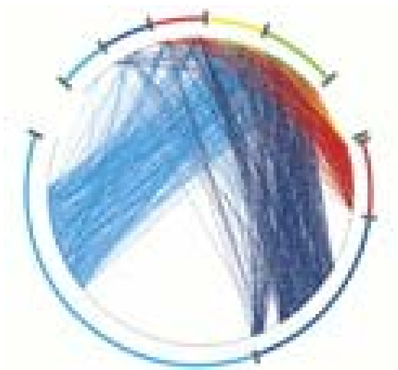
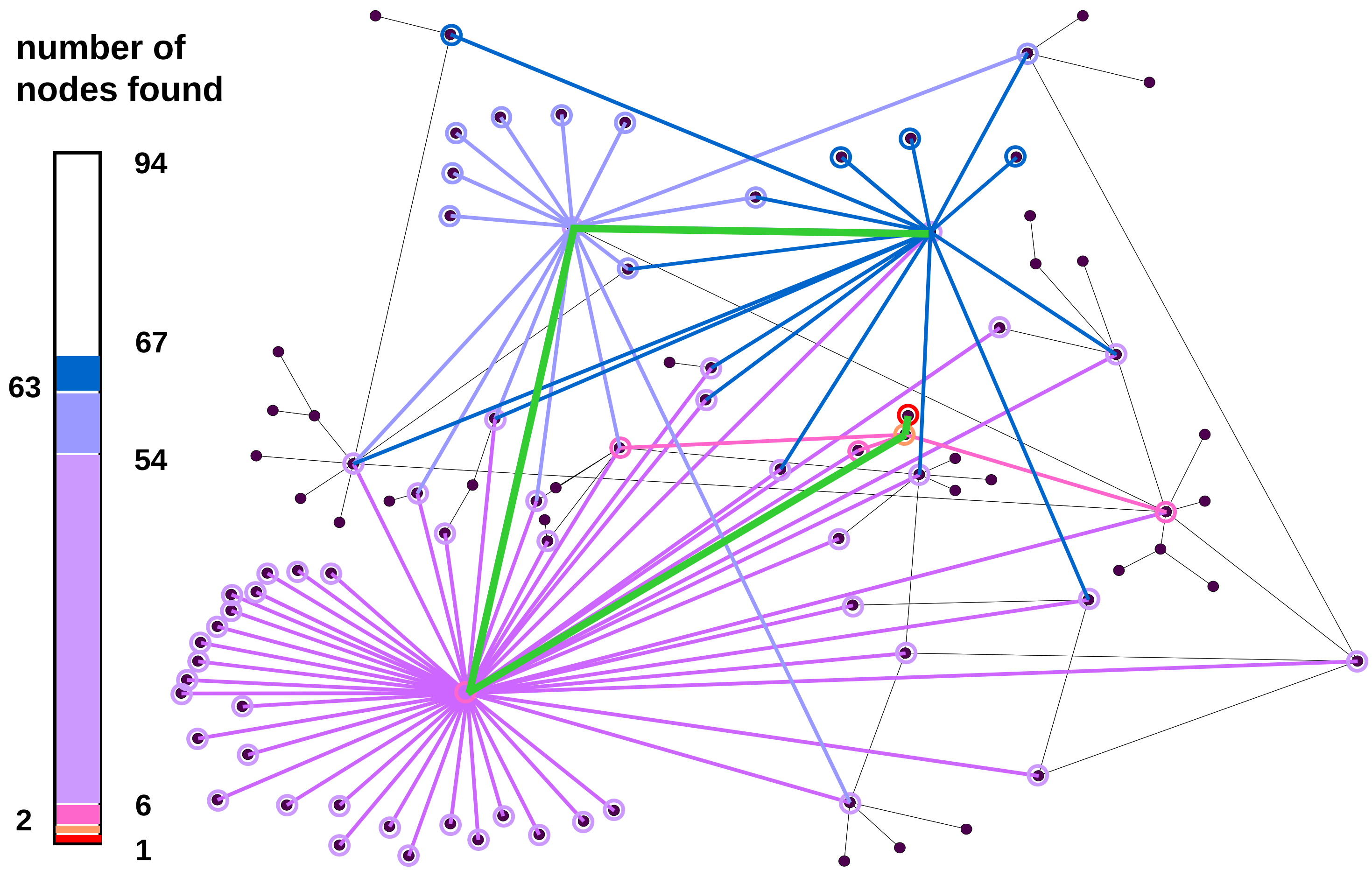
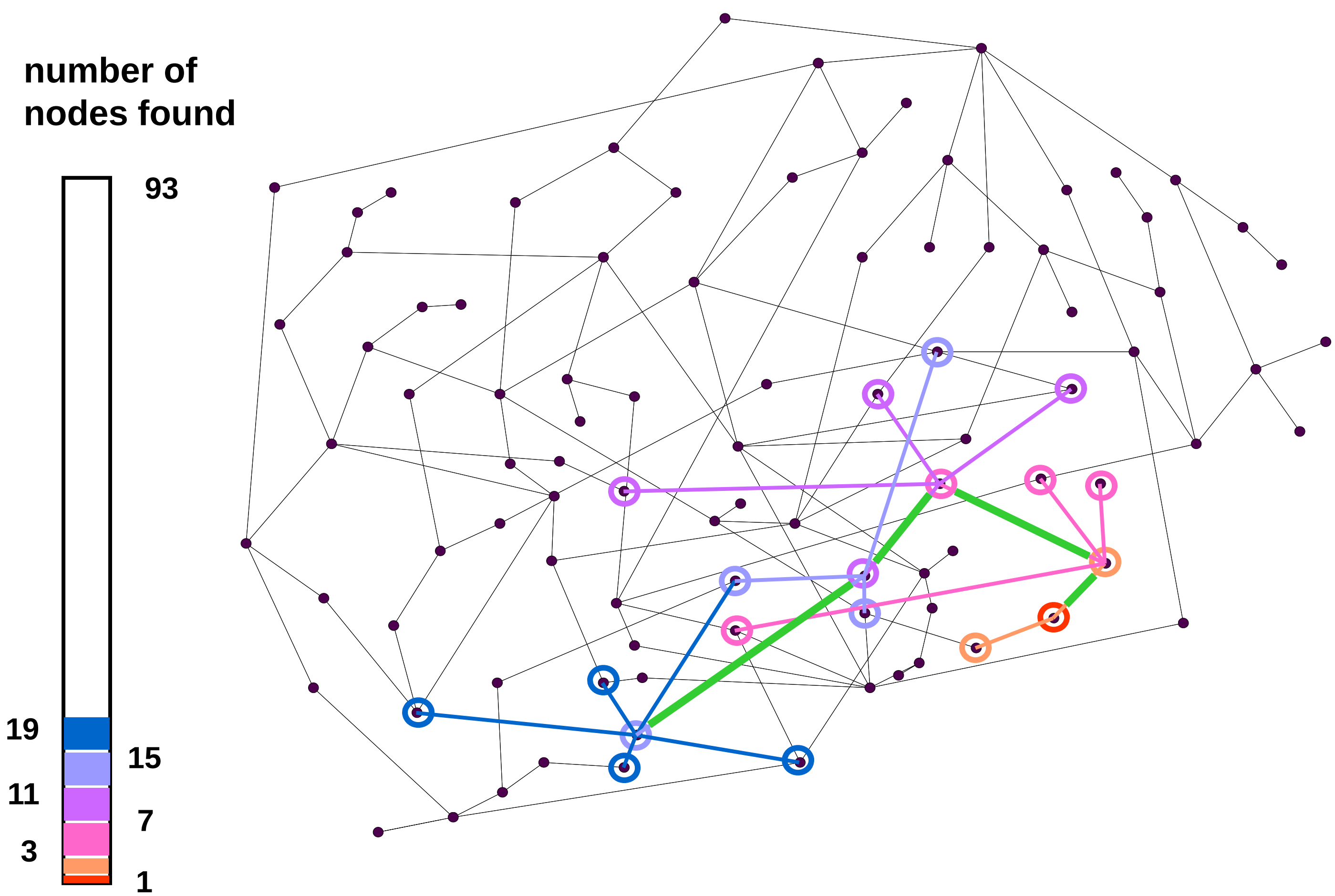
Lecture 13 & 14 & 15: Searching Networks



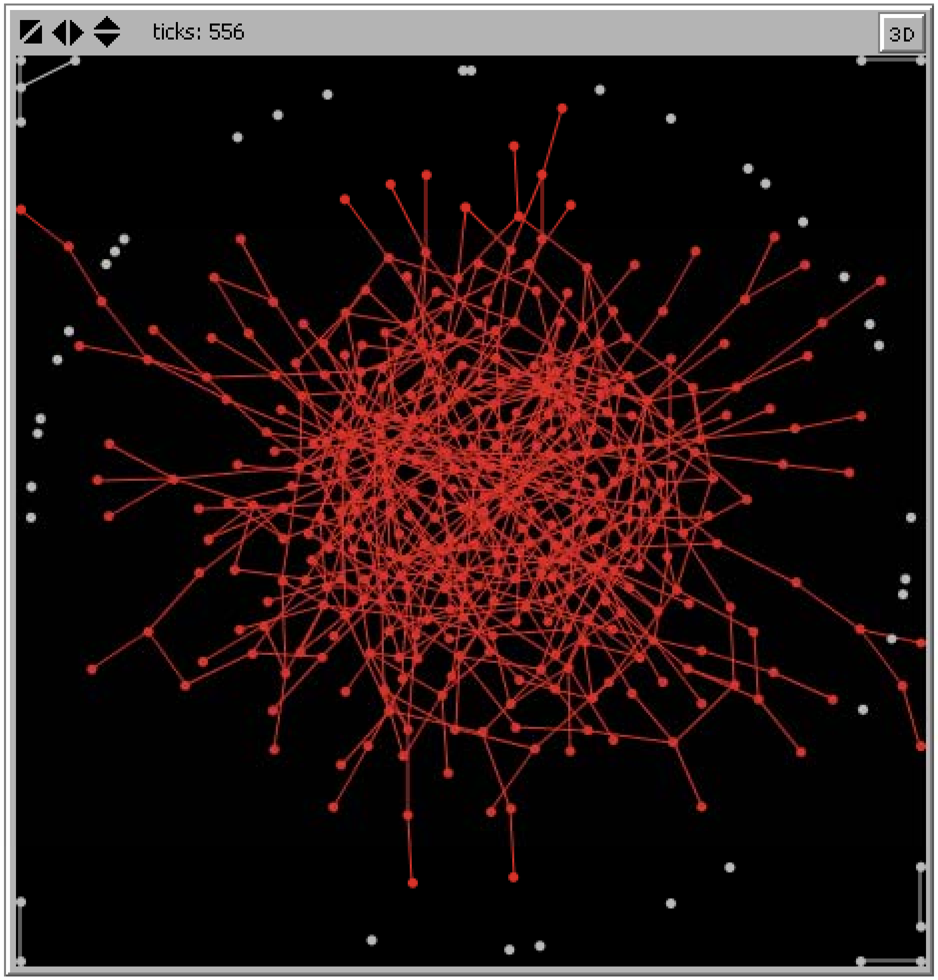
Power-law networks



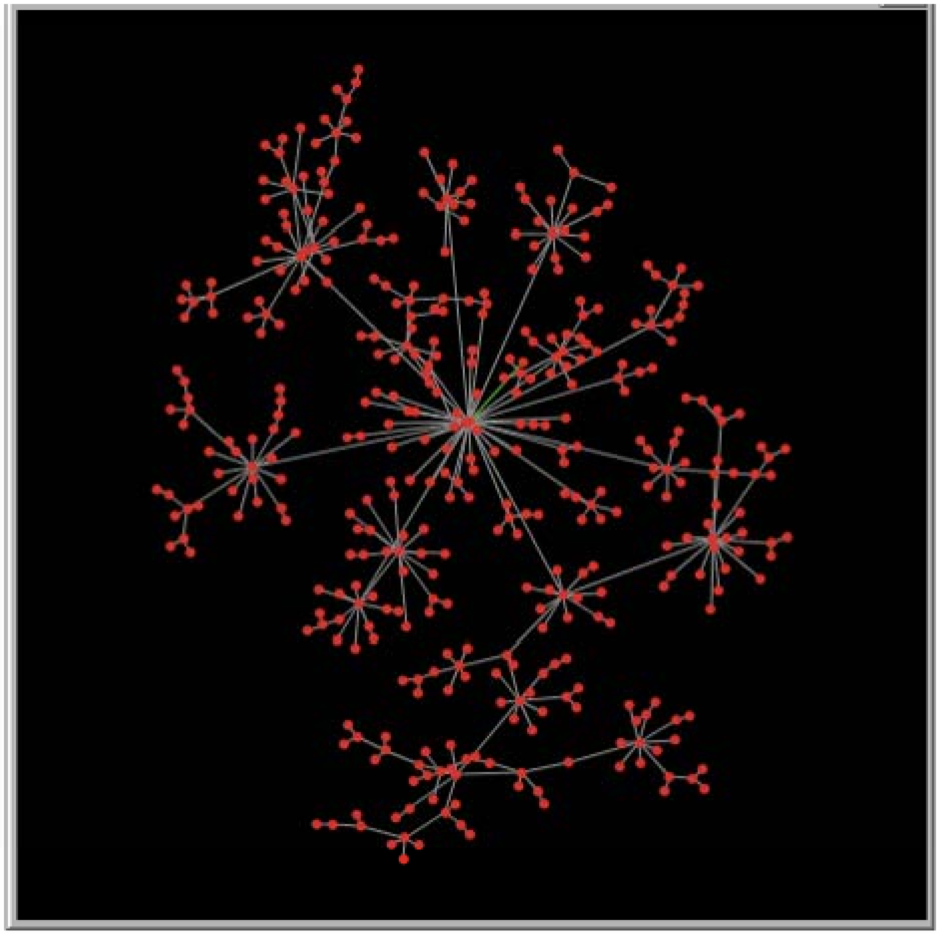
Poisson networks



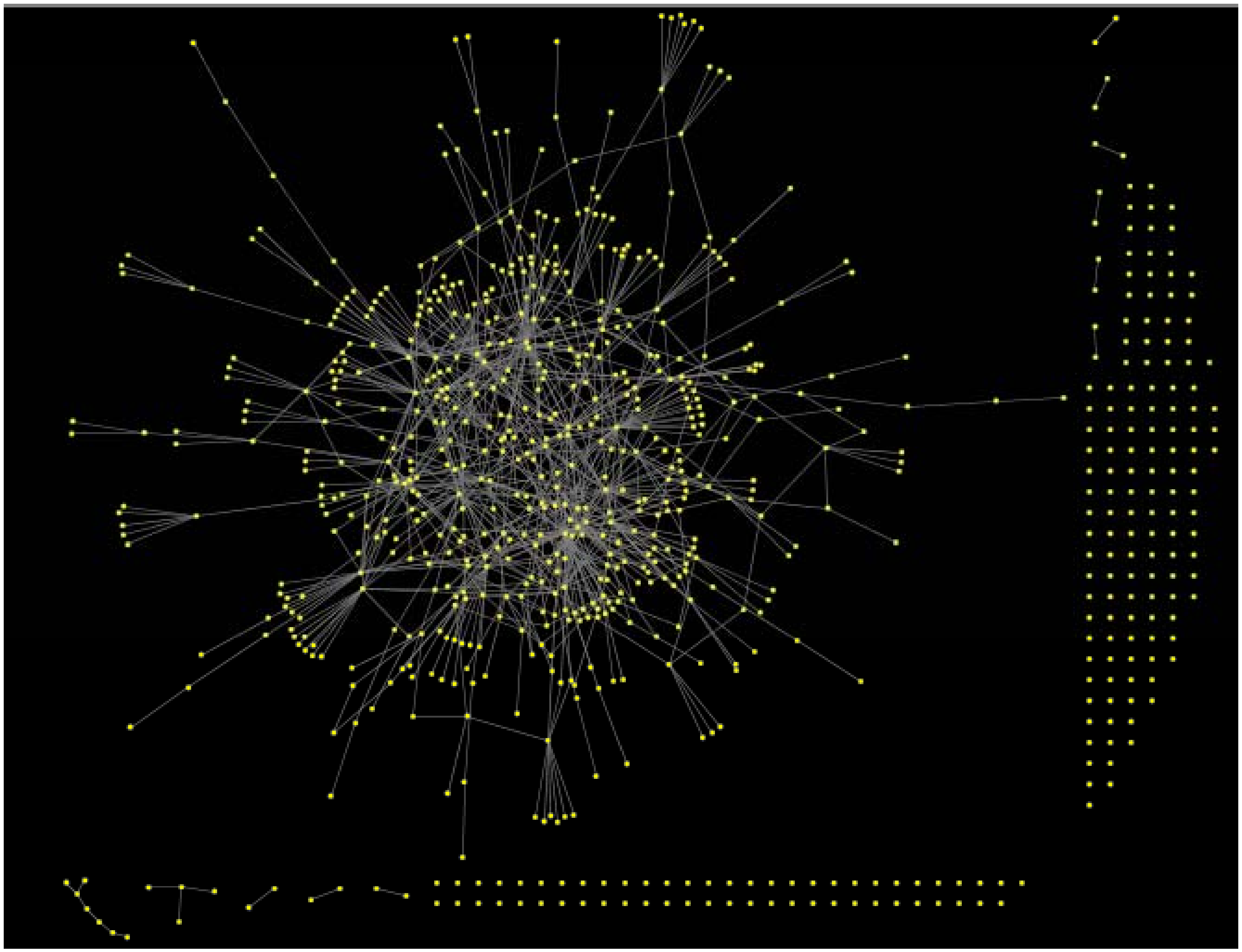
How would you search for a node here?



What about here?

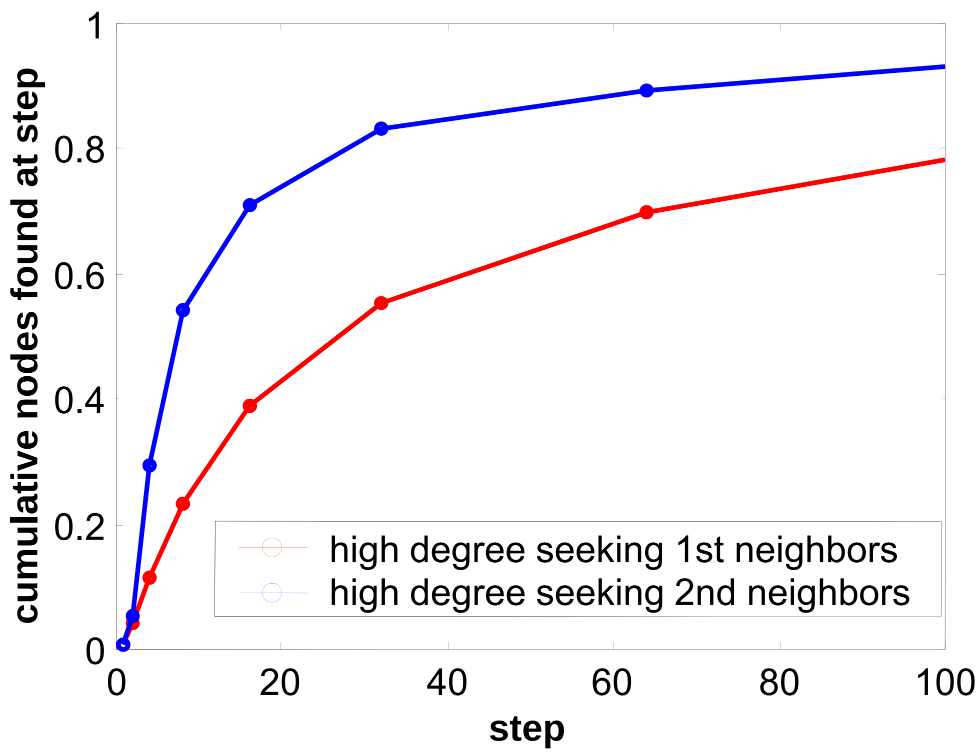


gnutella network fragment



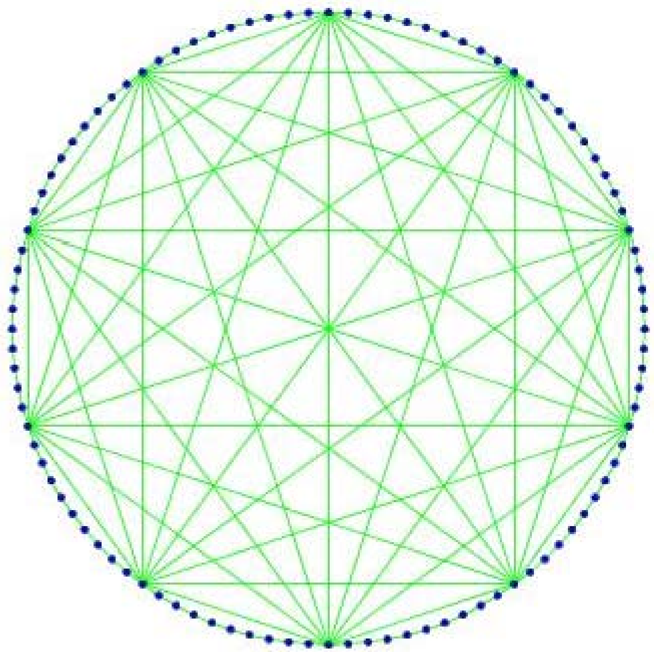
Gnutella network

50% of the files in a 700 node network can be found in < 8 steps

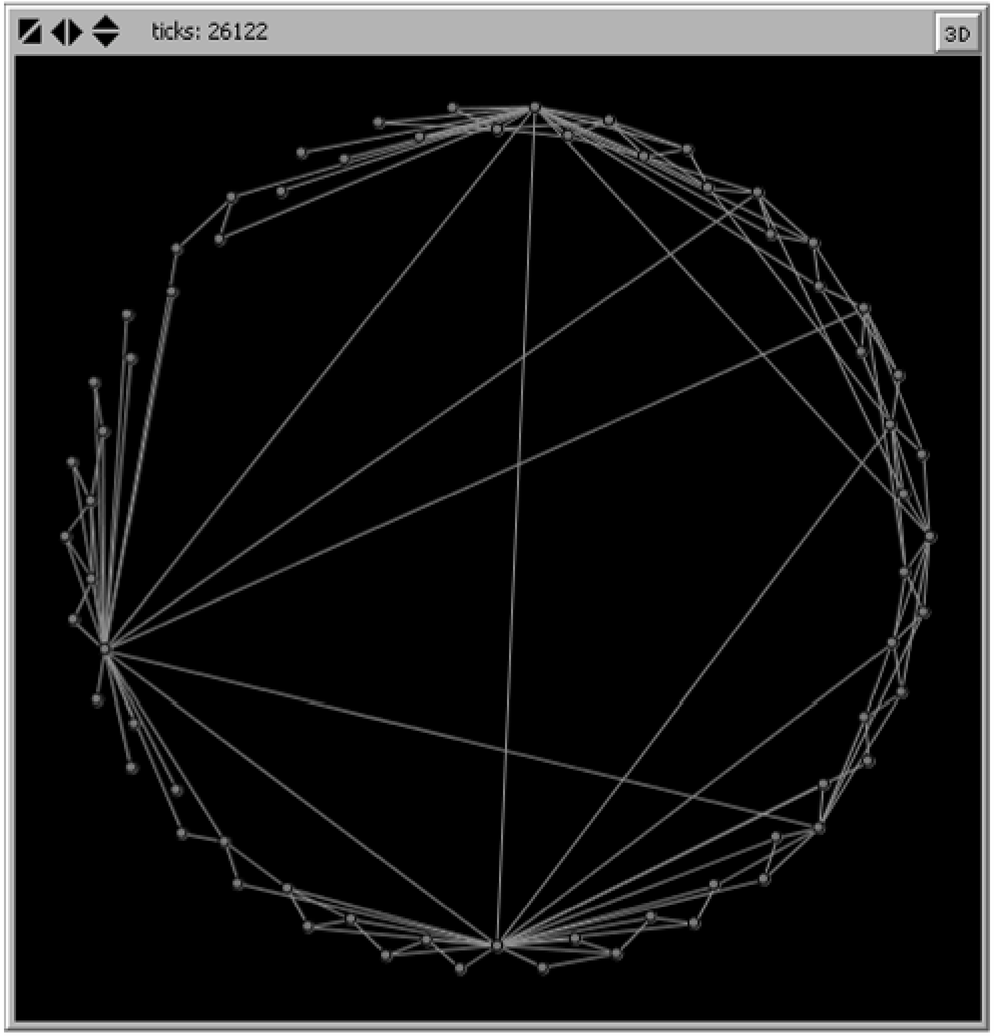


source: Adamic and Adar, **How to search a social network**, Social Networks, 27(3), p.187-203, 2005.

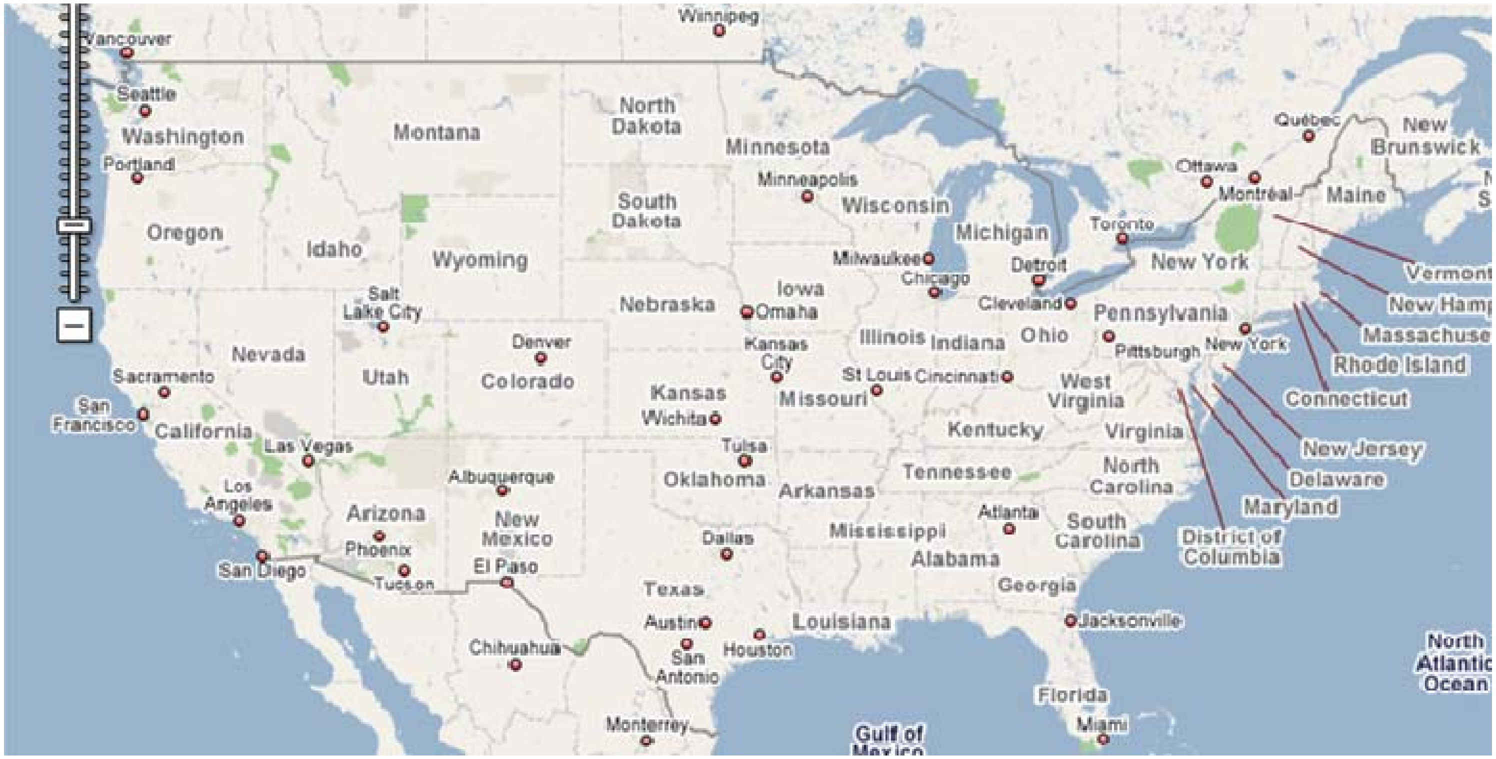
And here?



here?



here?

