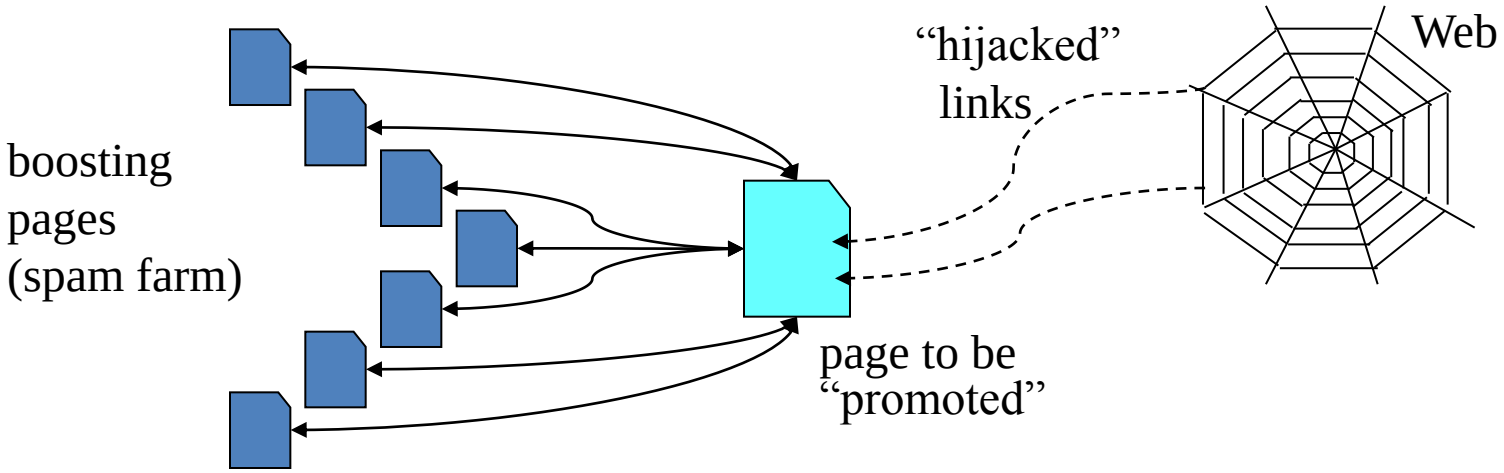
IV.5 Link Spam: Not Just E-mails Anymore

Distortion of search results by "spam farms" and "hijacked" links (aka. search engine optimization)



Susceptibility to manipulation and lack of trust model is a major problem:

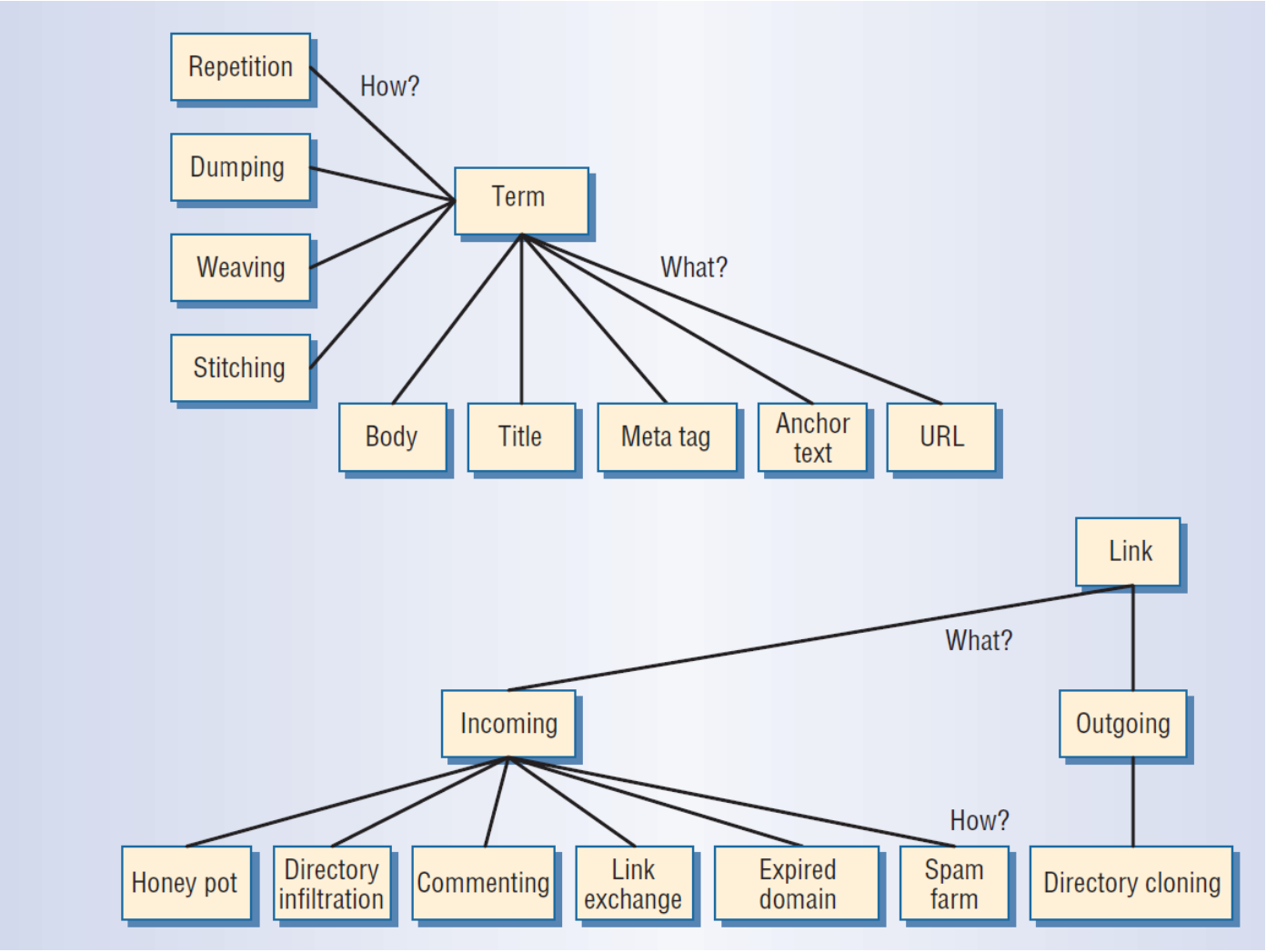
- Successful 2004 DarkBlue SEO Challenge: *"nigritude ultramarine"*
- Pessimists estimate 75 Mio. out of 150 Mio. Web hosts are spam

Research challenge:

- Robustness to egoistic and malicious behavior
- Trust/distrust models and mechanisms

→But often unclear borderline between spam and community opinions

Content Spam vs. Link Spam



Source: Z. Gyöngyi, H. Garcia-Molina: Spam: It's Not Just for Inboxes Anymore, IEEE Computer 2005

