Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Link Analysis goes MapReduce

Thomas Zimmermann
Philipp Berger

Overview

1. PageRank Motivation
2. PageRank Calculation
3. Efficient Computation with MapReduce
4. Extensions

Early Search Engines

- Page is ranked high for a query term if:
  - Term occurs often in page
  - Term occurs in header/title
- Problem: easy to fool → Term spam

"So this SEO copywriter walks into a bar, grill, pub, public house, Irish bar, bartender, drinks, beer, wine, liquor"

- Google's solutions:
  - Text of incoming links
  - PageRanks

PageRank Idea

- Random Surfer Model
  - Surfer starts at a random page
  - Follows random links
- PageRank:

  Probability of random surfer visiting a page

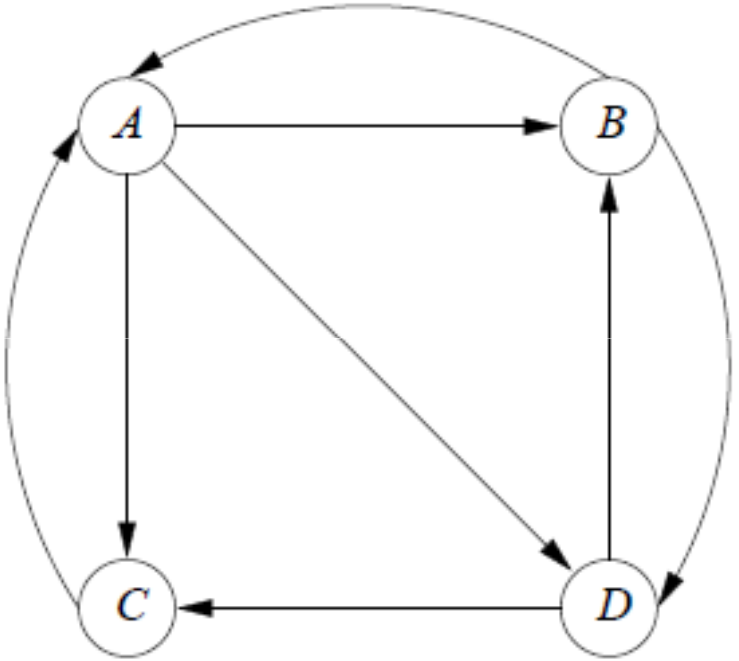
- Rationale:
  - Webmasters tend to link to useful pages

Overview

1. PageRank Motivation
2. PageRank Calculation
3. Efficient Computation with MapReduce
4. Extensions

PageRank Computation I

- Link structure as directed graph
- Transition matrix describes the random surfer‘s choices

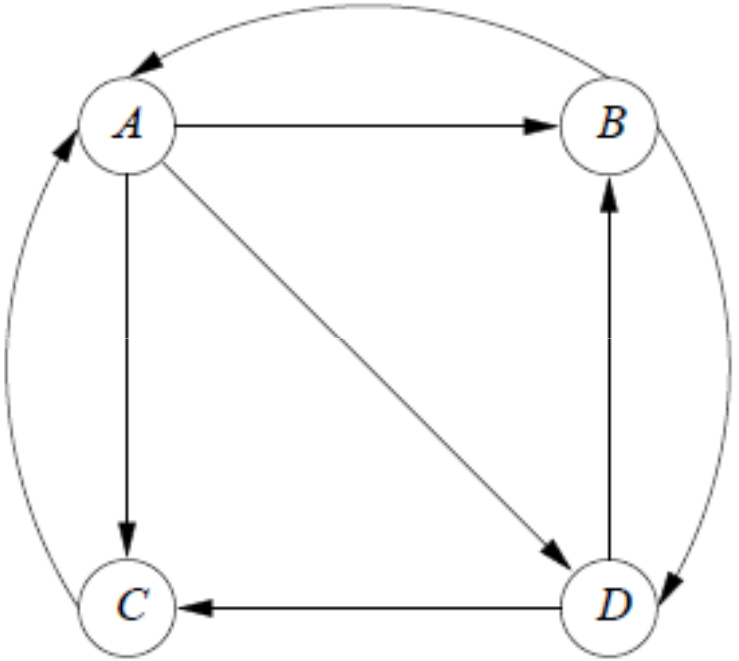


	A	B	C	D
A	0	1/2	1	0
B	1/3	0	0	1/2
C	1/3	0	0	1/2
D	1/3	1/2	0	0

PageRank Computation II

- PageRank function as vector
- i-th component: Probability that surfer is at page i
- Initially: $1/n$

