Data Mining and Analysis

Graph Mining: 3

CSCE 676 :: Fall 2019
Texas A&M University
Department of Computer Science & Engineering
Prof. James Caverlee

Resources

Networks, Crowds, and Markets. Chapter 2 and Chapter 3

MMDS Chapter 10.1, 10.2, 10.4 (ignore 10.4.4)

Louvain method (Wikipedia)

DMTT Chapter 17.4

Method 2: Personalized PageRank

Summary

Pick a see node s of interest

Run PPR with teleport set = {s}

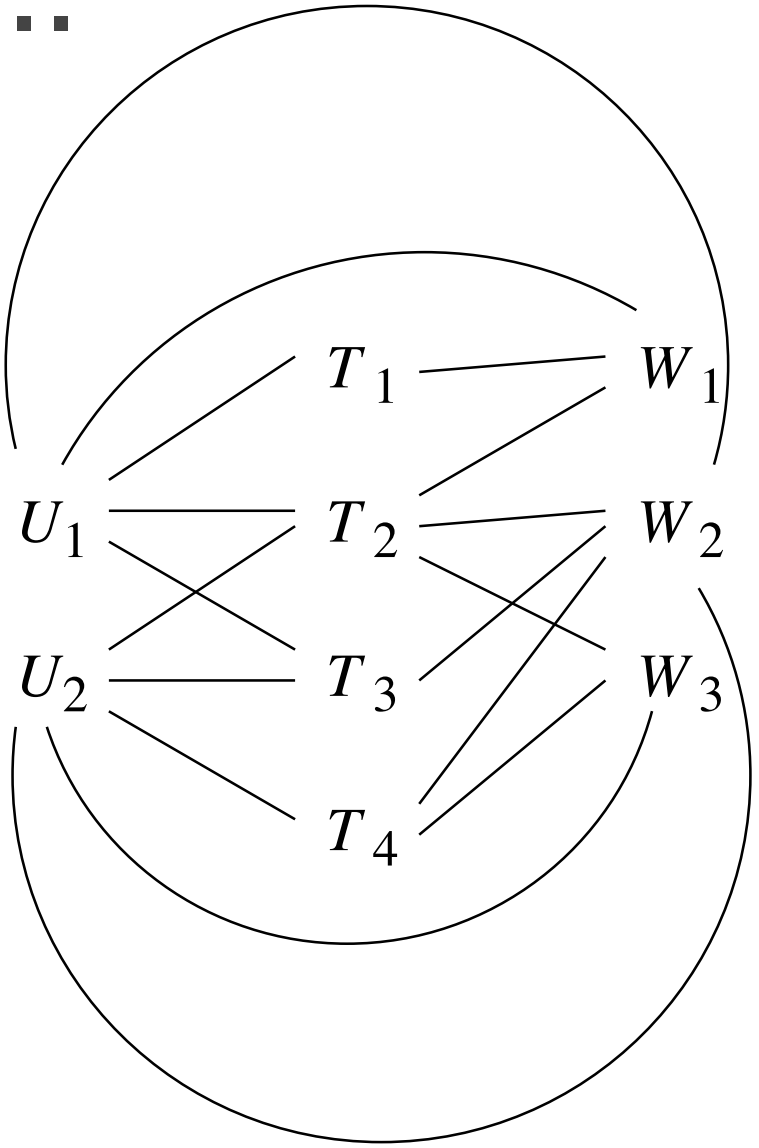
Sort the nodes by the decreasing PPR score

Sweep over the nodes and find good clusters

Intuition ...

Suppose a random walker starts at T1

Chance that random walker will end up at T2 or T4 ... which is the better chance?



Intuition ...

