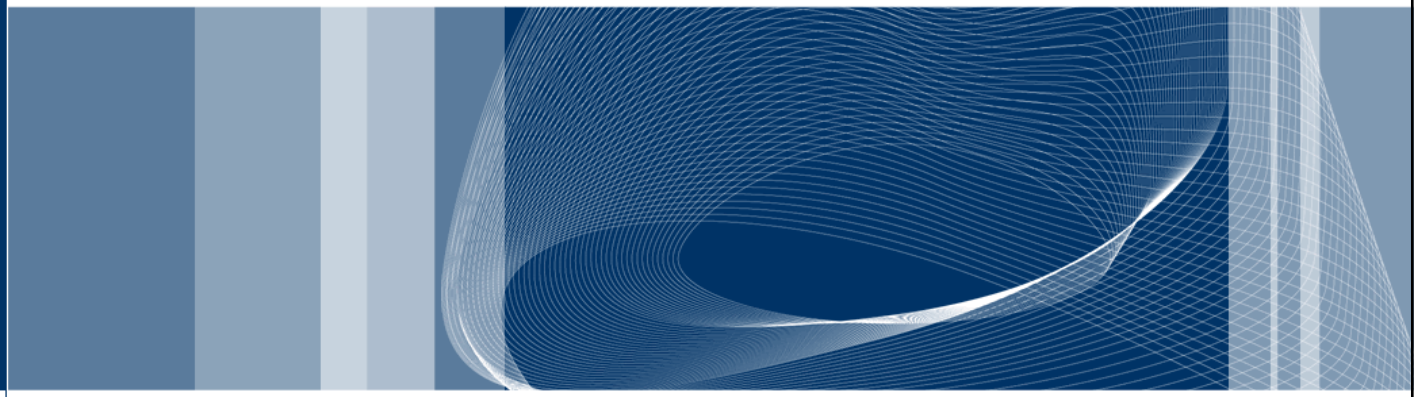
POLITECNICO DI MILANO



Graph Mining

Data Science for Mobility

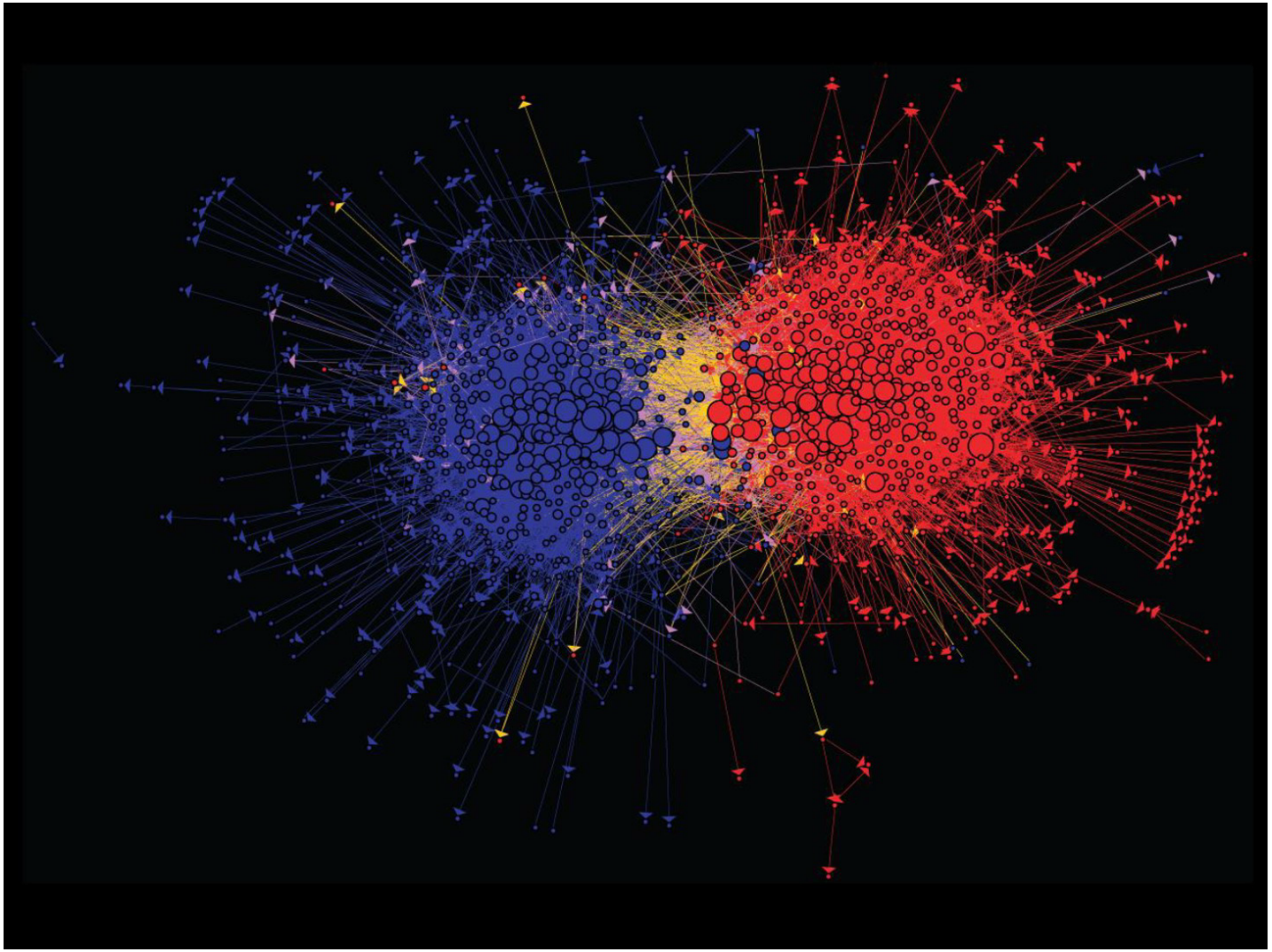
Prof. Pier Luca Lanzi

References

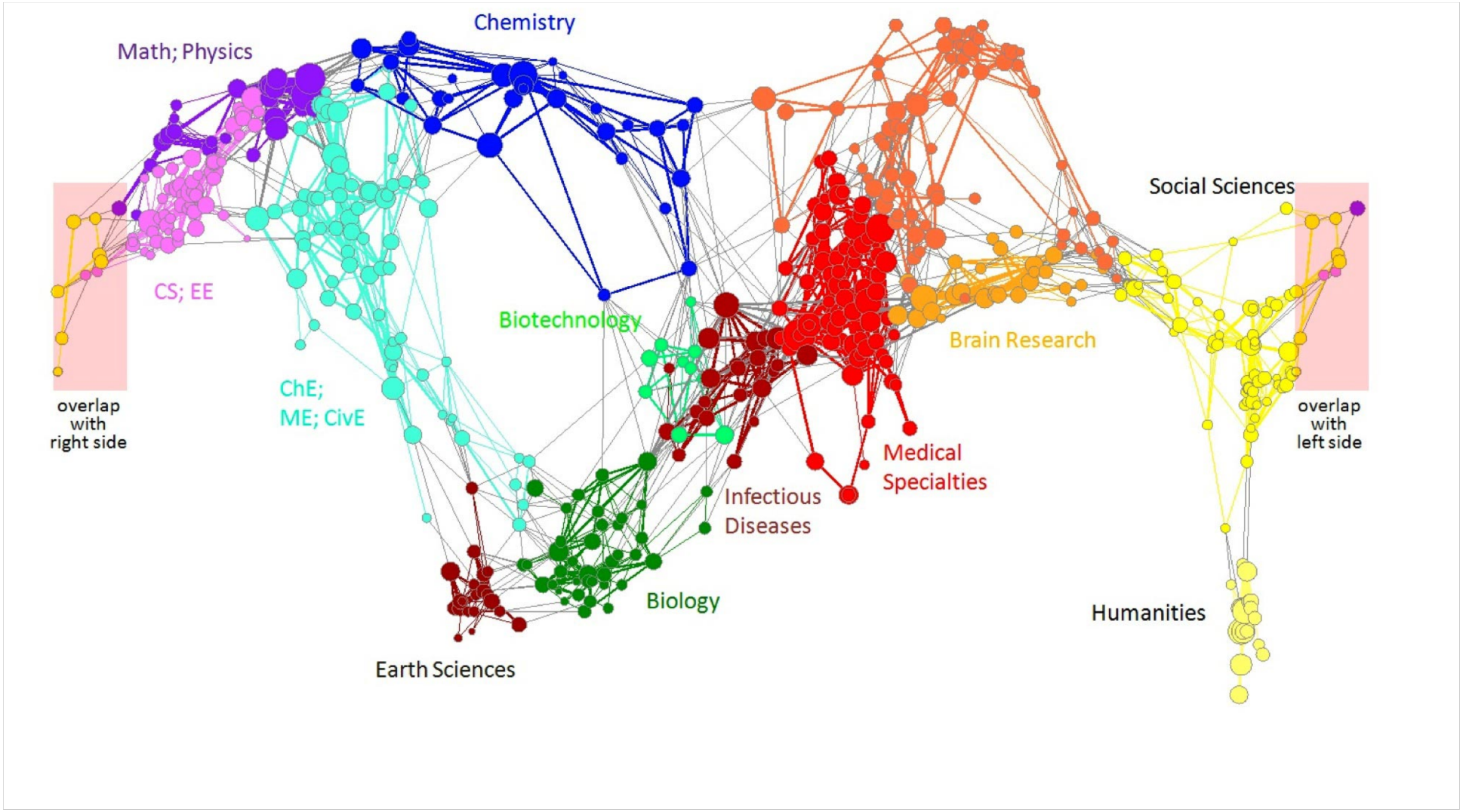
- Jure Leskovec, Anand Rajaraman, Jeff Ullman. Mining of Massive Datasets, Chapter 5 & Chapter 10

- Book and slides are available from http://www.mmds.org

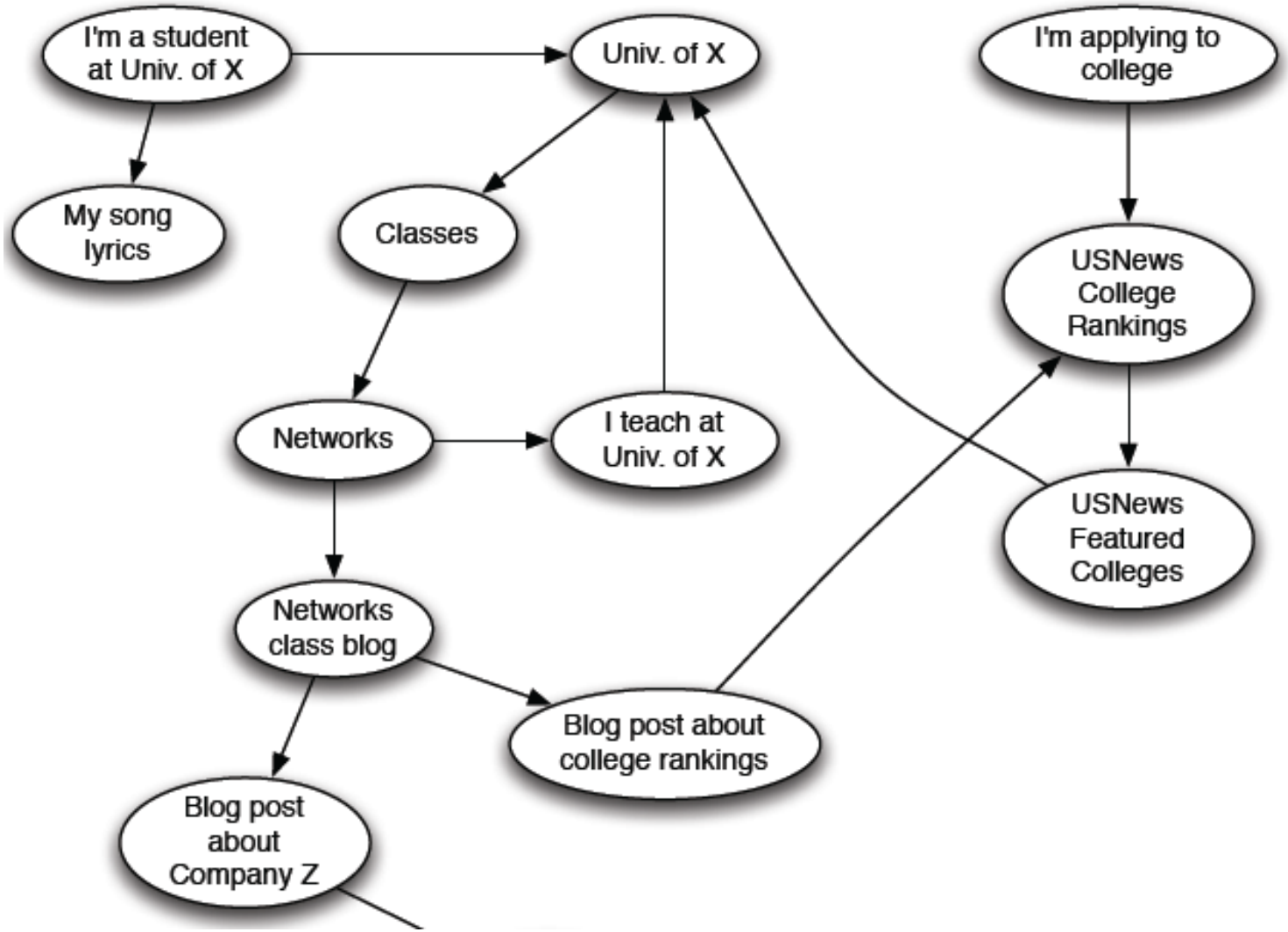
Facebook social graph
4-degrees of separation [Backstrom-Boldi-Rosa-Ugander-Vigna, 2011]



Connections between political blogs
Polarization of the network [Adamic-Glance, 2005]



Citation networks and Maps of science
[Börner et al., 2012]



Web as a graph: pages are nodes, edges are links

How is the Web Organized?

- Initial approaches
  - Human curated Web directories
  - Yahoo, DMOZ, LookSmart

- Then, Web search
  - Information Retrieval investigates: Find relevant docs in a small and trusted set
  - Newspaper articles, Patents, etc.

Web is huge, full of untrusted documents, random things, web spam, etc.

Web Search Challenges

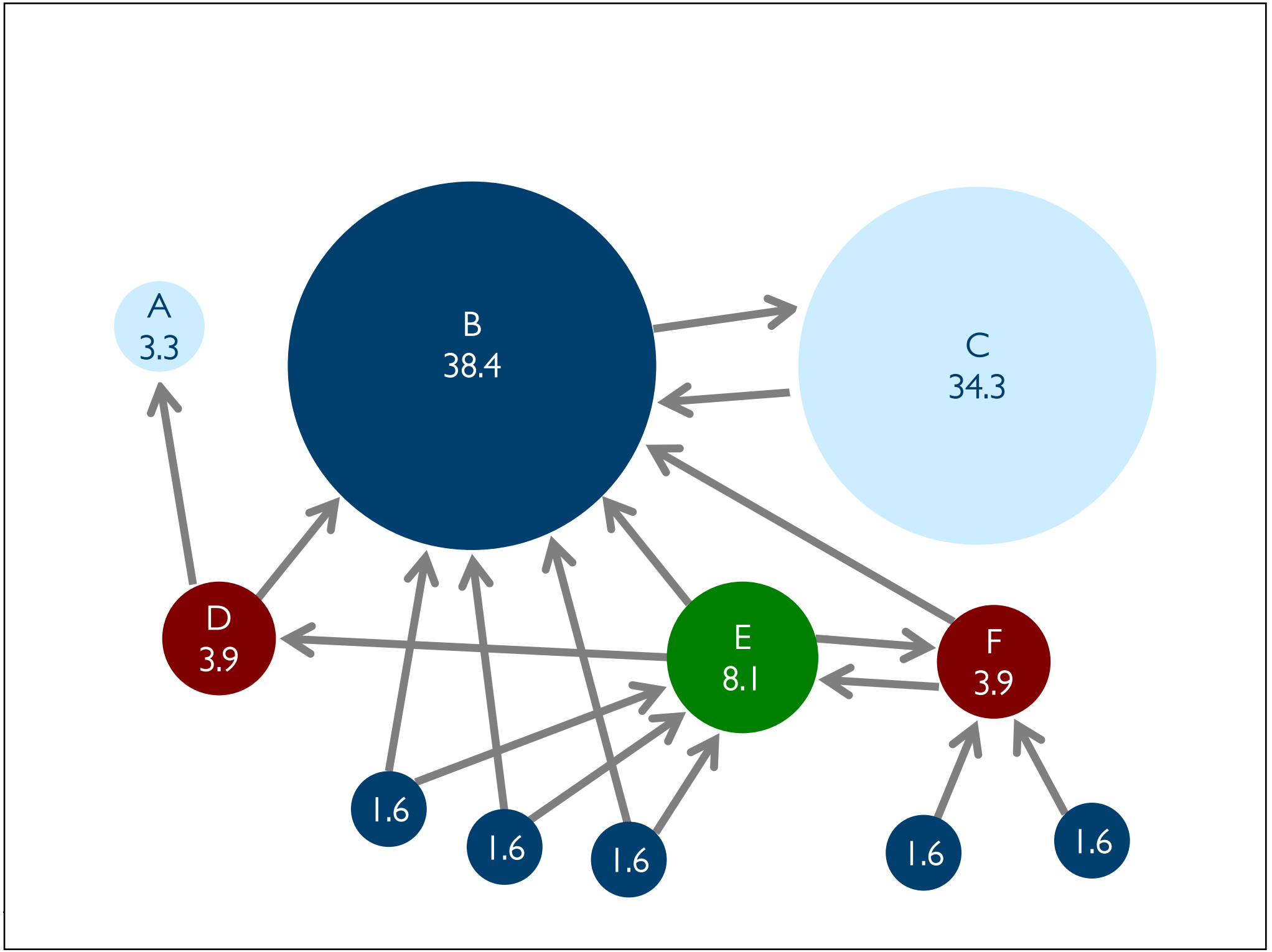
- Web contains many sources of information
  - Who should we "trust"?
  - Trick: Trustworthy pages may point to each other!
- What is the "best" answer to query "newspaper"?
  - No single right answer
  - Trick: Pages that actually know about newspapers might all be pointing to many newspapers