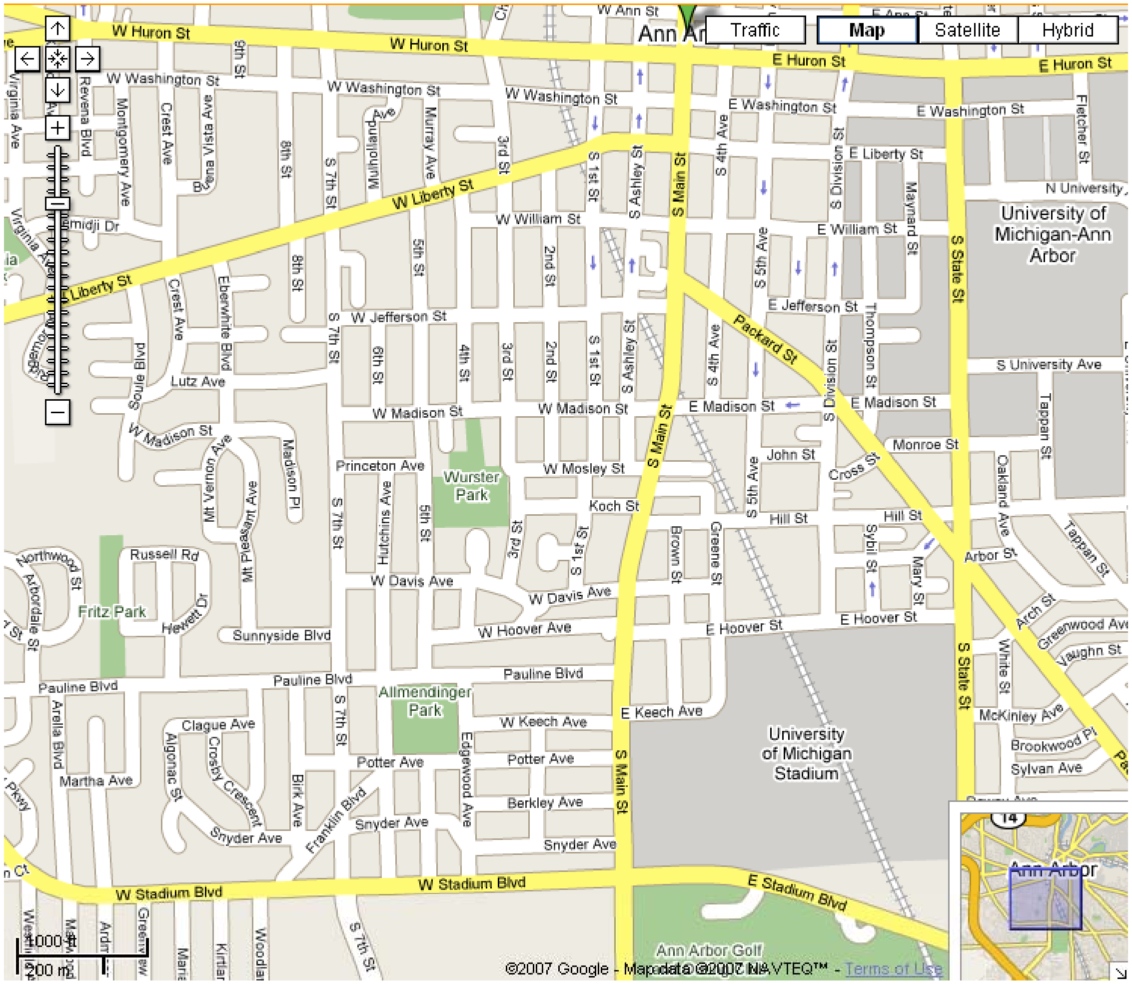


Source: http://maps.google.com

here?

Source: http://maps.google.com

here?



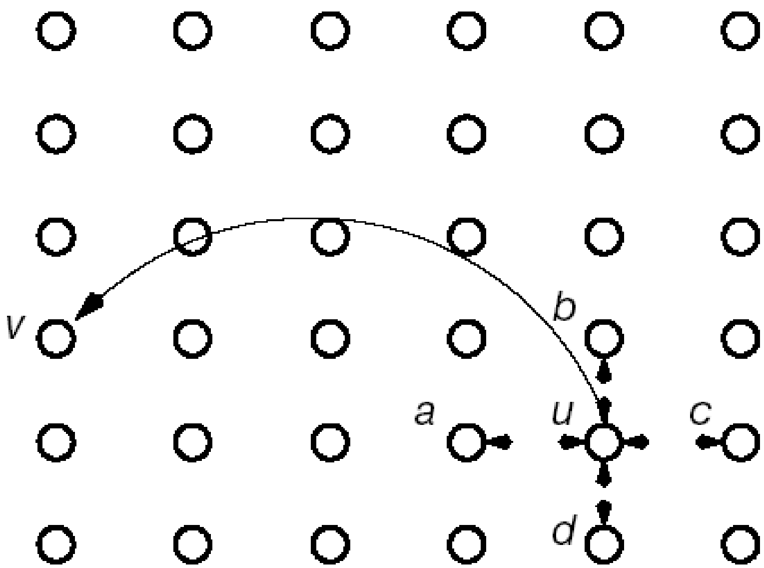
Source: http://maps.google.com

How people find shortest paths

- How to choose among hundreds of acquaintances?
- Remember the stories about small-world networks!
- **Strategy:**
  - Simple greedy algorithm - each participant chooses correspondent who is closest to target with respect to the given property
- **<u>Models</u>**
  - Geographical by Kleinberg (2000)
  - hierarchical groups by Watts, Dodds, Newman (2001), Kleinberg(2001)
  - high degree nodes by Adamic, Puniyani, Lukose, Huberman (2001), Newman(2003)

Spatial search

Kleinberg, 'The Small World Phenomenon, An Algorithmic Perspective' (Proc. 32nd ACM Symposium on Theory of Computing, 2000) (Nature 2000)



"The geographic movement of the [message] from Nebraska to Massachusetts is striking. There is a progressive closing in on the target area as each new person is added to the chain"
S.Milgram 'The small world problem', Psychology Today 1,61,1967

nodes are placed on a lattice and connect to nearest neighbors

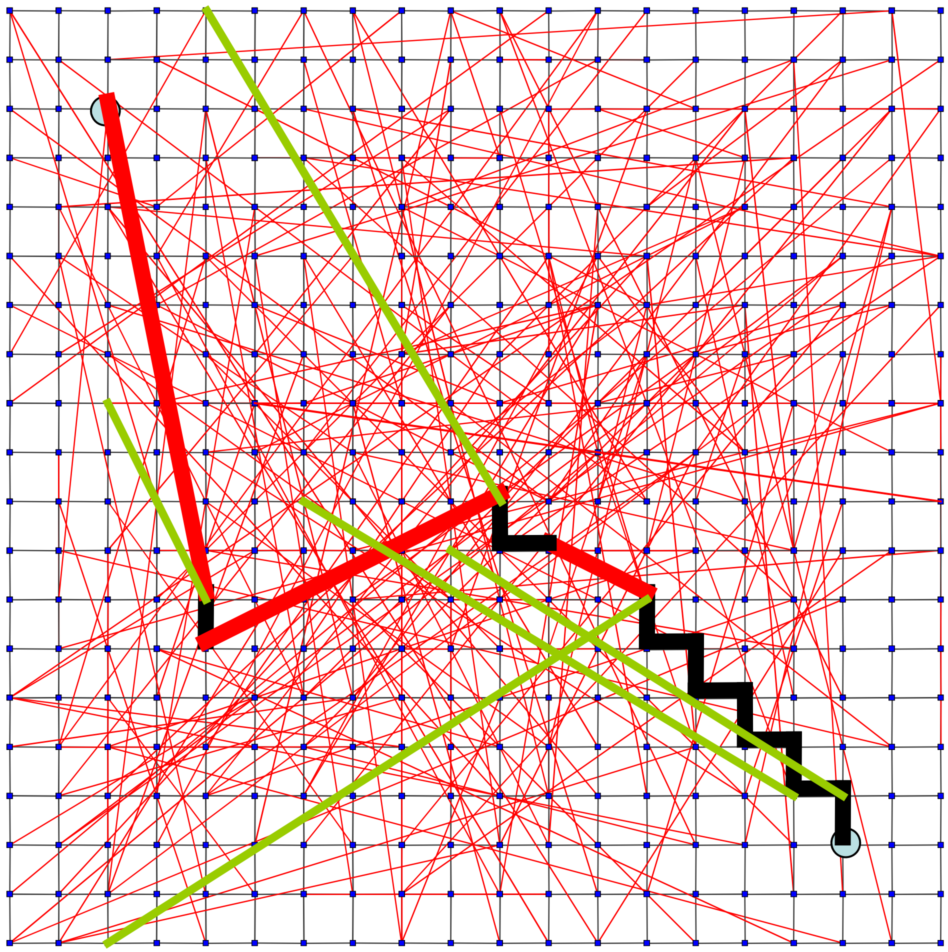
additional links placed with $p_{uv} \sim d_{uv}^{-r}$

No locality

When **r=0**, links are randomly distributed, ASP ~ **log(n)**, n size of grid

When **r=0**, any decentralized algorithm is at least $\mathbf{a_0 n^{2/3}}$

$p \sim p_0$

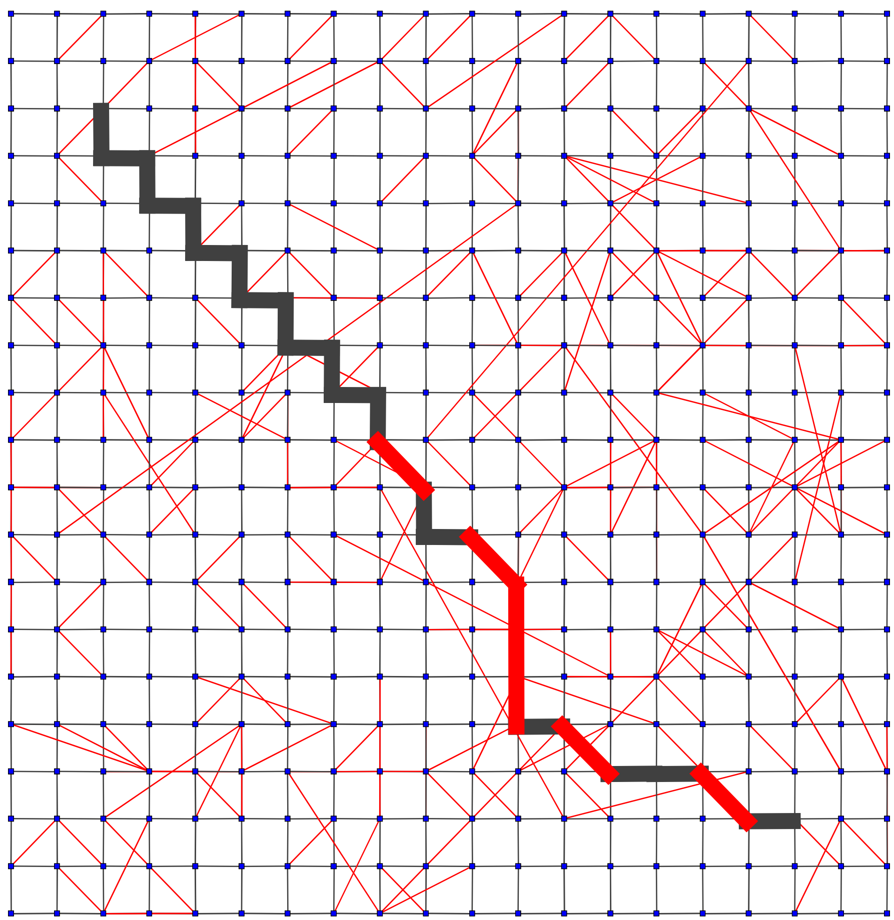


When **r<2**, expected time at least $\alpha_r n^{(2-r)/3}$

Overly localized links on a lattice

When **r>2** expected search time ~ $\mathbf{N^{(r-2)/(r-1)}}$

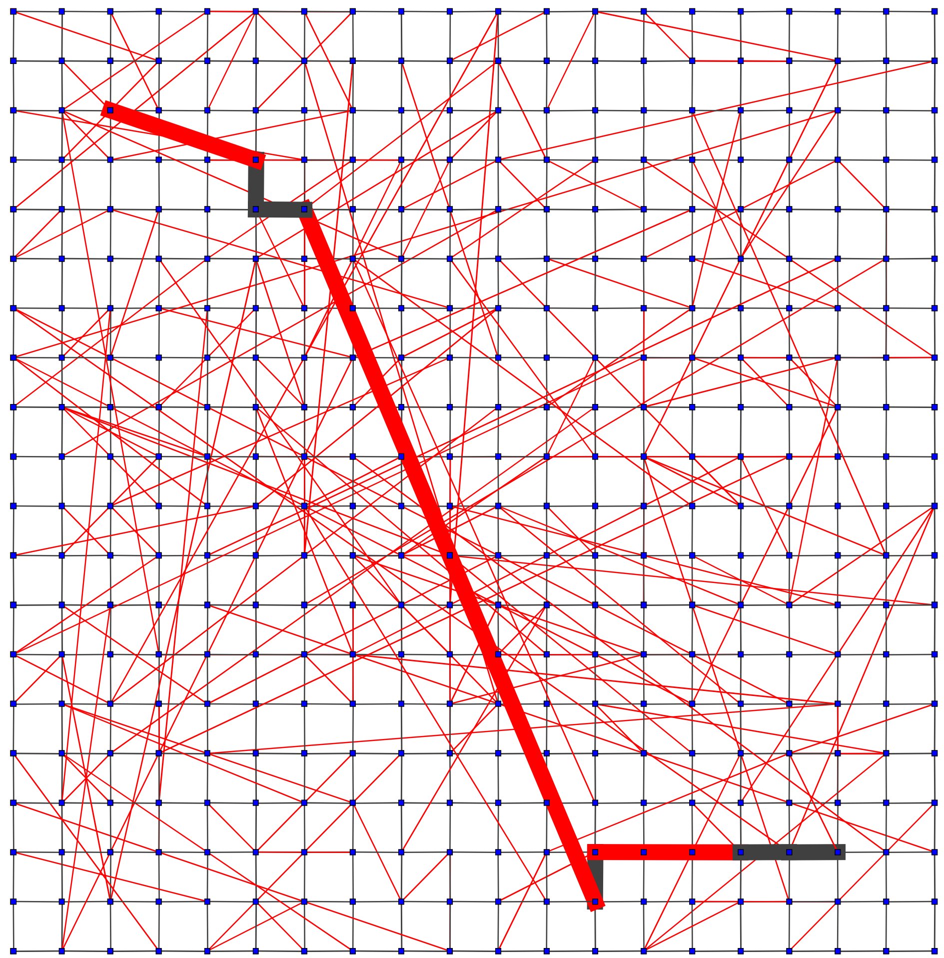
$$p \sim \frac{1}{d^4}$$



Links balanced between long and short range

When **r=2,** expected time of a decentralized algorithm is at most **C (log N)²**

$$p \sim \frac{1}{d^2}$$

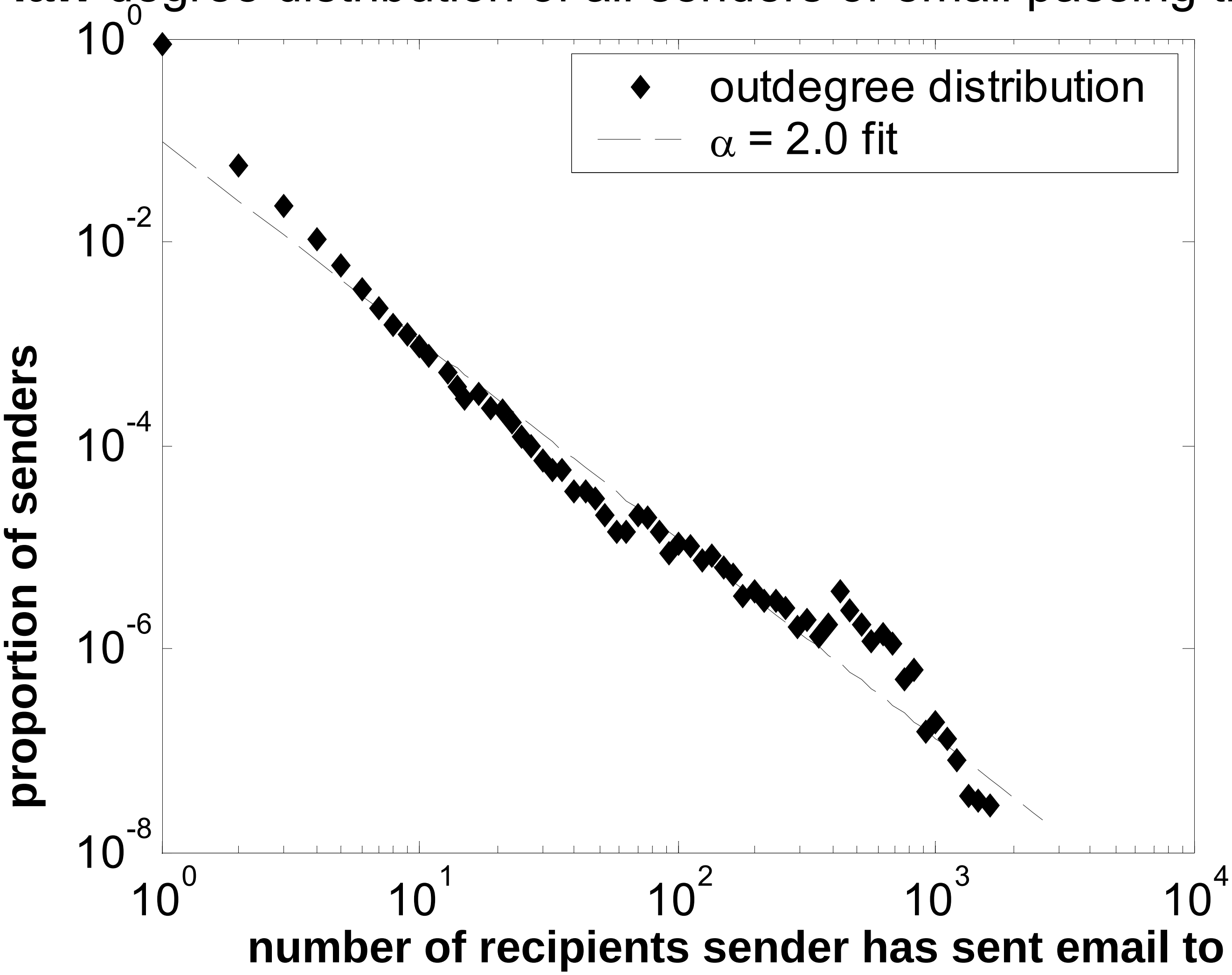


Testing search models on real social networks

- **Use a well defined network:**
- HP Labs email correspondence over 3.5 months
  - Edges are between individuals who sent at least 6 email messages each way
  - 450 users
  - median degree = 10
  - mean degree = 13
  - average shortest path = 3
- **Node properties specified:**
  - degree
  - geographical location
  - position in organizational hierarchy
- **Can greedy strategies work?**

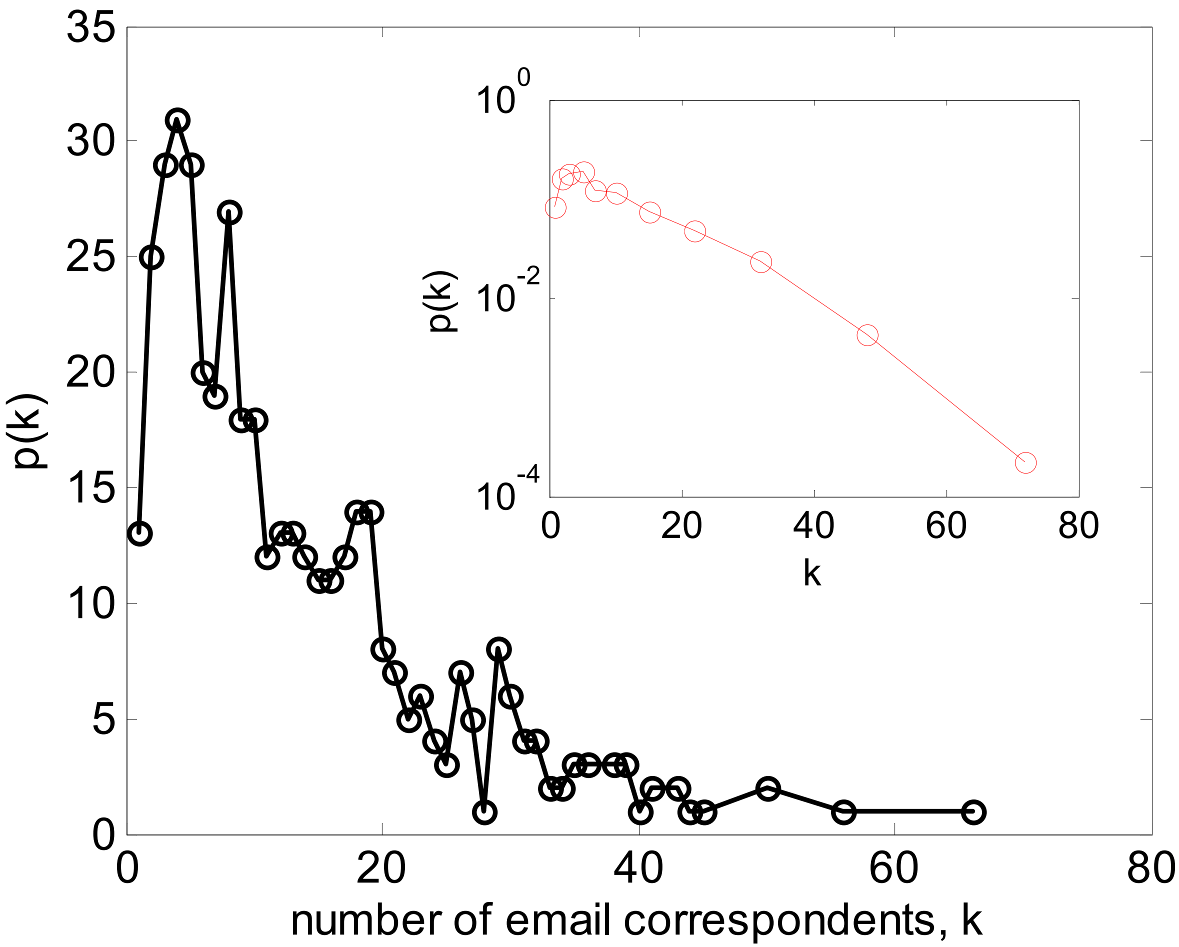
Strategy 1: high degree search

Power-law degree distribution of all senders of email passing through HP labs



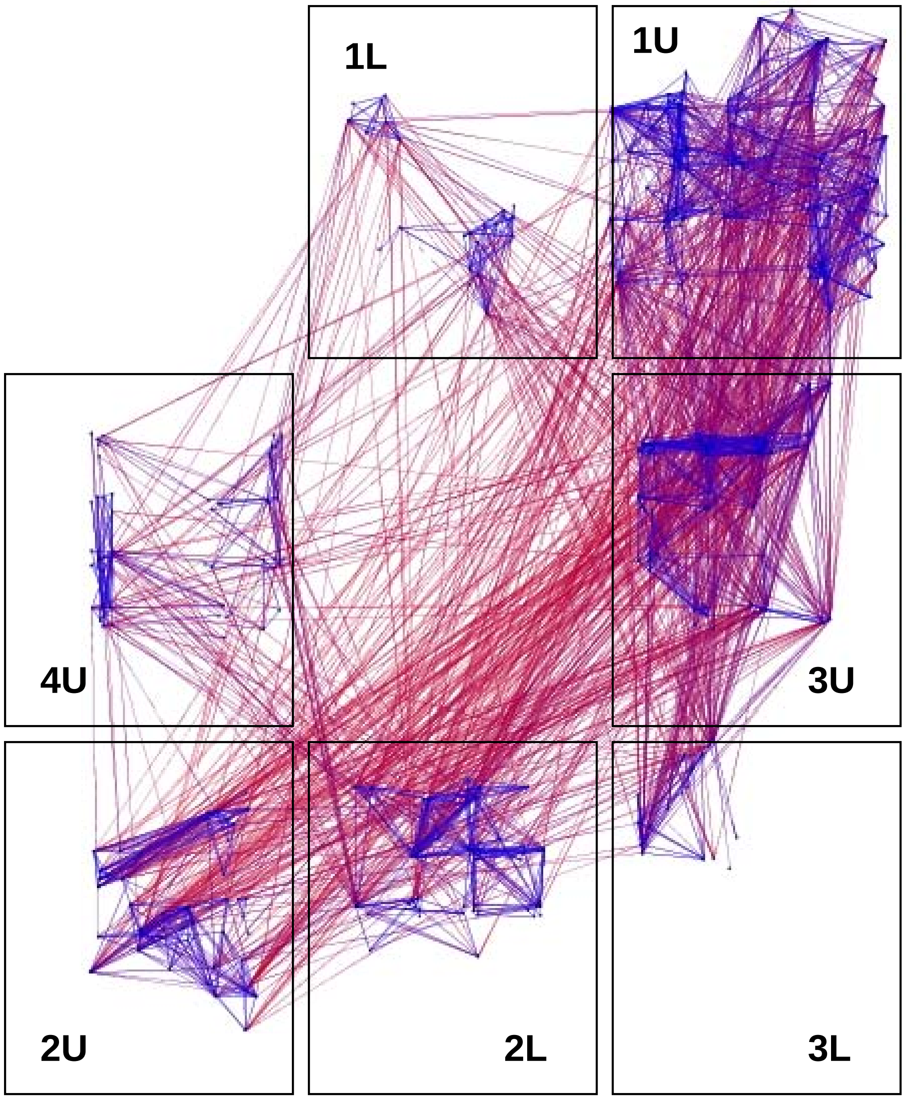
Filtered network (at least 6 messages sent each way)

Degree distribution no longer power-law, but Poisson



It would take 40 steps on average (median of 16) to reach a target!

