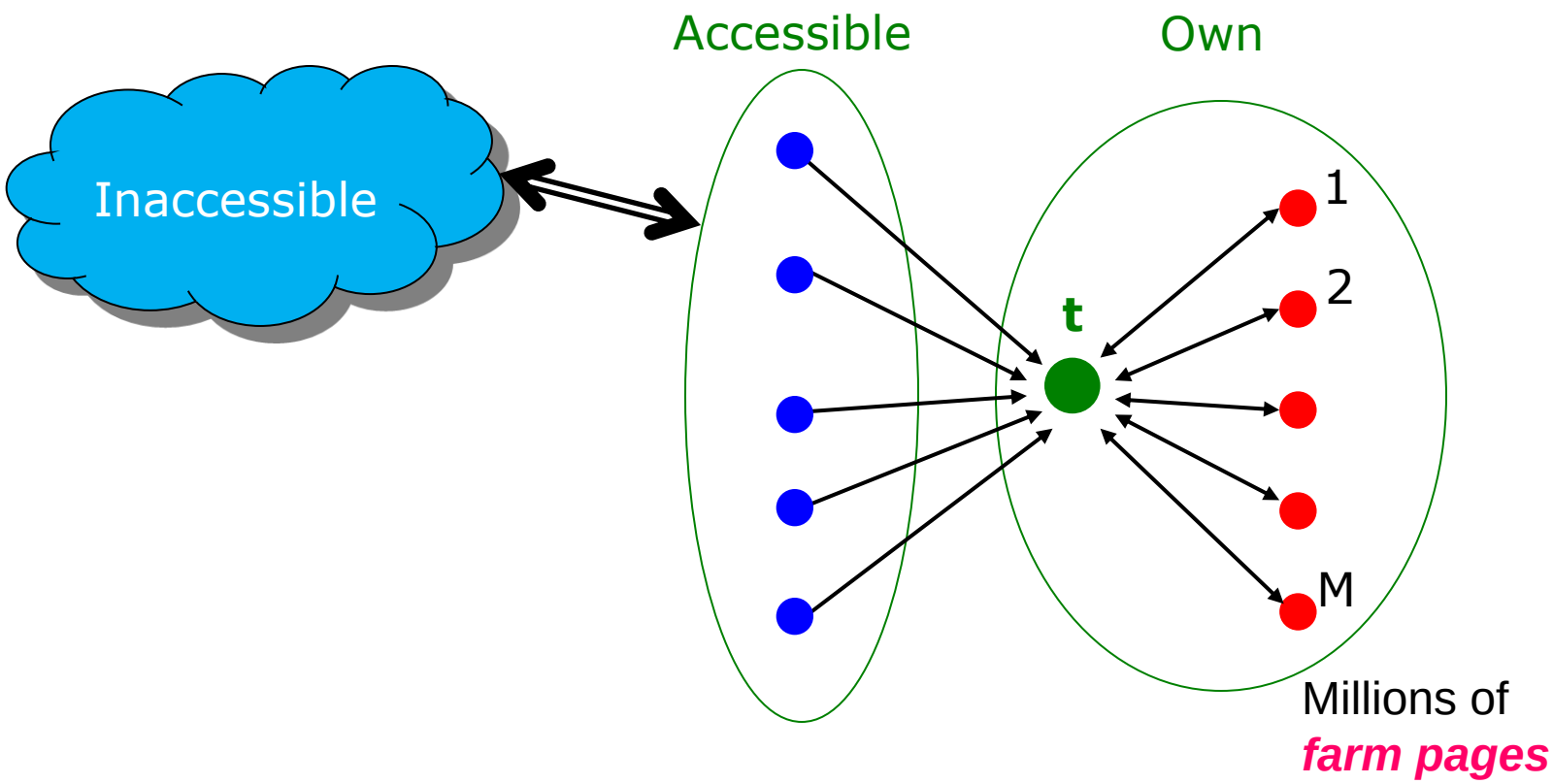
Link Analysis: TrustRank and Community Detection

CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
http://cs246.stanford.edu


SPAM FARMING

Link Farms



One of the most common and effective organizations for a link farm

TrustRank: Idea

- **Basic principle: Approximate isolation**
  - It is rare for a "good" page to point to a "bad" (spam) page
- Sample a set of **seed pages** from the web
- Have an **oracle** (**human**) to identify the good pages and the spam pages in the seed set
  - **Expensive task**, so we must make seed set as small as possible

Trust Propagation

- Call the subset of seed pages that are identified as **good** the **trusted pages**
- Perform a topic-sensitive PageRank with **teleport set = trusted pages**
  - **Propagate trust through links:**
    - Each page gets a trust value between **0** and **1**
- **Use a threshold value and mark all pages below the trust threshold as spam**

Simple Model

- Set trust of each trusted page to **1**
- Suppose trust of page ***p*** is t_p
  - Set of out-links o_p
- For each $q \in o_p$, ***p*** confers the trust:
  - $\beta t_p / |o_p|$ for $0 < \beta < 1$
- **Trust is additive**
  - Trust of ***p*** is the sum of the trust conferred on ***p*** by all its in-linked pages
- **Note similarity to Topic-Specific PageRank**
  - Within a scaling factor, **TrustRank = PageRank** with trusted pages as teleport set

Why is it a good idea?

- **Trust attenuation:**
  - The degree of trust conferred by a trusted page decreases with the distance in the graph
- **Trust splitting:**
  - The larger the number of out-links from a page, the less scrutiny the page author gives each out-link
  - Trust is **split** across out-links

Picking the Seed Set

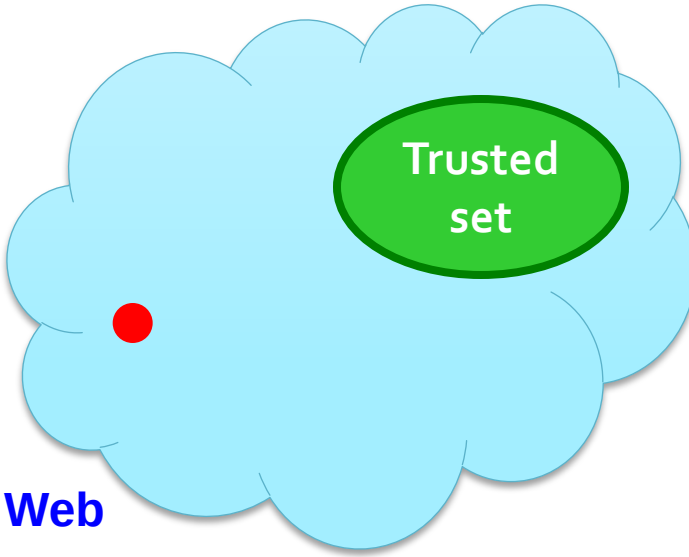
- **Two conflicting considerations:**
  - Human has to inspect each seed page, so seed set must be as small as possible
  - Must ensure every **good page** gets adequate trust rank, so need make all good pages reachable from seed set by short paths

Approaches to Picking Seed Set

- Suppose we want to pick a seed set of ***k*** pages
- **How to do that?**
- **(1) PageRank:**
  - Pick the top ***k*** pages by PageRank
  - The idea/hope is that you can't get a bad page's rank really really high
- **(2) Use trusted domains** whose membership is controlled, like .edu, .mil, .gov

Spam Mass

- In the **TrustRank** model, we start with good pages and propagate trust
- **Complementary view:**
  What fraction of a page's PageRank comes from **spam** pages?
- In practice, we don't know all the spam pages, so we need to estimate



Spam Mass Estimation

- r_p = PageRank of page ***p***
- r_p^+ = PageRank of ***p*** with teleport into **trusted** pages only

- **Then:** What fraction of a page's PageRank comes from **spam** pages?

$$r_p^- = r_p - r_p^+$$

- **Spam mass of *p* =** $\frac{r_p^-}{r_p}$

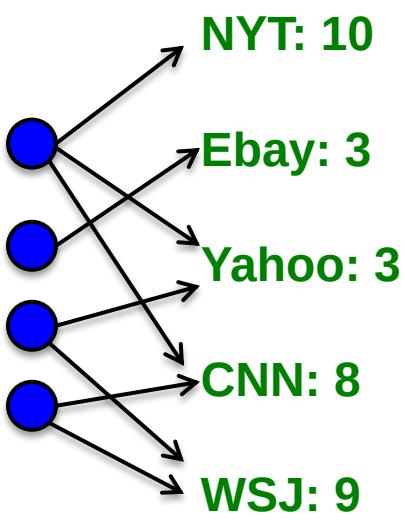
HITS: Hubs and Authorities

Hubs and Authorities

- **HITS (Hypertext-Induced Topic Selection)**
  - **Is a measure of importance of pages or documents, similar to PageRank**
  - Proposed at around same time as PageRank ('98)
- **Goal**: Say we want to find good newspapers
  - Don't just find newspapers. Find "experts" – people who link in a coordinated way to good newspapers
- **Idea: Links as votes**
  - **Page is more important if it has more links**
    - In-coming links? Out-going links?