Page Rank

https://www.youtube.com/watch?v=wPMZr9RDmVk

Page Rank Algorithm

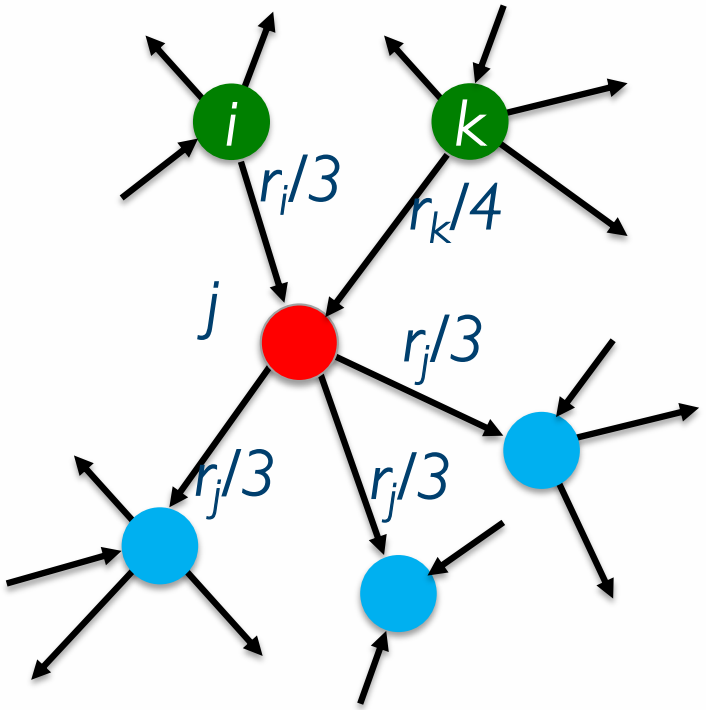
- The underlying idea is to look at links as votes
- A page is more important if it has more links
  - In-coming links? Out-going links?
- Intuition
  - www.stanford.edu has 23,400 in-links
  - www.joe-schmoe.com has one in-link
- Are all in-links are equal?
  - Links from important pages count more
  - Recursive question!


A
3.3
B
38.4
C
34.3
D
3.9
E
8.1
F
3.9
1.6
1.6
1.6
1.6
1.6

Simple Recursive Formulation

- Each link's vote is proportional to the importance of its source page
- If page j with importance r_j has n out-links, each link gets r_j/n votes
- Page j's own importance is the sum of the votes on its in-links

$$r_j = r_i/3 + r_k/4$$

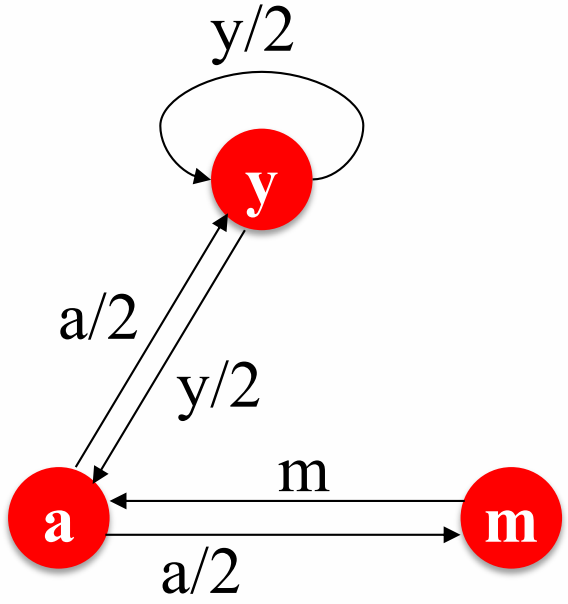


The "Flow" Model

- A "vote" from an important page is worth more
- A page is important if it is pointed to by other important pages
- Define a "rank" r_j for page j

$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$

where d_i is the out-degree of node i



- "Flow" equations

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

Solving the Flow Equations

- The equations, three unknowns variables, no constant
  - No unique solution
  - All solutions equivalent modulo the scale factor
- An additional constraint ($r_y+r_a+r_m=1$) forces uniqueness
- Gaussian elimination method works for small examples, but we need a better method for large web-size graphs

We need a different formulation that scales up!

The Matrix Formulation

- Represent the graph as a transition matrix M
  - Suppose page i has d_i out-links
  - If page i is linked to page j M_{ji} is set to 1/di else $M_{ji}=0$
  - M is a "column stochastic matrix" since the columns sum up to 1

- Given the rank vector r with an entry per page, where r_i is the importance of page i and the r_i sum up to one

$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d_i}}$$

The flow equation can be written as r = Mr

The Eigenvector Formulation

- Since the flow equation can be written as $r = Mr$, the rank vector r is also an eigenvector of M
- Thus, we can solve for r using a simple iterative scheme ("power iteration")
- **Power iteration:** a simple iterative scheme
  - Suppose there are *N* web pages
  - Initialize: $\mathbf{r}^{(0)} = [1/N,\ldots,1/N]^T$
  - Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \bullet \mathbf{r}^{(t)}$
  - Stop when $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}|_1 < \varepsilon$

The Random Walk Formulation

- Suppose that a random surfer that at time t is on page i and will continue it navigation by following one of the out-link at random
- At time t+1, will end up on page j and from there it will continue the random surfing indefinitely
- Let p(t) the vector of probabilities $p_i(t)$ that the surfer is on page i at time t (p(t) is the probability distribution over pages)
- Then, p(t+1) = Mp(t) so that

p(t) is the stationary distribution for the random walk

Existence and Uniqueness

For graphs that satisfy certain conditions, the stationary distribution is unique and eventually will be reached no matter what the initial probability distribution is

Hubs and Authorities (HITS)

Hubs and Authorities

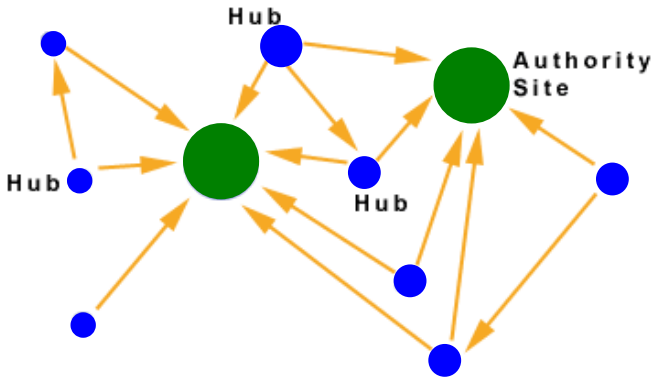
- HITS (Hypertext-Induced Topic Selection)
  - Is a measure of importance of pages and documents, similar to PageRank
  - Proposed at around same time as PageRank (1998)
- Goal: Say we want to find good newspapers
  - Don't just find newspapers. Find "experts", that is, people who link in a coordinated way to good newspapers
- The idea is similar, links are viewed as votes
  - Page is more important if it has more links
  - In-coming links? Out-going links?

Hubs and Authorities

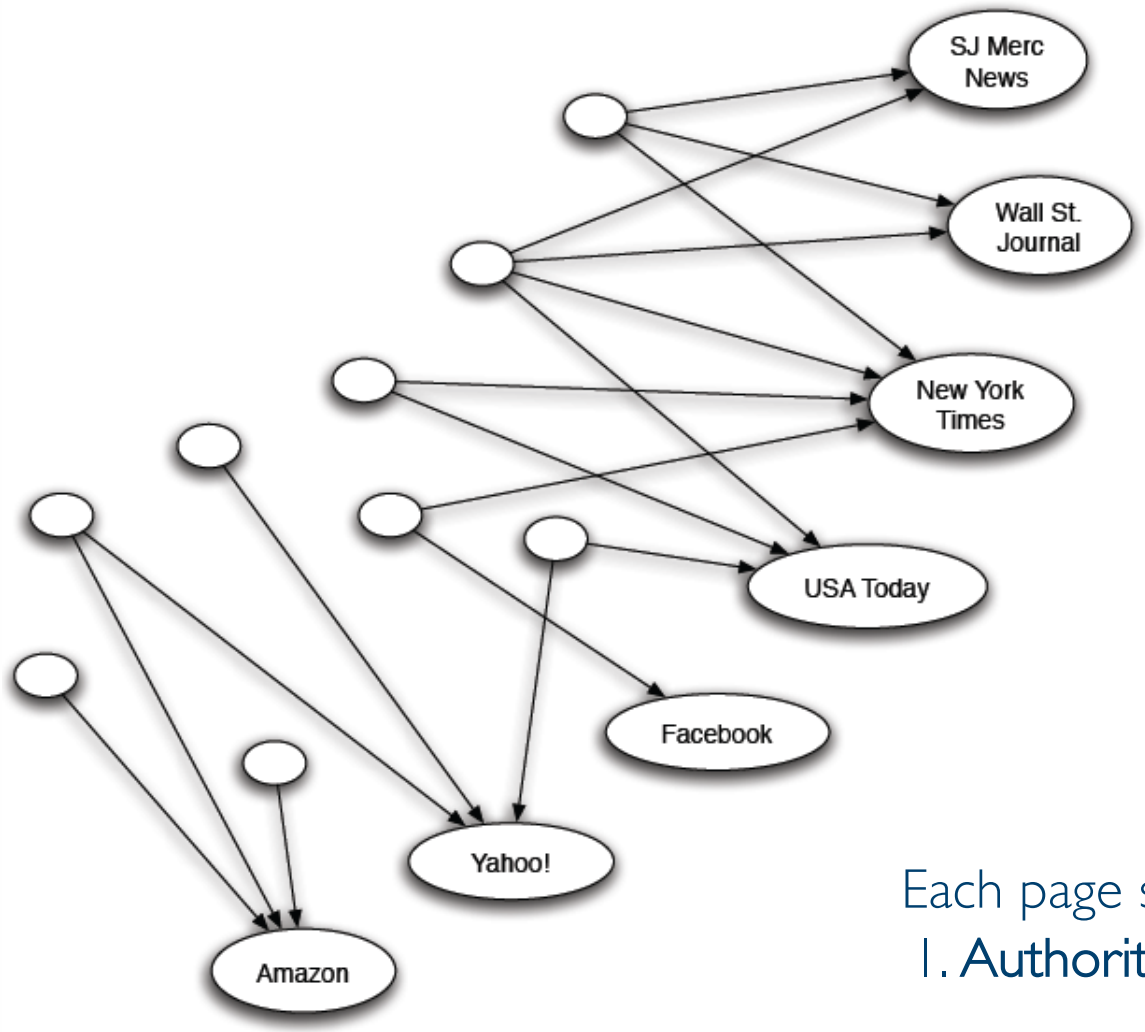
- Each page has 2 scores
- Quality as an expert (hub)
  - Total sum of votes of authorities pointed to
- Quality as a content (authority)
  - Total sum of votes coming from experts
- Principle of repeated improvement

Hubs and Authorities

- Authorities are pages containing useful information
  - Newspaper home pages
  - Course home pages
  - Home pages of auto manufacturers
- Hubs are pages that link to authorities
  - List of newspapers
  - Course bulletin
  - List of US auto manufacturers



Counting in-links: Authority

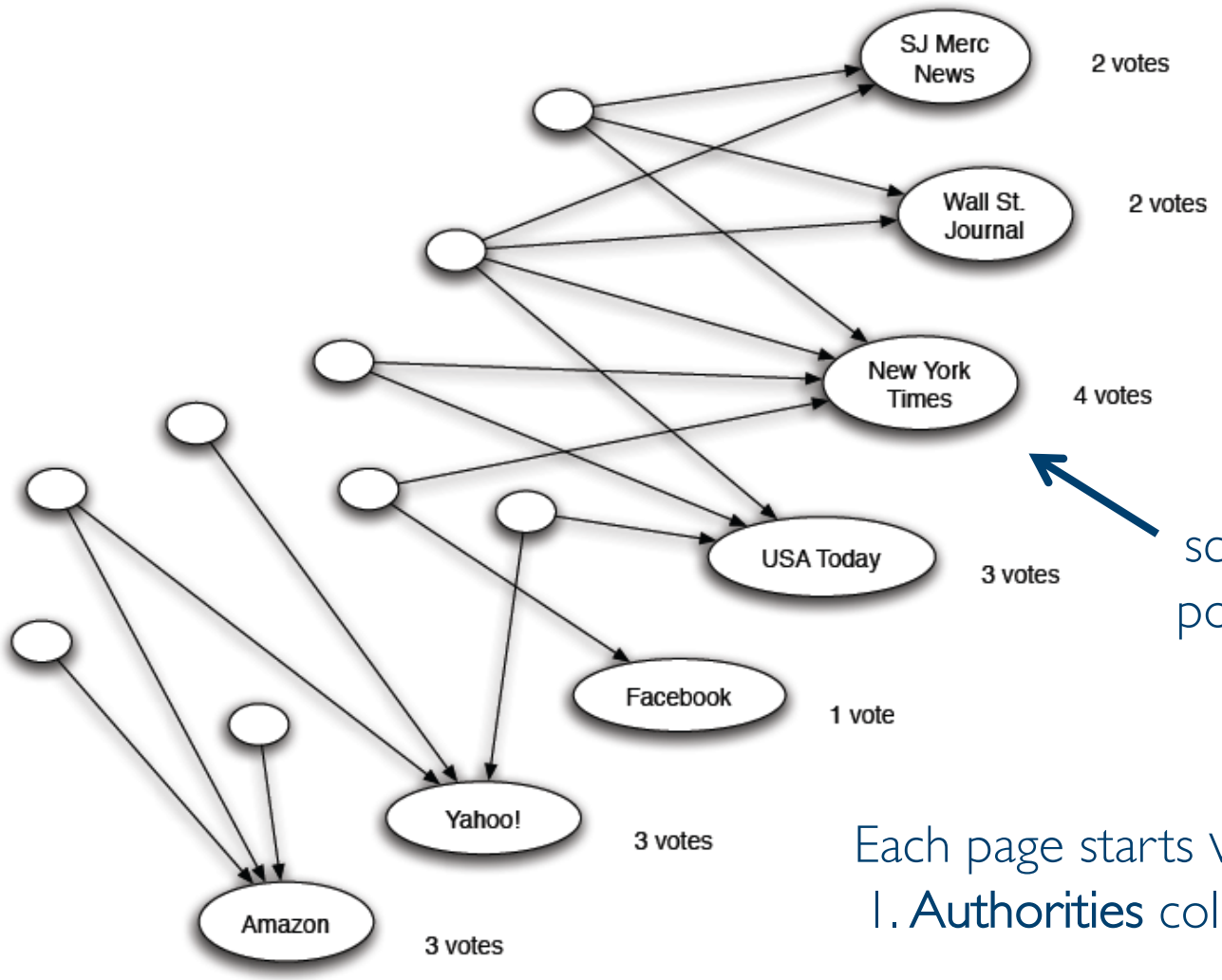


Each page starts with **hub** score

1. **Authorities** collect their votes

(Note this is idealized example. In reality graph is not bipartite and each page has both the hub and authority score)

