Microsoft Research

Ranking on Large-Scale Graphs with Rich Metadata

Bin Gao, Taifeng Wang, and Tie-Yan Liu
Microsoft Research Asia

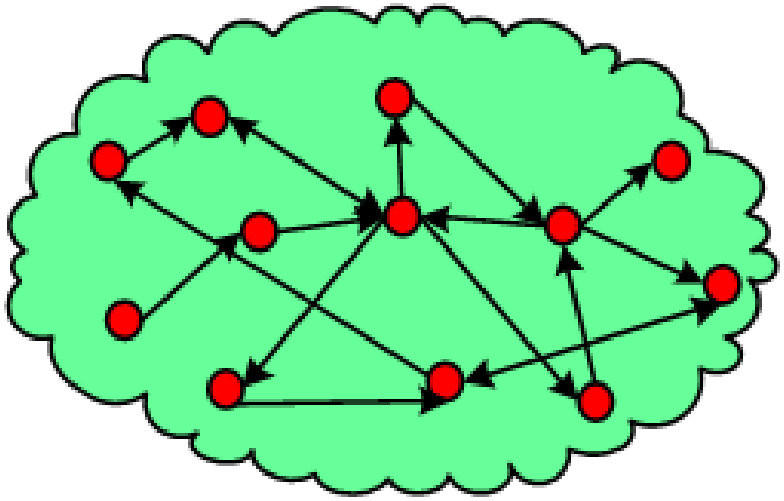
Presenters

Bin Gao
Researcher, MSR Asia
http://research.microsoft.com/en-us/people/bingao/

Taifeng Wang
Researcher, MSR Asia
http://research.microsoft.com/en-us/people/taifengw/

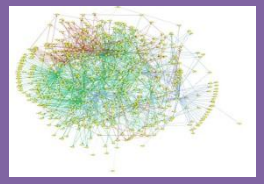
Tie-Yan Liu
Lead Researcher, MSR Asia
http://research.microsoft.com/users/tyliu/

Graph



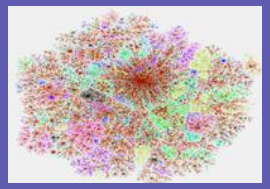
Everything in the world is connected.
There is graph where there is connection.

Large-scale Graph



Social graphs

- Messenger, Facebook, Twitter, Entity Cube, etc.

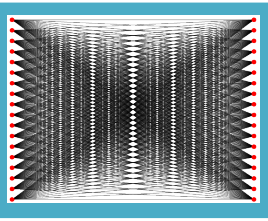


Endorsement graphs

- Web link graph, Paper citation graph, etc.

Location graphs

- Map, Power grid, Telephone network, etc.



Co-occurrence graphs

- Term-document bipartite, Click-through bipartite, etc.

How Large Are These Graphs?

- Web Link Graph
  - Tens of billions of nodes indexed and Over one trillion nodes discovered by major search engines
- Facebook
  - About 600 million nodes (14-Jan-2011)
- China Telephone Networks
  - 1.1 billion nodes (0.8 billion mobile + 0.3 billion land line) (22-Jul-2010)
- Click-through Bipartite
  - Several billion queries and tens of billion URLs (recent research papers)

Properties of Real Large-scale graphs

- Large-scale, of course ☺
- Very sparse
- Rich information on nodes and edges
- External knowledge on the graphs

Rich Metadata

Rich Information on Node & Edge

- Web Link graph
  - Node: page length, creation time, etc.
  - Edge: number of links, inter/intra-site link, etc.
- Facebook
  - Node: age, gender, interests, etc.
  - Edge: creation time, communication frequency, etc.
- China Telephone Networks
  - Node: service category, customer profile, etc.
  - Edge: communication frequency, bandwidth, types of calls, etc.
- Click-through Bipartite
  - Node: query frequency, language, page length, page importance, dwell time, etc.
  - Edge: click frequency, time of click, etc.

External Knowledge

- Point-wise
  - Entity A is popular.
  - Entity B is a spam.
- Pair-wise
  - Entity A is more important than entity B.
- List-wise
  - We have A > B > C, according to the user feedback on these entities.

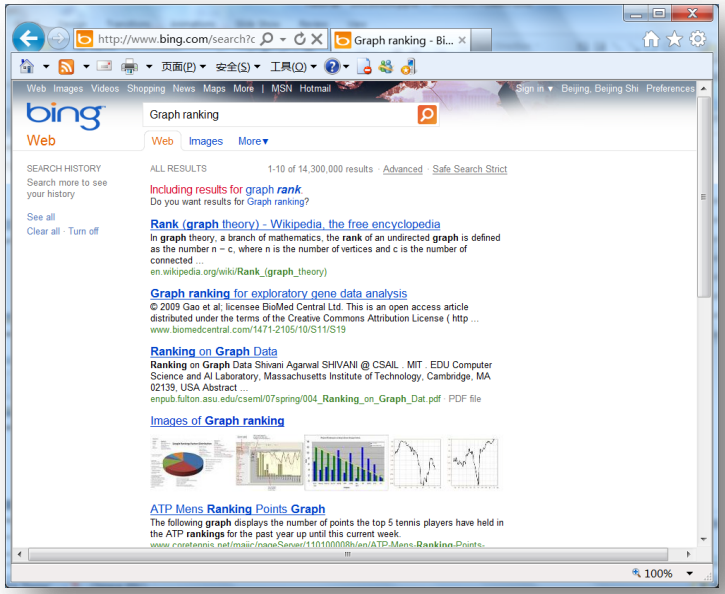
** Here entity can be website, people, phone subscriber, query, etc.*

Ranking on Large-scale Graph

- Problem definition
  - Given a large-scale directed graph and its rich metadata, calculate the ranking of the nodes in the graph according to their importance, popularity, or preference.
- Application
  - Webpage ranking
  - Paper ranking
  - Entity ranking in social networks

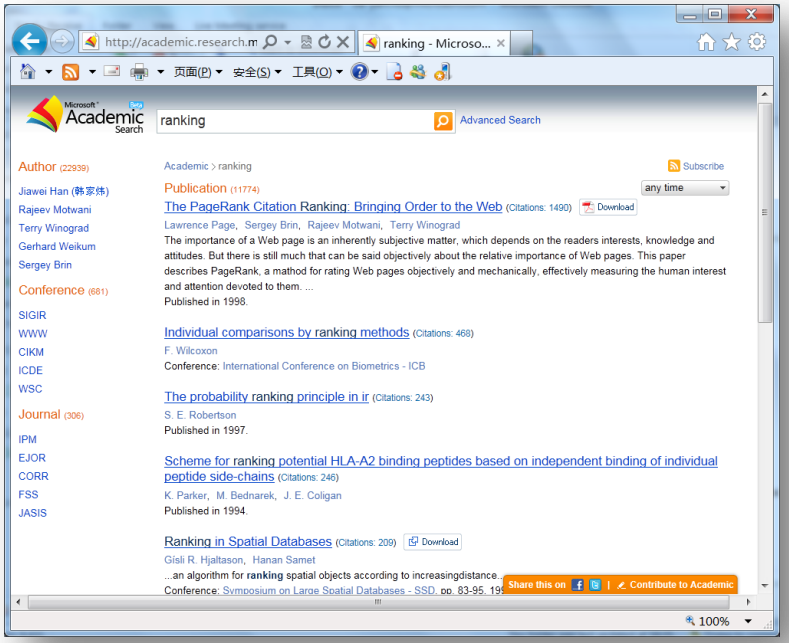
Example: Web Page Ranking

- Factors to consider
  - The quality of the web page
  - The visit frequency by users
  - User's dwell time
  - The mutual endorsement between pages
  - …



Example: Paper Ranking

- Factors to consider
  - Citation
  - Authors
  - Publication venue
  - Awards
  - Publication date
  - …



Example: Social Entity Ranking

- Factors to consider
  - Account creation time
  - Account activity
  - Friends related information
  - Liked or followed by others
  - … …

Key Questions to Answer

1. How to perform graph ranking based on graph structure?
2. How to leverage node and edge features for better graph ranking?
3. How to incorporate external knowledge in graph ranking?
4. How to implement large-scale graph ranking algorithms?

Scope of the Tutorial

- Node ranking on graphs
  - But not ranking of a number of graphs
  - But not retrieval and ranking problems for subgraphs
- Mainly based on papers at WWW, SIGIR, KDD, ICML
  - Papers at other conferences and journals might not be well covered
  - Not necessarily a comprehensive review of the literature
  - Your are welcome to contribute by sharing and discussing with us and our audience

Background Knowledge Required

- Information Retrieval
- Machine Learning
- Linear Algebra
- Probability Theory
- Optimization

We assume that you are familiar with these fields, and we will not give comprehensive introduction to them in this tutorial.

Notations

- Graph $G(V, E, X, Y)$
  - $V = \{v_i\}$: node set, $|V| = n$
  - $E = \{e_{ij}\}$: edge set, $|E| = m$
  - $X = \{x_{ij}\}$: edge features, $|x_{ij}| = l$, $x_{ij} = (x_{ij1}, x_{ij2}, \dots, x_{ijl})^T$
  - $Y = \{y_i\}$: node features, $|y_i| = h$, $y_i = (y_{i1}, y_{i2}, \dots, y_{ih})^T$
- Matrices
  - M: adjacency matrix or link matrix
  - P: transition probability matrix
- Rank score vectors
  - a: authority score vector
  - h: hub score vector
  - π: general rank score vector

Outline

I. Overview

II. Graph Ranking by Link Analysis

III. Graph Ranking with Node and Edge Features

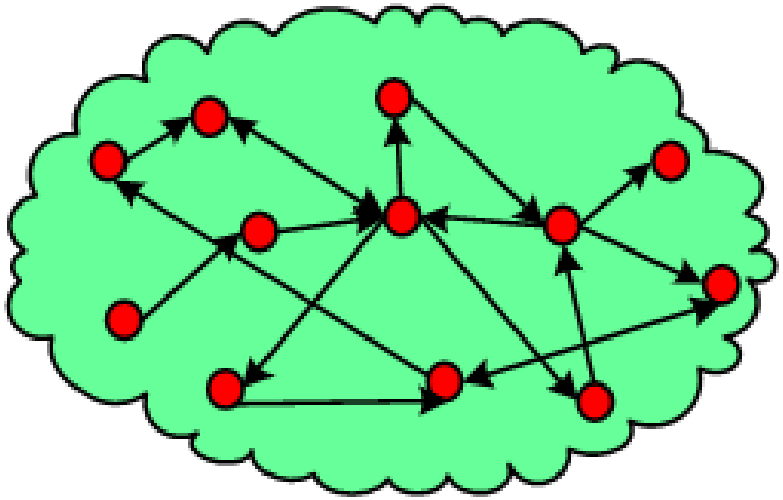
IV. Graph Ranking with Supervision

V. Implementation for Graph Ranking

VI. Summary

Link Analysis for Ranking

$G(V, E)$



- Only consider link structure, no metadata involved.
- A link from page v_i to page v_j may indicate:
  - v_i is related to v_j
  - v_i is recommending, citing, voting for, or endorsing v_j

Famous Link Analysis Algorithms

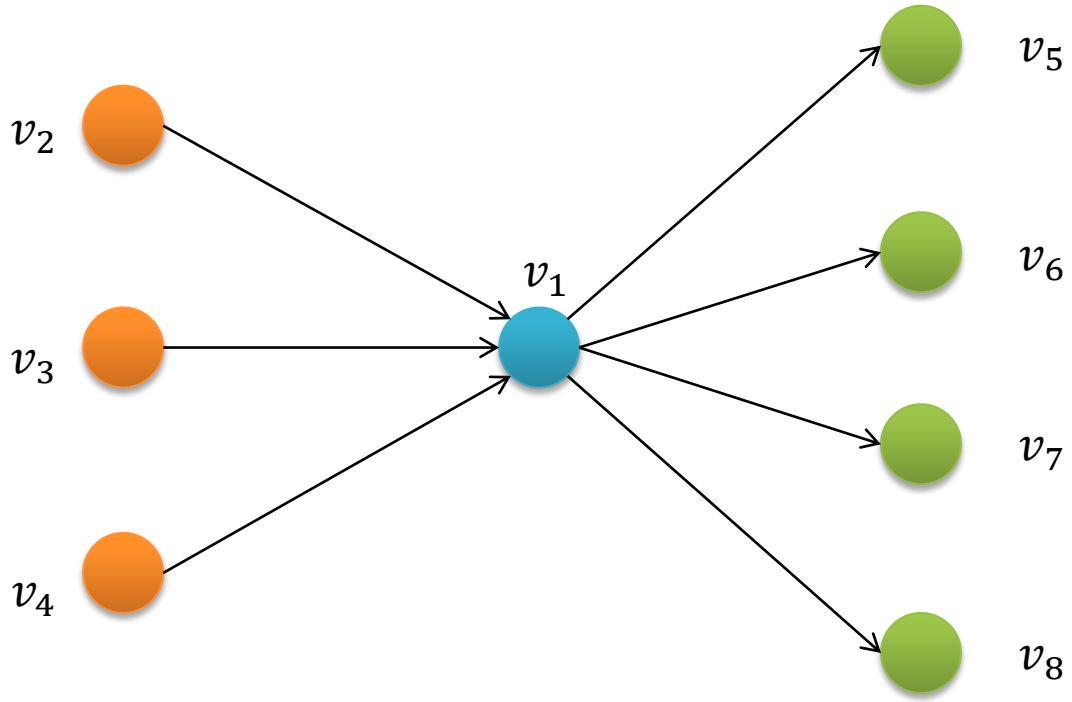
- HITS [Kleinberg, 1997]
- PageRank [Page et al, 1998]

HITS – Hypertext Induced Topic Selection

- For each vertex v_i in a subgraph of interest
  - $a(v_i)$ - the authority of v_i
  - $h(v_i)$ - the hub of v_i
- Authority
  - A site is very authoritative if it receives many citations. Citation from important sites weights more than citations from less-important sites.
- Hub
  - Hub shows the importance of a site. A good hub is a site that links to many authoritative sites.

Kleinberg. Authoritative sources in a hyperlinked environment. IBM Research Report RJ 10076, 1997.