Finding newspapers

- **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

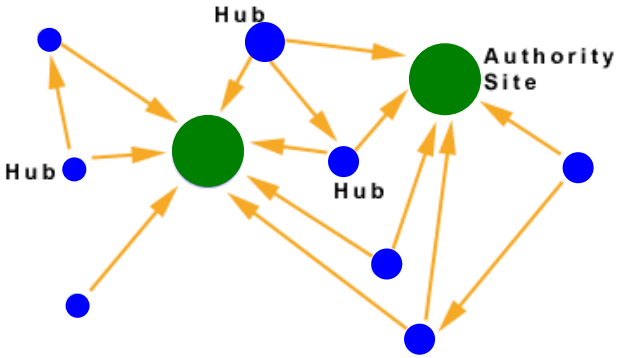


- **Principle of repeated improvement**

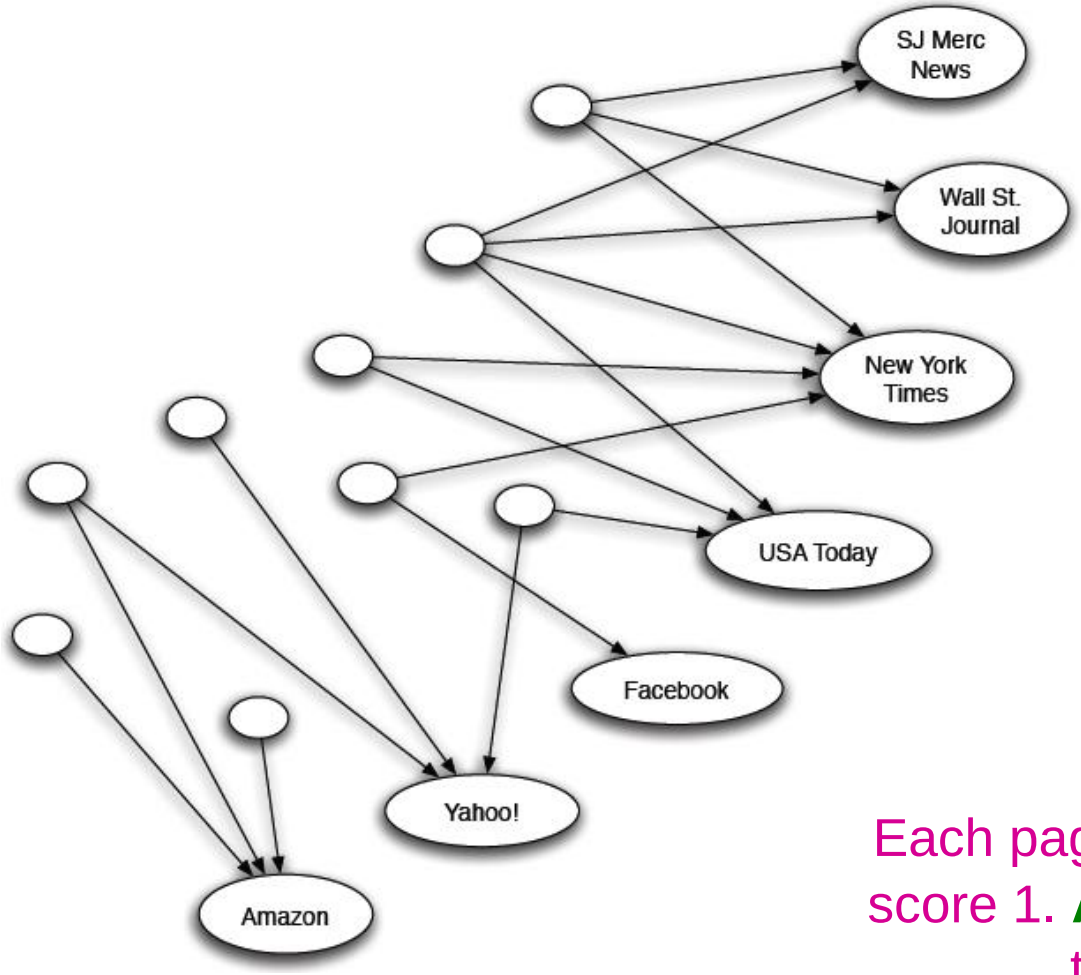
Hubs and Authorities

Interesting pages fall into two classes:

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



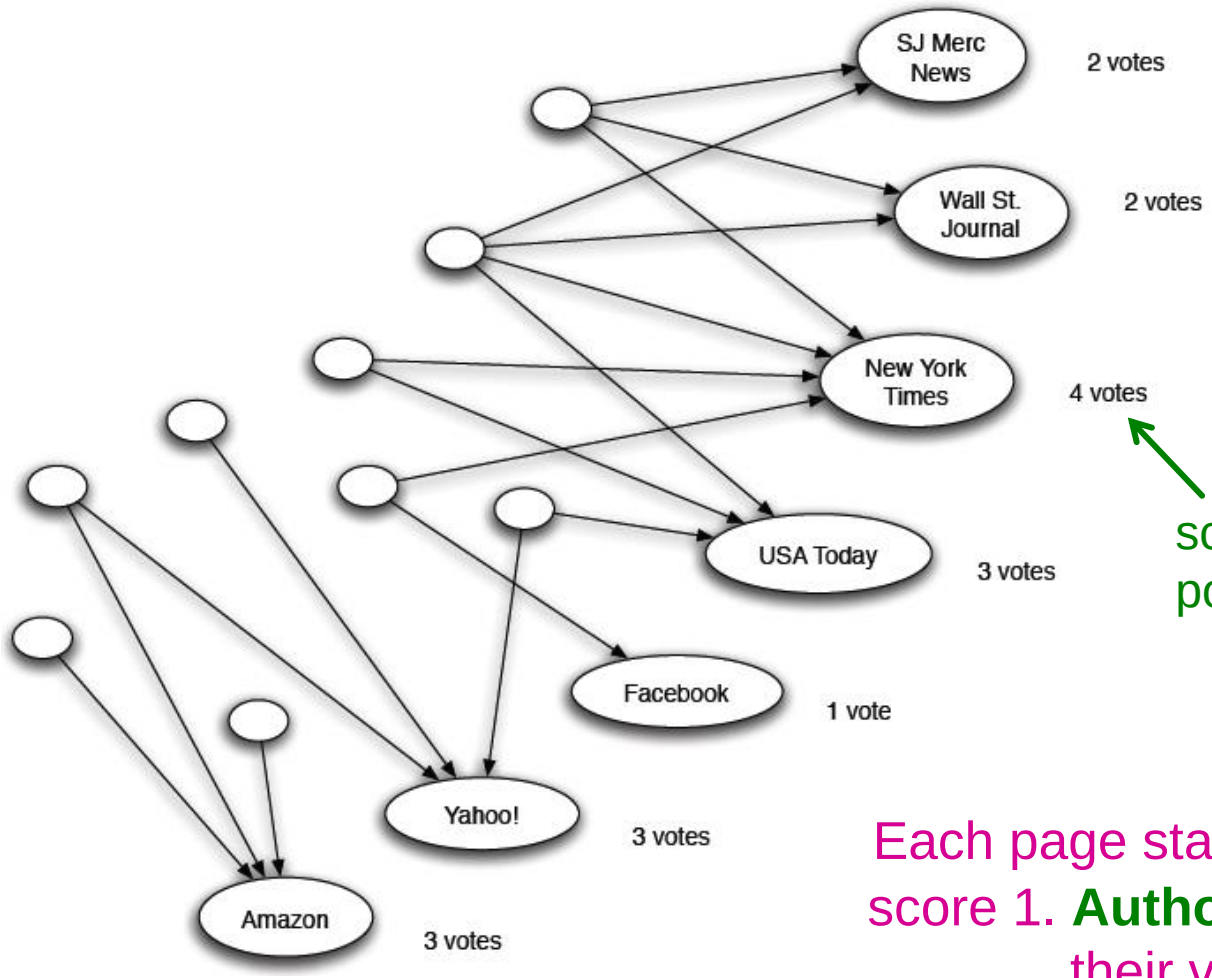
Counting in-links: Authority



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



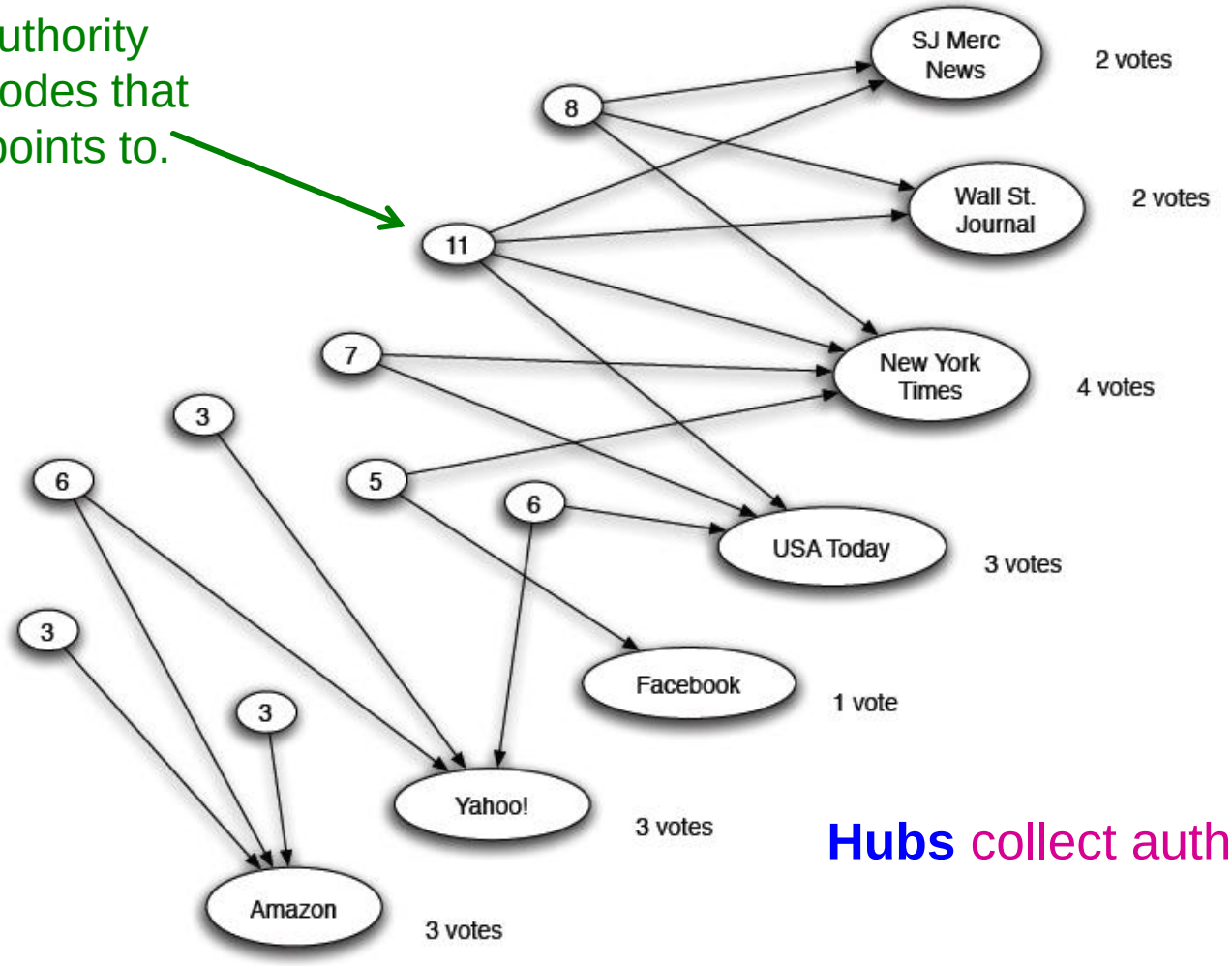
Sum of **hub** scores of nodes pointing to NYT.