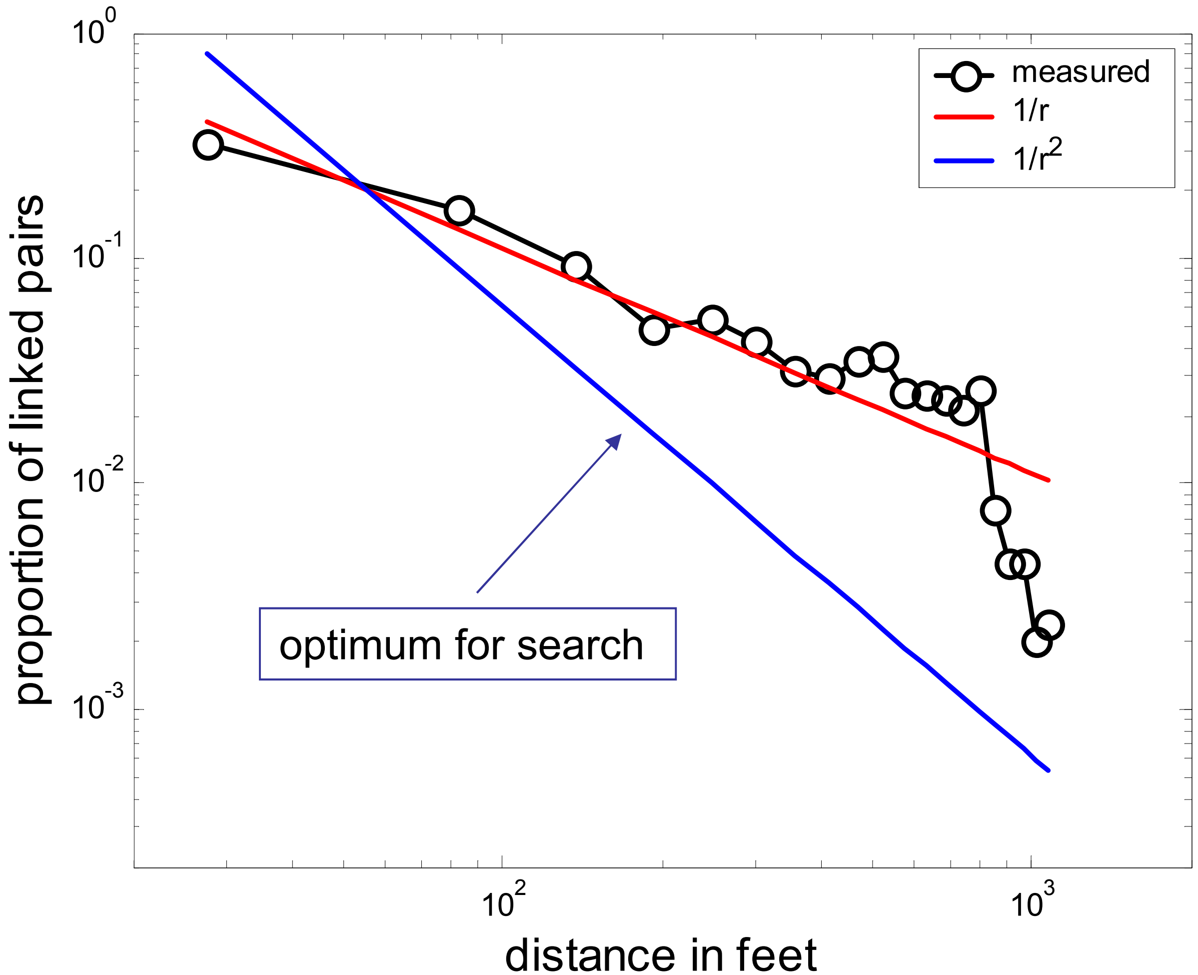
Strategy 2: communication across corporate geography



87 % of the 4000 links are between individuals on the same floor

source: Adamic and Adar, **How to search a social network**, Social Networks, 27(3), p.187-203, 2005.

Cubic distance vs. probability of being linked



source: Adamic and Adar, **How to search a social network**, Social Networks, 27(3), p.187-203, 2005.

LiveJournal

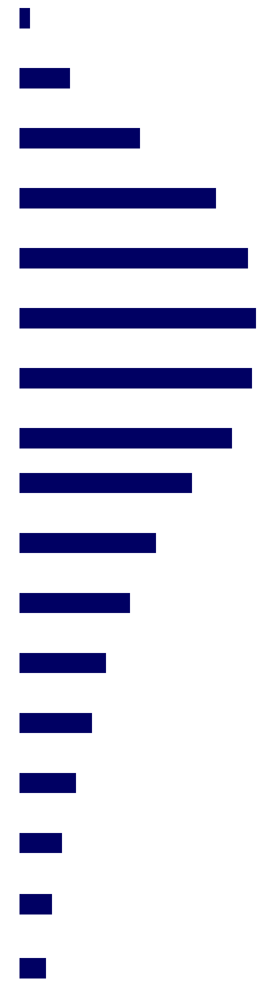
- LiveJournal provides an API to crawl the friendship network + profiles
  - friendly to researchers
  - great research opportunity
- basic statistics
  - Users
    - How many users, and how many of those are active?
    - Total accounts: 9980558
    - ... active in some way: 1979716
    - ... that have ever updated: 6755023
    - ... updating in last 30 days: 1300312
    - ... updating in last 7 days: 751301
    - ... updating in past 24 hours: 216581

Predominantly female & young demographic

- **Male:** 1370813 (32.4%)
- **Female:** 2856360 (67.6%)
- **Unspecified:** 1575389

Age distribution

13	18483
14	87505
15	211445
16	343922
17	400947
18	414601
19	405472
20	371789
21	303076
22	239255
23	194379
24	152569
25	127121
26	98900
27	73392
28	59188
29	48666

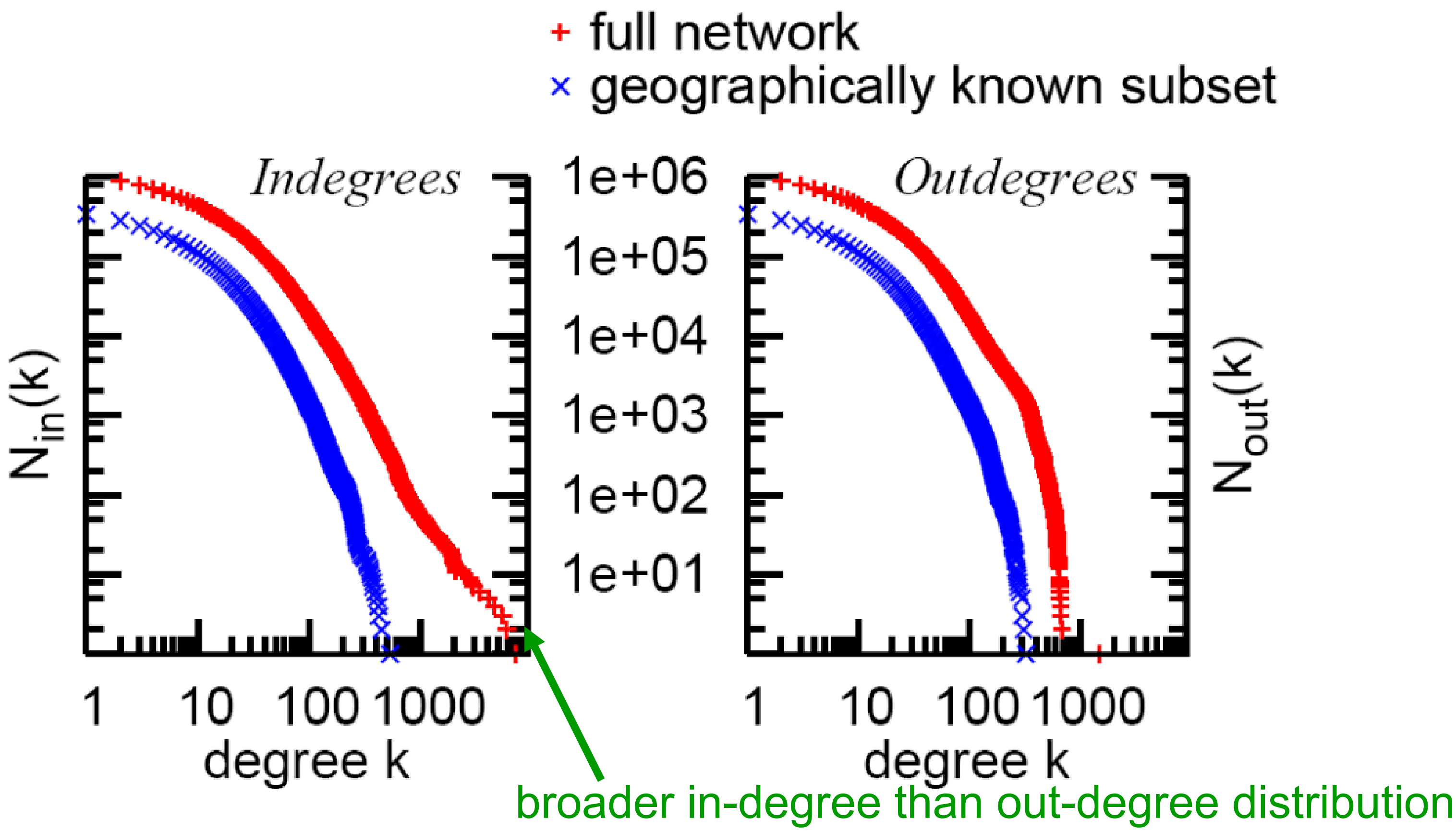


Geographic Routing in Social Networks

- David Liben-Nowell, Jasmine Novak, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins (PNAS 2005)
- data used
  - Feb. 2004
  - 500,000 LiveJournal users with US locations
  - giant component (77.6%) of the network
  - clustering coefficient: 0.2

Degree distributions

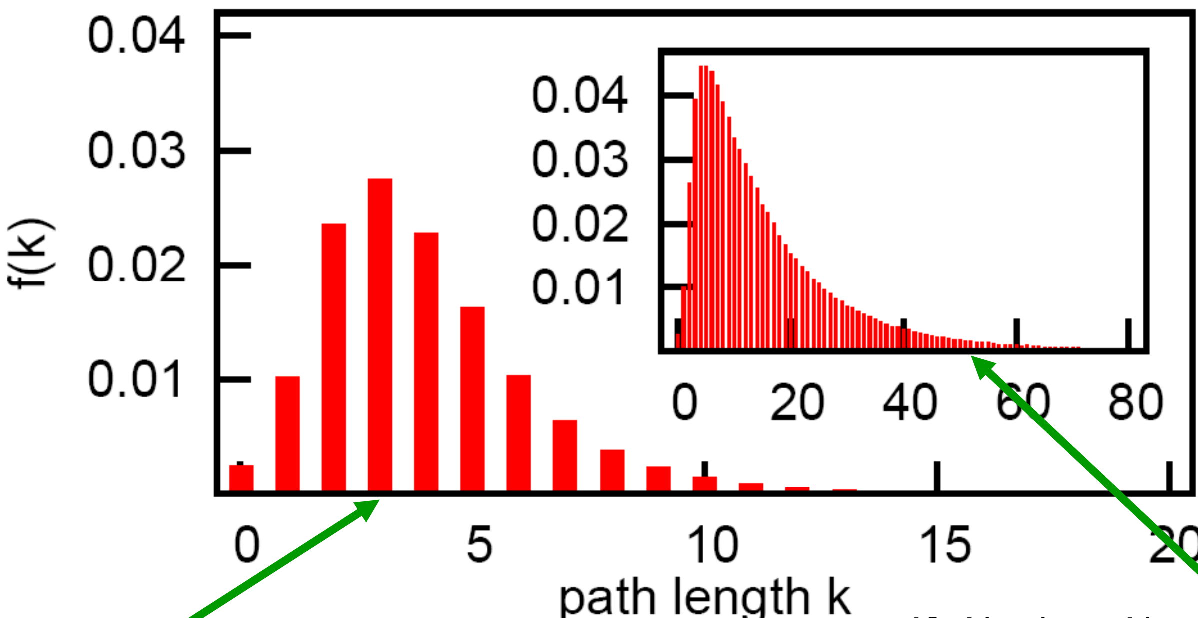
- The broad degree distributions we've learned to know and love



Source: http://www.tomkinshome.com/andrew/papers/science-blogs/pnas.pdf

Results of a simple greedy geographical algorithm

- Choose source *s* and target *t* randomly
- Try to reach target's city – not target itself
- At each step, the message is forwarded from the current message holder *u* to the friend *v* of u geographically closest to t



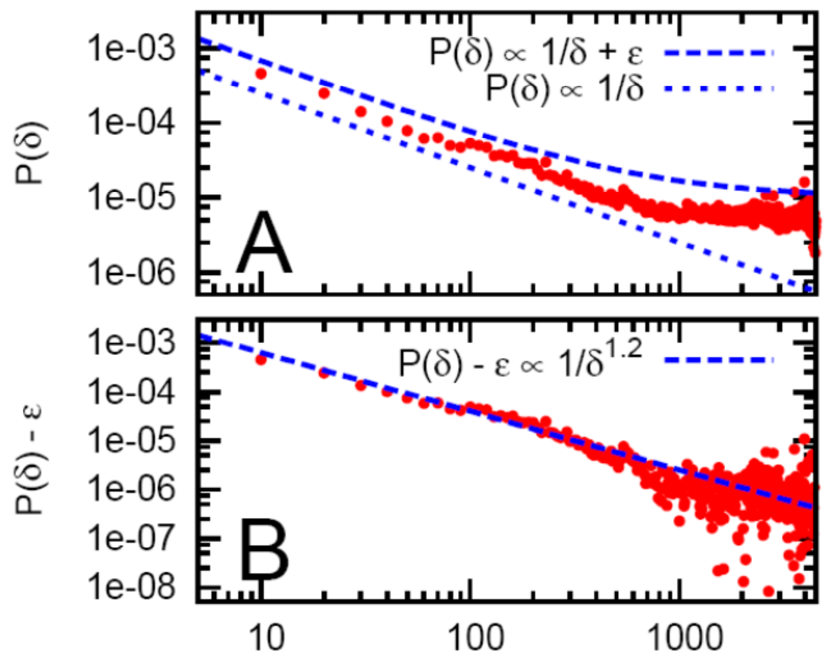
stop if d(v,t) > d(u,t)
13% of the chains are completed

stop if d(v,t) > d(u,t)
pick a neighbor at random in the same city if possible, else stop
80% of the chains are completed

Source: http://www.tomkinshome.com/andrew/papers/science-blogs/pnas.pdf

the geographic basis of friendship

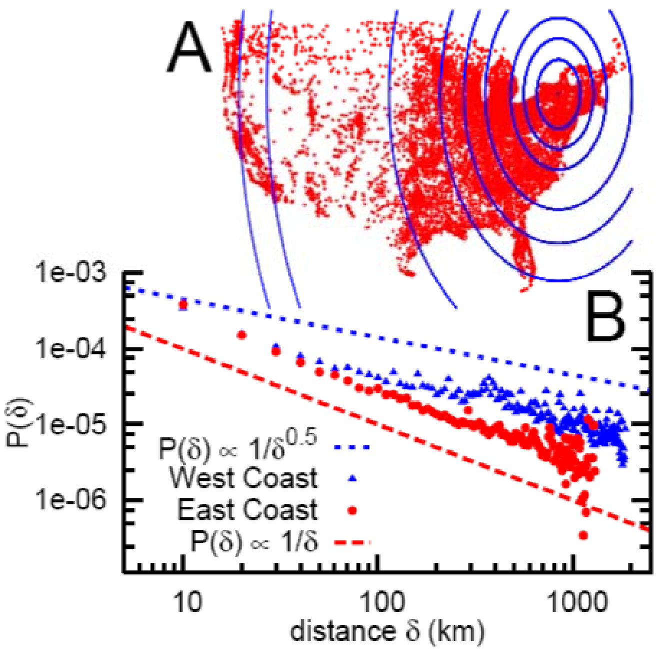
- $\delta = d(u,v)$ the distance between pairs of people
- The probability that two people are friends given their distance is equal to
  - $P(\delta) = \varepsilon + f(\delta)$, ε is a constant independent of geography
  - ε is 5.0×10^{-6} for LiveJournal users who are very far apart



Source: http://www.tomkinshome.com/andrew/papers/science-blogs/pnas.pdf

the geographic basis of friendship

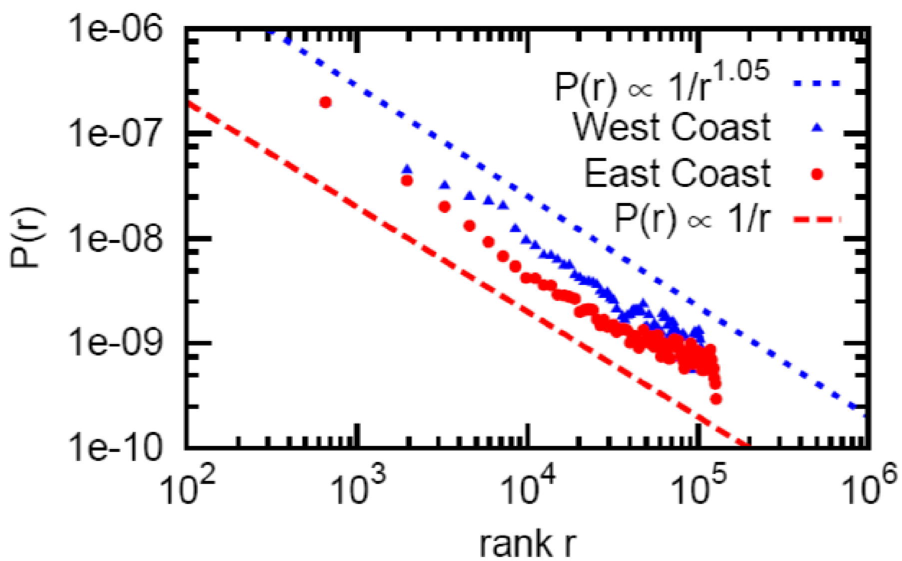
- The average user will have ~ 2.5 non-geographic friends
- The other friends (5.5 on average) are distributed according to an approximate 1/distance relationship
- But 1/d was proved not to be navigable by Kleinberg, so what gives?



Source: http://www.tomkinshome.com/andrew/papers/science-blogs/pnas.pdf

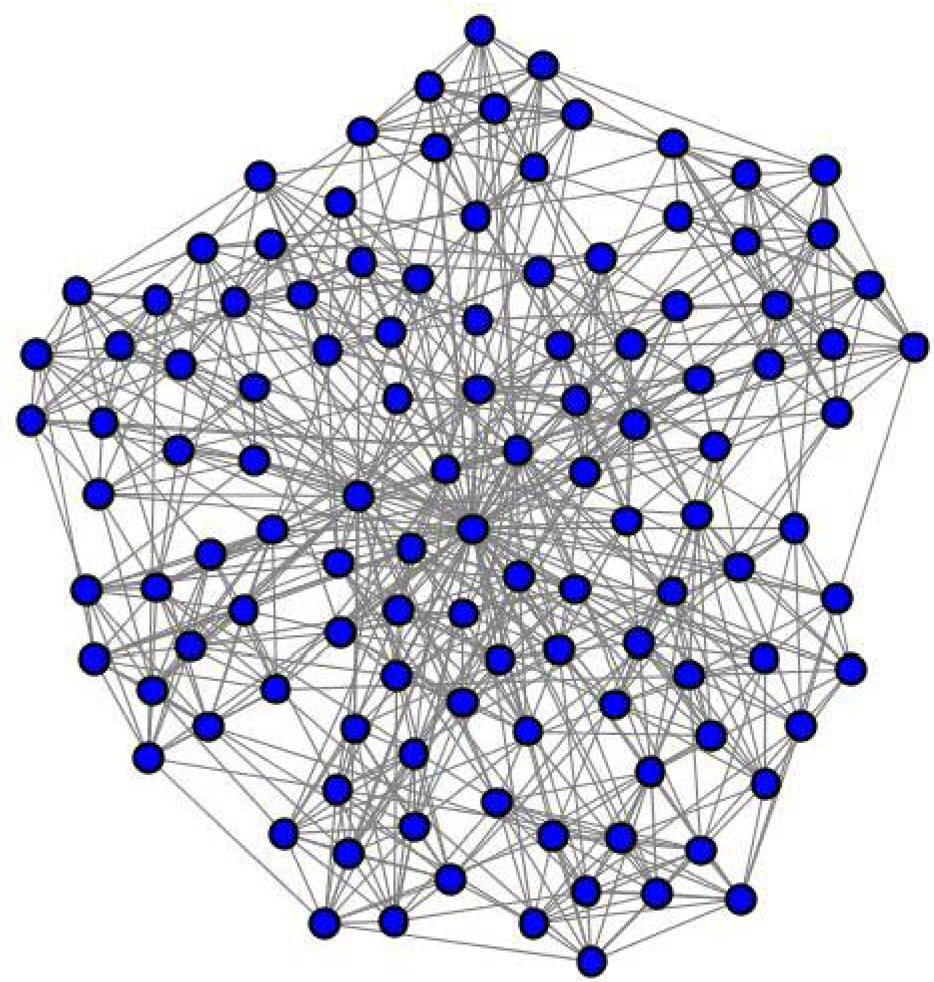
Navigability in networks of variable geographical density

- Kleinberg assumed a uniformly populated 2D lattice
- But population is far from uniform
- population networks and rank-based friendship
  - probability of knowing a person depends not on absolute distance but on relative distance (i.e. how many people live closer) Pr[u ->v] ~ 1/rank$_u$(v)

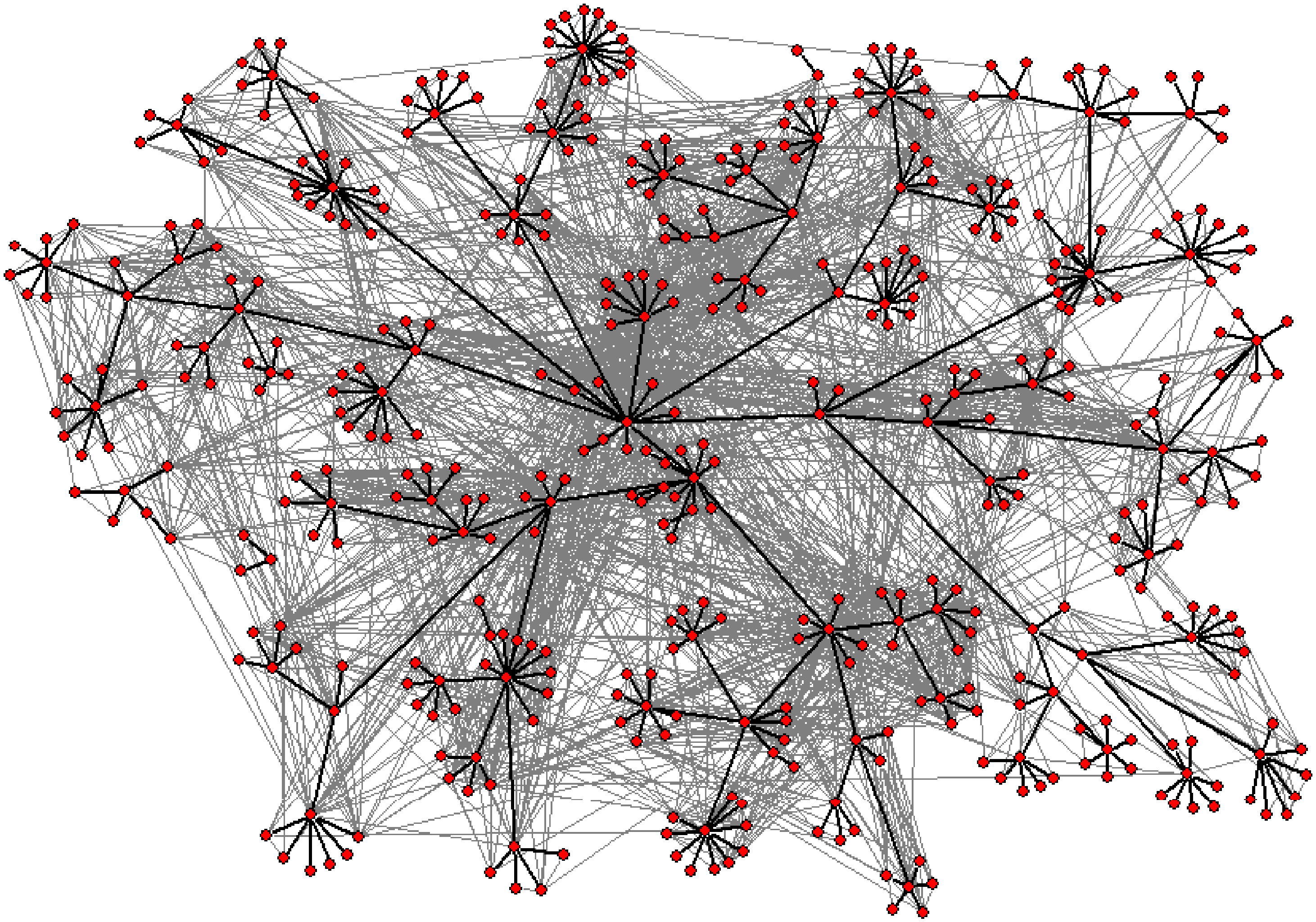


Source: http://www.tomkinshome.com/andrew/papers/science-blogs/pnas.pdf

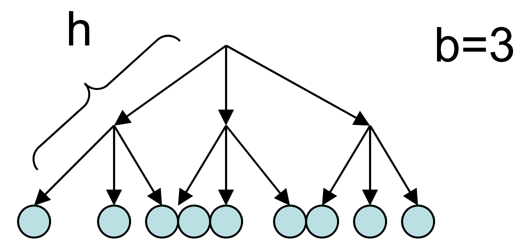
what if we don’t have geography?



does community structure help?



