



Link Analysis

Yannis Kotidis

<http://pages.cs.aueb.gr/~kotidis/>

Acknowledgments

- Material adapted from the textbook “Mining Massive Datasets” available at <http://infolab.stanford.edu/~ullman/mmds.html>

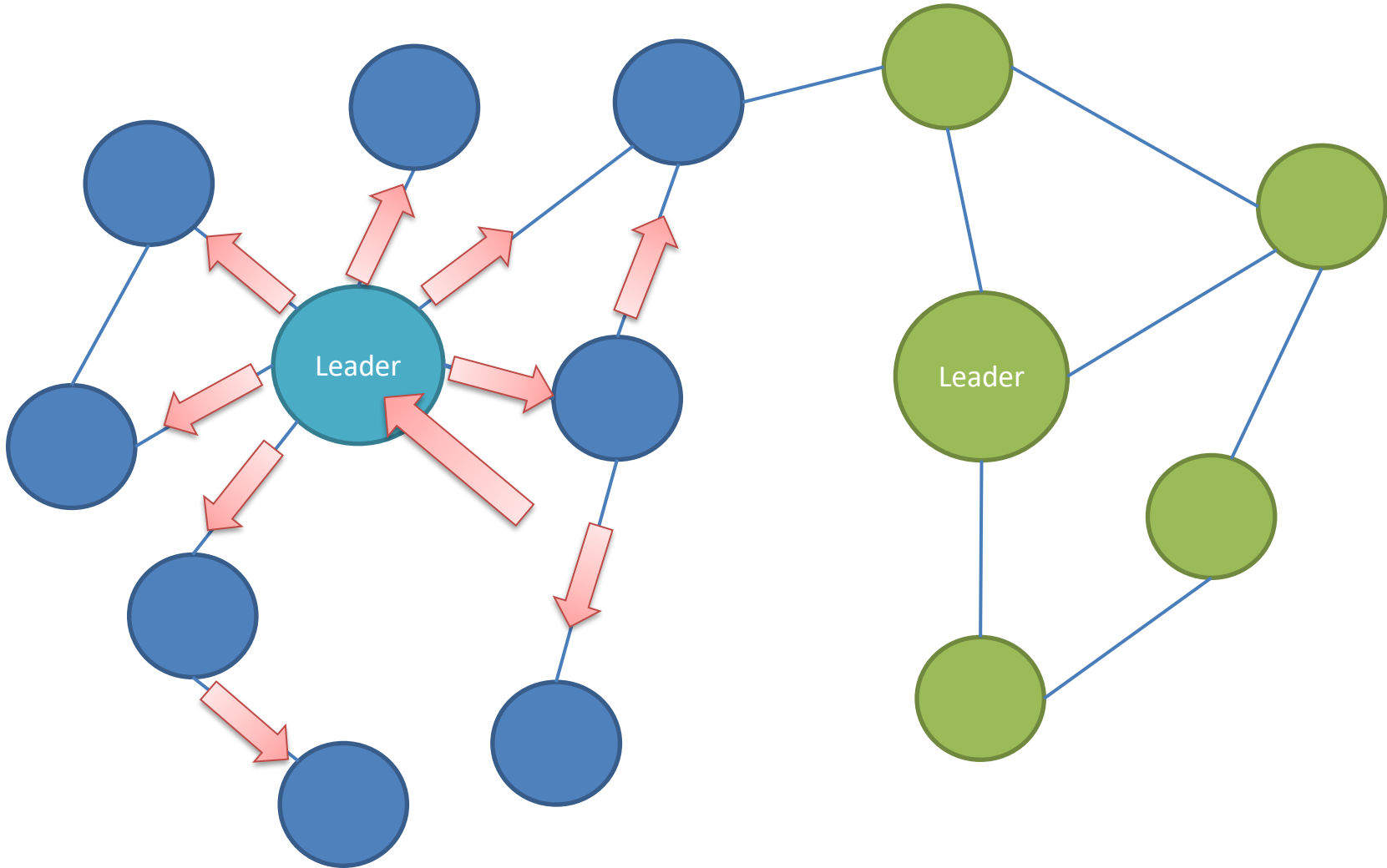
Roadmap

- Need for link analysis
- PageRank: intuition and formulation
- Computation of PageRank
 - Power Iteration Method
 - Web Surfer model
 - Markov Chain model
 - Google’s algorithm
- Topic-specific PageRank
- Circle of Trust (Twitter)
- HITS: hubs and authorities
- Spamming
 - Creating spam, spam-farms
 - Fighting spam, TrustRank
 - Spam estimation: Spam-mass
- Using pageRank as a measure of “importance”

Social Graphs



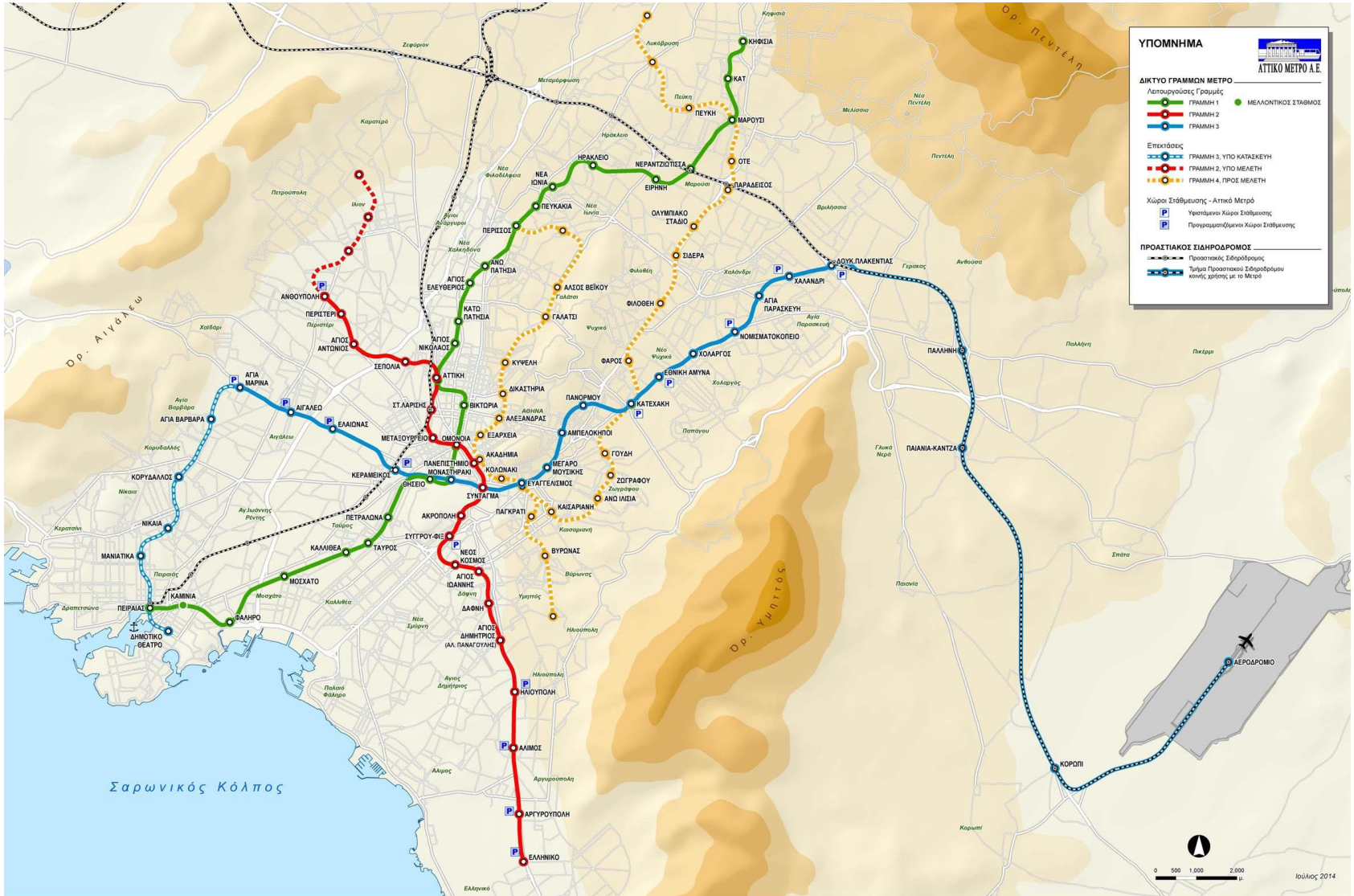
SNA: Marketing campaign



Citation Networks



Transport Networks



The WWW

Yannis Kallitsis

I am an associate professor at the Department of Informatics, Athens University of Economics and Business. I received my Ph.D. in Computer Science from the University of Athens in 2004. I was an assistant professor at the Department of Informatics, Athens University of Economics and Business from 2004 until January of 2006.

Research Interests

- Big Data, graph databases, On-Line Analytical Processing
- Sensor P2P networks, mobile data management, data mining
- Data streams, RFID data management, approximate query processing
- Database preservation, integration.

Publications available [here](#)

Selected Recent Publications

- 1) Inference Applications: Connections with Graph Databases
- 2) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 3) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 4) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 5) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 6) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 7) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 8) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 9) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 10) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 11) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 12) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 13) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 14) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 15) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 16) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 17) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 18) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 19) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.
- 20) Y. Kallitsis, Y. Kostas, & D. Papadimitriou, "Y. Kallitsis, Y. Kostas, D. Papadimitriou, Information Systems, Volume 31(1), 2010, pp. 1-14.



Department of Informatics
Athens University of Economics and Business

Home Page | Maps & Directions | Contact | Ελληνικά | English

Department Undergraduate Studies
People Infrastructure and Services
News
Log In

About the Department

The Department of Informatics traces its history to the Department of Statistics and Informatics originally established in 1984. It is the third University department in Greece to specifically address Information Technology and Computer Science in its curriculum.

ΑΝΑΖΗΤΗΣΗ

ΓΡΗΓΟΡΗ ΠΡΟΣΒΑΣΗ

Ε-Shop
Ε-Γραμματεία
Webmail
Webmail Προστυλιών
Διαχείριση Βιβλίων
Επισκοπείο για Σίματα Εκπαιδευτές
Σύστημα δήλωσης Βιβλίων

ΑΝΑΚΟΙΝΩΣΕΙΣ

Το Παιδαγωγικό τμήμα της Παιδαγωγικής Σχολής του Πανεπιστημίου Αθηνών στην Αθήνα Bank περιλαμβάνει...

Τα Νέα του Πανεπιστημίου

Η περίοδος υποβολής των αιτήσεων για τα ματς...