Authority and Hub



$$a(v_1) = h(v_2) + h(v_3) + h(v_4)$$

$$h(v_1) = a(v_5) + a(v_6) + a(v_7) + a(v_8)$$

Convergence of HITS

- Recursive dependency

$$a(v_i) = \sum_{v_j \in inlink[v_i]} h(v_j) \qquad h(v_i) = \sum_{v_j \in outlink[v_i]} a(v_j)$$

- Iterative algorithm

$$a^{(k+1)}(v_i) = \sum_{v_j \in inlink[v_i]} h^{(k)}(v_j)$$

$$h^{(k+1)}(v_i) = \sum_{v_j \in outlink[v_i]} a^{(k+1)}(v_j)$$

$$a^{(k+1)}(v_i) \leftarrow \frac{a^{(k+1)}(v_i)}{\sum_j a^{(k+1)}(v_j)} \qquad h^{(k+1)}(v_i) \leftarrow \frac{h^{(k+1)}(v_i)}{\sum_j h^{(k+1)}(v_j)}$$

- Using linear algebra, it is easy to prove that $a(v_i)$ and $h(v_i)$ converge.

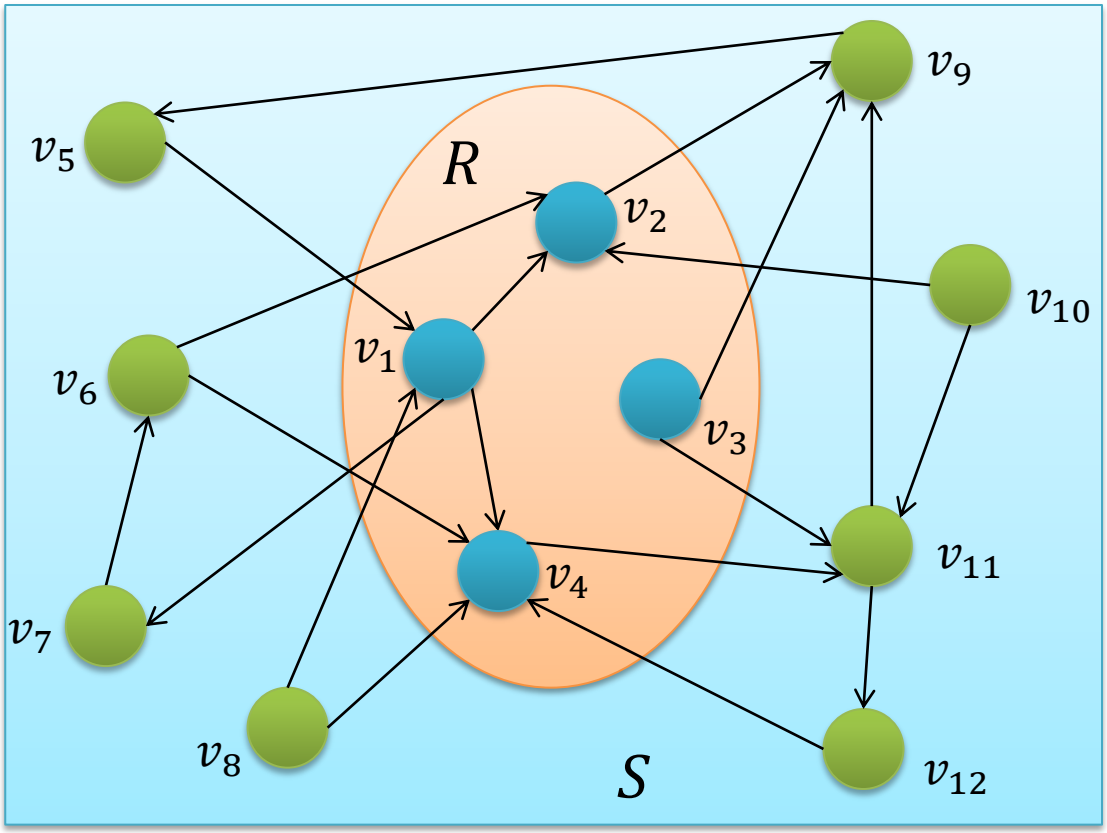
Convergence of HITS

- The authority and hub values calculated by HITS is the left and right singular vectors of the adjacency matrix of the base subgraph.

$$\begin{aligned} a &= M^T h \\ h &= Ma \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} a &= M^T M a \\ h &= M M^T h \end{aligned}$$

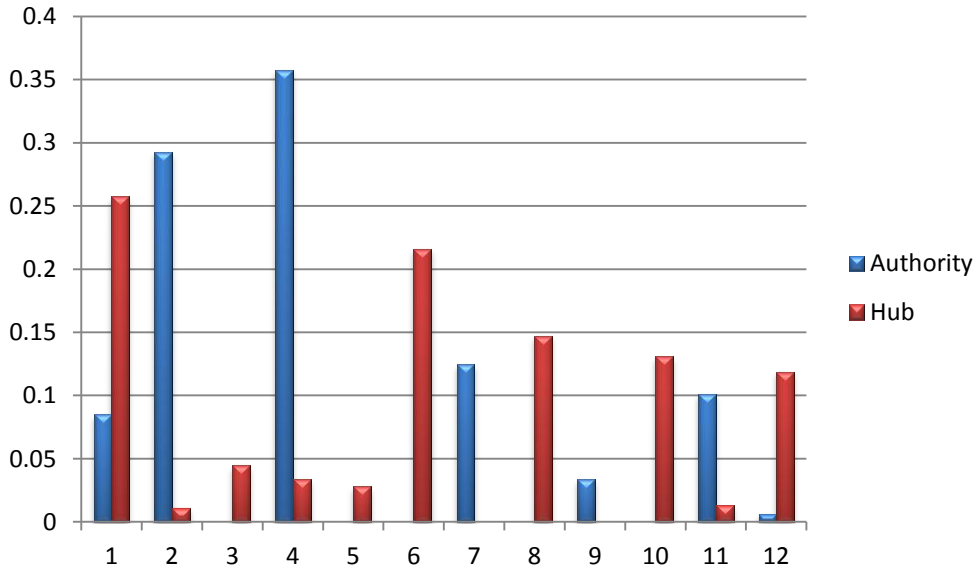
where $a = \left(a(v_1), a(v_2), \ldots, a(v_n)\right)^T$, $h = \left(h(v_1), h(v_2), \ldots, h(v_n)\right)^T$

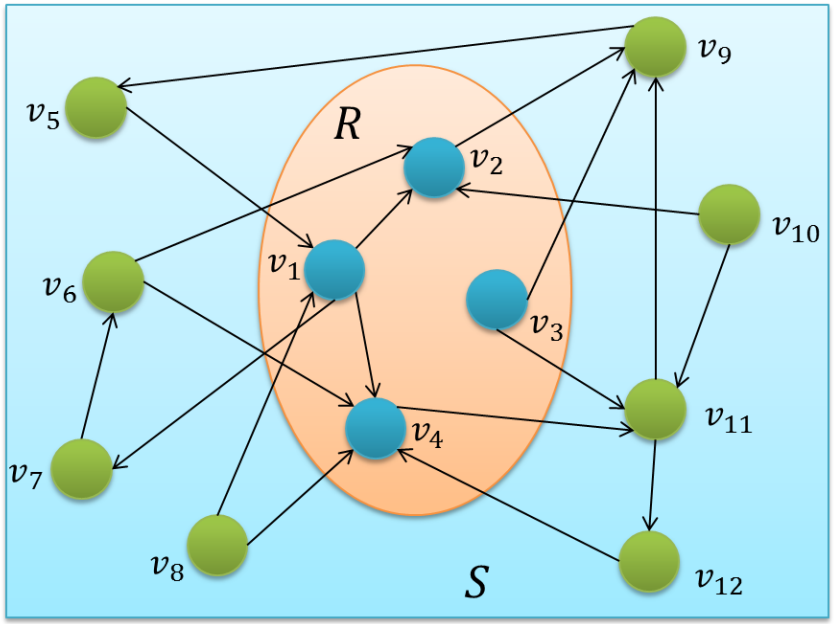
An Example



- Start with a root set $R = \{v_1, v_2, v_3, v_4\}$ by nodes relevant to the topic.
- Generate a new set S (base subgraph) by expanding R to include all the children and a fixed number of parents of nodes in R.

HITS of the Example

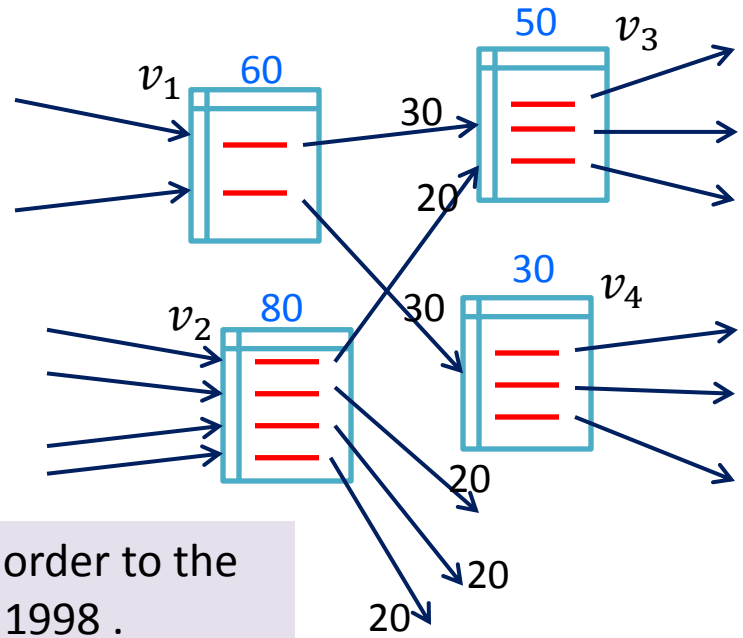




PageRank

- An interesting name!
  - The rank of a page or the rank defined by Mr. Page? ☺
- The page rank is proportional to its parents' rank, but inversely proportional to its parents' out-degrees.

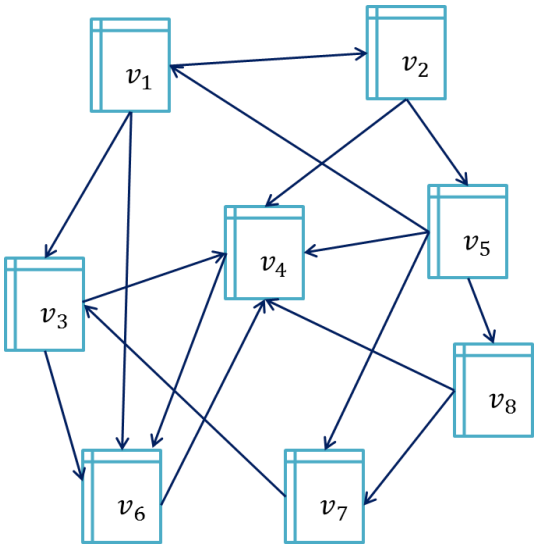
$$\pi(v_i) = \sum_{v_j \in inlink[v_i]} \frac{\pi(v_j)}{|outlink[v_j]|}$$

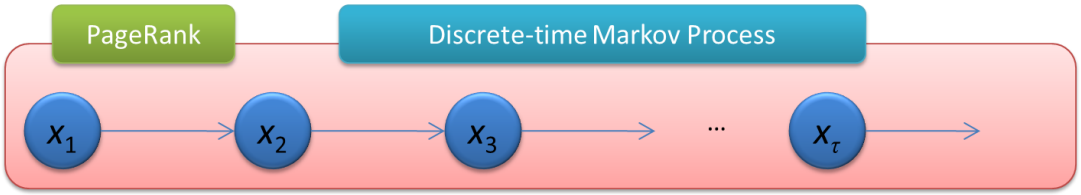


Page et al. The PageRank citation ranking: bringing order to the Web. Stanford Digital Library Technologies Project, 1998 .

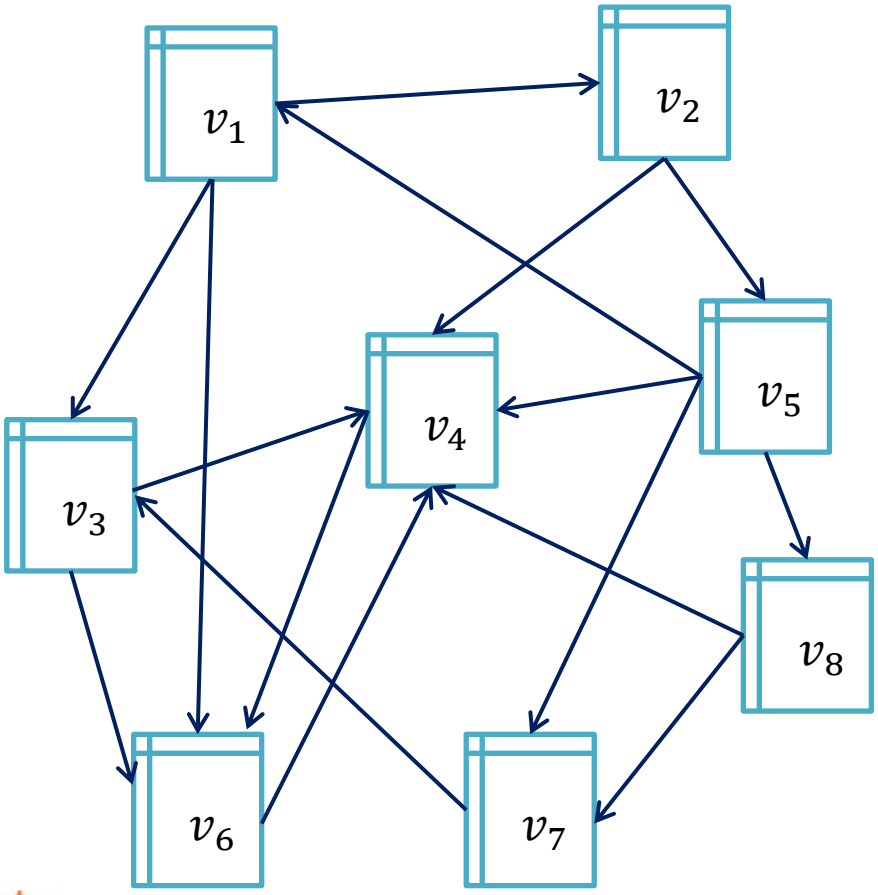
Markov Chain Explanation

- PageRank as a Random Surfer Model
  - Description of a random walk through the Web graph
  - Interpreted as a transition matrix with asymptotic probability that a surfer is currently browsing that page
  - Discrete-time Markov model