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

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

Expert Quality: Hub

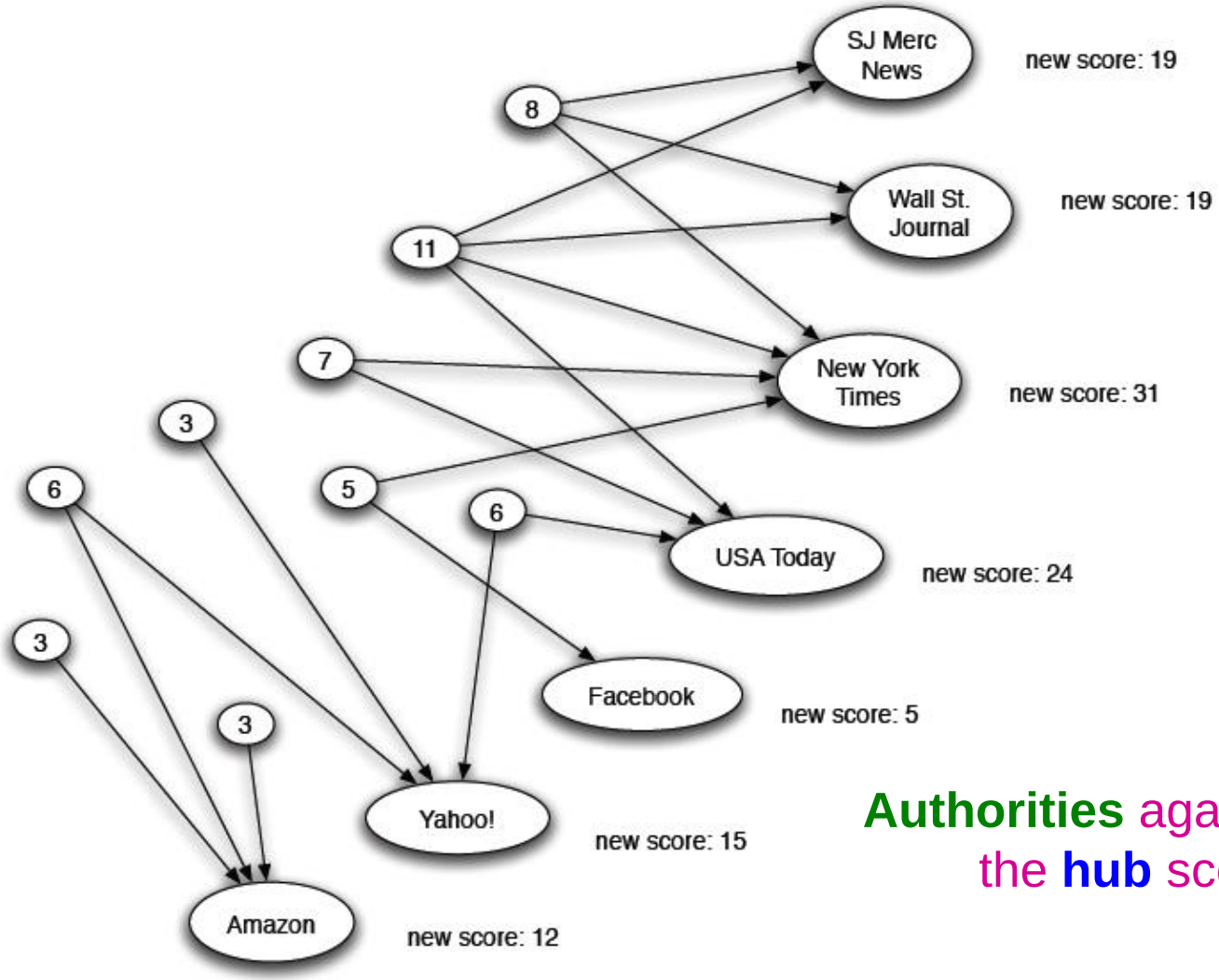
Sum of authority scores of nodes that the node points to.



Hubs collect authority scores

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

Reweighting



Authorities again collect the **hub** scores

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

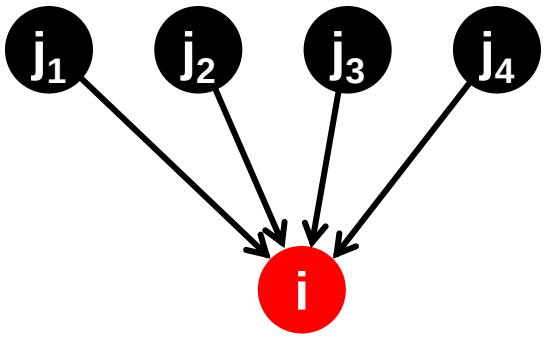
Mutually Recursive Definition

- **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 $\boldsymbol{h}$ and $\boldsymbol{a}$

[Kleinberg '98]

Hubs and Authorities

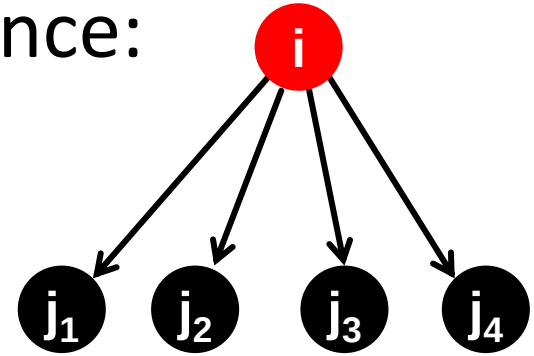
- **Each page i has 2 scores:**
  - Authority score: $\boldsymbol{a_i}$
  - Hub score: $\boldsymbol{h_i}$



$$a_i = \sum_{j \to i} h_j$$

HITS algorithm:

- Initialize: $\boldsymbol{a_i} = 1/\sqrt{n}$, $\boldsymbol{h_i} = 1/\sqrt{n}$
- Then keep iterating until convergence:
  - $\forall i$: **Authority**: $\boldsymbol{a_i} = \sum_{j \to i} \boldsymbol{h_j}$
  - $\forall i$: **Hub**: $\boldsymbol{h_i} = \sum_{i \to j} \boldsymbol{a_j}$
  - $\forall i$: **Normalize a, h such that:** $\sum_i \boldsymbol{a_i^2} = 1, \sum_i \boldsymbol{h_i^2} = 1$



$$h_i = \sum_{i \to j} a_j$$

[Kleinberg '98]

Hubs and Authorities

- **HITS converges to a single stable point**
- **Notation:**
  - Vector $\boldsymbol{a} = (a_1 \ldots, a_n)$, $\boldsymbol{h} = (h_1 \ldots, h_n)$
  - Adjacency matrix $\boldsymbol{A}$ (n x n): $\boldsymbol{A_{ij}} = 1$ if $i \rightarrow j$
- **Then** $\boldsymbol{h_i} = \sum_{i \rightarrow j} \boldsymbol{a_j}$

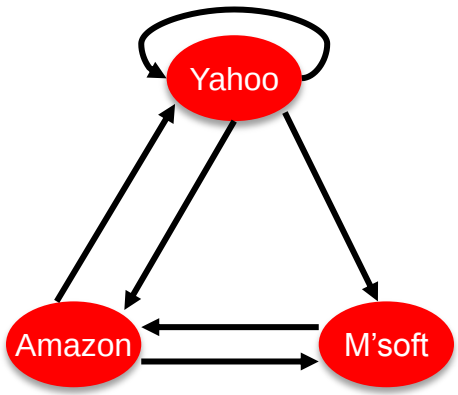
  can be rewritten as $\boldsymbol{h_i} = \sum_j \boldsymbol{A_{ij}} \cdot \boldsymbol{a_j}$

  So: $\boldsymbol{h} = \boldsymbol{A} \cdot \boldsymbol{a}$
- **Similarly,** $\boldsymbol{a_i} = \sum_{j \rightarrow i} \boldsymbol{h_j}$

  can be rewritten as $\boldsymbol{a_i} = \sum_j \boldsymbol{A_{ji}} \cdot \boldsymbol{h_i} = \boldsymbol{A^T} \cdot \boldsymbol{h}$