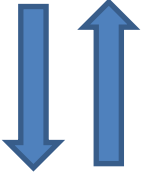
Σχέδιο Οργανισμού ΟΓΙΑ

Σημεία Συγκρήτου

ΕΠΙΧΕΙΡΑ

Παράκληση Υπότροφης Υποτροφίας
Προκήρυξη Φορέων Πανεπιστημίου
16-10-2014 11:30

Παράκληση μελών του συλλόγου στους εκπαιδευτές φοιτητές



Web graph is large

- How to query and locate relevant and trustworthy information in the web?
 - **Human curated approach**: build web directories (e.g. Yahoo, DMOZ)
 - **Automated approach**: adapt IR (keyword search)
 - Inverted lists, Latent Semantic Indexing,...
- Issues: scale of the web, heterogeneity of pages, lots of untrusted sources, **spam**, etc.

Spamming

- Deliberate action to boost a web page's position in search engine results
- (Assume that) search engines index pages by the keywords they contain
- I want to increase visibility of my web page that sells toasters: cheapToasters.com
 - Ideas?

cheapToasters.com



Simple trick

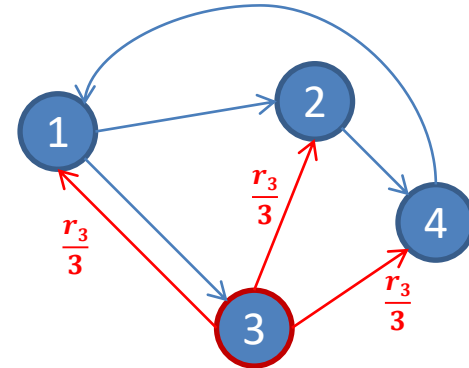
- Pick a popular search term, e.g. “**movies**”
- Want to make my *cheapToasters.com* page appear in movie searches
- Make my page look similar to the top-ranking pages in that result listing
 - Pick few pages that come on-top when searching for movies
 - Copy keywords from these pages (or whole paragraphs), paste them into my *cheapToasters.com* page but make them invisible when rendering the page in the browser.

Link Analysis

- Believe what people say about you, rather than what you say about yourself
 - Use information from the web link structure in order to measure the importance of web pages, when ranking the results of a query
- Assumption: trustworthy/authoritative sources may link to each other

How to measure importance

- An incoming link acts as a positive testimony (a “vote”)
 - Votes from important pages count more
- Weight of a vote is **equally split** between all outgoing edges
 - Fig: node 3 has three outgoing edges
- Votes received by node 1:
 - $r_1 = r_3/3 + r_4$



$$r_1 = r_3/3 + r_4$$

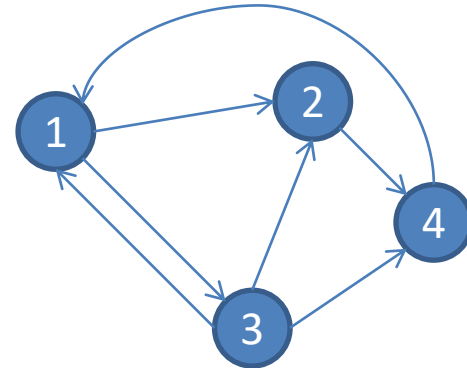
$$r_2 = r_1/2 + r_3/3$$

$$r_3 = r_1/2$$

$$r_4 = r_2 + r_3/3$$

Solution

- 4 equations/4 unknowns
 - No unique solution
- Add another constraint:
 - All ranks sum up to 1
- Solution:
 - $r_1 = 6/18, r_2 = 4/18$
 - $r_3 = 3/18, r_4 = 5/18$



$$r_1 = r_3/3 + r_4$$

$$r_2 = r_1/2 + r_3/3$$

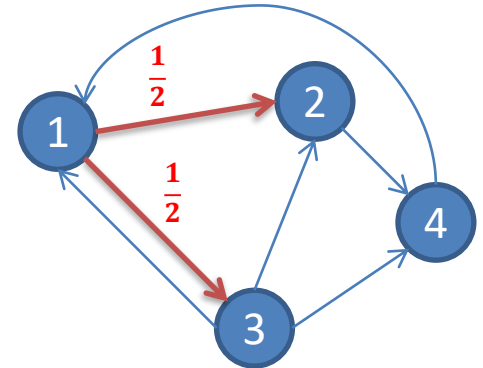
$$r_3 = r_1/2$$

$$r_4 = r_2 + r_3/3$$

$$r_1 + r_2 + r_3 + r_4 = 1$$

Matrix representation of the WWW

- Let matrix W denote the web graph
 - $W_{ji} = 1/d_i$, if link $i \rightarrow j$ appears in the web



From:

To:

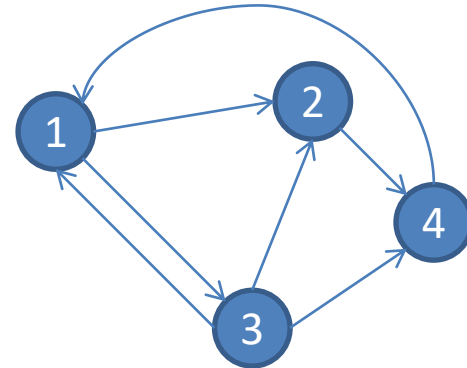
		1/3	1
1/2		1/3	
1/2			
	1	1/3	

columns summing to 1

Store ranks in a vector \mathbf{r}

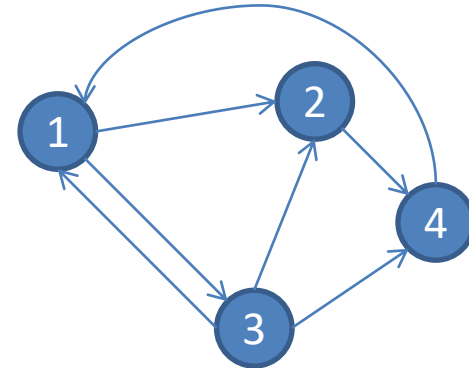
- $\mathbf{r} = (r_1, r_2, r_3, r_4)^T$
- $r_1 + r_2 + r_3 + r_4 = 1$

$$\mathbf{r} = \begin{array}{|c|} \hline r_1 \\ \hline r_2 \\ \hline r_3 \\ \hline r_4 \\ \hline \end{array}$$



Equivalent matrix formulation

- Previous equations are expressed as : $W^*r = r$



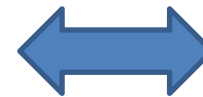
		1/3	1
1/2		1/3	
1/2			
	1	1/3	

 ·

r_1
r_2
r_3
r_4

 =

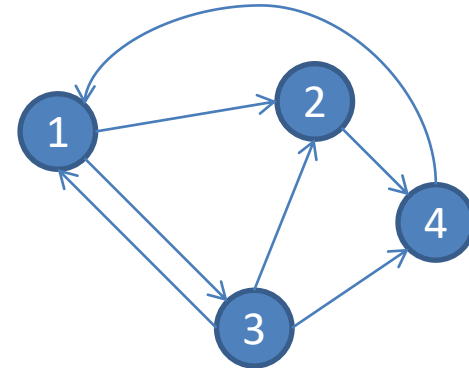
r_1
r_2
r_3
r_4



$$\begin{aligned}r_1 &= r_3/3 + r_4 \\r_2 &= r_1/2 + r_3/3 \\r_3 &= r_1/2 \\r_4 &= r_2 + r_3/3 \\r_1 + r_2 + r_3 + r_4 &= 1\end{aligned}$$

Ranks as eigenvectors

- $A^*x = \lambda^*x$, iff x is an eigenvector of matrix A
 - λ is the eigenvalue of x
- Recall $W^*r = 1^*r$
 - Thus, r is the principal eigenvector (for $\lambda=1$)



		1/3	1
1/2		1/3	
1/2			
	1	1/3	

 \cdot

1/3
2/9
1/6
5/18

 $=$

1/3
2/9
1/6
5/18

Power Iteration Method

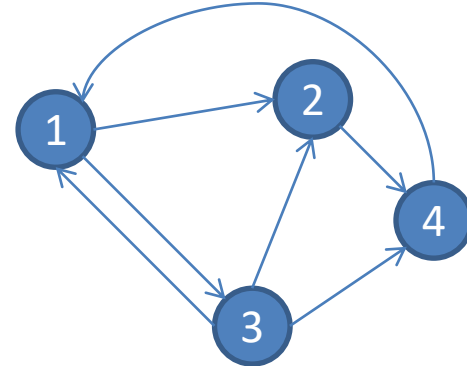
- Start with any valid (=ranks sum to 1) vector $r(0)$
 - For instance, initialize ranks as $r_i = 1/n$, where n is the number of nodes in the graph
- At step i compute $r(i) = W * r(i-1)$
- Stop when you have converged to a solution
 - E.g. $|r(i) - r(i-1)| < \epsilon$, for some small constant ϵ