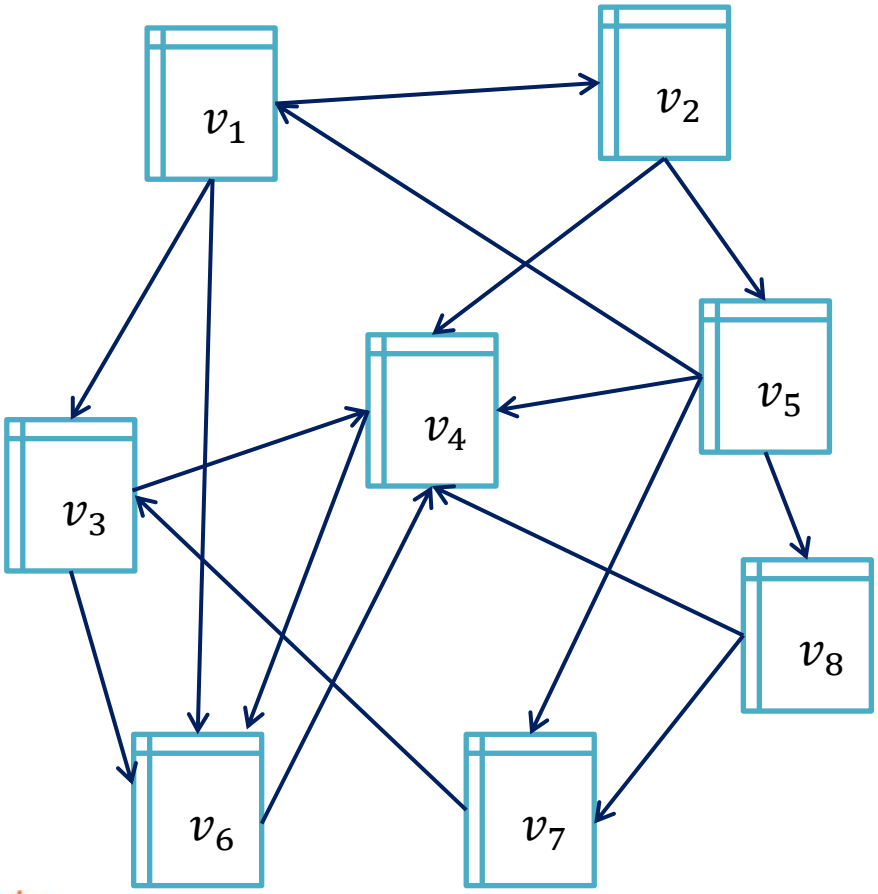


An Example



Node	Outlinks
v_1	v_2, v_3, v_6
v_2	v_4, v_5
v_3	v_4, v_6
v_4	v_6
v_5	v_1, v_4, v_7, v_8
v_6	v_4
v_7	v_3
v_8	v_4, v_7

Matrix Representation



Adjacent Matrix

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

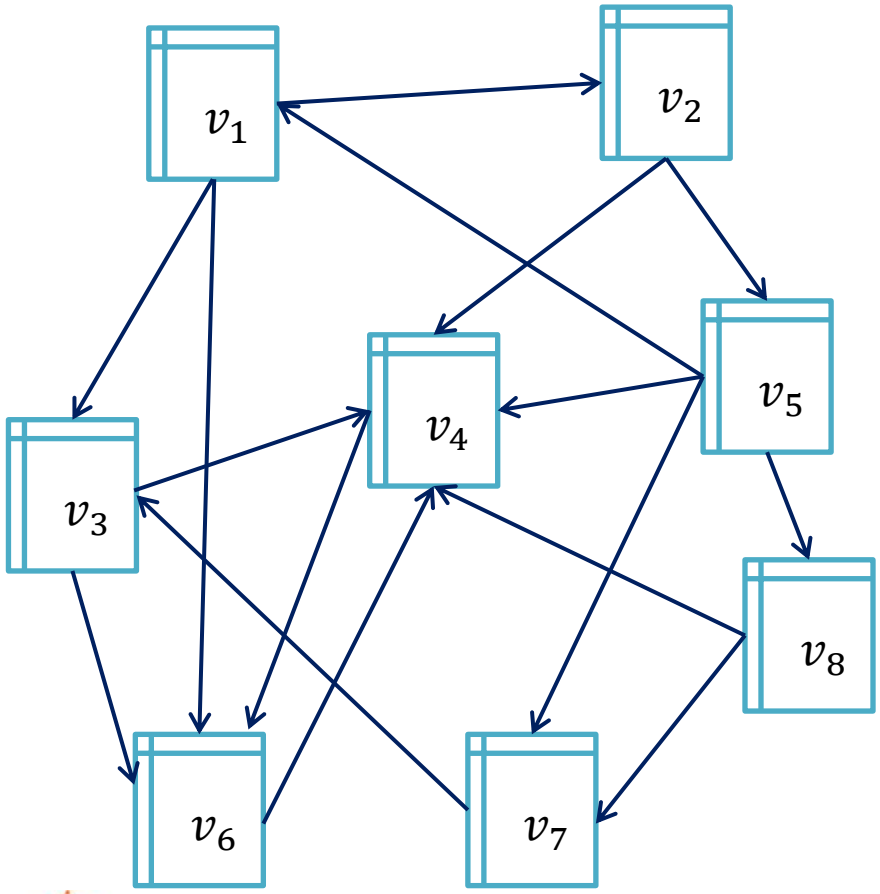
Matrix Representation

- Transition Probability Matrix $P = \{p_{ij}\}$

$$p_{ij} = \begin{cases} \dfrac{M(i,j)}{\sum_{v_{k \in outlink[v_i]}} M(i,k)}, & outlink[v_i] \neq 0 \\ M(i,j) = 0, & otherwise \end{cases}$$

$$P = \begin{bmatrix} 0 & 1/3 & 1/3 & 0 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1/4 & 0 & 0 & 1/4 & 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

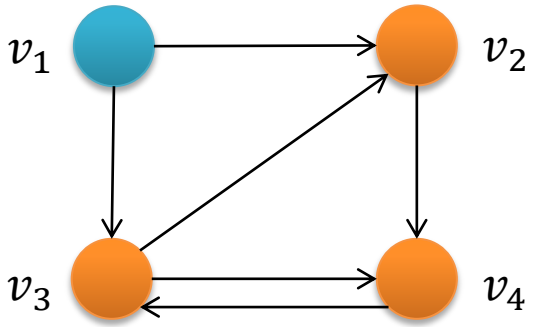
PageRank of the Example



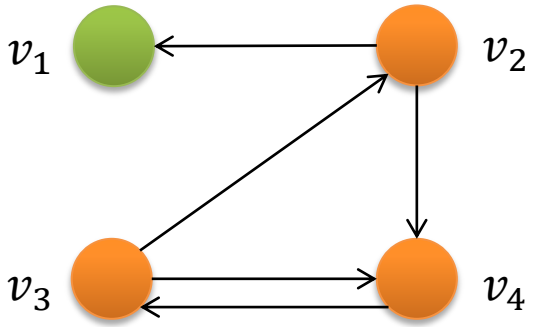
ID	PR	Inlink	Outlink
1	0.0250	v_5	v_2, v_3, v_6
2	0.0259	v_1	v_4, v_5
3	0.0562	v_1, v_7	v_4, v_6
4	0.4068	v_2, v_3, v_5, v_6, v_8	v_6
5	0.0298	v_2	v_1, v_4, v_7, v_8
6	0.3955	v_1, v_3, v_4	v_4
7	0.0357	v_5, v_8	v_3
8	0.0251	v_5	v_4, v_7

Rank Sink

- Many Web pages have no inlinks/outlinks
- It results in dangling edges in the graph
  - No inlink



  - No outlink



Modification – Transition Matrix

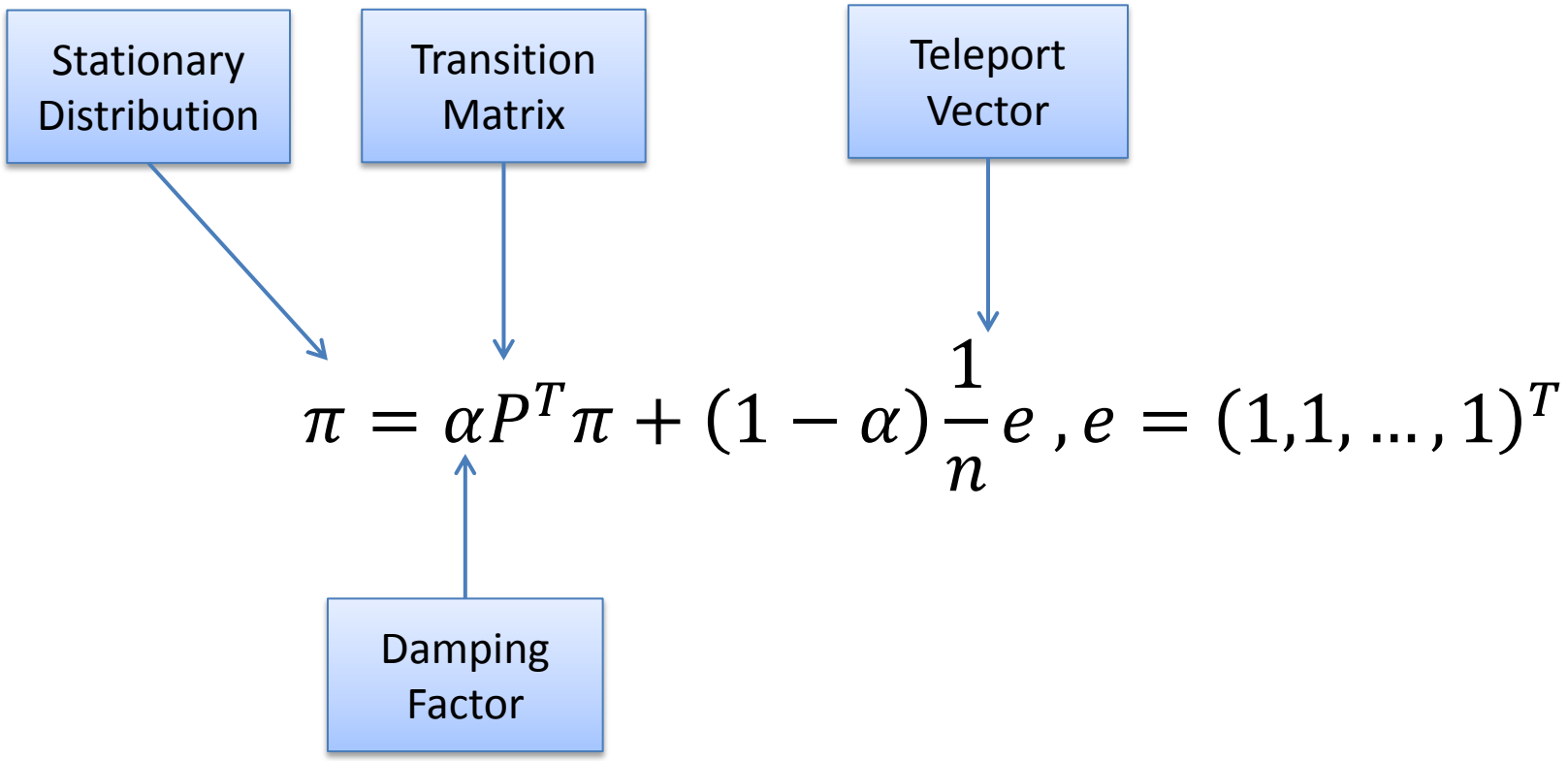
- Surfer will restart browsing by picking a new Web page at random

$P \leftarrow P + E$

P: stochastic matrix

$$E_{ij} = \begin{cases} 0, & if\,|outlink[v_i]| > 0 \\ \frac{1}{n}, & otherwise \end{cases}$$

Further Modification – Damping Factor



Remark: $\pi = P^T \pi$ for simplicity

Other Link Analysis Algorithms

- Following the success of PageRank and HITS, a lot of new algorithms were proposed.
  - Block-level PageRank
  - HostRank
  - ……

Block-Level PageRank

- Web page can be divided into different vision-based segmentation (block)

Block-level PageRank

- Block-to-page matrix W
  - s_i: number of pages the block links to

$$W_{ij} = \begin{cases} \frac{1}{s_i}, & if\ there\ is\ a\ link\ from\ block\ b_i\ to\ page\ v_j \\ 0, & otherwise \end{cases}$$

- Page-to-block matrix U

$$f_{v_i}(b_j) = \beta \frac{Size\ of\ block\ b_j\ in\ page\ v_i}{Distance\ from\ the\ center\ of\ b_j\ to\ the\ center\ of\ screen}$$

$$U_{ij} = \begin{cases} f_{v_i}(b_j), & b_j \in v_i \\ 0, & b_j \notin v_i \end{cases}$$

Cai et al. Block-level link analysis. SIGIR, 2004.