Review: hierarchical small-world models



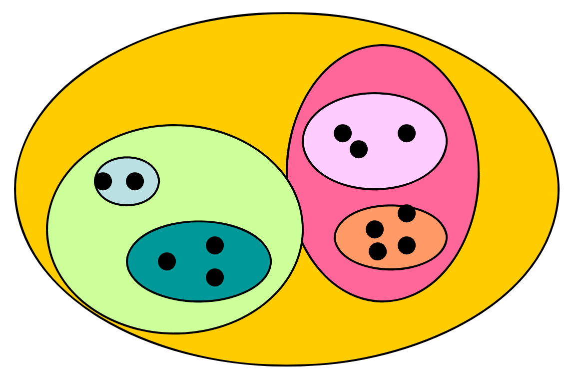
Individuals classified into a hierarchy,
h_{ij} = height of the least common ancestor.

$$p_{ij} \sim b^{-\alpha h_{ij}}$$

e.g. state-county-city-neighborhood
industry-corporation-division-group

Theorem: If α = 1 and outdegree is polylogarithmic, can
s ~ O(log n)

Group structure models:
Individuals belong to nested groups
q = size of smallest group that v,w belong to



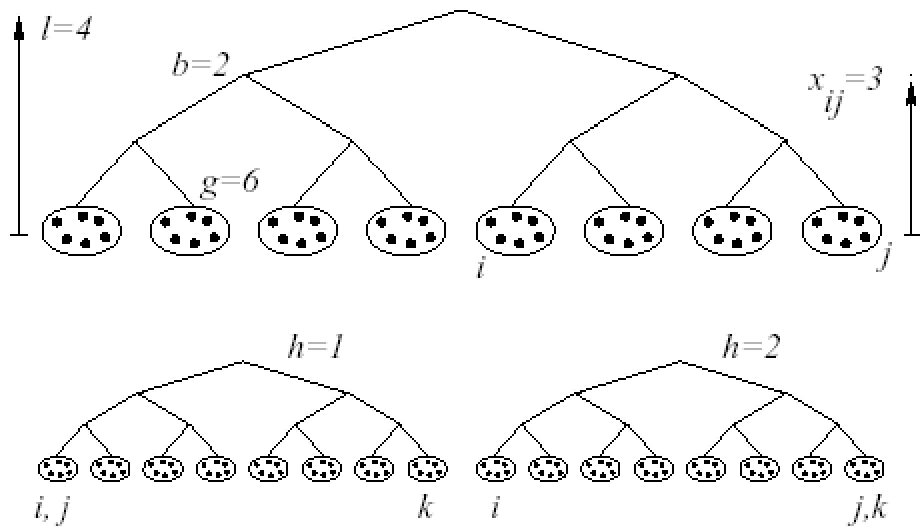
$$f(q) \sim q^{-\alpha}$$

Theorem: If α = 1 and outdegree is polylogarithmic, can
s ~ O(log n)

Kleinberg, 'Small-World Phenomena and the Dynamics of Information', NIPS 14, 2001

Hierarchical models with multiple hierarchies

individuals belong to hierarchically nested groups

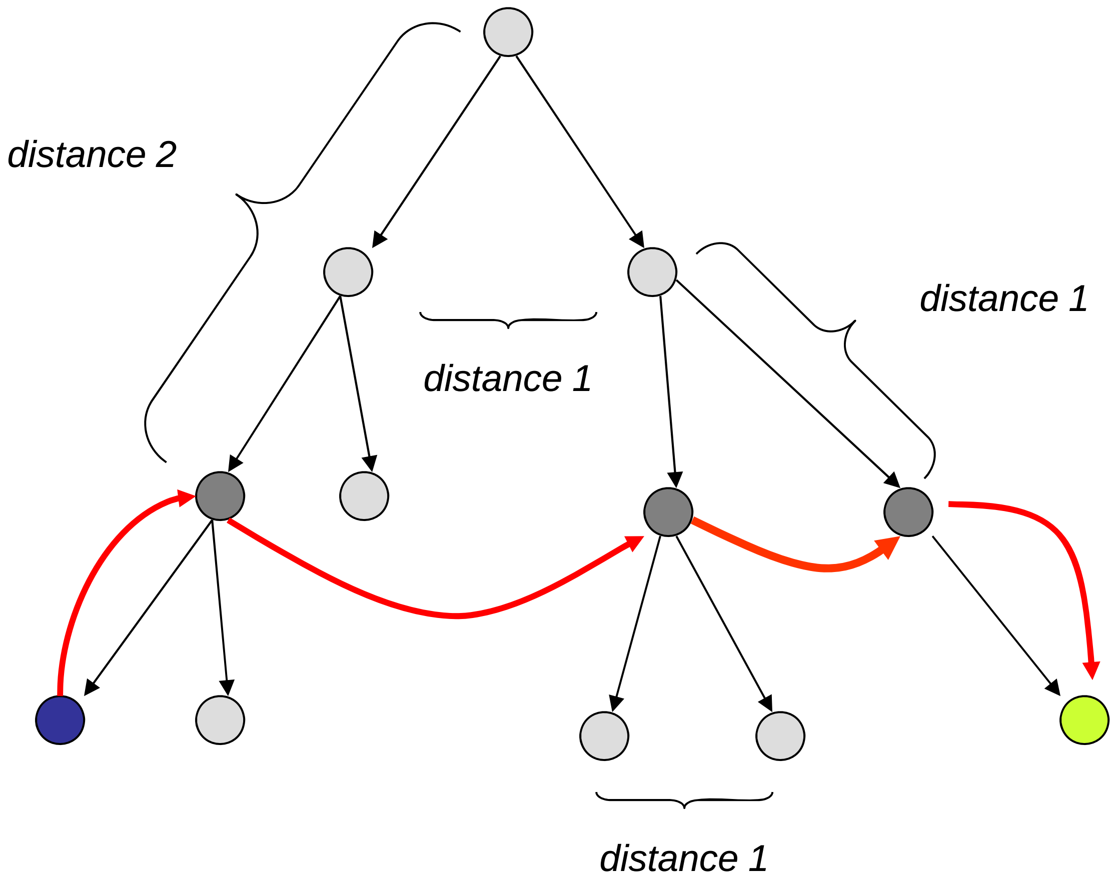


$p_{ij} \sim \exp(-\alpha\ x)$

multiple independent hierarchies h=1,2,..,H coexist corresponding to occupation, geography, hobbies, religion…

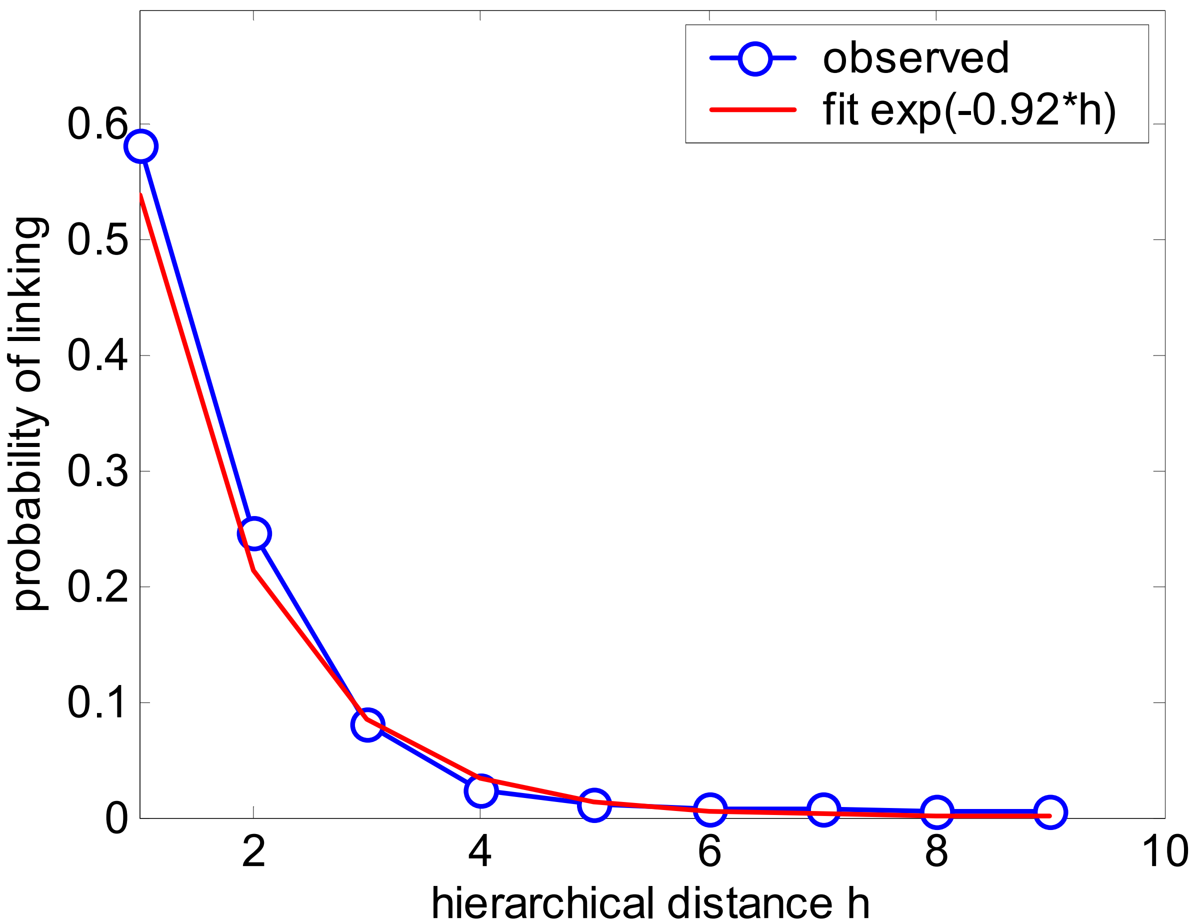
Source: Identity and Search in Social Networks: Duncan J. Watts, Peter Sheridan Dodds, and M. E. J. Newman; Science 17 May 2002 296: 1302-1305. < http://arxiv.org/abs/cond-mat/0205383v1 >

Example of search path



hierarchical distance = 5
search path distance = 4

Probability of linking vs. distance in hierarchy



in the 'searchable' regime: 0 < α < 2 (Watts, Dodds, Newman 2001)

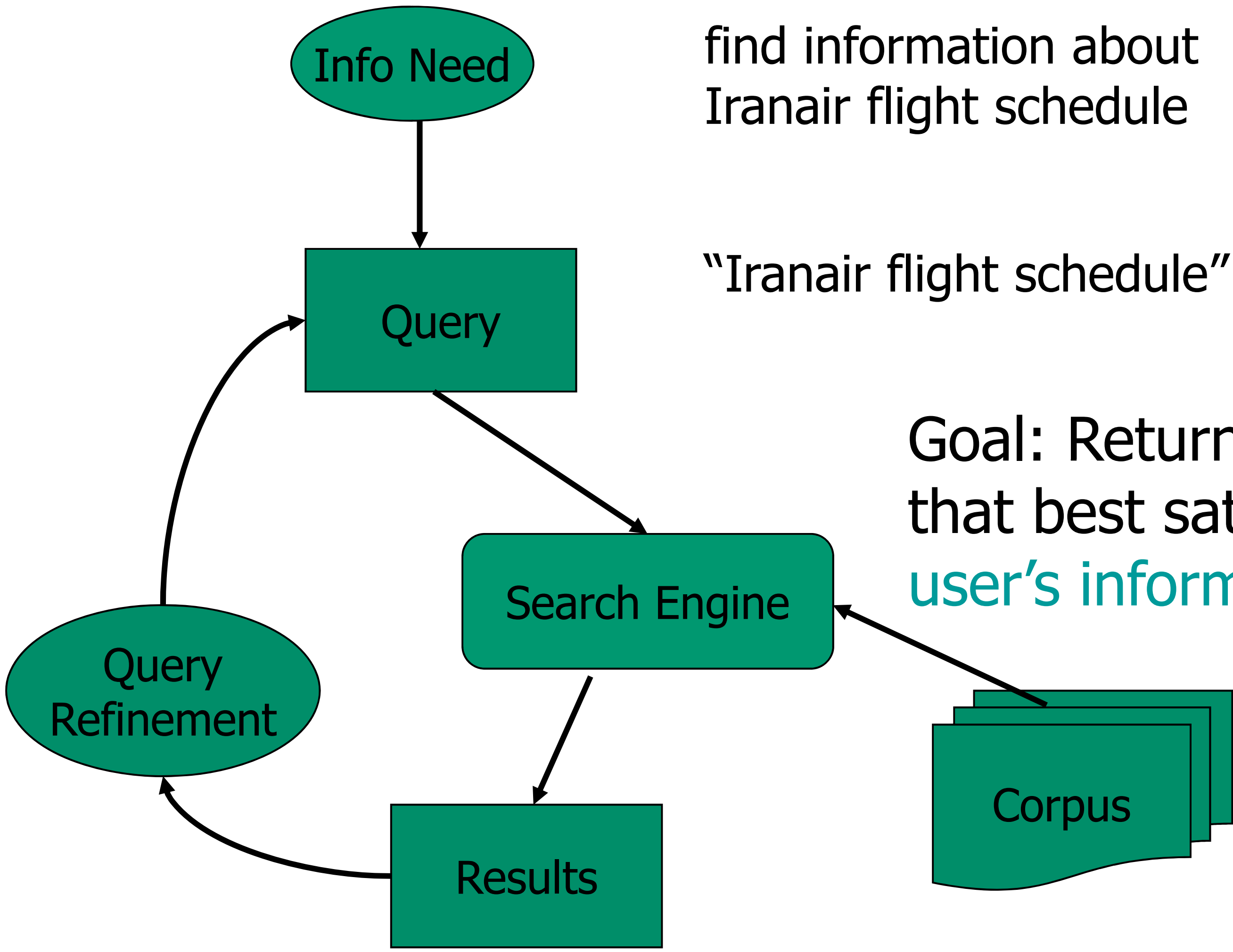
Web Search

Searching the Web

Why Web Search?

- Search is the main motivation for the development of the Web
  - people post information because they want it to be found
  - people are conditioned to searching for information on the Web ("Google it")
  - The main tool is text search
    - directories cover less than 0.05% of the Web
    - 13% of traffic is generated by search engines
- Great motivation for academic and research work
  - Information Retrieval and data mining of massive data
  - Graph theory and mathematical models
  - Security and privacy issues

Classical Information Retrieval (IR)



Goal: Return the documents that best satisfy the user's information need

Classical IR

- Implicit Assumptions
  - fixed and well structured corpus of manageable size
  - trained cooperative users
  - controlled environment
- Classic Relevance
  - For each query Q and document D assume that there exists a relevance score S(D,Q)
    - score average over all users U and contexts C
  - Rank documents according to S(D,Q) as opposed to S(D,Q,U,C)
    - Context ignored
    - Individual users ignored

IR Concepts

- Models
  - Boolean model: retrieve all documents that contain the query terms
    - rank documents according to some term-weighting scheme
  - Term-vector model: docs and queries are vectors in the term space
    - rank documents according to the cosine similarity
  - Term weights
    - tf × idf : (tf = term frequency, idf = log of inverse document frequency – promote rare terms)
- Measures
  - Precision: percentage of relevant documents over the returned documents
  - Recall: percentage of relevant documents over all existing relevant documents

IR Concepts - Boolean Model

- Boolean model: Data is represented as a 0/1 matrix
- Query: a boolean expression
  - the ∧ world ∧ war
  - the ∧ (world ∨ civil) ∧war
- Return all the results that match the query
  - docs D_1 and D_2
- How are the documents ranked?

D_1

… … … … …
…the civil war
… world … …

D_2

… … … … …
the world war
… … civil …

D_3

… … … … …
… the war …
… …. …. ….

	the	civil	world	war
D_1	1	1	1	1
D_2	1	1	1	1
D_3	1	0	0	1

IR Concepts - Term weighting

- Assess the importance w_{ij} of term i in a document j
- tf_{ij} = term frequency
  - frequency of term i in document j

D_1	D_2	D_3
… … … … … …the civil war … world … …	… … … … … the world war … … civil …	… … … … … … the war … … …. …. ….

	the	civil	world	war
D_1	1	1	1	1
D_2	1	1	1	1
D_3	1	0	0	1

IR Concepts – Term weighting

- Assess the importance w_{ij} of term i in a document j
- tf_{ij} = term frequency
  - frequency of term i in document j

D_1

… … … … …
…the civil war
… world … …

D_2

… … … … …
the world war
… … civil …

D_3

… … … … …
… the war …
… …. …. ….

	the	civil	world	war
D_1	100	20	5	25
D_2	200	20	50	40
D_3	150	0	0	50

IR Concepts – Term weighting

- Assess the importance w_{ij} of term i in a document j
- tf_{ij} = term frequency
  - frequency of term i in document j
  - normalized by max

D_1

… … … … …
…the civil war
… world … …

D_2

… … … … …
the world war
… … civil …

D_3

… … … … …
… the war …
… …. …. ….

	the	civil	world	war
D_1	1	0.20	0.05	0.25
D_2	1	0.10	0.25	0.20
D_3	1	0	0	0.33

IR Concepts – Term weighting

- Assess the importance w_{ij} of term i in a document j
- tf_{ij} = term frequency
- not all words are interesting
  - df_i = document frequency of term i

D_1

… … … … …
…the civil war
… world … …

D_2

… … … … …
the world war
… … civil …

D_3

… … … … …
… the war …
… …. …. ….

	the	civil	world	war
D_1	1	0.20	0.05	0.25
D_2	1	0.10	0.25	0.20
D_3	1	0	0	0.33

df	1	0.66	0.66	1

IR Concepts – Term weighting

- Assess the importance w_{ij} of term i in a document j
- tf_{ij} = term frequency
- not all words are interesting
  - df_i = document frequency of term i
  - idf_i = inverse document frequency
    - $idf_i = \log(1/df_i)$

D_1

… … … … …
…the civil war
… world … …

D_2

… … … … …
the world war
… … civil …

D_3

… … … … …
… the war …
… …. …. ….

	the	civil	world	war
D_1	1	0.20	0.05	0.25
D_2	1	0.10	0.25	0.20
D_3	1	0	0	0.33

idf	0	0.17	0.17	0

IR Concepts – Term weighting

- Assess the importance w_{ij} of term i in a document j
- tf_{ij} = term frequency
- idf_i = inverse document frequency
- $w_{ij} = tf_{ij} \times idf_i$

D_1	D_2	D_3
… … … … … …the civil war … world … …	… … … … … the world war … … civil …	… … … … … … the war … … …. …. ….

	the	civil	world	war
D_1	0	0.034	0.008	0
D_2	0	0.017	0.042	0
D_3	0	0	0	0

IR Concepts – Term weighting

- Assess the importance w_{ij} of term i in a document j
- tf_{ij} = term frequency
- idf_i = inverse document frequency
- $w_{ij} = tf_{ij} \times idf_i$
- Query: "the civil war"
  - document D_1 is more important

D_1

… … … … …
…the civil war
… world … …

D_2

… … … … …
the world war
… … civil …

D_3

… … … … …
… the war …
… …. …. ….

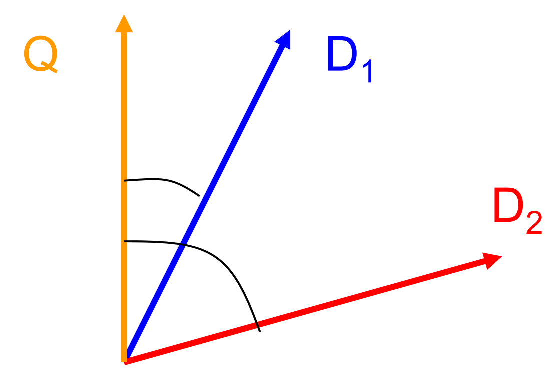
	the	civil	world	war
D_1	0	0.034	0.008	0
D_2	0	0.017	0.042	0
D_3	0	0	0	0

IR Concepts – Vector model

- Documents are vectors in the term space (weighted by w_{ij}), normalized on the unit sphere
- Query: "the civil war"
  - Q is a mini document - vector
- Similarity of Q and D is the cosine of the angle between Q and D
  - returns a set of ranked results

	the	civil	world	war
D_1	0	0.97	0.22	0
D_2	0	0.37	0.92	0
D_3	0	0	0	0

Q	0	1	1	0



IR Concepts – Measures

- There are A relevant documents to the query in our dataset.
- Our algorithm returns D documents.
- How good is it?

- Precision: Fraction of returned documents that are relevant

$$P = \frac{|D \cap A|}{D}$$

- Recall: Fraction of all relevant documents that are returned

$$R = \frac{|D \cap A|}{A}$$

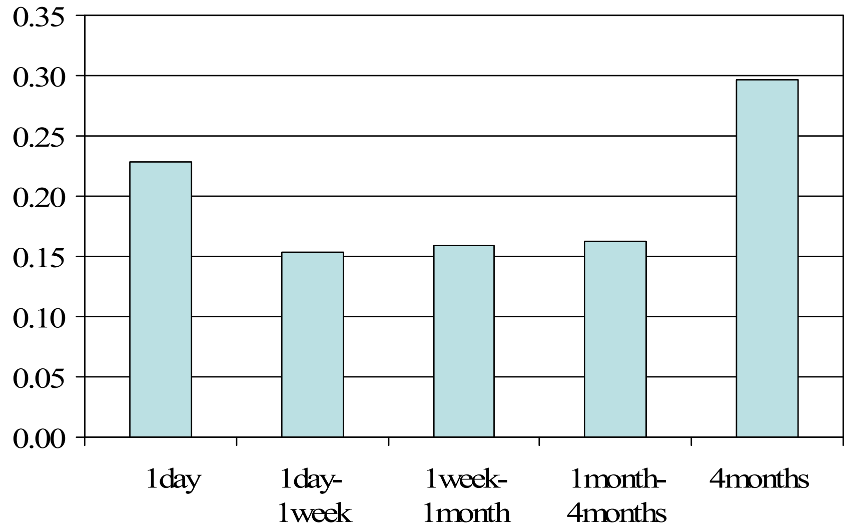
Web users

- They ask a lot but they offer little in return
  - Make ill-defined queries
    - short (2.5 avg terms, 80% <3 terms – AV, 2001)
    - imprecise terms
    - poor syntax
    - low effort
  - Unpredictable
    - wide variance in needs/expectations/expertise
  - Impatient
    - 85% look one screen only (mostly “above the fold”)
    - 78% queries not modified (one query per session)
- …but they know how to spot correct information
  - follow “the scent of information”…

Web corpus

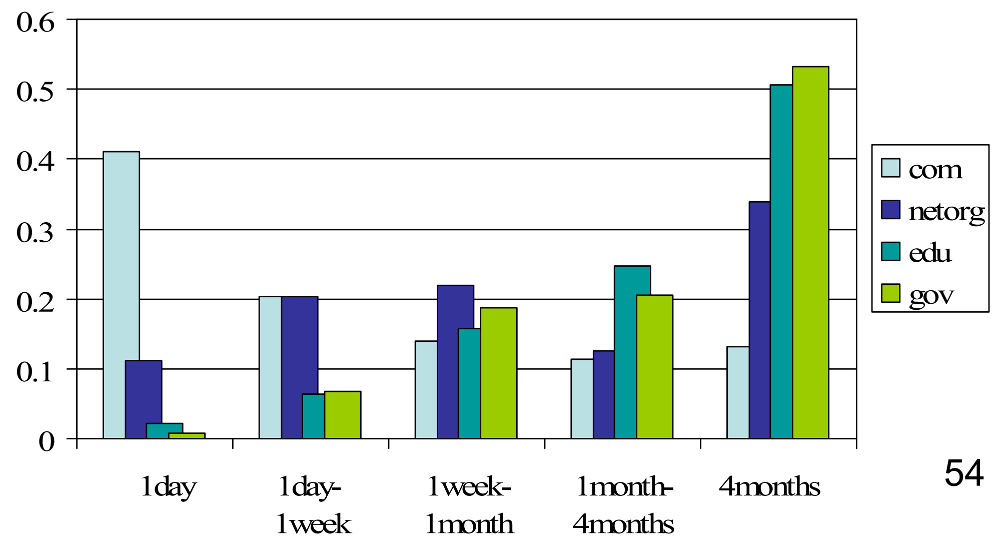
- Immense amount of information
  - 2008, Google: 100(!) Billion pages,
  - fast growth rate (double every 8-12 months)
  - Huge Lexicon: 10s-100s millions of words
- Highly diverse content
  - many different authors, languages, encodings
  - different media (text, images, video)
  - highly un-structured content
- Static + Dynamic (“the hidden Web”)
- Volatile
  - crawling challenge

Rate of change [CGM00]

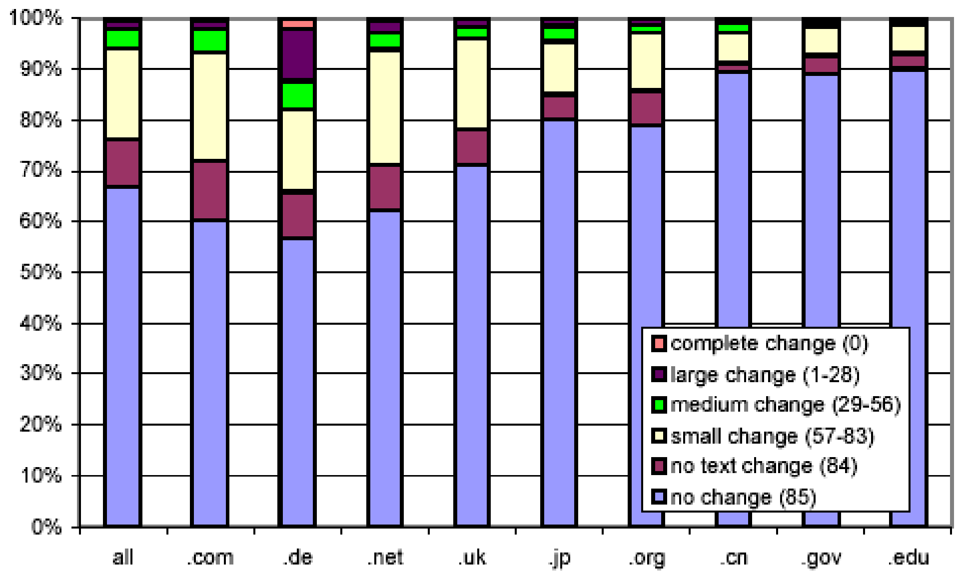


average rate of change

average rate of change
per domain

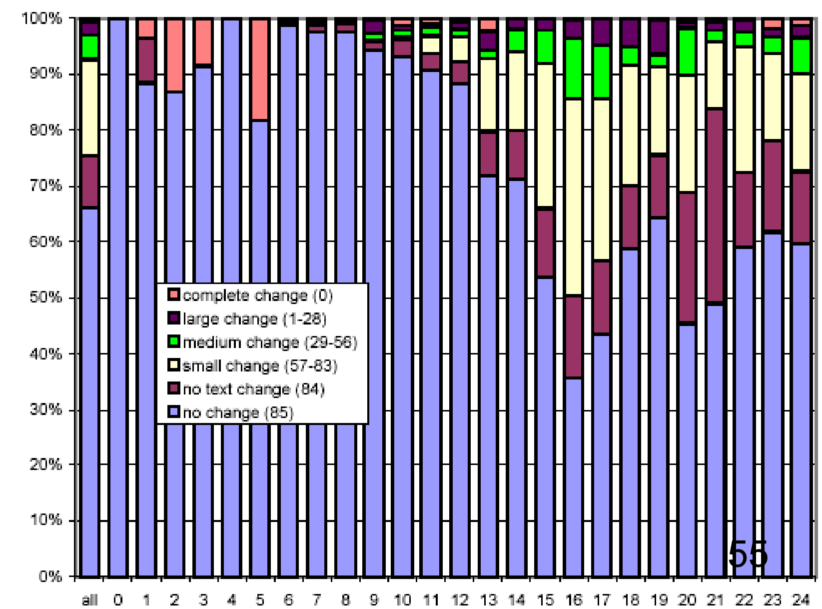


Rate of Change [FMNW03]