Distance (e.g. L_2) of the two solutions is small

Example

- Set $r(0) = (1/4, 1/4, 1/4, 1/4)^T$



		1/3	1
1/2		1/3	
1/2			
	1	1/3	

 ·

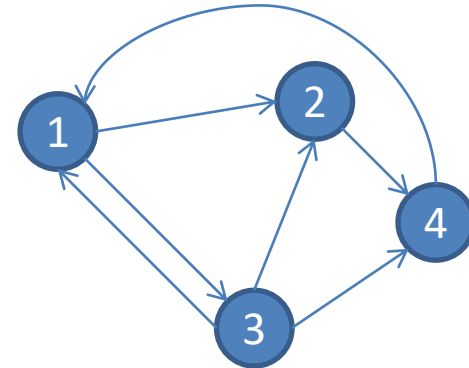
1/4
1/4
1/4
1/4

 =

1/3
5/24
1/8
1/3

Example

- Set $r(1) = (1/3, 5/24, 1/8, 1/3)^T$



		1/3	1
1/2		1/3	
1/2			
	1	1/3	

 ·

1/3
5/24
1/8
1/3

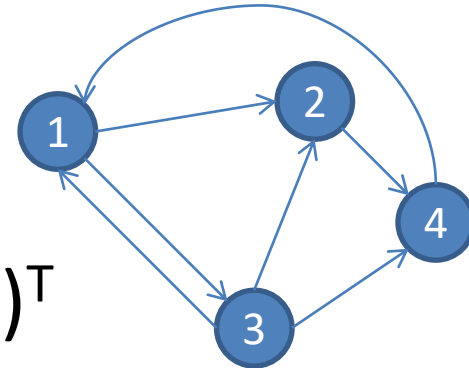
 =

27/72
17/72
1/6
14/72

Example

- Set

$$r(2) = (27/72, 17/72, 1/6, 14/72)^T$$



		1/3	1
1/2		1/3	
1/2			
	1	1/3	

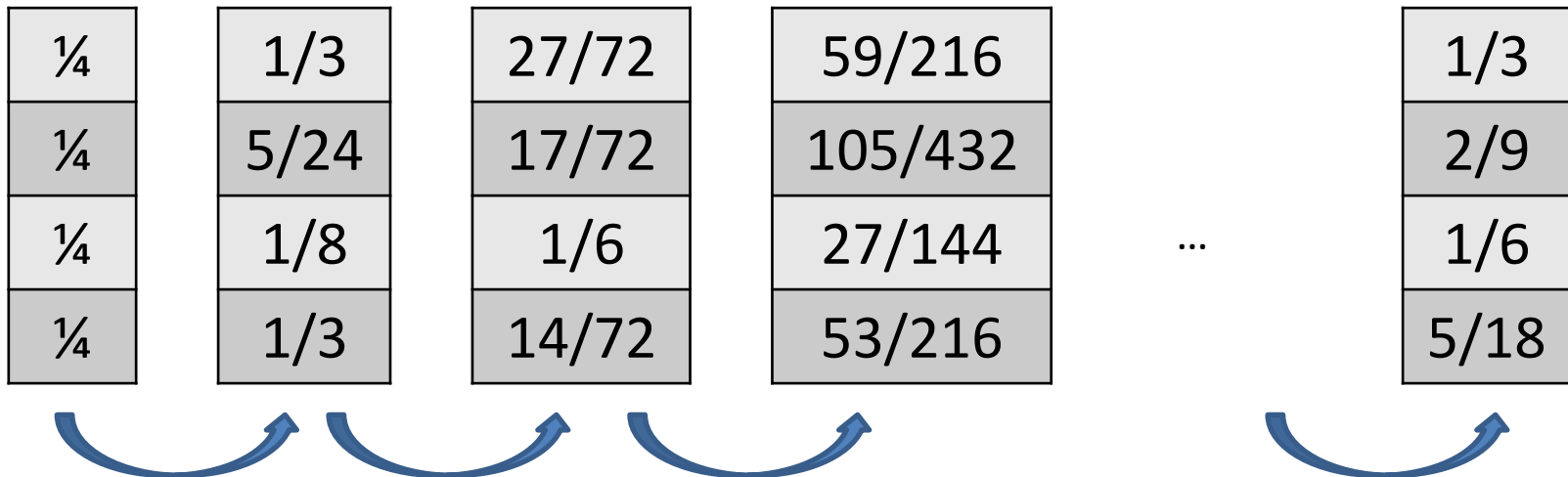
 \cdot

27/72
17/72
1/6
14/72

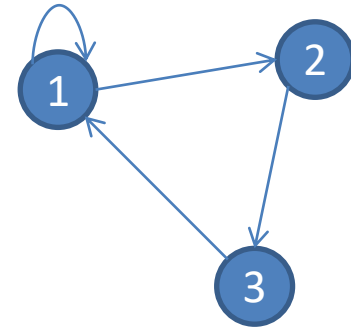
 $=$

59/216
105/432
27/144
53/216

Convergence



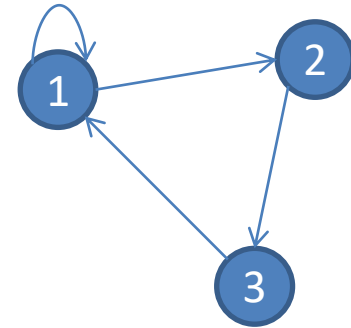
Example 2



- Set $r(0) = (1/3, 1/3, 1/3)^T$

$$\begin{bmatrix} 1/2 & & 1 \\ 1/2 & & \\ & 1 & \end{bmatrix} \cdot \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/6 \\ 1/3 \end{bmatrix}$$

Example 2



- Set $r(1) = (1/2, 1/6, 1/3)^T$

1/2		1
1/2		
	1	

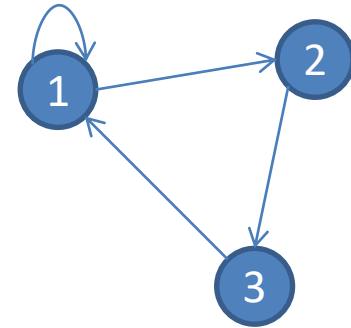
 \cdot

1/2
1/6
1/3

 $=$

7/12
1/4
1/6

Example 2



- Set $r(2) = (7/12, 1/4, 1/6)^T$

1/2		1
1/2		
	1	

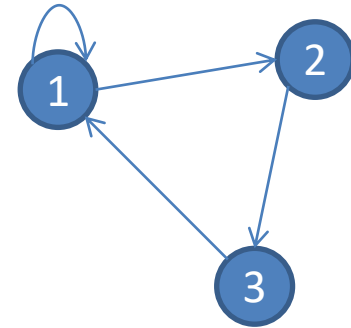
 ·

7/12
1/4
1/6

 =

11/24
7/24
1/4

Example 2



- Set $r(3) = (11/24, 7/24, 1/4)^T$

1/2		1
1/2		
	1	

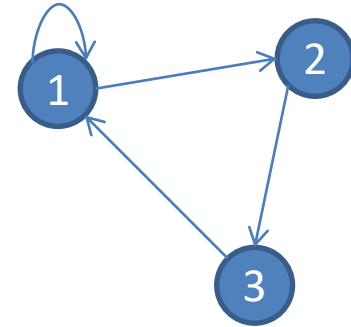
 \cdot

11/24
7/24
1/4

 $=$

23/48
11/48
7/24

Example 2



- Set $r(3) = (23/48, 11/48, 7/24)^T$

1/2		1
1/2		
	1	

 ·

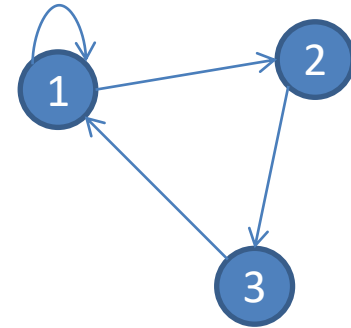
23/48
11/48
7/24

 =

51/96
23/96
11/48

Example 2

- Next iteration



$1/2$		1
$1/2$		
	1	

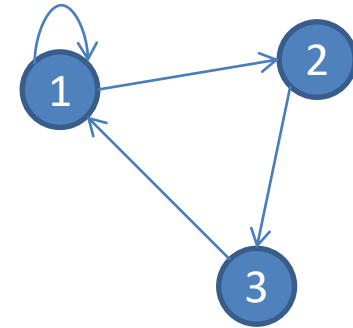
 ·

$51/96$
$23/96$
$11/48$

 =

$95/192$
$51/192$
$23/96$

Example 2

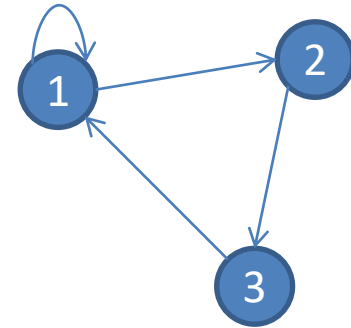


- Next iteration

$$\begin{array}{|c|c|c|} \hline 1/2 & & 1 \\ \hline 1/2 & & \\ \hline & 1 & \\ \hline \end{array} \cdot \begin{array}{|c|} \hline 95/192 \\ \hline 51/192 \\ \hline 23/96 \\ \hline \end{array} = \begin{array}{|c|} \hline 1/2 \\ \hline 95/384 \\ \hline 51/192 \\ \hline \end{array}$$

Example 2

- In the end...



1/2		1
1/2		
	1	

 \cdot

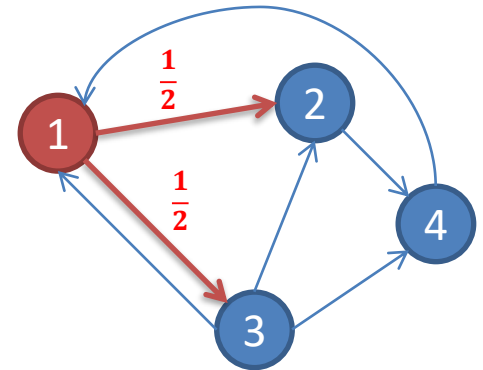
1/2
1/4
1/4

 $=$

1/2
1/4
1/4

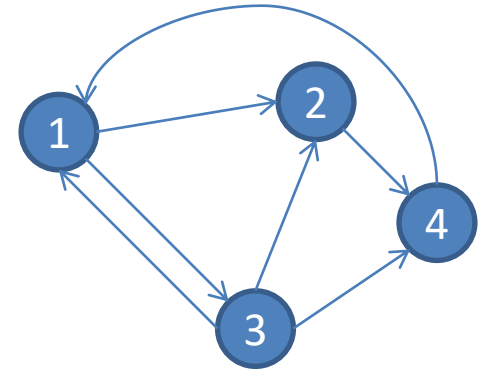
Random Web Surfer

- Surfer starts at arbitrary node
- Picks one outgoing link at random and follows it
- E.g. assume surfer is at node 1
 - with probability $\frac{1}{2}$ she jumps to node 2, OR
 - with probability $\frac{1}{2}$ she jumps to node 3



Matrix formulation

- Let vector $\mathbf{p}^t = (p^t_1, p^t_2, \dots, p^t_n)$
- Each coordinate p^t_i denotes the probability that the surfer is at node i at time t
- Where is the surfer at time $t+1$?



$$\mathbf{p}^{t+1} = \mathbf{W} * \mathbf{p}^t$$

Markov process/random walk

What happens in the end

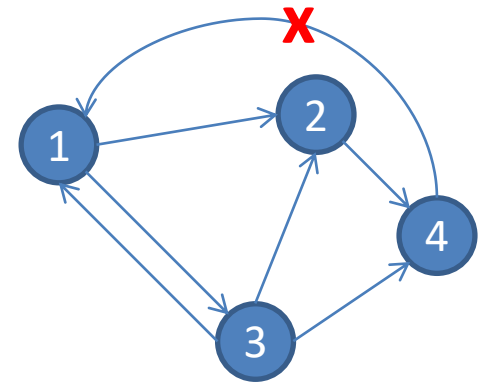
- Assume that after some (possibly long) time, the probability distribution on the location of the surfer reaches a state where $p^t = W^* p^t$
- Then, p^t is **stationary distribution** of a random walk
- Recall that in the power iteration method we were looking for ranks such that $r = W^* r$
- Thus, the ranks in r define a stationary distribution of the random walk performed by the surfer

Convergence

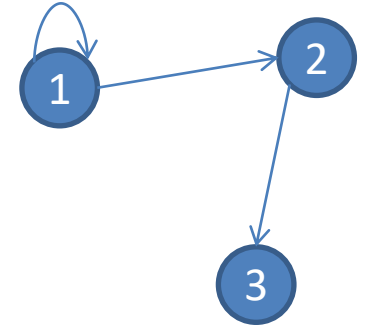
- For graphs that satisfy certain condition, the random walk always reaches a stationary distribution, no matter the initial conditions
 - What can go wrong?