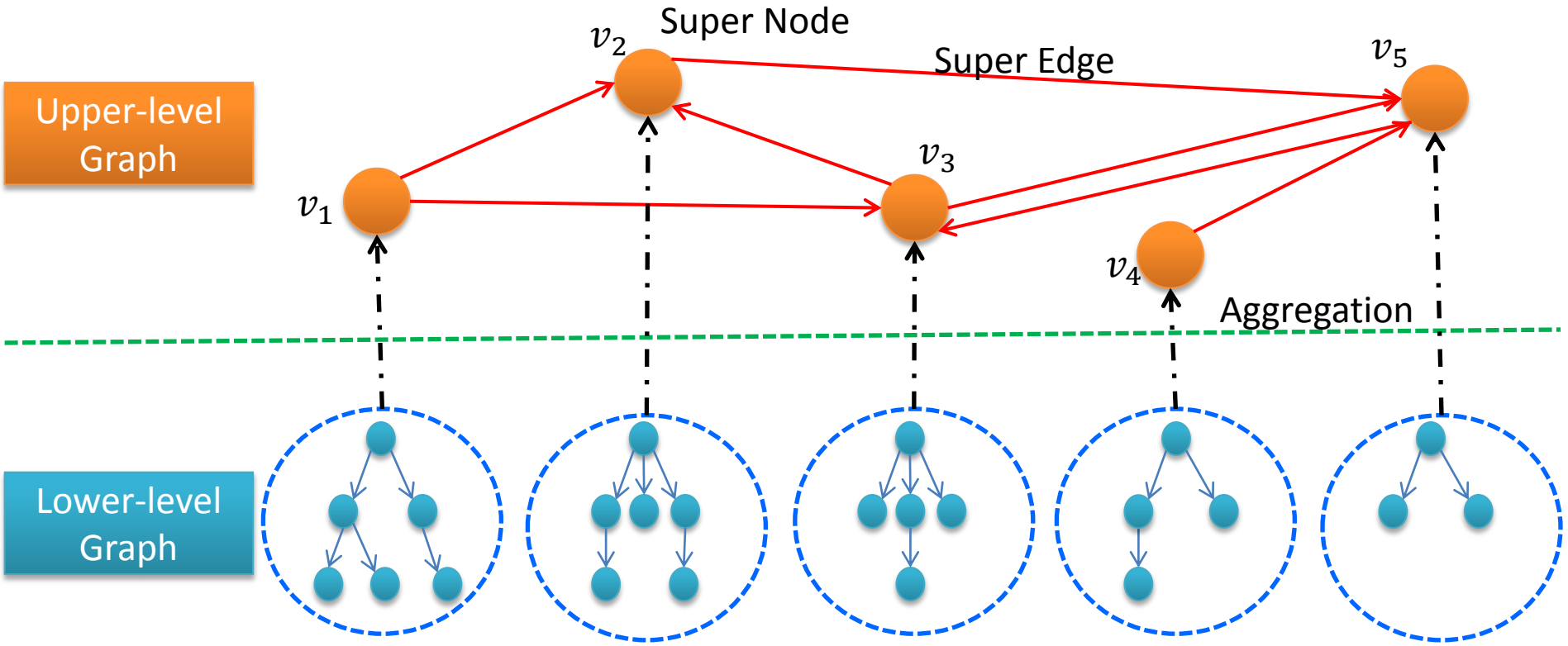
Block-level PageRank (cont.)

- A weight matrix can be defined as UW
- A probability transition matrix $P^{(b)}$ can be constructed by renormalizing each row of UW to sum to 1.
- Block-level PageRank can be computed as

$$\pi = \alpha(P^{(b)})^T\pi + (1-\alpha)\frac{1}{n}e$$

HostRank

- The Web graph has a hierarchical structure.



Xue et al. Exploiting the Hierarchical Structure for Link Analysis. SIGIR, 2004.

HostRank

- Construct two-layer hierarchical graph
  - $S = \{S_1, S_2, \ldots, S_\kappa\}$ is a partition on the vertex set V of graph $G(V, E)$
  - Upper-layer graph contains κ vertices called supernodes, one for each element of the partition
  - Lower-layer graph organizes all the pages in one supernode by the node relationship.

HostRank (cont.)

- Calculate supernode importance $S^{(I)}$

$$S^{(I)} = \alpha (P^{(I)})^T S^{(I)} + (1-\alpha)\frac{1}{n}e$$

- Calculate page importance π

$$index(v_j) = \begin{cases} 1, & if\ v_j\ is\ an\ index\ page \\ \delta, & otherwise \end{cases}$$

$OIL(v_j)$: Number of inter-link to v_j

$IIL(v_j)$: Number of intra-link to v_j

$$link(v_j) = \beta \frac{OIL(v_j)}{\sum_{v_k \in S_j} OIL(v_k)} + (1-\beta)\frac{IIL(v_j)}{\sum_{v_k \in S_j} IIL(v_k)}$$

$$w_j = \eta\ index(v_j) + (1-\eta) link(v_j)$$

$$w_{ij} = \prod_{v_k \in \{nodes\ from\ v_i\ to\ root\}} \gamma w_k \quad \Rightarrow \quad \pi_j = S^{(I)}{}_i w_{ij}$$

Summary

- Link analysis is a key technology in Web search
  - Link analysis algorithms like PageRank have achieved great success and contribute significantly to today's search engines.
- Link analysis technologies also have limitations
  - They only use the structure of the graph, while many other informative factors are ignored, such as user clicks and content information.

Beyond Link Analysis

- More metadata besides link structure
  - Information on nodes and edges
  - Supervision information for the ranking order

Outline

I. Overview
II. Graph Ranking by Link Analysis
III. Graph Ranking with Node and Edge Features
IV. Graph Ranking with Supervision
V. Implementation for Graph Ranking
VI. Summary

Beyond Link Graph

- In conventional link analysis, link graph is simply represented by a binary adjacency matrix.
- In practice, we have rich metadata associated with the nodes and edges, and thus the representation of the graph can be more complex.

Examples

- Non-uniform Teleport Vector
  - Node metadata: bias to some nodes
- Weighted Link Graph
  - Edge metadata: number of links from one node to another
- User Browsing Graph
  - Node metadata: user staying time on each node; frequency of user visits on each node.
  - Edge metadata: number of user transition from one page to another.

Personalized PageRank

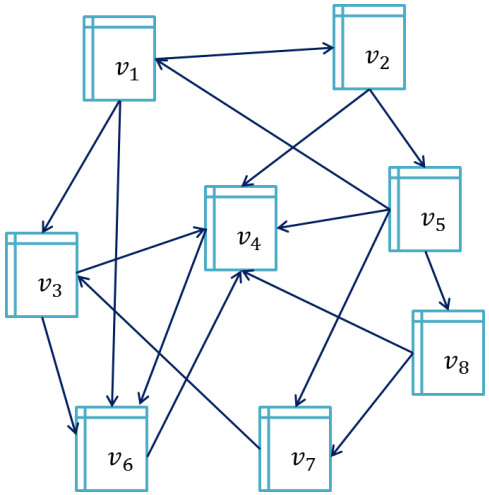
$$\pi = \alpha P^T \pi + (1 - \alpha) r$$

Personalized Teleport Vector

- Change $\frac{1}{n} e$ with r
- Instead of teleporting uniformly to any page, we bias the jump on some pages over others
  - E.g., r_i is 1 for your homepage and 0 otherwise.
  - E.g., r_i prefers the topics you are interested in.

Haveliwala et al. An analytical comparison of approaches to personalizing PageRank. Stanford University Technical Report, 2003

Examples



ID	π	r
1	0.0250	0.125
2	0.0259	0.125
3	0.0562	0.125
4	0.4068	0.125
5	0.0298	0.125
6	0.3955	0.125
7	0.0357	0.125
8	0.0251	0.125

uniform vector

ID	π	r
1	0.1024	0.65
2	0.0365	0.05
3	0.0515	0.05
4	0.3774	0.05
5	0.0230	0.05
6	0.3792	0.05
7	0.0177	0.05
8	0.0124	0.05

bias on v_1

ID	π	r
1	0.0100	0.05
2	0.0103	0.05
3	0.0225	0.05
4	0.4384	0.05
5	0.0119	0.05
6	0.4825	0.65
7	0.0143	0.05
8	0.0100	0.05

bias on v_6

Personalized PageRank

$$\pi = \alpha P^T \pi + (1-\alpha) r$$

$$\pi = (1-\alpha)(I - \alpha P^T)^{-1} r$$

Fixed Matrix

Personalized Teleport Vector

Topic-Sensitive PageRank

- Instead of using one single PageRank value to represent the importance of Web page, calculate a vector of PageRank values, according to 16 topics in ODP.
- For each value in this vector, when making the PageRank metric primitive, use different transition matrix (only randomly jump to those pages of the given topic).

Haveliwala et al. Topic-sensitive PageRank. WWW, 2002 .

Topic-Sensitive PageRank (cont.)

- Category biasing
  - T_j: set of pages in category c_j
  - $r^{(j)}$: teleport vector

$$r_i^{(j)} = \begin{cases} \frac{1}{|T_j|}, & i \in T_j \\ 0, & i \notin T_j \end{cases} \quad \Longrightarrow \quad \pi^{(j)}$$

Topic-Sensitive PageRank (cont.)

- Query-time importance score
  - q: query or query context

$$\pi_i = \sum_j p(c_j|q)\pi_i^{(j)}$$

$$p(c_j|q) = \frac{p(c_j)p(q|c_j)}{p(q)}$$

Weighted Link Graph

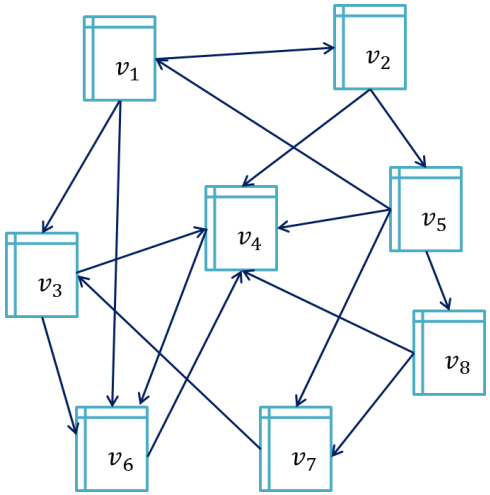
$$M = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Adjacent Matrix

$$M = \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 2 & 0 & 0 & 3 & 0 & 0 & 5 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Weighted Adjacent Matrix

Weighted PageRank



$$M = \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 2 & 0 & 0 & 3 & 0 & 0 & 5 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 \end{bmatrix}$$

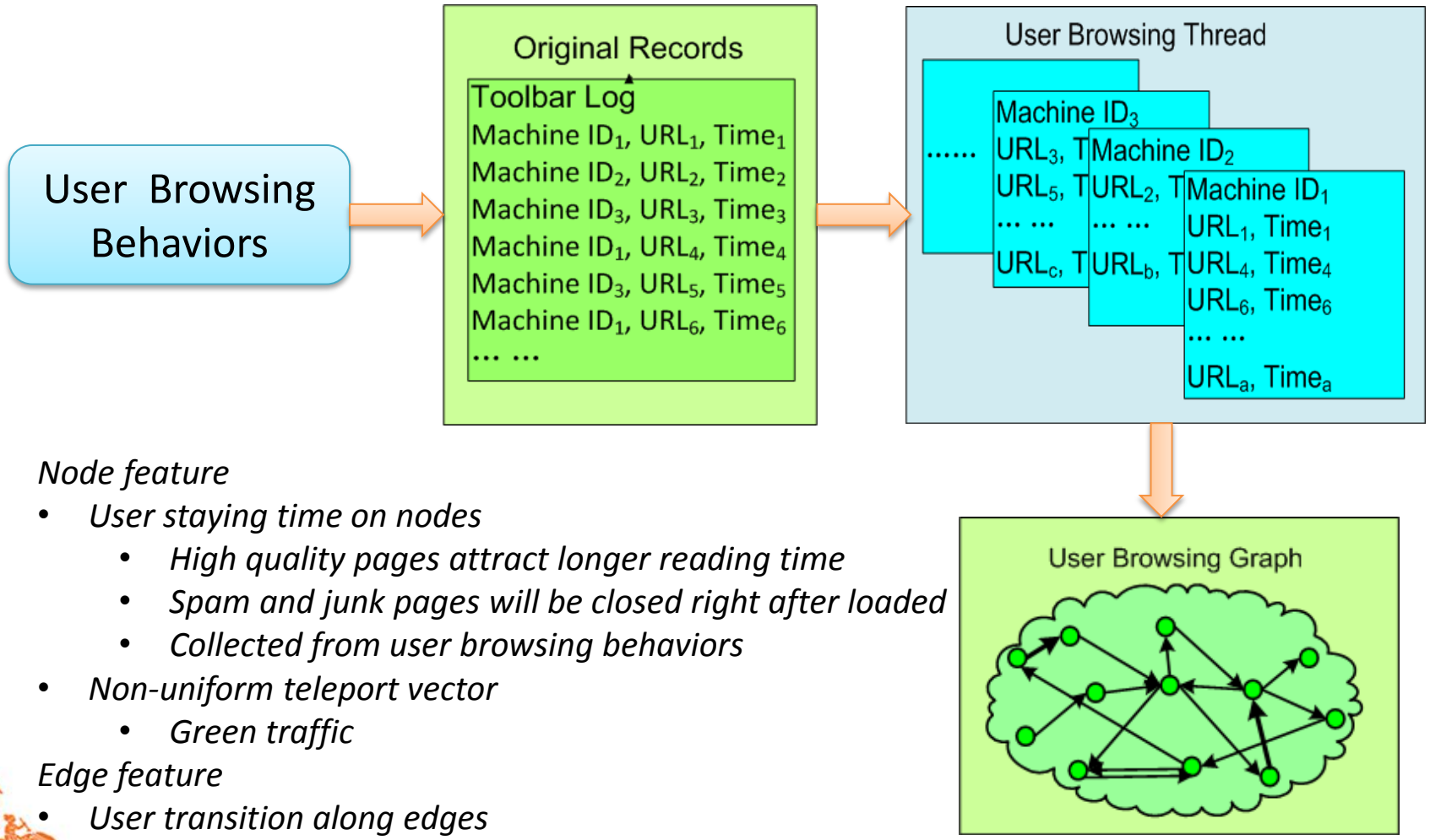
ID	π	r
1	0.0250	0.125
2	0.0259	0.125
3	0.0562	0.125
4	0.4068	0.125
5	0.0298	0.125
6	0.3955	0.125
7	0.0357	0.125
8	0.0251	0.125

Un-weighted

ID	π	r
1	0.0239	0.125
2	0.0255	0.125
3	0.0541	0.125
4	0.4142	0.125
5	0.0332	0.125
6	0.3902	0.125
7	0.0376	0.125
8	0.0213	0.125

Weighted

User Browsing Graph



Node feature

- *User staying time on nodes*
  - *High quality pages attract longer reading time*
  - *Spam and junk pages will be closed right after loaded*
  - *Collected from user browsing behaviors*
- *Non-uniform teleport vector*
  - *Green traffic*

