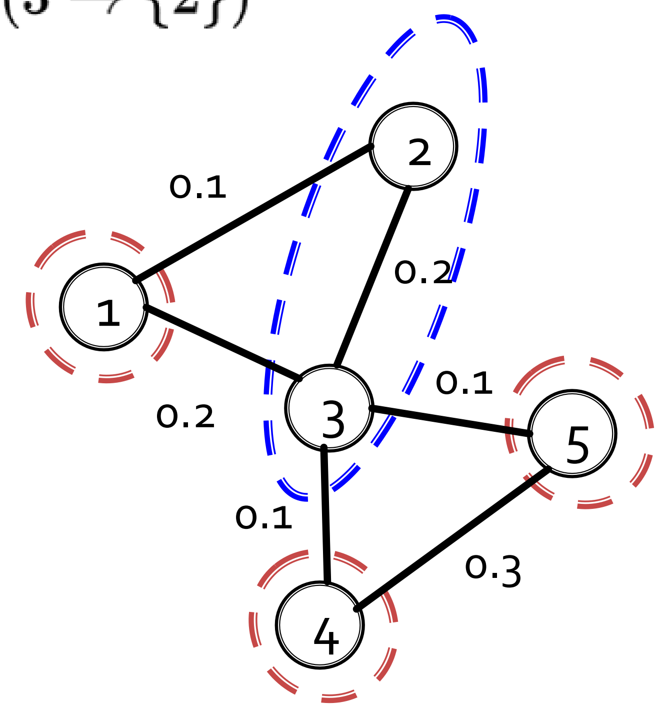
processing node 3

Movement score (Modularity gain) of moving node 3 to community {2}

$$\Delta Q(3 \rightarrow \{2\})$$



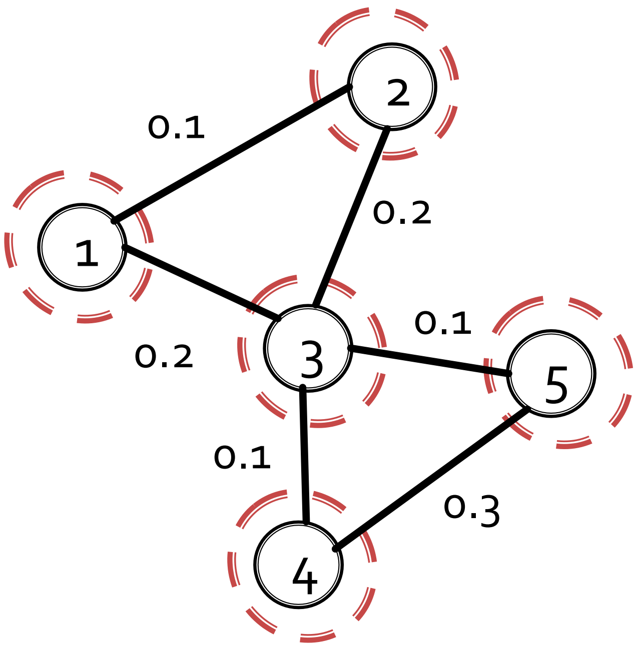
Q_{before}



Q_{after}

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 3



Movement score (Modularity gain) of moving node 3 to community {2}

The link (1,3) and (2,3) have the same weight.

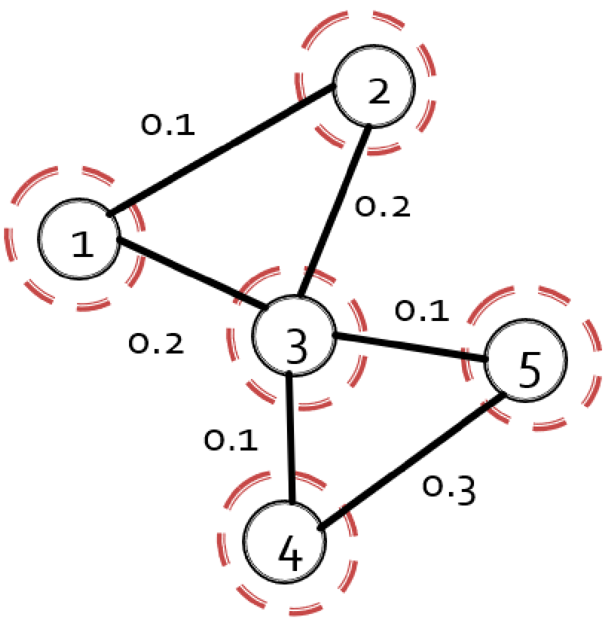
The gain of moving node 3 to {2} is the same as $\Delta Q(3 \to \{1\})$

$$\Delta Q(3 \to \{2\}) \quad = \quad 0.11$$

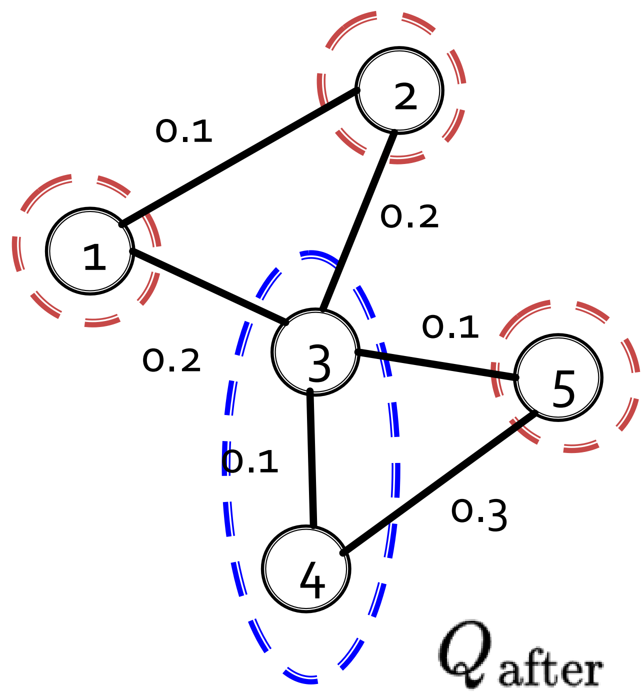
Sequentially process the node list {3, 1, 2, 4, 5}

processing node 3 The movement score of moving node 3 to {4}:

$$\begin{aligned}\Delta Q(3 \rightarrow \{4\}) &= Q_{after} - Q_{before} \\ &= Q(\{4,3\}) - \left[Q(\{4\}) + Q(\{3\})\right]\end{aligned}$$



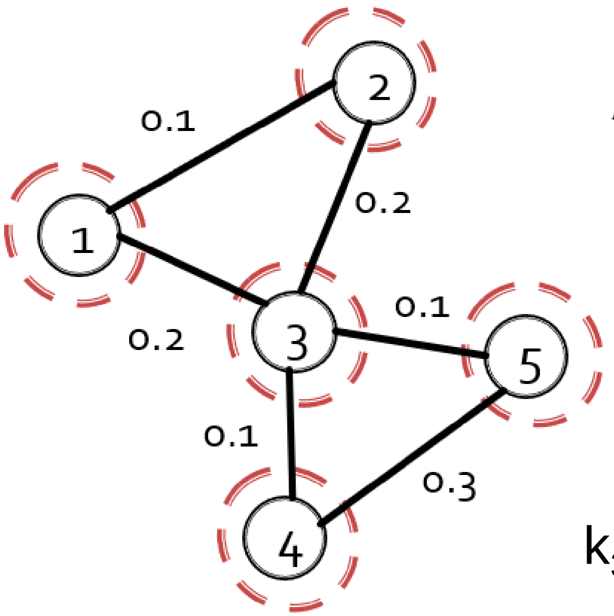
Q_{before}



Q_{after}

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 3



The movement score of moving node 3 to {4}:

$$\begin{aligned}
\Delta Q(3 \rightarrow \{4\}) &= Q_{after} - Q_{before} \\
&= Q(\{4\} + \{3\}) - \left[Q(\{4\}) + Q(\{3\})\right] \\
&= Q(\{4, 3\}) - \left[Q(\{4\}) + Q(\{3\})\right] \\
&= -0.15 - (-0.04 - 0.09) \\
&= -0.02
\end{aligned}$$

k_3= 0.2+0.2+0.1+0.1 =0.6; k_4= 0.1+0.3 =0.4

$$\begin{aligned}
Q(\{4, 3\}) &= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \\
&= \frac{2 \times e_{34}}{2m} - \left(\frac{k_4 + k_3}{2m}\right)^2 \\
&= \frac{2 \times 0.1}{2} - \left(\frac{0.4 + 0.6}{2}\right)^2 \\
&= -0.15
\end{aligned}$$

$$\begin{aligned}
Q(\{3\}) &= \left[0 - \left(\frac{k_3}{2m}\right)^2\right] \\
&= -0.09
\end{aligned}$$

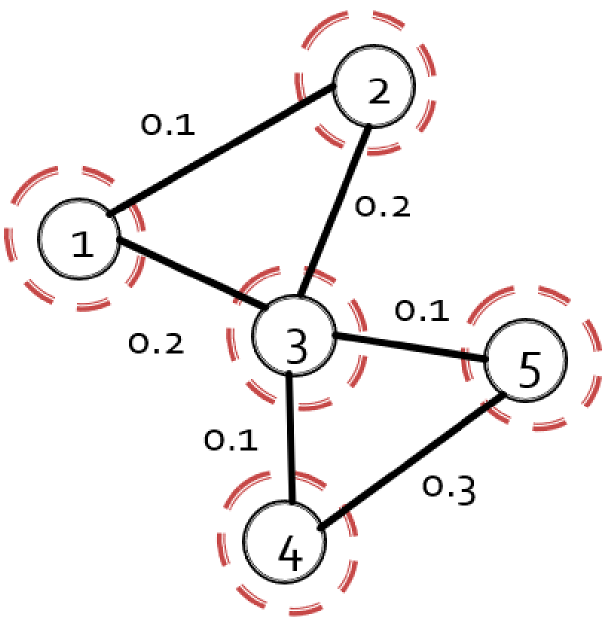
$$\begin{aligned}
Q(\{4\}) &= \left[0 - \left(\frac{k_4}{2m}\right)^2\right] \\
&= -0.04
\end{aligned}$$