Problem #1: Dead ends

- Node #4 has no outgoing edge
- Ranks (votes) that flow to that node, disappear in the next iteration!
- Equivalently: the surfer gets trapped on that node



Simpler example



1/2	0	0
1/2	0	0
0	1	0

 \cdot

1/3
1/3
1/3

 $=$

1/6
1/6
1/3

} SUM = 2/3

1/2	0	0
1/2	0	0
0	1	0

 \cdot

1/6
1/6
1/3

 $=$

1/12
1/12
1/6

} SUM = 1/3

1/2	0	0
1/2	0	0
0	1	0

 \cdot

1/12
1/12
1/6

 $=$

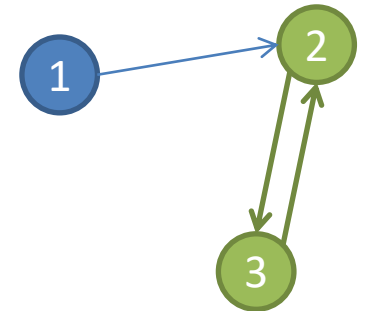
1/24
1/24
1/12

} converges to?

0
0
0

Problem #2: Closed communities

- Also called spider-traps
- Flow of ranks gets trapped inside them
- Surfer has no way out!

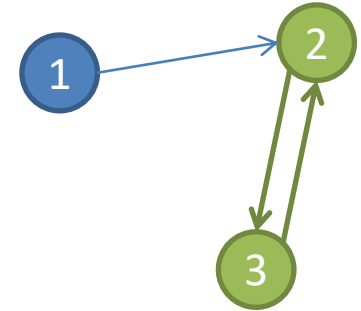


Problem #2: Closed communities

$$\begin{array}{|c|c|c|} \hline & & \\ \hline 1 & & 1 \\ \hline & 1 & \\ \hline \end{array} \cdot \begin{array}{|c|} \hline 1/3 \\ \hline 1/3 \\ \hline 1/3 \\ \hline \end{array} = \begin{array}{|c|} \hline 0 \\ \hline 2/3 \\ \hline 1/3 \\ \hline \end{array}$$

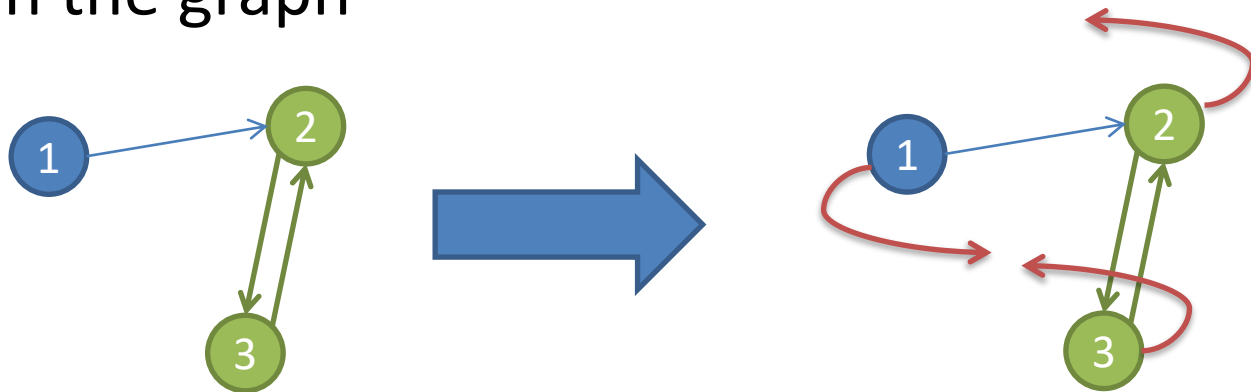
$$\begin{array}{|c|c|c|} \hline & & \\ \hline 1 & & 1 \\ \hline & 1 & \\ \hline \end{array} \cdot \begin{array}{|c|} \hline 0 \\ \hline 2/3 \\ \hline 1/3 \\ \hline \end{array} = \begin{array}{|c|} \hline 0 \\ \hline 1/3 \\ \hline 2/3 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|} \hline & & \\ \hline 1 & & 1 \\ \hline & 1 & \\ \hline \end{array} \cdot \begin{array}{|c|} \hline 0 \\ \hline 1/3 \\ \hline 2/3 \\ \hline \end{array} = \begin{array}{|c|} \hline 0 \\ \hline 2/3 \\ \hline 1/3 \\ \hline \end{array}$$



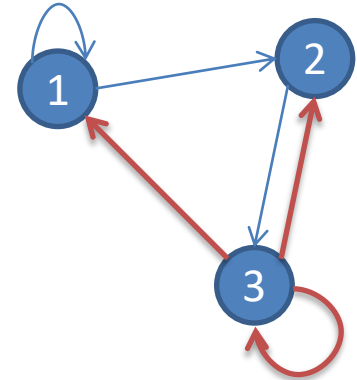
Google's solution: add teleports

- At each iteration, the surfer does one of the following
 - With probability β , she follows an outgoing link at random
 - With probability $1-\beta$, she jumps to a random node in the graph



Dead ends

- Always perform a random teleport from dead-ends



Google's Page Rank formulation*

$$r_j = \sum_{i \rightarrow j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{n}$$

Remember:

- r_i : page rank of node i
- d_i : degree (fan-out) of node i
- n : number of nodes in the graph (web)
- β : a number close to 1 (e.g. 0.8-0.9)

* Formula assumes no-dead-ends (explicitly perform random jumps with prob=1 from these nodes)

Does this converge?

- The power-iteration method converges to a stationary distribution if matrix W is stochastic, irreducible and aperiodic.

$$p^{t+1} = W * p^t$$

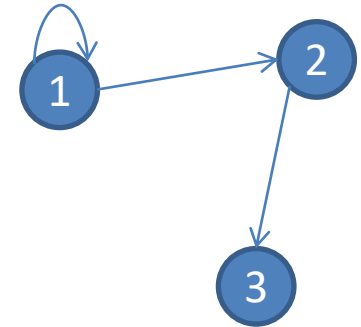
Is W stochastic?

- Stochastic \Leftrightarrow columns add to 1

1/2		
1/2		
	1	



Sum = 0 for dead-ends



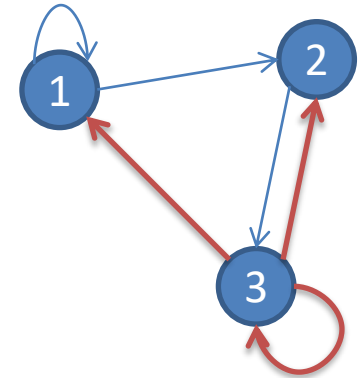
Is W stochastic?

- Stochastic \Leftrightarrow columns add to 1

1/2		1/3
1/2		1/3
	1	1/3



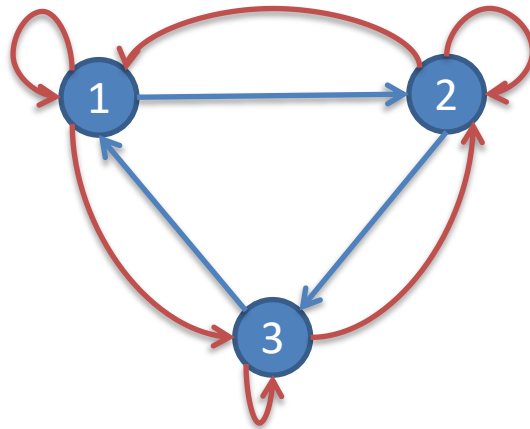
Make column stochastic



Note: while teleports are the solution for dead-ends, in practice the pageRank algorithm does not change the matrix W on dead-ends (otherwise these columns require $O(n)$ space). Instead it explicitly makes a random jump from these nodes.

Is W aperiodic?

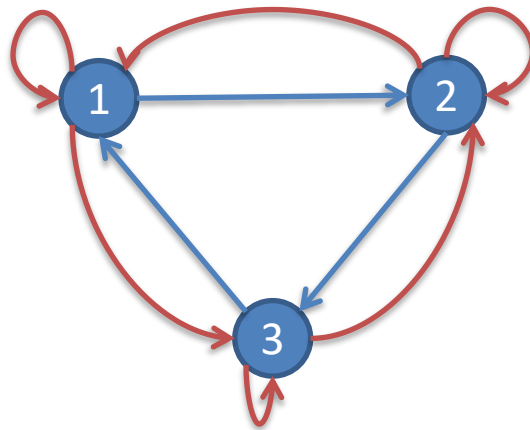
- A Markov-chain is periodic if there exists $k > 1$ such that the interval between two visits to some state s is always a multiple of k



Note: not all teleports required for making W aperiodic

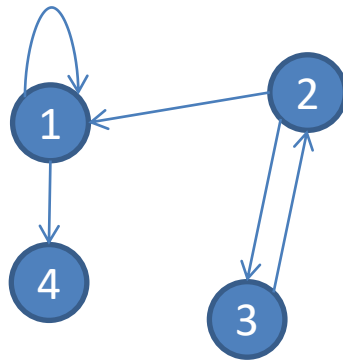
Is W irreducible?

- A Markov-chain is irreducible if from any state, there is a non-zero probability of going from any one state to any another



Google Matrix (no dead-ends)

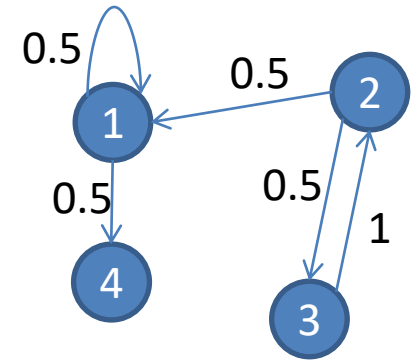
- $G = \beta * W + (1 - \beta) / n * E$
 - E: $n \times n$ matrix with all ones
- Example: make the matrix of the following graph stochastic, aperiodic and irreducible



Example

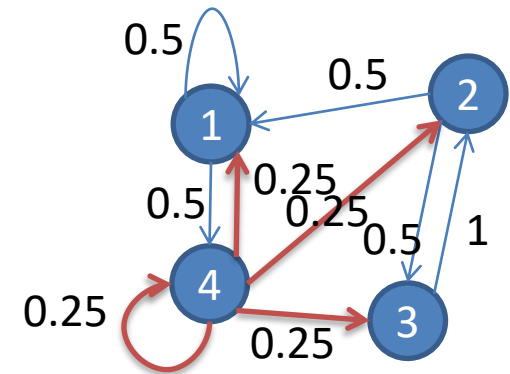
- Original matrix W :

0.5	0.5		
		1	
	0.5		
0.5			



- Node 4 is a dead-end. Compute W' :

0.5	0.5		0.25
		1	0.25
	0.5		0.25
0.5			0.25



Example (cont.)