Edge feature

- *User transition along edges*

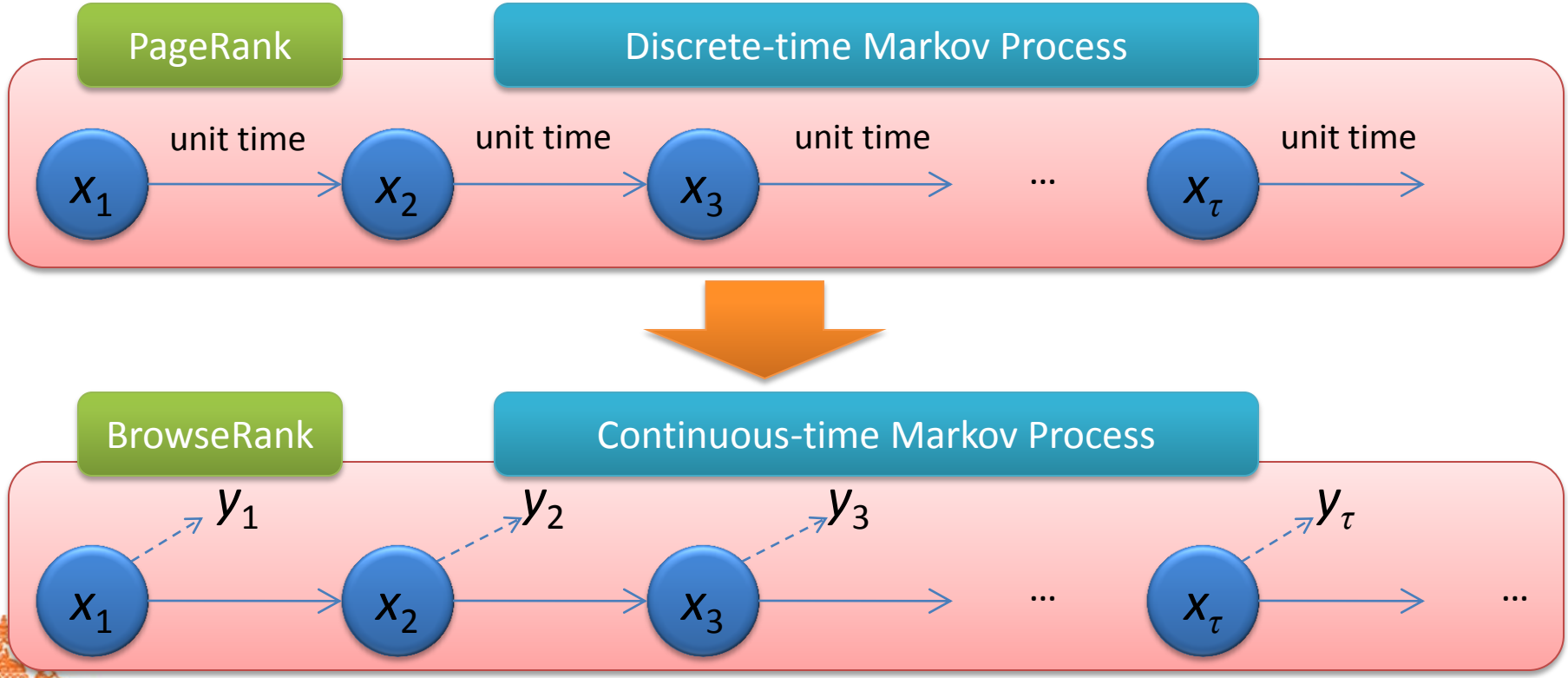
BrowseRank

- Computed from User Browsing Graph
- Random surfer model
  - Start from a page selected from the distribution of “green traffic”
  - Stay a period of time
  - Jump to next page by weighted adjacent matrix
- Challenge
  - Discrete-time Markov model does not work here
    - Cannot model the non-unit staying time

Liu et al. BrowseRank: Letting Web users vote for page importance. SIGIR, 2008.

Continuous-time Markov Model

- Model the real user browsing behavior on the Web as a continuous-time Markov process on the user browsing graph



Stationary Distribution

- Calculate the page importance as the stationary probability distribution of such stochastic process

PageRank

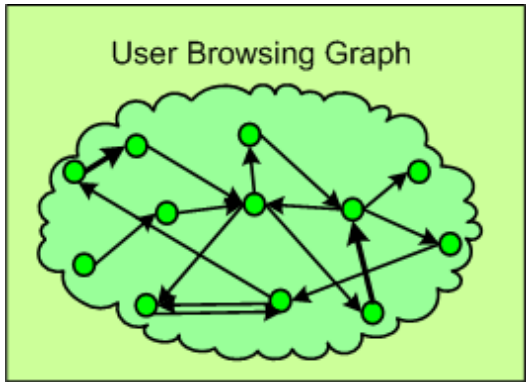
$$\pi = P^T \pi$$

BrowseRank

$$\pi = P(t)^T \pi, \ \forall t > 0$$

Hard to compute

Decomposition



$P(t)$ → $\pi = P(t)^T \pi,\ \forall t > 0$

Calculating π

$$\pi_i = \frac{\tilde{\pi}_i / \lambda_i}{\sum_{j=1}^{n} \tilde{\pi}_j / \lambda_j}$$

=

Estimating **staying time** distribution $1 - e^{-\lambda_i t}$

+

Computing the stationary distribution $\tilde{\pi} = \{\tilde{\pi}_i, i = 1, \ldots, n\}$ of a discrete-time Markov chain (*called **embedded Markov chain***)

Staying Time Calculation

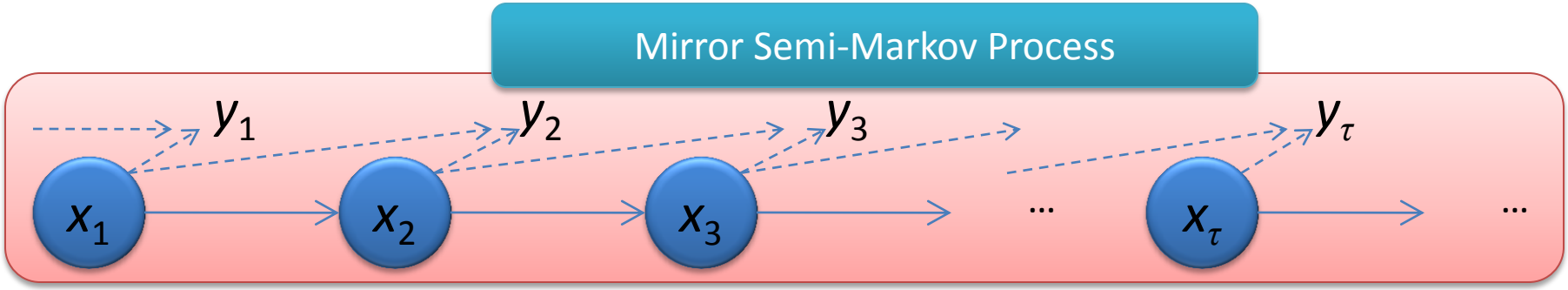
- T_i: random variable of staying time on v_i
- $F_{T_i}(t)$: cumulative probability distribution of random variable T_i

$$F_{T_i}(t) = 1 - e^{-\lambda_i t}$$

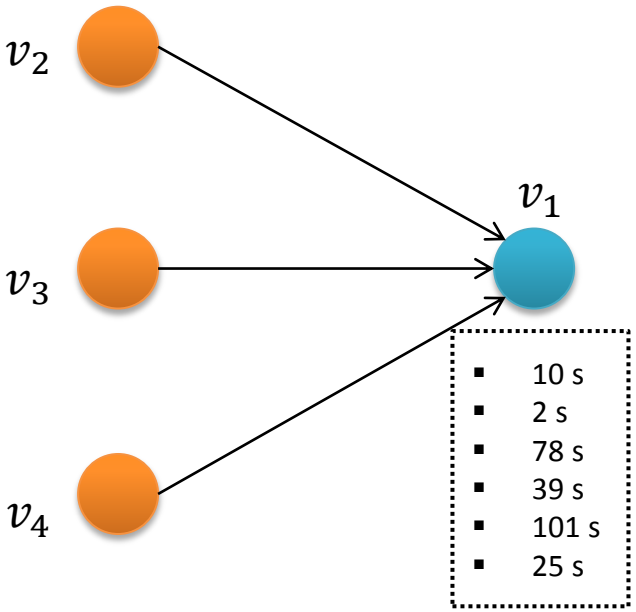
$$\tilde{T}_i = E(T_i) = \int_0^\infty t F_{T_i}(t) dt = \frac{1}{\lambda_i}$$

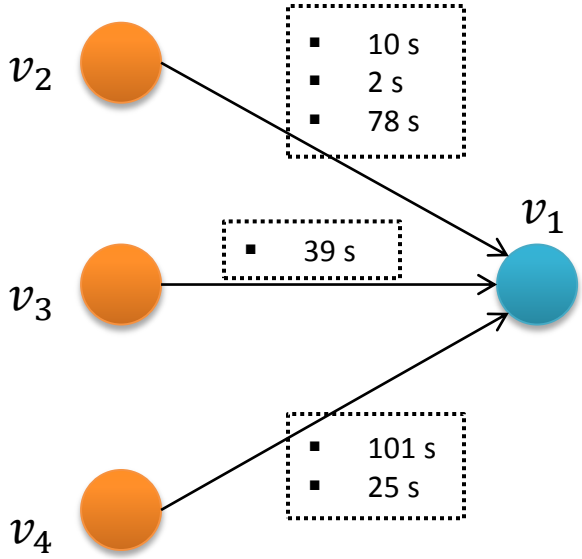
Discussions on Staying Time

- Staying time might not only depend on the current page; It also depends on the source page from which users transits to the current page.



From BrowseRank to BrowseRank Plus





BrowseRank Plus

- T_j: random variable of staying time on v_j
- ${}_iF_{T_j}(t)$: cumulative probability distribution of random on v_j from v_i
- c_{ij}: contribution probability of v_i to v_j

$$ {}_iF_{T_j}(t) = 1 - e^{-\lambda_{ij}t} $$

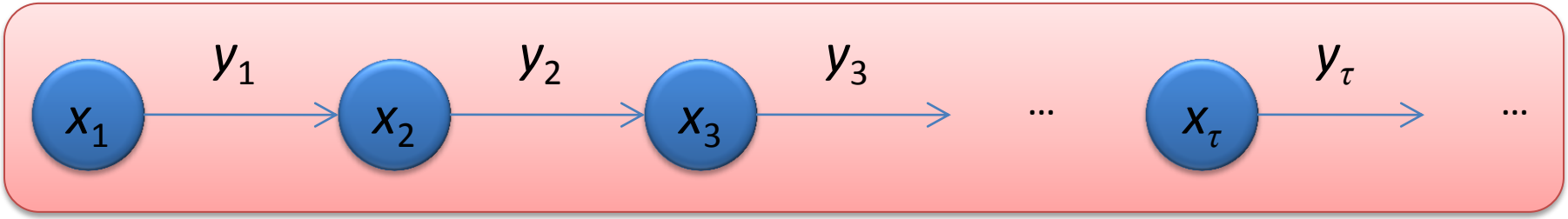
$$ \tilde{T}_j = E(T_j) = \sum_i c_{ij} \int_0^\infty t \times {}_iF_{T_j}(t)dt = \sum_i \frac{c_{ij}}{\lambda_{ij}} $$

$$ c_{ij} = \frac{p_{ij}\tilde{\pi}_i}{\tilde{\pi}_j} $$

Gao et al. A general Markov framework for page importance computation. CIKM, 2009.