From PageRank to TrustRank

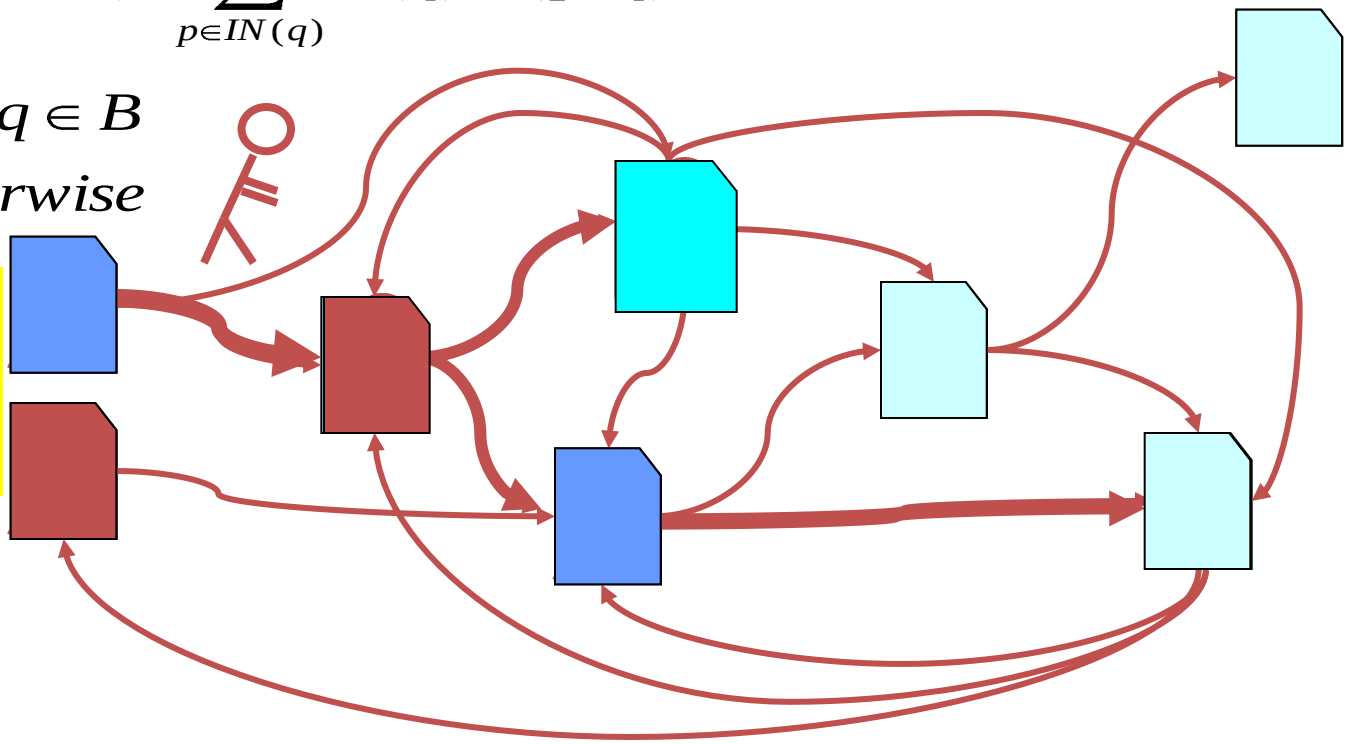
[Kamvar et al.: WWW'03, Gyöngyi et al.: VLDB'04]

Idea: PRP random jumps favor designated high-quality pages (B) such as personal bookmarks, frequently visited pages, etc.

$$PR(q) = \varepsilon \cdot j(q) + (1-\varepsilon) \cdot \sum_{p \in IN(q)} PR(q) \cdot t(p,q)$$

$$j(q) = \begin{cases} 1/|B| & for\ q \in B \\ 0 & otherwise \end{cases}$$

Authority (page q) = stationary prob. of visiting q



Random walk: uniformly random choice of links
+ biased jumps to trusted pages

Counter Measures: TrustRank and BadRank

TrustRank:

Start with explicit **set T of trusted pages** with trust values t_i and define random-jump vector r by setting $r_i = 1/|T|$ if $i \in T$, and 0 else.
Propagate TrustRank mass to successors:

$$TR(q) = \tau\, r + (1-\tau) \sum_{p \in IN(q)} TR(p) / \text{outdegree}(p)$$

BadRank:

Start with explicit **set B of blacklisted pages** and
define random-jump vector r by setting $r_i = 1/|B|$ if $i \in B$, and 0 else.
Propagate BadRank mass to predecessors

$$BR(p) = \beta\, r + (1-\beta) \sum_{q \in OUT(p)} BR(q) / \text{indegree}(q)$$

Problems:

- Difficult maintenance of explicit page lists
- Difficult to understand (& guarantee) effects

Spam, Damn Spam, and Statistics

Spam detection based on **statistical deviation**:

- **Content spam:**
  compare the word frequency distribution
  to the general distribution in "good sites"
- **Link spam:**
  find outliers in outdegree and indegree distributions
  and inspect intersection

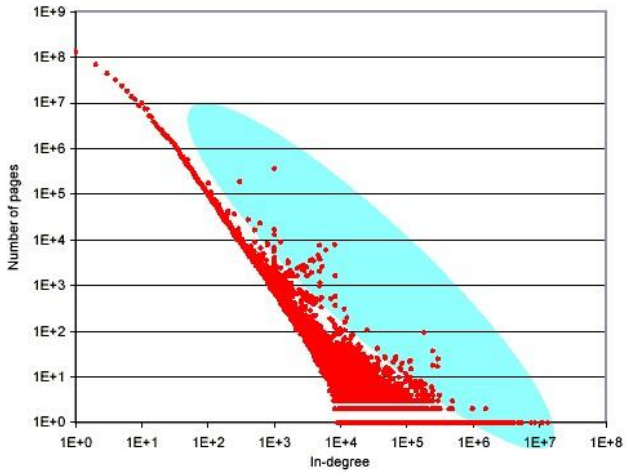


Figure 5: Distribution of in-degrees

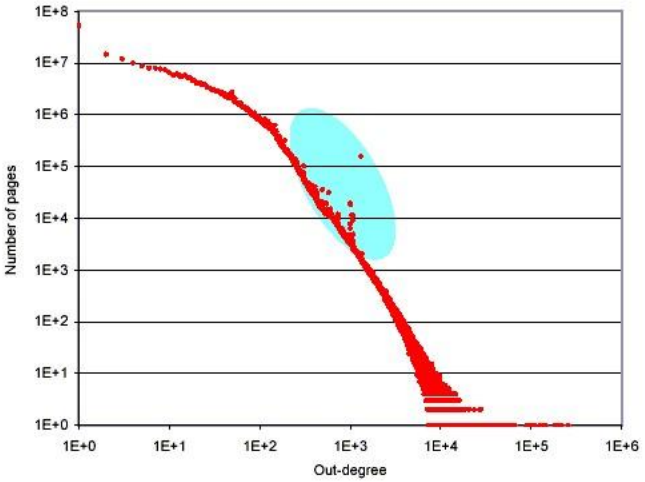


Figure 4: Distribution of out-degrees

Typical for the Web:
$P[\text{degree}=k] \sim (1/k)^{\alpha}$
$\alpha \approx 2.1$ for indegrees
$\alpha \approx 2.7$ for outdegrees
(Zipfian distribution)

Source: D. Fetterly, M. Manasse, M. Najork: WebDB 2004

SpamRank [Benczur et al. 2005]

Key idea:

Inspect **PR distribution** among a suspected page's neighborhood in a power-law graph.

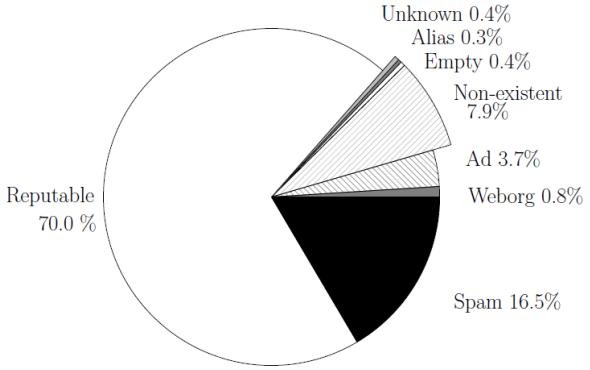
→ Should also be **power-law distributed**, and deviation is suspicious (e.g., pages that receive their PR from very many low-PR pages).

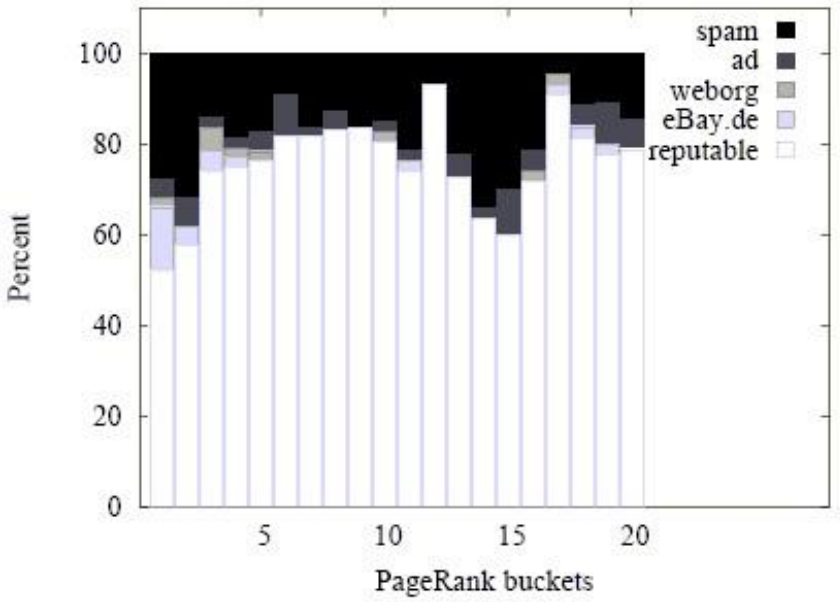
3-Phase computation:

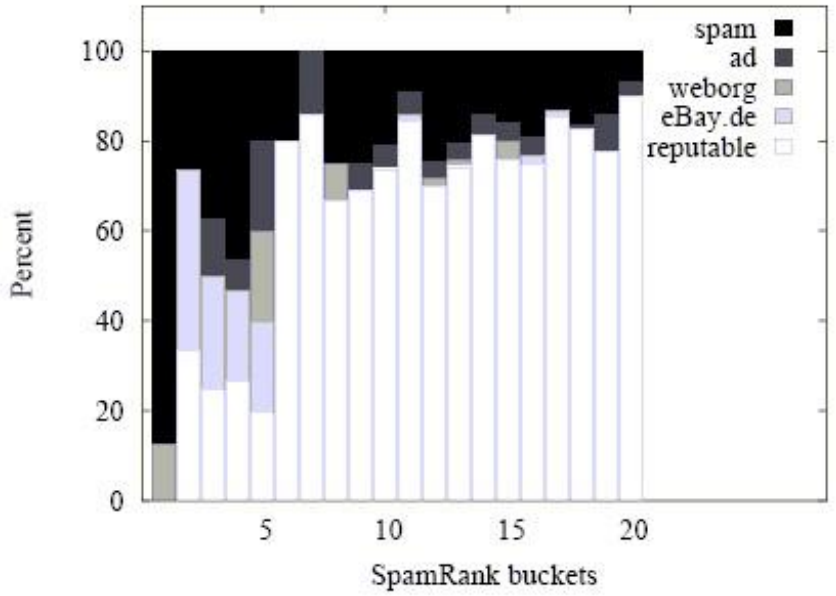
1) For each page q and supporter p compute approximate PPR(q) with random-jump vector r_p=1, and 0 otherwise.
   → **$PPR_p(q)$** is interpreted as **support of p for q**.
2) For each page p compute a penalty based on PPR vectors.
3) Define one PPR vector with penalties as random-jump prob's and compute SpamRank as "personalized" BadRank.

→ **TrueAuthority(p) = PageRank(p) – SpamRank(p)**

SpamRank Experimental Results







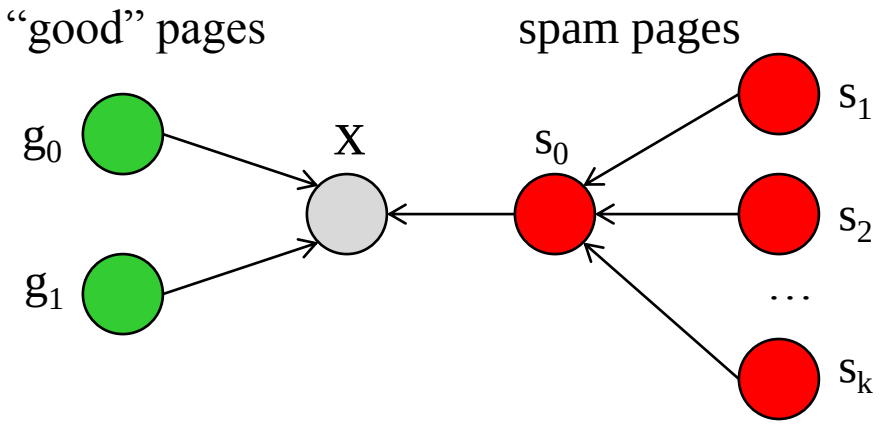
Distribution of PageRank and SpamRank Mass over Web-Page Categories (1000 pages sample)

Source: Benczur et al., AIRWeb Workshop 2005

How to Estimate "Spam Mass"

[Gyöngyi et al.: VLDB 05/06]

Naïve approach:
Only consider number of immediate in-neighbors for spam detection.



Consider general PR formula:

$$PR(q) = \varepsilon \cdot j(q) + (1-\varepsilon) \cdot \sum_{p \in IN(q)} PR(q) \cdot t(p,q)$$

For the above graph, we obtain

$$PR(x) = (1 + 3(1-\varepsilon) + k(1-\varepsilon)^2)\frac{\varepsilon}{n}$$

where $((1-\varepsilon) + k(1-\varepsilon)^2)\frac{\varepsilon}{n}$

is due to spam pages s_i.

For $\varepsilon = 0.15$ and $k \geq \text{ceil}(1/\varepsilon) = 2$, the largest part of PR(x) comes from spam pages!

SpamMass Score

[Gyöngyi et al.: VLDB 05/06]

PR contribution of page p to page q:

$$PC_p(q) = P[\,jump\ to\ \ p] \cdot \sum_{\substack{tourst: \\ p \to q}} P[t](1-\varepsilon)\,\varepsilon^{length(t)}$$

→ Compute by PPR with jump to p only

PR of page q: $PR(q) = \sum_{all\ pages\ p} PC_p(q)$

Method:

Assume Web W is partitioned into good pages W^+ and bad pages W^- .

Assume that "**good core**" $V^+ \subset W^+$ is known.

Estimate **SpamMass** of page q: $SM(q) = PR(q) - \sum_{p \in V^+} PC_p(q)$

and **relative SpamMass** of q: $rSM(q) = SM(q) / PR(q)$

Learning Spam Features [Drost/Scheffer 2005]

Use page classifier (e.g., Naïve Bayes, SVM) to **predict "spam vs. ham"** based on page and page-context features

But spammers may learn to adjust to the anti-spam measures. It's an arms race!

Most discriminative features are:

- tfidf weights of words in p0 and IN(p0)
- avg. #inlinks of pages in IN(p0)
- avg. #words in title of pages in OUT(p0)
- #pages in IN(p0) that have same length as some other page in IN(p0)
- avg. # inlinks and outlinks of pages in IN(p0)
- avg. #outlinks of pages in IN(p0)
- avg. #words in title of p0
- total #outlinks of pages in OUT(p0)
- total #inlinks of pages in IN(p0)
- clustering coefficient of pages in IN(p0) (#linked pairs / m(m-1) possible pairs)
- total #words in titles of pages in OUT(p0)
- total #outlinks of pages in OUT(p0)
- avg. #characters of URLs in IN(p0)
- #pages in IN(p0) and OUT(p0) with same MD5 hash signature as p0
- #characters in domain name of p0
- #pages in IN(p0) with same IP number as p0

IV.6 Online and Distributed Link Analysis

Goals:

- Compute Page-Rank-style authority measures **online**, i.e., without having to store the complete link graph.
- Recompute authority **incrementally** as the graph changes.
- Compute authority in **decentralized**, asynchronous manner with the graph distributed across many peers.

Online Link Analysis

[Abiteboul et al.: WWW 2003]

Key idea:

- Compute small fraction of authority as crawler proceeds **without storing the Web graph**.
- Each page holds some "**cash**" that reflects its importance.
- When a page is visited, it **distributes its cash** among its successors.
- When a page is not visited, it can still accumulate cash.
- This random process has a **stationary limit** that captures the **importance of pages** (but generally not the same as the actual PageRank score).

OPIC Algorithm

(Online Page Importance Computation)

Maintain for each page i (out of n pages):

- **C[i]** – cash that page i currently has and distributes
- **H[i]** – history of how much cash page has ever had in total

Plus global counter:

- **G** – total amount of cash that has ever been distributed

```
G := 0; for each i do { C[i] := 1/n; H[i] := 0 };
do forever {
   choose page i (e.g., by crawling randomly or greedily);
   H[i] := H[i] + C[i];
   for each successor j of i do C[j] := C[j] + (C[i] / outdegree(i));
   G := G + C[i];
   C[i] := 0; };
```

Note: 1) for convergence, every page needs to be visited infinitely often
2) the link graph is assumed to be strongly connected

OPIC Importance Measure

At each step t, an estimate of the importance of page i is:
$X_t[i] = H_t[i] / G_t$ or alternatively: $X_t[i] = (H_t[i] + C_t[i]) / (G_t + 1)$

Theorem:
Let $X_t = H_t / G_t$ denote the vector of cash fractions accumulated by pages until step t.
The limit $X = \lim_{t\to\infty} X_t$ exists with $||X||_1 = \Sigma_i X[i] = 1$.