Counting in-links: Authority

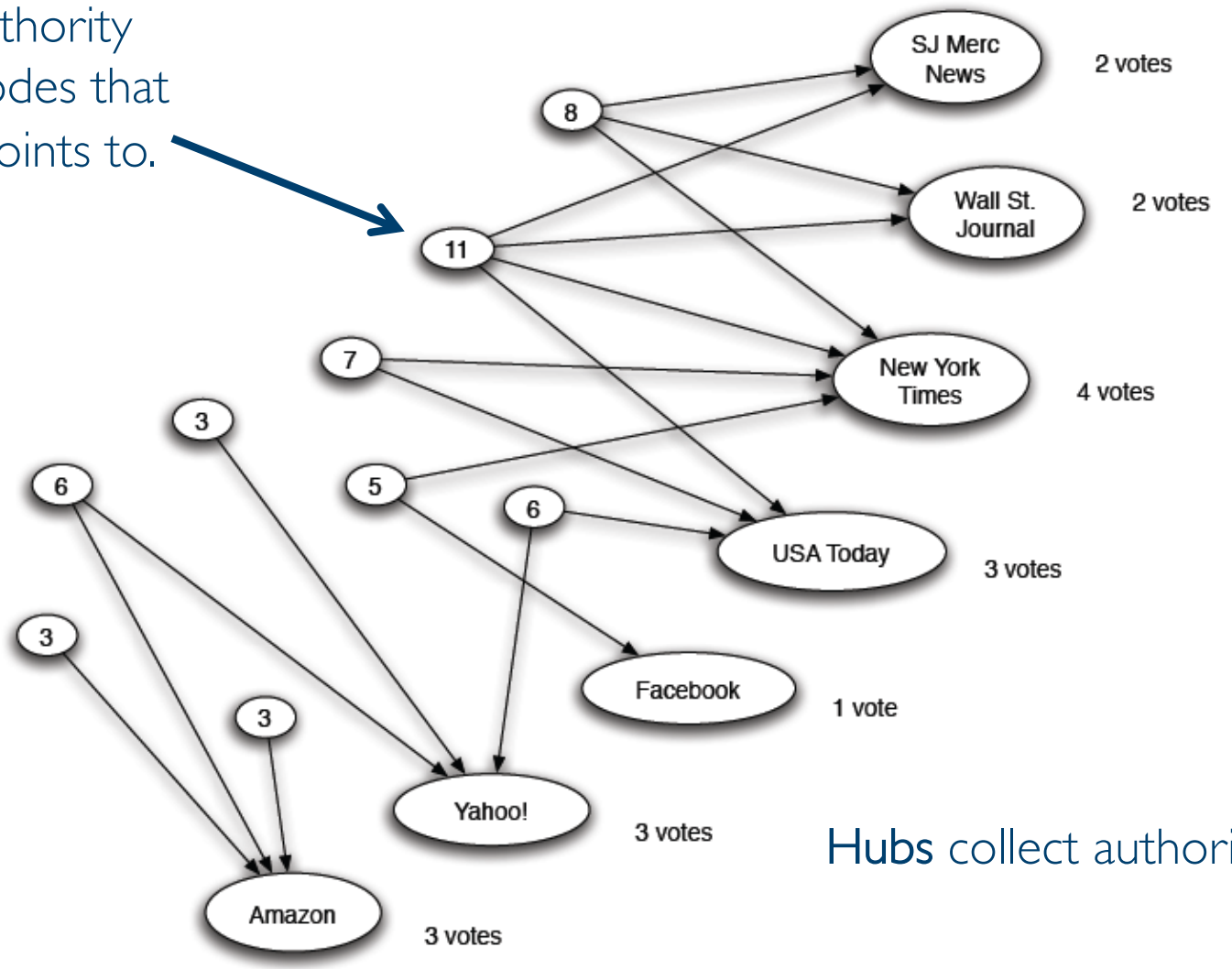


Sum of **hub** scores of nodes pointing to NYT.

Each page starts with **hub** score 1. **Authorities** collect their votes

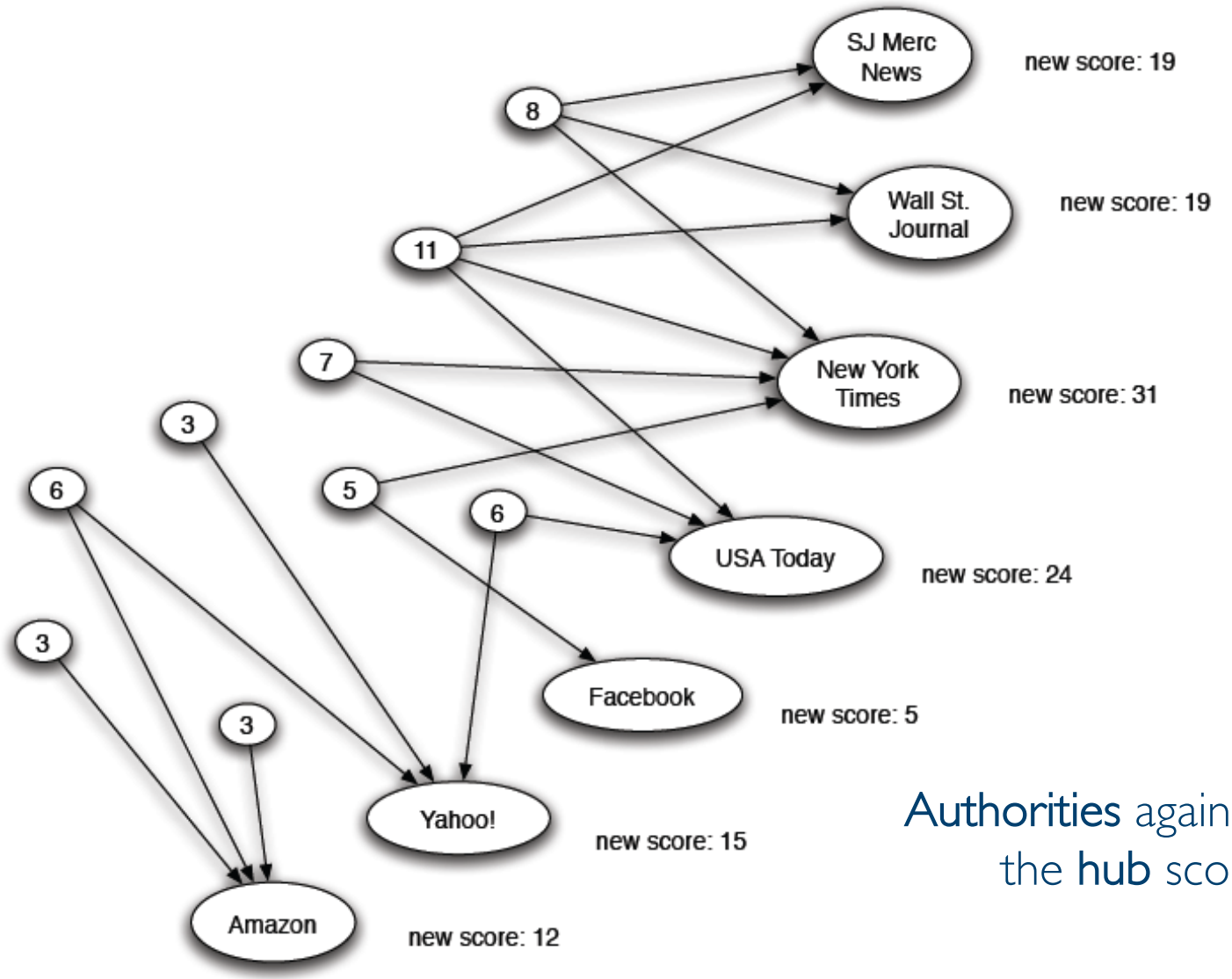
Expert Quality: Hub

Sum of authority scores of nodes that the node points to.



Hubs collect authority scores

Reweighting



Authorities again collect the hub scores

Mutually Recursive Definition

- A good hub links to many good authorities
- A good authority is linked from many good hubs
- Model using two scores for each node:
  - Hub score and Authority score
  - Represented as vectors and

The HITS Algorithm

- Initialize scores
- Iterate until convergence:
  - Update authority scores
  - Update hub scores
  - Normalize

- Two vectors a = (a_1, …, a_n) and h=(h_1, …, h_n) and the adjacency matrix A, with Aij=1 is 1 if i connects to j are connected, 0 otherwise

The HITS Algorithm (vector notation)

- Set $a_i = h_i = 1/\sqrt{n}$
- Repeat until convergence
  - $h = Aa$
  - $a = A^T h$
- Convergence criteria

$$\sum_i (h_i(t) - h_i(t-1))^2 < \varepsilon$$

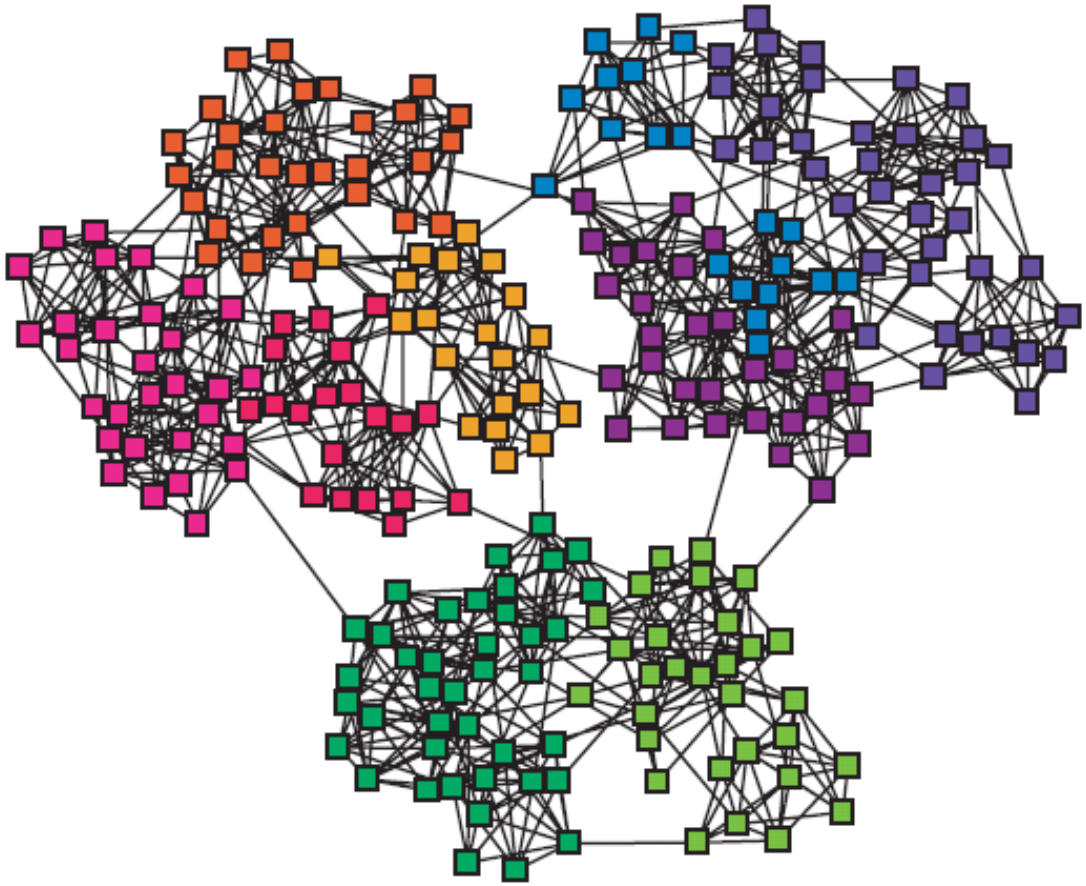
$$\sum_i (a_i(t) - a_i(t-1))^2 < \varepsilon$$

- Under reasonable assumptions about A, HITS converges to vectors h^* and a^* where
  - h^* is the principal eigenvector of matrix $A\,A^T$
  - a^* is the principal eigenvector of matrix $A^T A$

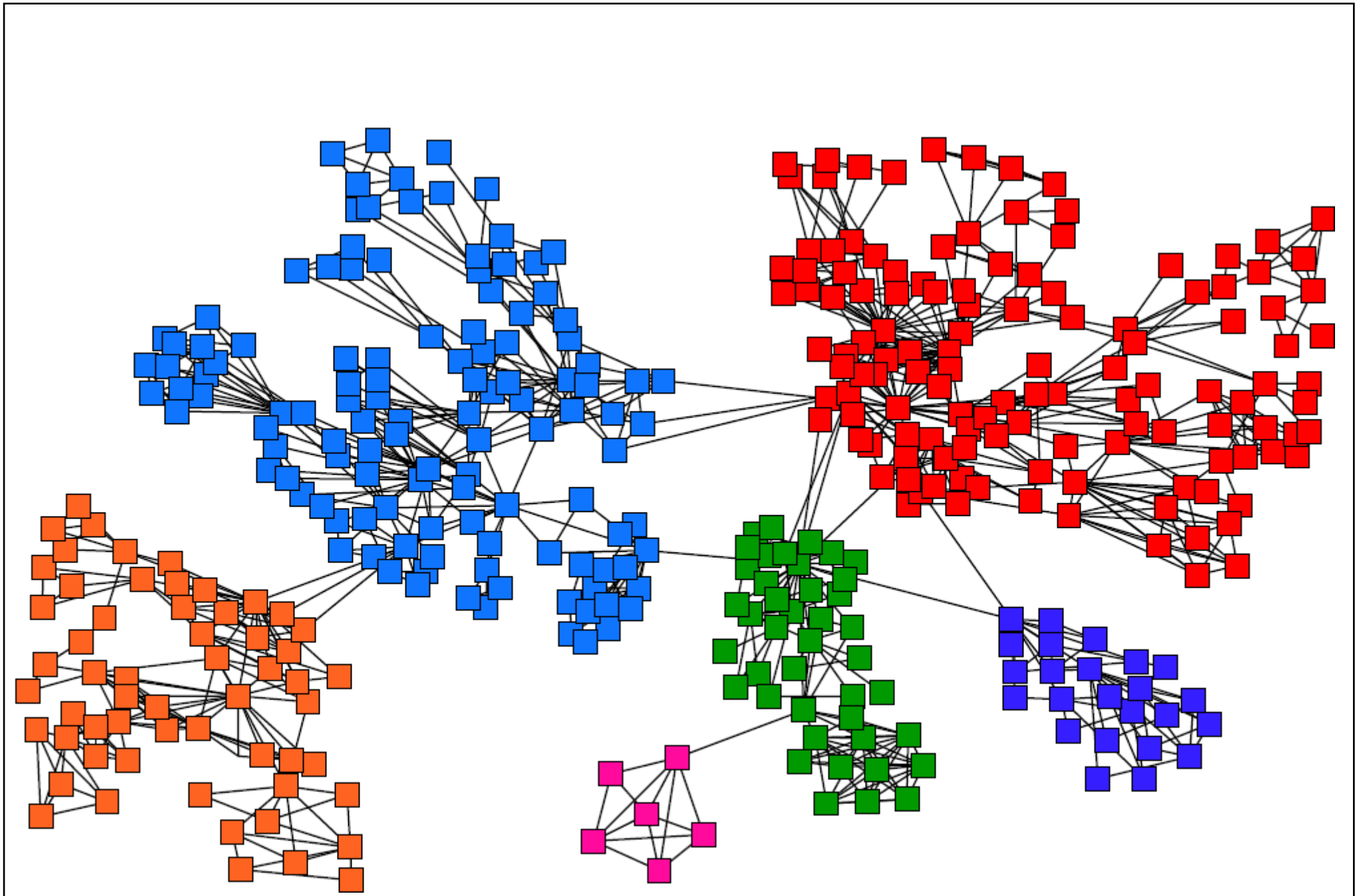
PageRank vs HITS

- PageRank and HITS are two solutions to the same problem
  - What is the value of an in-link from u to v?
  - In the PageRank model, the value of the link depends on the links into u
  - In the HITS model, it depends on the value of the other links out of u
- The destinies of PageRank and HITS after 1998 were very different