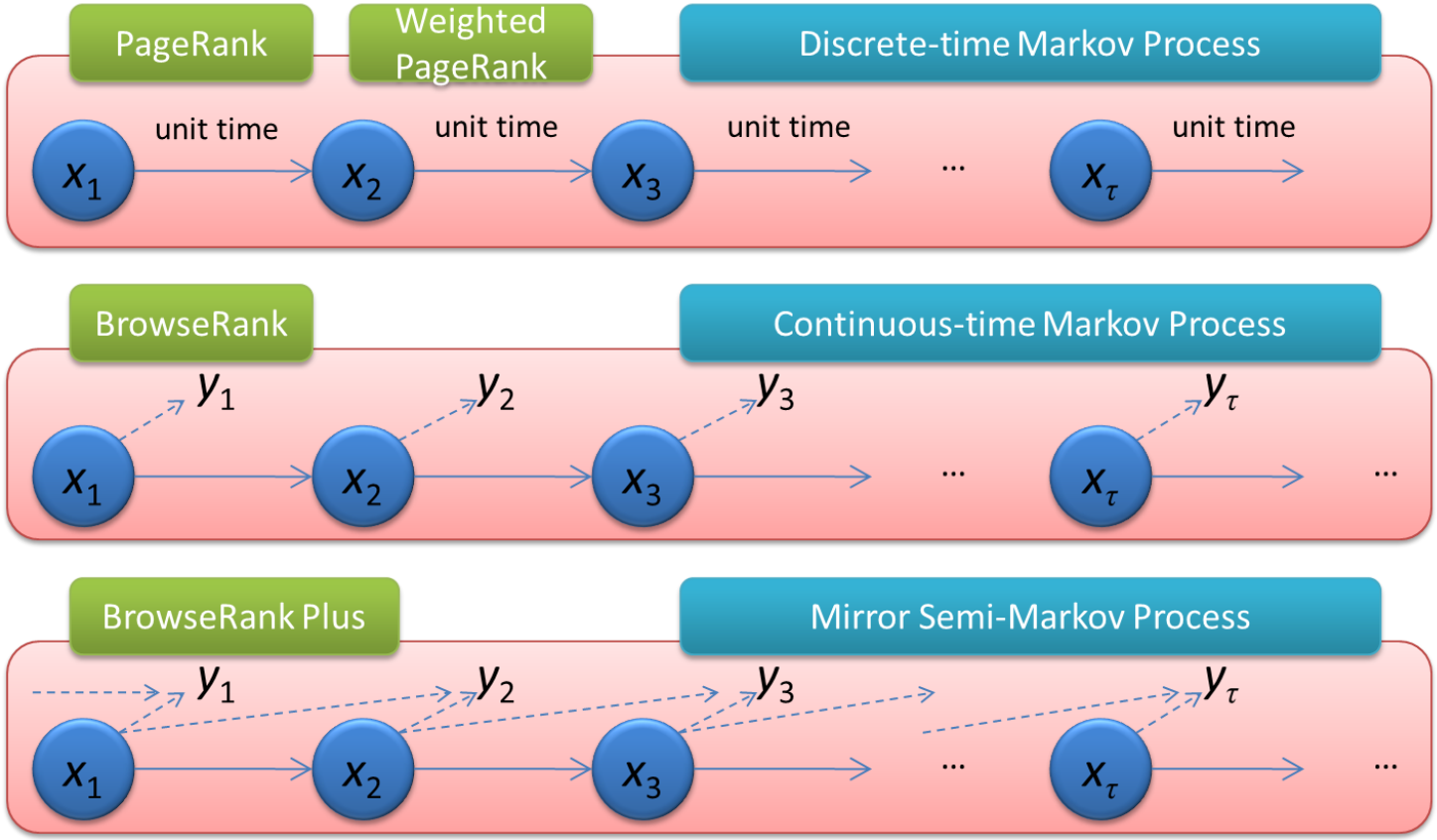
A Unified Model

Markov Skeleton Process Model

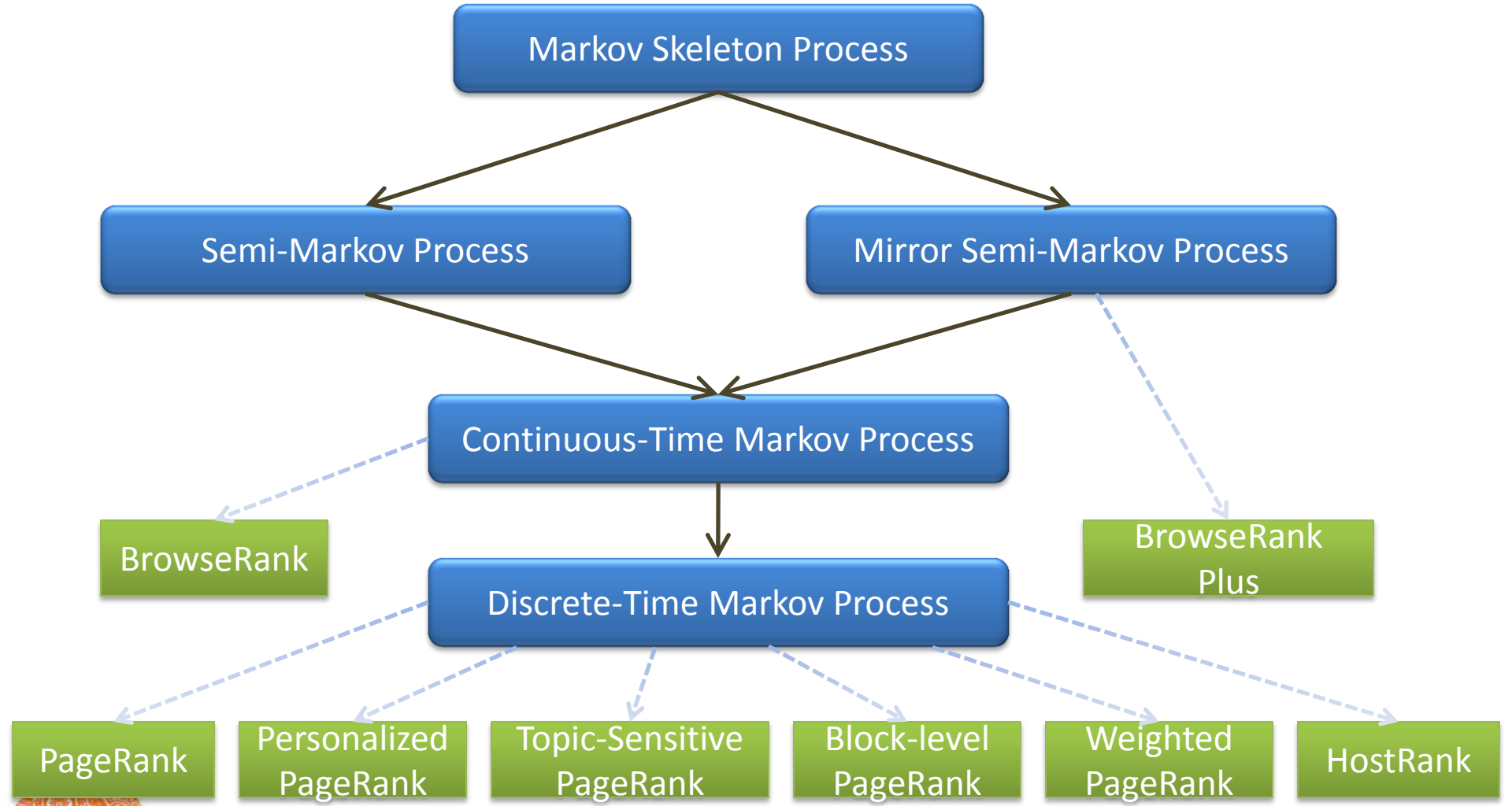


x: Markov chain – to model the jump of the random surfer
y: Random variable dependent on x (can be understood as staying time for simplicity) – to model page utility.

Algorithms in the Framework



Algorithms in the Framework



Outline

I. Overview
II. Graph Ranking by Link Analysis
III. Graph Ranking with Node and Edge Features
IV. Graph Ranking with Supervision
V. Implementation for Graph Ranking
VI. Summary

Beyond the Graph

- In addition to the weights associated with nodes and edges, we sometimes also have supervision on the nodes.
- Typical supervision
  - Binary labels
    - Spam/non-spam, Junk/non-junk, etc.
  - Pairwise preference
    - A is preferred to B
  - List of partial order
    - A > B > C, according to user visiting frequency.

Notations

- Graph $G(V, E, X, Y)$
  - $V = \{v_i\}$: node set, $|V| = n$
  - $E = \{e_{ij}\}$: edge set, $|E| = m$
  - $X = \{x_{ij}\}$: edge features, $|x_{ij}| = l$, $x_{ij} = (x_{ij1}, x_{ij2}, \dots, x_{ijl})^T$
  - $Y = \{y_i\}$: node features, $|y_i| = h$, $y_i = (y_{i1}, y_{i2}, \dots, y_{ih})^T$
- Matrices
  - M: adjacency matrix or link matrix
  - P: transition probability matrix
- Rank score vectors
  - a: authority score vector
  - h: hub score vector
  - π: general rank score vector

Notations for Supervision

- Supervision
  - B: $\bar{n}$-by-n supervision matrix, each row of B represents a pairwise preference $v_i \succ v_j$ ($\bar{n}$ is the number of pairwise constraints)

$$B = \begin{bmatrix} & & 0 \cdots i & \cdots & j \cdots 0 & & \\ & & & \vdots & & & \\ 0 \cdots & 1 & \cdots & & -1 & \cdots 0 \\ & & & \vdots & & & \end{bmatrix}_{\bar{n} \times n}$$

$$v_i \succ v_j \leftrightarrow \pi_i \geq \pi_j \leftrightarrow a\ row\ in\ B\pi \geq 0$$

$$v_i \succ v_j \leftrightarrow \min\{1 - (\pi_i - \pi_j)\} \leftrightarrow \min -e^T(B\pi - e)$$

- Parameters
  - $\omega = (\omega_1, \omega_2, \ldots, \omega_l)^T$: weight of edge features
  - $\phi = (\phi_1, \phi_2, \ldots, \phi_h)^T$: weight of node features

Supervised Graph Ranking Algorithms

- LiftHITS
- Adaptive PageRank
- NetRank I & II
- Laplacian Rank
- Semi-supervised PageRank

LiftHITS

- Adjust adjacency matrix of HITS using one-step gradient ascent, to satisfy the supervision
- Methodology

$$a(v_i) = \sum_j M_{ji} h(v_j)$$
$$h(v_i) = \sum_j M_{ij} a(v_j)$$

$$\Rightarrow \quad a^+ = M^T M a \quad \Rightarrow \quad a^+(v_j) = \sum_i \left(\sum_k M_{ki} M_{kj} \right) a(v_j)$$

$$\Downarrow$$

$$\frac{\partial a^+(v_j)}{\partial M_{ki}} = M_{kj} a(v_i)$$

Chang et al. Learning to create customized authority lists. ICML, 2000.

LiftHITS

- Algorithm (to lift the rank of v_j)
  1. Apply HITS to compute authorities a based on M
  2. Compute gradient $\forall k, i, \Delta M_{ki} \triangleq \frac{\partial a^+(v_j)}{\partial M_{ki}} = M_{kj} a(v_i)$
  3. Update $M^+{}_{ki} = M_{ki} + \gamma \frac{\Delta M_{ki}}{\sum_i \Delta M_{ki}}$
  4. Normalize weights, setting all $M_{ki} \geq 0$
  5. Re-compute HITS authorities a using updated M^+
- Discussion
  - May affect the ranking of neighborhood nodes
  - M^+ will become denser than M

Adaptive PageRank

- Adjust teleport vector to produce a ranking result
  - To satisfy the supervision
  - To be as close to PageRank as possible

Tsoi et al. Adaptive ranking of Web pages. WWW, 2003.

Adaptive PageRank

- Methodology
  - Transform PageRank equation

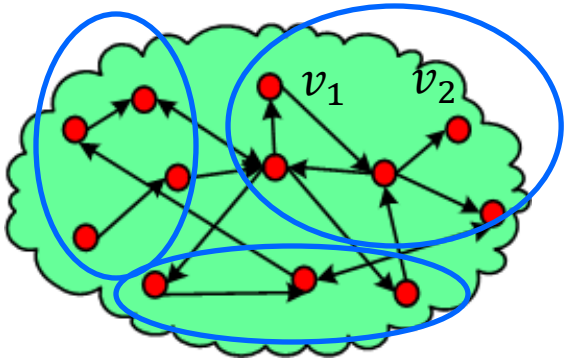
$$\pi = \alpha P^T \pi + (1-\alpha) r \;\rightarrow \pi = (1-\alpha)(I - \alpha P^T)^{-1} r \triangleq Qr$$

  - Optimize teleport vector r

$$\begin{aligned} \min_{r} \; & \left\| Qr - Qr^{(0)} \right\|_2 \\ \text{s.t.} \quad & BQr \geq 0, r \geq 0 \\ & e^T r = 1, \; r^{(0)} = \frac{1}{n} e \\ & e = (1,1,\ldots,1)^T \end{aligned}$$

Adaptive PageRank

- Reduce the complexity
  - Compute cluster-level adaptive PageRank
  - Organize nodes into clusters according to some criteria
  - Assign back the scores in node-level
- Discussion
  - Some supervision will become invalid



$$v_1 \succ v_2$$

NetRank I

- Adjust PageRank flow to produce a ranking result
  - To satisfy the supervision
  - To maximize the entropy of the PageRank flow

All PageRank flows between nodes are equal to each other.

The sum of flows in one node equals its inlink number.

NetRank I

- Notations
  - v_d: dummy node having two-way edges with all $v_i \in V$
  - $V' = V \cup \{v_d\}$
  - $G' = \{V', E'\}$
  - $V^{(0)}$: the set of nodes which have at least one outlink
  - ρ_{ij}: PageRank flow from v_i to v_j

Agarwal et al. Learning to rank networked entities. KDD, 2006.

Optimization Problem in NetRank I

$$\min_{0 \le \rho_{ij} \le 1} \sum_{(i,j) \in E'} \rho_{ij} \log \rho_{ij}$$ ⟸ Objective

s.t.

$$\sum_{(i,j) \in E'} \rho_{ij} - 1 = 0$$ ⟸ Total

$$\forall v_j \in V' : -\sum_{(i,j) \in E'} \rho_{ij} + \sum_{(j,k) \in E'} \rho_{jk} = 0$$ ⟸ Balance

$$\forall v_j \in V^{(0)} : -\alpha \rho_{jd} + (1-\alpha) \sum_{(j,k) \in E} \rho_{jk} = 0$$ ⟸ Teleport

$$\forall v_i \prec v_j : \sum_{(k,i) \in E'} \rho_{ki} - \sum_{(k,j) \in E'} \rho_{kj} \le 0$$ ⟸ Preference

- Discussion
  - Converted to the duel formulation and solved by gradient method
  - Too many variables (edge number)

NetRank II

- Adjust the parametric transition matrix to produce a ranking result
  - To satisfy the supervision
  - To be as close to PageRank as possible

NetRank II

- Step 1: Build a parametric transition matrix
  - v_d: dummy node having two-way edges with all $v_i \in V$
  - x_{ijk}: edge features from v_i to v_j
  - $f(\cdot)$: edge feature combination function

$$P(i,j) = \begin{cases} 0, & v_i \neq v_d, v_j \neq v_d, v_i \in leaf(V) \\ \alpha f(\omega, x_{ij}), & v_i \neq v_d, v_j \neq v_d, v_i \notin leaf(V) \\ 1, & v_i \neq v_d, v_j = v_d, v_i \in leaf(V) \\ 1-\alpha, & v_i \neq v_d, v_j = v_d, v_i \notin leaf(V) \\ r_j & v_i = v_d, v_j \neq v_d \\ 0, & v_i = v_d, v_j = v_d \end{cases}$$

Chakrabarti et al. Learning parameters in entity relationship graphs from ranking preference. PKDD, 2006.

NetRank II

- Step 2: Minimize the loss following function calculated from the parametric matrix
  - H: iteration number for PageRank calculation

$$\min \sum_{v_i \prec v_j} \{1 + ((P^T)^H \pi^{(0)})_i - ((P^T)^H \pi^{(0)})_j\}$$

- Discussion
  - NetRank I → II, reduce the number of parameters
  - Newton method for computation
  - Need to compute successive matrix multiplication.

Laplacian Rank

- Adjust the ranking result directly
  - To satisfy the supervision
  - To make connected nodes have similar ranking

Laplacian Rank

- Laplacian matrix
  - Π: diagonal matrix with $\Pi_{ii} = \pi^{(0)}(v_i)$

$$L = I - \frac{\Pi^{\frac{1}{2}} P \Pi^{-\frac{1}{2}} + \Pi^{-\frac{1}{2}} P^T \Pi^{\frac{1}{2}}}{2}$$

  - Optimization with Regularization

$$\min \frac{1}{2} \pi^T L \pi + \lambda \sum_{v_i \prec v_j} \varepsilon_{ij}$$

$$\text{s.t.} \quad \pi_j - \pi_i \geq 1 - \varepsilon_{ij}, \forall v_i \prec v_j$$

- Discussion
  - Need to compute pseudo matrix inversion

Agarwal. Ranking on graph data. ICML, 2006.