- Adjusted matrix W' :

0.5	0.5		0.25
		1	0.25
	0.5		0.25
0.5			0.25

- Assume $\beta=0.8$. Thus, $G = 0.8*W + 0.2/4*E$

- New matrix G :

$0.8*0.5 + 0.05$	$0.8*0.5 + 0.05$	0.05	$0.8*0.25 + 0.05$
0.05	0.05	$0.8*1 + 0.05$	$0.8*0.25 + 0.05$
0.05	$0.8*0.5 + 0.05$	0.05	$0.8*0.25 + 0.05$
$0.8*0.5 + 0.05$	0.05	0.05	$0.8*0.25 + 0.05$

=

0.45	0.45	0.05	0.25
0.05	0.05	0.85	0.25
0.05	0.45	0.05	0.25
0.45	0.05	0.05	0.25

Original Matrix

- Notice that original matrix is very sparse

0.5	0.5		
		1	
	0.5		
0.5			

However

- Matrix G requires n^2 space:

0.45	0.45	0.05	0.25
0.05	0.05	0.85	0.25
0.05	0.45	0.05	0.25
0.45	0.05	0.05	0.25

- Do not want to compute it for large graphs (or the web)
- Instead, write recursion as:
$$r = \beta * W * r + (1 - \beta) / n * e$$
 - e: a vector with n coordinates that are all 1
- Formula assumes no dead-ends (\rightarrow ranks sum to 1)
 - Thus, in practice some pagerank will get lost
 - Google's algorithm: correct ranks after each iteration to make them sum up to one

Google's algorithm

- **Given web graph W (with dead-ends and closed communities), β**

Let $r^0 = (1/n, \dots, 1/n)$;

$t=0$;

Do {

$t++$;

 Compute $r^t = \beta * W * r^{t-1}$;

 Let $L = \sum r^t(i)$; // $L \leq \beta$ due to dead-ends

 //Re-insert leaked page-rank:

$r^t(i) = r^t(i) + (1-L)/n$; //now all ranks add to 1

 } while $|r^t(i) - r^{t-1}(i)| > \epsilon$;

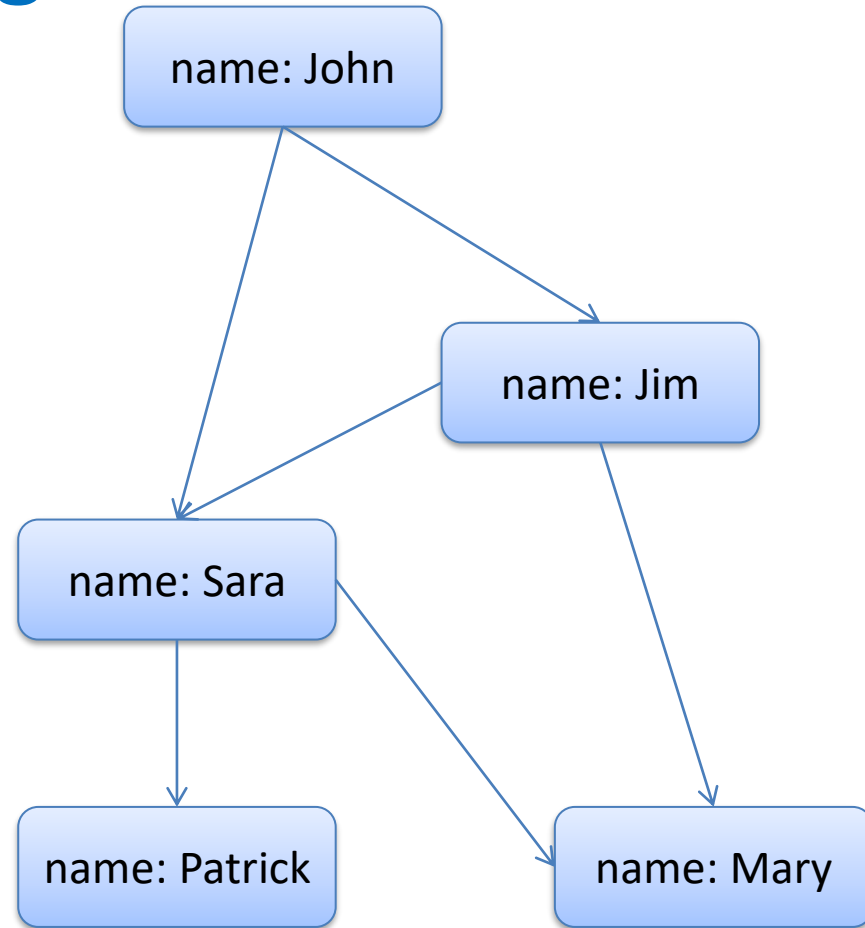
Note

- Assume $n=1$ billion web nodes
- Assume ranks are stored as 4 bytes numbers
- Dense matrix would require:
 $1\text{B} \times 1\text{B}$ numbers = $4 * 10^{18}$ bytes = ~ 3.5 Zettabytes
- Google's algorithm does not pre-adjust the matrix W . Instead it re-inserts leaked page-rank back into computation
- Original matrix is very sparse: few links per node
 - Store W row-wise (=for each node, keep list of incoming edges)
 - Inverted web-graph computed easily via map-reduce
 - We need $14 * 1$ billion ids $\sim 56\text{GB}$
 - $r(t)$, $r(t-1)$ vectors need $\sim 4\text{GB}$ each
 - Thus, computation requires a server with 64GB available memory

GraphX:

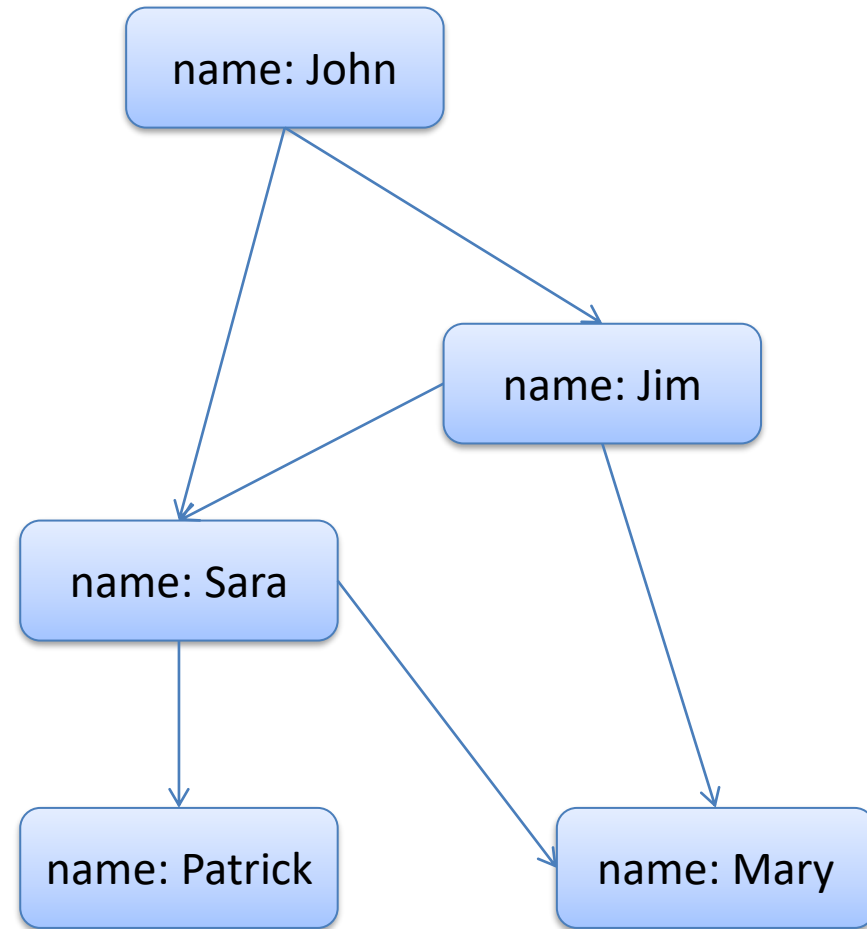
Define nodes using a DataFrame

```
val v =  
  spark.sqlContext.create  
  DataFrame(List(  
    ("john", "John", 29),  
    ("sara", "Sara", 22),  
    ("jim", "Jim", 42),  
    ("patrick", "Patrick", 19),  
    ("mary", "Mary", 31)  
  )).toDF("id", "name",  
    "age")
```



Now define edges & GraphFrame

```
val e =  
  spark.sqlContext.createData  
  Frame(List(  
    ("john", "sara", "knows"),  
    ("john", "jim", "knows"),  
    ("jim", "sara", "knows"),  
    ("jim", "mary", "knows"),  
    ("sara", "patrick", "knows"),  
    ("sara", "mary", "knows")  
  )).toDF("src", "dst",  
    "relationship")  
val g = GraphFrame(v, e)
```