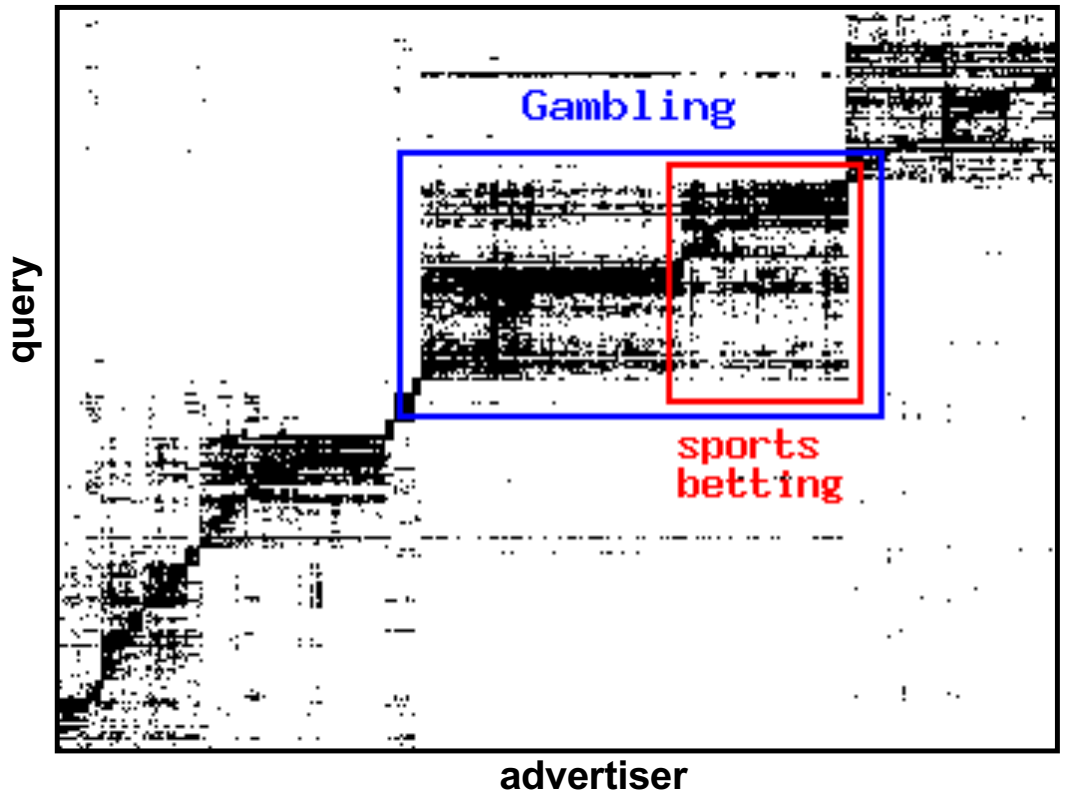
Community Detection



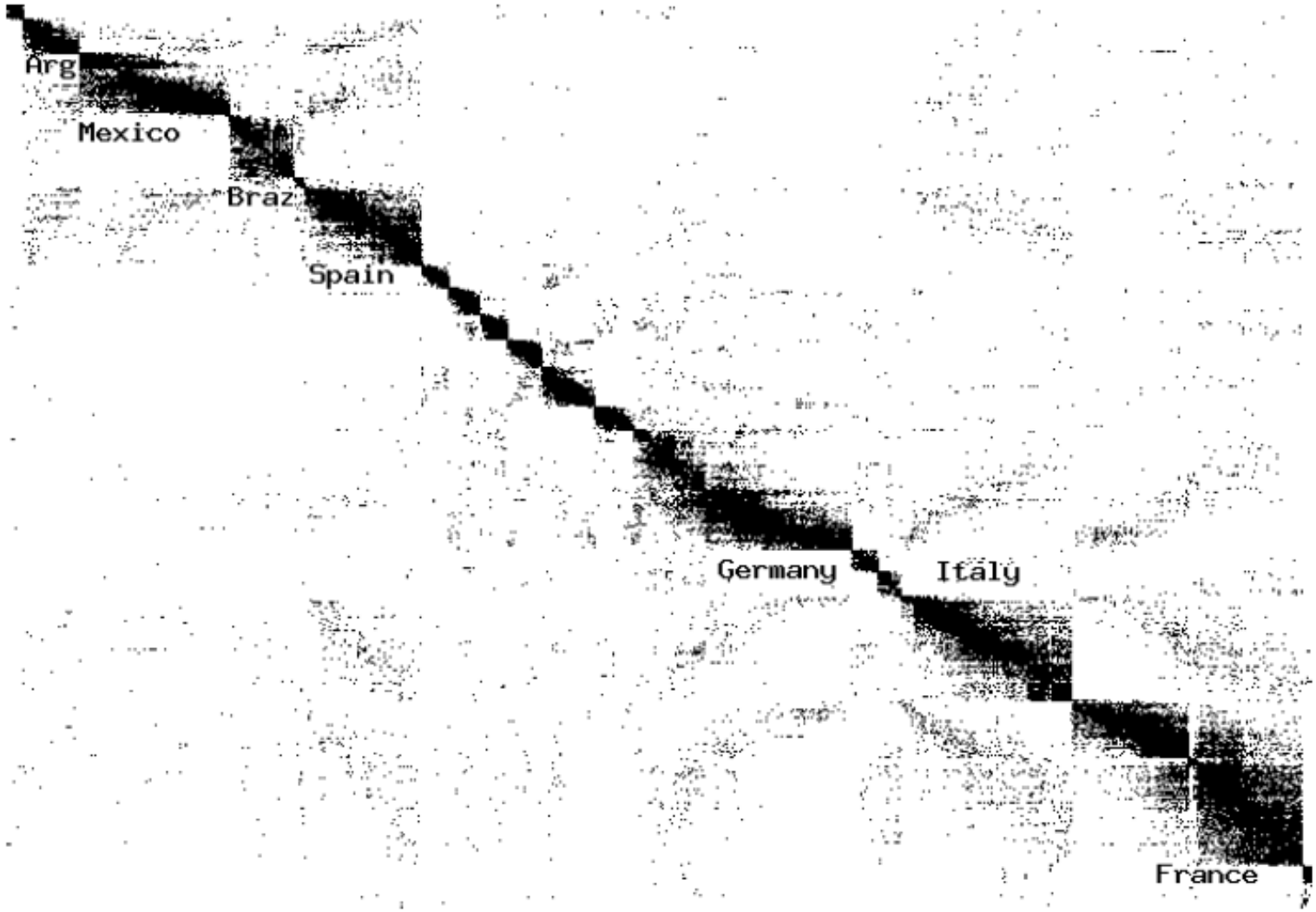
We often think of networks being organized into modules, cluster, communities:



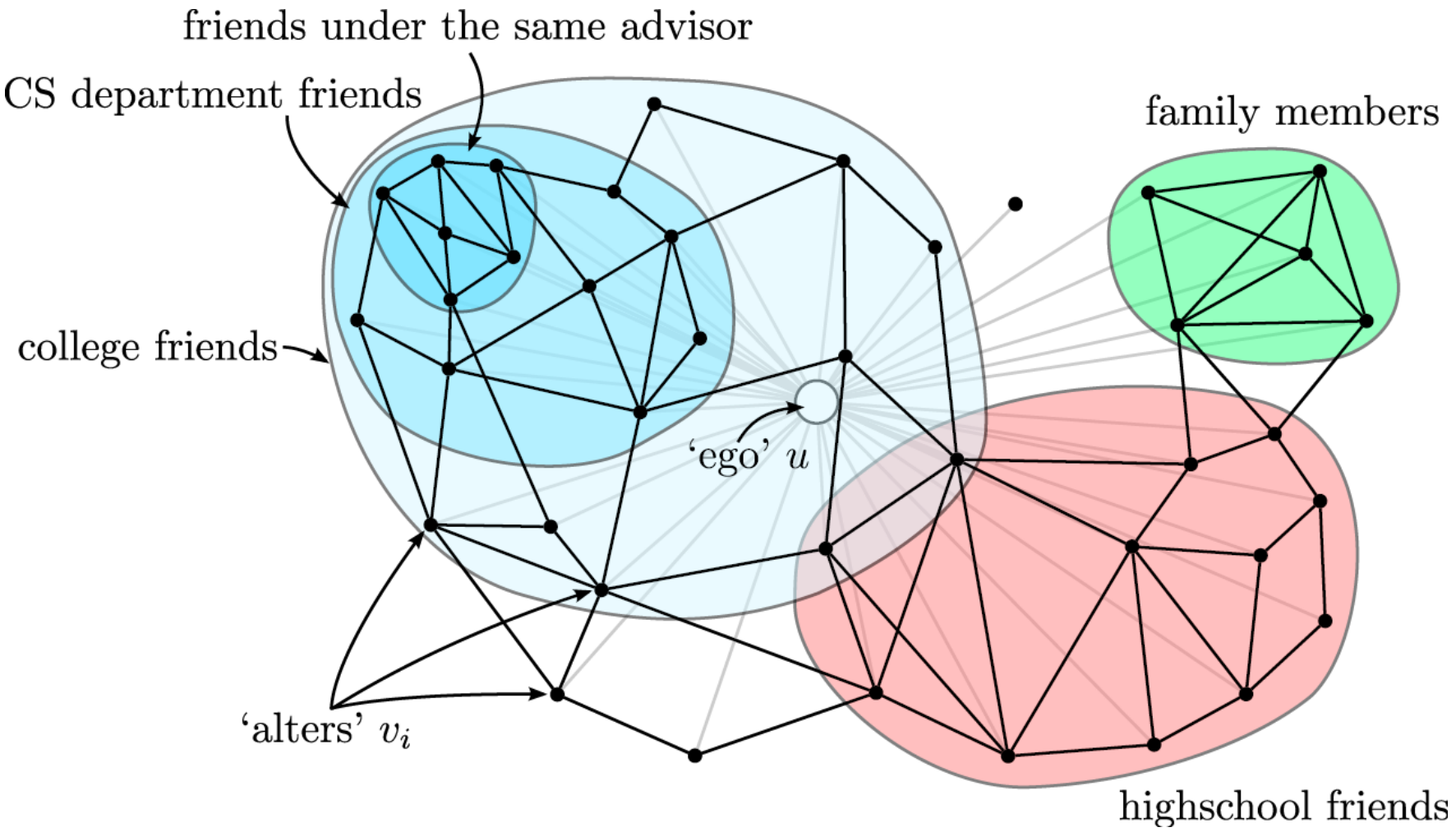
The goal is typically to find densely linked clusters



Micro-Markets in Sponsored Search: find micro-markets by partitioning the query-to-advertiser graph (Andersen, Lang: Communities from seed sets, 2006)



Clusters in Movies-to-Actors graph
(Andersen, Lang: Communities from seed sets, 2006)

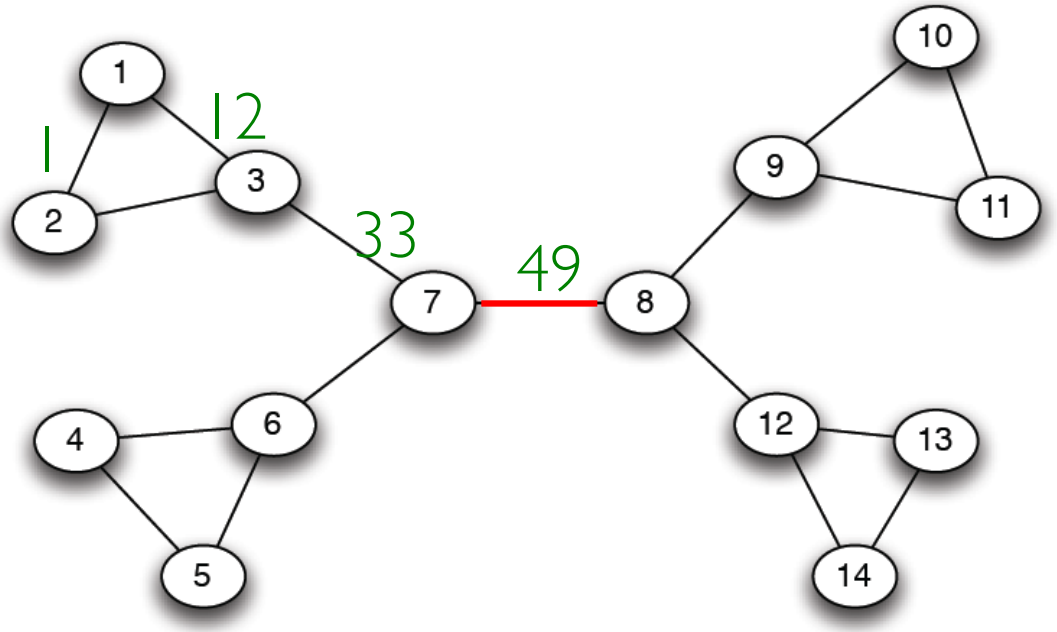


Discovering social circles, circles of trust
(McAuley, Leskovec: Discovering social circles in ego networks, 2012)

how can we identify communities?

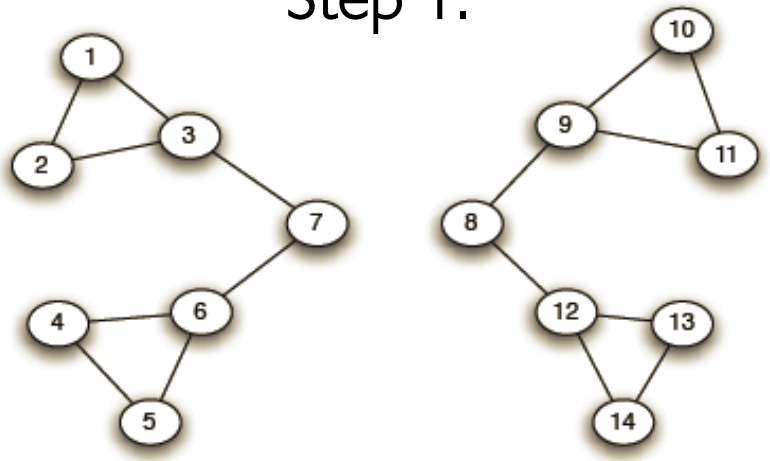
Girvan-Newman Method

- Define edge betweenness as the number of shortest paths passing over the edge
- Divisive hierarchical clustering based on the notion of edge betweenness
- The Algorithm
  - Start with an undirected graph
  - Repeat until no edges are left
    - Calculate betweenness of edges
    - Remove edges with highest betweenness
- Connected components are communities
- Gives a hierarchical decomposition of the network

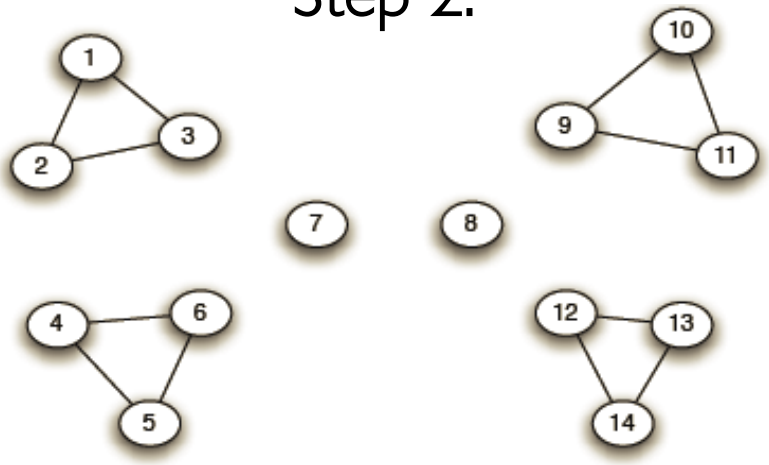


Need to re-compute
betweenness at every step

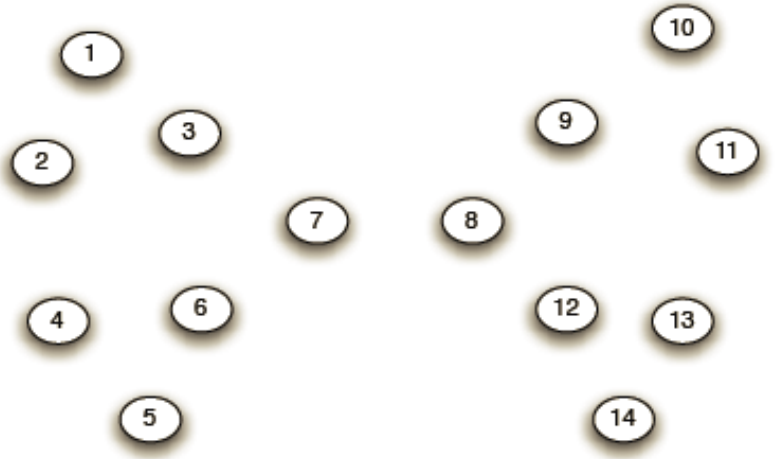
Step 1:



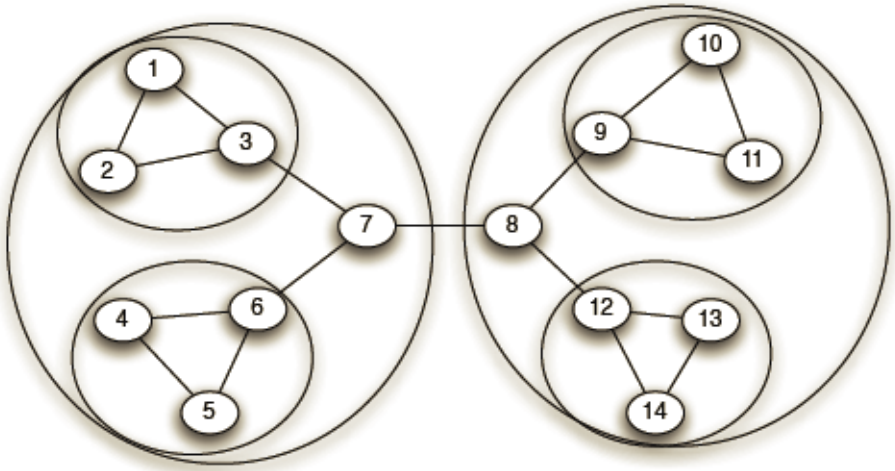
Step 2:

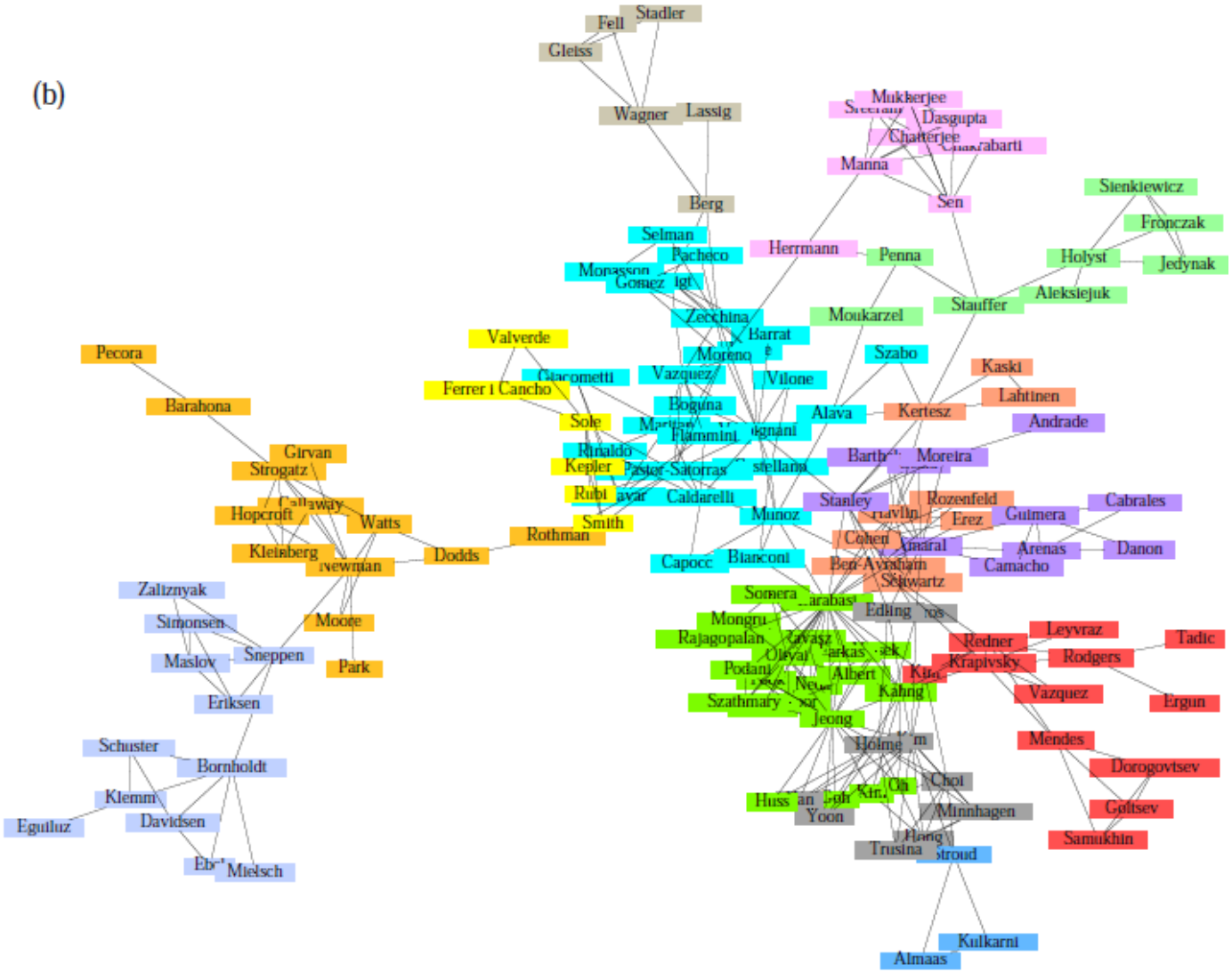


Step 3:



Hierarchical network decomposition



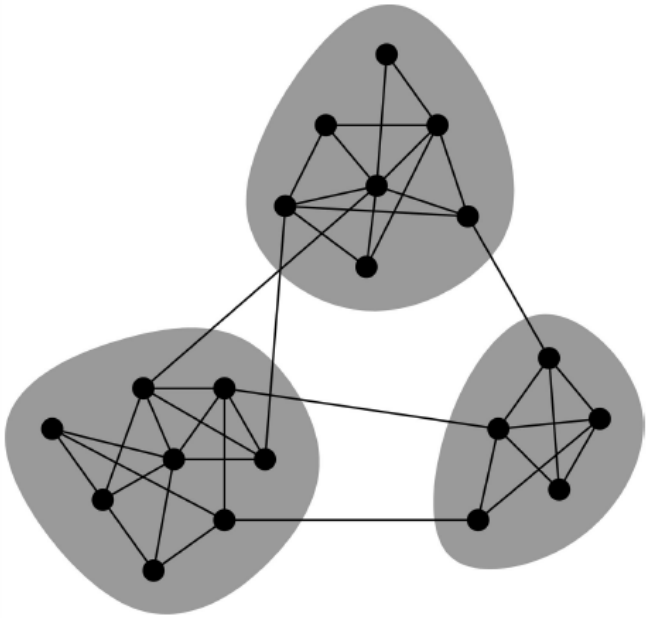


Communities in physics collaborations

how to select the number of clusters?

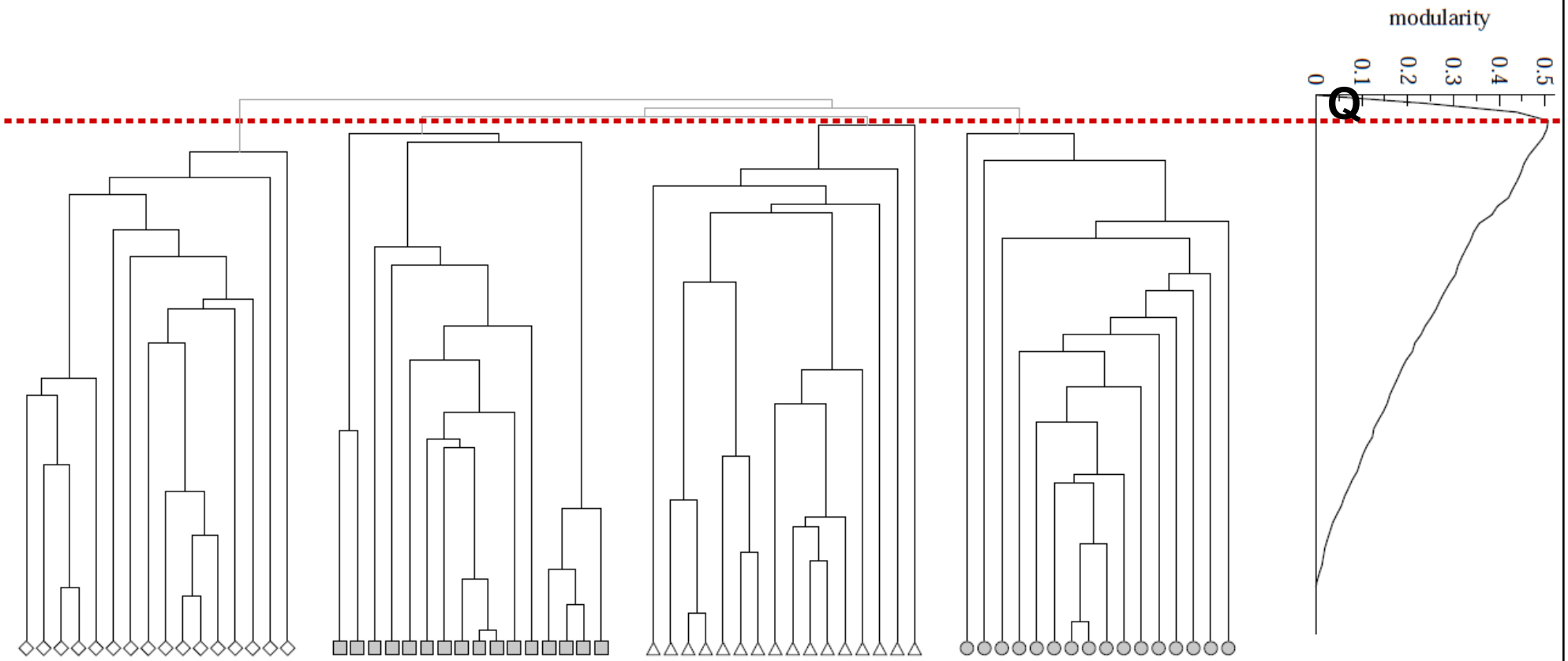
Network Communities

- Communities are viewed as sets of tightly connected nodes
- We define modularity as a measure of how well a network is partitioned into communities
- Given a partitioning of the network into a set of groups S we define the modularity Q as



$$Q \propto \sum_{s \in S} [(\#\text{ edges within group s}) - (\text{expected }\#\text{ edges within group s})]$$

Need a null model!

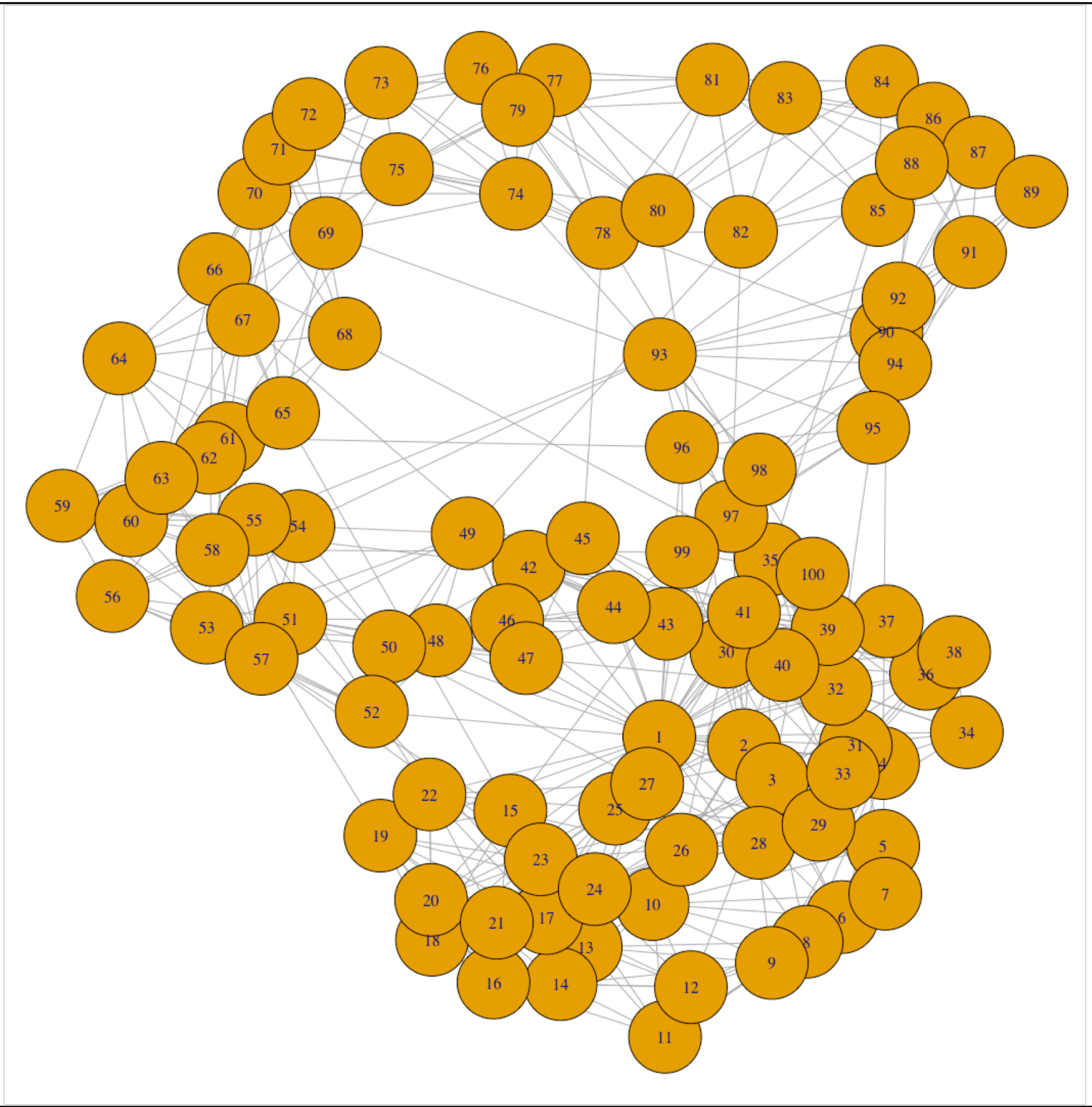


Modularity is useful for selecting the number of clusters:

Example

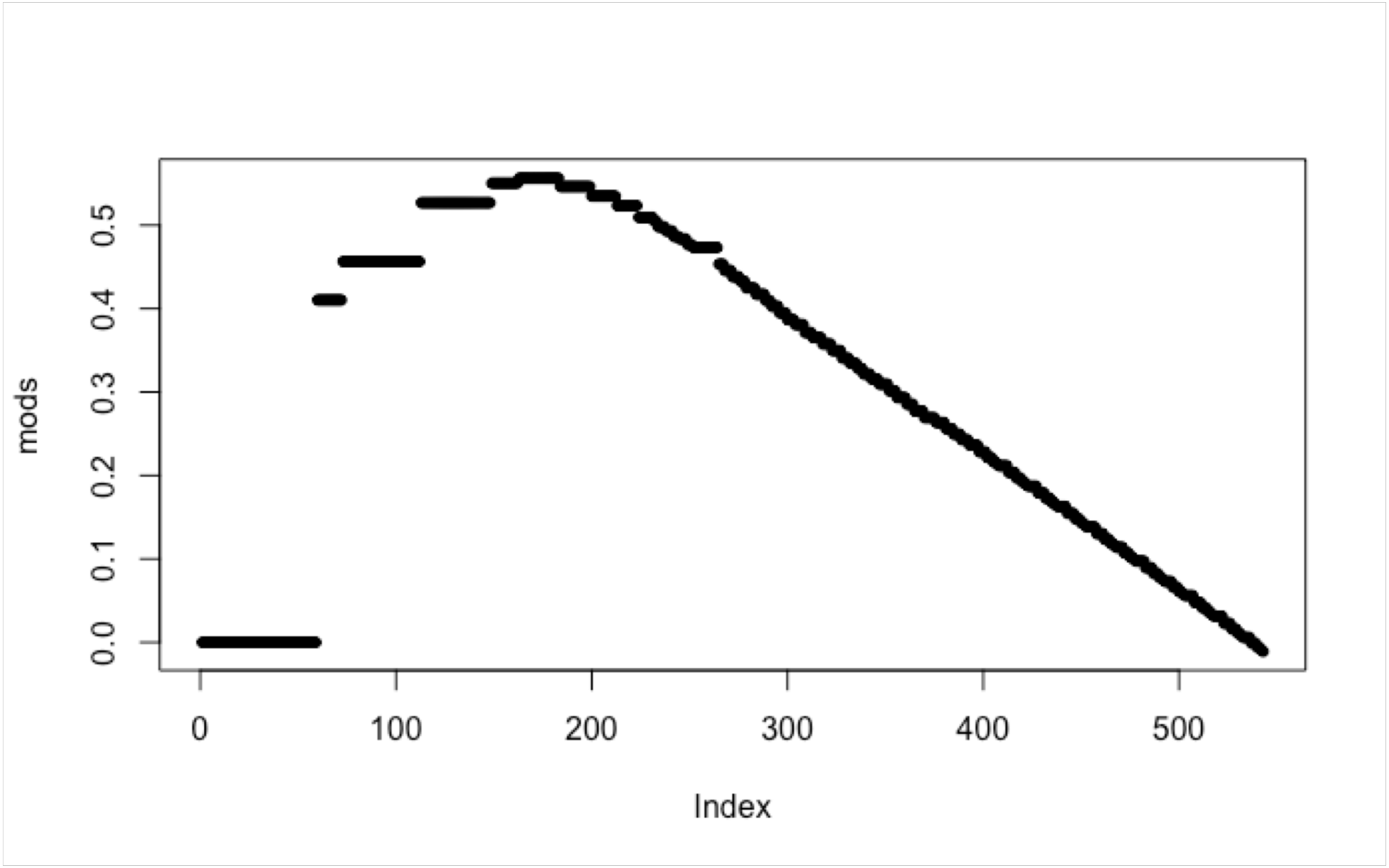
Example #1 – Compute betweenness


```
# First we load the ipgrah package
library(igraph)

# let's generate two networks and merge them into one graph.
g2 <- barabasi.game(50, p=2, directed=F)
g1 <- watts.strogatz.game(1, size=100, nei=5, p=0.05)
g <- graph.union(g1,g2)

# let's remove multi-edges and loops
g <- simplify(g)

plot(g)

# compute betweenness
ebc <- edge.betweenness.community(g, directed=F)
```