Semi-Supervised PageRank

- Adjust the parametric transition matrix and the parametric teleport vector to produce a ranking result
  - To satisfy the supervision
  - To be as close to PageRank as possible

Semi-Supervised PageRank

- Methodology
  - Define parametric transition matrix
    - $p_{ij} = \begin{cases} \frac{\sum_k \omega_k x_{ijk}}{\sum_j \sum_k \omega_k x_{ijk}} & e_{ij} \in E \\ 0, & otherwise \end{cases}$
  - Define parametric teleport vector
    - $r_i(\phi) = \phi^T y_i$
  - Minimize the sum of a propagation term and a loss term

Gao et al. Semi-supervised ranking on very large graph with rich metadata. Microsoft Research Technical Report, MSR-TR-2011-36, 2011.

Semi-Supervised PageRank

$$\pi = \alpha P^T(\omega)\pi + (1-\alpha)r(\phi)$$

$$\min_{\omega \geq 0,\, \phi \geq 0,\, \pi \geq 0} \{\beta_1 \|\alpha P^T(\omega)\pi + (1-\alpha)r(\phi) - \pi\|^2 + \beta_2 \mu(e - B\pi)\}$$

Propagation term: based on PageRank propagation, combining edge features and node features by $P(\omega)$ and $r(\phi)$.

Loss term: compared with supervised information in pairwise preference fashion.

Gradient based Optimization

Denote $G(\omega, \phi, \pi) = \alpha \| dP^T(\omega)\pi + (1-d)g(\phi) - \pi \|^2 + \beta \mu^T (e - B\pi)$

Derivatives

$$\frac{\partial G}{\partial \omega} = 2\alpha d [P^T \pi \otimes \pi - \pi \otimes \pi + (1-d) g \otimes \pi]^T \frac{\partial vec(P)}{\partial \omega^T}$$

$$\frac{\partial G}{\partial \phi} = 2\alpha (1-d)[(1-d)g + dP^T \pi - \pi] \frac{\partial g}{\partial \phi}$$

$$\frac{\partial G}{\partial \pi} = 2\alpha [(dPP^T - dP - dP^T + I)\pi - (1-d)(I - dP)g] - \beta B^T \mu$$

More Details

$$x \otimes_G y$$

$$\frac{\partial g}{\partial \phi} = \begin{pmatrix} \frac{\partial g}{\partial \phi_1} \\ \vdots \\ \frac{\partial g}{\partial \phi_i} \\ \vdots \\ \frac{\partial g}{\partial \phi_n} \end{pmatrix}$$

$$\frac{\partial vec(P)}{\partial \omega^T} = \begin{pmatrix} \frac{\partial p_{11}}{\partial \omega_1} & \cdots & \frac{\partial p_{11}}{\partial \omega_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial p_{n1}}{\partial \omega_1} & \cdots & \frac{\partial p_{n1}}{\partial \omega_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial p_{1n}}{\partial \omega_1} & \cdots & \frac{\partial p_{1n}}{\partial \omega_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial p_{nn}}{\partial \omega_1} & \cdots & \frac{\partial p_{nn}}{\partial \omega_n} \end{pmatrix}$$

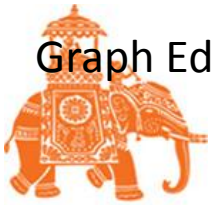
X1	X2	X3	X4
1	2	3	4

$\otimes$

Y1	Y2	Y3	Y4
1	2	3	4

=

X1 Y1	X1 Y2	X1 Y3	X1 Y4	X2 Y1	X2 Y2	X2 Y3	X2 Y4	X3 Y1	X3 Y2	X3 Y3	X3 Y4	X4 Y1	X4 Y2	X4 Y3	X4 Y4
1	2	3	4	2	4	6	8	3	6	9	12	4	8	12	16
Graph Edges	1	1	0	0	1	1	1	0	1	1	0	0	0	0	1	1



A Unified Framework

- Define the graph propagation term based on a Markov random walk on the web graph
- Incorporate edge features into the transition probability of the Markov process, and incorporate node features to its teleport vector
- Convert the constraints to loss functions using L_2 distance between the ranking results given by the parametric model and the supervision
- Keep the sparsity of the graph when updating the parameters of the model during the optimization process

A Unified Framework

$$\min_{\omega \geq 0,\ \phi \geq 0,\ \pi \geq 0} R(\pi; f(\omega, X), g(\phi, Y))$$
$$\text{s.t.} \quad S(\pi; B, \mu) \geq 0$$

$$\min_{\omega \geq 0,\ \phi \geq 0,\ \pi \geq 0} \beta_1 R(\pi; f(\omega, X), g(\phi, Y)) - \beta_2 S(\pi; B, \mu)$$

Propagation term: based on a certain graph ranking algorithm, combining graph structure and rich metadata.

Loss term: compared with supervision

Algorithms in the Framework

Algorithm	Link Structure	Edge Feature	Node Feature	Supervision	Objective	Parameterized Model
Adaptive PageRank	Yes	No	No	Pairwise	PageRank	No
NetRank I	Yes	No	No	Pairwise	Inlink number	No
NetRank II	Yes	Yes	No	Pairwise	PageRank	Yes
Laplacian Rank	Yes	No	No	Pairwise	Laplacian	No
Semi-supervised PageRank	Yes	Yes	Yes	Pairwise	PageRank	Yes

Outline

I. Overview
II. Graph Ranking by Link Analysis
III. Graph Ranking with Node and Edge Features
IV. Graph Ranking with Supervision
V. Implementation for Graph Ranking
VI. Summary

Many Algorithms

PageRank

Topic-sensitive PageRank

Personalized PageRank

HITS

HostRank

BrowseRank

LiftHITS

Block-level PageRank

BrowseRank+

Adaptive PageRank

NetRank I, II

LaplacianRank

Semi-supervised PageRank

Basic Operations in Algorithms

Algorithms	Operation
Almost All	Matrix-vector multiplication
NetRank II	Matrix-matrix multiplication
Semi-supervised PageRank	Graph-based Kronecker product between vectors
Adaptive PageRank, LaplacianRank	Matrix (pseudo) inversion
…	…

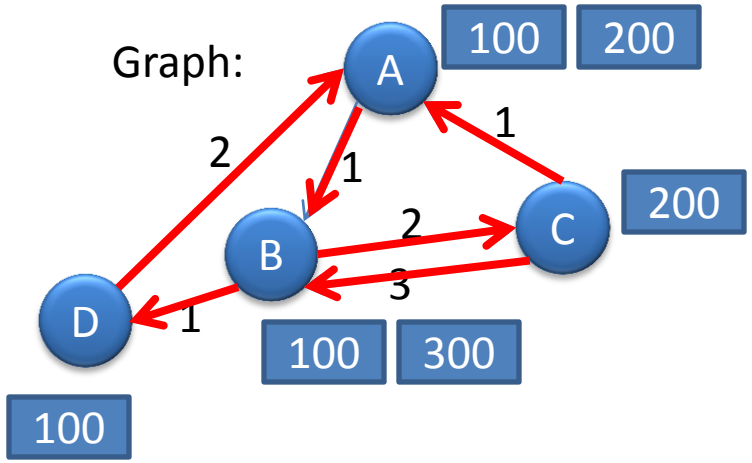
Graph Propagation

- For many operations, propagation of values along edges in a graph is their basic computational unit.

Example: Matrix-Vector Multiplication

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

Px



$x = [$

A	B	C	D
100	100	100	100

$]^T$

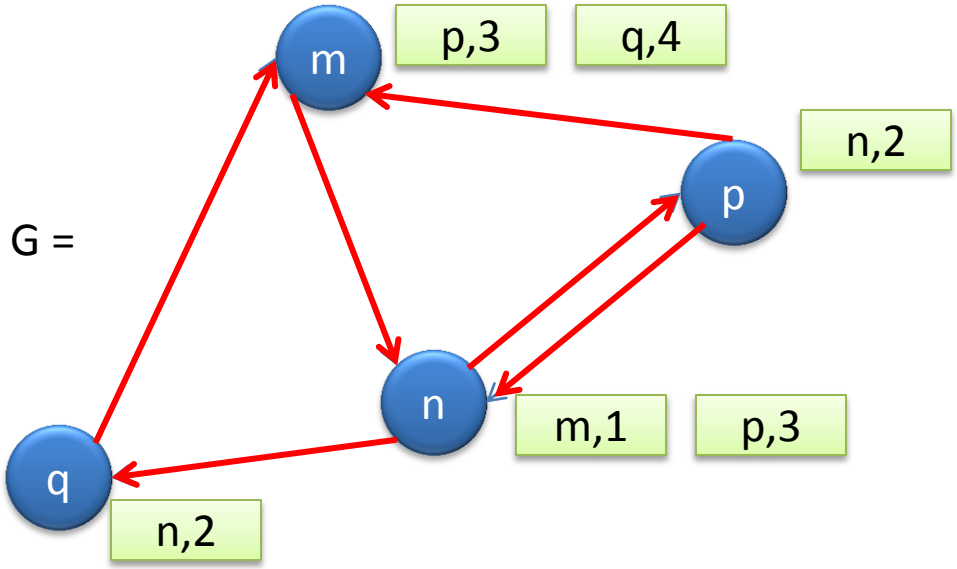
$Px = [$

A	B	C	D
300	400	200	100

$]^T$

- ✓ Propagate elements in *x* in the graph defined by *P*
- ✓ Aggregate the propagated values per node.

Example: Kronecker Product $x \otimes_G y$



- ✓ *Propagate y along graph G*
- ✓ *For each node, multiply x with the received y values*

y =

m	n	p	q
1	2	3	4

m	n	p	q
[P,3] [q,4]	[m,1] [p,3]	[n,2]	[n,2]

x =

m	n	p	q
5	6	7	8

Result = | [mp,15] [mq,20] | [nm,6] [np,18] | [pn,14] | [qn,16] |

Large-scale Implementation of Graph Propagation

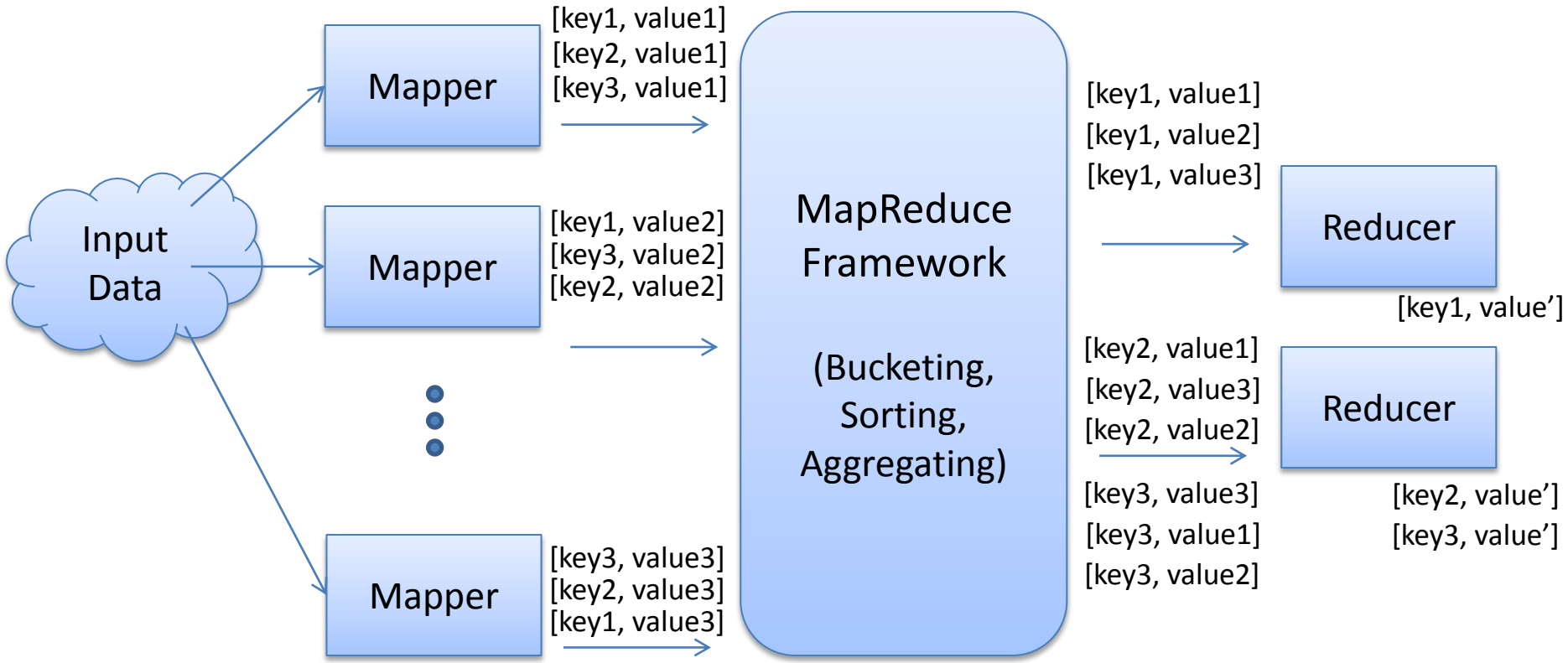
- Distributed Computation Models
  - MapReduce Model
  - Bulk Synchronous Parallel(BSP) Model

MapReduce

- Mapper
  - Process input data into (key, value) pairs
  - Output (key, value) pairs into MapReduce framework
- MapReduce Framework
  - Sort and aggregate (key, value) pairs according to buckets
- Reducer
  - Receive (key, value) pairs with the same range of keys from MapReduce infrastructure
  - Aggregate the values for the same key
  - Output the result

Dean et al. MapReduce: simplified data processing on large clusters. OSDI, 2004.