Rate of change per domain. Change between two successive downloads

Rate of change as a function of document length



Other corpus characteristics

- Links, graph topology, anchor text
  - this is now part of the corpus!
- Significant amount of duplication
  - ~30% (near) duplicates [FMN03]
- Spam!
  - 100s of million of pages
  - Add-URL robots

Query Results

- Static documents
  - text, images, audio, video,etc
- Dynamic documents ("the invisible Web")
  - dynamic generated documents, mostly database accesses
- Extracts of documents, combinations of multiple sources
  - www.googlism.com

The evolution of Search Engines

- First Generation – text data only
  - word frequencies, tf × idf
- Second Generation – text and web data
  - Link analysis
  - Click stream analysis
  - Anchor Text
- Third Generation – the need behind the query
  - Semantic analysis: what is it about?
  - Integration of multiple sources
  - Context sensitive
    - personalization, geographical context, browsing context

1995-1997: AltaVista Lycos, Excite

1998 - now : Google leads the way

Still experimental

First generation Web search

- Classical IR techniques
  - Boolean model
  - ranking using tf × idf relevance scores
- good for informational queries
- quality degraded as the web grew
- sensitive to spamming

Second generation Web search

- Boolean model
- Ranking using web specific data
  - HTML tag information
  - click stream information (Direct Hit)
    - people vote with their clicks
  - directory information (Yahoo! directory)
  - anchor text
  - link analysis

Link Analysis Ranking

- Intuition: a link from q to p denotes endorsement
  - people vote with their links
- Popularity count
  - rank according to the incoming links
- PageRank algorithm
  - perform a random walk on the Web graph. The pages visited most often are the ones most important. (n: total number of pages, F(q): number of outbound links on page q, a: damping factor)

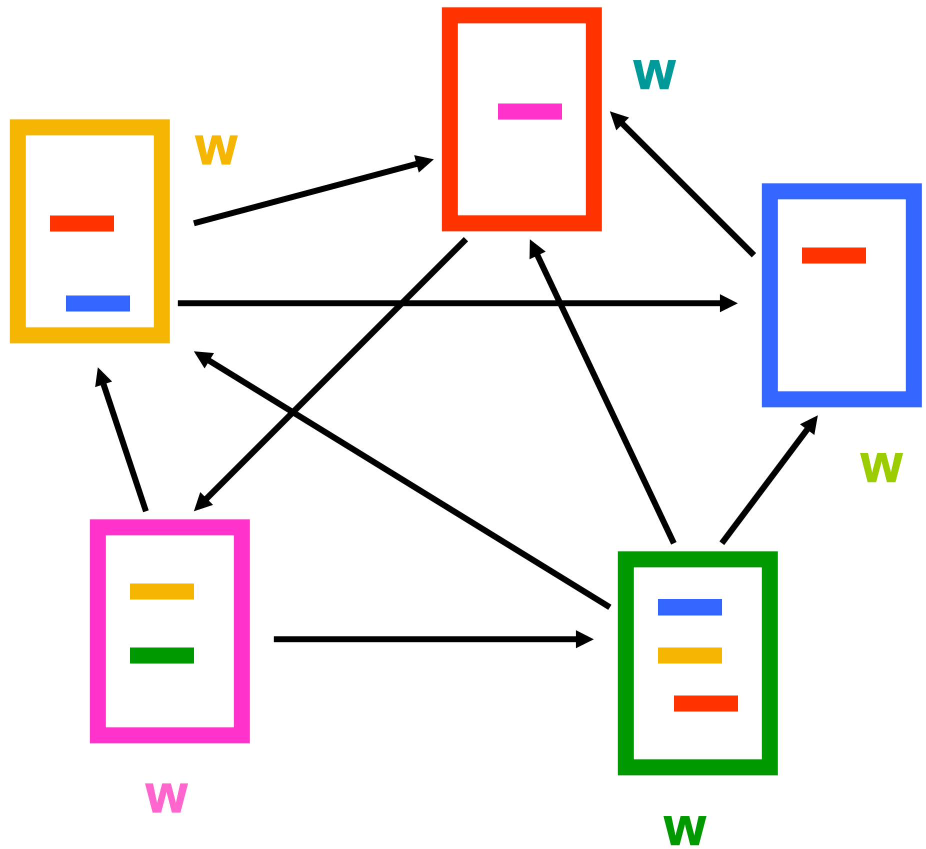
$$PR(p) = \alpha \sum_{q \to p} \frac{PR(q)}{|F(q)|} + (1 - \alpha)\frac{1}{n}$$

Link Analysis: Intuition

- A link from page p to page q denotes endorsement
  - page p considers page q an authority on a subject
  - mine the web graph of recommendations
  - assign an authority value to every page

Link Analysis Ranking (LAR) Algorithms

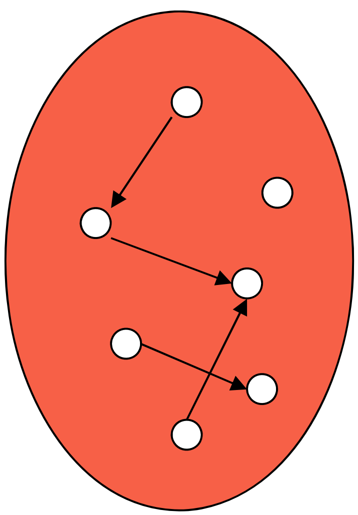
- Start with a collection of web pages
- Extract the underlying hyperlink graph
- Run the LAR algorithm on the graph
- Output: an authority weight for each node



Algorithm input

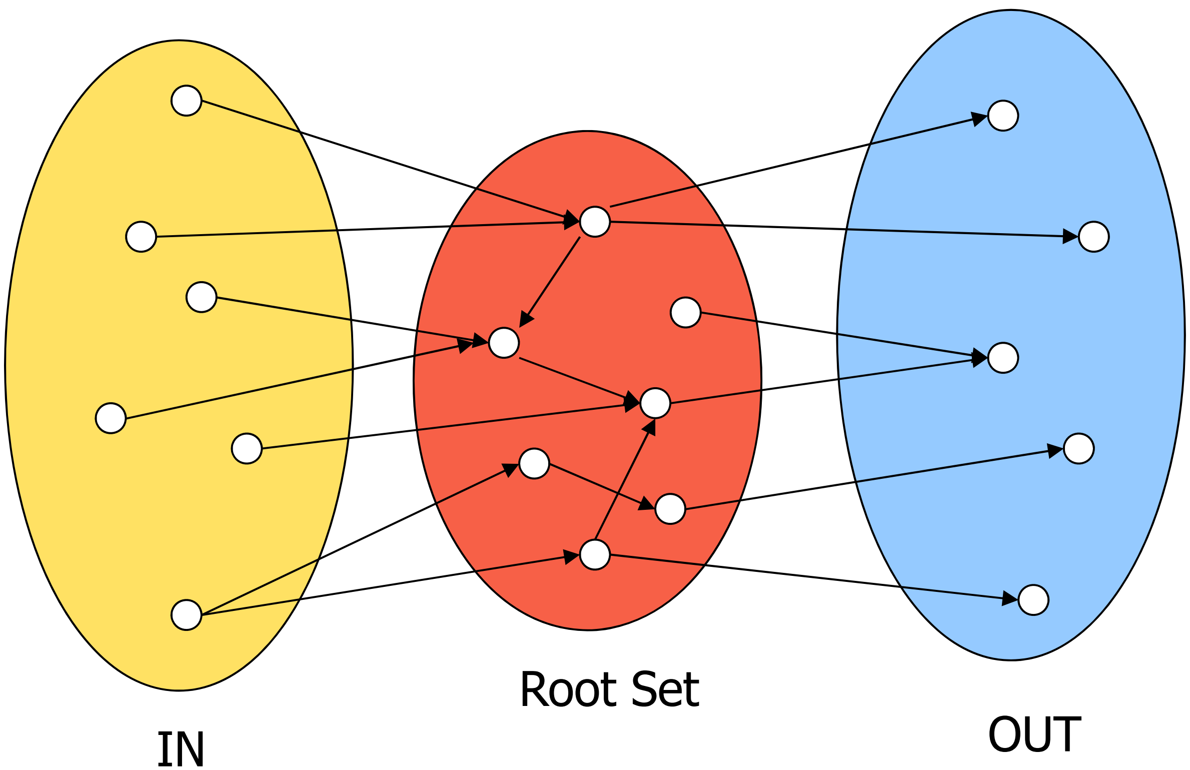
- Query independent: rank the whole Web
  - PageRank (Brin and Page 98) was proposed as query independent
- Query dependent: rank a small subset of pages related to a specific query
  - Hyperlink-Induced Topic Search (HITS) was proposed as query dependent (Kleinberg 98)

Query dependent input

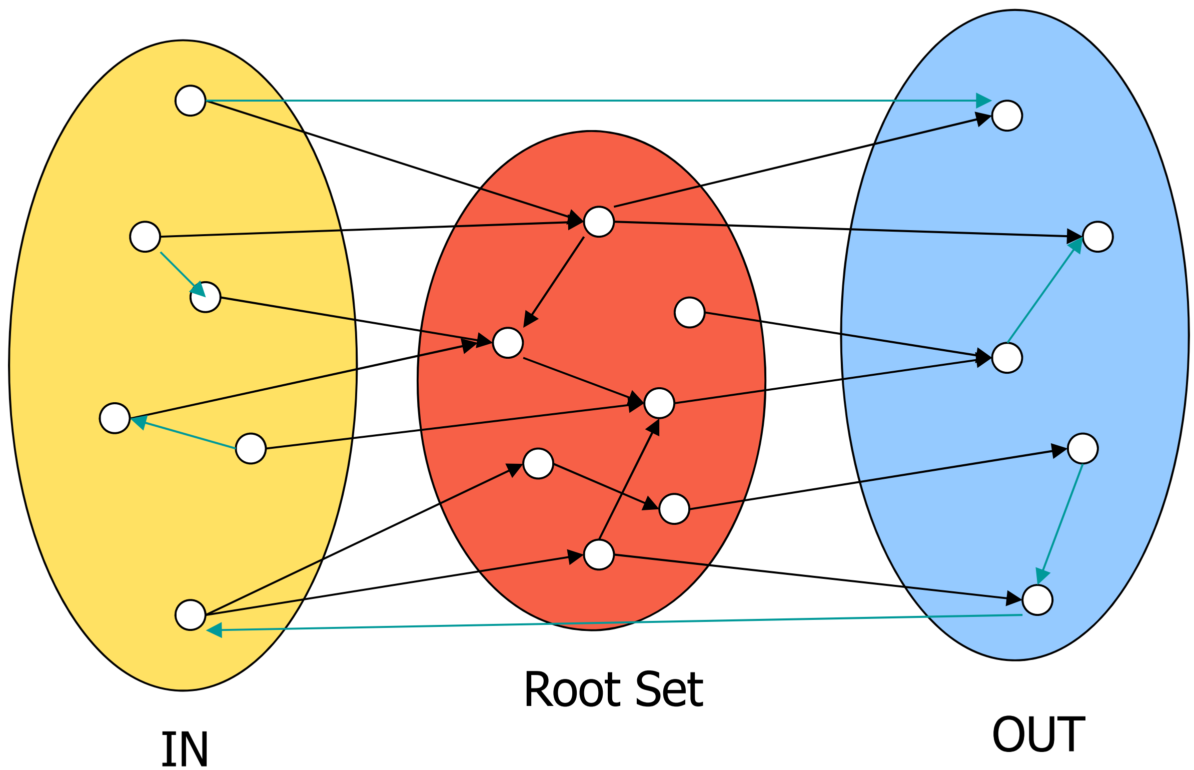


Root Set

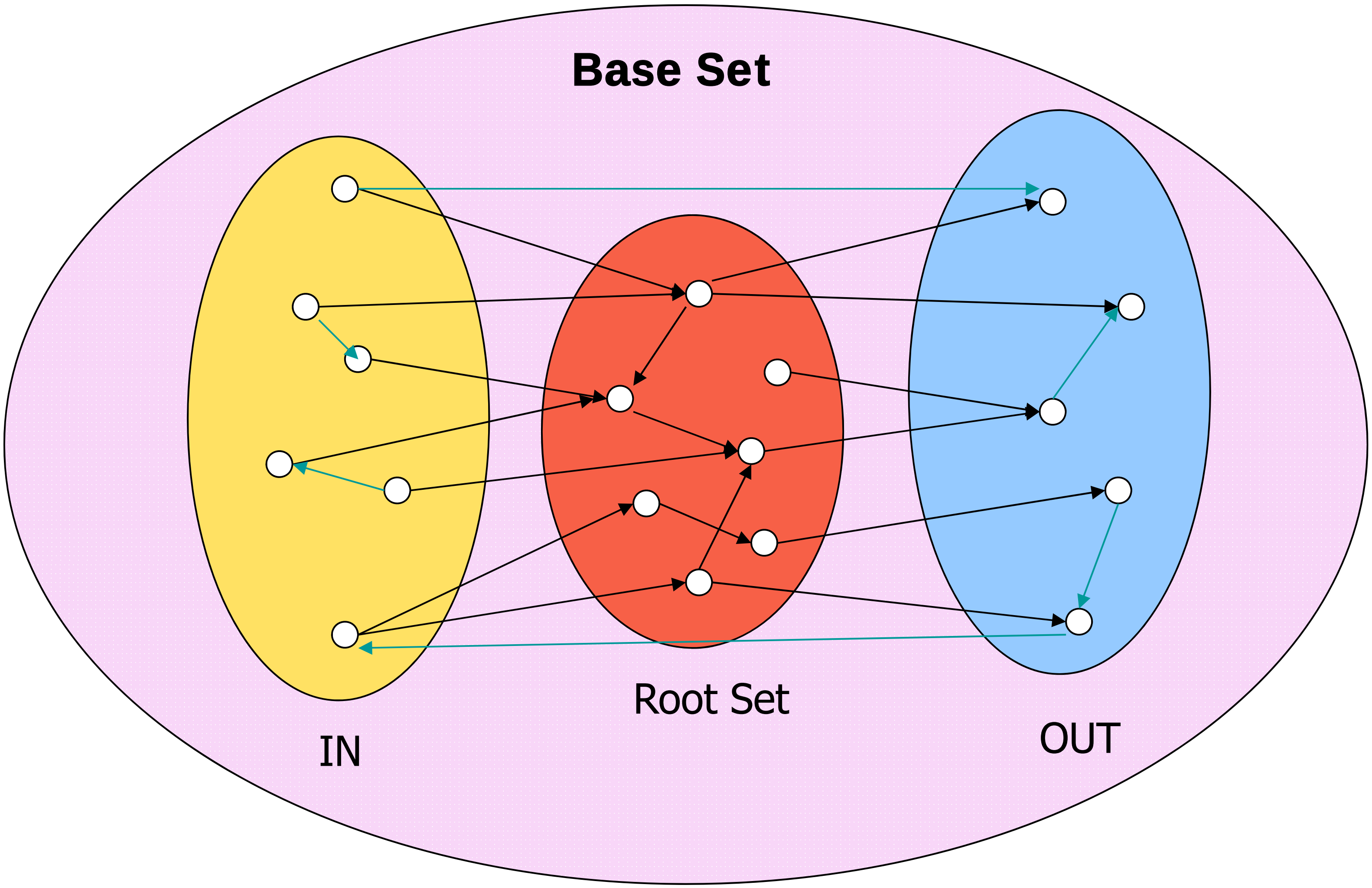
Query dependent input



Query dependent input



Query dependent input



Social network analysis

- Evaluate the centrality of individuals in social networks
  - degree centrality
    - the (weighted) degree of a node
  - Closeness centrality (distance centrality)
    - the average (weighted) distance of a node to the rest in the graph
      $$D_c(v) = \frac{1}{\sum_{u \neq v} d(v,u)}$$
  - betweenness centrality
    - the average number of (weighted) shortest paths that use node v
      $$B_c(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Random walks on undirected graphs

- In the stationary distribution of a random walk on an undirected graph, the probability of being at node i is proportional to the (weighted) degree of the vertex
- Random walks on undirected graphs are not "interesting" as compared to directed graphs!

Counting paths – Katz 53

- The importance of a node is measured by the weighted sum of paths that lead to this node
- $A^m[i,j]$ = number of paths of length m from i to j
- Compute

$$P = bA + b^2A^2 + \cdots + b^mA^m + \cdots = (I - bA)^{-1} - I$$

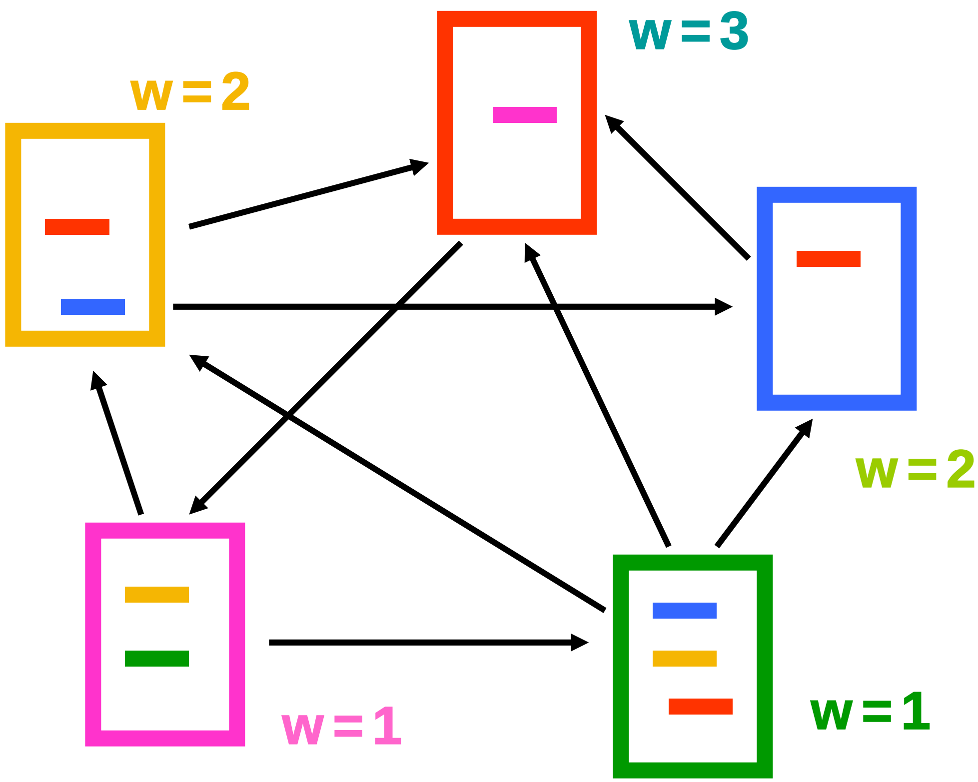
- converges when $b < \lambda_1(A)$
- Rank nodes according to the column sums of the matrix P

Bibliometrics

- Impact factor (E. Garfield 72)
  - counts the number of citations received for papers of the journal in the previous two years
- Pinsky-Narin 76
  - perform a random walk on the set of journals
  - P_{ij} = the fraction of citations from journal i that are directed to journal j

InDegree algorithm

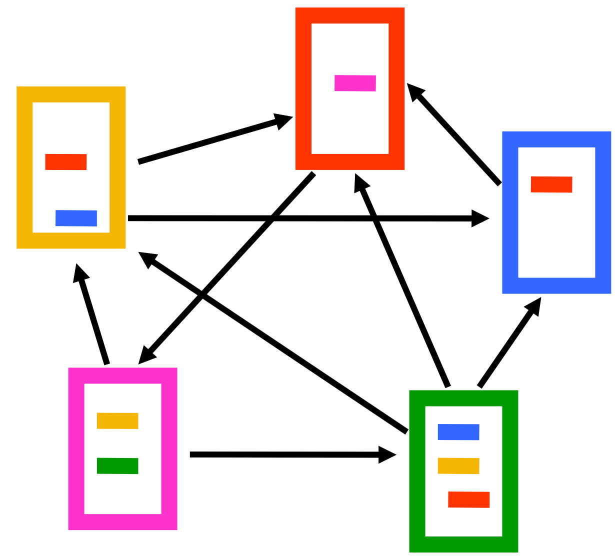
- Rank pages according to in-degree
  - $w_i = |B(i)|$



1. Red Page
2. Yellow Page
3. Blue Page
4. Purple Page
5. Green Page

PageRank algorithm [BP98]

- Good authorities should be pointed by good authorities
- Random walk on the web graph
  - pick a page at random
  - with probability $1-\alpha$ jump to a random page
  - with probability α follow a random outgoing link
- Rank according to the stationary distribution
- $$PR(p) = \alpha \sum_{q \to p} \frac{PR(q)}{|F(q)|} + (1-\alpha)\frac{1}{n}$$



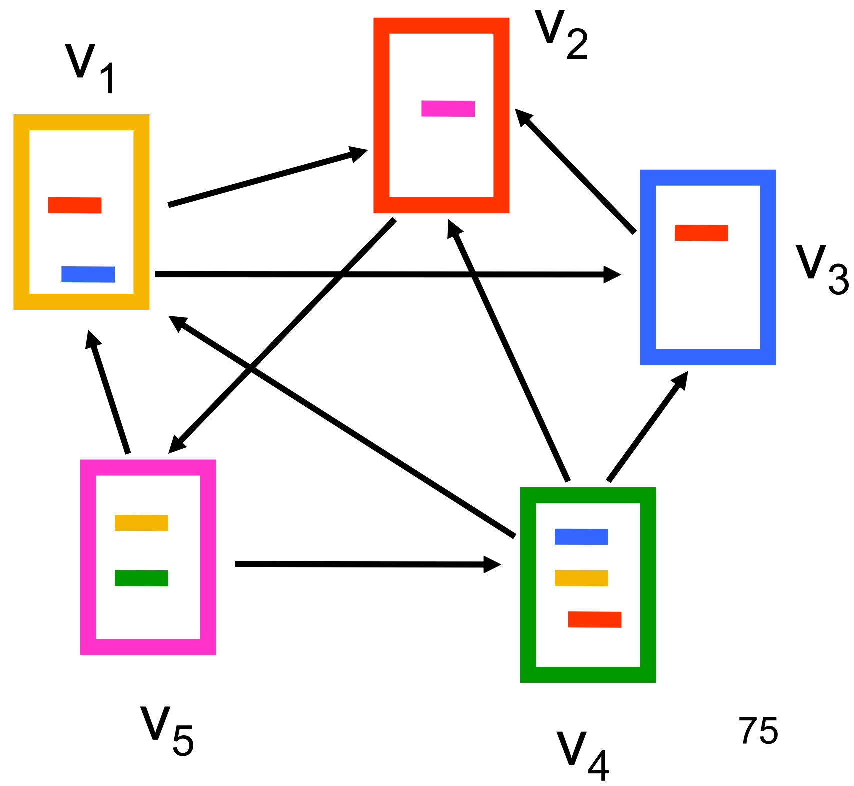
1. Red Page
2. Purple Page
3. Yellow Page
4. Blue Page
5. Green Page

Random walks

- Random walks on graphs correspond to Markov Chains
  - The set of states S is the set of nodes of the graph G
  - The transition probability matrix P is the probability that we follow an edge from one node to another

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

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix}$$



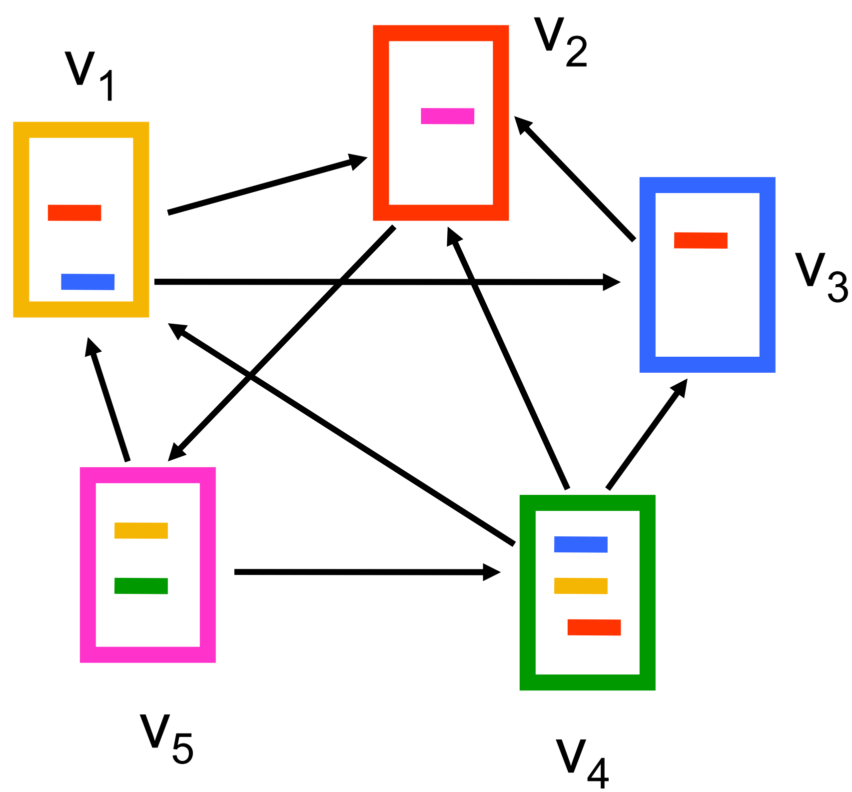
State probability vector

- The vector $q^t = (q^t_1, q^t_2, \ldots, q^t_n)$ that stores the probability of being at state i at time t
  - q^0_i = the probability of starting from state i

$$q^t = q^{t-1} P$$

An example

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$



$q^{t+1}_1 = 1/3\ q^t_4 + 1/2\ q^t_5$

$q^{t+1}_2 = 1/2\ q^t_1 + q^t_3 + 1/3\ q^t_4$

$q^{t+1}_3 = 1/2\ q^t_1 + 1/3\ q^t_4$

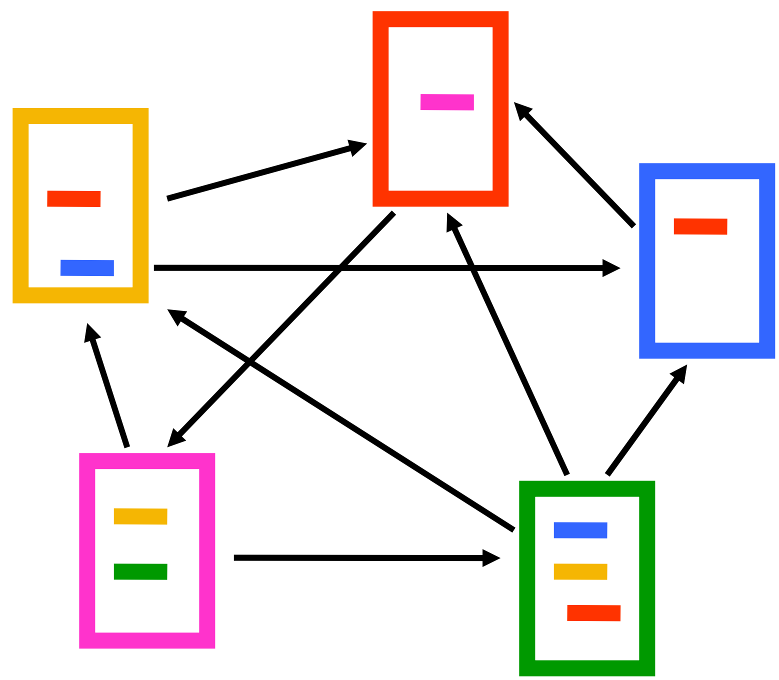
$q^{t+1}_4 = 1/2\ q^t_5$

$q^{t+1}_5 = q^t_2$

The PageRank random walk

- Vanilla random walk
  - make the adjacency matrix **stochastic** and run a random walk

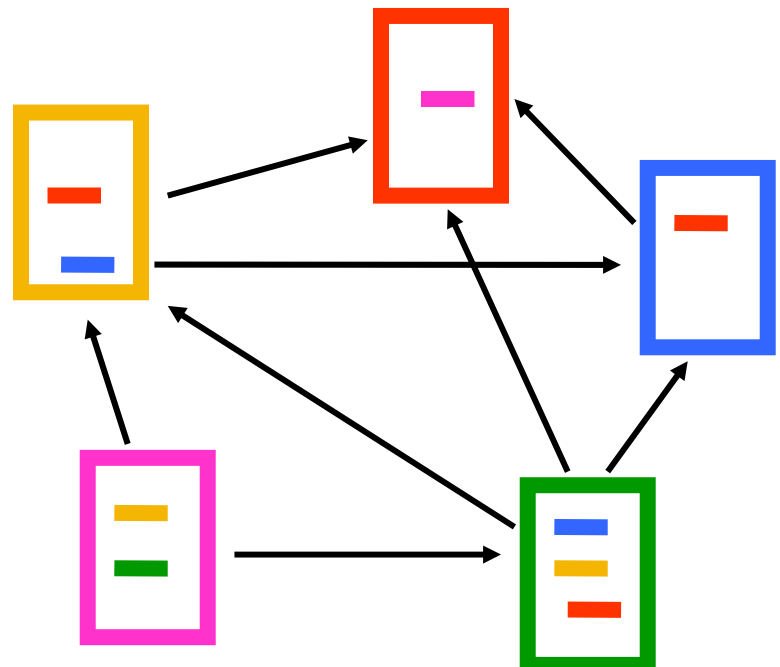
$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$



The PageRank random walk

- What about sink nodes?
  - what happens when the random walk moves to a node without any outgoing inks?