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

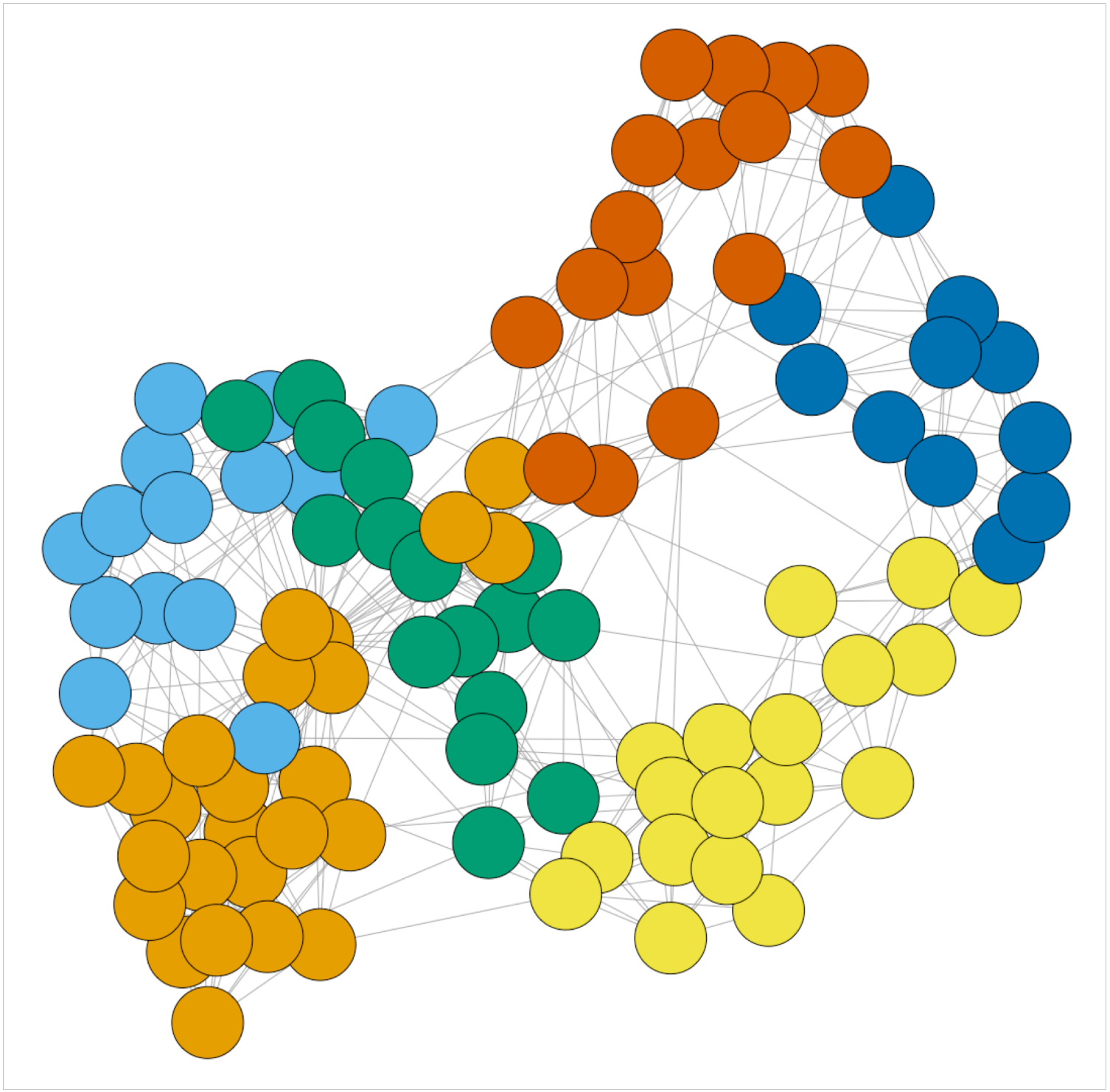
Example #1 – Build Dendrogram and Compute Modularity


```
mods <- sapply(0:ecount(g), function(i){
  g2 <- delete.edges(g, ebc$removed.edges[seq(length=i)])
  cl <- clusters(g2)$membership
  # March 13, 2014 - compute modularity on the original graph g
  # (Thank you to Augustin Luna for detecting this typo) and not on the
induced one g2.
  modularity(g,cl)
})

# we can now plot all modularities
plot(mods, pch=20)
```


mods
0.0
0.1
0.2
0.3
0.4
0.5
0
100
200
300
400
500
Index

Example #1 – Select the cut

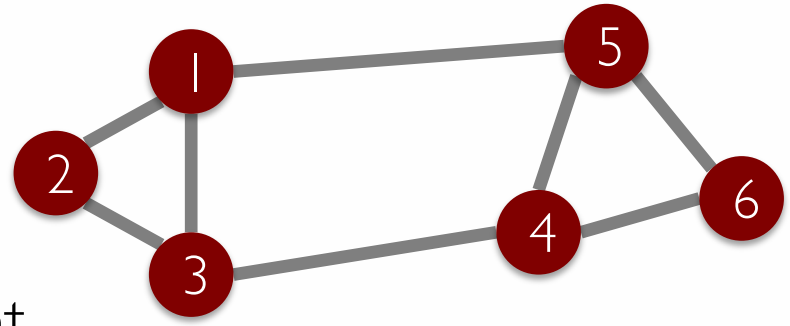

```
# Now, let's color the nodes according to their membership
g2<-delete.edges(g, ebc$removed.edges[seq(length=which.max(mods)-1)])
V(g)$color=clusters(g2)$membership

# Let's choose a layout for the graph
g$layout <- layout.fruchterman.reingold

# plot it
plot(g, vertex.label=NA)
```



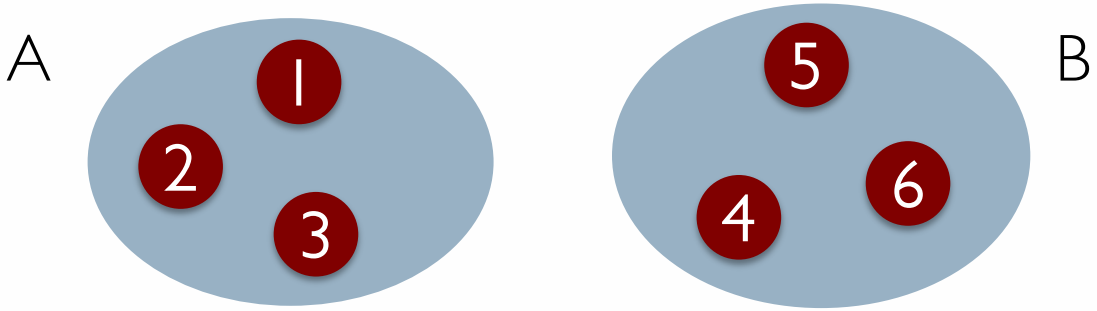
Spectral Clustering

What Makes a Good Cluster?

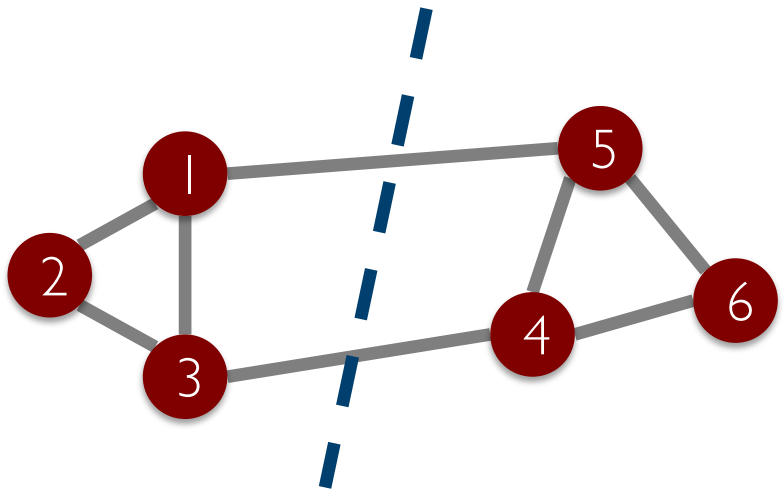
- Undirected graph G(V,E)



- Partitioning task
  - Divide the vertices into two disjoint groups A, B=V\A



- Questions
  - How can we define a "good partition" of G?
  - How can we efficiently identify such a partition?

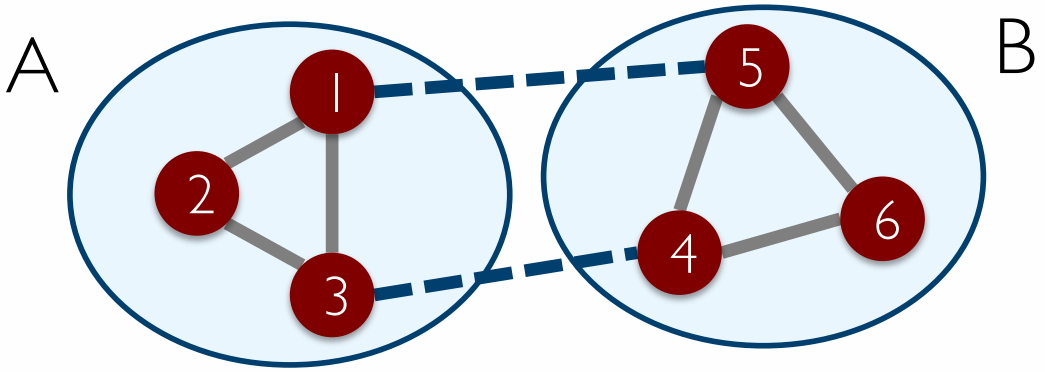


What makes a good partition?

Maximize the number of within-group connections

Minimize the number of between-group connections

- Express partitioning objectives as a function of the "edge cut" of the partition
- Cut is defined as the set of edges with only one vertex in a group

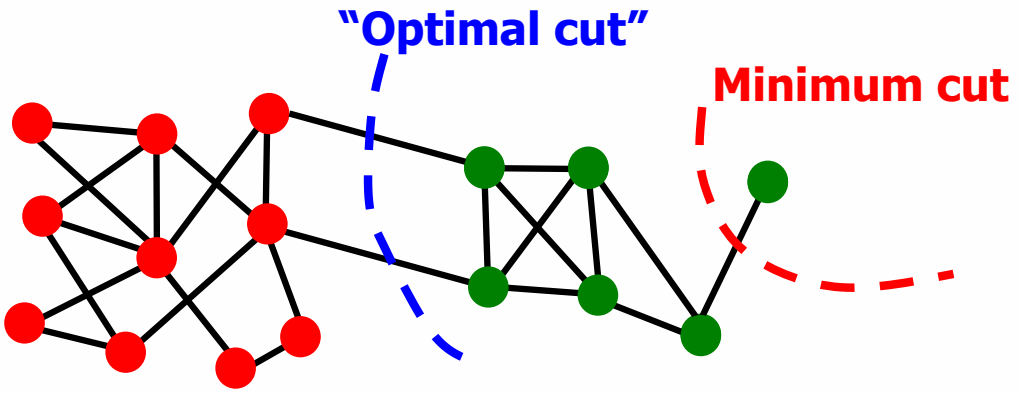


- The cut of the set A, B is cut(A,B) = 2 or in more general

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

Graph Cut Criterion

- Partition quality
  - Minimize weight of connections between groups, i.e., arg $\min_{A,B}$ cut(A,B)
- Degenerate case:



- Problems
  - Only considers external cluster connections
  - Does not consider internal cluster connectivity

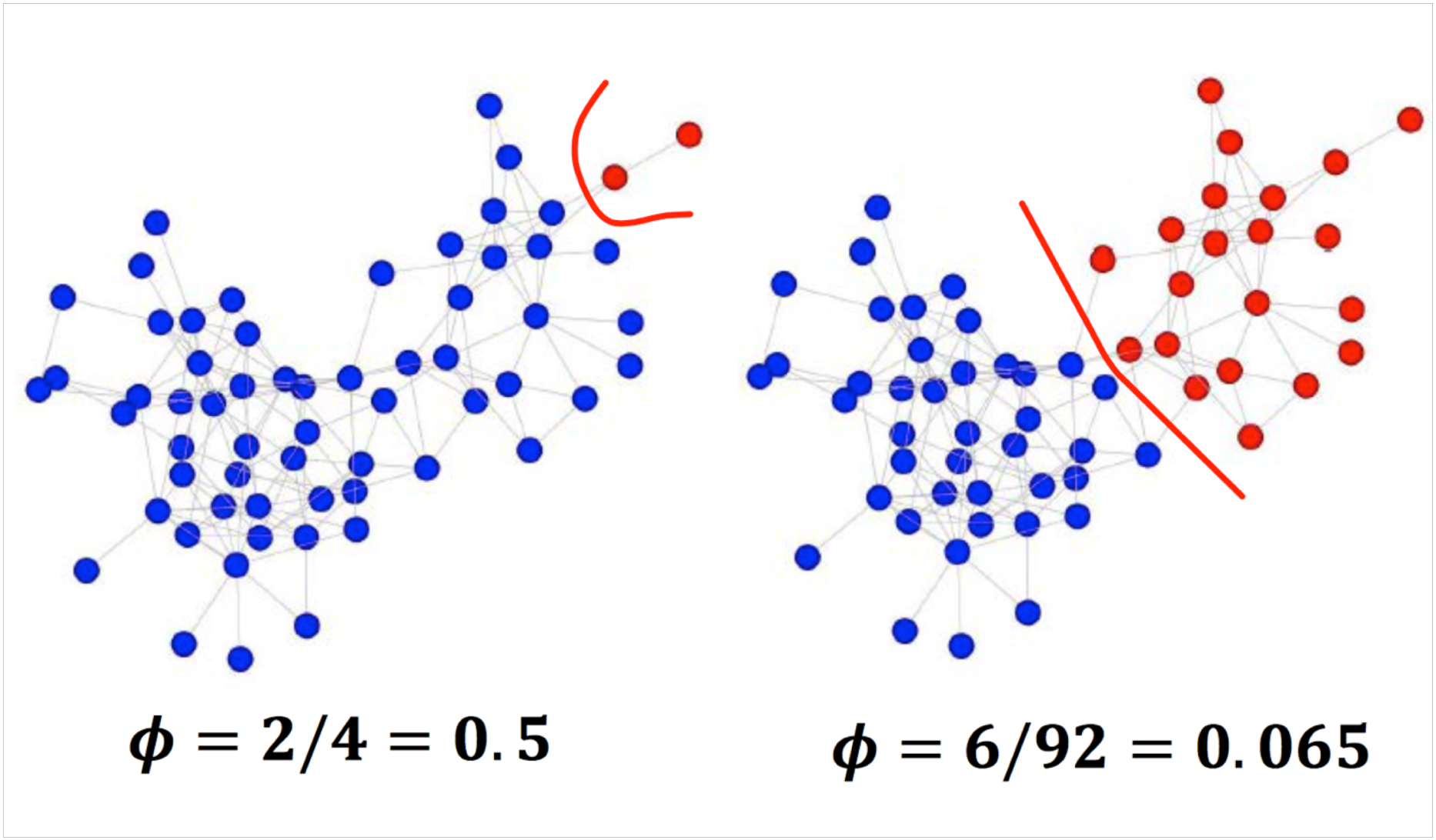
Graph Partitioning Criteria: Normalized cut (Conductance)

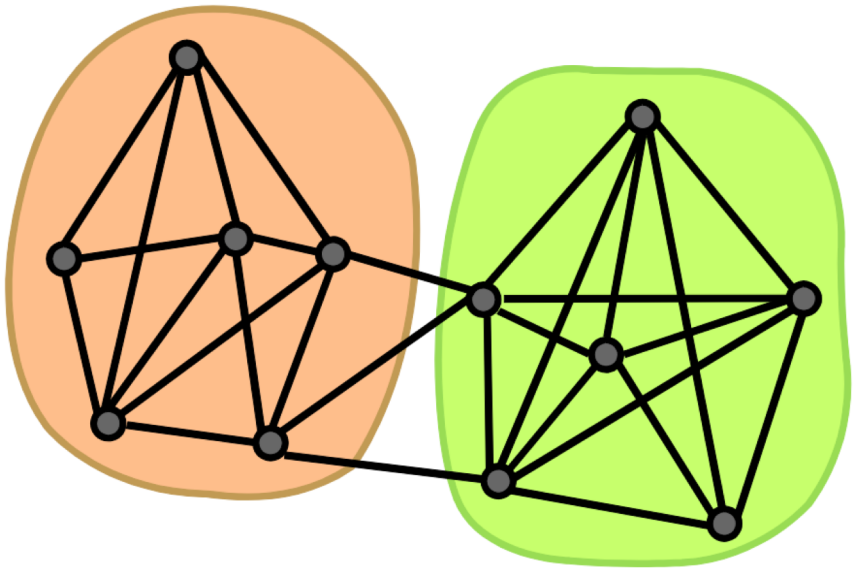
- Connectivity of the group to the rest of the network should be relative to the density of the group

$$\text{ncut}(\text{A},\text{B}) = \frac{\text{cut}(\text{A},\text{B})}{\text{vol}(\text{A})} + \frac{\text{cut}(\text{A},\text{B})}{\text{vol}(\text{B})}$$