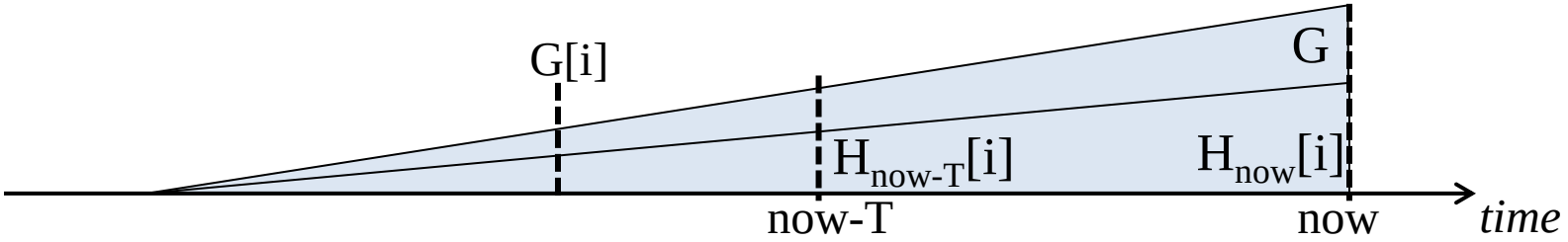
With crawl strategies such as:
- random
- greedy: read page i with highest cash C[i]
  (fair because non-visited pages accumulate cash until eventually read)
- cyclic (round-robin)

Adaptive OPIC for Evolving Link Graph

Consider a time window [now-T, now] where time is the value of G.

The estimated importance of page i is:

$X_{now}[i] = (H_{now}[i] - H_{now-T}[i]) / T$



For a new crawl at time "now",

update page history $H_{now}[i]$ by a simple interpolation:

- Let $H_{now-T}[i]$ be cash acquired by page i until time (now-T)
- $C_{now}[i]$ the current cash of page i
- Let G[i] denote the time G at which i was crawled previously

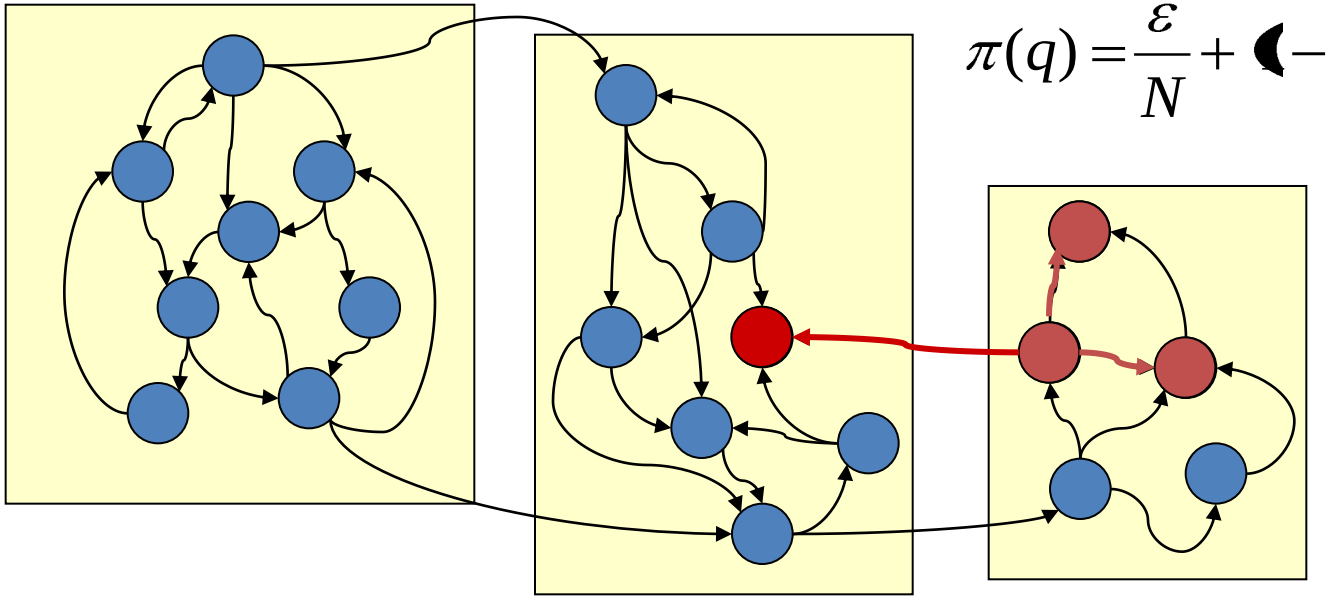
Then set

$$H_{now}[i] := \begin{cases} H_{now-T}[i] \cdot \dfrac{T - (G - G[i])}{T} + C_{now}[i] & \text{if } G - G[i] < T \\ C_{now}[i] \cdot \dfrac{T}{G - G[i]} & \text{otherwise} \end{cases}$$

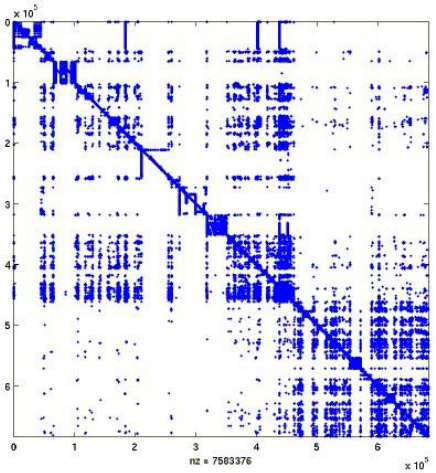
Distributed Link Analysis

Page authority is important for final result scoring.

Exploit locality in Web link graph: construct **block structure** (disjoint graph partitioning) based on sites or domains.



$$\pi(q) = \frac{\varepsilon}{N} + (1-\varepsilon) \sum_{p \in IN(q)} \frac{\pi(p)}{out(p)}$$



Compute page PR within site/domain & across site/domain weights:

- Combine local page scores with site/domain scores.

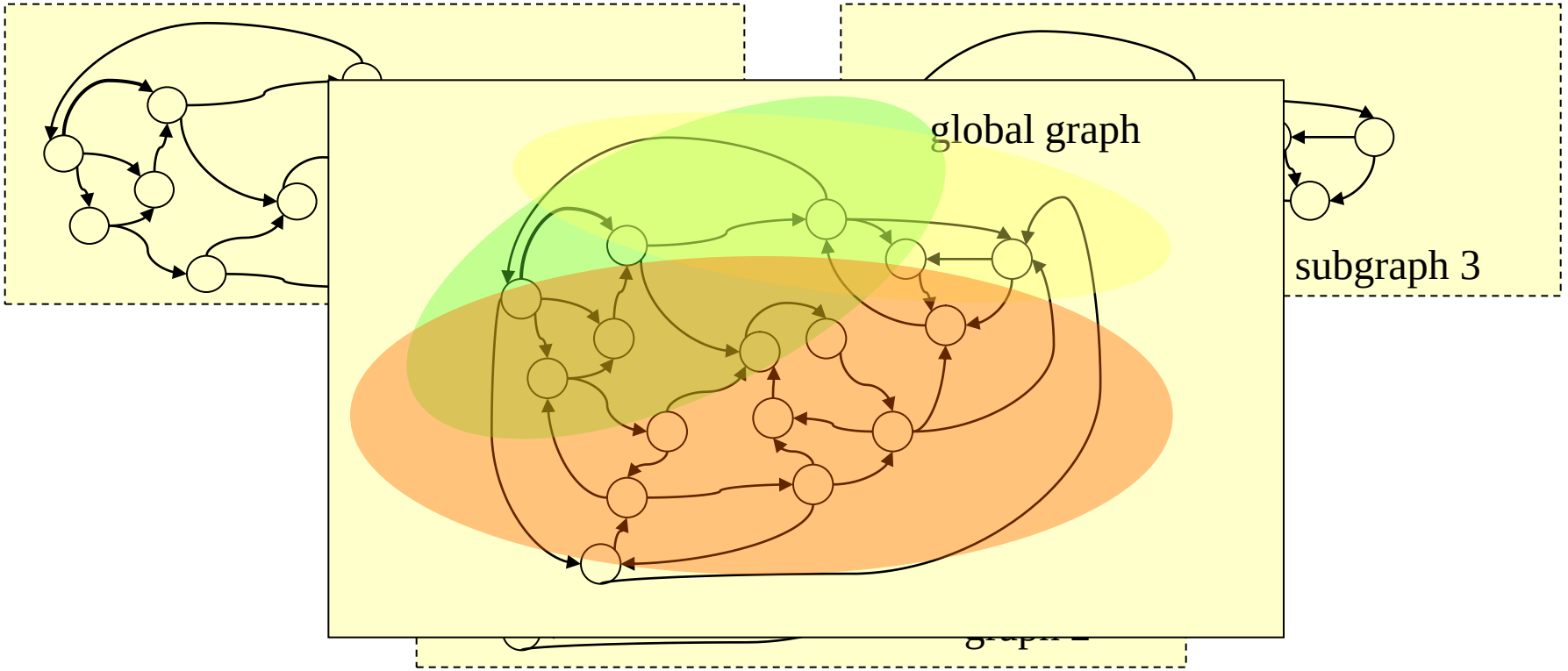
  [Kamvar03, Lee03, Broder04, Wang04, Wu05]