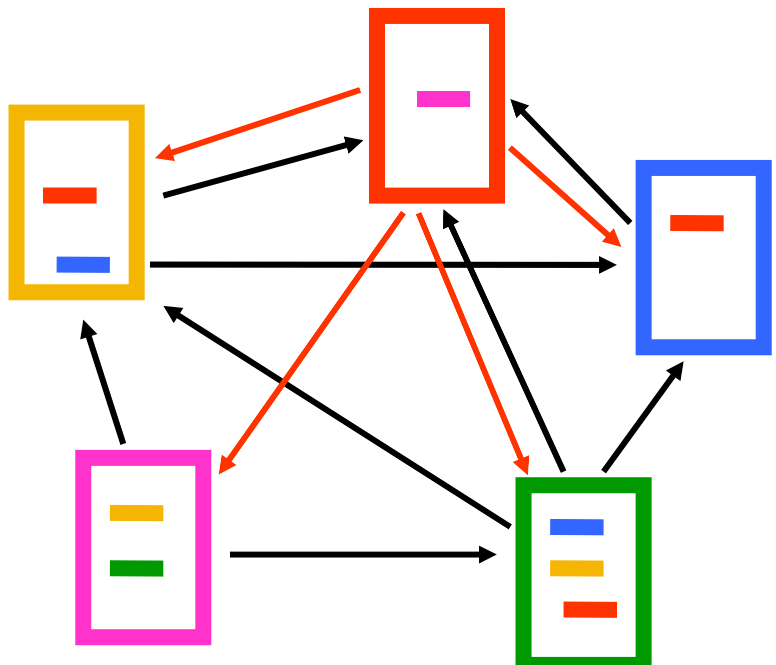
$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ \boxed{0 \quad 0 \quad 0 \quad 0 \quad 0} \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$



The PageRank random walk

- Replace these row vectors with a vector v
  - typically, the uniform vector

$$P' = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$



$P' = P + dv^T$

$$d = \begin{cases} 1 & \text{if i is sink} \\ 0 & \text{otherwise} \end{cases}$$

The PageRank random walk

- How do we guarantee irreducibility (a **directed graph** is **irreducible** if, given any two vertices, there exists a path from the first vertex to the second)?
  - add a random jump to vector v with probability α
    - typically, to a uniform vector

$$
P'' = \alpha \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix} + (1-\alpha) \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{bmatrix}
$$

$P'' = \alpha P' + (1-\alpha)uv^T$, where u is the vector of all 1s

A PageRank algorithm

- Performing Vanilla power method is now too expensive – the matrix is not sparse

$q^0 = v$
$t = 1$
repeat
$$q^t = (P'')^T q^{t-1}$$
$$\delta = \|q^t - q^{t-1}\|$$
$t = t + 1$
until $\delta < \varepsilon$

Efficient computation of $y = (P'')^T x$

$$y = \alpha P^T x$$
$$\beta = \|x\|_1 - \|y\|_1$$
$$y = y + \beta v$$

P = normalized adjacency matrix

$P' = P + dv^T$, where d_i is 1 if i is sink and 0 o.w.

$P'' = \alpha P' + (1-\alpha)uv^T$, where u is the vector of all 1s

Prestige

- adjacency matrix A
  - if document cites document A(u,v) = 1
  - Otherwise A(u,v) = 0
- prestige score

$$p(u) = \sum_v A(v,u)\, p(v)$$

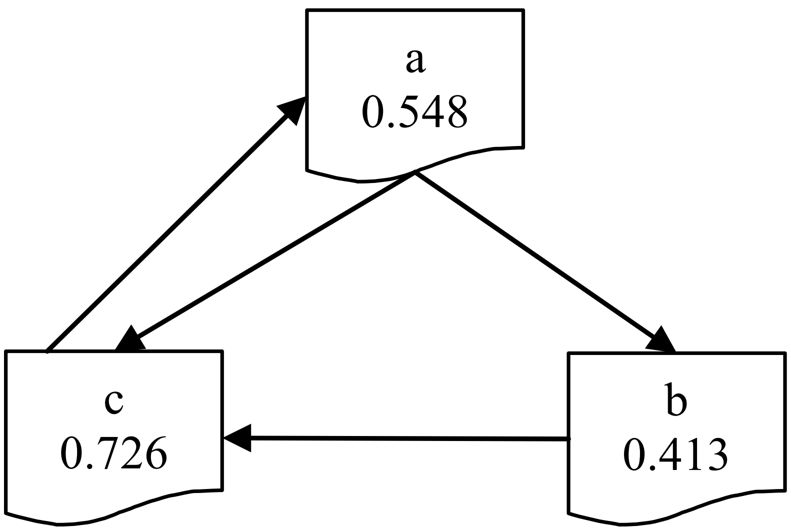
- Computing prestige

$$P' = A^T P$$

- Eigen decomposition

$$\lambda P = A^T P$$

Example



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

$$\lambda P = A^T P$$

$$\lambda = 1.325$$

$$P = \begin{pmatrix} 0.548 & 0.414 & 0.726 \end{pmatrix}^T$$

$$1.325 \begin{pmatrix} 0.548 \\ 0.414 \\ 0.726 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0.548 \\ 0.414 \\ 0.726 \end{pmatrix}$$

And

Power Iteration

- $P \leftarrow P_0$
- *Loop:*

$$Q \leftarrow P$$

$$P \leftarrow A^T Q$$

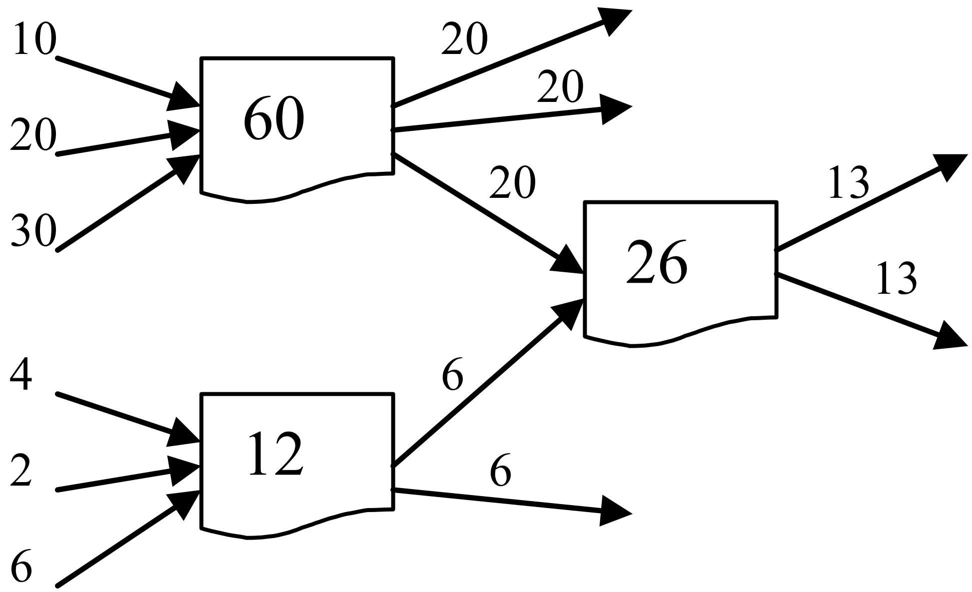
$$P \leftarrow \frac{1}{\|P\|} P$$

- *While* $\|P - Q\| > \varepsilon$

PageRank once more

- "Random web surfer" keeps clicking on hyperlinks at random with uniform probability
- Implements *random walk* on the web graph
- Page *u* links to N_u web pages
- Probability of visiting page *v* will be $1/N_u$
- Amount of prestige that page *v* receives from page *u* is $1/N_u$ of the prestige of *u*

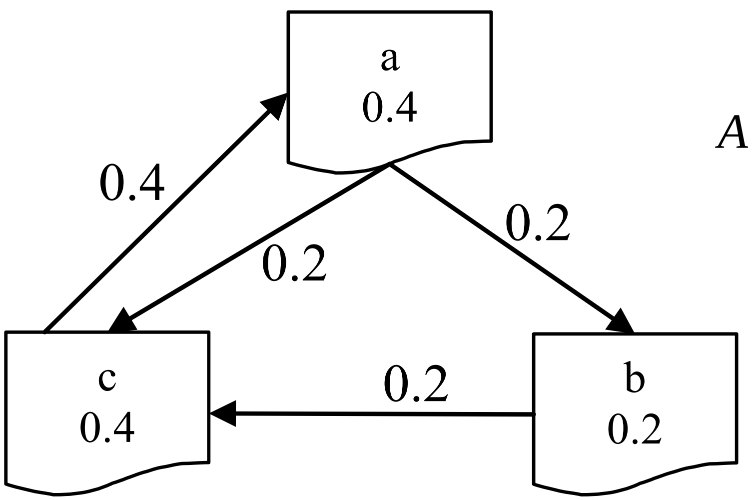
Propagation of PageRank R(u)



$$R(u) = \lambda \sum_{v} \frac{A(v,u)R(v)}{N_v}$$

$$N_v = \sum_w A(v,w)$$

Calculation of PageRank



$$A = \begin{pmatrix} 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\lambda P = A^T P$$

$$\begin{pmatrix} 0.4 \\ 0.2 \\ 0.4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0.5 & 0 & 0 \\ 0.5 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0.4 \\ 0.2 \\ 0.4 \end{pmatrix}$$

Norm $\|X\|_1 = x_1 + x_2 + \ldots + x_n$

$$P^T = \begin{pmatrix} 0.666 & 0.333 & 0.666 \end{pmatrix}$$

Norm $\|X\|_2 = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$

$$P^T = \begin{pmatrix} 2 & 1 & 2 \end{pmatrix}$$

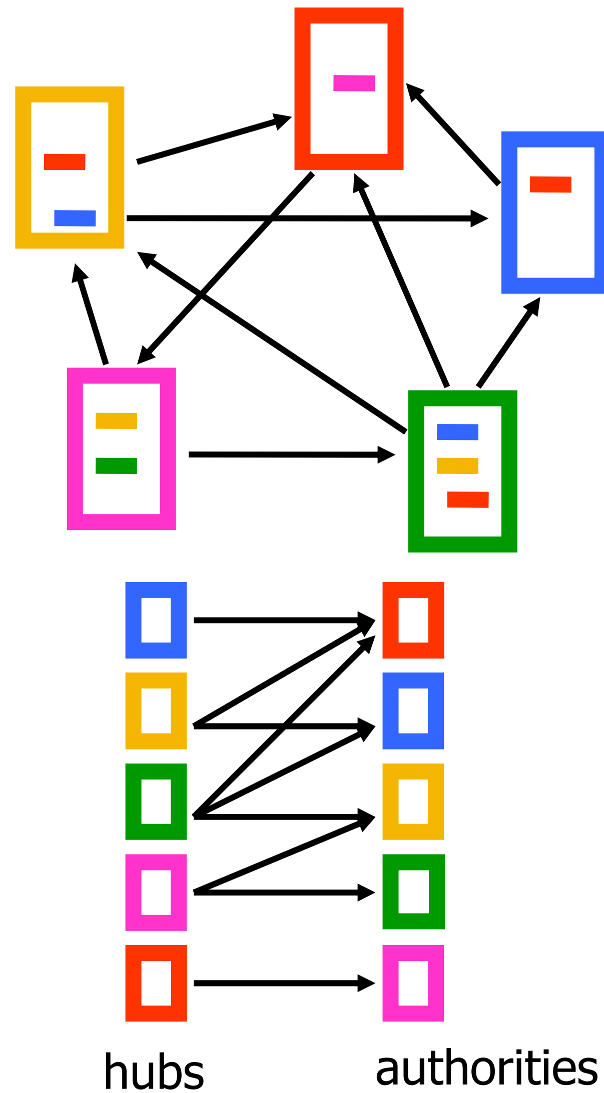
Integers

Research on PageRank

- Specialized PageRank
  - personalization [BP98]
    - instead of picking a node uniformly at random favor specific nodes that are related to the user
  - topic sensitive PageRank [H02]
    - compute many PageRank vectors, one for each topic
    - estimate relevance of query with each topic
    - produce final PageRank as a weighted combination
- Updating PageRank [Chien et al 2002]
- Fast computation of PageRank
  - numerical analysis tricks
  - node aggregation techniques
  - dealing with the "Web frontier"
  - ….

Hubs and Authorities [K98]

- Authority is not necessarily transferred directly between authorities
- Pages have double identity
  - hub identity
  - authority identity
- Good hubs point to good authorities
- Good authorities are pointed by good hubs



HITS Algorithm (Klienberg 1998)

- Initialize all weights to 1.
- Repeat until convergence
  - O operation: hubs collect the weight of the authorities

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

  - I operation: authorities collect the weight of the hubs

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

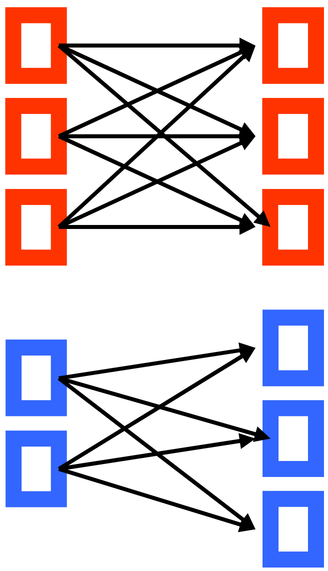
  - Normalize weights under some norm

HITS and eigenvectors

- The HITS algorithm is a power-method eigenvector computation
  - in vector terms $a^t = A^T h^{t-1}$ and $h^t = A a^{t-1}$
  - so $a = A^T A a^{t-1}$ and $h^t = A A^T h^{t-1}$
  - The authority weight vector a is the eigenvector of $A^T A$ and the hub weight vector h is the eigenvector of $A A^T$
  - Why do we need normalization?
- The vectors a and h are singular vectors of the matrix A

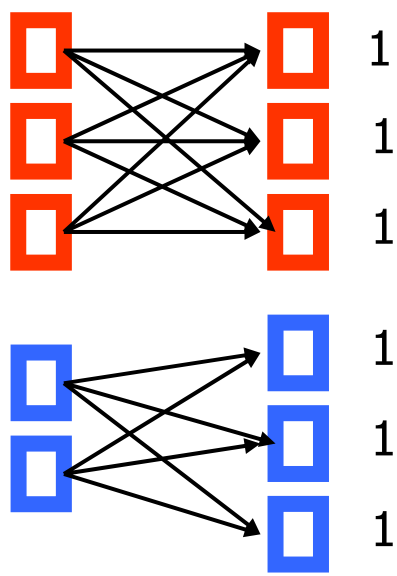
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



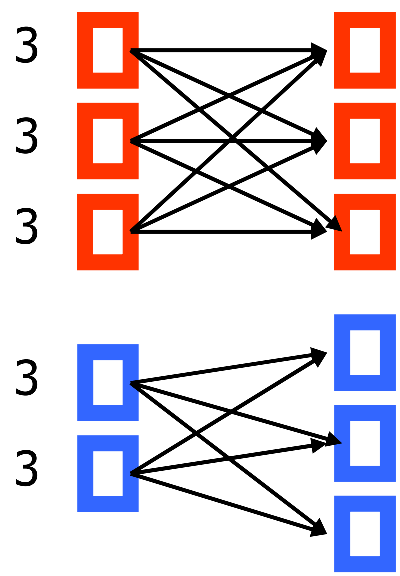
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



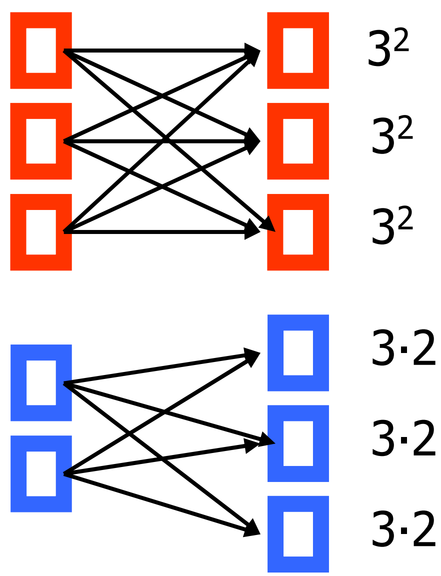
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



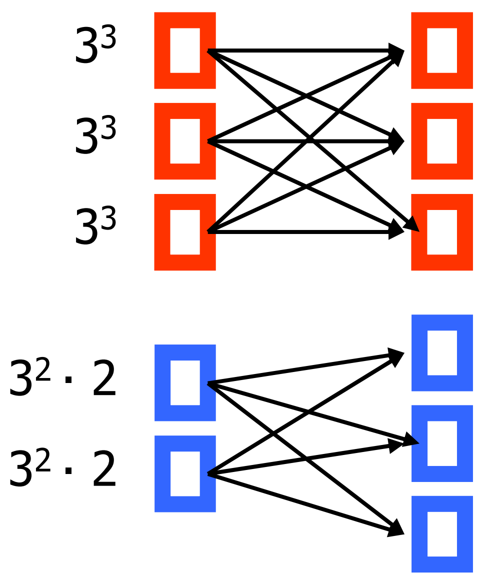
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



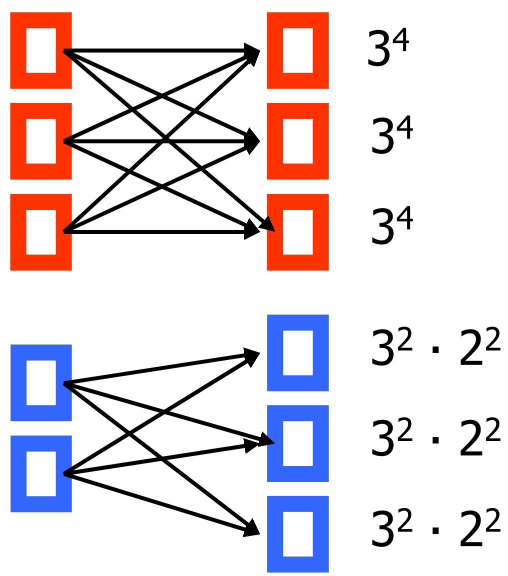
HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



HITS and the TKC effect

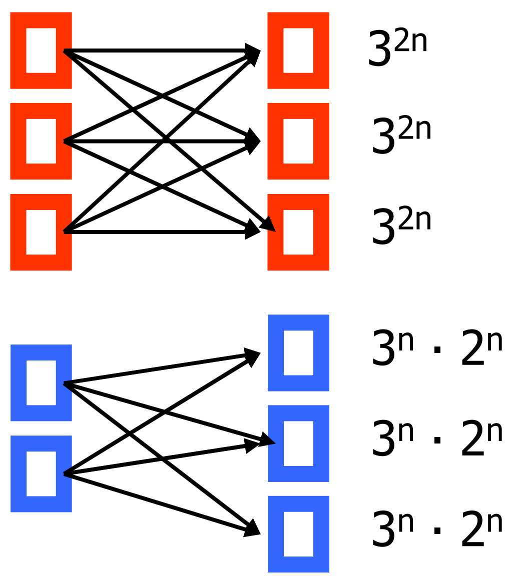
- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



HITS and the TKC effect

- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect

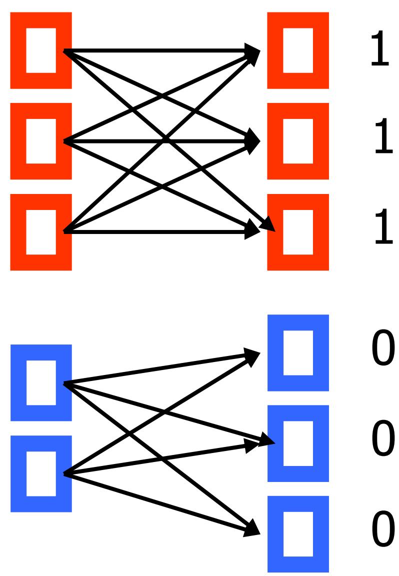
weight of node p is proportional to the number of $(BF)^n$ paths that leave node p



after n iterations

HITS and the TKC effect

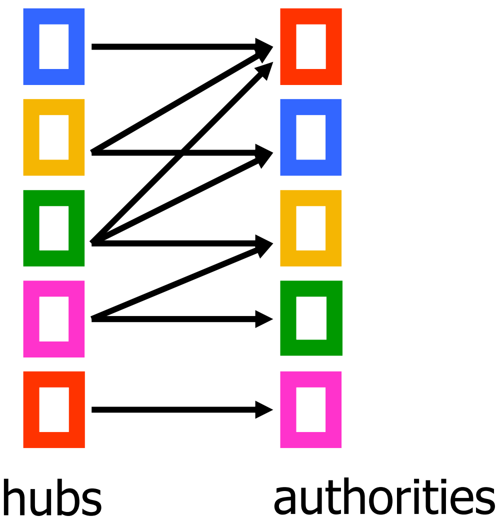
- The HITS algorithm favors the most dense community of hubs and authorities
  - Tightly Knit Community (TKC) effect



after normalization with the max element as $n \rightarrow \infty$

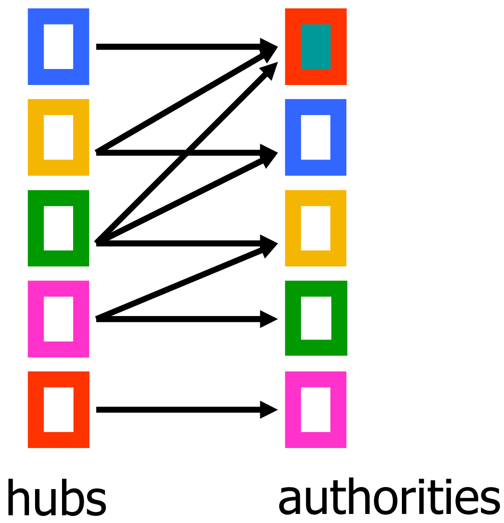
The SALSA algorithm [LM00]

- Perform a random walk alternating between hubs and authorities
- SALSA: Scoring ALgorithm for Spectral Analysis



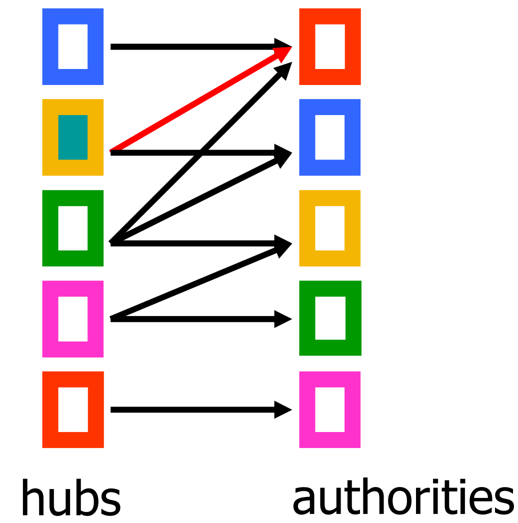
The SALSA algorithm [LM00]

- Start from an authority chosen uniformly at random
  - e.g. the red authority



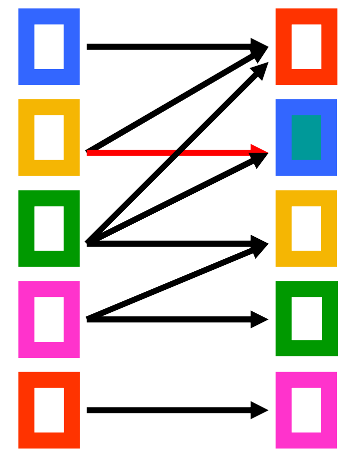
The SALSA algorithm [LM00]

- Start from an authority chosen uniformly at random
  - e.g. the red authority
- Choose one of the in-coming links uniformly at random and move to a hub
  - e.g. move to the yellow authority with probability 1/3



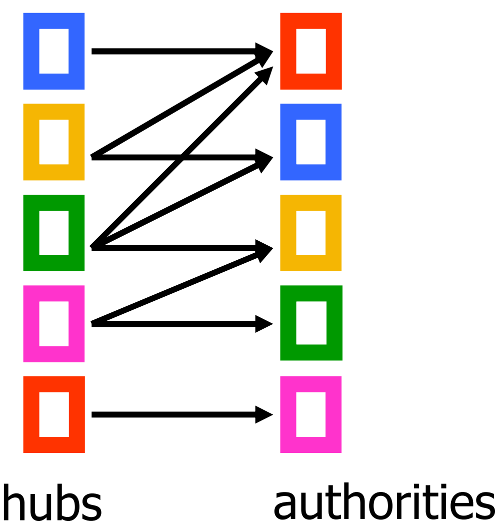
The SALSA algorithm [LM00]

- Start from an authority chosen uniformly at random
  - e.g. the red authority
- Choose one of the in-coming links uniformly at random and move to a hub
  - e.g. move to the yellow authority with probability 1/3
- Choose one of the out-going links uniformly at random and move to an authority
  - e.g. move to the blue authority with probability 1/2



The SALSA algorithm [LM00]