A	1/4
B	1/4
C	1/4
D	1/4



PageRank Computation III

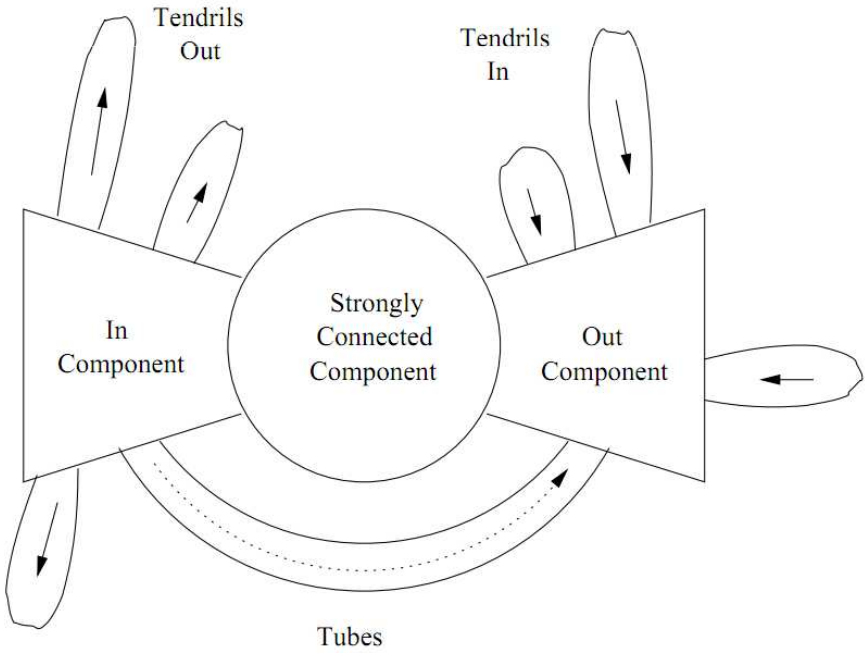
- Multiplying vector and transition matrix
  gives probabilities for next iteration

$$\begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix} * \begin{pmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{pmatrix} = \begin{pmatrix} 0+1/8+1/4+0 = 3/8 \\ 1/12+0+0+1/8 = 5/24 \\ 1/12+0+0+1/8 = 5/24 \\ 1/12+1/8+0+0 = 5/24 \end{pmatrix}$$

- If vector does not change → Found eigenvector $\mathbf{v} = \lambda M v$ for $\lambda = 1$
- Eigenvector contains PageRank values

Dead Ends and Spider Traps

- Two problems:
  - Dead ends (no outlinks) → All probabilities become 0
  - Spider traps (part that cannot be left once entered)
    → All pages not in the trap get PageRank 0



The Random Jump Component

- Solution:
  - Add random jump probability to matrix
  - Probability β: We follow an outlink
  - Probability (1-β): We jump to any page at random

$$\beta * \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 0 \end{bmatrix} * v + (1-\beta) \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}$$

Overview

1. PageRank Motivation
2. PageRank Calculation
3. Efficient Computation with MapReduce
4. Extensions

Naïve MapReduce Approach

- Critical part of algorithm: Vector-matrix multiplication
- Key: Row number → Reducer just sums up