Sequentially process the node list {3, 1, 2, 4, 5}

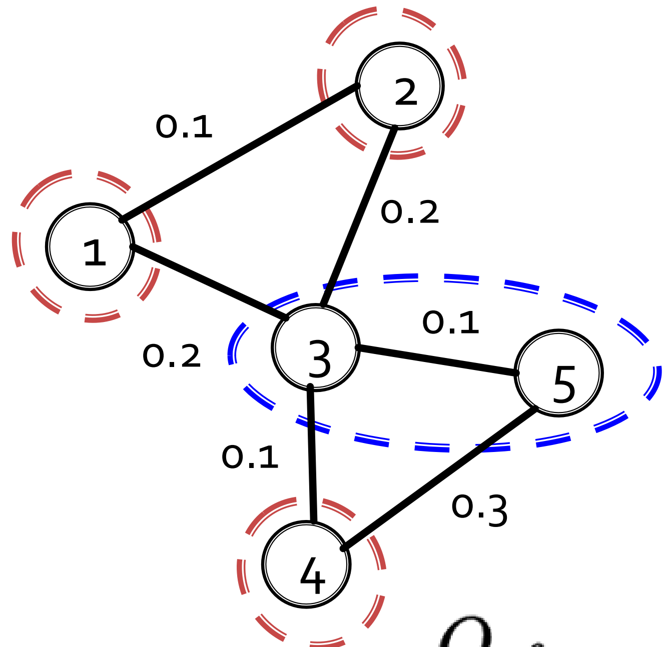
processing node 3

The movement score of moving node 3 to {5}:

$$\Delta Q(3 \rightarrow \{5\})$$



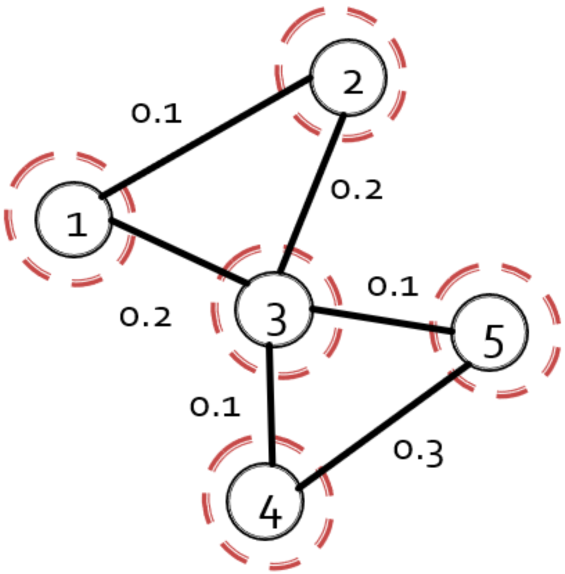
Q_{before}



Q_{after}

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 3



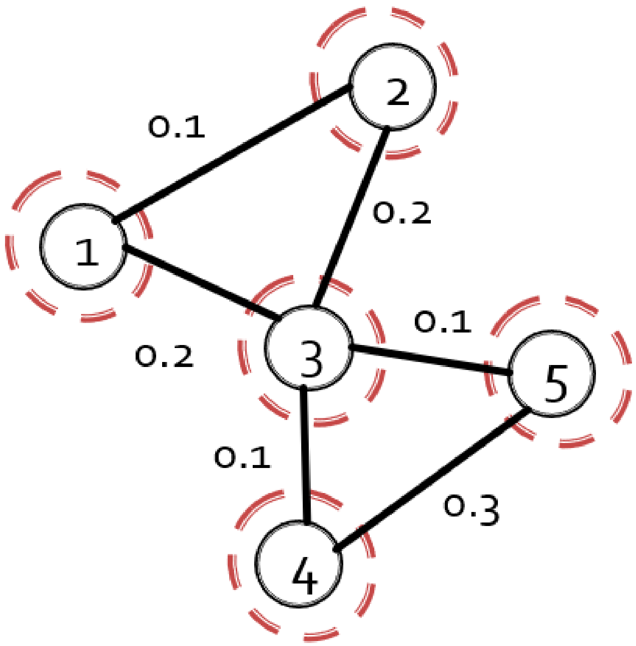
The movement score of moving node 3 to {5}:

The gain of moving node 3 to {5} is the same as: $\Delta Q(3 \rightarrow \{4\})$

$$\Delta Q(3 \rightarrow \{5\}) \quad = \quad -0.02$$

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 3



Summary:
Scores for all possible movements:

$$\Delta Q(3 \rightarrow \{1\}) = \Delta Q(3 \rightarrow \{2\}) = 0.11$$

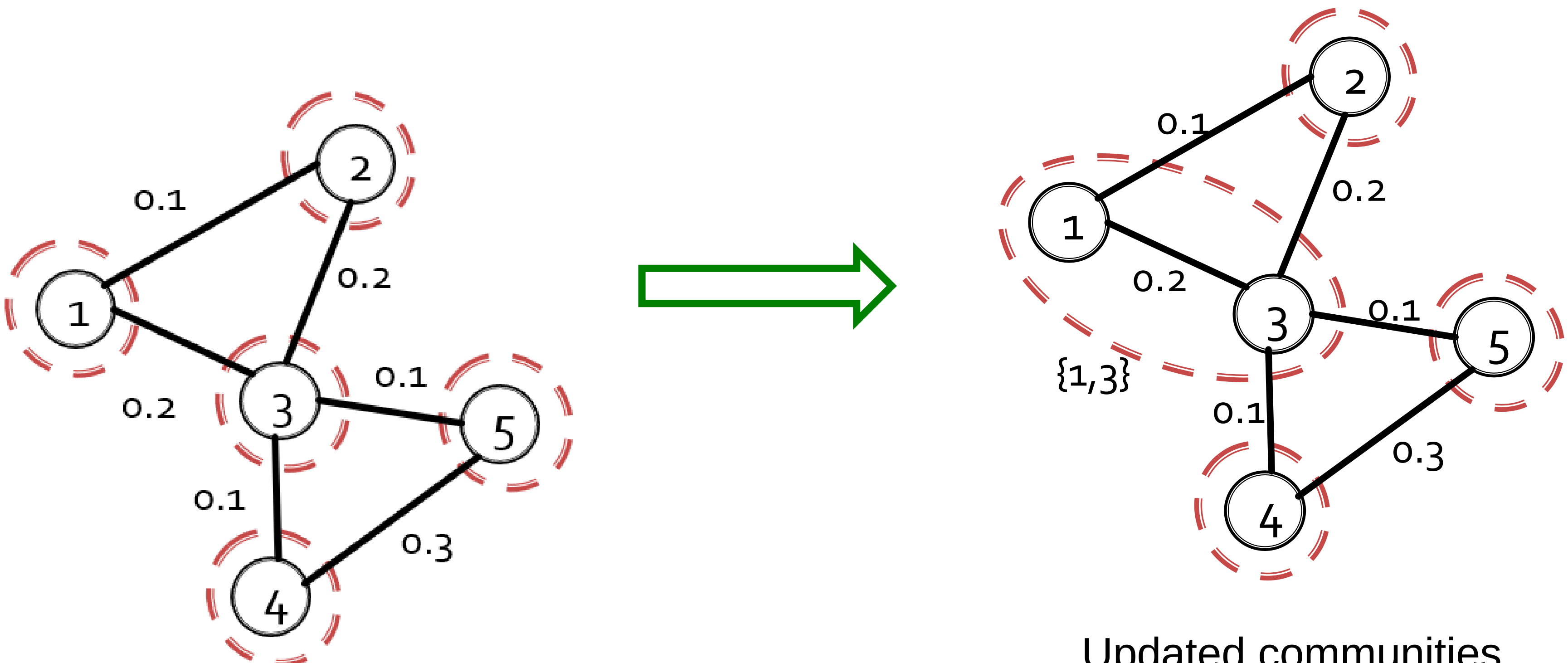
$$\Delta Q(3 \rightarrow \{4\}) = \Delta Q(3 \rightarrow \{5\}) = -0.02$$

There is a tie for 3->{1} and 3->{2}.
Randomly choose one community to proceed.

Here we choose {1}.
Move node #3 to community {1},
now we have {1,3}.

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 3



Updated communities

After processing node 3, we move node 3 to {1}.

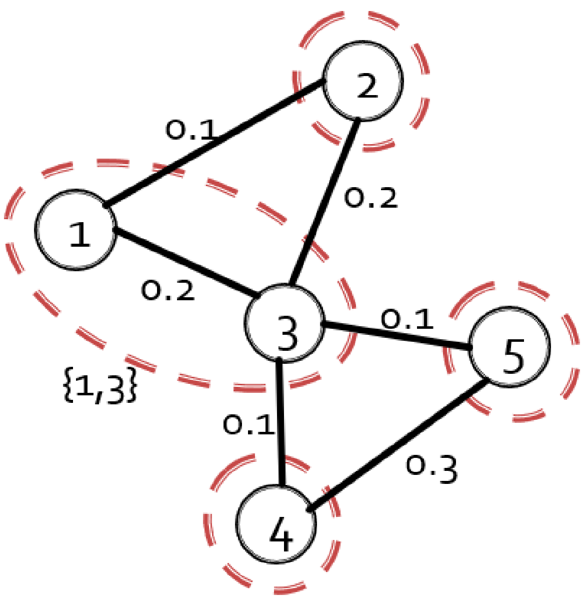
Sequentially process the node list {3, 1, 2, 4, 5}

processing node 1

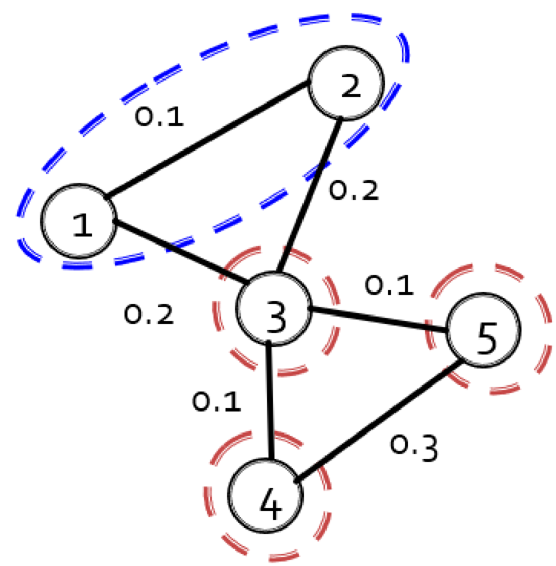
The neighbouring (directly connected) communities of node 1: Community {2}
Only one possible movement:

Move node 1 out from {1, 3} and add node 1 to {2}:

$$\Delta Q(\{1,3\} \to 1 \to \{2\}) = Q_{\text{after}} - Q_{\text{before}}$$



$$Q_{\text{before}} = Q(\{1,3\}) + Q(\{2\})$$



$$Q_{\text{after}} = Q(\{3\}) + Q(\{1,2\})$$

Sequentially process the node list {3, 1, 2, 4, 5}

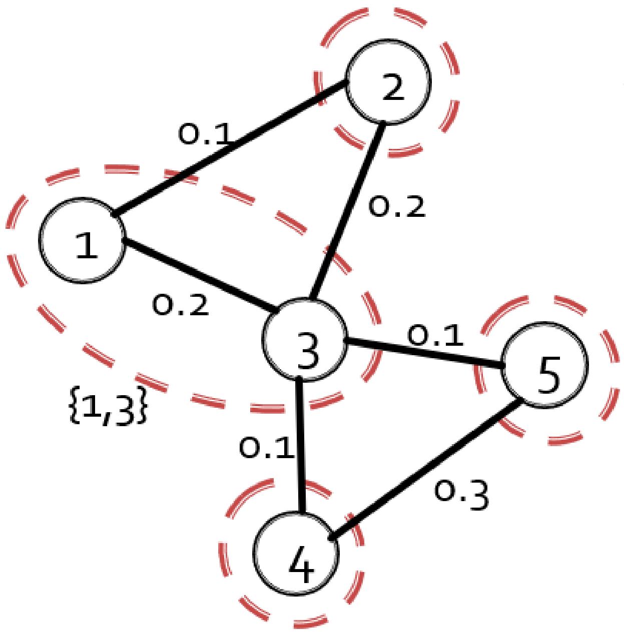
processing node 1

The neighbouring (directly connected) communities of node 1: Community {2}
Only one possible movement:

Move node 1 out from {1, 3} and add node 1 to {2}:

$$\Delta Q(\{1,3\} \rightarrow 1 \rightarrow \{2\}) = Q_{\text{after}} - Q_{\text{before}}$$

$$Q_{\text{before}} = Q(\{1,3\}) + Q(\{2\})$$



Current communities

From the previous steps,
we already have the following values:

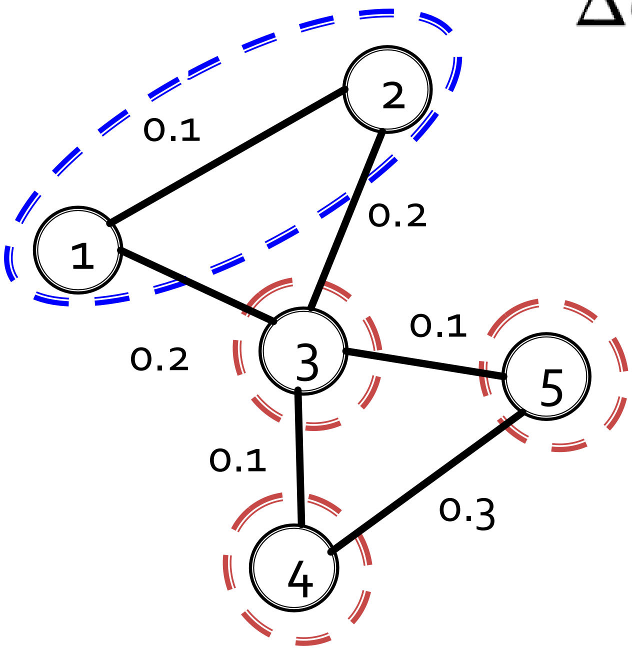
$$Q(\{1,3\}) = -0.0025$$

$$Q(\{2\}) = Q(\{1\}) = -0.0225$$

processing node 1

Move node 1 out from {1, 3} and add node 1 to {2}:

$$\Delta Q(\{1,3\} \to 1 \to \{2\}) = Q_{\text{after}} - Q_{\text{before}}$$



$$Q_{\text{after}} = Q(\{3\}) + Q(\{1,2\})$$

$$\begin{aligned} Q(\{1,2\}) &= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \\ &= \frac{2\times e_{12}}{2m} - \left(\frac{k_2+k_1}{2m}\right)^2 \\ &= \frac{2\times 0.1}{2} - \left(\frac{0.3+0.3}{2}\right)^2 \\ &= 0.01 \end{aligned}$$

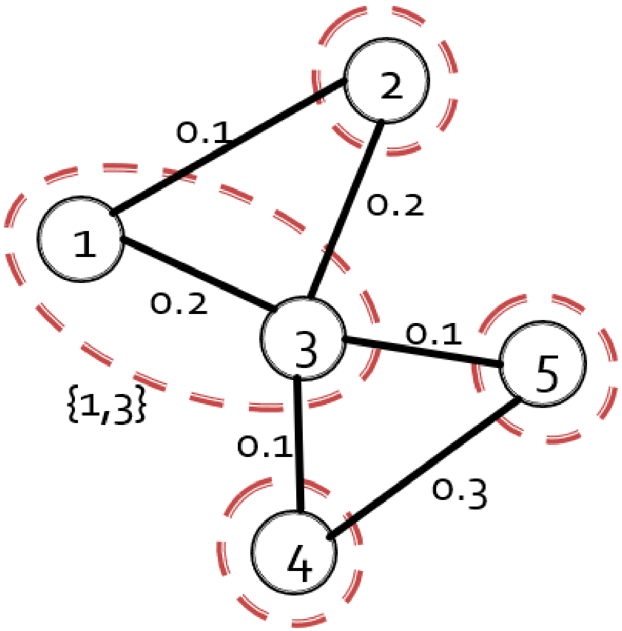
$$Q_{\text{after}} = Q(\{3\}) + Q(\{1,2\})$$

From the previous steps,
we already have the following values:

$$Q(\{3\}) = -0.09$$

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 1



Current communities

Move node 1 out from {1, 3} and add node 1 to {2}:

$$\Delta Q(\{1,3\} \to 1 \to \{2\}) = Q_{\text{after}} - Q_{\text{before}}$$
$$= Q(\{3\}) + Q(\{1,2\}) - (Q(\{1,3\}) + Q(\{2\}))$$
$$= -0.09 + 0.01 - (-0.0025 - 0.0225)$$
$$= -0.055$$

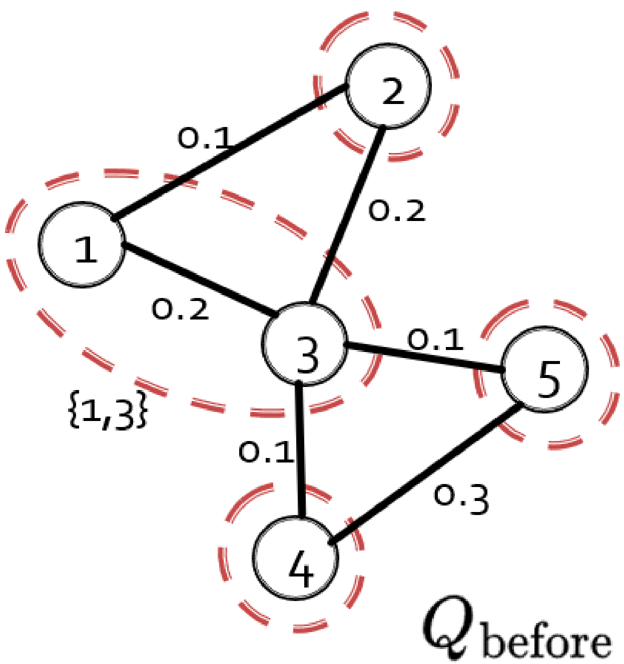
The largest gain is a negative value (there is only one neighboring community), do not move node 1 to {2}

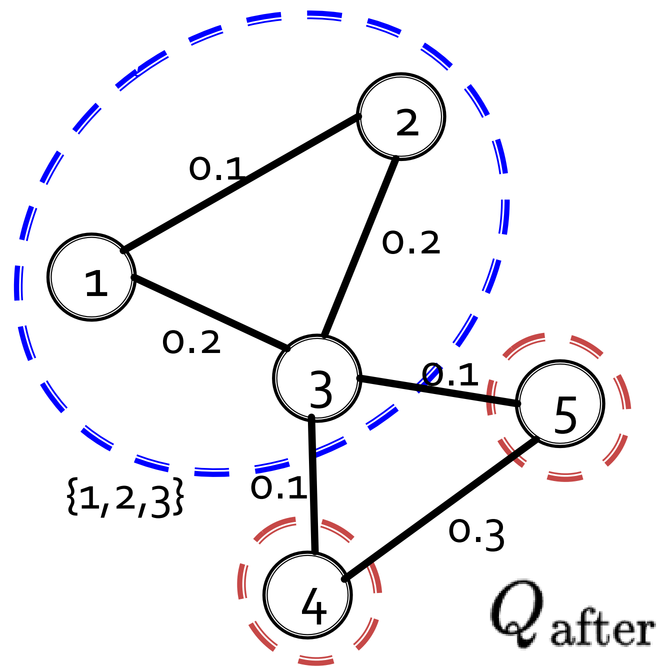
Sequentially process the node list {3, 1, 2, 4, 5}

processing node 2

The neighbouring communities of node 2:
Community {1, 3}. Only one possible movement:

$$\begin{aligned}\Delta Q(2 \rightarrow \{1,3\}) &= Q_{after} - Q_{before} \\ &= Q(\{1,2,3\}) - Q(\{1,3\}) - Q(\{2\}))\end{aligned}$$