Random walk provides a measure of similarity between two nodes

Maybe we can rank all nodes with respect to a seed node using random walks

Find breaks in the ranks to identify clusters

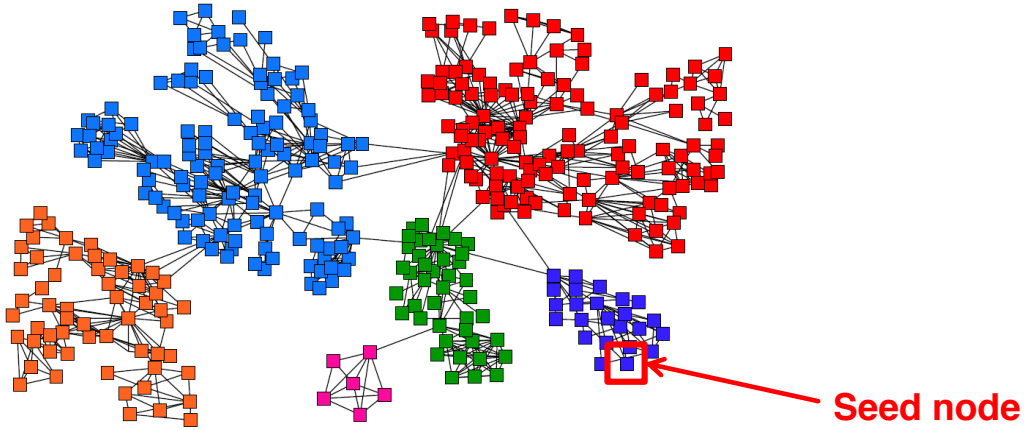
Idea: Seed nodes

Discovering clusters based on seed nodes

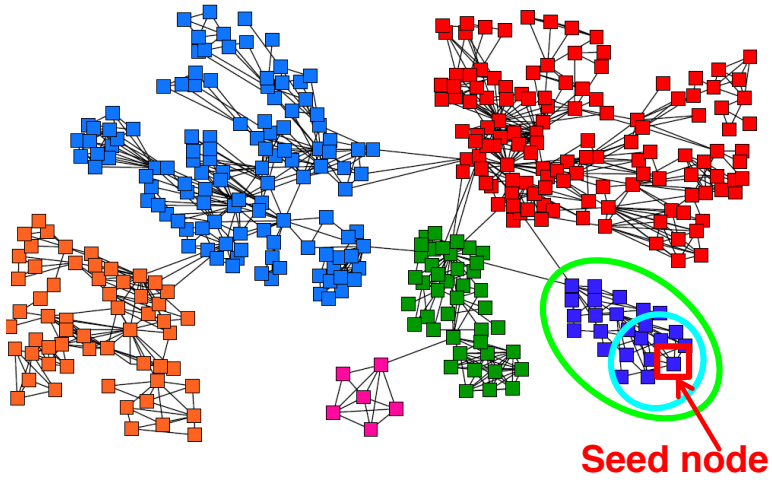
Given: Seed node S

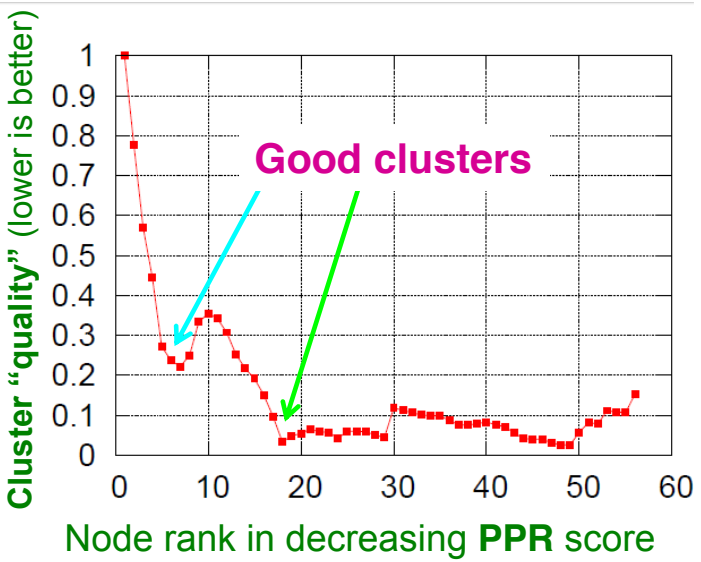
Compute (approximate) Personalized PageRank (PPR) around node S (teleport set={S})

Idea is that if S belongs to a nice cluster, the random walk will get trapped inside the cluster



Seed Node: Intuition

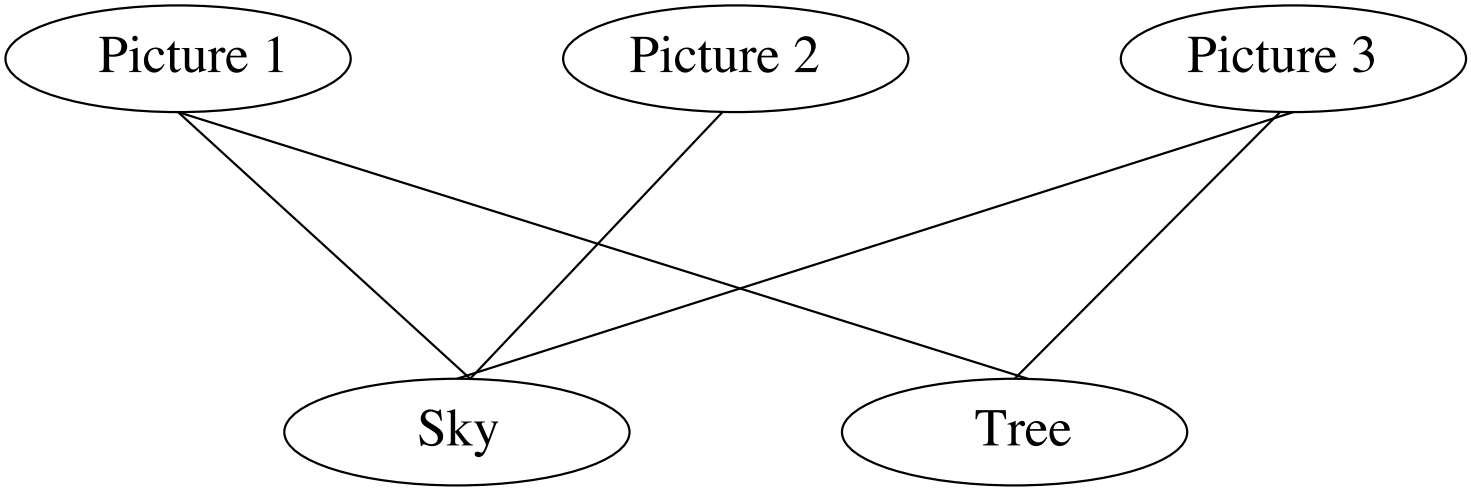




Algorithm outline:

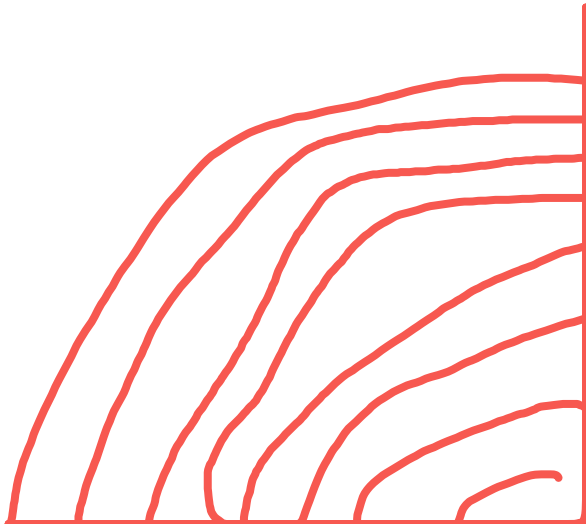
- Pick a seed node S of interest
- Run PPR with teleport set = {S}
- Sort the nodes by the decreasing PPR score
- Sweep over the nodes and find good clusters

Issue: Random Walks with Restart (or Personalized PageRank or PPR)



P1, P2, P3, Sky, Tree

$$\begin{array}{c} \\ P1 \\ P2 \\ P3 \\ S \\ T \end{array} \begin{bmatrix} 0 & 0 & 0 & 1/3 & 1/2 \\ 0 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1/3 & 1/2 \\ 1/2 & 1 & 1/2 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \end{bmatrix}$$



算法解釋:
https://www.youtube.com/watch?v=P8Kt6Abq_rM

$$\mathbf{v}' = \begin{bmatrix} 0 & 0 & 0 & 4/15 & 2/5 \\ 0 & 0 & 0 & 4/15 & 0 \\ 0 & 0 & 0 & 4/15 & 2/5 \\ 2/5 & 4/5 & 2/5 & 0 & 0 \\ 2/5 & 0 & 2/5 & 0 & 0 \end{bmatrix} \mathbf{v} + \begin{bmatrix} 1/5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{v}' = \begin{bmatrix} 1/5 & 1/5 & 1/5 & 7/15 & 3/5 \\ 0 & 0 & 0 & 4/15 & 0 \\ 0 & 0 & 0 & 4/15 & 2/5 \\ 2/5 & 4/5 & 2/5 & 0 & 0 \\ 2/5 & 0 & 2/5 & 0 & 0 \end{bmatrix} \mathbf{v}$$

參照課本

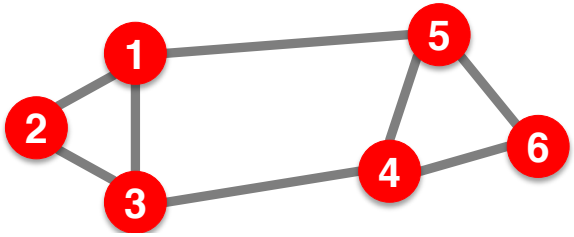
Page 379, MMDS Book

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1/5 \\ 0 \\ 0 \\ 2/5 \\ 2/5 \end{bmatrix} \begin{bmatrix} 35/75 \\ 8/75 \\ 20/75 \\ 6/75 \\ 6/75 \end{bmatrix} \begin{bmatrix} 95/375 \\ 8/375 \\ 20/375 \\ 142/375 \\ 110/375 \end{bmatrix} \begin{bmatrix} 2353/5625 \\ 568/5625 \\ 1228/5625 \\ 786/5625 \\ 690/5625 \end{bmatrix} \cdots \begin{bmatrix} .345 \\ .066 \\ .145 \\ .249 \\ .196 \end{bmatrix}$$

What Makes a Good Cluster?

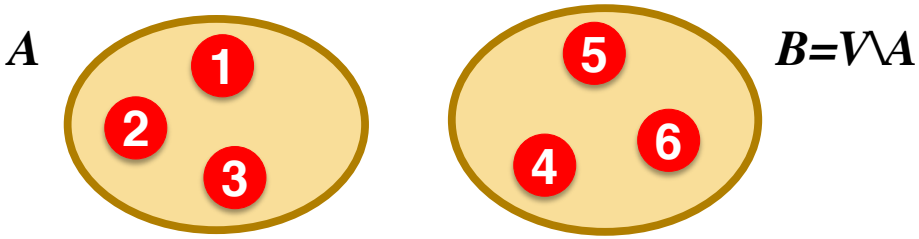
What makes a good cluster?