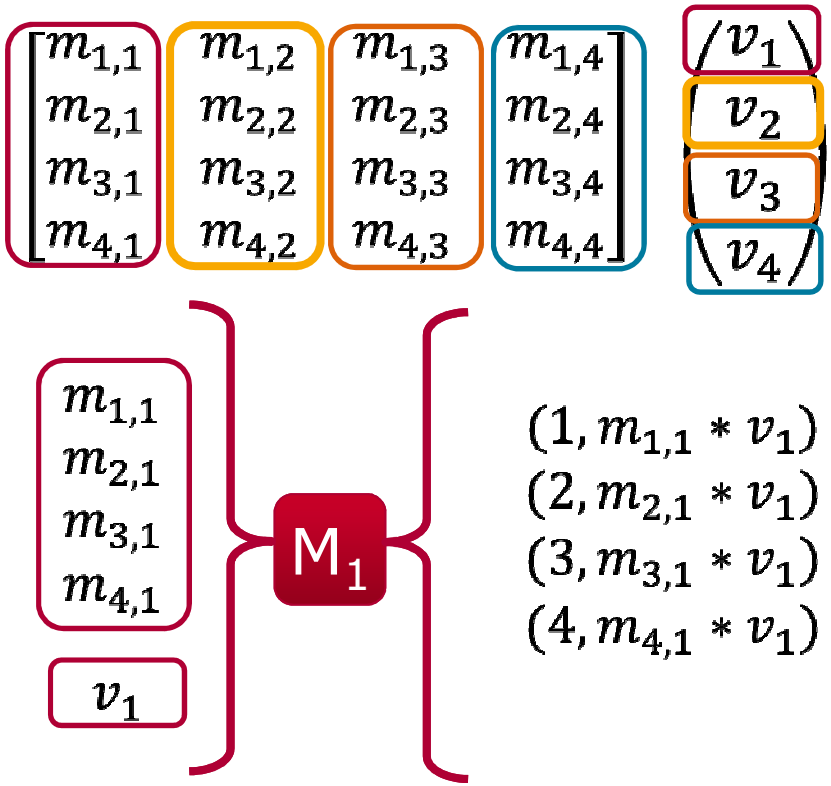
$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{bmatrix} \quad \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

Mapper

$$(1, m_{1,1} * v_1) \quad (1, m_{1,3} * v_3)$$
$$(2, m_{2,1} * v_1) \quad (2, m_{2,3} * v_3)$$
$$(3, m_{3,1} * v_1) \quad (3, m_{3,3} * v_3)$$
$$(1, m_{1,2} * v_2)$$
$$(2, m_{2,2} * v_2)$$
$$(3, m_{3,2} * v_2)$$

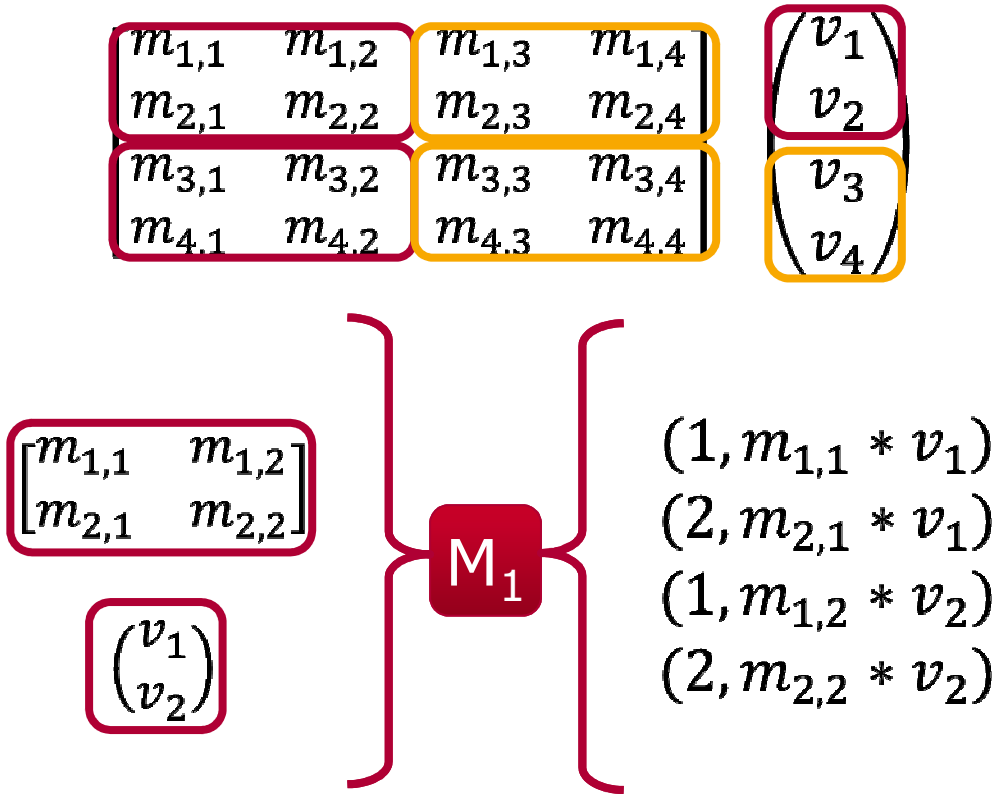
- Problems: Entire vector is needed on one map instance, no parallelization