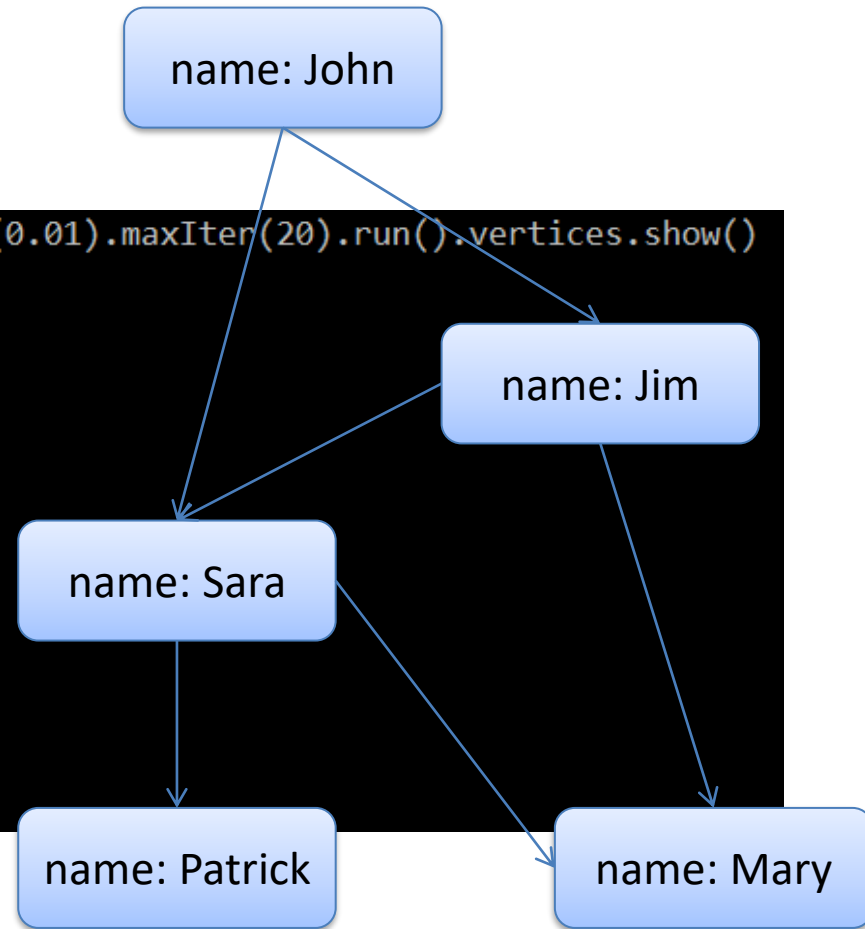


Computing PageRank: Spark/GraphX

```
scala> val results = g.pageRank.resetProbability(0.01).maxIter(20).run().vertices.show()
+-----+-----+-----+-----+
|   id|   name|age|          pagerank|
+-----+-----+-----+-----+
| mary|   Mary|31|1.4698147724378927|
| john|   John|29|0.5163835727128357|
| sara|   Sara|22|1.1541301946025058|
| jim|    Jim|42|0.7719934412056895|
|patrick|Patrick|19|1.0876780190410762|
+-----+-----+-----+-----+

results: Unit = ()

scala> |
```



Create Social Graph script: Neo4j

```
//create social graph
//label each node as "Person"
create (john:Person {name:"John"})
create (sara:Person {name:"Sara"})
create (jim:Person {name:"Jim"})
create (patrick:Person {name:"Patrick"})
create (mary:Person {name:"Mary"})
create (john)-[:Knows]->(jim)
create (john)-[:Knows]->(sara)
create (jim)-[:Knows]->(sara)
create (sara)-[:Knows]->(patrick)
create (sara)-[:Knows]->(mary)
create (jim)-[:Knows]->(mary);
```

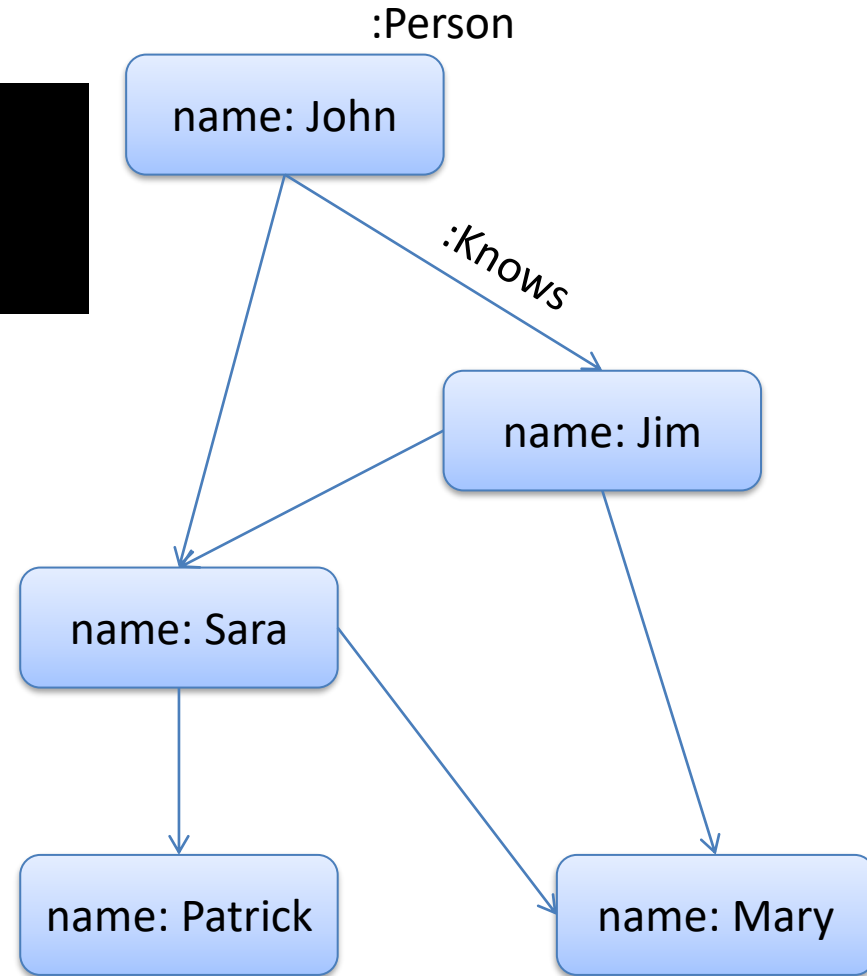
```
create (john:Person {name:"John"})
```

label property

variable, used later-on while
defining edges for this node

Computing PageRank: Neo4j

```
CALL algo.pageRank("Person", "Knows",  
{iterations: 20, damping: 0.85})
```



Topic-specific search

Search term: “apple”



?



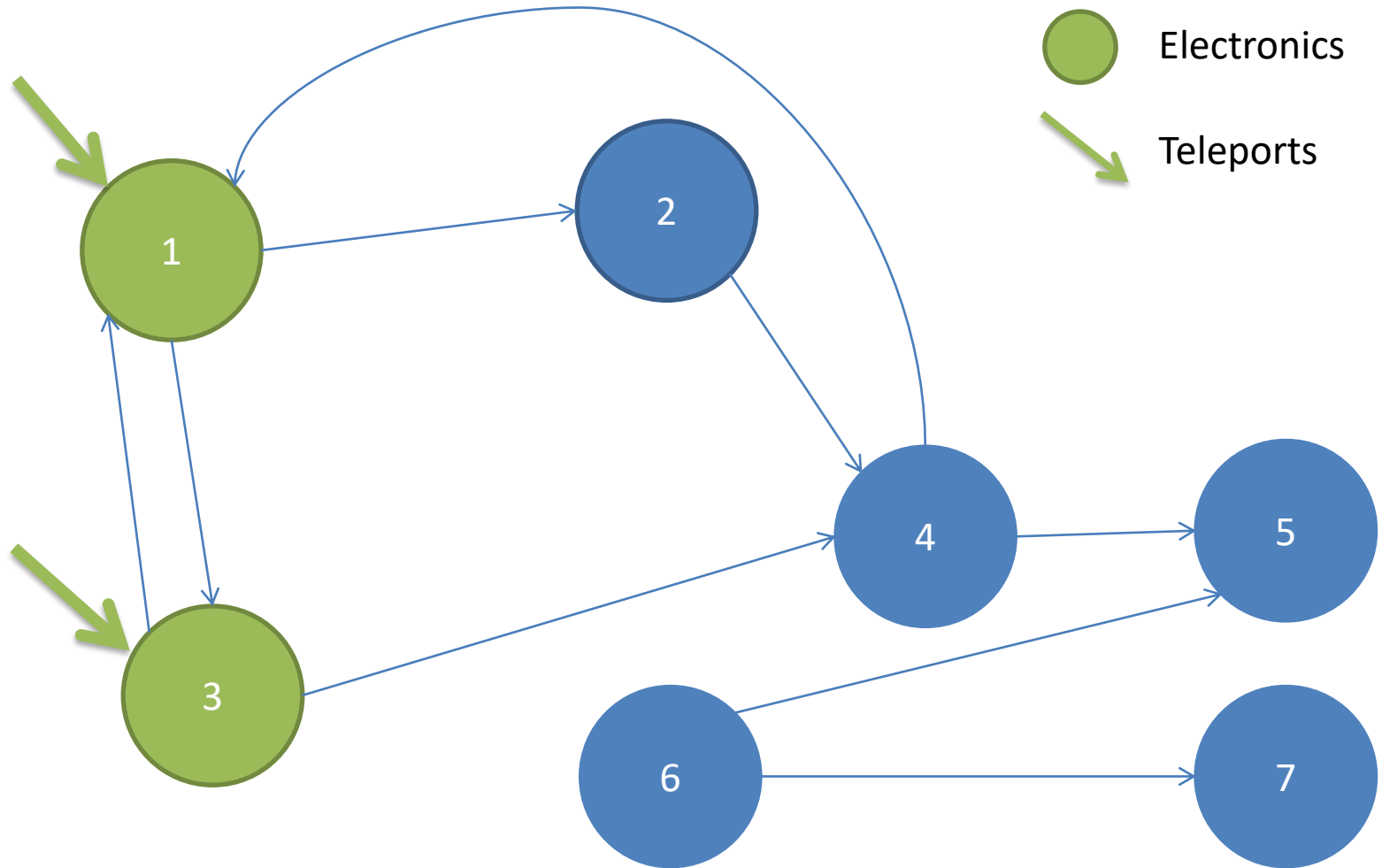
Topic-specific search

- Assume user is interested in a specific topic, e.g. 'electronics'
- Can we adjust PageRank calculations so that search favors pages from that topic?