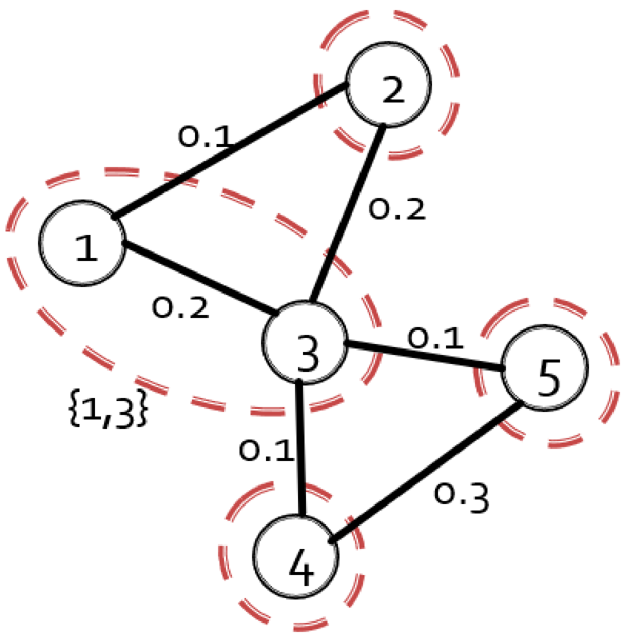




Sequentially process the node list {3, 1, 2, 4, 5}

processing node 2

The neighbouring communities of node 2: Community {1, 3}
Only one possible movement:



Current communities

$$\Delta Q(2 \to \{1,3\}) \quad = \quad Q_{after} - Q_{before}$$
$$= \quad Q(\{1,3\} + \{2\}) - \left[Q(\{1,3\}) + Q(\{2\})\right]$$
$$= \quad Q(\{1,2,3\}) - Q(\{1,3\}) - Q(\{2\}))$$

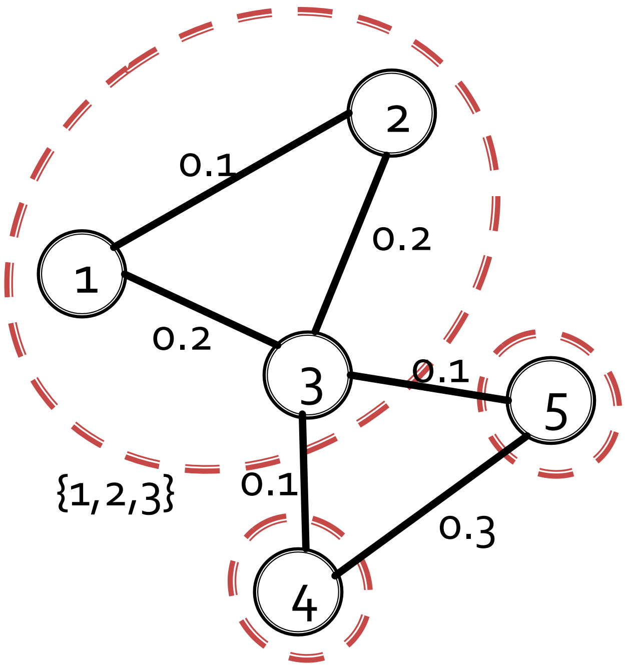
$$\begin{aligned} Q(\{1,2,3\}) &= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \\ &= \frac{2\times(e_{13}+e_{21}+e_{23})}{2m} - \left(\frac{k_1+k_2+k_3}{2m}\right)^2 \\ &= 0.14 \end{aligned}$$

$$\begin{aligned} Q(\{2\}) &= -\left(\frac{k_2}{2m}\right)^2 \\ &= -0.0225 \end{aligned}$$

$$\begin{aligned} Q(\{1,3\}) &= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \\ &= \frac{2\times e_{13}}{2m} - \left(\frac{k_1+k_3}{2m}\right)^2 \\ &= -0.0025 \end{aligned}$$

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 2



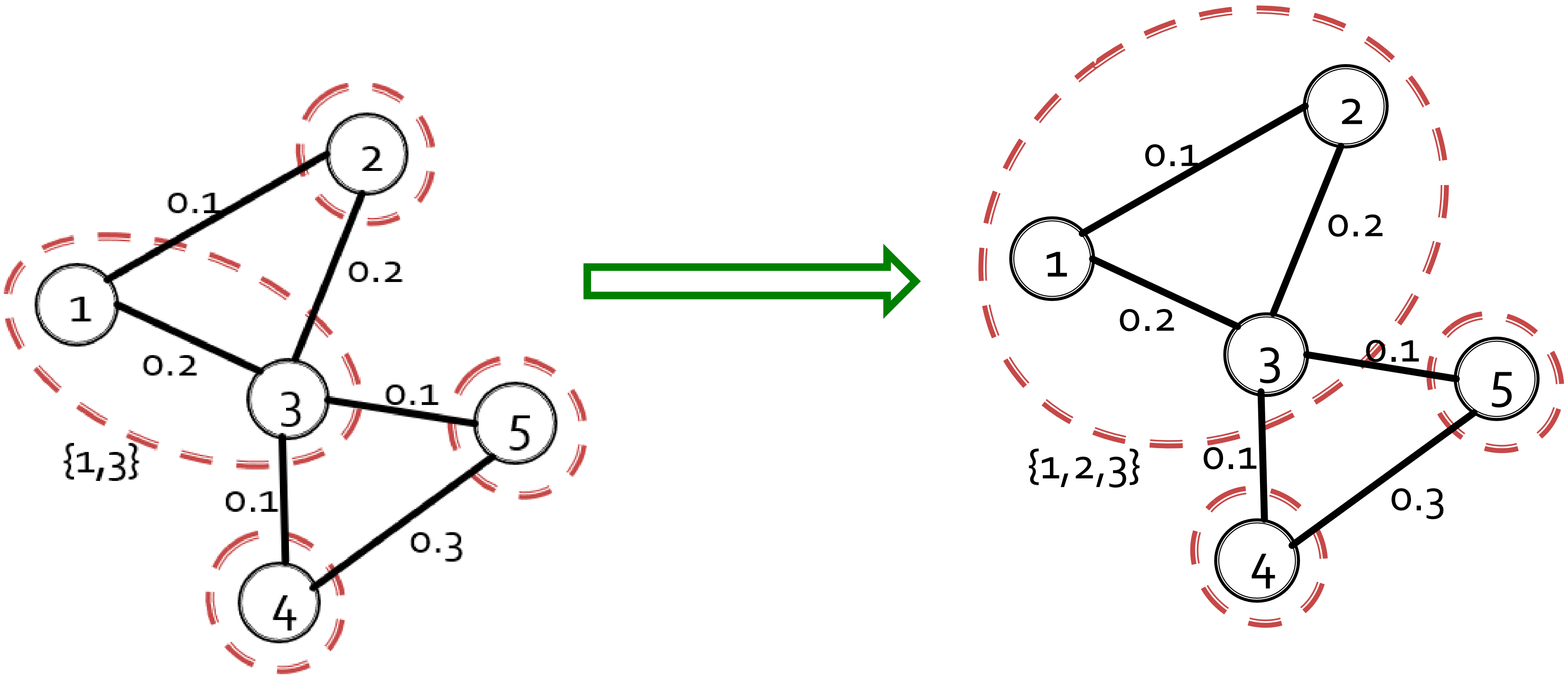
Updated communities after processing node 2

$$
\begin{aligned}
\Delta Q(2 \rightarrow \{1,3\}) &= Q_{after} - Q_{before} \\
&= Q(\{1,3\} + \{2\}) - [Q(\{1,3\}) + Q(\{2\})] \\
&= Q(\{1,2,3\}) - Q(\{1,3\}) - Q(\{2\})) \\
&= 0.165
\end{aligned}
$$

The gain is a positive value, and there is only 1 neighboring community, so it's the largest positive gain. We move 2 to {1,3}.

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 2

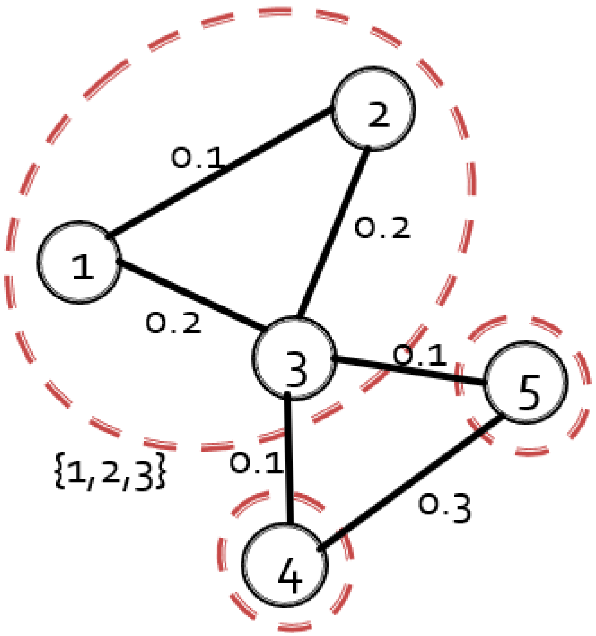


Updated communities

After processing node 2, we move node 2 to {1,3}.

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 4



Current communities

The neighbouring communities of node 4:
1. Community {1, 2, 3}
2. Community {5}

There are 2 possible movements.
We need to calculate the following two modularity gains:

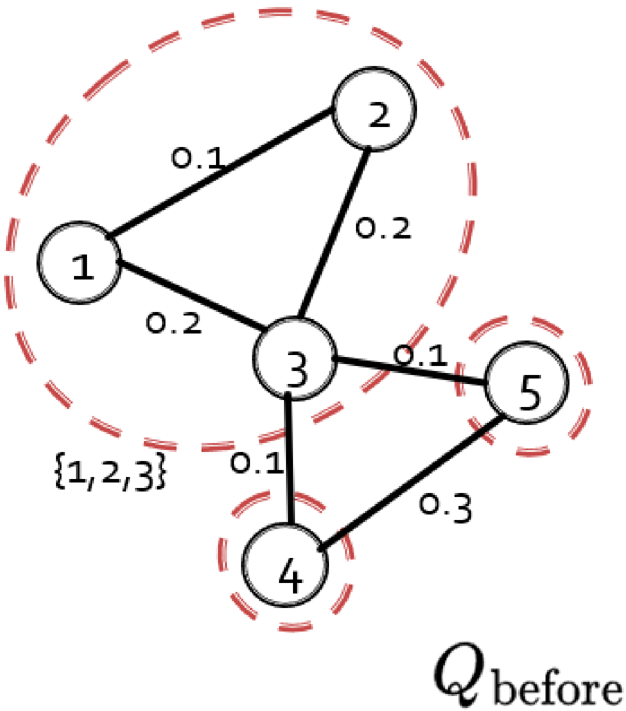
$$\Delta Q(4 \rightarrow \{1,2,3\})$$

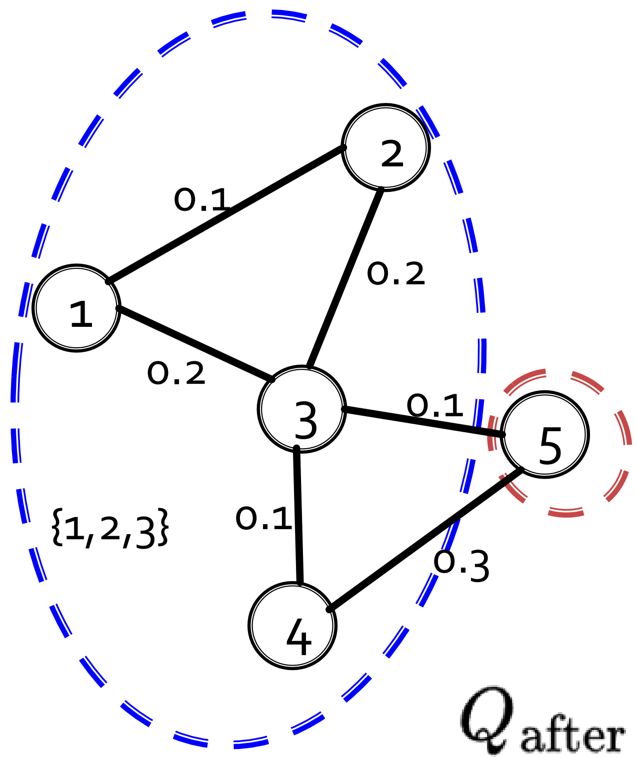
$$\Delta Q(4 \rightarrow \{5\})$$

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 4

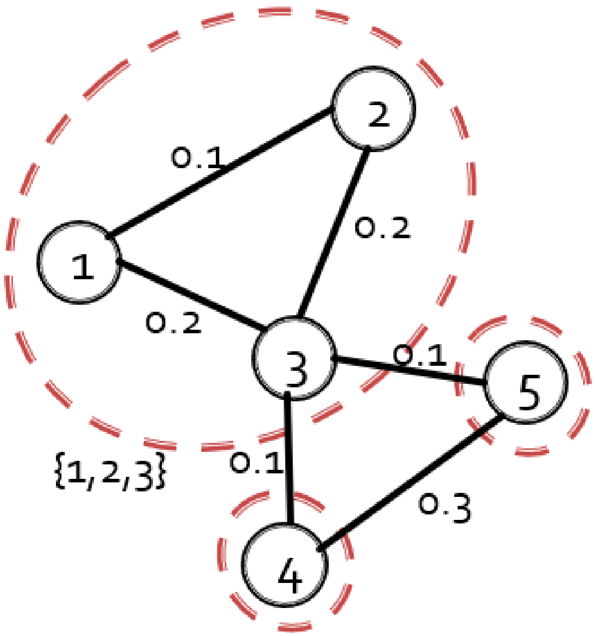
$$\Delta Q(4 \rightarrow \{1,2,3\}) \;=\; Q_{after} - Q_{before}$$





Sequentially process the node list {3, 1, 2, 4, 5}

processing node 4



Current communities

$$\Delta Q(4 \rightarrow \{1,2,3\}) = Q_{after} - Q_{before}$$

$$= Q(\{4\} + \{1,2,3\}) - [Q(\{4\}) + Q(\{1,2,3\})]$$

$$= Q(\{1,2,3,4\}) - Q(\{4\}) - Q(\{1,2,3\})$$

$$= (-0.04) - (-0.04) - 0.14$$

$$= -0.14$$

k_4= 0.1+0.3 =0.4

$$Q(\{4\}) = -(\frac{k_4}{2m})^2 = -0.04$$

$$Q(\{1,2,3\}) = 0.14$$

$$Q(\{1,2,3,4\}) = \frac{\Sigma_{in}}{2m} - (\frac{\Sigma_{tot}}{2m})^2$$

$$= \frac{2\times(e_{12}+e_{13}+e_{23}+e_{34})}{2m} - (\frac{k_1+k_2+k_3+k_4}{2m})^2$$

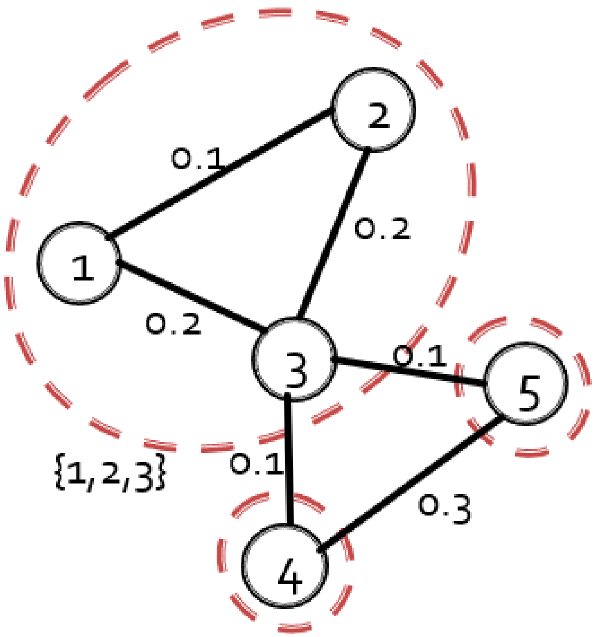
$$= \frac{2\times(0.1+0.2+0.2+0.1)}{2} - (\frac{0.3+0.3+0.6+0.4}{2})^2$$

$$= -0.04$$

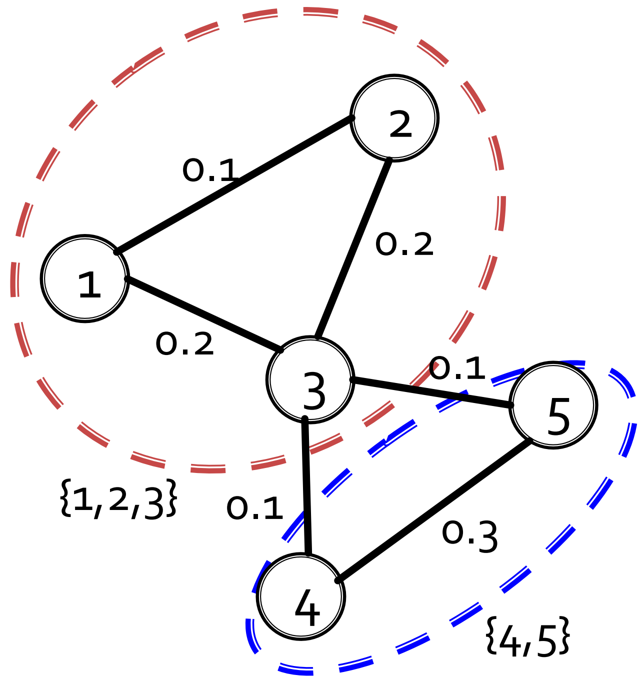
Sequentially process the node list {3, 1, 2, 4, 5}

processing node 4

$$\Delta Q(4 \rightarrow \{5\}) = \quad Q_{after} - Q_{before}$$



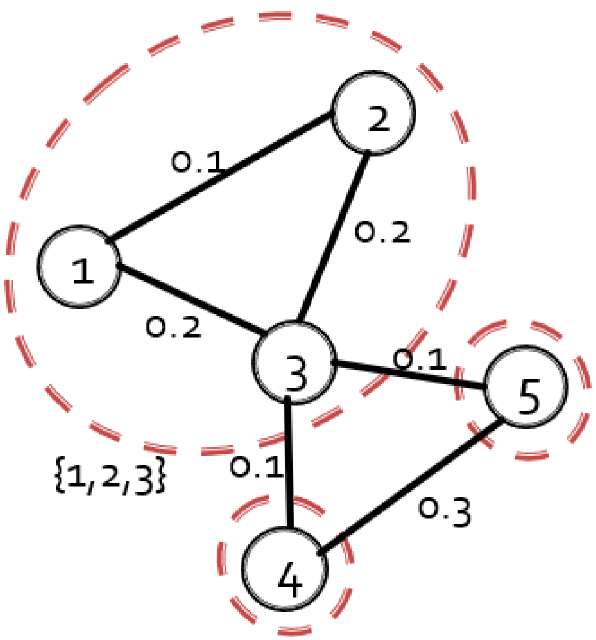
Q_{before}



Q_{after}

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 4



Current communities

$$
\begin{aligned}
\Delta Q(4 \rightarrow \{5\}) =\ & Q_{after} - Q_{before} \\
=\ & Q(\{4\}+\{5\}) - [Q(\{4\}) + Q(\{5\})] \\
=\ & Q(\{4,5\}) - Q(\{4\}) - Q(\{5\}) \\
=\ & 0.14 - (-0.04) - (-0.04) \\
=\ & 0.22
\end{aligned}
$$

$$Q(\{4\}) = -\left(\frac{k_4}{2m}\right)^2 = -0.04$$

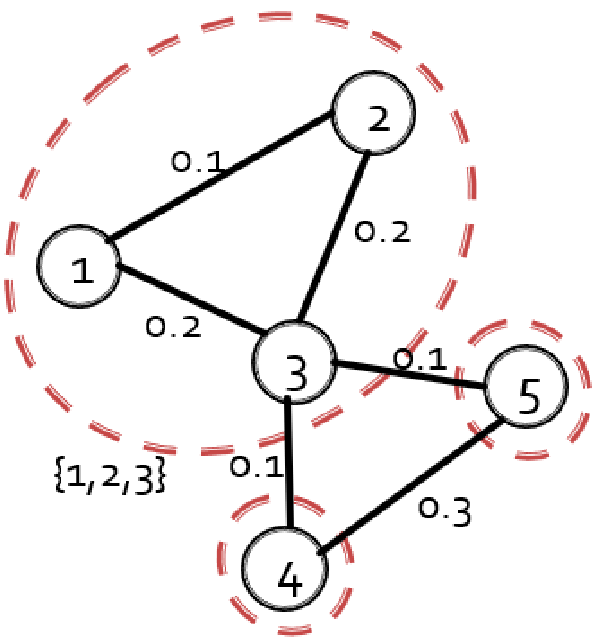
k_5= 0.1+0.3 =0.4

$$Q(\{5\}) = -\left(\frac{k_5}{2m}\right)^2 = -0.04$$

$$
\begin{aligned}
Q(\{4,5\}) &= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \\
&= \frac{2\times e_{45}}{2m} - \left(\frac{k_4+k_5}{2m}\right)^2 \\
&= 0.14
\end{aligned}
$$