Strip and Conquer



- Just part of the vector needed in memory
- But: No benefit from combining

Partitioning into Blocks



- Just part of the vector needed in memory
- And: Combining reduces network load

Overview

1. PageRank Motivation
2. PageRank Calculation
3. Efficient Computation with MapReduce
4. Extensions

Topic-sensitive PageRank

- User searches for „jaguar"
  - We know the user is interested in cars
  - → We need a special PageRank for cars!
- Idea: We start on a random page about cars
  - Either, we click on a random link
  - Or we jump randomly **to another car page**
- Intuition: Pages linked to by car pages are likely to be also about cars
- Challenges:
  - Decide for a set of topics
  - Detect the topic of a page

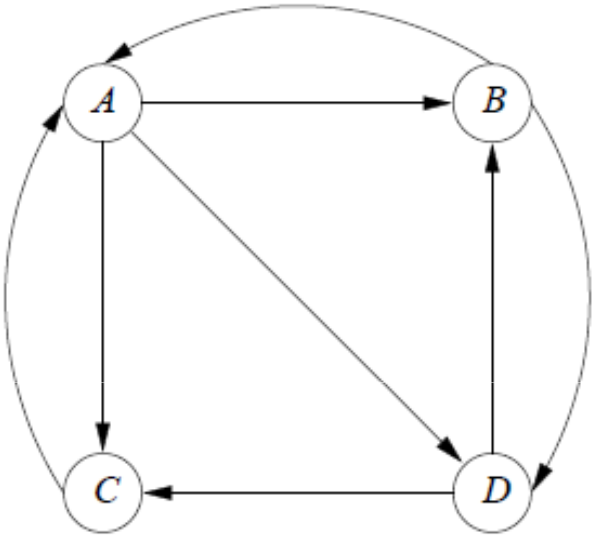
Fighting Link Spam with TrustRank

- PageRank is still not perfect
- How to give low PageRank to spam pages?
  → TrustRank

- Intuition: trustworthy pages don't link to spam pages
- Modify random surfer: We start on a random **trustworthy** page
- Either, we click on a random link
  - Or we jump randomly **to another trustworthy page**

- How to know trustworthy pages?
  - Let humans decide
  - Pick controlled domain (.edu, .gov)

Summary



$$\begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}$$

Data Set

Innsbruck

Country	Austria
State	Tyrol
Administrative region	Statutory city
Population	117,342 (2006)
Area	104.91 km²
Population density	1,119 /km²
Elevation	574 m
Coordinates	47°16' N 11°23' E
Postal code	6010-6080
Area code	0512
Licence plate code	I
Mayor	Hilde Zach
Website	www.innsbruck.at

Language	Number of Abstracts
English	3,550,000
German	1,137,000
French	1,111,000
Spanish	825,000
Japanese	821,000
Polish	776,000
Dutch	747,000
Italian	746,000
Portuguese	703,000
Swedish	391,000
Chinese	340,000