Example

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad A^T = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$



a(yahoo)	=	.58	.80	.80	.79	. . .	.788
a(amazon)	=	.58	.53	.53	.57	. . .	.577
a(m'soft)	=	.58	.27	.27	.23	. . .	.211
h(yahoo)	=	.58	.58	.62	.62	. . .	.628
h(amazon)	=	.58	.58	.49	.49	. . .	.459
h(m'soft)	=	.58	.58	.62	.62	. . .	.628

Hubs and Authorities

- **HITS algorithm in vector notation:**
  - Set: $\boldsymbol{a_i} = \boldsymbol{h_i} = \frac{\mathbf{1}}{\sqrt{\boldsymbol{n}}}$

    Repeat until convergence:
  - $\boldsymbol{h} = \boldsymbol{A} \cdot \boldsymbol{a}$
  - $\boldsymbol{a} = \boldsymbol{A}^{\boldsymbol{T}} \cdot \boldsymbol{h}$
  - Normalize $\boldsymbol{a}$ and $\boldsymbol{h}$
- **Then:** $\boldsymbol{a} = \underbrace{\boldsymbol{A}^{\boldsymbol{T}} \cdot (\underbrace{\boldsymbol{A} \cdot \boldsymbol{a}}_{\text{new } \boldsymbol{h}})}_{\text{new } \boldsymbol{a}}$
- **Thus, in $2k$ steps:**

$$a = (A^T \cdot A)^k \cdot a$$
$$h = (A \cdot A^T)^k \cdot h$$

Convergence criterion:

$$\sum_i \left(h_i^{(t)} - h_i^{(t-1)}\right)^2 < \varepsilon$$
$$\sum_i \left(a_i^{(t)} - a_i^{(t-1)}\right)^2 < \varepsilon$$

a is updated (in 2 steps):

$$a = A^T(A\,a) = (A^T A)\,a$$

h is updated (in 2 steps):

$$h = A\,(A^T h) = (A\,A^T)\,h$$

Repeated matrix powering

Existence and Uniqueness

- $\boldsymbol{h} = \lambda A \boldsymbol{a}$
- $\boldsymbol{a} = \mu A^T \boldsymbol{h}$
- $\boldsymbol{h} = \lambda \mu A A^T \boldsymbol{h}$
- $\boldsymbol{a} = \lambda \mu A^T A \boldsymbol{a}$

$\lambda = 1/\sum h_i$
$\mu = 1/\sum a_i$

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

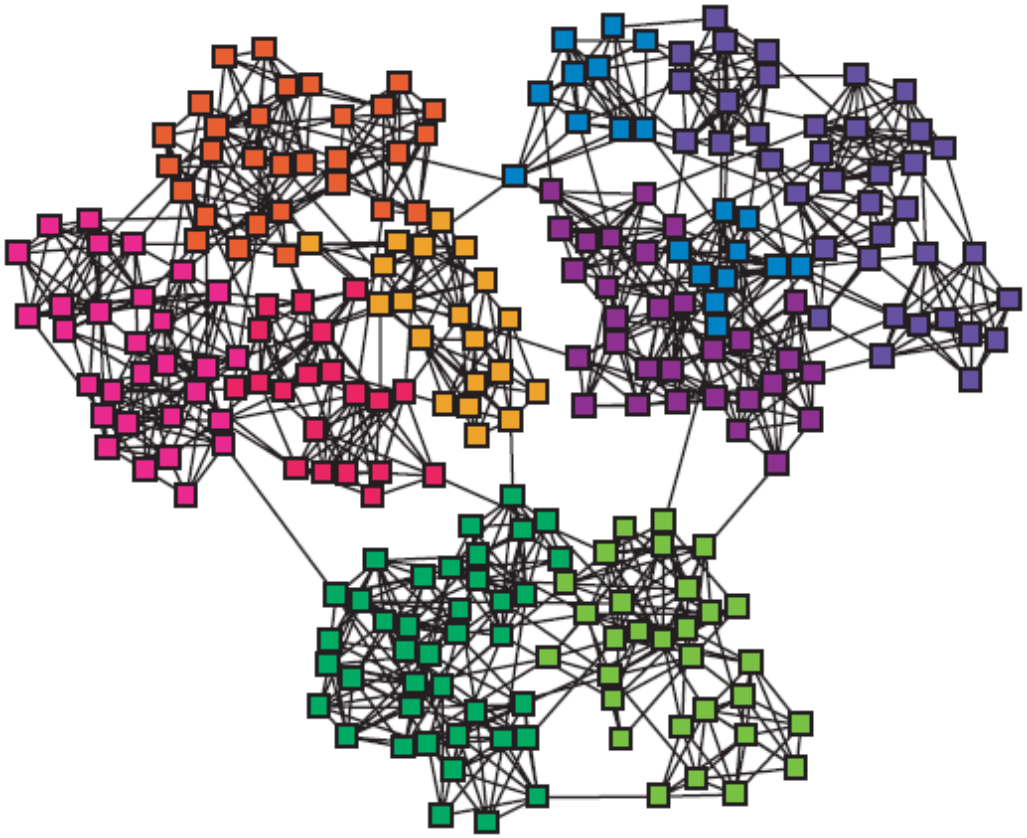
PageRank and HITS

- **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 post-1998 were very different**

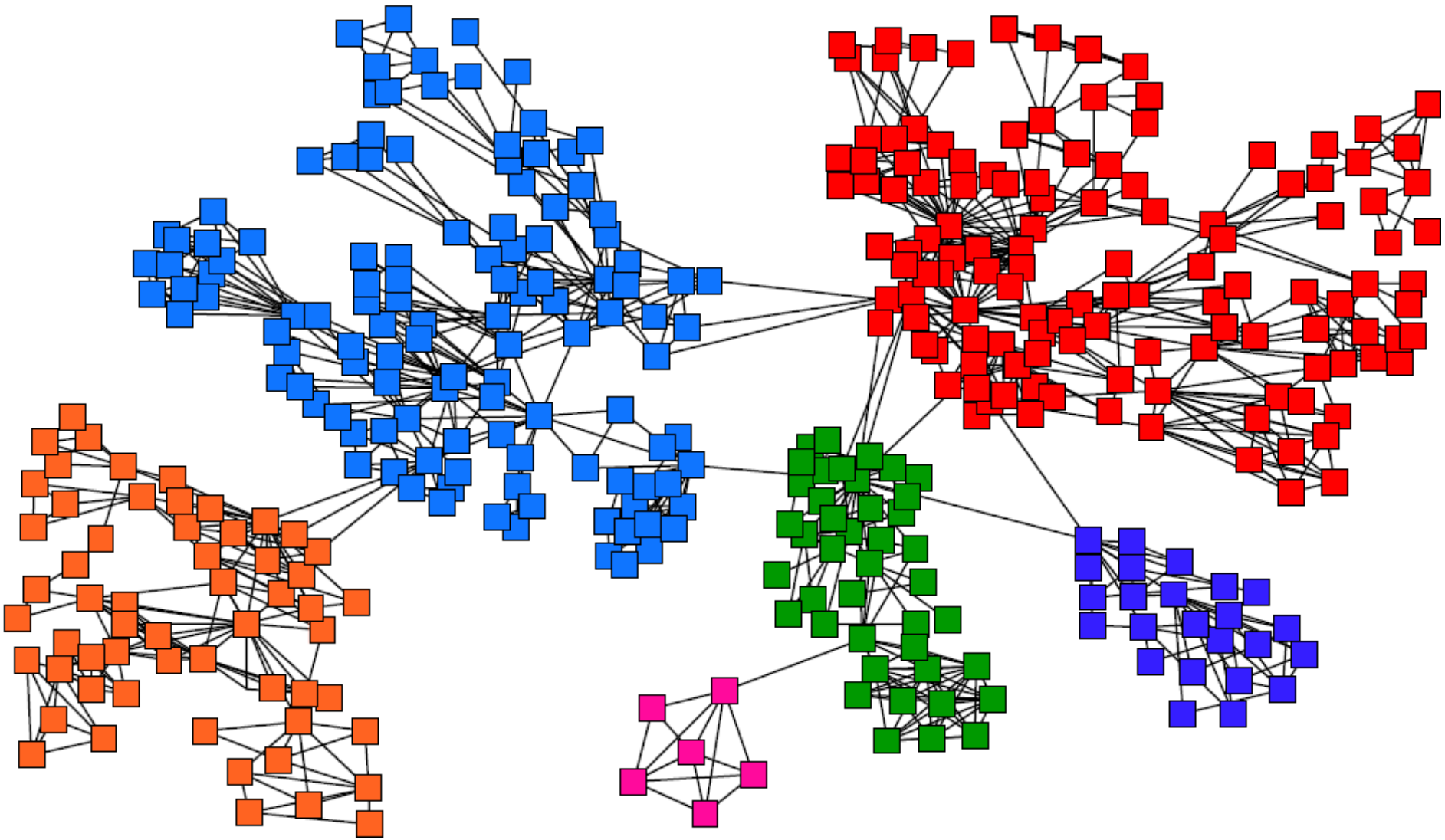
Finding Communities in Networks using PageRank