- Communicate PR mass propagation across sites.

  [Abiteboul00, Sankaralingam03, Shi03, Kempe04, Jelasity07]

Decentralized PageRank in P2P Network

Decentralized computation in peer-to-peer network
with arbitrary, a-priori unknown **overlaps of graph fragments**.



Generalizable to graph analysis applied to:

- Pages, sites, tags, users, groups, queries, clicks, opinions, etc. as nodes
- Assessment and interaction relations as weighted edges
- Can compute various notions of authority, reputation, trust, quality

JXP (Juxtaposed Approximate PageRank)

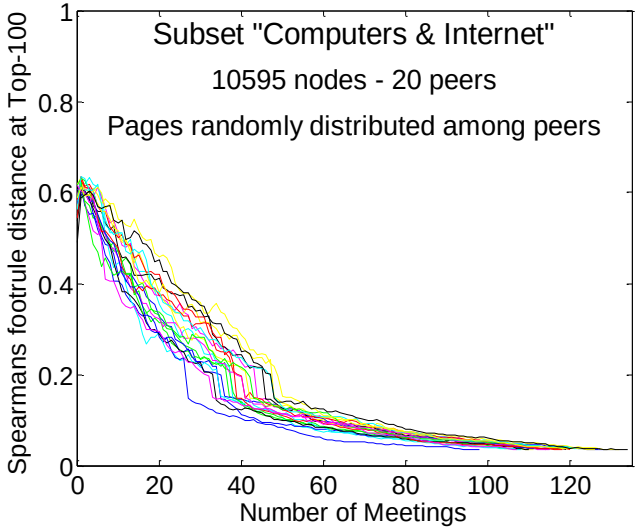
[J.X. Parreira et al.: WebDB 05, VLDB 06, VLDB Journal]

Scalable, decentralized P2P algorithm based on:
Markov-chain aggregation [Courtois 1977, Meyer 1988]

- Each peer represents external, a priori unknown part of the global graph by one superstate: a "**world node**"

Peers meet randomly:

- Exchange local graph fragments & PR vectors
- Learn incoming edges to nodes of local graph
- Compute local PR on enhanced local graph
- Keep only improved PR and own local graph
- Don't keep other peers' graph fragments



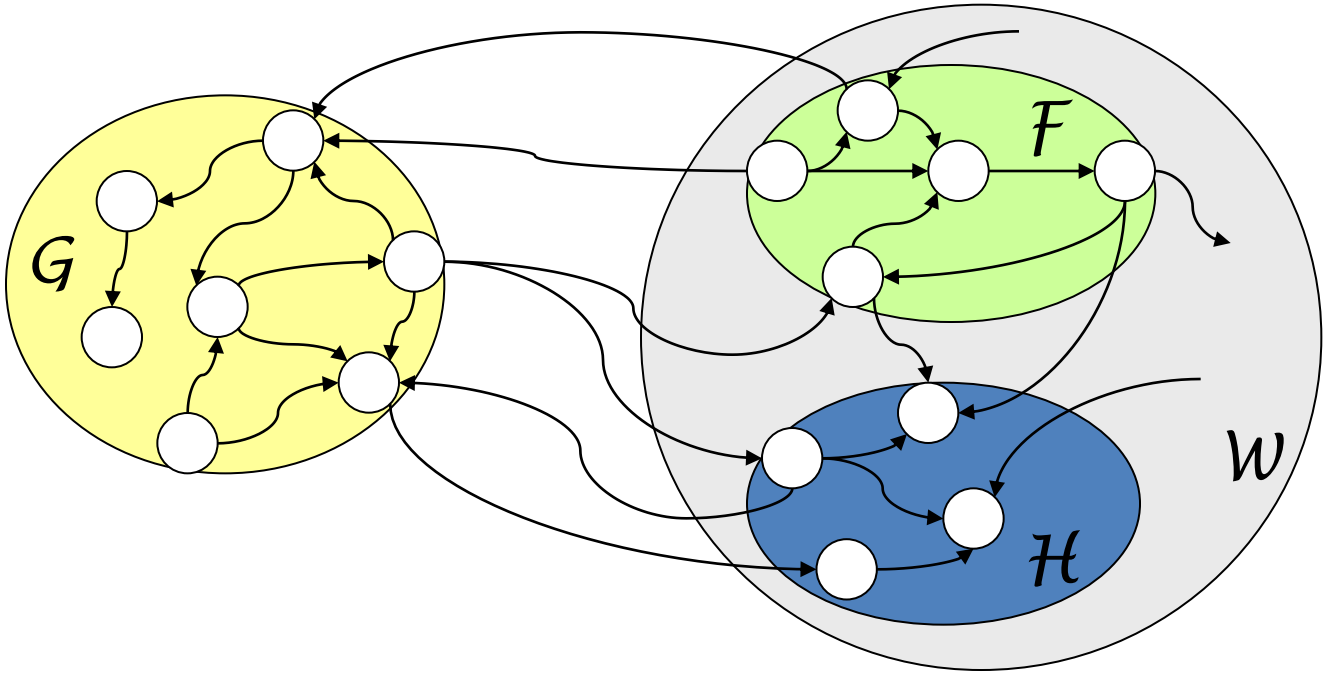
Theorem: JXP scores converge to global PR scores.

Convergence sped up by **biased p2pDating** strategy:
Prefer peers whose node set of outgoing links
has high overlaps with our node set (e.g., Bloom filters as synopses).

JXP Algorithm at Work (1)

Intput:
G: local graph
GOUT: {q∈G | q→ s ∧ s∈W}
n: #pages in G; N: #pages in U = G∪W
WIN(G): {p∈W | p→ q ∧ q∈G}
WIN*(G) ⊆ WIN(G): known part of WIN(G)

Output:
π*(q) for q∈G:
est. stationary prob's (PR)
$\pi^*(G) = \sum_{q \in G} \pi^*(q) = 1 - \pi^*(W)$
est. total mass of G



At each meeting with another peer G, compute:

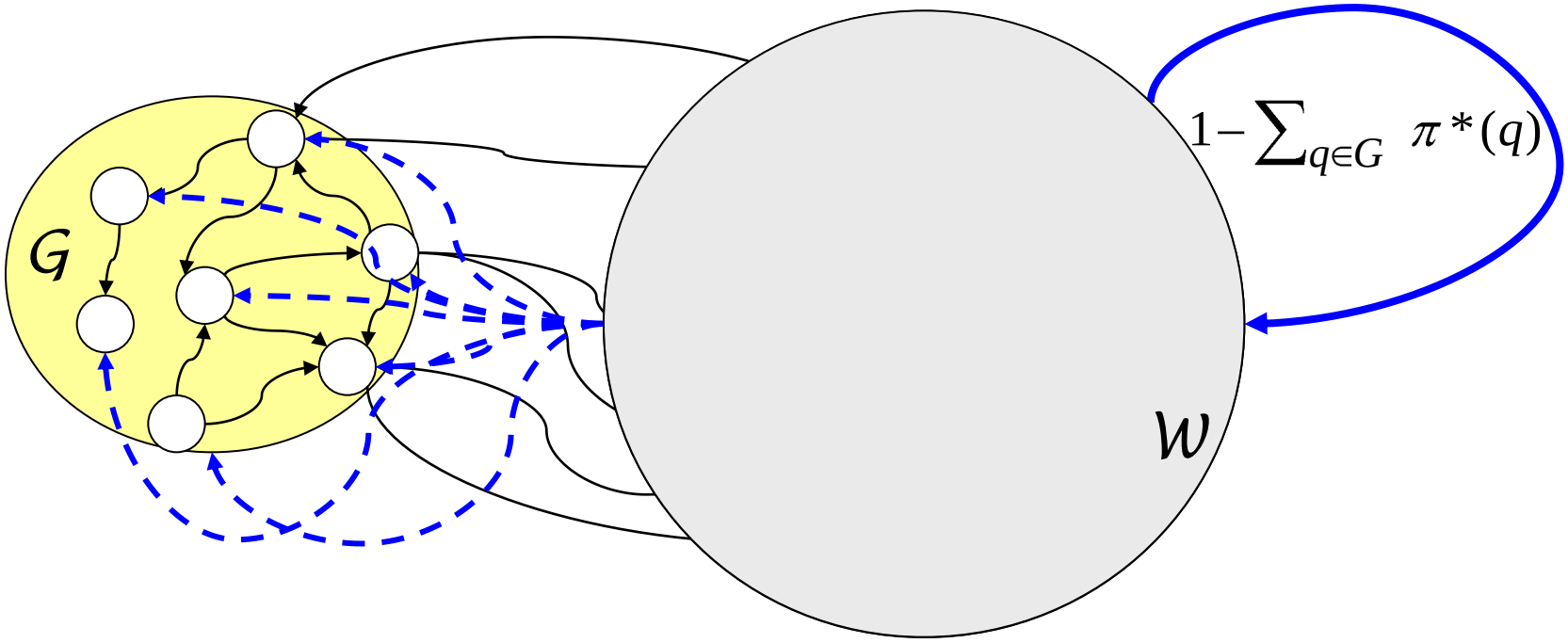
- For all q∈G: $\pi^*(q) = \pi^*(W)\ \varepsilon / N\ + \sum_{p \in WIN^*(G)} \left(\frac{\pi^*(p)}{\pi^*(W)} (1-\varepsilon) / out(p) \right)$
- World self-loop: $\pi^*(W) = 1 - \sum_{q \in G}\ \pi^*(q)$

Compute all π* values for G∪W; remember only WIN*(G) info.

JXP Algorithm at Work (2)

Intput:
G: local graph
GOUT: {q∈G | q→ s ∧ s∈W}
n: #pages in G; N: #pages in U = G∪W
WIN(G): {p∈W | p→ q ∧ q∈G}
WIN*(G) ⊆ WIN(G): known part of WIN(G)

Output:
π*(q) for q∈G:
est. stationary prob's (PR)
π*(G) = $\sum_{q \in G}$ π*(q)=1- π*(W)
est. total mass of G



At each meeting with another peer G, compute:

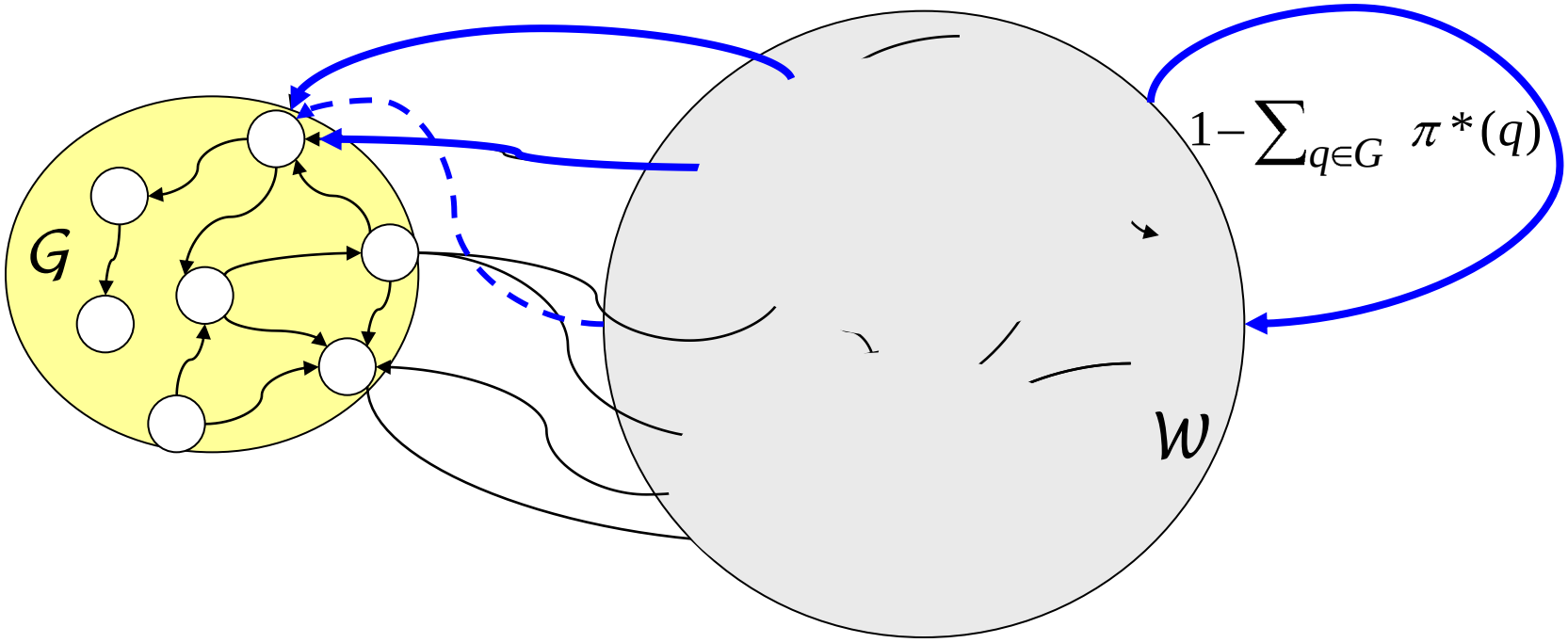
- For all q∈G: $\pi^*(q) = \pi^*(W)\ \varepsilon / N\ + \sum_{p \in WIN^*(G)} \left(\frac{\pi^*(p)}{\pi^*(W)} (1-\varepsilon) / out(p) \right)$
- World self-loop: $\pi^*(W) = 1 - \sum_{q \in G}\ \pi^*(q)$

Compute all π* values for G∪W; remember only WIN*(G) info.

JXP Algorithm at Work (3)

Intput:
G: local graph
GOUT: {q∈G | q→ s ∧ s∈W}
n: #pages in G; N: #pages in U = G∪W
WIN(G): {p∈W | p→ q ∧ q∈G}
WIN*(G) ⊆ WIN(G): known part of WIN(G)

Output:
π*(q) for q∈G:
est. stationary prob's (PR)
π*(G) = $\sum_{q \in G}$ π*(q)=1- π*(W)
est. total mass of G



At each meeting with another peer G, compute:

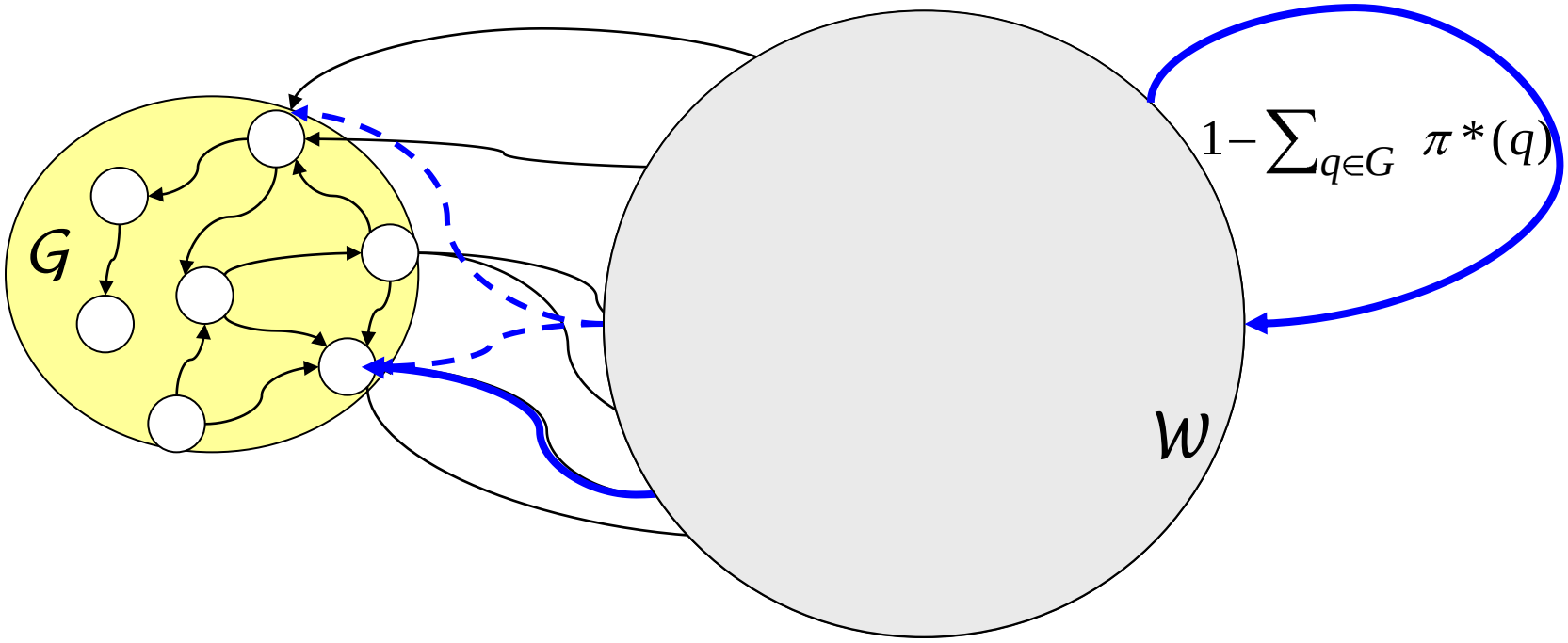
- For all q∈G: $\pi^*(q) = \pi^*(W)\ \varepsilon / N\ + \sum_{p \in WIN^*(G)} \left(\frac{\pi^*(p)}{\pi^*(W)} (1-\varepsilon) / out(p) \right)$
- World self-loop: $\pi^*(W) = 1 - \sum_{q \in G}\ \pi^*(q)$

Compute all π* values for G∪W; remember only WIN*(G) info.

JXP Algorithm at Work (4)

Intput:
G: local graph
GOUT: {q∈G | q→ s ∧ s∈W}
n: #pages in G; N: #pages in U = G∪W
WIN(G): {p∈W | p→ q ∧ q∈G}
WIN*(G) ⊆ WIN(G): known part of WIN(G)

Output:
π*(q) for q∈G:
est. stationary prob's (PR)
π*(G) = $\sum_{q \in G}$ π*(q)=1- π*(W)
est. total mass of G



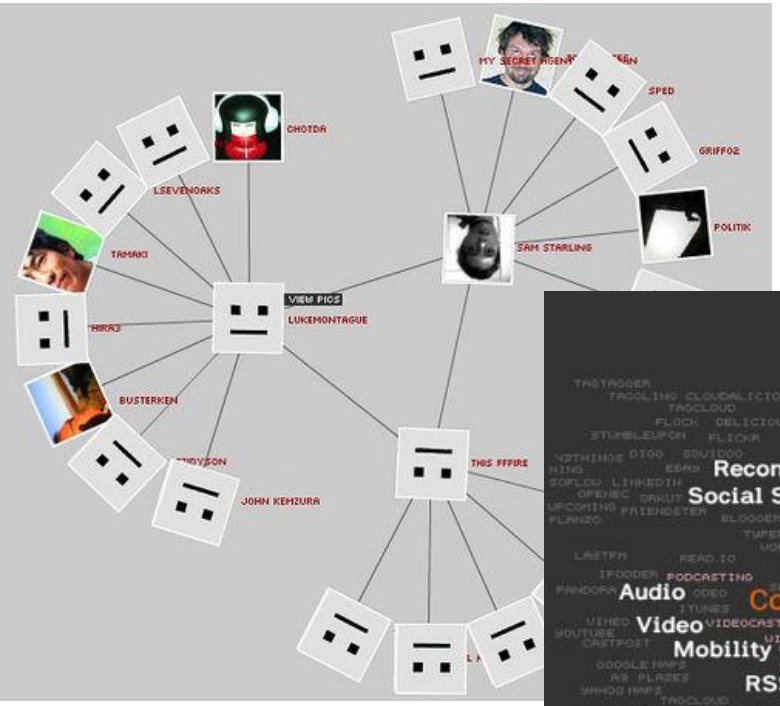
At each meeting with another peer G, compute:

- For all q∈G: $\pi^*(q) = \pi^*(W)\ \varepsilon / N\ + \sum_{p \in WIN^*(G)} \left(\frac{\pi^*(p)}{\pi^*(W)} (1-\varepsilon) / out(p) \right)$
- World self-loop: $\pi^*(W) = 1 - \sum_{q \in G}\ \pi^*(q)$

Compute all π* values for G∪W; remember only WIN*(G) info.

Outlook: Social Networks

http://www.flickr.com/photos/lukemontague/14038129/

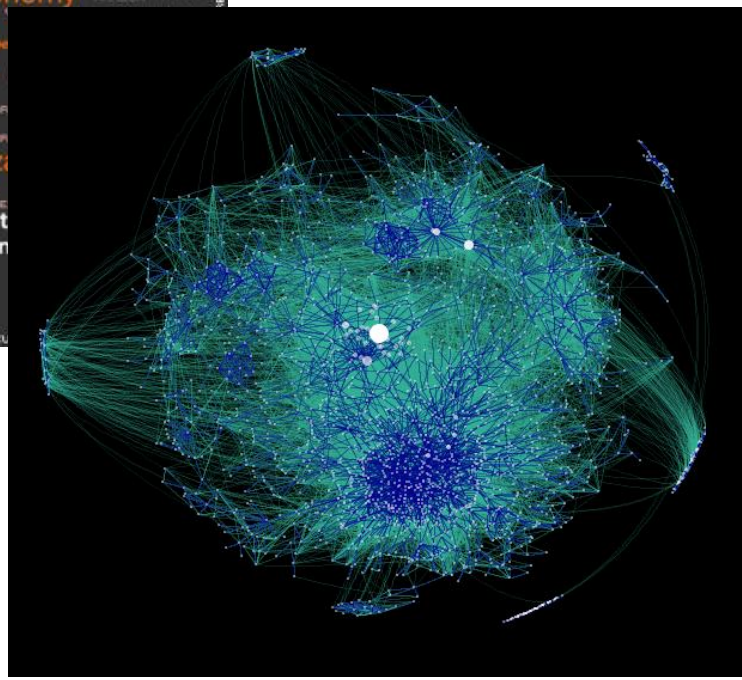


People

Examples:
myspace, facebook, Google+, linkedIn, flickr, del.icio.us, youtube, groups/communities, blogs, etc.