- In matrix terms
  - A_c = the matrix A where columns are normalized to sum to 1
  - A_r = the matrix A where rows are normalized to sum to 1
  - p = the probability state vector
- The first step computes
  - $y = A_c p$
- The second step computes
  - $p = A_r^T y = A_r^T A_c p$
- Or, the transition matrix
  - $P = A_r A_c^T$

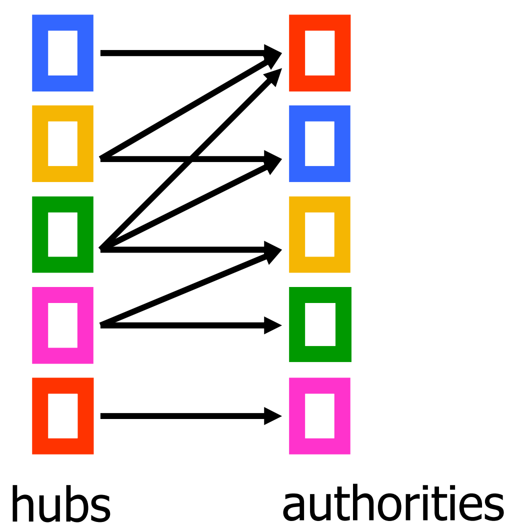


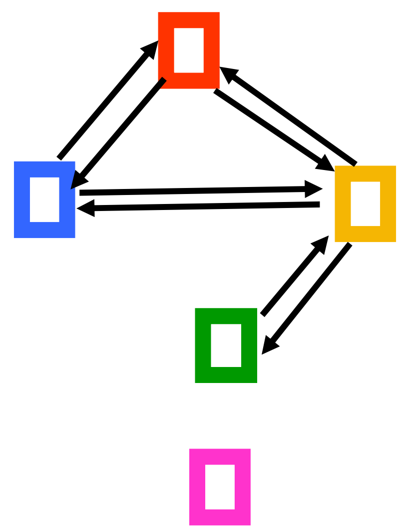
$$y_2 = 1/3\ p_1 + 1/2\ p_2$$

$$p_1 = y_1 + 1/2\ y_2 + 1/3\ y_3$$

The SALSA algorithm [LM00]

- The SALSA performs a random walk on the authority (right) part of the bipartite graph
  - There is a transition between two authorities if there is a BF path between them





$$P(i,j) = \sum_{\substack{k:k\to j \\ i\to k}} \frac{1}{\mathrm{in}(i)} \frac{1}{\mathrm{out}(k)}$$

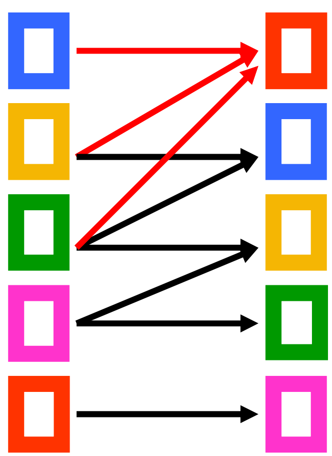
The SALSA algorithm [LM00]

- Stationary distribution of SALSA
  - authority weight of node i =

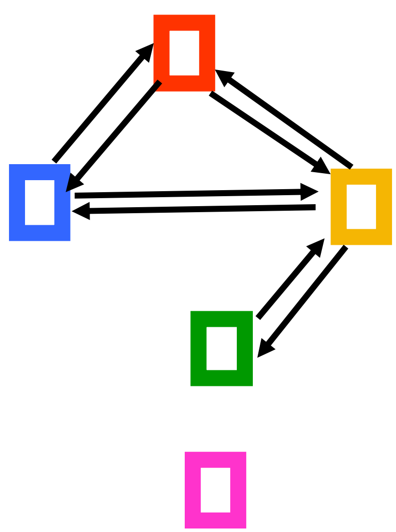
    fraction of authorities in the hub-authority community of i

    ×

    fraction of links in the community that point to node i
  - Reduces to InDegree for single community graphs



hubs authorities

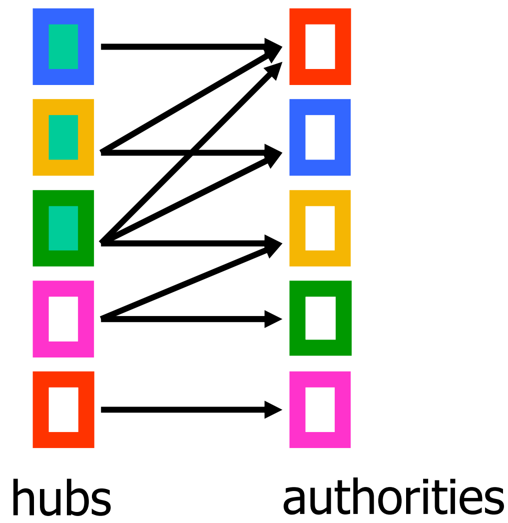


w = 4/5 × 2/8

w = 1/5 × 1

The BFS algorithm [BRRT01]

- BFS: breadth-first search
- Rank a node according to the reachability of the node
- Create the neighborhood by alternating between Back and Forward steps
- Apply exponentially decreasing weight as you move further away



Implicit properties of the HITS algorithm

- Symmetry
  - both hub and authority weights are defined in the same way (through the sum operator)
  - reversing the links, swaps values
- Equality
  - the sum operator assumes that all weights are equally important

Authority Threshold AT(k) algorithm

- Small authority weights should not contribute to the computation of the hub weights

- Repeat until convergence
  - O operation: hubs collect the k highest authority weights

$$h_i = \sum_{j:i \to j} a_j : a_j \in F_k(i)$$

  - I operation: authorities collect the weight of hubs

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

  - Normalize weights under some norm

Norm(p) algorithm

- Small authority weights should contribute less to the computation of the hub weights

- Repeat until convergence
  - O operation : hubs compute the p-norm of the authority weight vector

$$h_i = \left(\sum_{j:i \to j} a_j^{\ p} \right)^{1/p} = \left\| \overrightarrow{F(i)} \right\|_p$$

  - I operation: authorities collect the weight of hubs

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

  - Normalize weights under some norm

The MAX algorithm

- A hub is as good as the best authority it points to

- Repeat until convergence
  - O operation : hubs collect the highest authority weight

$$h_i = \max_{j:i \to j} a_j$$

  - I operation: authorities collect the weight of hubs

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

  - Normalize weights under some norm

- Special case of AT(k) (for k=1) and Norm(p) (p=∞)

Dynamical Systems

- Discrete Dynamical System: The repeated application of a function g on a set of weights

```
Initialize weights to w^0
For t=1,2,…
    w^t=g(w^(t-1))
```

- LAR algorithms: the function g propagates the weight on the graph G
- Linear vs Non-Linear dynamical systems
  - eigenvector analysis algorithms (PageRank, HITS) are linear dynamical systems
  - AT(k), Norm(p) and MAX are non-linear

Some experimental results

- 34 different queries
- user relevance feedback
  - high relevant/relevant/non-relevant
- measures of interest
  - “high relevance ratio”
  - “relevance ratio”
- Data available at

http://www.cs.toronto.edu/~tsap/experiments

Aggregate Statistics

	AVG HR	STDEV HR	AVG R	STDEV R
HITS	22%	24%	45%	39%
PageRank	24%	14%	46%	20%
In-Degree	35%	22%	58%	29%
SALSA	35%	21%	59%	28%
MAX	38%	25%	64%	32%
BFS	43%	18%	73%	19%

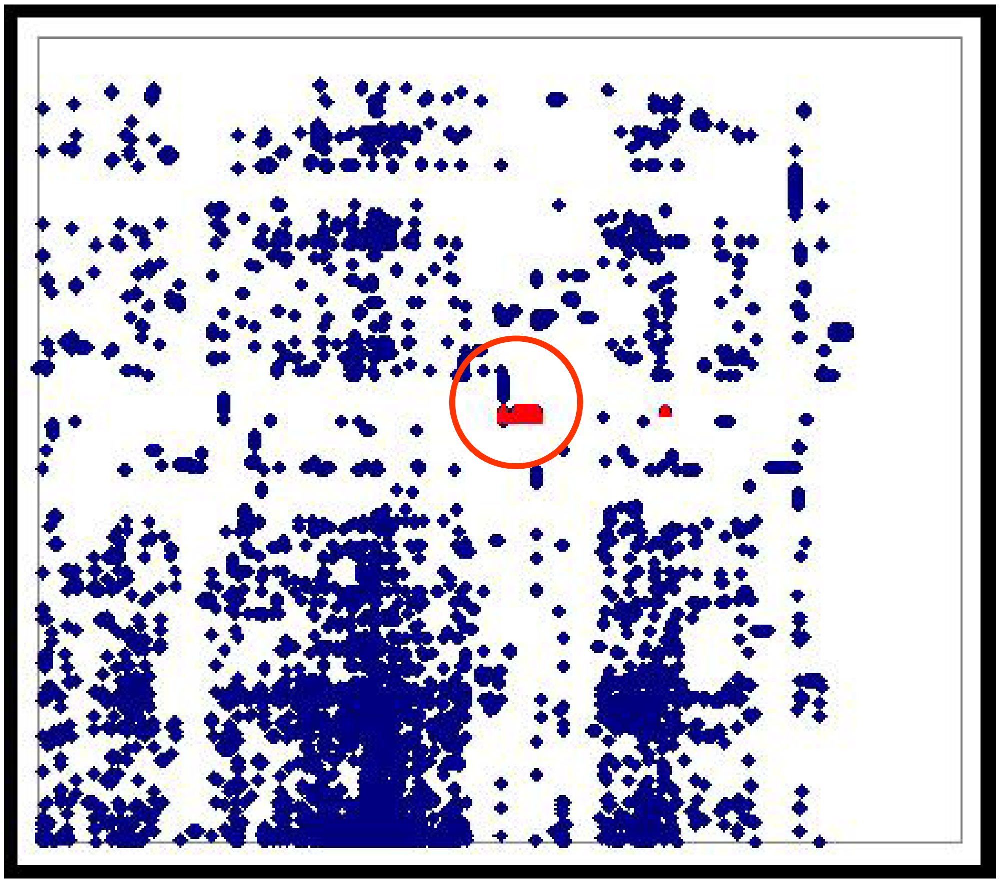
Aggregate Statistics

	AVG HR	STDEV HR	AVG R	STDEV R
HITS	22%	24%	45%	39%
PageRank	24%	14%	46%	20%
In-Degree	35%	22%	58%	29%
SALSA	35%	21%	59%	28%
MAX	38%	25%	64%	32%
BFS	43%	18%	73%	19%

Aggregate Statistics

	AVG HR	STDEV HR	AVG R	STDEV R
HITS	22%	24%	45%	39%
PageRank	24%	14%	46%	20%
In-Degree	35%	22%	58%	29%
SALSA	35%	21%	59%	28%
MAX	38%	25%	64%	32%
BFS	43%	18%	73%	19%

HITS and the TKC effect



"recipes"

- **1.** (1.000) HonoluluAdvertiser.com
  URL: http://www.hawaiisclassifieds.com
- **2.** (0.999) Gannett Company, Inc.
  URL: http://www.gannett.com
- **3.** (0.998) AP MoneyWire
  URL: http://apmoneywire.mm.ap.org
- **4.** (0.990) e.thePeople : Honolulu Advertiser
  URL: http://www.e-thepeople.com/
- **5.** (0.989) News From The Associated Press
  URL: http://customwire.ap.org/
- **6.** (0.987) Honolulu Traffic
  URL: http://www.co.honolulu.hi.us/
- **7.** (0.987) News From The Associated Press
  URL: http://customwire.ap.org/
- **8.** (0.987) News From The Associated Press
  URL: http://customwire.ap.org/
- **9.** (0.987) News From The Associated Press
  URL: http://customwire.ap.org/

10. (0.987) News From The Associated Press
URL: http://customwire.ap.org/

MAX – “net censorship”

- **1.** (1.000) EFF: Homepage
  URL: http://www.eff.org
- **2.** (0.541) Internet Free Expression Alliance
  URL: http://www.ifea.net
- **3.** (0.517) The Center for Democracy and Technology
  URL: http://www.cdt.org
- **4.** (0.517) American Civil Liberties Union
  URL: http://www.aclu.org
- **5.** (0.386) Vtw Directory Page
  URL: http://www.vtw.org
- **6.** (0.357) P E A C E F I R E
  URL: http://www.peacefire.org
- **7.** (0.277) Global Internet Liberty Campaign Home Page
  URL: http://www.gilc.org
- **8.** (0.254) libertus.net: about censorship and free speech
  URL: http://libertus.net
- **9.** (0.196) EFF Blue Ribbon Campaign Home Page
  URL: http://www.eff.org/blueribbon.html
- **10.** (0.144) The Freedom Forum
  URL: http://www.freedomforum.org

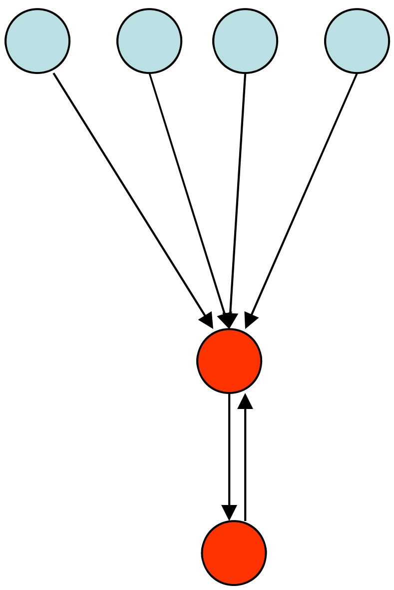
MAX – "affirmative action"

- **1.** (1.000) Copyright Information
  URL: http://www.psu.edu/copyright.html
- **2.** (0.447) PSU Affirmative Action
  URL: http://www.psu.edu/dept/aaoffice
- **3.** (0.314) Welcome to Penn State's Home on the Web
  URL: http://www.psu.edu
- **4.** (0.010) University of Illinois
  URL: http://www.uiuc.edu
- **5.** (0.009) Purdue University-West Lafayette, Indiana
  URL: http://www.purdue.edu
- **6.** (0.008) UC Berkeley home page
  URL: http://www.berkeley.edu
- **7.** (0.008) University of Michigan
  URL: http://www.umich.edu
- **8.** (0.008) The University of Arizona
  URL: http://www.arizona.edu
- **9.** (0.008) The University of Iowa Homepage
  URL: http://www.uiowa.edu
- **10.** (0.008) Penn: University of Pennsylvania
  URL: http://www.upenn.edu

PageRank

- **1.** (1.000) WCLA Feedback
  URL: http://www.janeylee.com/wcla
- **2.** (0.911) Planned Parenthood Action Network
  URL: http://www.ppaction.org/ppaction/
- **3.** (0.837) Westchester Coalition for Legal Abortion
  URL: http://www.wcla.org
- **4.** (0.714) Planned Parenthood Federation
  URL: http://www.plannedparenthood.org
- **5.** (0.633) GeneTree.com Page Not Found
  URL: http://www.qksrv.net/click
- **6.** (0.630) Bible.com Prayer Room
  URL: http://www.bibleprayerroom.com
- **7.** (0.609) United States Department of Health
  URL: http://www.dhhs.gov

  8. (0.538) Pregnancy Centers Online
  URL: http://www.pregnancycenters.org
- **9.** (0.517) Bible.com Online World
  URL: http://bible.com
- **10.** (0.516) National Organization for Women
  URL: http://www.now.org



link-spam structure

Theoretical Analysis of LAR algorithms [BRRT05]

- Why bother?
  - Plethora of LAR algorithms: we need a formal way to compare and analyze them
  - Need to define properties that are useful
    - sensitivity to spam
  - Need to discover the properties that characterize each LAR algorithm

A Theoretical Framework

- A Link Analysis Ranking Algorithm is a function that maps a graph to a real vector

$$A: G_n \rightarrow \mathbf{R}^n$$

- G_n : class of graphs of size n
- LAR vector the output A(G) of an algorithm A on a graph G
- $\mathcal{G}_n$: the class of all possible graphs of size n
- Comparing LAR vectors:

$$w_1 = [\ 1 \quad 0.8 \quad 0.5 \quad 0.3 \quad 0\ ]$$

$$w_2 = [\ 0.9 \quad 1 \quad 0.7 \quad 0.6 \quad 0.8\ ]$$

- How close are the LAR vectors w_1, w_2?

Distance between LAR vectors

- Geometric distance: how close are the numerical weights of vectors w_1, w_2?

$$d_1(w_1, w_2) = \sum \left| w_1[i] - w_2[i] \right|$$

$$w_1 = [\ 1.0\ \ 0.8\ \ 0.5\ \ 0.3\ \ 0.0\ ]$$

$$w_2 = [\ 0.9\ \ 1.0\ \ 0.7\ \ 0.6\ \ 0.8\ ]$$

$$d_1(w_1, w_2) = 0.1+0.2+0.2+0.3+0.8 = 1.6$$

- Rank distance: how close are the ordinal rankings induced by the vectors w_1, w_2?
  - Kendal's τ distance

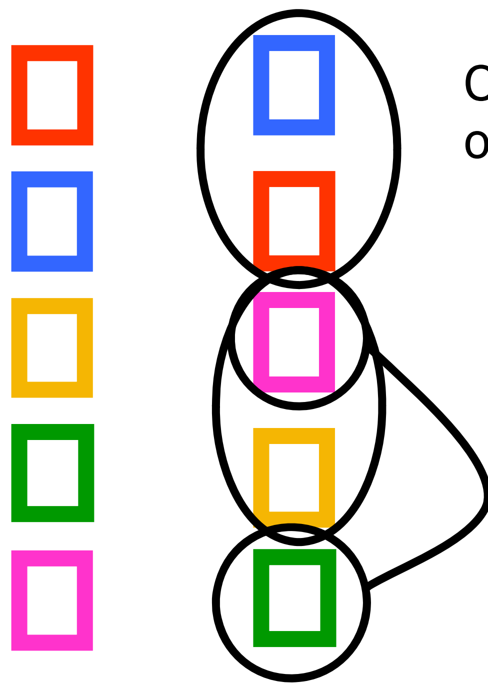
$$d_r(w_1, w_2) = \frac{\text{pairs ranked in a different order}}{\text{total number of distinct pairs}}$$

Rank distance

$$w_1 = [\ 1 \quad 0.8 \quad 0.5 \quad 0.3 \quad 0\ ]$$

$$w_2 = [\ 0.9 \quad 1 \quad 0.7 \quad 0.6 \quad 0.8\ ]$$

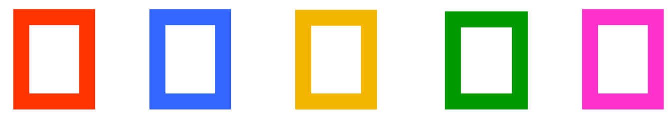
Ordinal Ranking of vector w_1



Ordinal Ranking of vector w_2

$$d_r(w_1, w_2) = \frac{3}{5 * 4/2} = 0.3$$

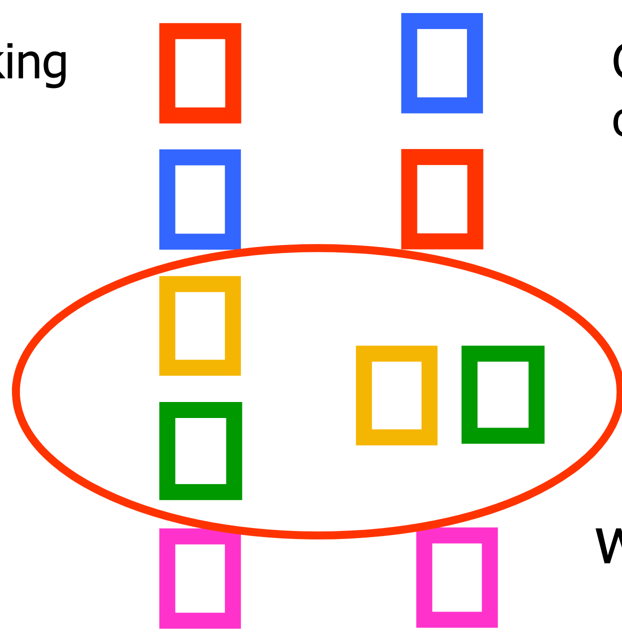
Rank distance of partial rankings



$$w_1 = [\ 1 \quad 0.8 \quad 0.5 \quad 0.3 \quad 0\ ]$$

$$w_2 = [\ 0.9 \quad 1 \quad 0.7 \quad 0.7 \quad 0.3\ ]$$

Ordinal Ranking of vector w_1



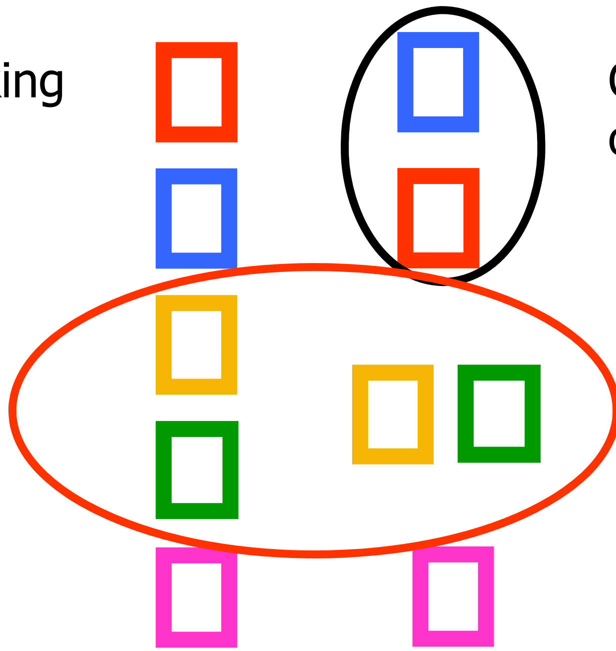
Ordinal Ranking of vector w_2

what do we do with such pairs?

Rank distance of partial rankings

- Charge penalty p for each pair (i,j) of nodes such that $w_1[i] \neq w_1[j]$ and $w_2[i] = w_2[j]$

Ordinal Ranking of vector w_1



Ordinal Ranking of vector w_2

$$d_r(w_1, w_2) = \frac{1+p}{10}$$

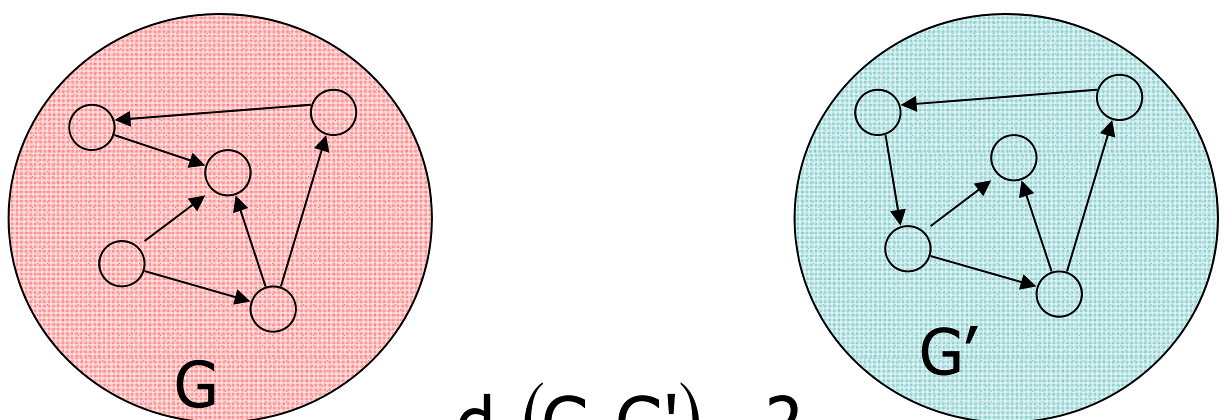
Rank distance of partial rankings

- Extreme value $p = 1$
  - charge for every potential conflict
- Extreme value $p = 0$
  - charge only for inconsistencies
  - problem: not a metric
- Intermediate values $0 < p < 1$
  - Details [FMNKS04] [T04]
  - Interesting case $p = 1/2$

Stability: graph distance

- Intuition: a small change on a graph should cause a small change on the output of the algorithm.
- Definition: Link distance between graphs $G=(P,E)$ and $G'=(P,E')$

$$d_\ell(G,G') = |E \cup E'| - |E \cap E'|$$



$$d_\ell(G,G') = 2$$

Stability

- $C_k(G)$: set of graphs G' such that $d_\ell(G,G') \le k$
- Definition: Algorithm A is stable if

$$\lim_{n\to\infty} \max_{G} \max_{G' \in C_k(G)} d_1(A(G),A(G')) = 0$$

- Definition: Algorithm A is rank stable if

$$\lim_{n\to\infty} \max_{G} \max_{G' \in C_k(G)} d_r\big(A(G),A(G')\big) = 0$$

- Results
  - InDegree algorithm is stable and rank stable on the class $\mathcal{G}_n$
  - HITS and Max are neither stable nor rank stable on the class $\mathcal{G}_n$

Stability of PageRank

- Perturbations to unimportant nodes have small effect on the PageRank values [NZJ01][BGS03]

- Lee Borodin model [LB03]
  - upper bounds depend on authority and hub values
  - PageRank, Randomized SALSA are stable
  - HITS, SALSA are unstable

- Open question: Can we derive conditions for the stability of PageRank in the general case?