Networks & Communities

- **We often think of networks "looking" like this:**

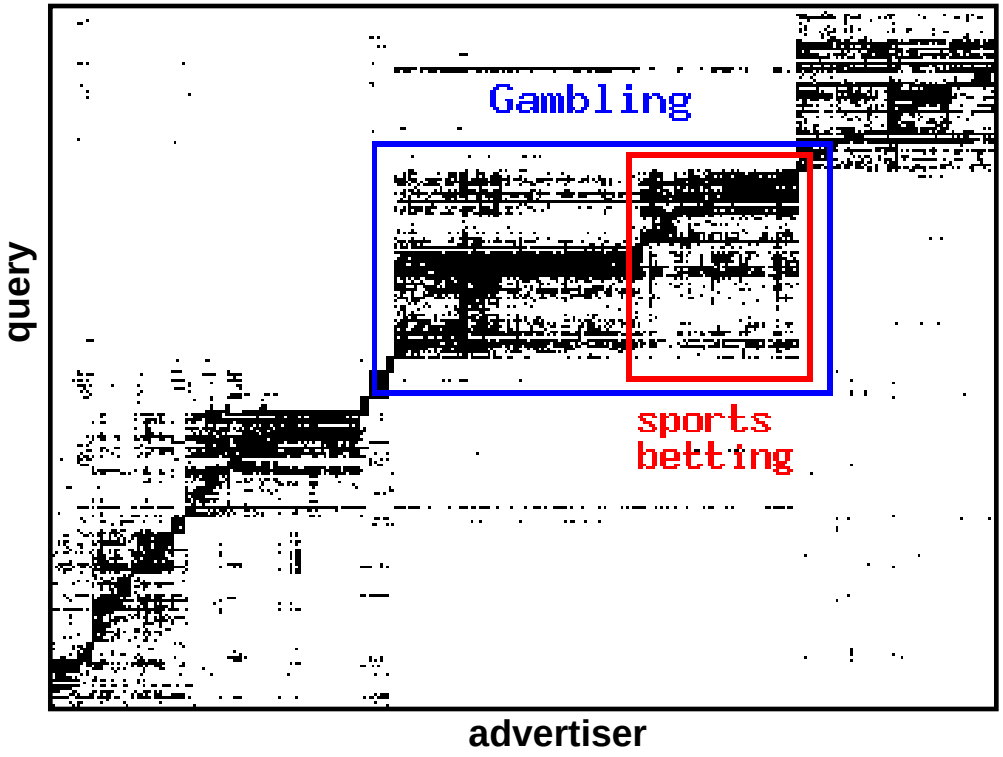


Goal: Find Densely Linked Clusters



Micro-Markets in Sponsored Search

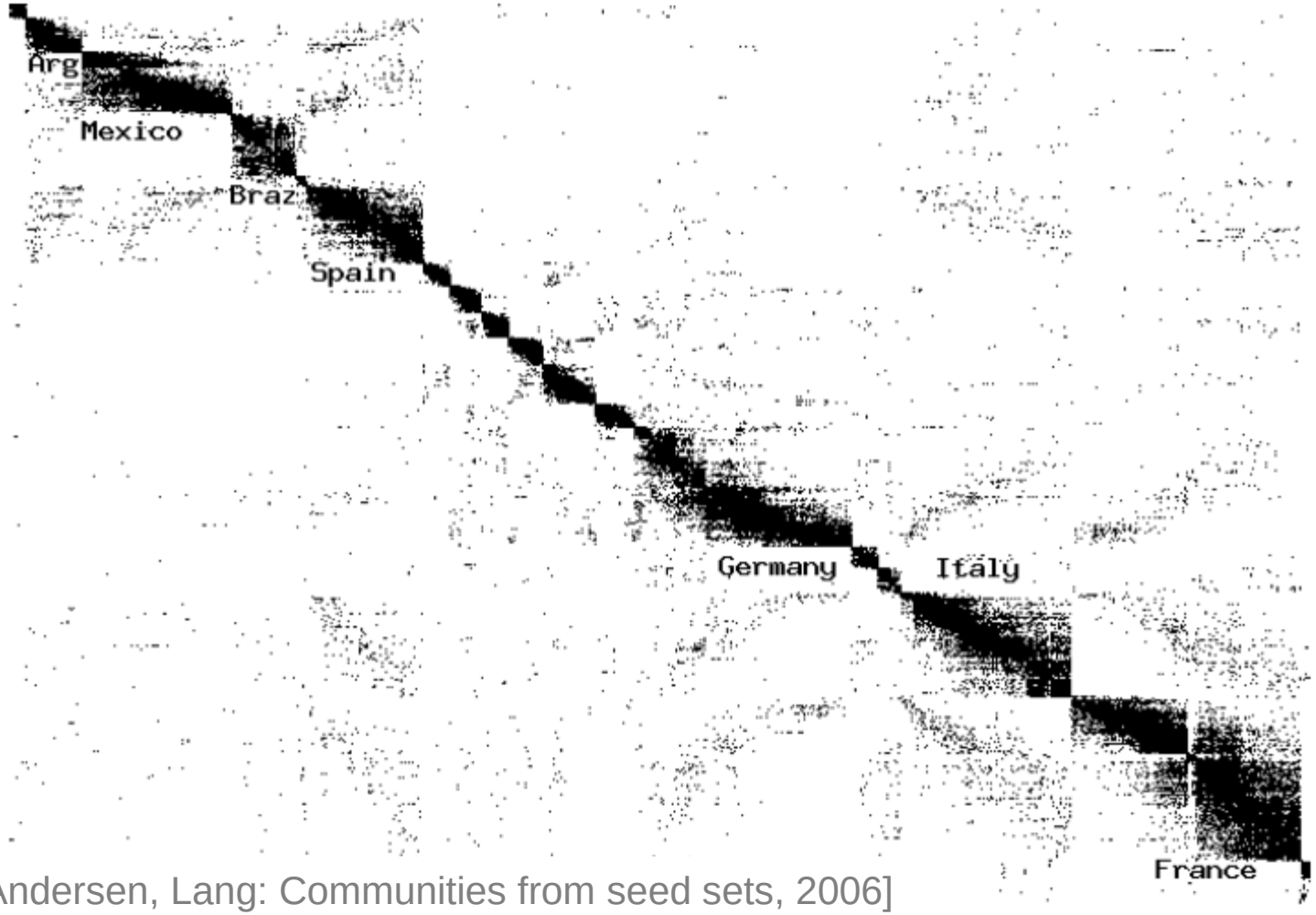
- **Find micro-markets by partitioning the query-to-advertiser graph:**



[Andersen, Lang: Communities from seed sets, 2006]

Movies and Actors

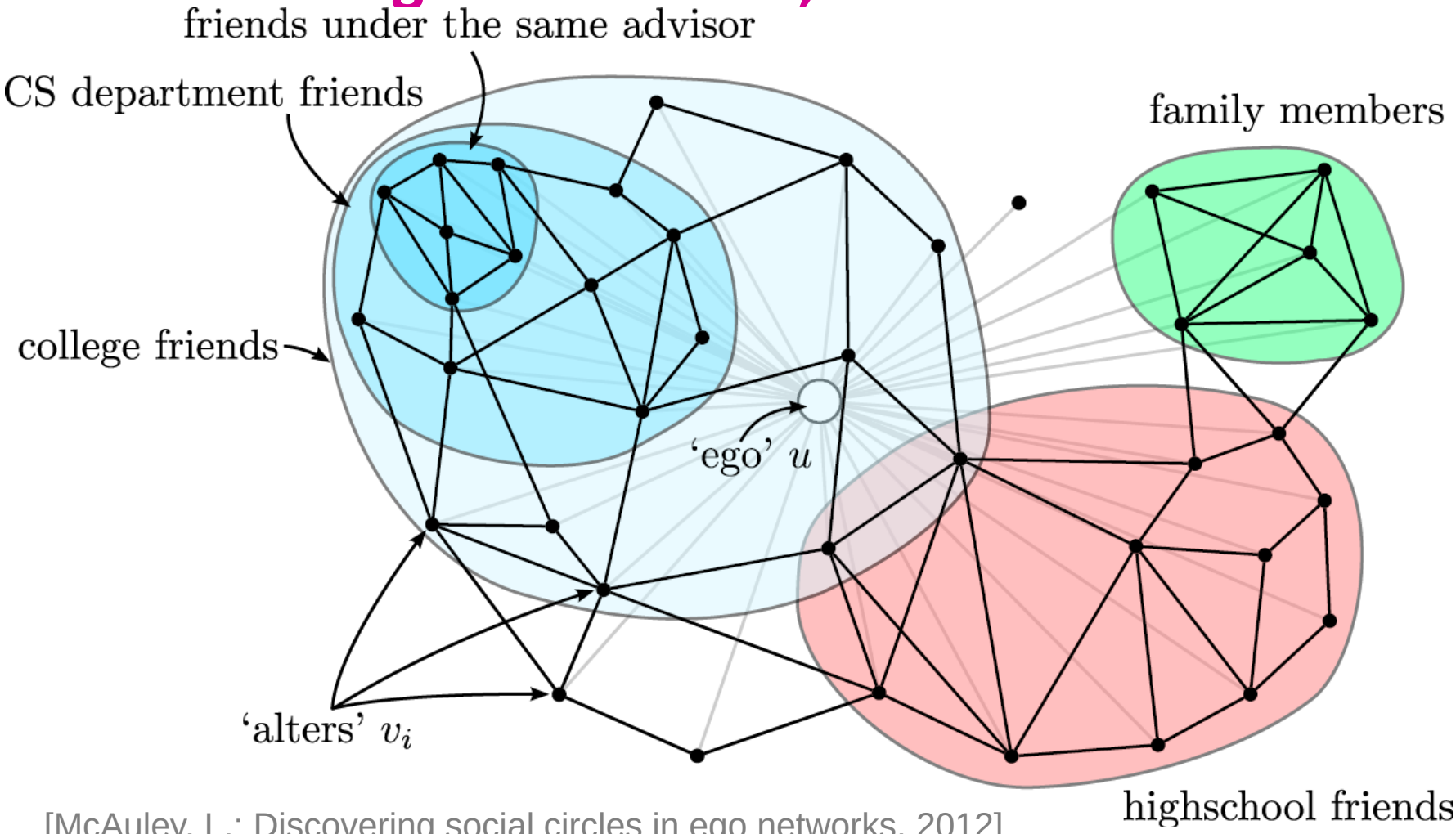
- **Clusters in Movies-to-Actors graph:**