Undirected graph G(V,E):



Partitioning task:

Divide vertices into 2 disjoint groups A,B = V \ A



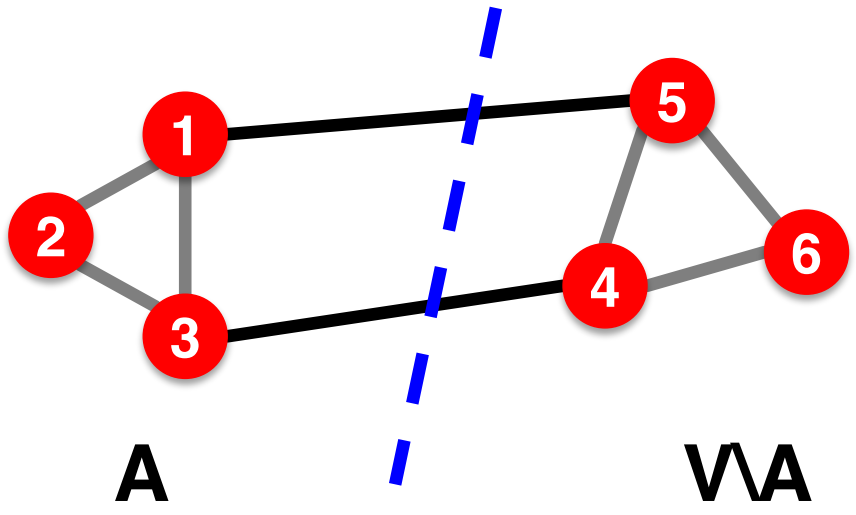
Question:

How can we define a “good” cluster in G?

What makes a good cluster?

Maximize the number of within-cluster connections

Minimize the number of between-cluster connections



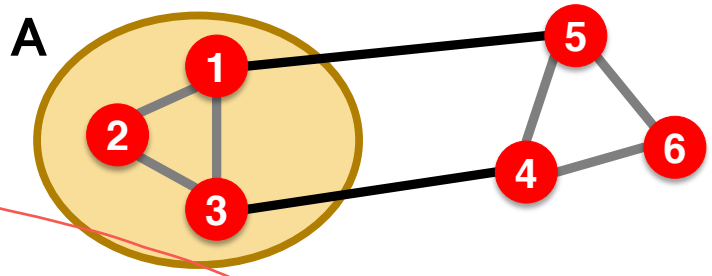
Graph Cuts

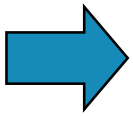
Express cluster quality as a function of the "edge cut" of the cluster

Cut: Set of edges with only one node in the cluster

$$cut(A) = \sum_{i \in A, j \notin A} w_{ij}$$

Note: This works for weighed and unweighted (set all $w_{ij}=1$) graphs





$cut(A) = 2$

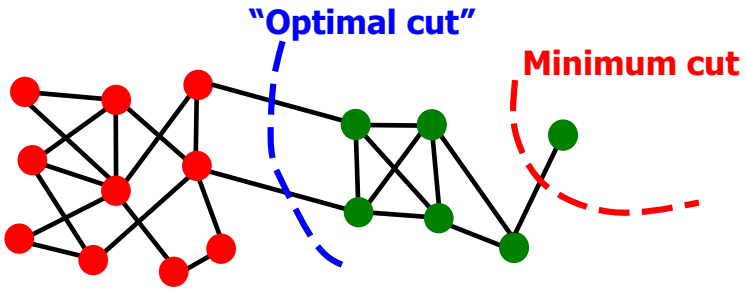
Cut Score

Partition quality: Cut score

Quality of a cluster is the weight of connections pointing outside the cluster

Degenerate case:

Problem:



Only considers external cluster connections

Does not consider internal cluster connectivity

图聚类划分conductance score
https://blog.csdn.net/pipisorry/article/details/49052255

Graph Partitioning Criteria

Criterion: Conductance

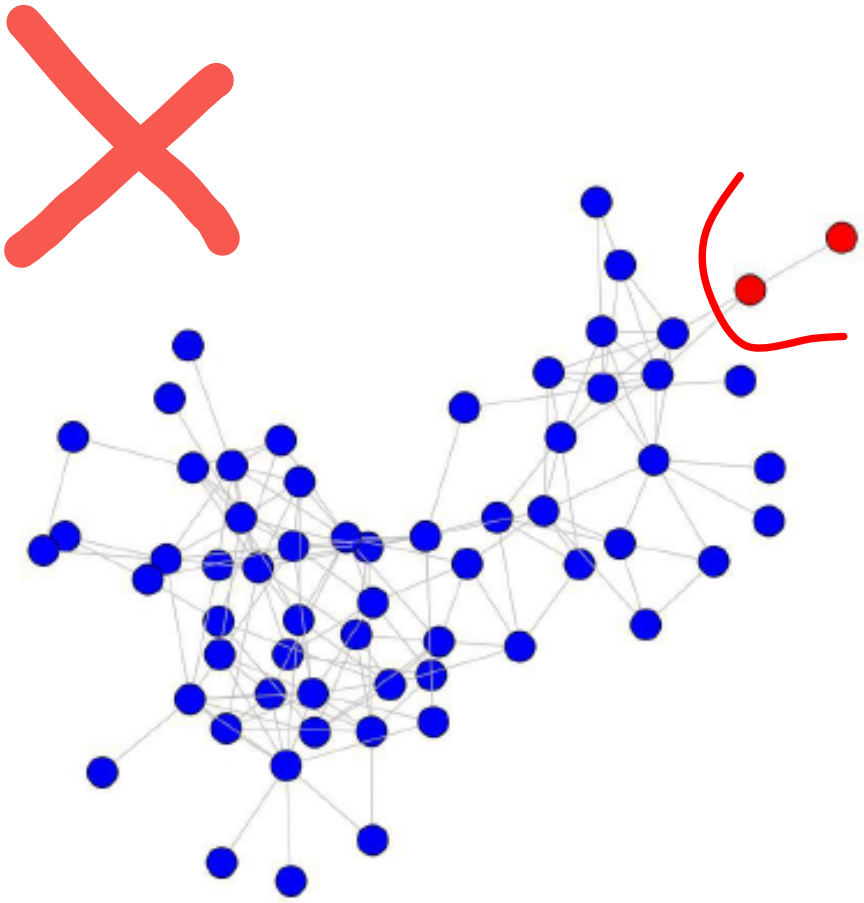
Connectivity of the group to the rest of the network relative to the density of the group

$$\phi(A) = \frac{|\{(i,j) \in E; i \in A, j \notin A\}|}{\min(vol(A), 2m - vol(A))}$$

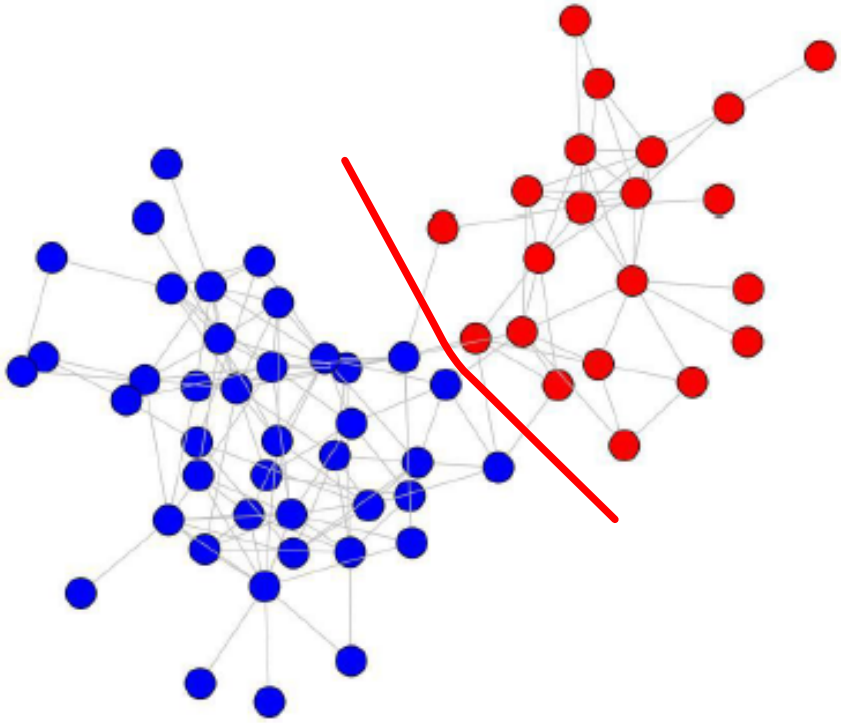
$\boldsymbol{vol(A)}$: total weight of the edges with at least one endpoint in $\boldsymbol{A}$: $\mathbf{vol}(\boldsymbol{A}) = \sum_{i \in A} \boldsymbol{d_i}$