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 4



Current communities

Summary:
Scores for the 2 possible movements

$$\Delta Q(4 \rightarrow \{1,2,3\}) = -0.14$$

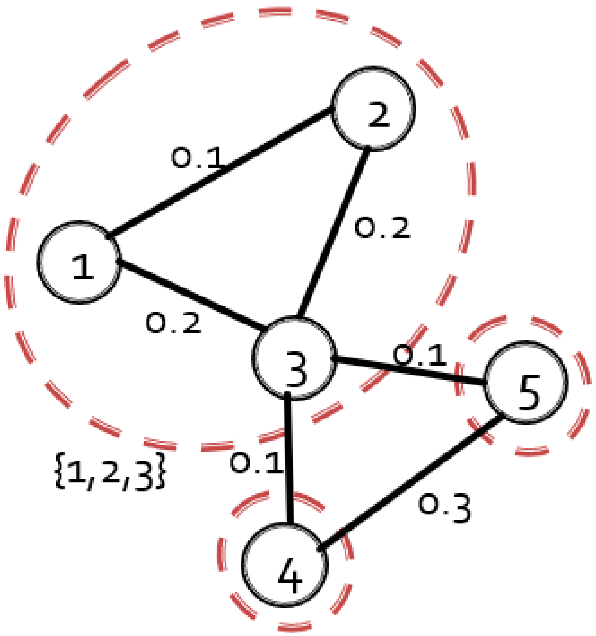
$$\Delta Q(4 \rightarrow \{5\}) = 0.22$$

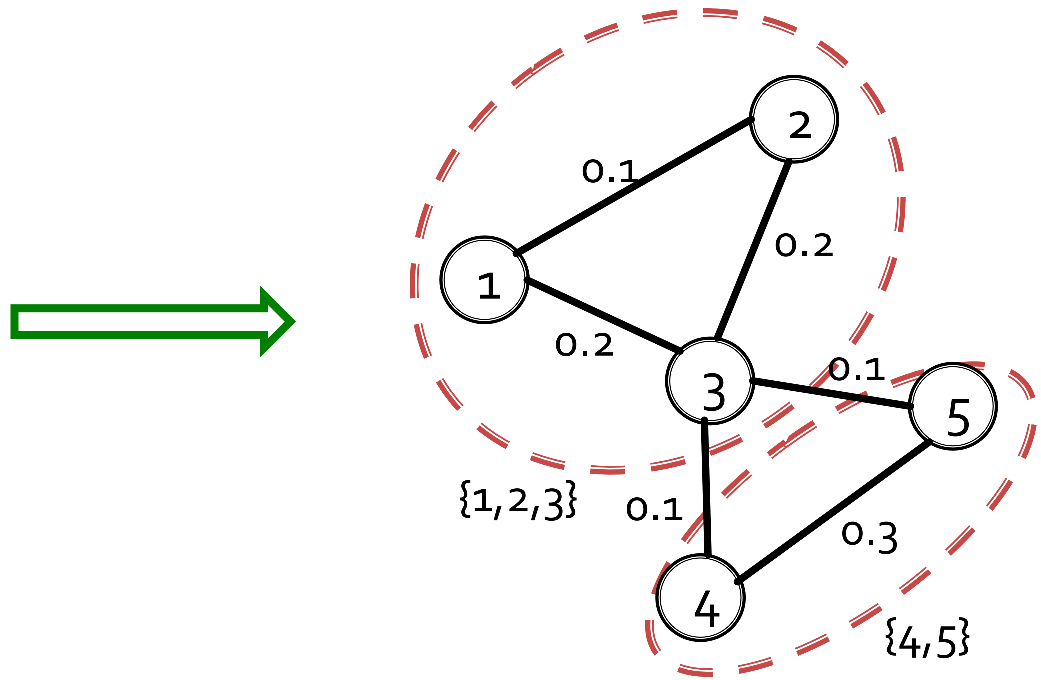
$\Delta Q(4 \rightarrow \{5\})$ has the largest positive gain.

We move node 4 to {5}.

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 4



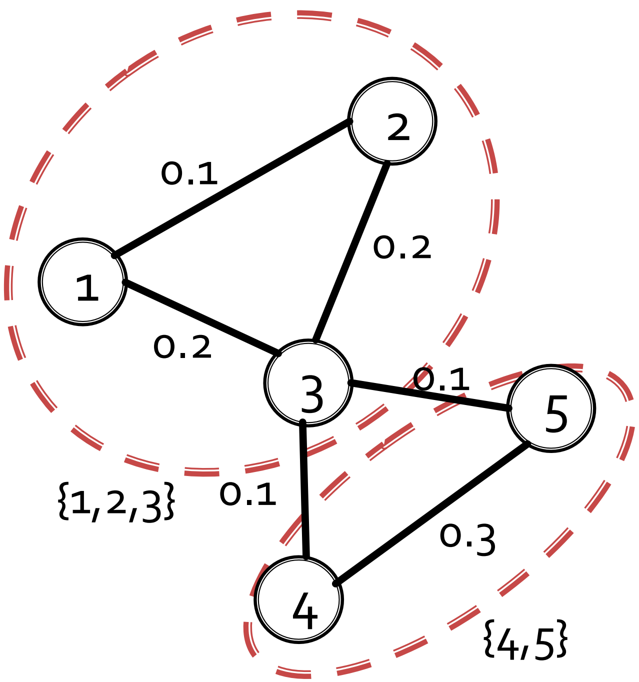


updated communities

After processing node 4, we move node 4 to {5}.

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 5



Current communities

The neighbouring communities of node 5:   Community {1, 2, 3}

There is only one neighbouring community for node 5, so there is only one possible movement

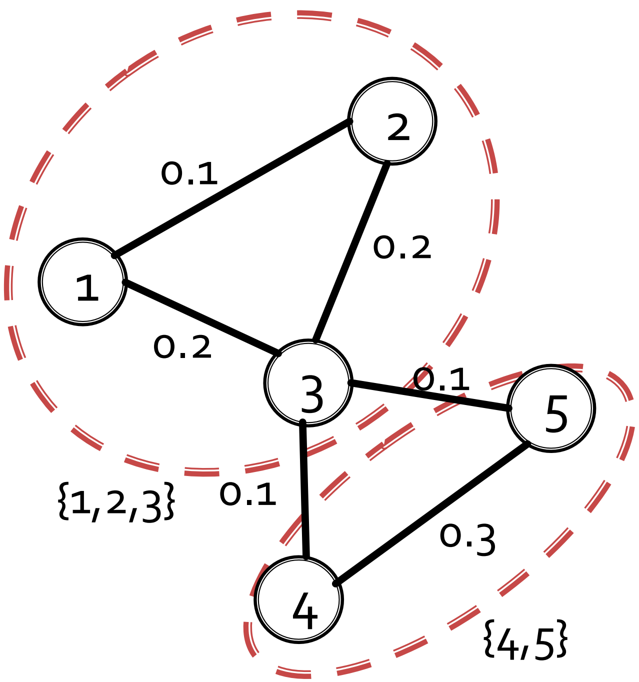
We need to calculate the following modularity gain:

$$\Delta Q(\{4,5\} \rightarrow 5 \rightarrow \{1,2,3\})$$

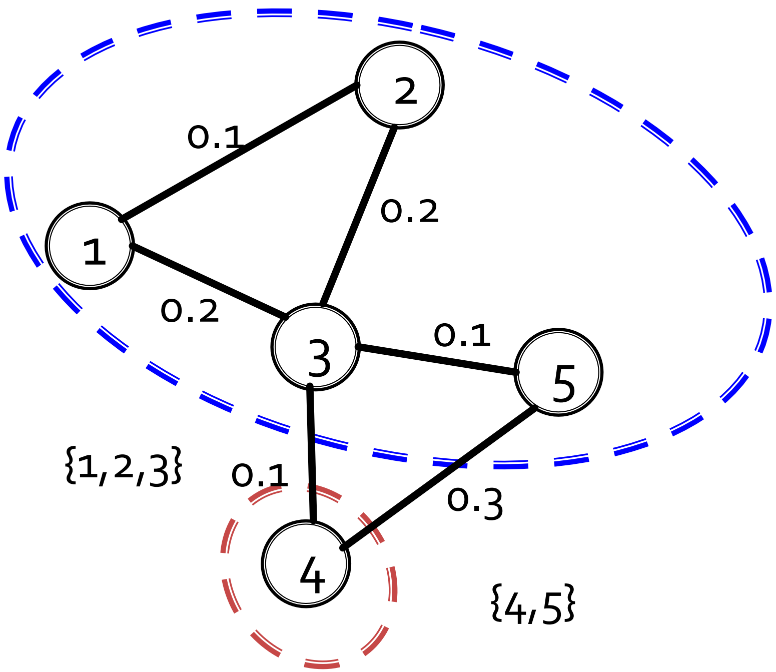
Sequentially process the node list {3, 1, 2, 4, 5}

processing node 5

$$\Delta Q(\{4,5\} \rightarrow 5 \rightarrow \{1,2,3\})$$



Q_{before}



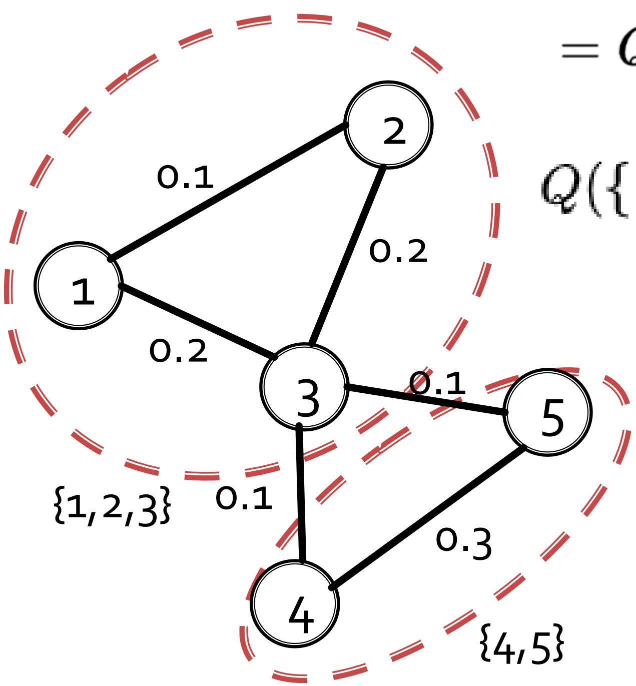
Q_{after}

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 5

$$\Delta Q(\{4,5\} \to 5 \to \{1,2,3\}) \;=\; Q_{after} - Q_{before}$$

$$= Q(\{4\}) + Q(\{1,2,3,5\}) - [Q(\{4,5\}) + Q(\{1,2,3\})]$$



Current communities

$$\begin{aligned} Q(\{1,2,3,5\}) &= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \\ &= \frac{2\times(e_{12}+e_{13}+e_{23}+e_{35})}{2m} - \left(\frac{k_1+k_2+k_3+k_5}{2m}\right)^2 \\ &= \frac{2\times(0.1+0.2+0.2+0.1)}{2} - \left(\frac{0.3+0.3+0.6+0.4}{2}\right)^2 \\ &= -0.04 \end{aligned}$$