Adjust definition of PageRank

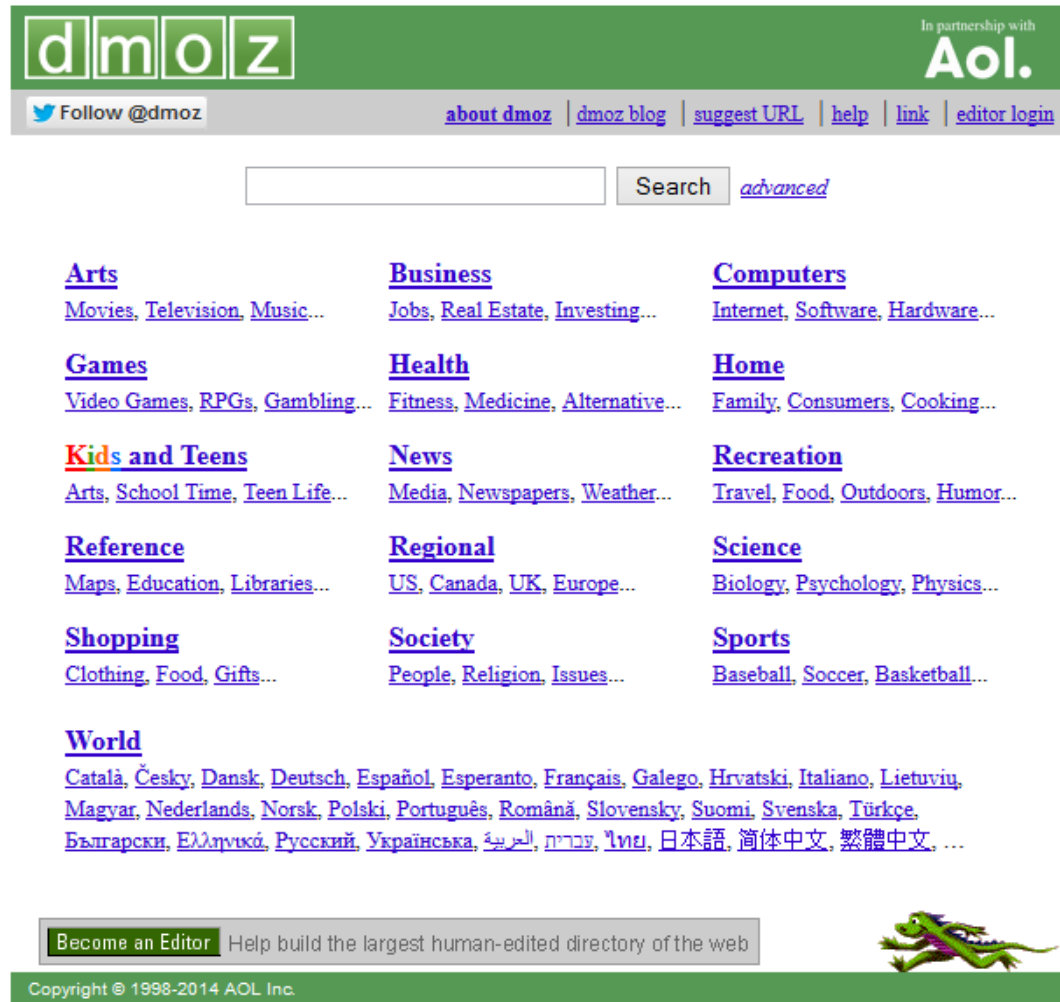
- An **electronics** web-site is important if it has incoming links from other important **electronics** sites
- Web-surfer: initiate random walks from a selected pool of good web-sites on electronics
 - = make restarts from selected electronics sites
 - = make teleports point back to selected electronics sites

Assume $T=\{1,3\}$ are well known electronics sites



DMOZ: The Open Directory

- Example: use 16 top-level categories from DMOZ



The screenshot shows the DMOZ website interface. At the top, there is a green header with the 'dmoz' logo and 'In partnership with AOL.' Below the header is a navigation bar with a Twitter follow button for '@dmoz' and links for 'about dmoz', 'dmoz blog', 'suggest URL', 'help', 'link', and 'editor login'. A search bar is located below the navigation bar, with a 'Search' button and a link to 'advanced'. The main content area displays 16 top-level categories in a grid format:

- Arts**: [Movies](#), [Television](#), [Music](#)...
- Business**: [Jobs](#), [Real Estate](#), [Investing](#)...
- Computers**: [Internet](#), [Software](#), [Hardware](#)...
- Games**: [Video Games](#), [RPGs](#), [Gambling](#)...
- Health**: [Fitness](#), [Medicine](#), [Alternative](#)...
- Home**: [Family](#), [Consumers](#), [Cooking](#)...
- Kids and Teens**: [Arts](#), [School Time](#), [Teen Life](#)...
- News**: [Media](#), [Newspapers](#), [Weather](#)...
- Recreation**: [Travel](#), [Food](#), [Outdoors](#), [Humor](#)...
- Reference**: [Maps](#), [Education](#), [Libraries](#)...
- Regional**: [US](#), [Canada](#), [UK](#), [Europe](#)...
- Science**: [Biology](#), [Psychology](#), [Physics](#)...
- Shopping**: [Clothing](#), [Food](#), [Gifts](#)...
- Society**: [People](#), [Religion](#), [Issues](#)...
- Sports**: [Baseball](#), [Soccer](#), [Basketball](#)...
- World**: [Català](#), [Česky](#), [Dansk](#), [Deutsch](#), [Español](#), [Esperanto](#), [Français](#), [Galego](#), [Hrvatski](#), [Italiano](#), [Lietuvių](#), [Magyar](#), [Nederlands](#), [Norsk](#), [Polski](#), [Português](#), [Română](#), [Slovensky](#), [Suomi](#), [Svenska](#), [Türkçe](#), [Български](#), [Ελληνικά](#), [Русский](#), [Українська](#), [العربية](#), [עברית](#), [עברית](#), [עברית](#), [日本語](#), [简体中文](#), [繁體中文](#), ...

At the bottom of the page, there is a green footer with a 'Become an Editor' button and the text 'Help build the largest human-edited directory of the web'. To the right of the footer is a small green lizard logo. Below the footer, the copyright information 'Copyright © 1998-2014 AOL Inc.' is displayed. At the very bottom, the statistics '4,144,204 sites - 89,923 editors - over 1,023,485 categories' and the build information 'Build 2.1.7.1-764145 Mon Nov 3 13:14:33 EST 2014' are shown.

4,144,204 sites - 89,923 editors - over 1,023,485 categories

Build 2.1.7.1-764145 Mon Nov 3 13:14:33 EST 2014

Topic-specific PageRank

- Let **T** be a set of pages on the topic
 - E.g. use Open Directory (DMOZ) pages for a given topic
- We want to **bias** PageRank calculations in favor of pages in set **T**
- Simple trick: make teleports point to those pages
 - Random surfers jumps to a random page from **T**, instead of teleporting anywhere
 - Can be adjusted so that pages in **T** have different bias

Topics?

- This trick implies that topic-specific pageRank calculations have been computed a-priori
 - E.g. each page has different PageRanks depending on the topic
 - Utilize appropriate topic-specific PageRank for a given query
- Each page now has a **PageRank vector** instead of a single value

Adjusted Google Matrix

$$G_{ij} = \begin{cases} \beta * W_{ij} + (1-\beta)/|T| , & \text{if page } i \text{ in } T \\ \beta * W_{ij} & , \text{ otherwise} \end{cases}$$

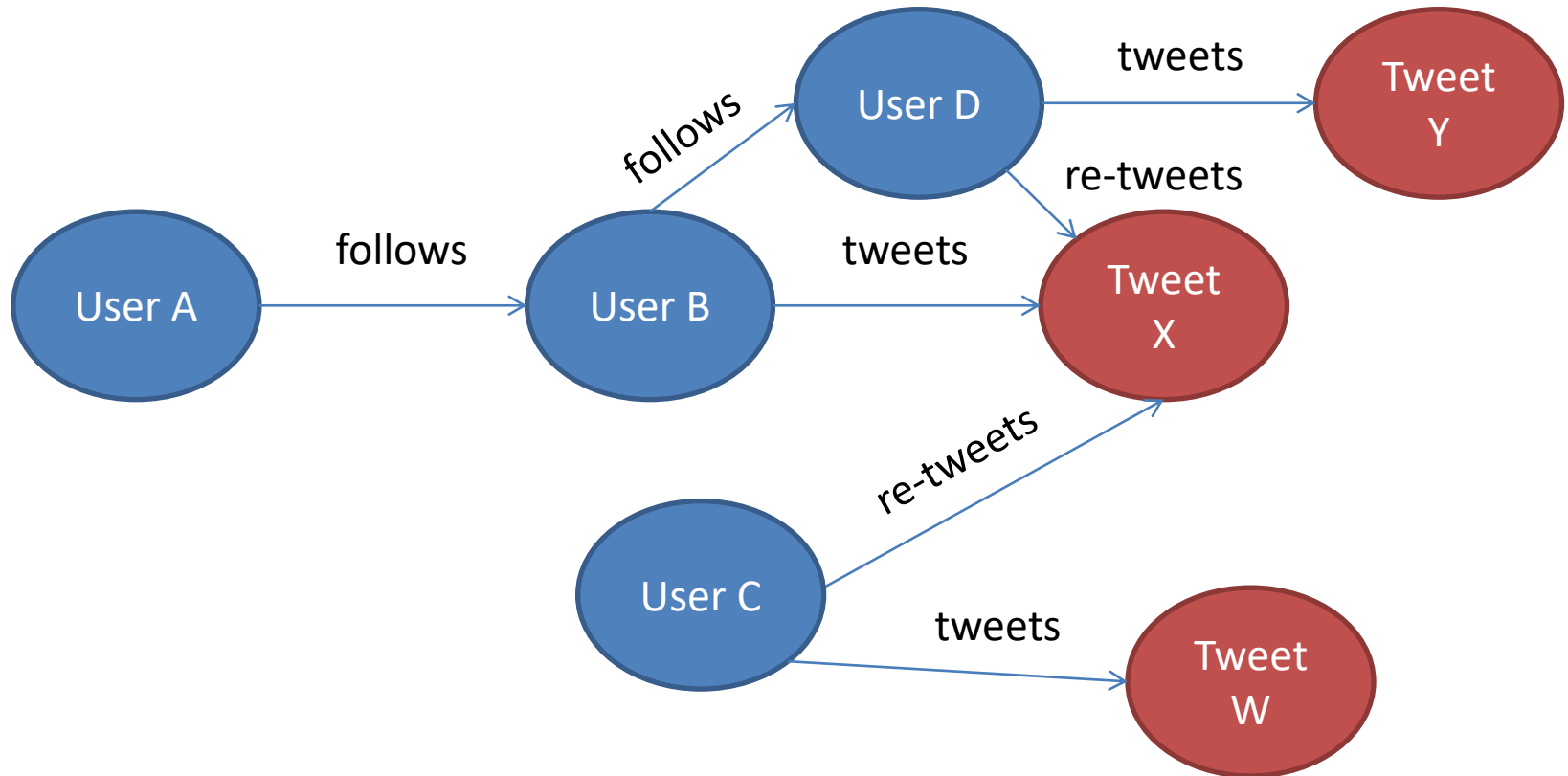
Who defines the topic on a given query?

- The user...
- Infer topic from other keywords in the same query (classification problem)
- Context (e.g. query coming from an electronics store, user search history)

More applications of pageRank

- pageRank ranks network nodes using incoming links as a notion of positive testimony for the worthiness of a node
- Many problems can be modeled in a similar setting
- Note difference between global and personalized ranking

Ranking tweets for a specific user

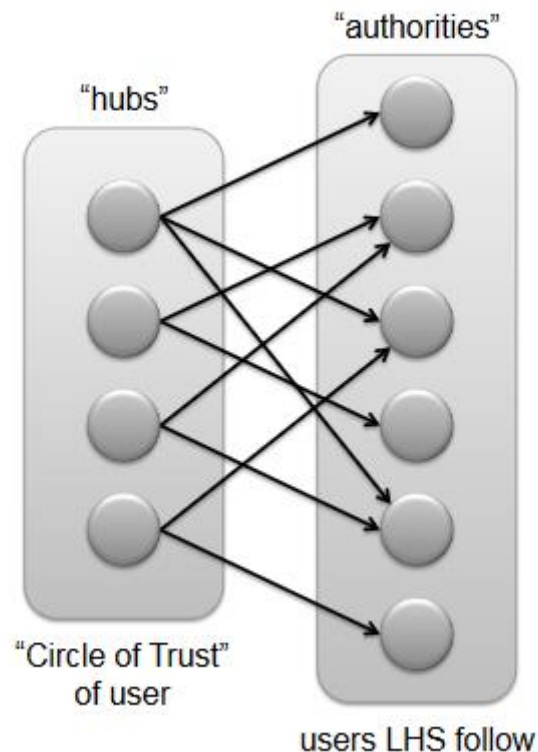


Question: pageRank provides global ranking of tweets.
Can I get personalized rankings for e.g. User A?