m… number of edges of the graph
$\mathbf{d_i}$… degree of node ***i***

Example: Conductance Score

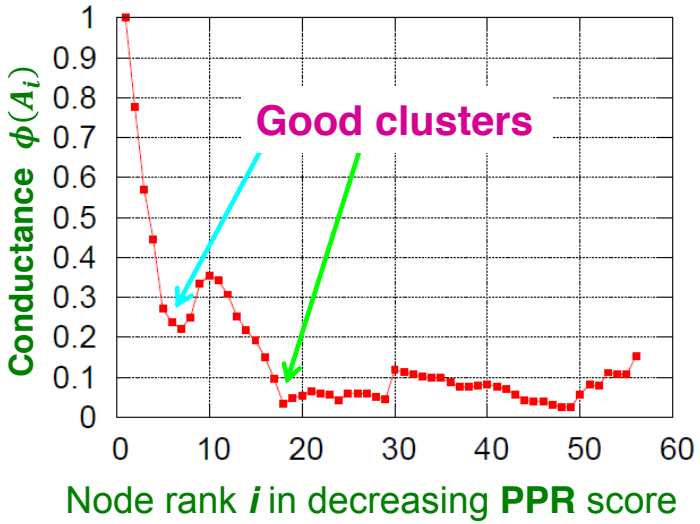


$\phi = 2/4 = 0.5$



$\phi = 6/92 = 0.065$

Algorithm Outline: Sweep



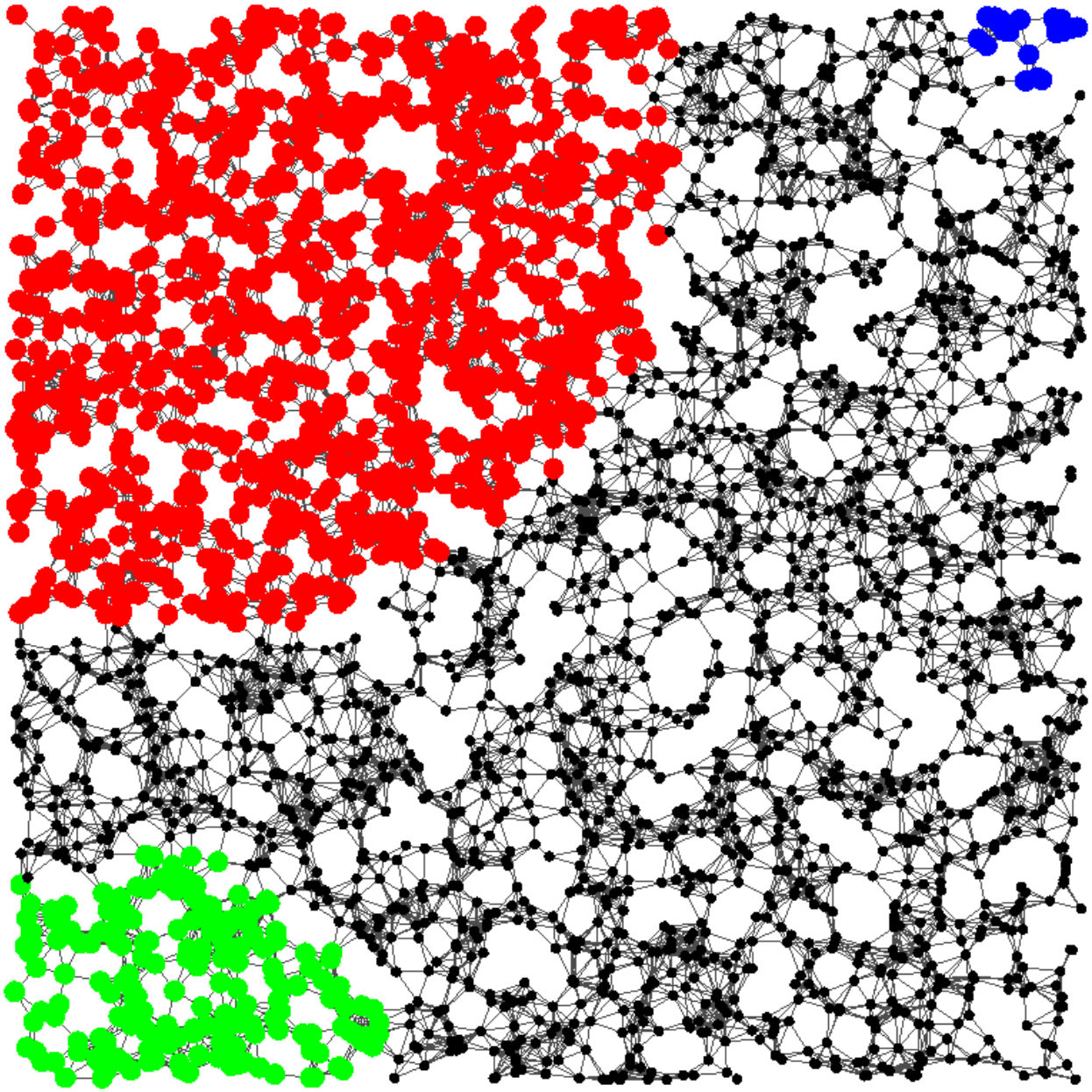
Algorithm outline:

- Pick a seed node S of interest
- Run PPR w/teleport={S}
- Sort the nodes by the decreasing PPR score
- Sweep over the nodes and find good clusters

Sort nodes in decreasing PPR score $r_1 > r_2 > \cdots > r_n$

For each $\boldsymbol{i}$ compute $\boldsymbol{\phi(A_i = \{r_1, \ldots r_i\})}$

Local minima of $\boldsymbol{\phi(A_i)}$ correspond to good clusters



Method 3: Louvain Method

Louvain Method

Greedy algorithm for community detection O(n log n) run time

Supports weighted graphs
Provides hierarchical partitions

Widely utilized to study large networks because:

- Fast
- Rapid convergence properties
- High modularity output (i.e., “better communities”)