The below values can be obtained from previous steps:

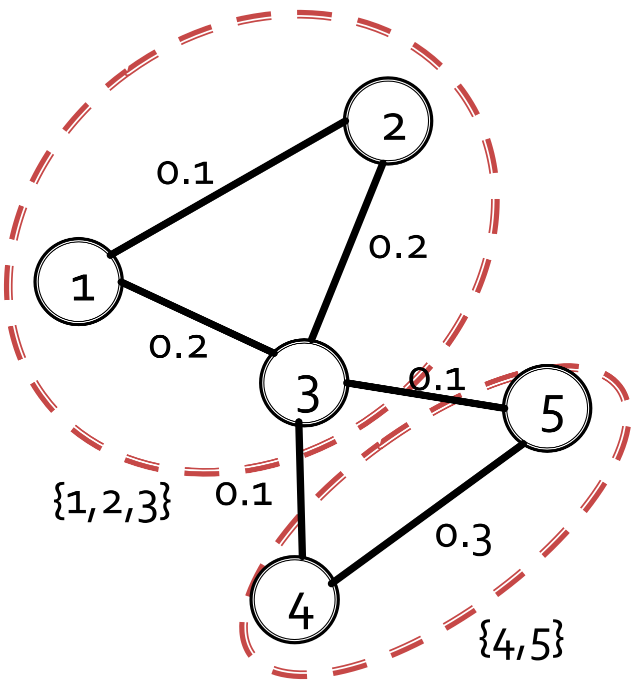
$$Q(\{4\}) \;=\; -\left(\frac{k_4}{2m}\right)^2 \;=\; -0.04$$

$$Q(\{4,5\}) = 0.14$$

$$Q(\{1,2,3\}) \;=\; 0.14$$

Sequentially process the node list {3, 1, 2, 4, 5}

processing node 5



Current communities

$$\Delta Q(\{4,5\} \to 5 \to \{1,2,3\})$$

$$= \quad Q_{after} - Q_{before}$$

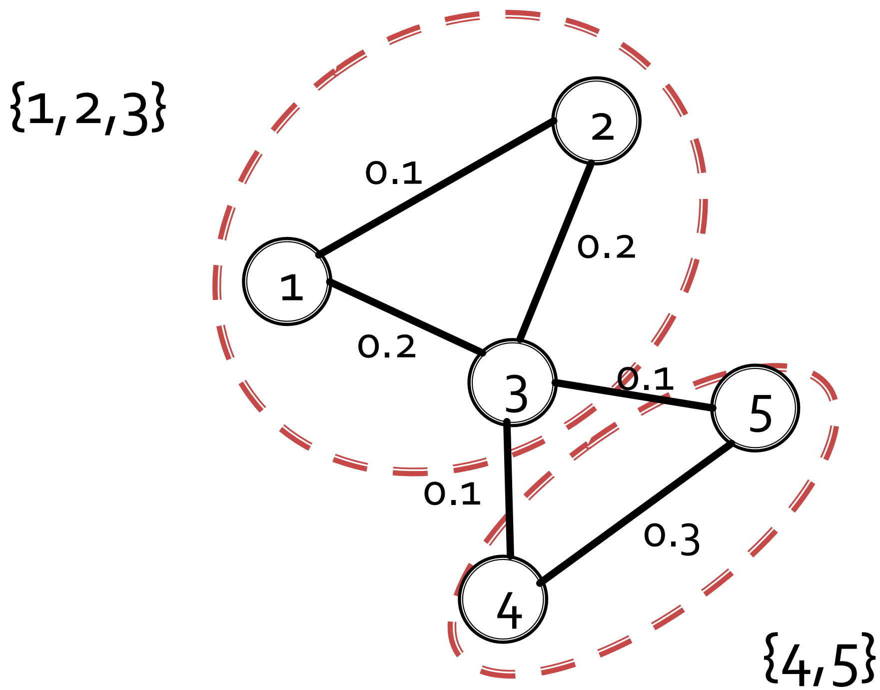
$$= Q(\{4\}) + Q(\{1,2,3,5\}) - [Q(\{4,5\}) + Q(\{1,2,3\})]$$

$$= -0.04 - 0.04 - (0.14 + 0.14) = -0.36$$

The largest gain is a negative value (there is one neighboring community). Do not move node 5.

Result

After processing the node list {3, 1, 2, 4, 5}, we have two communities: {1, 2, 3} and {4, 5}.



Additional discussion: (beyond the question)

We have finished the 1st scan after processing the node list.

To fully complete community generation, continue to do the 2nd scan, 3rd scan, ….

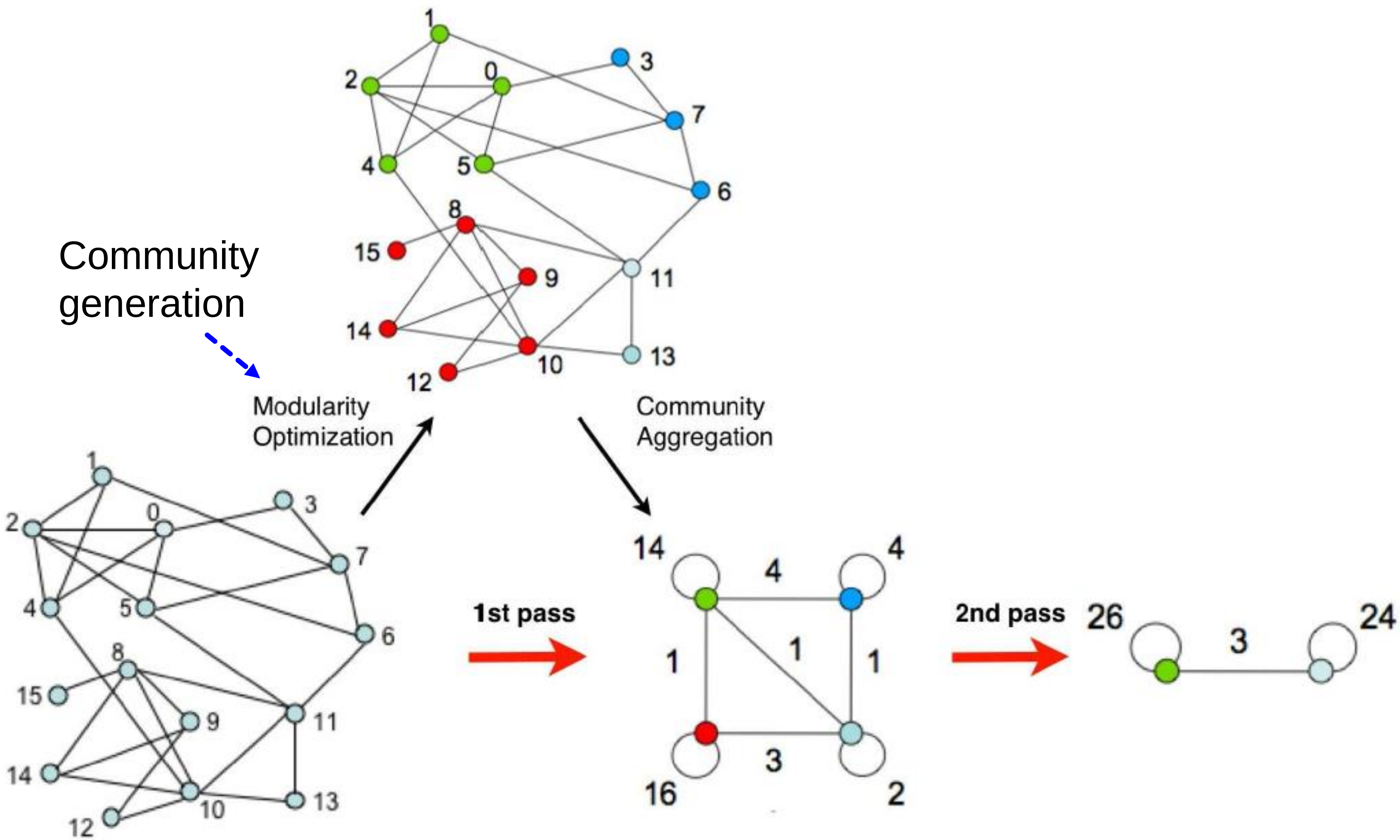
In each scan, we randomly generate a node list for processing.

Converge:

if there is no node movement in one scan, the community generation step converges.

Non-examinable!