http://www.flickr.com/photos/shopping2null/395271855/

Opinions

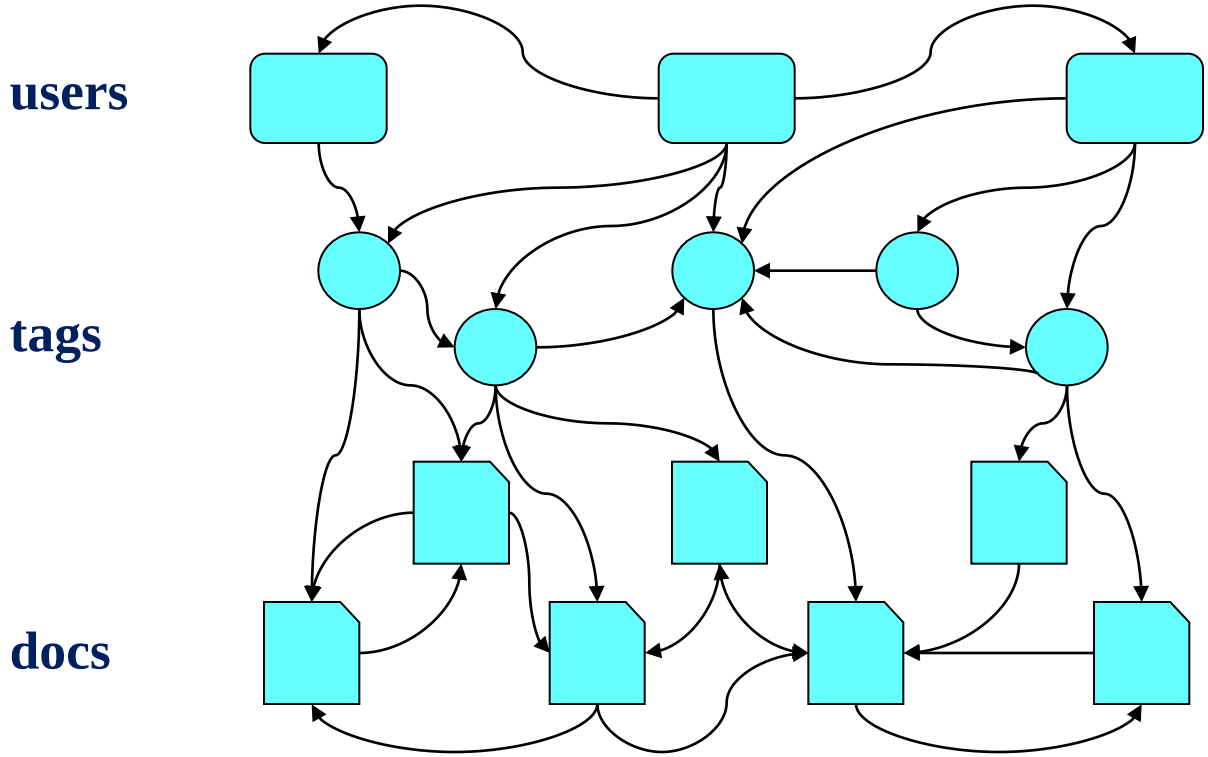


Data

Graphs are everywhere!

http://datamining.typepad.com/gallery/newblog-crop.png

Analyzing Social Networks



Typed graphs: data items, users, friends, groups, postings, ratings, queries, clicks, …
with **weighted edges**

Social-Network Database

Simplified and cast into relational schema:

Users (UId, Nickname, …)

Docs (DId, Author, PostingDate, …)

Tags (TId, String)

Friendship (UId1, UId2, FScore)

Content (DId, TId, Score)

Rating (UId, DId, RScore)

Tagging (UId, TId, DId, TScore)

TagSim (TId1, TId2, TSim)

- Actually several kinds of "friends": same group, fan & star, true friend, etc.
- Tags could be typed or explicitly organized in hierarchies.
- Numeric values for FScore, RScore, TScore, TSim may be explicitly specified or derived from co-occurrence statistics.

Social-Network Graphs

Tagging relation is central:

- Ternary relationship between <users, tags, docs>
- Could be represented as hypergraph (edges connect mult. nodes)
  or (lossfully) decomposed into 3 binary projections (graphs):

UsersTags (UId, TId, UTscore)

x.UTscore := Σ_d {s | (x.UId, x.TId, d, s) $\in$ Ratings}

TagsDocs (TId, DId, TDscore)

x.TDscore := Σ_u {s | (u, x.TId, x.DId, s) $\in$ Ratings}

DocsUsers (DId, UId, DUscore)

x.DUscore := Σ_t {s | (x.UId, t, x.DId, s) $\in$ Ratings}

Authority in Social Networks

Apply link analysis (PR etc.) to appropriately defined matrices!

- **SocialPageRank** [Bao et al.: WWW 2007]:

  Let $\mathbf{M_{UT}}$, $\mathbf{M_{TD}}$, $\mathbf{M_{DU}}$ be the matrices corresponding to relations DocsUsers, TagsDocs, UsersTags

  Compute iteratively:

$$\vec{r}_U = M^T{}_{DU} \times \vec{r}_D$$
$$\vec{r}_D = M^T{}_{TD} \times \vec{r}_T$$
$$\vec{r}_T = M^T{}_{UT} \times \vec{r}_U$$

- **FolkRank** [Hotho et al.: ESWC 2006]:

  Define **graph G as union of graphs** UsersTags, TagsDocs, DocsUsers

  Assume each user has personal preference vector $\vec{p}$

  Compute iteratively:

$$\vec{r}_D = \alpha \vec{r}_D + \beta M_G \times \vec{r}_D + \gamma \vec{p}$$

Search & Ranking with Social Relations

Web search (or search in social network) can benefit from the taste, expertise, experience, recommendations of friends.

Naive method:
Look up your best friend's bookmarks or search with her tags.
→ Combine content scoring with FolkRank, SocialPR, etc.

Better approach:
Integrate friendship strengths, tag similarities, user & page PR, e.g.:

$$s(q,d,u) = \sum_{t \in q} \sum_{c \in SimTags(t)} \sum_{f \in Friends(u)} TScore(f,c,d) \cdot TSim(t,c) \cdot FScore(u,f) \cdot UR(f) \cdot PR(d)$$

Additional Literature for Chapter IV.5

Spam-Resilient Authority Scoring:

- Z. Gyöngyi, H. Garcia-Molina: Spam: It‘s Not Just for Inboxes Anymore, IEEE Computer 2005
- Z. Gyöngyi, P. Berkhin, H. Garcia-Molina, J. Pedersen: Link Spam Detection based on Mass Estimation, VLDB‘06
- Z. Gyöngyi, H. Garcia-Molina: Combating Web Spam with TrustRank, VLDB‘04
- D.Fetterly, M.Manasse, M.Najork: Spam, Damn Spam, and Statistics, WebDB‘05
- I. Drost, T. Scheffer: Thwarting the Nigritude Ultramarine: Learning to Identify Link Spam, ECML‘05
- A.A. Benczur, K. Csalongany, T. Sarlos, M. Uher: SpamRank – Fully Automatic Link Spam Detection, AIRWeb Workshop 2005
- R. Guha, R. Kumar, P. Raghavan, A. Tomkins: Propagation of Trust and Distrust, WWW 2004
- C. Castillo, D. Donato, A. Gionis, V. Murdock, F. Silvestri: Know your neighbors: web spam detection using the web topology, SIGIR 2007
- L. Becchetti, C. Castillo, D. Donato, R.A. Baeza-Yates, S. Leonardi: Link analysis for Web spam detection. TWEB 2(1): (2008)
- Workshop on Adversarial Information Retrieval on the Web, http://airweb.cse.lehigh.edu/

Additional Literature for Chapter IV.6

Online and Distributed Link Analysis:

- S. Abiteboul, M. Preda, G. Cobena: Adaptive on-line page importance computation, WWW 2003.
- J.X. Parreira, D.Donata, C. Castillo, S. Michel, G. Weikum: The JXP Method for Robust PageRank Approximation in a Peer-to-Peer Web Search Network, VLDB Journal 2008
- D. Kempe, F. McSherry: A decentralized algorithm for spectral analysis. STOC'04
- A.Z. Broder, R. Lempel, F. Maghoul, J.O. Pedersen: Efficient PageRage Approximation via Graph Aggregation. Inf. Retr. 9(2), 2006

Ranking in Social Networks:

- S. Bao, X. Wu, B. Fei, G. Xue, Z. Su, Y. Yu: Optimizing Web Search Using Social Annotations, WWW 2007
- Christoph Schmitz, Andreas Hotho, Robert Jäschke, Gerd Stumme: Content Aggregation on Knowledge Bases Using Graph Clustering. ESWC 2006
- Andreas Hotho, Robert Jäschke, Christoph Schmitz, Gerd Stumme: FolkRank : A Ranking Algorithm for Folksonomies. LWA 2006

Summary of Chapter IV

- **PageRank**, **HITS**, etc. are major achievements for better Web search.
- Improvements compared to in-/out-degree mostly for **highly specific queries**, best results with **good content ranking** function.
- Link analysis built on **well-founded theory**, but full understanding of sensitivity and special properties still missing.
- **Personalized link analysis** is promising and viable.
- **Link spam** is major problem; addressed by statistical methods (but may need deeper adversary theory).
- **Online and distributed link analysis** practically viable.
- Link analysis has potential for **generalization to social networks**.