[Andersen, Lang: Communities from seed sets, 2006]

Twitter & Facebook

- **Discovering social circles, circles of trust:**



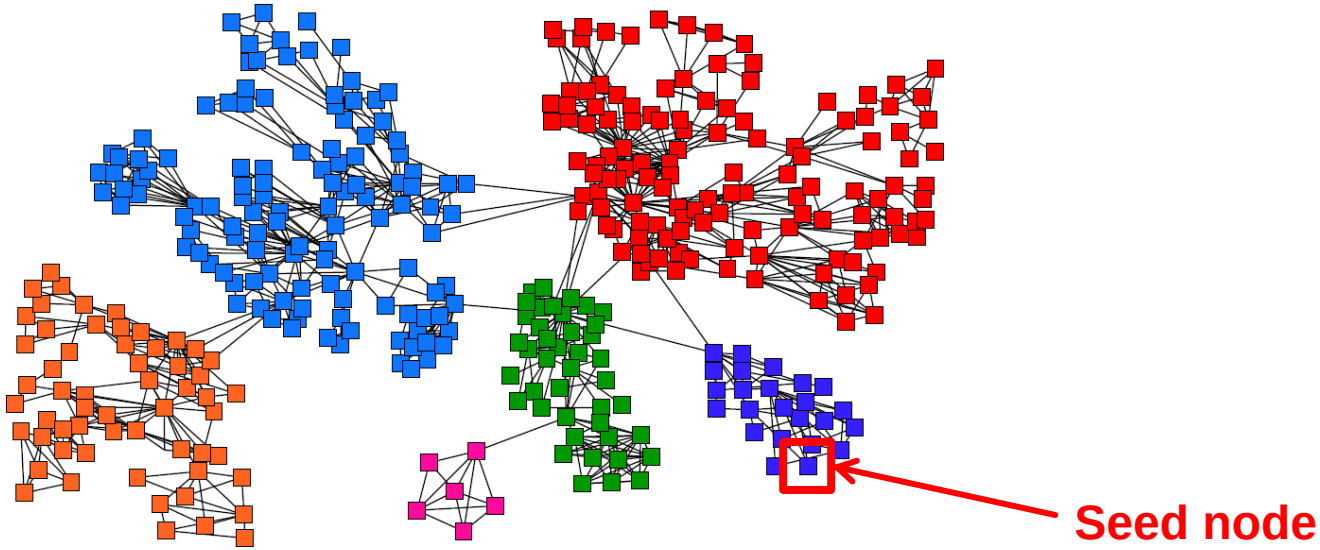
[McAuley, L.: Discovering social circles in ego networks, 2012]

The Setting

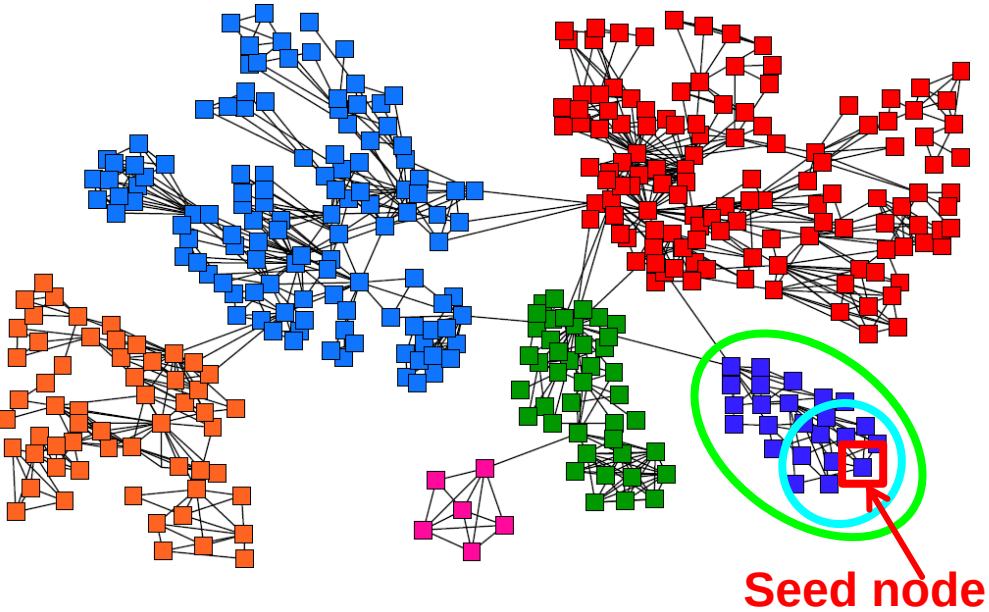
- **Graph is large**
  - **Assume the graph fits in main memory**
    - For example, to work with a 200M node and 2B edge graph one needs approx. 16GB RAM
  - But the graph is too big for running anything more than linear time algorithms
- **We will cover a PageRank based algorithm for finding dense clusters**
  - **The runtime of the algorithm will be proportional to the cluster size (not the graph size!)**

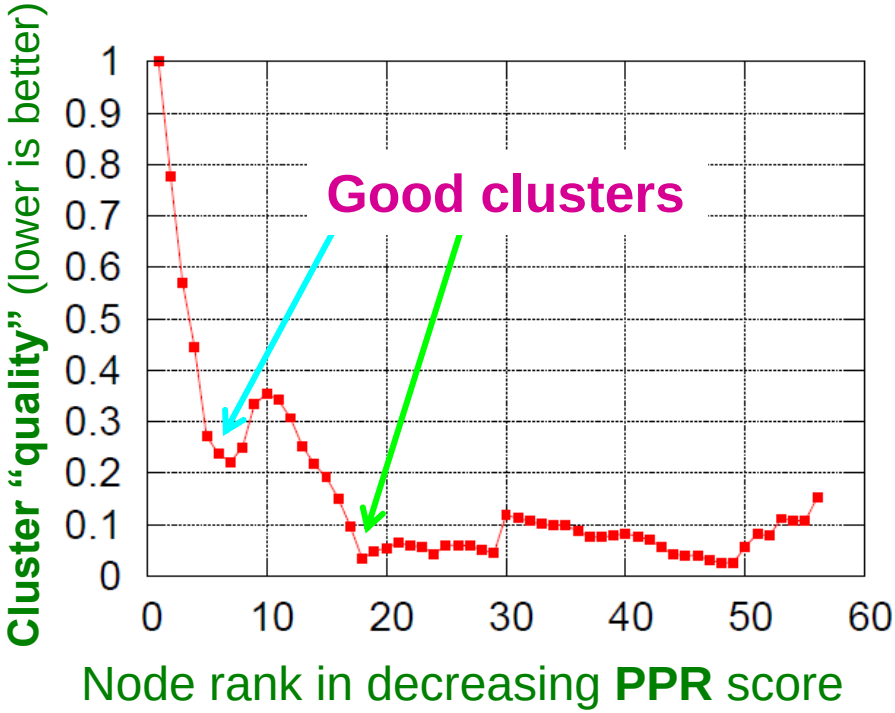
Idea: Seed nodes

- **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

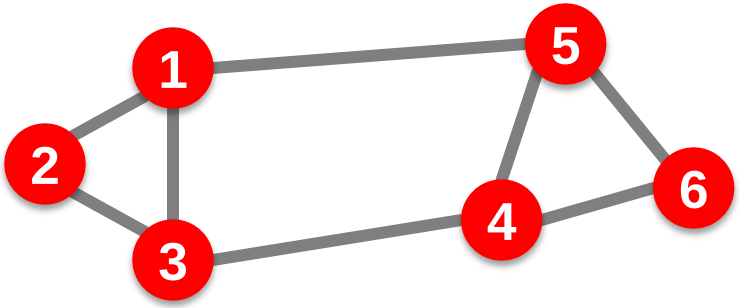




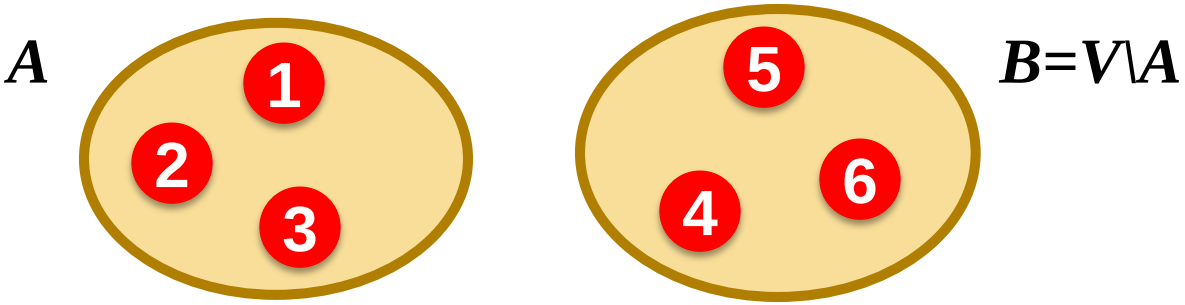
- **Algorithm outline:**
  - Pick a seed node ***S*** of interest
  - Run **PPR** with teleport set = {***S***}
  - Sort the nodes by the decreasing **PPR** score
  - **Sweep** over the nodes and find **good clusters**

What makes a good cluster?