- Multi-pass Louvain Algorithm
  - Multiple passes.
    - Produce hierarchical results
    - One pass corresponds to one hierarchical level
  - Each pass is made of 2 phases:
    - Phase 1: community generation (we have learnt)
    - Phase 2: community aggregation

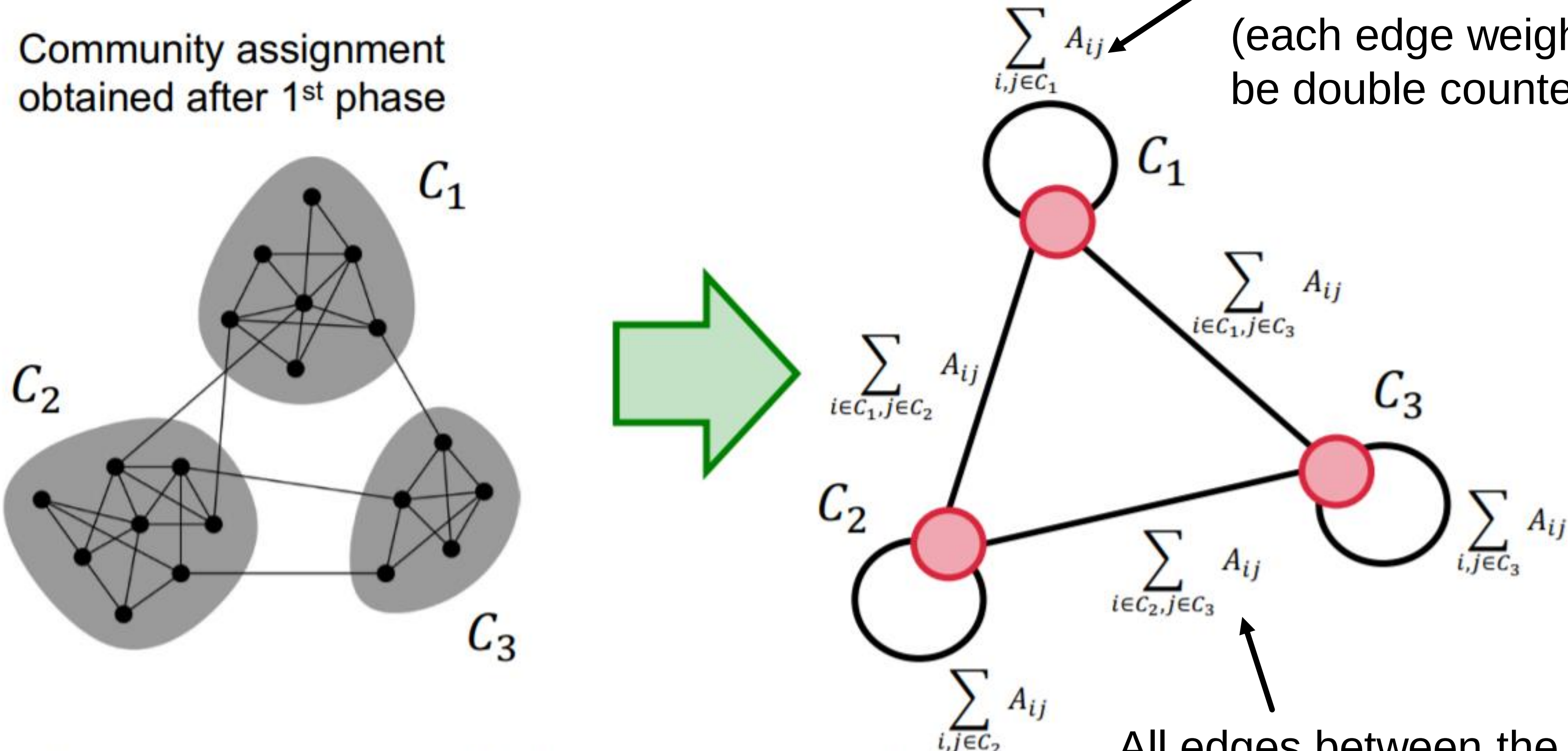


Multi-pass Louvain Algorithm high-level illustration

For un-weighted graph: the weight for each edge is 1

- Phase 2: community aggregation
  - Generate a new graph based on the communities
    - Create one node (super node) for one community in the new graph
    - Create edges of the super nodes if the original communities are connected by at least one edge
      - The weight of the edge between the two super nodes is the sum of the edge weights between the original communities
    - Create self-connections of the super nodes based on the internal connections of the original communities.

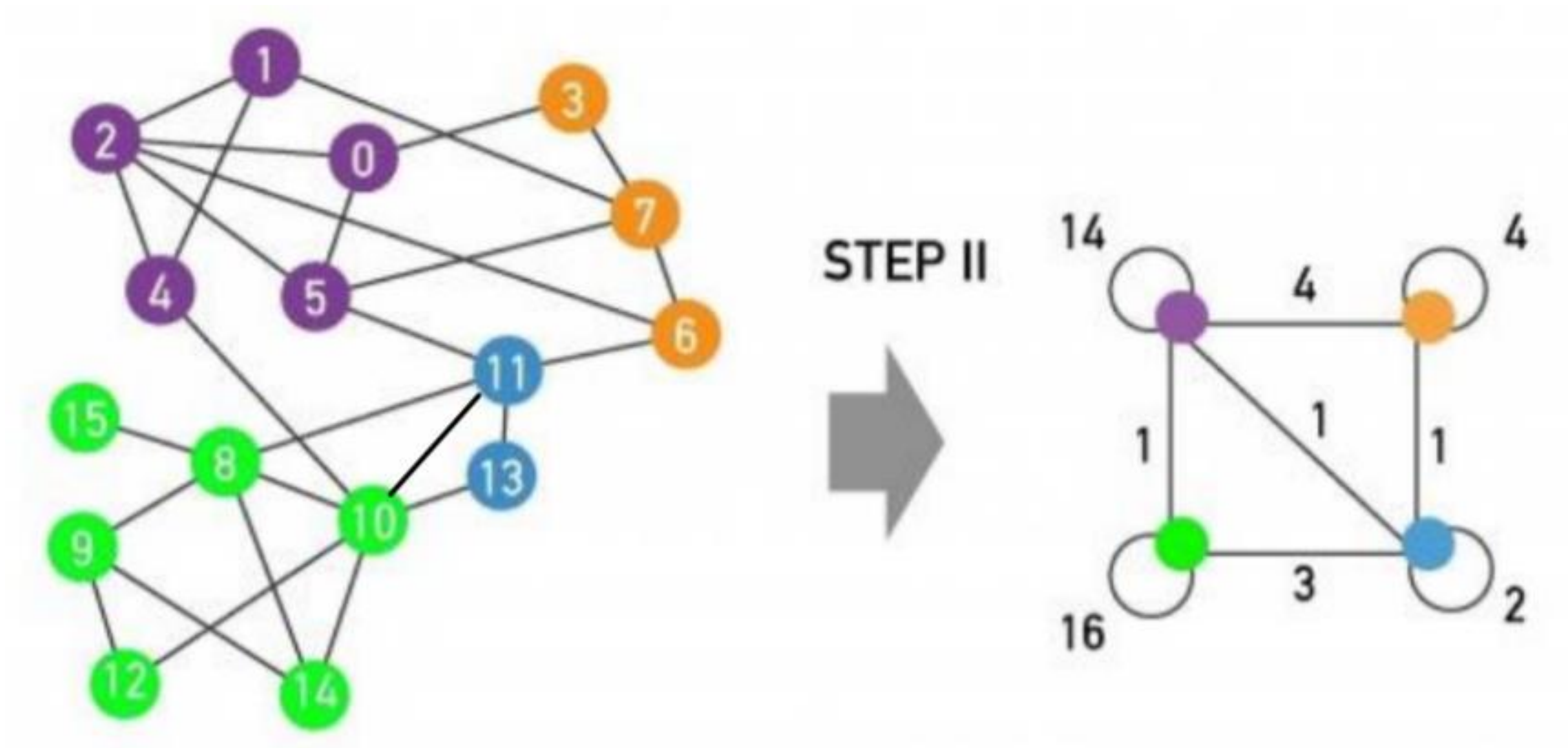
- Super nodes are constructed by merging nodes in the same community.



Internal edges in one cluster
(each edge weight will be double counted)

All edges between the two communities

Example

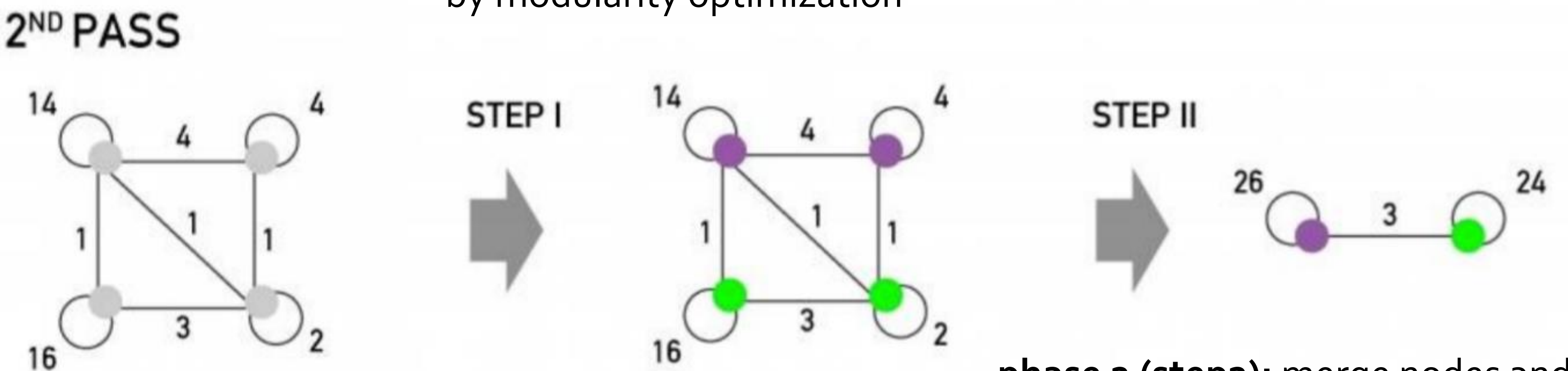


generate communities in phase 1 (step 1) generate new graph in phase 2 (step 2)

The 1st pass.

Example

phase 1 (step1): community generation by modularity optimization



phase 2 (step2): merge nodes and generate a new graph

In the 2nd pass:
run phase 1 (step 1 in the figure) and phase 2 (step 2) again on the new graph

- Further reading
  - BigCLAM
    - For overlapping community detection
  - Spectral clustering
    - A well-know clustering method on graph
    - http://snap.stanford.edu/class/cs224w-2019/
    - https://en.wikipedia.org/wiki/Spectral_clustering