Similarity

- Definition: Two algorithms A_1, A_2 are similar if

$$\lim_{n\to\infty}\frac{\max_{G\in G_n} d_1\left(A_1(G), A_2(G)\right)}{\max_{w_1, w_2} d_1\left(w_1, w_2\right)} = 0$$

- Definition: Two algorithms A_1, A_2 are rank similar if

$$\lim_{n\to\infty}\max_{G\in G_n} d_r\left(A_1(G), A_2(G)\right) = 0$$

- Definition: Two algorithms A_1, A_2 are rank equivalent if

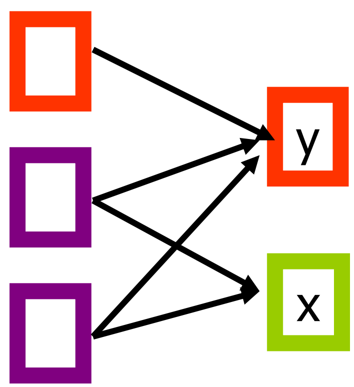
$$\max_{G\in G_n} d_r\left(A_1(G), A_2(G)\right) = 0$$

- Results:
  - No pairwise combination of InDegree, SALSA, HITS and MAX algorithms is similar, or rank similar on the class of all possible graphs $\mathcal{G}_n$

Monotonicity

- Monotonicity: Algorithm A is strictly monotone if for any nodes x and y

$$B_N(x) \subset B_N(y) \Leftrightarrow A(G)[x] < A(G)[y]$$



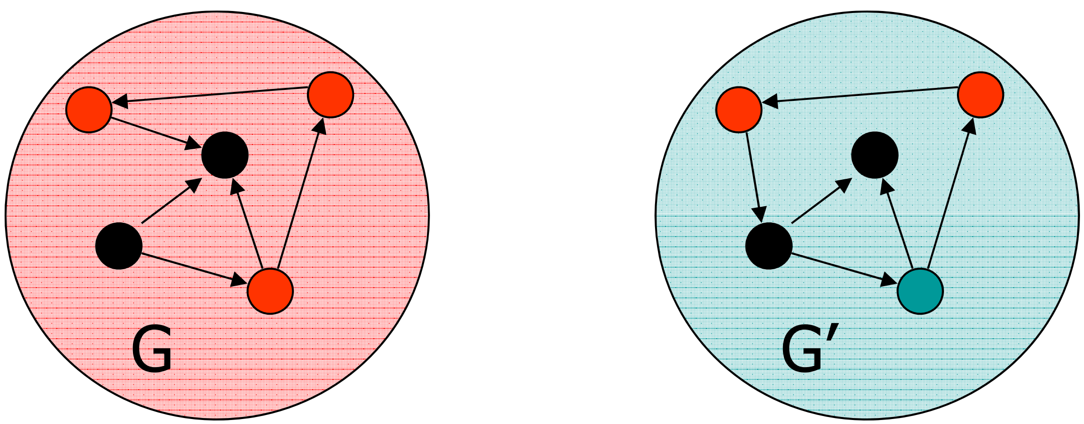
$\mathbf{w}_x < \mathbf{w}_y$

Locality

- Locality: An algorithm A is strictly rank local if, for every pair of graphs $G=(P,E)$ and $G'=(P,E')$, and for every pair of nodes x and y, if $B_G(x)=B_{G'}(x)$ and $B_G(y)=B_{G'}(y)$ then

$$A(G)[x] < A(G)[y] \Leftrightarrow A(G')[x] < A(G')[y]$$

  - the relative order of the nodes remains the same



- The InDegree algorithm is strictly rank local

Label Independence

- Label Independence: An algorithm is label independent if a permutation of the labels of the nodes yields the same permutation of the weights
  - the weights assigned by the algorithm do not depend on the labels of the nodes

Axiomatic characterization of the InDegree algorithm [BRRT05]

- **Theorem:** Any algorithm that is strictly rank local, strictly monotone and label independent is rank equivalent to the InDegree algorithm.

Axiomatic characterization

- All three properties are needed
  - locality
    - PageRank is also strictly monotone and label independent
  - monotonicity
    - consider an algorithm that assigns 1 to nodes with even degree, and 0 to nodes with odd degree
  - label independence
    - consider and algorithm that gives the more weight to links that come from some specific page (e.g. the Yahoo page)

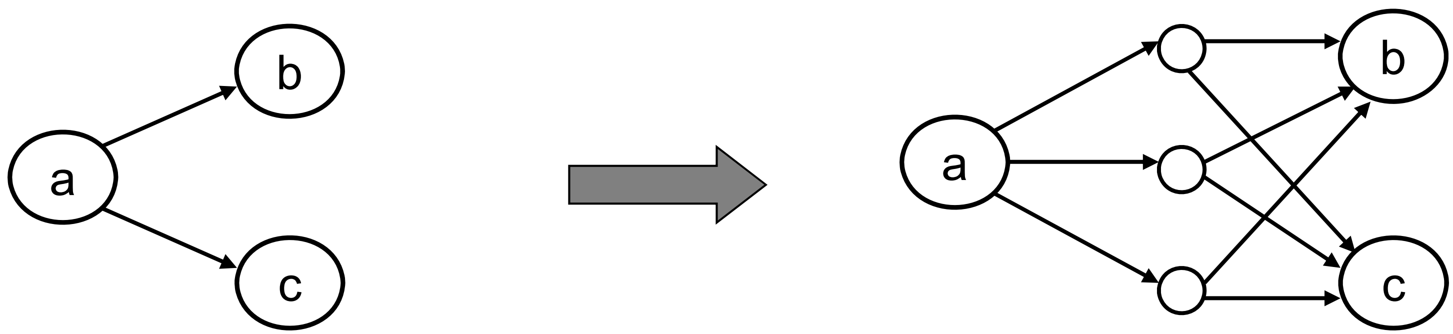
Self-edge axiom

- Algorithm A satisfies the self-edge axiom if the following is true: If page a is ranked at least as high as page b in a graph $G(V,E)$, where a does not have a link to itself, then a should be ranked higher than b in $G(V,E \cup \{v,v\})$

Vote by committee axiom

- Algorithm A satisfies the vote by committee axiom if the following is true: If page *a* links to pages *b* and *c*, then the relative ranking of all the pages should be the same as in the case where the direct links from *a* to *b* and *c* are replaced by links from *a* to a new set of pages which link (only) to *b* and *c*

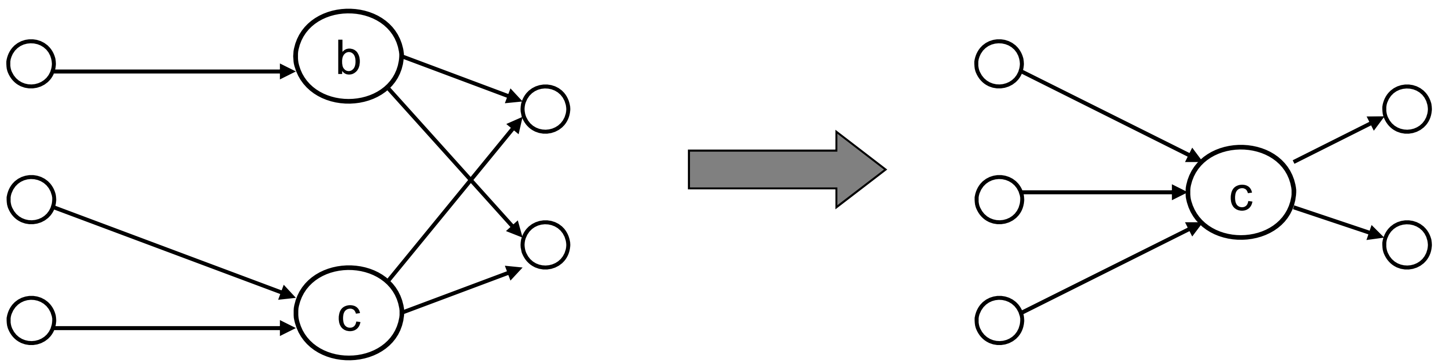
Vote by committee (example)



Collapsing axiom

- If there is a pair of pages a and b that link to the same set of pages, but the set of pages that link to a and b are disjoint, then if a and b are collapsed into a single page (a), where links of b become links of a, then the relative rankings of all pages (except a and b) should remain the same.

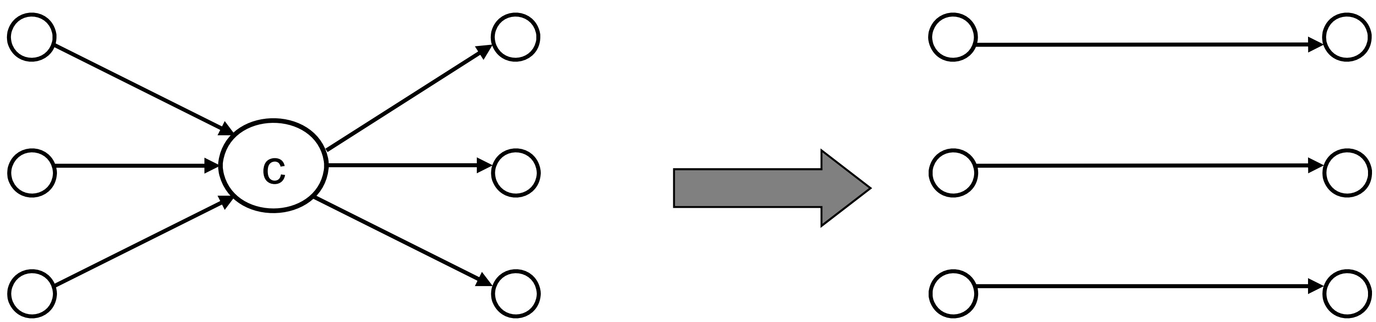
Collapsing axiom (example)



Proxy axiom

- If there is a set of k pages with the same importance that link to a, and a itself links to k other pages, then by dropping a and connect the pages in N(a) and P(a), the relative ranking of all pages (excluding a) should remain the same

Proxy axiom (example)



Axiomatic Characterization of PageRank Algorithm [AT04]

- The PageRank algorithm satisfies label independence, self-edge, vote by committee, collapsing and proxy axioms.

Rank Aggregation

- Given a set of rankings $R_1, R_2, \ldots, R_m$ of a set of objects $X_1, X_2, \ldots, X_n$ produce a single ranking R that is in agreement with the existing rankings
- Examples: Voting
    - rankings $R_1, R_2, \ldots, R_m$ are the voters, the objects $X_1, X_2, \ldots, X_n$ are the candidates.

Examples

- Combining multiple scoring functions
  - rankings $R_1, R_2, \ldots, R_m$ are the scoring functions, the objects $X_1, X_2, \ldots, X_n$ are data items.
    - Combine the PageRank scores with term-weighting scores
    - Combine scores for multimedia items
      - color, shape, texture
    - Combine scores for database tuples
      - find the best hotel according to price and location
- Combining multiple sources
  - rankings $R_1, R_2, \ldots, R_m$ are the sources, the objects $X_1, X_2, \ldots, X_n$ are data items.
    - meta-search engines for the Web
    - distributed databases
    - P2P sources

Variants of the problem

- Combining scores
  - we know the scores assigned to objects by each ranking, and we want to compute a single score
- Combining ordinal rankings
  - the scores are not known, only the ordering is known
  - the scores are known but we do not know how, or do not want to combine them
    - e.g. price and star rating

Combining scores

- Each object X_i has m scores $(r_{i1},r_{i2},\ldots,r_{im})$
- The score of object X_i is computed using an aggregate scoring function $f(r_{i1},r_{i2},\ldots,r_{im})$

	R_1	R_2	R_3
X_1	1	0.3	0.2
X_2	0.8	0.8	0
X_3	0.5	0.7	0.6
X_4	0.3	0.2	0.8
X_5	0.1	0.1	0.1

Combining scores

- Each object X_i has m scores $(r_{i1}, r_{i2}, \ldots, r_{im})$
- The score of object X_i is computed using an aggregate scoring function $f(r_{i1}, r_{i2}, \ldots, r_{im})$
  - $f(r_{i1}, r_{i2}, \ldots, r_{im}) = \min\{r_{i1}, r_{i2}, \ldots, r_{im}\}$

	R_1	R_2	R_3	R
X_1	1	0.3	0.2	0.2
X_2	0.8	0.8	0	0
X_3	0.5	0.7	0.6	0.5
X_4	0.3	0.2	0.8	0.2
X_5	0.1	0.1	0.1	0.1

Combining scores

- Each object X_i has m scores $(r_{i1}, r_{i2}, \dots, r_{im})$
- The score of object X_i is computed using an aggregate scoring function $f(r_{i1}, r_{i2}, \dots, r_{im})$
  - $f(r_{i1}, r_{i2}, \dots, r_{im}) = \max\{r_{i1}, r_{i2}, \dots, r_{im}\}$

	R_1	R_2	R_3	R
X_1	1	0.3	0.2	1
X_2	0.8	0.8	0	0.8
X_3	0.5	0.7	0.6	0.7
X_4	0.3	0.2	0.8	0.8
X_5	0.1	0.1	0.1	0.1

Combining scores

- Each object X_i has m scores $(r_{i1}, r_{i2}, \ldots, r_{im})$
- The score of object X_i is computed using an aggregate scoring function $f(r_{i1}, r_{i2}, \ldots, r_{im})$
  - $f(r_{i1}, r_{i2}, \ldots, r_{im}) = r_{i1} + r_{i2} + \ldots + r_{im}$

	R_1	R_2	R_3	R
X_1	1	0.3	0.2	1.5
X_2	0.8	0.8	0	1.6
X_3	0.5	0.7	0.6	1.8
X_4	0.3	0.2	0.8	1.3
X_5	0.1	0.1	0.1	0.3

Top-k

- Given a set of n objects and m scoring lists sorted in decreasing order, find the top-k objects according to a scoring function f
- top-k: a set T of k objects such that $f(r_{j1},\ldots,r_{jm}) \leq f(r_{i1},\ldots,r_{im})$ for every object X_i in T and every object X_j not in T
- Assumption: The function f is monotone
  - $f(r_1,\ldots,r_m) \leq f(r_1',\ldots,r_m')$ if $r_i \leq r_i'$ for all i
- Objective: Compute top-k with the minimum cost

Cost function

- We want to minimize the number of accesses to the scoring lists
- Sorted accesses: sequentially access the objects in the order in which they appear in a list
  - cost C_s
- Random accesses: obtain the cost value for a specific object in a list
  - cost C_r
- If s sorted accesses and r random accesses minimize $s\,C_s + r\,C_r$

Example

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

- Compute top-2 for the sum aggregate function

Fagin's Algorithm

1. Access sequentially all lists in parallel until there are k objects that have been seen in all lists

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin's Algorithm

1. Access sequentially all lists in parallel until there are k objects that have been seen in all lists

R_1		R_2		R_3	
X_1	1	X_2	0.8	X_4	0.8
X_2	0.8	X_3	0.7	X_3	0.6
X_3	0.5	X_1	0.3	X_1	0.2
X_4	0.3	X_4	0.2	X_5	0.1
X_5	0.1	X_5	0.1	X_2	0

Fagin's Algorithm

1. Access sequentially all lists in parallel until there are k objects that have been seen in all lists

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin's Algorithm

1. Access sequentially all lists in parallel until there are k objects that have been seen in all lists

R_1		R_2		R_3	
X_1	1	X_2	0.8	X_4	0.8
X_2	0.8	X_3	0.7	X_3	0.6
X_3	0.5	X_1	0.3	X_1	0.2
X_4	0.3	X_4	0.2	X_5	0.1
X_5	0.1	X_5	0.1	X_2	0

Fagin's Algorithm

1. Access sequentially all lists in parallel until there are k objects that have been seen in all lists

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin's Algorithm

2. Perform random accesses to obtain the scores of all seen objects

R_1		R_2		R_3	
X_1	1	X_2	0.8	X_4	0.8
X_2	0.8	X_3	0.7	X_3	0.6
X_3	0.5	X_1	0.3	X_1	0.2
X_4	0.3	X_4	0.2	X_5	0.1
X_5	0.1	X_5	0.1	X_2	0

Fagin's Algorithm

3. Compute score for all objects and find the top-k

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

R	
X_3	1.8
X_2	1.6
X_1	1.5
X_4	1.3

Fagin's Algorithm

- X_5 cannot be in the top-2 because of the monotonicity property

$$f(X_5) \le f(X_1) \le f(X_3)$$

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

R	
X_3	1.8
X_2	1.6
X_1	1.5
X_4	1.3

Threshold algorithm

1. Access the elements sequentially

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Threshold algorithm

1. At each sequential access
   a. Set the threshold t to be the aggregate of the scores seen in this access

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 2.6$

Threshold algorithm

1. At each sequential access
   b. Do random accesses and compute the score of the objects seen

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 2.6$

X_1	1.5
X_2	1.6
X_4	1.3

Threshold algorithm

1. At each sequential access
   c. Maintain a list of top-k objects seen so far

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 2.6$

X_2	1.6
X_1	1.5

Threshold algorithm

1. At each sequential access
   d. When the scores of the top-k are greater or equal to the threshold, stop

R_1		R_2		R_3	
X_1	1	X_2	0.8	X_4	0.8
X_2	0.8	X_3	0.7	X_3	0.6
X_3	0.5	X_1	0.3	X_1	0.2
X_4	0.3	X_4	0.2	X_5	0.1
X_5	0.1	X_5	0.1	X_2	0

$t = 2.1$

X_3	1.8
X_2	1.6

Threshold algorithm

1. At each sequential access
   d. When the scores of the top-k are greater or equal to the threshold, stop

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 1.0$

X_3	1.8
X_2	1.6

Threshold algorithm

2. Return the top-k seen so far

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 1.0$

X_3	1.8
X_2	1.6

Combining rankings

- In many cases the scores are not known
  - e.g. meta-search engines – scores are proprietary information
- … or we do not know how they were obtained
  - one search engine returns score 10, the other 100. What does this mean?
- … or the scores are incompatible
  - apples and oranges: does it make sense to combine price with distance?
- In this cases we can only work with the rankings
- **Input:** a set of rankings $R_1,R_2,\ldots,R_m$ of the objects $X_1,X_2,\ldots,X_n$. Each ranking R_i is a total ordering of the objects
  - for every pair X_i,X_j either X_i is ranked above X_j or X_j is ranked above X_i
- **Output:** A total ordering R that aggregates rankings $R_1,R_2,\ldots,R_m$

Voting theory

- A voting system is a rank aggregation mechanism
- Long history and literature
  - criteria and axioms for good voting systems
- The Condorcet criterion
  - if object A defeats every other object in a pairwise majority vote, then A should be ranked first
- Extended Condorcet criterion
  - if the objects in a set X defeat in pairwise comparisons the objects in the set Y then the objects in X should be ranked above those in Y
- Not all voting systems satisfy the Condorcet criterion!

Pairwise majority comparisons

- Unfortunately the Condorcet winner does not always exist
  - irrational behavior of groups

	V_1	V_2	V_3
1	A	B	C
2	B	C	A
3	C	A	B

A > B B > C C > A

Pairwise majority comparisons

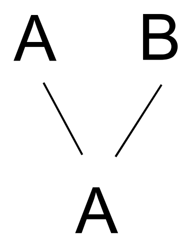
- Resolve cycles by imposing an agenda

	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D

Pairwise majority comparisons

- Resolve cycles by imposing an agenda

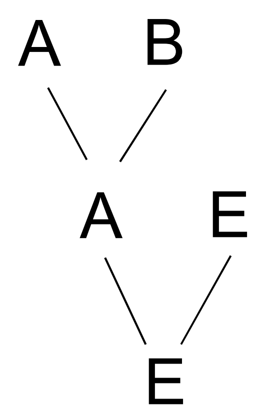
	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D



Pairwise majority comparisons

- Resolve cycles by imposing an agenda

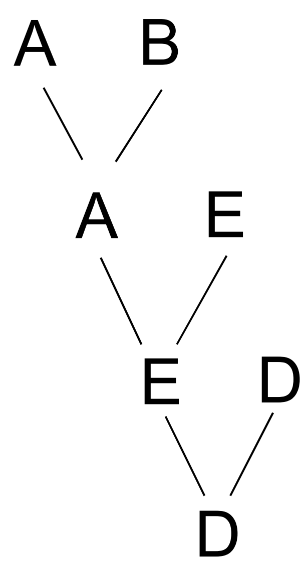
	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D



Pairwise majority comparisons

- Resolve cycles by imposing an agenda

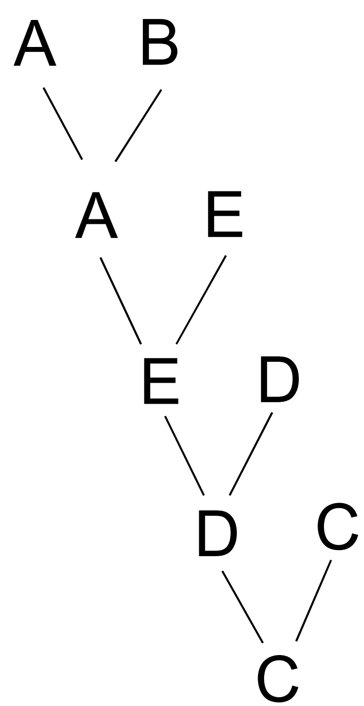
	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D



Pairwise majority comparisons

- Resolve cycles by imposing an agenda

	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D

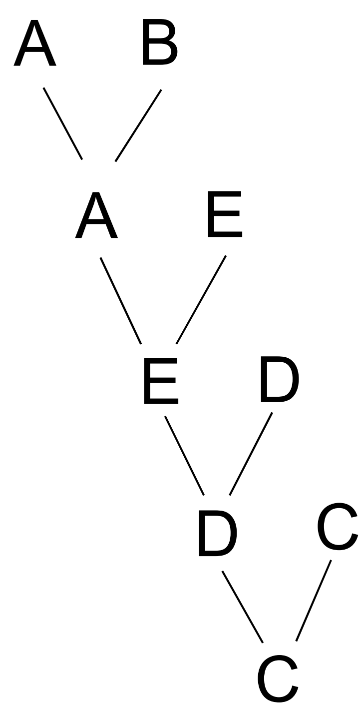


- C is the winner

Pairwise majority comparisons

- Resolve cycles by imposing an agenda

	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D



- But everybody prefers A or B over C

Pairwise majority comparisons

- The voting system is not Pareto optimal
  - there exists another ordering that everybody prefers
- Also, it is sensitive to the order of voting

Plurality vote

- Elect first whoever has more 1st position votes

voters	10	8	7
1	A	C	B
2	B	A	C
3	C	B	A

- Does not find a Condorcet winner (C in this case)

Plurality with runoff

- If no-one gets more than 50% of the 1st position votes, take the majority winner of the first two

voters	10	8	7	2
1	A	C	B	B
2	B	A	C	A
3	C	B	A	C

first round: A 10, B 9, C 8
second round: A 18, B 9
winner: A

Plurality with runoff

- If no-one gets more than 50% of the 1st position votes, take the majority winner of the first two

voters	10	8	7	2
1	A	C	B	A
2	B	A	C	B
3	C	B	A	C

change the order of A and B in the last column

first round: A 12, B 7, C 8
second round: A 12, C 15
winner: C!

Positive Association axiom

- Plurality with runoff violates the positive association axiom
- Positive association axiom: positive changes in preferences for an object should not cause the ranking of the object to decrease

Borda Count

- For each ranking, assign to object X, number of points equal to the number of objects it defeats
  - first position gets n-1 points, second n-2, …, last 0 points
- The total weight of X is the number of points it accumulates from all rankings

Borda Count

voters	3	2	2
1 (3p)	A	B	C
2 (2p)	B	C	D
3 (1p)	C	D	A
4 (0p)	D	A	B

A: 3*3 + 2*0 + 2*1 = 11p
B: 3*2 + 2*3 + 2*0 = 12p
C: 3*1 + 2*2 + 2*3 = 13p
D: 3*0 + 2*1 + 2*2 = 6p

BC
C
B
A
D

- Does not always produce Condorcet winner

Borda Count

- Assume that D is removed from the vote

voters	3	2	2
1 (2p)	A	B	C
2 (1p)	B	C	A
3 (0p)	C	A	B

A: 3*2 + 2*0 + 2*1 = 7p
B: 3*1 + 2*2 + 2*0 = 7p
C: 3*0 + 2*1 + 2*2 = 6p

BC
B
A
C

- Changing the position of D changes the order of the other elements!

Independence of Irrelevant Alternatives

- The relative ranking of X and Y should not depend on a third object Z
  - heavily debated axiom

Borda Count

- The Borda Count of an an object X is the aggregate number of pairwise comparisons that the object X wins
  - follows from the fact that in one ranking X wins all the pairwise comparisons with objects that are under X in the ranking

Arrow's Impossibility Theorem

- There is no voting system that satisfies the following axioms
  - Universality
    - all inputs are possible
  - Completeness and Transitivity
    - for each input we produce an answer and it is meaningful
  - Positive Assosiation
  - Independence of Irrelevant Alternatives
  - Non-imposition
  - Non-dictatoriship

- **KENNETH J. ARROW** *Social Choice and Individual Values* (1951). Won Nobel Prize in 1972

Kemeny Optimal Aggregation

- Kemeny distance $K(R_1,R_2)$: The number of pairs of nodes that are ranked in a different order (Kendall-tau)
  - number of bubble-sort swaps required to transform one ranking into another
- Kemeny optimal aggregation minimizes

$$K(R,R_1,\dots,R_m)=\sum_{i=1}^{m}K(R,R_i)$$

- Kemeny optimal aggregation satisfies the Condorcet criterion and the extended Condorcet criterion
  - maximum likelihood interpretation: produces the ranking that is most likely to have generated the observed rankings
- …but it is NP-hard to compute
  - easy 2-approximation by obtaining the best of the input rankings, but it is not "interesting"

Locally Kemeny optimal aggregation

- A ranking R is locally Kemeny optimal if there is no bubble-sort swap that produces a ranking R′ such that

$$K(R,R_1,\ldots,R_m) \leq K(R',R_1,\ldots,R_m)$$

- Locally Kemeny optimal is not necessarily Kemeny optimal
- Definitions apply for the case of partial lists also
- Locally Kemeny optimal aggregation can be computed in polynomial time
  - At the i-th iteration insert the i-th element x in the bottom of the list, and bubble it up until there is an element y such that the majority places y over x
- Locally Kemeny optimal aggregation satisfies the Condorcet and extended Condorcet criterion

Rank Aggregation algorithm [DKNS01]

- Start with an aggregated ranking and make it into a locally Kemeny optimal aggregation
- How do we select the initial aggregation?
  - Use another aggregation method
  - Create a Markov Chain where you move from an object X, to another object Y that is ranked higher by the majority

Spearman's footrule distance

- Spearman's footrule distance: The difference between the ranks R(i) and R'(i) assigned to object i

$$F(R,R') = \sum_{i=1}^{n} \left| R(i) - R'(i) \right|$$

- Relation between Spearman's footrule and Kemeny distance

$$K(R,R') \leq F(R,R') \leq 2K(R,R')$$

- Find the ranking R, that minimizes

$$F(R,R_1,\ldots,R_m) = \sum_{i=1}^{m} F(R,R_i)$$

- The optimal Spearman's footrule aggregation can be computed in polynomial time
  - It also gives a 2-approximation to the Kemeny optimal aggregation
- If the median ranks of the objects are unique then this ordering is optimal

Example

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	B
2	A
3	C
4	D

A: (1 , $\boxed{2}$, 3)
B: (1 , $\boxed{1}$, 2)
C: (3 , $\boxed{3}$, 4)
D: (3 , $\boxed{4}$, 4)

The MedRank algorithm

- Access the rankings sequentially

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	
2	
3	
4	

The MedRank algorithm

- Access the rankings sequentially
  - when an element has appeared in more than half of the rankings, output it in the aggregated ranking

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	B
2	
3	
4	

The MedRank algorithm

- Access the rankings sequentially
  - when an element has appeared in more than half of the rankings, output it in the aggregated ranking

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	B
2	A
3	
4	

The MedRank algorithm

- Access the rankings sequentially
  - when an element has appeared in more than half of the rankings, output it in the aggregated ranking

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	B
2	A
3	C
4	

The MedRank algorithm

- Access the rankings sequentially
  - when an element has appeared in more than half of the rankings, output it in the aggregated ranking

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	B
2	A
3	C
4	D

The Spearman’s rank correlation

- Spearman’s rank correlation

$$S(R,R') = \sum_{i=1}^{n} \left(R(i) - R'(i)\right)^2$$

- Computing the optimal rank aggregation with respect to Spearman’s rank correlation is the same as computing Borda Count
  - Computable in polynomial time

Readings

- "Networks, Crowds, and Markets" by Easley and Kleinberg (Chapters 13 and 14)
- Adamic and Adar, How to search a social network, Social Networks, 27(3), p.187-203, 2005.
- David Liben-Nowell, Jasmine Novak, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins, Geographical routing in social networks, Proceedings of the National Academy of USA, 102(33), p.11623-11628, 2005.