- **Undirected graph $G(V, E)$:**



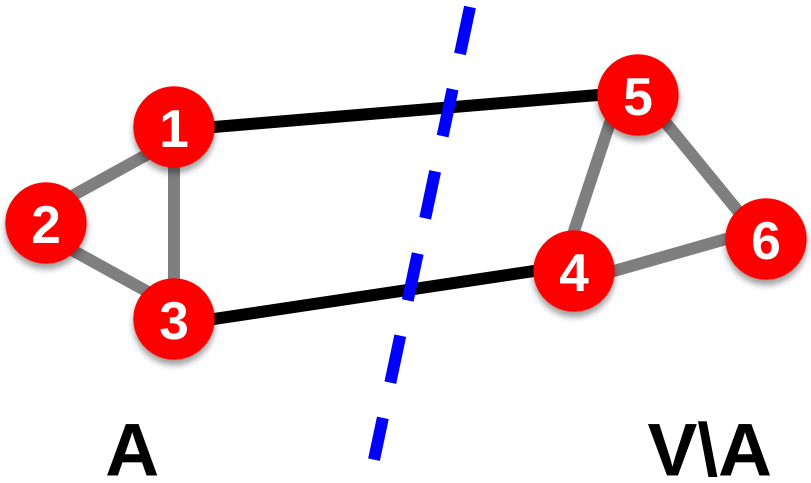
- **Partitioning task:**
  - Divide vertices into 2 disjoint groups $A, B = V \backslash A$



- **Question:**
  - **How can we define a "good" cluster in G?**

What makes a good cluster?

- **What makes a good cluster?**
  - **Maximize the number of within-cluster connections**
  - **Minimize the number of between-cluster connections**

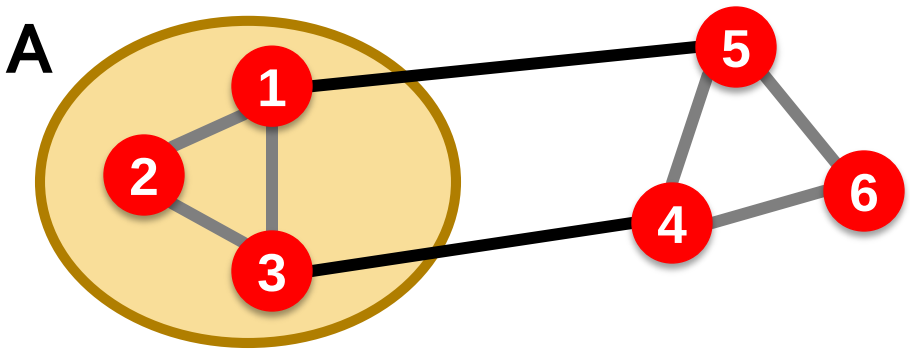


Graph Cuts

- **Express cluster quality as a function of the "edge cut" of the cluster**
- **Cut:** Set of edges with only one node in the cluster:

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

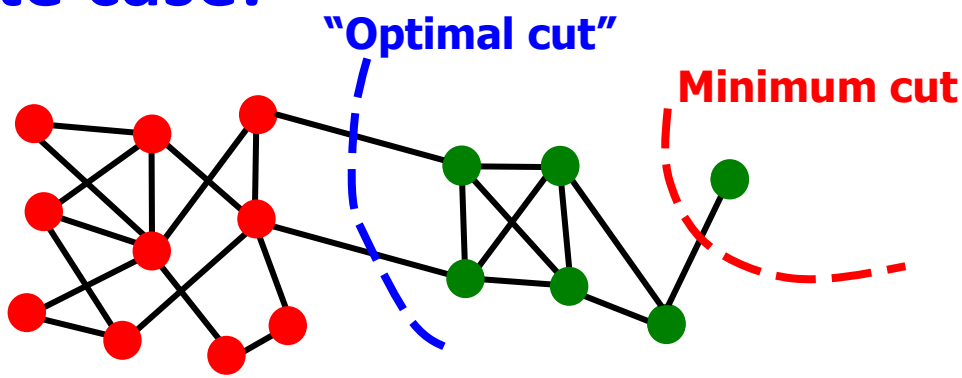
Note: This works for weighed and unweighted (set all $\mathbf{w_{ij}=1}$) graphs



$$cut(A) = 2$$

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**

[Shi-Malik]

Graph Partitioning Criteria

- **Criterion: Conductance:**
  Connectivity of the group to the rest of the network relative to the density of the group

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

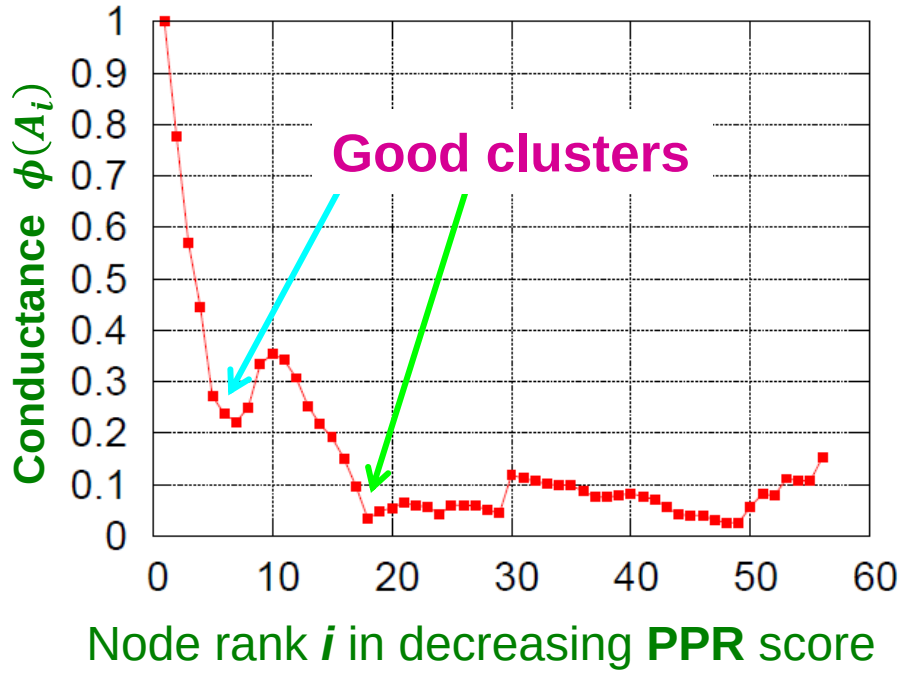
$\boldsymbol{vol(A)}$: total weight of the edges with at least one endpoint in $\boldsymbol{A}$: $\mathbf{vol}(\boldsymbol{A}) = \sum_{\boldsymbol{i} \in \boldsymbol{A}} \boldsymbol{d_i}$

m… number of edges of the graph
d_i… degree of node ***i***

- **Why use this criterion?**
  - Produces more balanced partitions

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



- **Sweep:**
  - Sort nodes in decreasing PPR score $r_1 > r_2 > \cdots > r_n$
  - For each $\boldsymbol{i}$ compute $\boldsymbol{\phi}(\boldsymbol{A_i} = \{\boldsymbol{r_1}, \ldots \boldsymbol{r_i}\})$
  - Local minima of $\boldsymbol{\phi}(\boldsymbol{A_i})$ correspond to good clusters

Computing PPR

- **How to compute Personalized PageRank without touching the whole graph?**
  - Power method won't work since each single iteration accesses each node of the graph:
    $\mathbf{r} = \boldsymbol{\beta}\mathbf{M} \cdot \mathbf{r} + (\mathbf{1} - \boldsymbol{\beta})\boldsymbol{a}^{\boldsymbol{T}}$
    - Where vector $\boldsymbol{a} = [\mathbf{0} \ \ldots \mathbf{0} \ \mathbf{1} \ \mathbf{0} \ \ldots \mathbf{0}]$ At index **S**
- **PageRank-Nibble** [Andersen,Chung, Lang, '07]
  - A fast method for computing approximate Personalized PageRank (**PPR**) with teleport set ={**S**}
  - **ApproxPageRank(S, β, ε)**
    - **S** … seed node
    - **β** … teleportation parameter
    - **ε** … approximation error parameter

Approximate PPR: Overview

- **Overview of the approximate PPR**
  - **Lazy random walk**, which is a variant of a random walk that stays put with probability 1/2 at each time step, and walks to a random neighbor the other half of the time:

$$r_u^{(t+1)} = \frac{1}{2} r_u^{(t)} + \frac{1}{2} \sum_{i \to u} \frac{1}{d_i} r_i^{(t)}$$

  - Keep track of residual **PPR** score q_u
  - If residual $\boldsymbol{q_u}$ of node **u** is too big $\frac{q_u}{d_u} \geq \varepsilon$ then spread the walk further, else don't touch the node

Approximate PPR

- **ApproxPageRank(S, β, ε):**

  Set $\boldsymbol{r} = \vec{0}$, $\boldsymbol{q} = [0\,..\,0\ 1\ 0\ \ldots\ 0]$

  At index **S**

  While $\max_{u \in V} \frac{q_u}{d_u} \geq \boldsymbol{\varepsilon}$:

  Choose any vertex $\boldsymbol{u}$ where $\frac{q_u}{d_u} \geq \varepsilon$

  Push(u, r, q):

  $$\boldsymbol{r}' = \boldsymbol{r},\ \boldsymbol{q}' = \boldsymbol{q}$$

  $$\boldsymbol{r}'_u = \boldsymbol{r}_u + (\mathbf{1} - \boldsymbol{\beta})\boldsymbol{q}_u$$

  $$\boldsymbol{q}'_u = \frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{\beta}\boldsymbol{q}_u$$