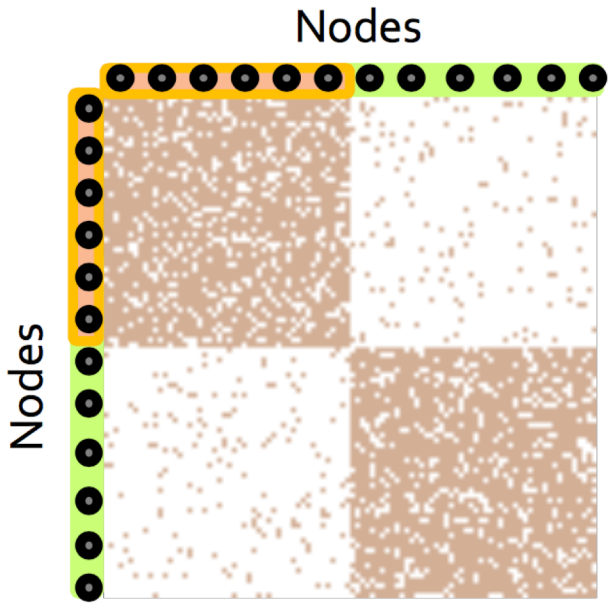
$$\phi(A) = \frac{\text{cut}(\text{A})}{\text{vol}(\text{A})}$$

- Where vol(A) is the total weight of the edges that have at least one endpoint in A


$\phi = 2/4 = 0.5$
$\phi = 6/92 = 0.065$



Network



Adjacency matrix

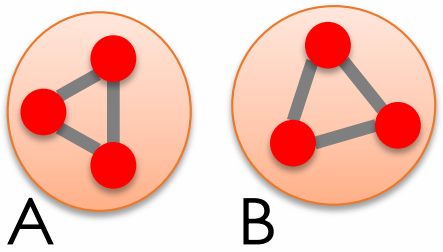
Spectral Graph Partitioning

- Let A be the adjacent matrix of the graph G with n nodes
  - A_{ij} is 1 if there is an edge between i and j, 0 otherwise
  - x a vector of n components $(x_1, \ldots, x_n)$ that represents labels/values assigned to each node of G
  - Ax returns a vector in which each component j is the sum of the labels of the neighbors of node j

- Spectral Graph Theory
  - Analyze the spectrum of G, that is, the eigenvectors x_i of the graph corresponding to the eigenvalues Λ of G sorted in increasing order
  - $\Lambda = \{ \lambda_1, \ldots, \lambda_n \}$ such that $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$

Example: d-regular Graph

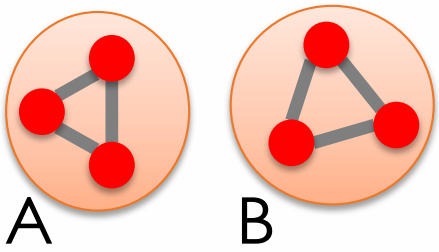
- Suppose that all the nodes in G have degree d and G is connected
- What are the eigenvalues/eigenvectors of G? $Ax=\lambda x$
  - Ax returns the sum of the labels of each node's neighbors and since each node has exactly d neighbors, x = (1, …, 1) is an eigenvector and d is an eigenvalue
- What if G is not connected but still d-regular



- A vector with all the ones is A and all the zeros in B (or viceversa) is still an eigenvector of A with eigenvalue d

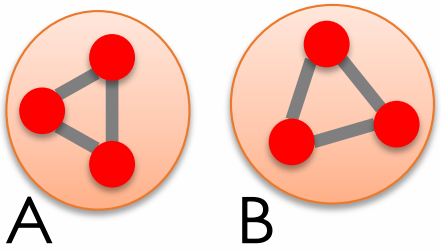
Example: d-regular Graph (not connected)

- What if G has two separate components but it is still d-regular

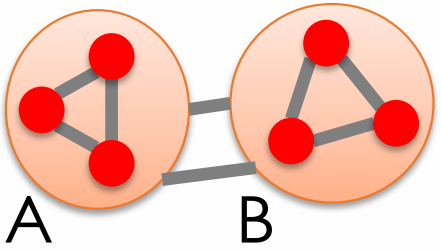


- A vector with all the ones is A and all the zeros in B (or viceversa) is still an eigenvector of A with eigenvalue d

- Underlying intuition



$\lambda_1 = \lambda_2$



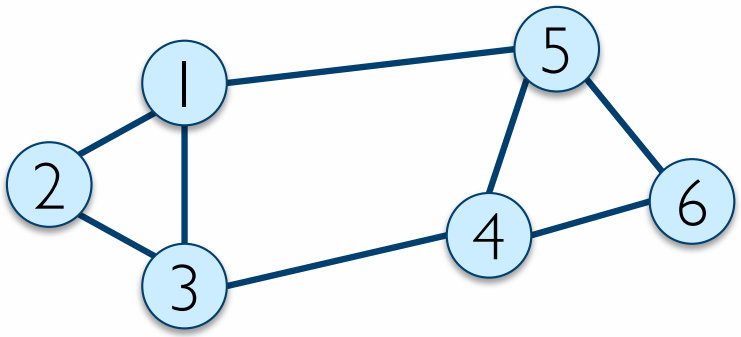
$\lambda_1 \approx \lambda_2$

Spectral Graph Partitioning

- Adjacency matrix A (nxn)
  - Symmetric
  - Real and orthogonal eigenvectors
- Degree Matrix
  - nxn diagonal matrix
  - D_{ii} = degree of node i

	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	3	0
6	0	0	0	0	0	2



Graph Laplacian Matrix

- Computed as L = D-A
  - nxn symmetric matrix
  - x=(1,…,1) is a trivial eigenvector since Lx=0 so λ_1=0

- Important properties of L
  - Eigenvalues are non-negative real numbers
  - Eigenvectors are real and orthogonal

	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

λ_2 as optimization problem

- For symmetric matrix M,

$$\lambda_2 = \min_x \frac{x^T M x}{x^T x}$$

- What is the meaning of $x^T L x$ on G? We can show that,

$$x^T L x = \sum_{(i,j) \in E} (x_i - x_j)^2$$

- So that, considering that the second eigenvector x is the unit vector, and x is orthogonal to the unit vector (1, …, 1)

$$\lambda_2 = \min_x \frac{\sum_{(i,j) \in E} (x_i - x_j)^2}{\sum_i x_i^2}$$

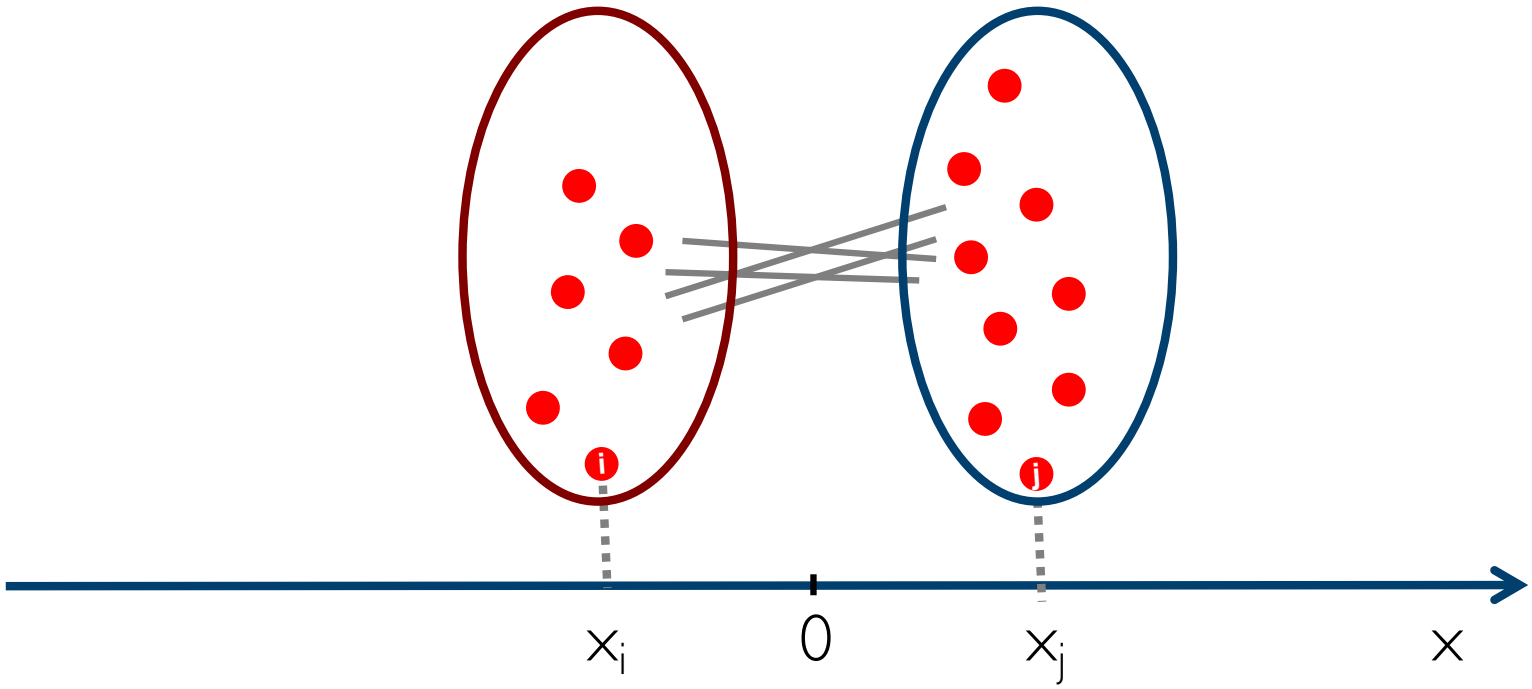
λ_2 as optimization problem

- So that, considering that the second eigenvector x is the unit vector, and x is orthogonal to the unit vector (1, …, 1)

$$\lambda_2 = \min_x \frac{\sum_{(i,j)\in E}(x_i - x_j)^2}{\sum_i x_i^2}$$

- Such that,

$$\sum_i x_i = 0$$



λ_2 and its eigenvector x balance to minimize

Finding the Optimal Cut

- Express the partition (A,B) as a vector y where,
  - $y_i = +1$ if node i belongs to A
  - $y_i = -1$ if node i belongs to B

- We can minimize the cut of the partition by finding a non-trivial vector that minimizes

$$\arg \min_{y \in [-1,+1]^n} f(y) = \sum_{(i,j) \in E} (y_i - y_j)^2$$

Can't solve exactly! Let's relax y and allow y to take any real value.

Rayleigh Theorem

$$\arg\min_{y \in R^n} f(y) = \sum_{(i,j) \in E} (y_i - y_j)^2 = y^T L y$$

- We know that,

$$\lambda_2 = \min_y f(y)$$

- The minimum value of f(y) is given by the second smallest eigenvalue $\boldsymbol{\lambda}_2$ of the Laplacian matrix L

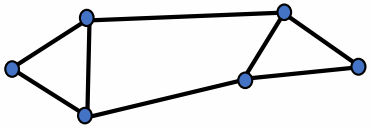
- Thus, the optimal solution for y is given by the corresponding eigenvector x, referred as the Fiedler vector

Spectral Clustering Algorithms

1. Pre-processing
   - Construct a matrix representation of the graph
2. Decomposition
   - Compute eigenvalues and eigenvectors of the matrix
   - Map each point to a lower-dimensional representation based on one or more eigenvectors
3. Grouping
   - Assign points to two or more clusters, based on the new representation

Spectral Partitioning Algorithm

- Pre-processing:
  - Build Laplacian matrix L of the graph



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

- Decomposition:
  - Find eigenvalues λ and eigenvectors x of the matrix L
  - Map vertices to corresponding components of λ_2

$\lambda =$

0.0
1.0
3.0
3.0
4.0
5.0

$X =$

0.4	0.3	-0.5	-0.2	-0.4	-0.5
0.4	0.6	0.4	-0.4	0.4	0.0
0.4	0.3	0.1	0.6	-0.4	0.5
0.4	-0.3	0.1	0.6	0.4	-0.5
0.4	-0.3	-0.5	-0.2	0.4	0.5
0.4	-0.6	0.4	-0.4	-0.4	0.0

1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6

Spectral Partitioning Algorithm

- Grouping:
  - Sort components of reduced 1-dimensional vector
  - Identify clusters by splitting the sorted vector in two
- How to choose a splitting point?
  - Naïve approaches: split at 0 or median value
  - More expensive approaches: Attempt to minimize normalized cut in 1-dimension (sweep over ordering of nodes induced by the eigenvector)

1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6

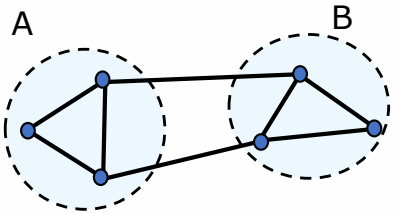
Split at 0:

Cluster A: Positive points

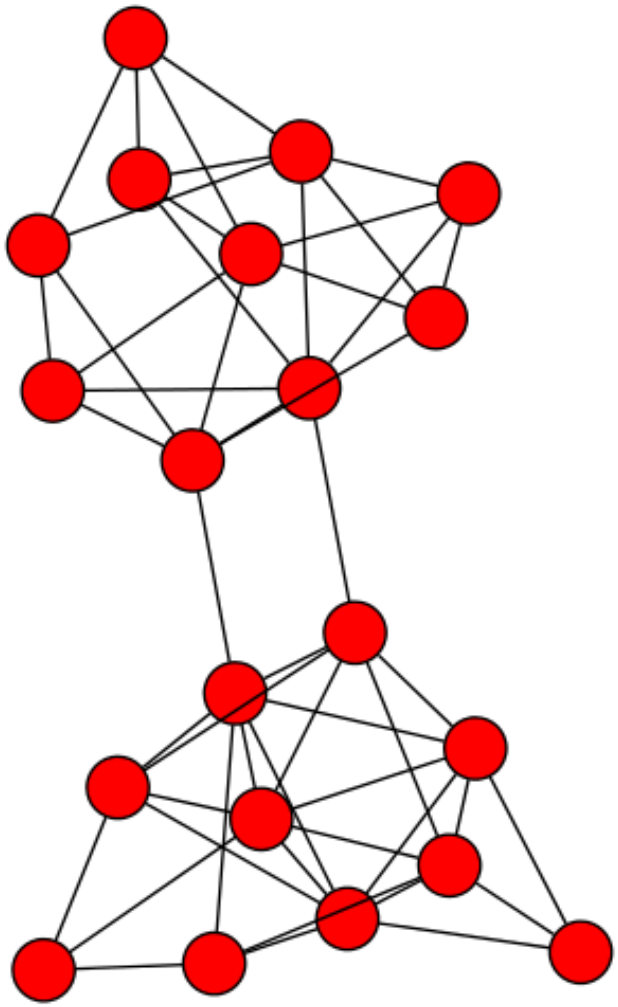
Cluster B: Negative points

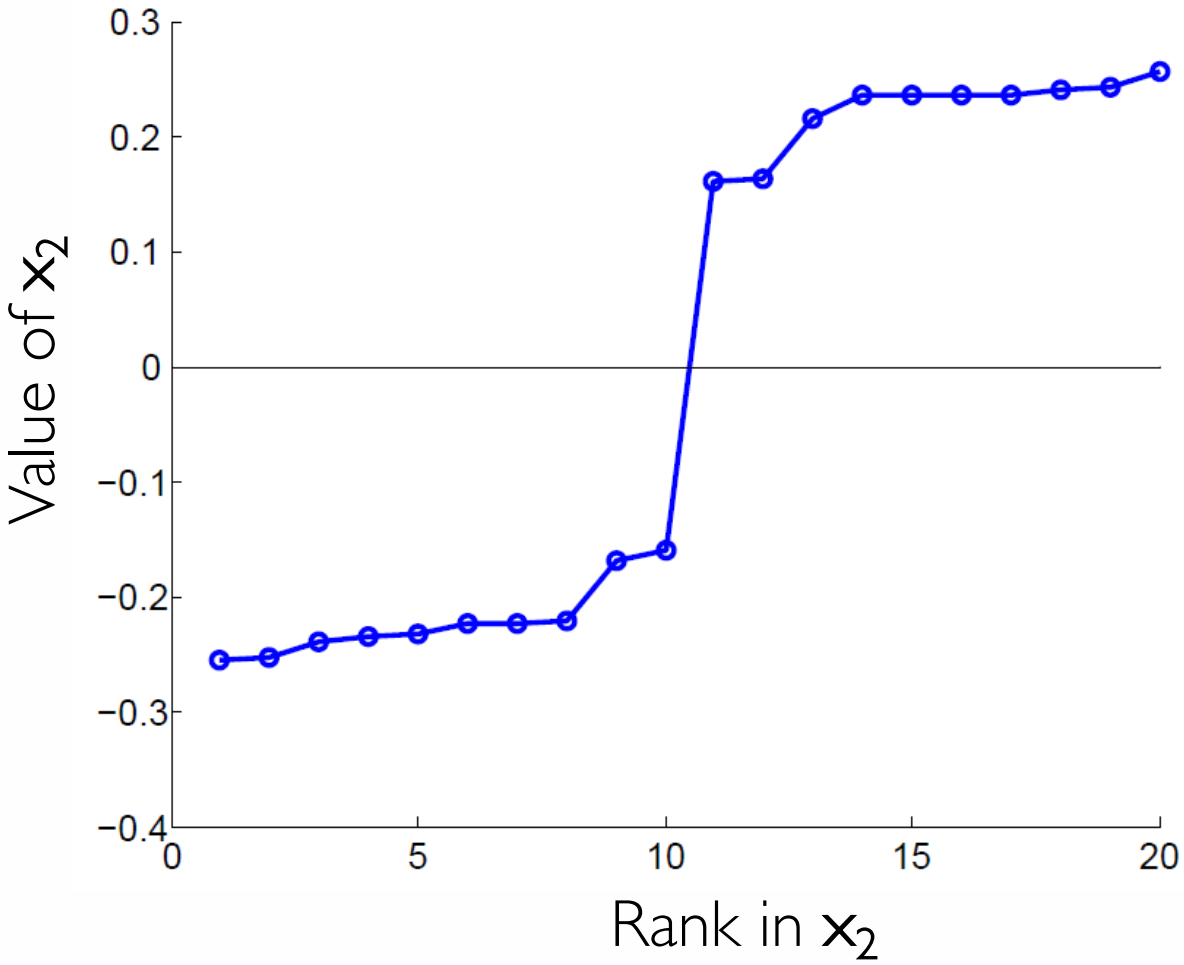
1	0.3
2	0.6
3	0.3

4	**-0.3**
5	-0.3
6	-0.6

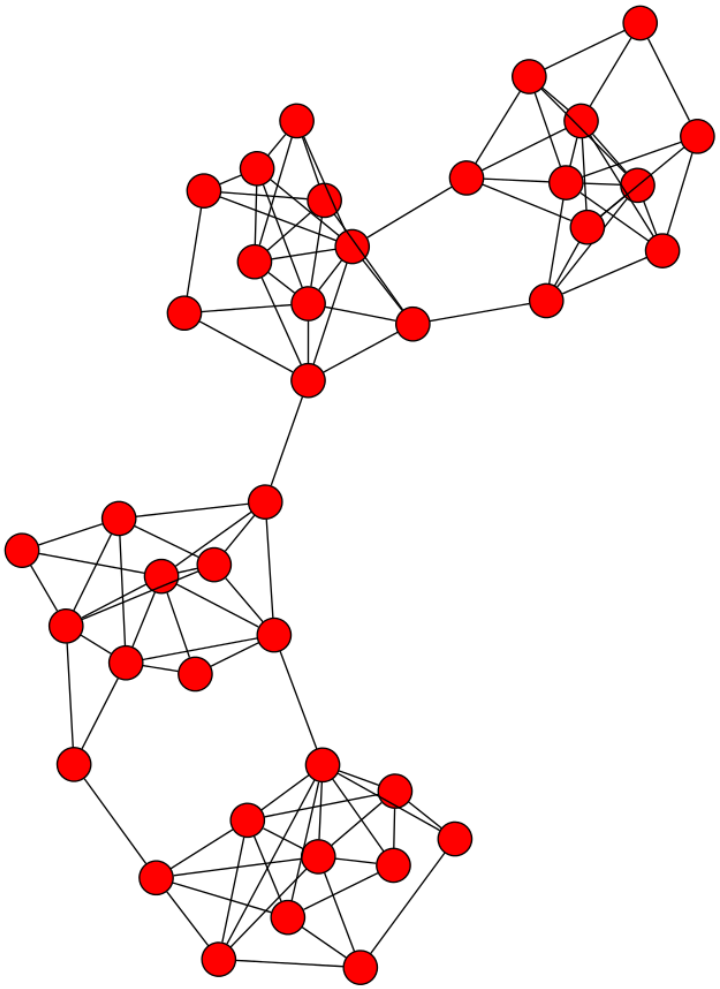


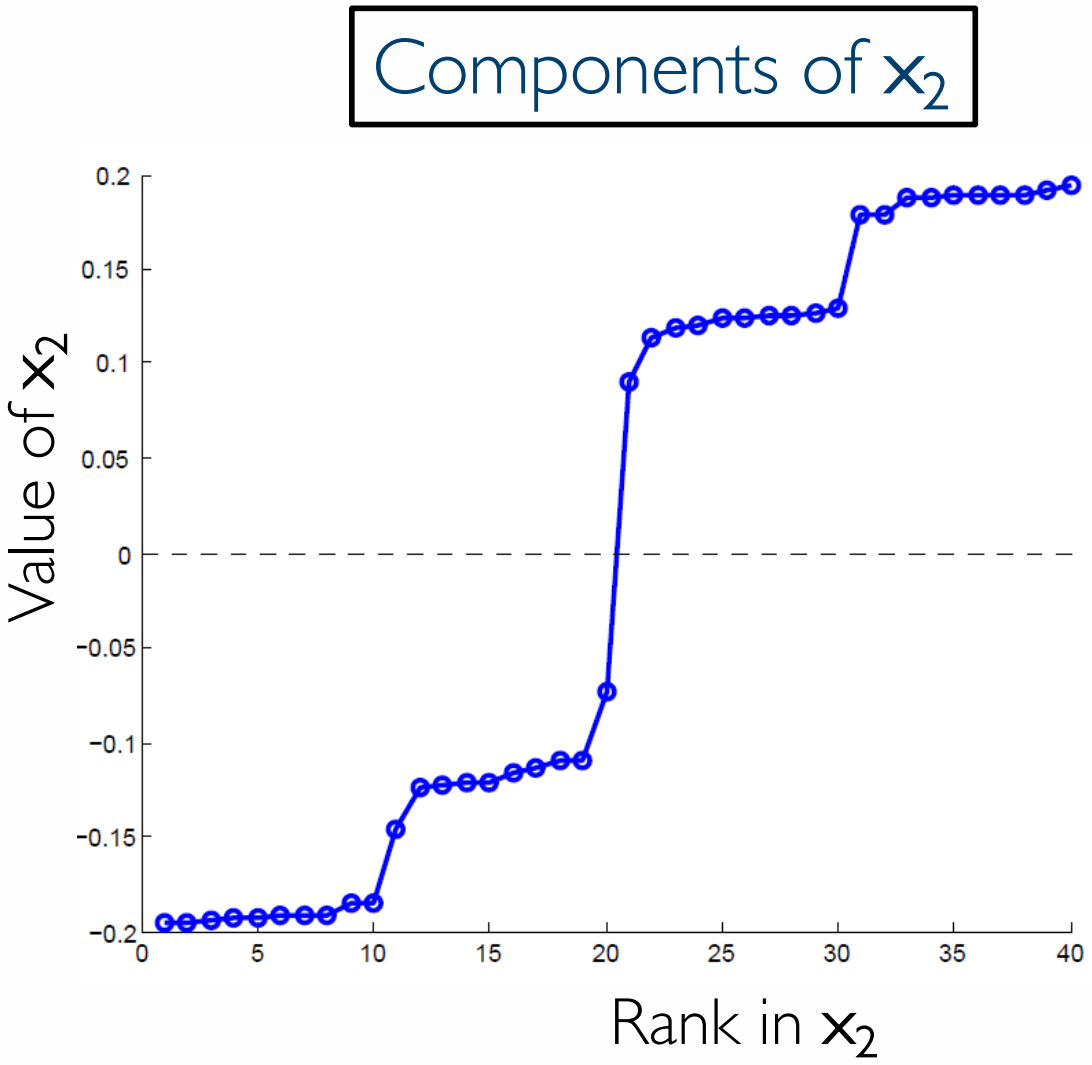
Example: Spectral Partitioning



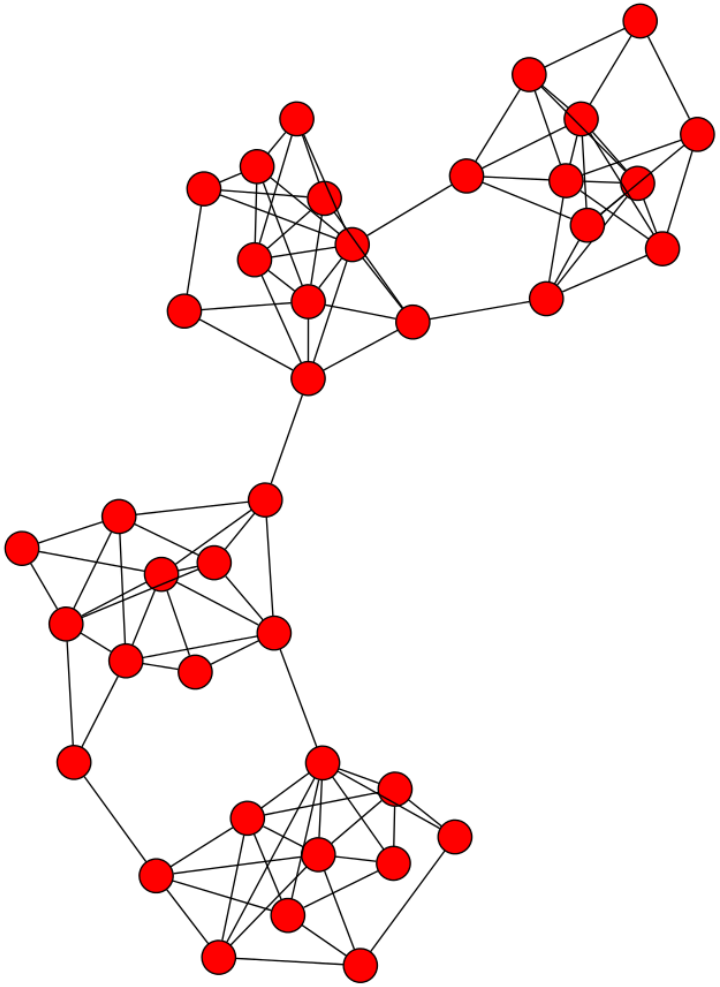


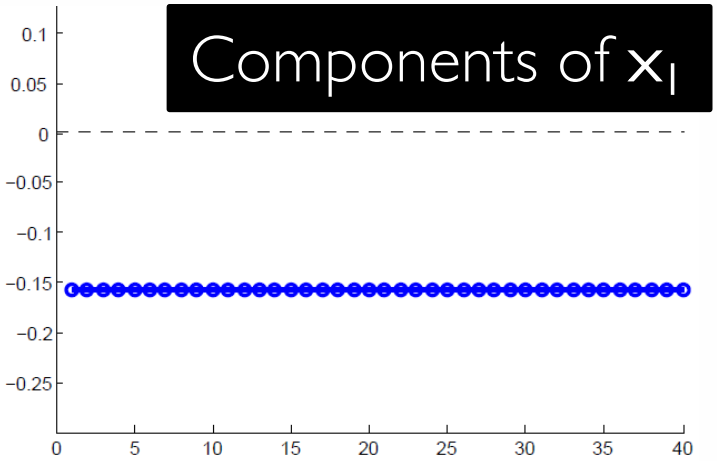
Example: Spectral Partitioning

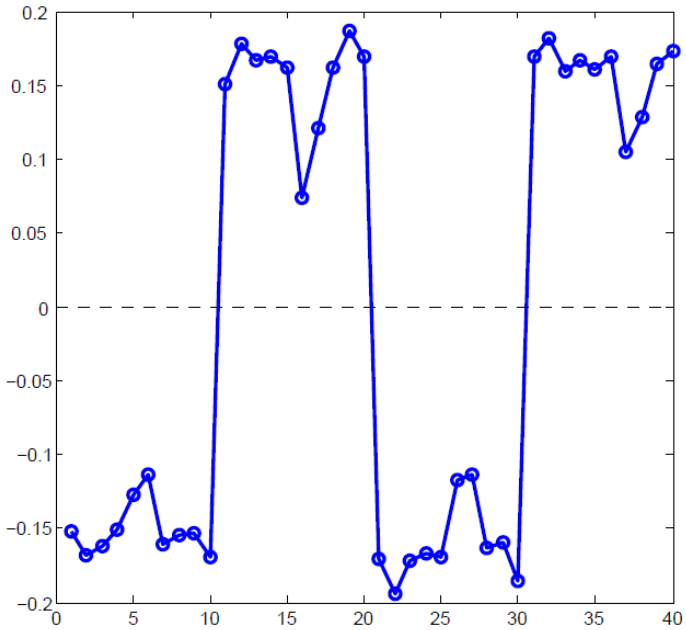




Example: Spectral partitioning







Components of x_3

Example using R

Example #1


```
ComputeDegreeMatrix <- function(v) {
   n <- max(dim(v))
   m <- diag(n)
   d = v %*% rep(1,n)
   for(i in 1:n ) {
      m[i,i] = d[i,1]
   }
   return(m)
}

ComputeLaplacianMatrix <- function(v) {
  D = ComputeDegreeMatrix(v)
  L = D-v
  return(L)
}
```

Example #1


```
####    2 ----6
####  /  \    |
#### 1    4   |
####  \  /  \ |
####   3      5
```

```
G = rbind(
  c(0, 1, 1, 0, 0, 0),
  c(1, 0, 0, 1, 0, 1),
  c(1, 0, 0, 1, 0, 0),
  c(0, 1, 1, 0, 1, 0),
  c(0, 0, 0, 1, 0, 1),
  c(0, 1, 0, 0, 1, 0)
)
L = ComputeLaplacianMatrix(G)
E = eigen(L)
second_eigen_value = (E$value)[max(dim(L))-1]
second_eigen_vector = (E$vectors)[,max(dim(L))-1]
```

```
5.000000e-01  1.165734e-15  5.000000e-01  4.718448e-16 -5.000000e-01 -
5.000000e-01
```

Example #2


```
G = rbind(
  c(0, 0, 1, 0, 0, 1, 0, 0),
  c(0, 0, 0, 0, 1, 0, 0, 1),
  c(1, 0, 0, 1, 0, 1, 0, 0),
  c(0, 0, 1, 0, 1, 0, 1, 0),
  c(0, 1, 0, 1, 0, 0, 0, 1),
  c(1, 0, 1, 0, 0, 0, 1, 0),
  c(0, 0, 0, 1, 0, 1, 0, 1),
  c(0, 1, 0, 0, 1, 0, 1, 0)
)

L = ComputeLaplacianMatrix(G)

E = eigen(L)

second_eigen_value = (E$value)[max(dim(L))-1]
second_eigen_vector = (E$vectors)[,max(dim(L))-1]
```

```
5.000000e-01 -5.000000e-01  3.535534e-01 -3.885781e-16 -3.535534e-01
3.535534e-01 -4.996004e-16  -3.535534e-01
```

How Do We Partition a Graph into k Clusters?

- Two basic approaches:
- Recursive bi-partitioning [Hagen et al., '92]
  - Recursively apply bi-partitioning algorithm in a hierarchical divisive manner
  - Disadvantages: inefficient, unstable
- Cluster multiple eigenvectors [Shi-Malik, '00]
  - Build a reduced space from multiple eigenvectors
  - Commonly used in recent papers
  - A preferable approach…

Why Use Multiple Eigenvectors?

- Approximates the optimal cut [Shi-Malik, '00]
  - Can be used to approximate optimal k-way normalized cut
- Emphasizes cohesive clusters
  - Increases the unevenness in the distribution of the data
  - Associations between similar points are amplified, associations between dissimilar points are attenuated
  - The data begins to "approximate a clustering"
- Well-separated space
  - Transforms data to a new "embedded space", consisting of k orthogonal basis vectors
- Multiple eigenvectors prevent instability due to information loss

Run the Python notebooks
for this lecture