MapReduce


Input Data
Mapper
Mapper
Mapper
[key1, value1]
[key2, value1]
[key3, value1]
[key1, value2]
[key3, value2]
[key2, value2]
[key3, value3]
[key2, value3]
[key1, value3]
MapReduce Framework
(Bucketing, Sorting, Aggregating)
[key1, value1]
[key1, value2]
[key1, value3]
[key2, value1]
[key2, value3]
[key2, value2]
[key3, value3]
[key3, value1]
[key3, value2]
Reducer
Reducer
[key1, value']
[key2, value']
[key3, value']

Graph Propagation on MapReduce

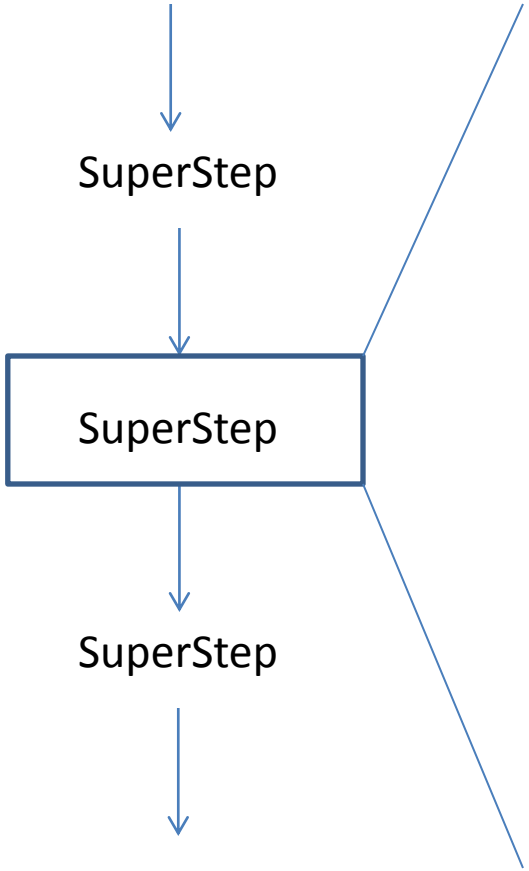
- Graph data is partitioned into sub graphs according to source node; input vector is also partitioned in the same way.
- One mapper processes one partition of graph data and vector.

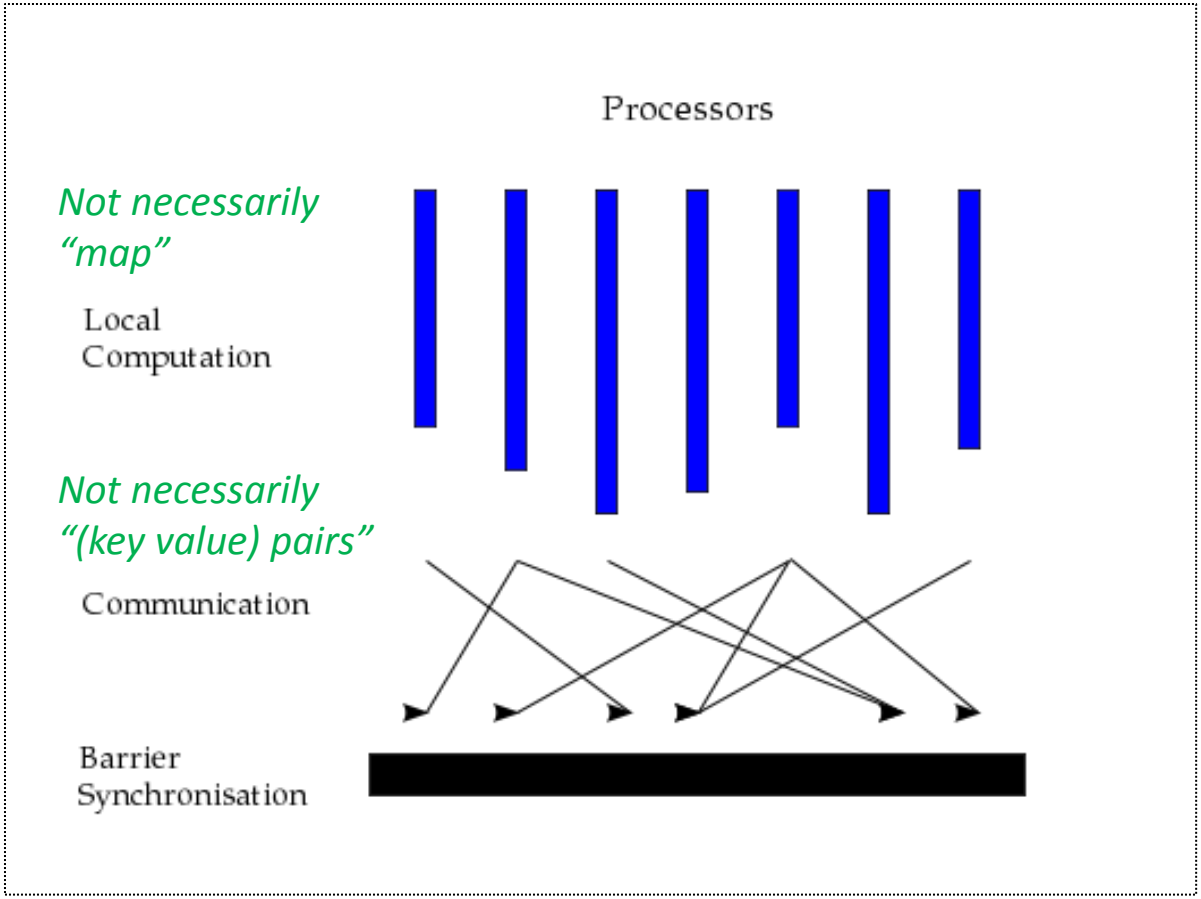
- (Key, value) = (Dest node, Value on source node)
- Reducer aggregates data according to destination node.

Discussions

- Pros: system stability and maturity
  - MapReduce has been widely used for web-scale data processing, e.g., indexing, PageRank computation, etc.
  - Simple interface: (Map, Reduce)
  - Clear system-level logic and fault tolerance
- Cons: system performance
  - **Data locality**, graph data and output data are not guaranteed to be on the same machine, which causes potential network transfer
  - **Intensive access to disk**, at every stage the system needs to serialize data to disk for the purpose of fault tolerance.

BSP





BSP is more general, and MapReduce can be regarded as a special version of BSP

http://en.wikipedia.org/wiki/Bulk_synchronous_parallel

Graph Propagation on BSP

- Graph propagation on MapReduce can be easily converted to that on BSP.

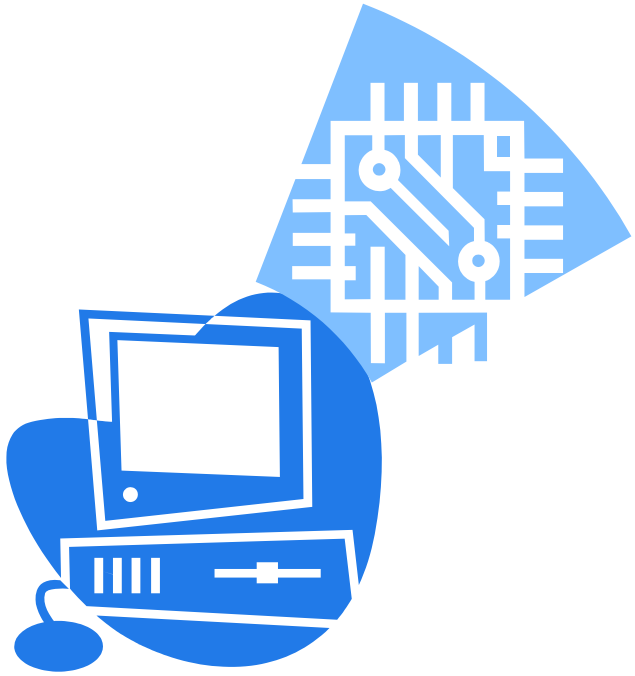
- Since BSP is more flexible, it can potentially ensure locality during the iterative propagation, and thus improve the efficiency of the computation.

Discussions

- Pros
  - Can support co-location of graph partition and processing node, so as to avoid unnecessary data transfer
  - Node communication logic provides more flexible message passing
- Cons
  - Flexibility vs. usability
  - Need to implement fault tolerance logic
  - Not as widely used as MapReduce in industry.

Graph Processing Systems

- Pegasus
- Hama
- Pregel
- Trinity
- Graphor



Pegasus

- A large-scale graph mining system based on Hadoop
- Computation model
  - MapReduce
  - Optimized matrix-vector multiplication by partitioning data into blocks
- Supported algorithms
  - PageRank
  - Random walk with restart
  - Graph diameter computing
  - Graph components mining

Kang et al. PEAGSUS: a peta-scale graph mining system – implementation and observations. ICDM, 2009.
Kang et al. PEAGSUS: mining peta-scale graphs, knowledge and information systems. DOI: 10.1007/s10115-010-0305-0, 2010.

Hama

- Graph processing library in Hadoop
- Computation model
  - MapReduce to handle matrix computation
  - BSP to handle other graph processing
- Supported algorithm
  - Large scale matrix multiplication
  - Shortest path finding in graph
  - PageRank

The Apache Hama Project: http://incubator.apache.org/hama/
Seo et al. HAMA: an efficient matrix computation with the MapReduce framework. IEEE CloudCom Workshop, 2010.

Pregel

- Google's large scale graph processing engine
- Computation model
  - BSP
  - Ram-based system
- Supported algorithms
  - PageRank
  - Single source shortest path
  - Graph component finding

Malewicz et al. Pregel: a system for large-scale graph processing. PODC 2009.
Malewicz et al. Pregel: a System for large-scale graph processing. SIGMOD, 2010.

Trinity

- A graph database and computation platform by Microsoft Research
- Computation model
  - BSP, with asynchronous mode for message passing
  - Ram-based system
- Supported algorithms
  - PageRank
  - Breadth first search

Trinity - a graph data base and computation platform. http://research.microsoft.com/en-us/projects/trinity/

- A graph computation engine by Microsoft Research
- Computation model
  - MapReduce
  - Additional logic to keep graph locality inspired by BSP
- Supported algorithms
  - PageRank
  - Matrix-vector multiplication
  - Graph-based Kronecker product of vectors

Graphor - a billion scale web graph processing platform
http://research.microsoft.com/en-us/projects/graphor/

Comparison

System	Model	Fault tolerance	Supported algorithms
Pegasus	MapReduce	MapReduce	PageRank, graph components finding, etc.
Hama	MapReduce / BSP	MapReduce	PagRank, matrix vector multiplication, etc.
Pregel	BSP	Self designed	PageRank, shortest path, graph components finding, etc.
Trinity	BSP +	None	PagRank, breadth frist search on graph, etc.
Graphor	MapReduce + BSP	MapReduce	Pagerank, multiplication of matrix and vector, graph based vector kronecker product, etc.

Outline

I. Overview
II. Graph Ranking by Link Analysis
III. Graph Ranking with Node and Edge Features
IV. Graph Ranking with Supervision
V. Implementation for Graph Ranking
VI. Summary

Summary

- Link analysis is a classical graph ranking method
- Rich information in nodes and edges can help graph ranking
- External knowledge on ranking orders can make graph ranking more consistent with human intuition
- Many systems have been developed for large-scale graph ranking

Answers to Question #1

- How to perform graph ranking based on graph structure?
  - Link analysis
  - Hierarchical structure in graph
  - Random surfer model / Markov chain

Answers to Question #2

- How to leverage node and edge features for better graph ranking?
  - Node weight
  - Edge weight
  - Continuous-time Markov process / Mirror semi-Markov process / Markov skeleton process

Answers to Question #3

- How to incorporate external knowledge in graph ranking?
  - Different optimization objectives
  - Supervised learning framework
  - Large-scale optimization

Answers to Questions #4

- How to implement large-scale graph ranking algorithms?
  - MapReduce and BSP
  - Several systems
  - Select the proper platform for your application!

Future Directions

- Theoretical study
  - Various Markov processes for graph ranking
  - Learning theory for graph ranking (non-i.i.d.)
- Novel algorithms
  - Ranking on a time series of graphs
  - Ranking on heterogeneous graphs
- Implementation
  - Tradeoff of efficiency, flexibility, and reliability
  - Dealing with more complex graph operations

References

- Agarwal et al. Learning to rank networked entities. KDD, 2006.
- Agarwal. Ranking on graph data. ICML, 2006.
- Cai et al. Block-level link analysis. SIGIR, 2004.
- Chakrabarti et al. Learning parameters in entity relationship graphs from ranking preference. PKDD, 2006.
- Chang et al. Learning to create customized authority lists. ICML, 2000.
- Gao et al. A general Markov framework for page importance computation. CIKM, 2009.
- Gao et al. Semi-supervised ranking on very large graph with rich metadata. Microsoft Research Technical Report, MSR-TR-2011-36, 2011.
- Haveliwala et al. An analytical comparison of approaches to personalizing PageRank. Stanford University Technical Report, 2003
- Kang et al. PEAGSUS: a peta-scale graph mining system – implementation and observations. ICDM, 2009.

References

- Kang et al. PEAGSUS: mining peta-scale graphs, knowledge and information systems. DOI: 10.1007/s10115-010-0305-0, 2010.
- Kleinberg. Authoritative sources in a hyperlinked environment. IBM Research Report RJ 10076, 1997.
- Liu et al. BrowseRank: Letting Web users vote for page importance. SIGIR, 2008.
- Malewicz et al. Pregel: a system for large-scale graph processing. PODC 2009.
- Malewicz et al. Pregel: a System for large-scale graph processing. SIGMOD, 2010.
- Page et al. The PageRank citation ranking: bringing order to the Web. Stanford Digital Library Technologies Project, 1998 .
- Seo et al. HAMA: an efficient matrix computation with the MapReduce framework. IEEE CloudCom Workshop, 2010.
- Tsoi et al. Adaptive ranking of Web pages. WWW, 2003.
- Xue et al. Exploiting the Hierarchical Structure for Link Analysis. SIGIR, 2004.

Acknowledgement

- Wei-Ying Ma (Microsoft Research Asia)
- Hang Li (Microsoft Research Asia)
- Haixun Wang (Microsoft Research Asia)
- Tao Qin (Microsoft Research Asia)
- Zhi-Ming Ma (Chinese Academy of Sciences)
- Yuting Liu (Beijing Jiaotong University)
- Ying Zhang (Nankai University)
- Wei Wei (Huazhong University of Science and Technology)
- Wenkui Ding (Tsinghua University)
- Changhao Jiang (Tsinghua University)
- Chenyan Xiong (Chinese Academy of Sciences)
- Di He (Peking University)

Thank You!

bingao@microsoft.com

taifengw@microsoft.com

tyliu@microsoft.com