  For each $\boldsymbol{v}$ such that $(\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{E}$:

  $$\boldsymbol{q}'_v = \boldsymbol{q}_v + \frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{\beta}\boldsymbol{q}_u/\boldsymbol{d}_u$$

  Update $\boldsymbol{r} = \boldsymbol{r}'$, $\boldsymbol{q} = \boldsymbol{q}'$

  Return $\boldsymbol{r}$

r … **PPR** vector
$\mathbf{r_u}$ …PPR score of **u**
q …residual PPR vector
$\mathbf{q_u}$ … residual of node **u**
$\mathbf{d_u}$ … degree of **u**

Update **r** and **q**

Move $(1 - \beta)$ of the probability from $\mathbf{q_u}$ to $\mathbf{r_u}$

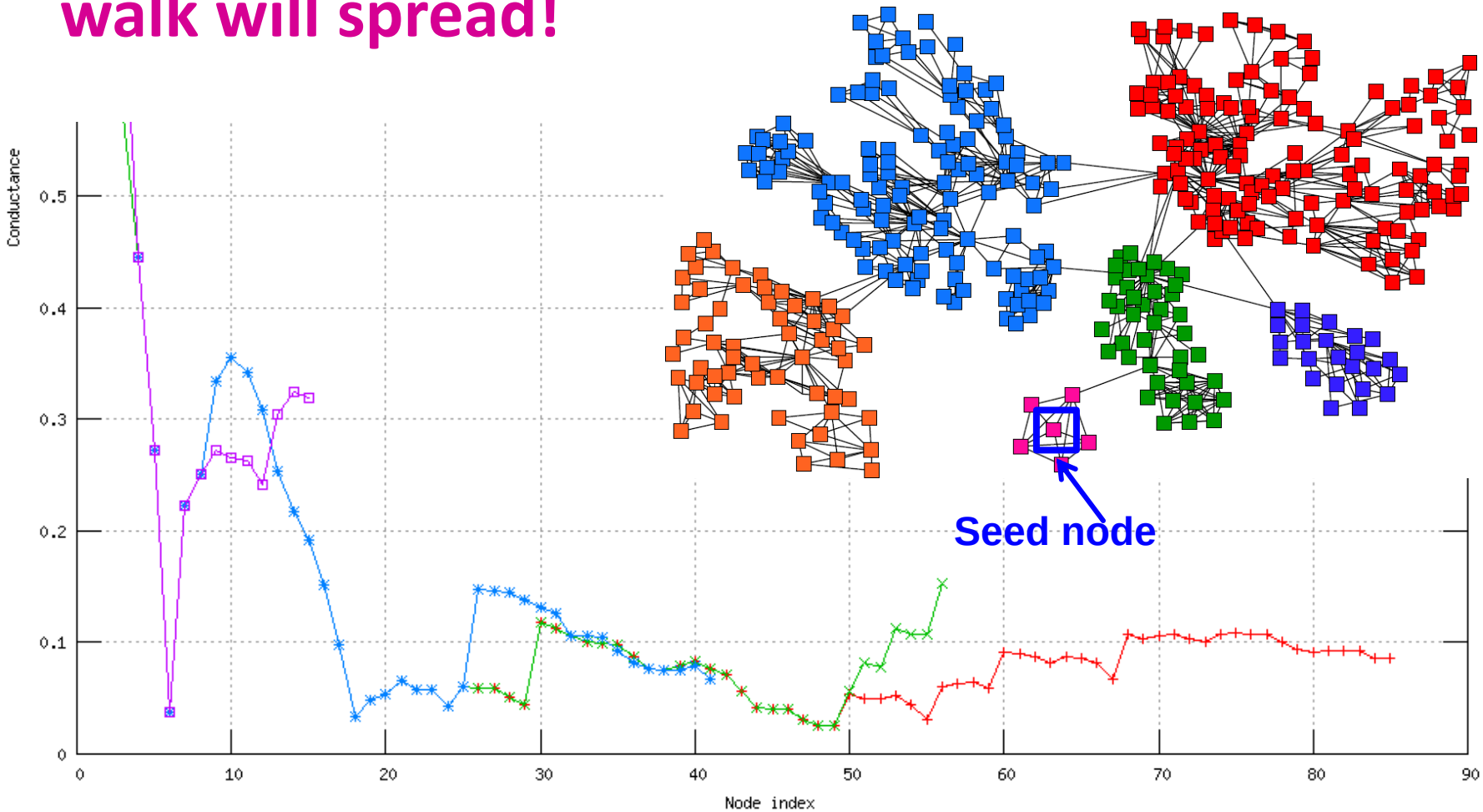
Spread remaining $\boldsymbol{\beta}$ fraction of $\mathbf{q_u}$ as if a single step of random walk were applied to **u**

Observations (1)

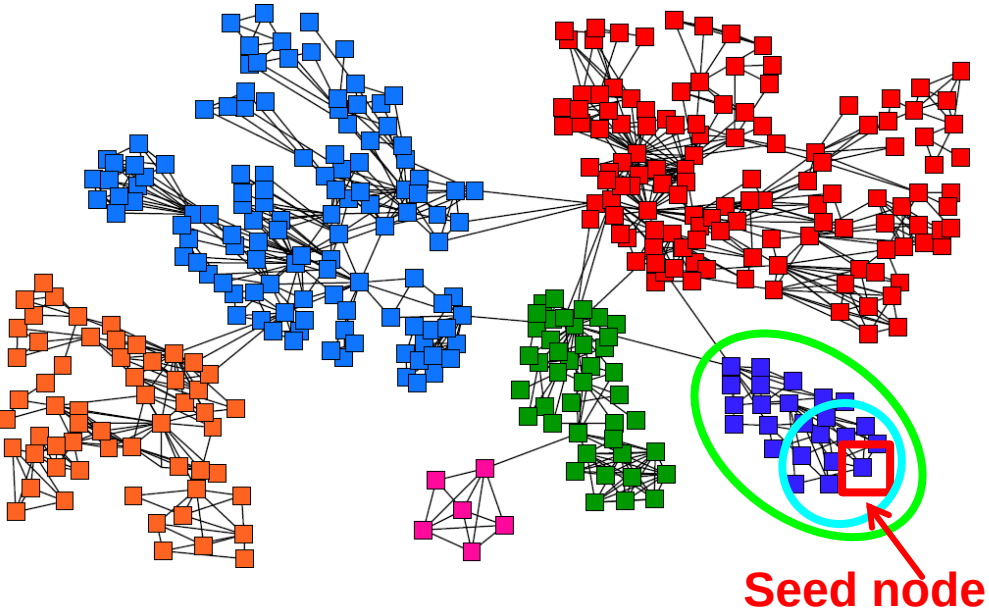
- **Runtime:**
  - PageRank-Nibble computes PPR in time $\left(\frac{\mathbf{1}}{\boldsymbol{\varepsilon}(\mathbf{1}-\boldsymbol{\beta})}\right)$
  - Power method would take time $\boldsymbol{O}\left(\frac{\mathbf{log}\,\boldsymbol{n}}{\boldsymbol{\varepsilon}(\mathbf{1}-\boldsymbol{\beta})}\right)$
- **Approximation guarantee:**
  - If there exists a cut of conductance ϕ then the method finds a cut of conductance $\Omega(\phi^2/\log m)$
  - Details in [Andersen, Chung, Lang. *Local graph partitioning using PageRank vectors*, 2007]

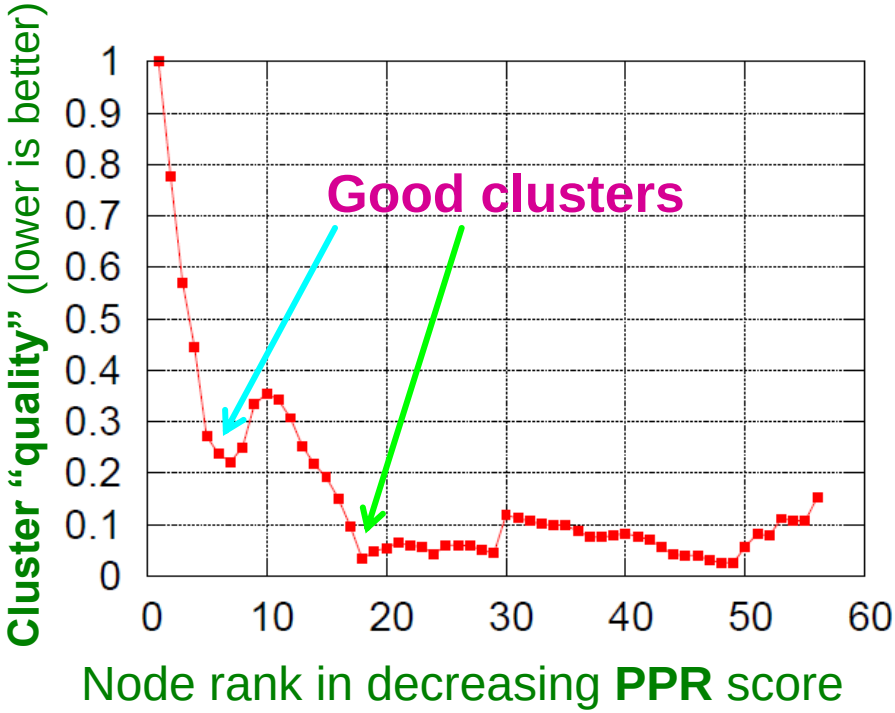
Observations (2)

- The smaller the ε the farther the random walk will spread!



Summary





- **Algorithm summary:**
  - 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