Modularity

A measure of how well a network is partitioned into communities

= the fraction of the edges that fall within the given groups minus the expected fraction if edges were distributed at random

[-1,+1]

Modularity for Two Communities

$$Q = \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \frac{s_v s_w + 1}{2}$$

Actual number of edges -
Expected number of edges

A_{vw} = 1 (edge) or 0 (no edge)

Node v has degree k_v

Node w has degree k_w

m = number of links in the network

S_v = +1 (belongs to community 1) or -1 (belongs to community 2)

https://en.wikipedia.org/wiki/Louvain_modularity

Louvain Method

Louvain algorithm greedily maximizes modularity

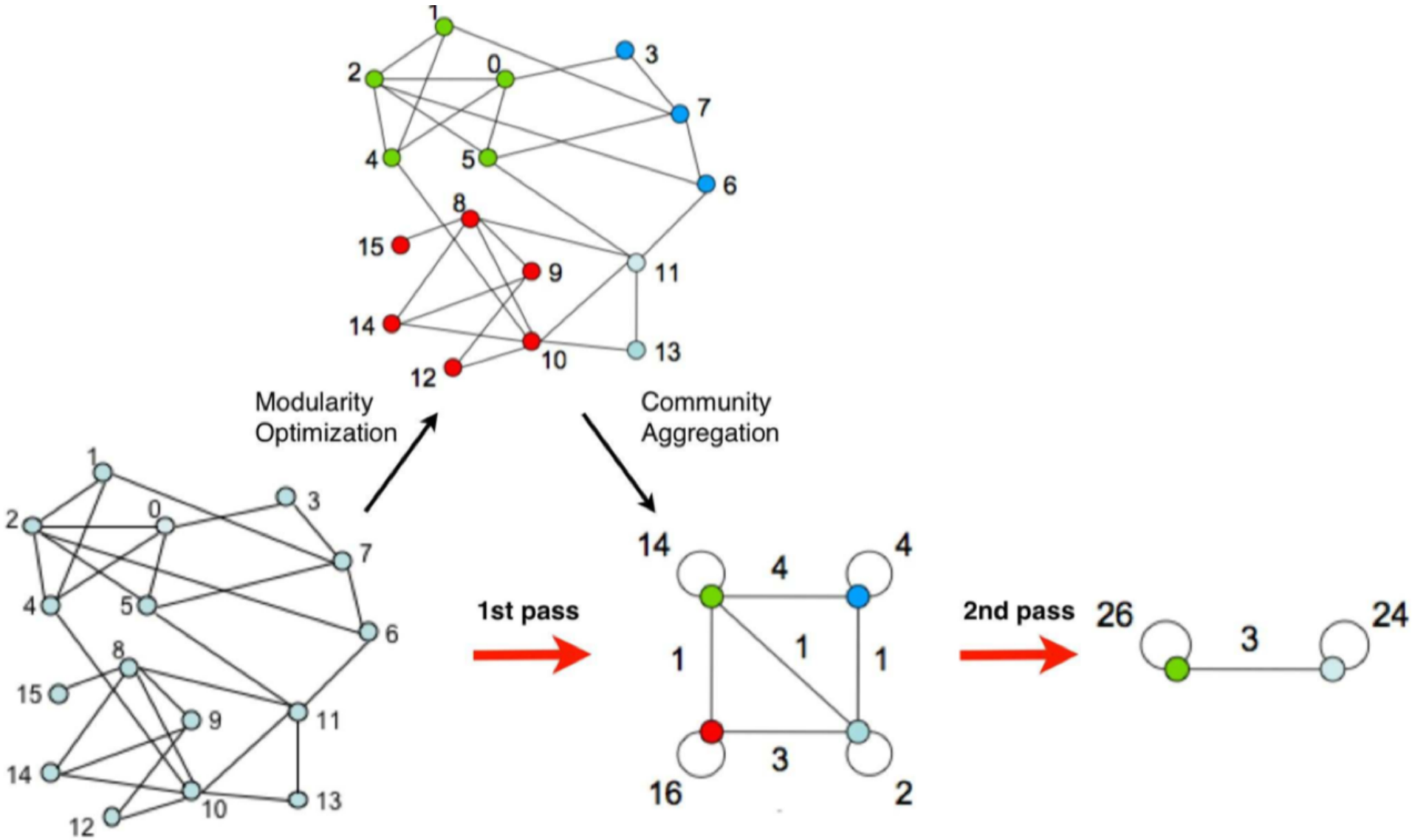
Each pass is made of 2 phases:

Phase 1: Modularity is optimized by allowing only local changes of communities

Phase 2: The identified communities are aggregated in order to build a new network of communities

Goto Phase 1

Until no increase of modularity is possible


Modularity
Optimization
Community
Aggregation
1st pass
2nd pass
26
24
3
14
4
16
2
1
3

Phase 1: Partitioning

Put each node in a graph into a distinct community (one node per community)

For each node i, the algorithm performs two calculations:

1. Compute the modularity gain (ΔQ) when putting node i from its current community into the community of some neighbor j of i
2. Move i to a community that yields the largest modularity gain ΔQ

The loop runs until no movement yields a gain

Modularity Gain

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

$$\Delta Q(i \to C) = \left[\frac{\sum_{in}+k_{i,in}}{2m} - \left(\frac{\sum_{tot}+k_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

where:

Σ_{in}... sum of link weights between nodes in C

Σ_{tot}..... sum of all link weights of nodes in C

$k_{i,in}$… sum of link weights between node i and nodes in C

k_i… sum of all link weights (i.e., degree) of node i

m… number of links in the network

⇒ Also need to derive ΔQ(D→i) of taking node i out of community D

And then: ΔQ = ΔQ(i→C) + ΔQ(D→i)

Phase 2: Restructuring

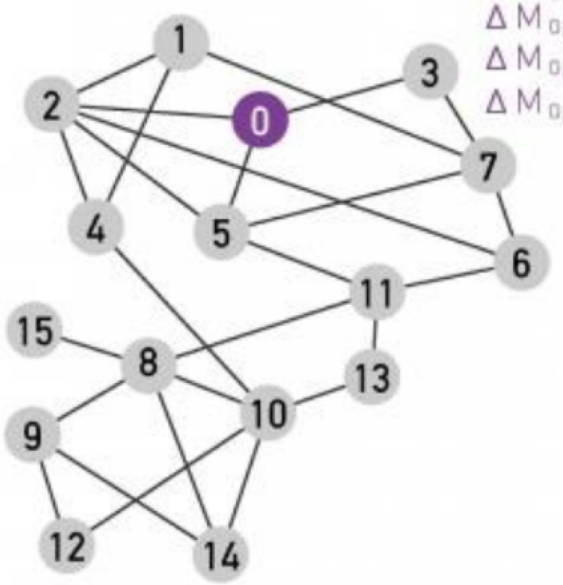
The partitions obtained in the first phase are contracted into super-nodes and the weighted network is created as follows

1. Super-nodes are connected if there is at least one edge between nodes of the corresponding communities
2. The weight of the edge between the two super-nodes is the sum of the weights from all edges between their corresponding partitions

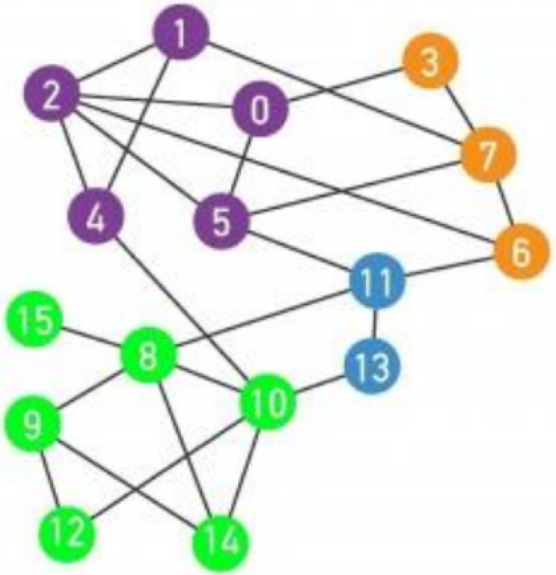
The loop runs until the community configuration does not change anymore

1ST PASS

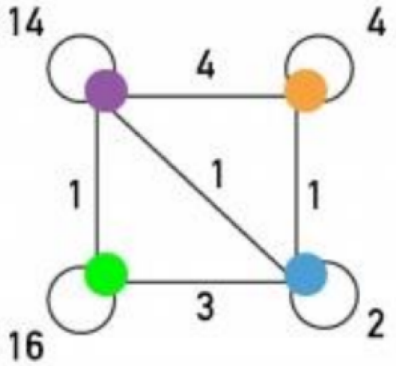
$\Delta M_{0,2} = 0.023$
$\Delta M_{0,3} = 0.032$
$\Delta M_{0,4} = 0.026$
$\Delta M_{0,5} = 0.026$



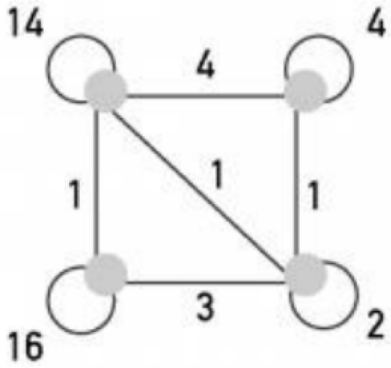
STEP I



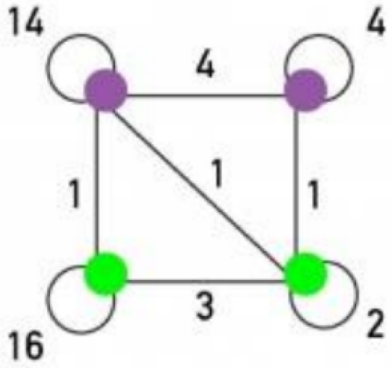
STEP II



2ND PASS



STEP I



STEP II