Circle-of-Trust (Twitter)

source: “**WTF**: The Who to Follow Service at Twitter”

- For each user compute her **circle-of-trust** (“hubs”) containing ~500 of the top-ranked nodes using **personalized pageRank**
- Authorities are users that the hubs follow
- Perform random walks (SALSA*), use top-pics from RHS



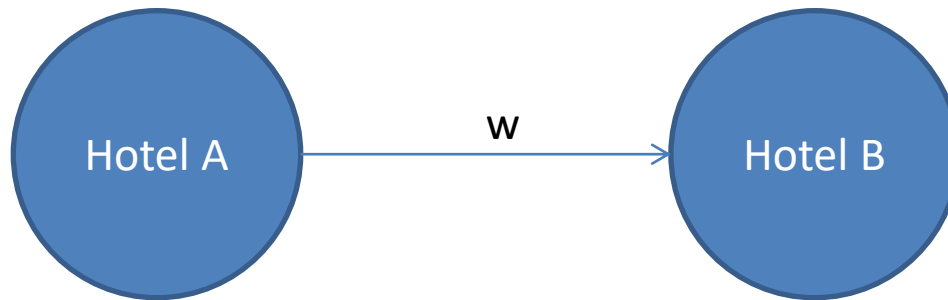
*SALSA=Stochastic Approach for Link-Structure Analysis

Hotel Recommendations

example adapted from Antonis Dimakis slides http://users.ece.utexas.edu/~dimakis/GraphDay_wNotes.pdf

- Ask hotel site to find hotels within 10km from Athens center on specific dates
- System pulls 1000 hotels from database with available rooms, must present top-10 to the user
- Idea: form a graph at runtime to rank hotels

Hotel Preference Graph



Create a link if a past user ranked/rated hotel B higher than hotel A

Weight w may denote the difference between the two ratings, possibly adjusted by the trustworthiness of a user

Recall: edges were used to cast “votes” in our initial consideration of pageRank

HITS:

HYPERLINK-INDUCED TOPIC SEARCH

HITS: Hyperlink-Induced Topic Search

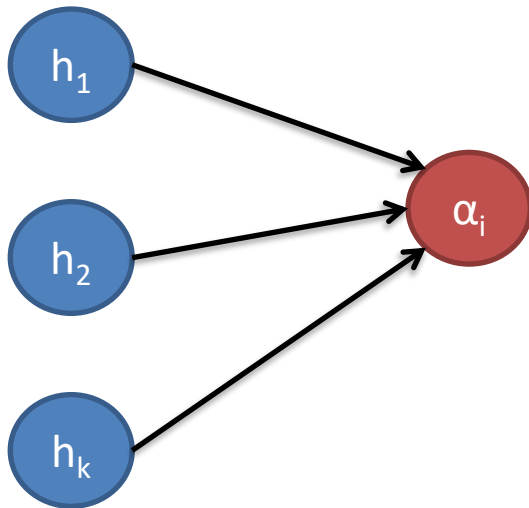
- Developed by Jon Kleinberg (Cornell) at about the same time as PageRank
 - Used in Ask.com search engine
- Web search: looking for pages that are *authoritative* for a given query
 - E.g. query = “automotive makers”
- **Authorities**: *Ferrari.com, bmw.com, honda.com, hyundai.com, etc.*

Hubs

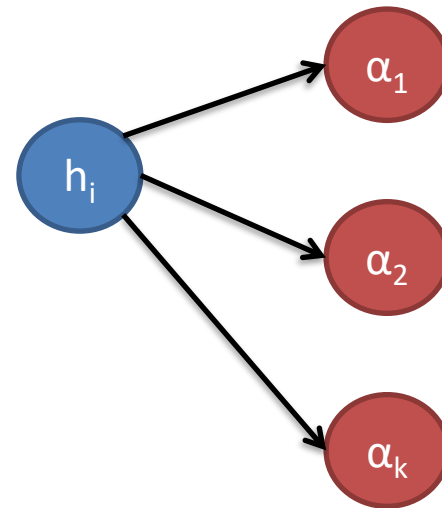
- Hubs are pages that helps you find relevant information by providing links towards the authoritative pages
 - E.g. cars.com, edmunds.com, 4troixoi.gr point to car manufacturers' pages
- A good hub points to valuable authoritative pages
- A good authoritative page is pointed to by valuable hubs

Formulation

- For each page i maintain
 - Authority weight α_i
 - Hub weight h_i

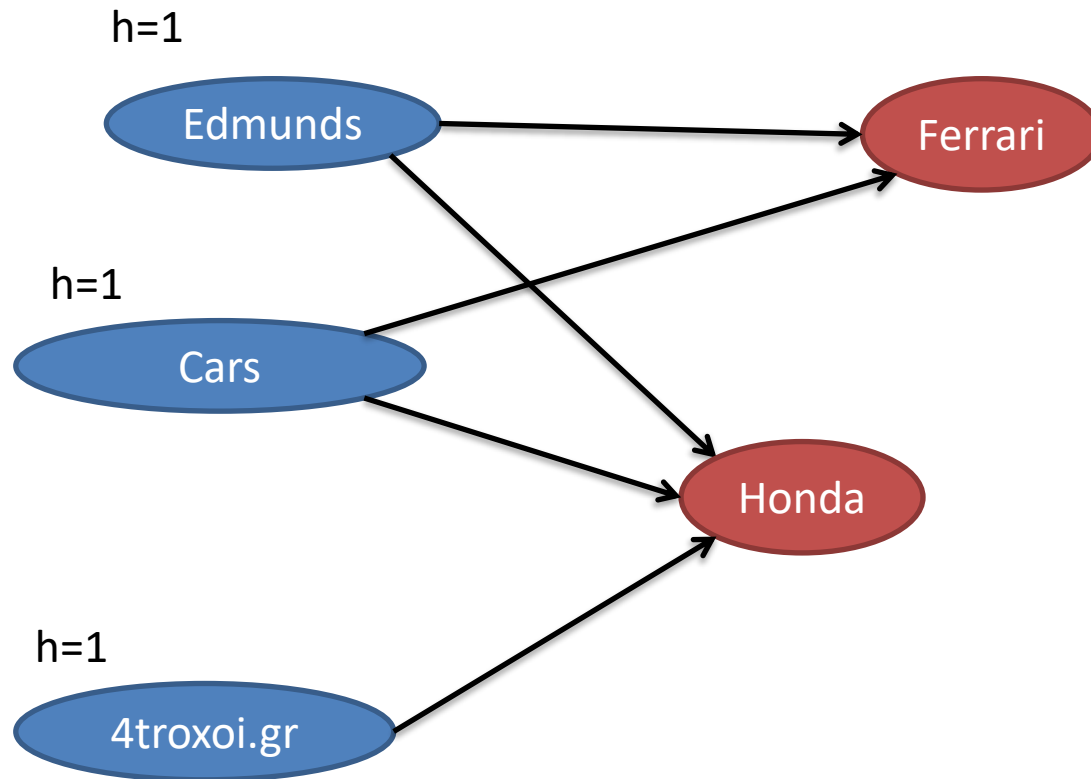


$\alpha_i = \text{sum of } h_j \text{ for all pages } j \text{ pointing to page } i$



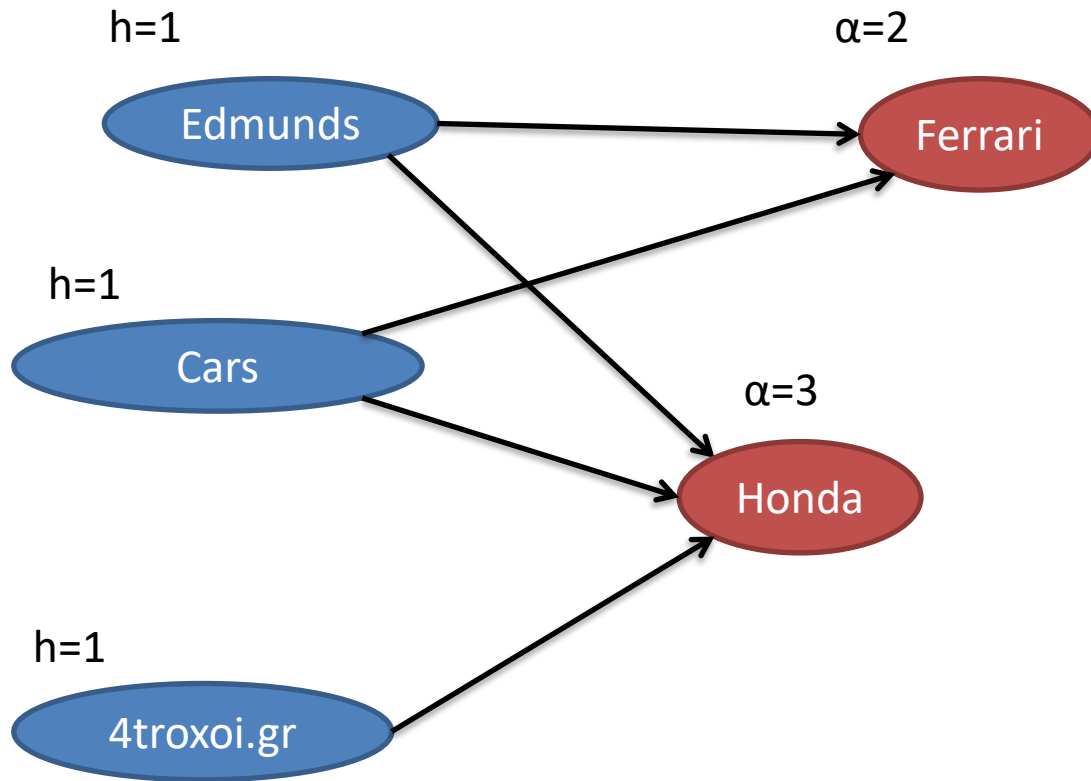
$h_i = \text{sum of } \alpha_j \text{ for all pages } j \text{ pointed to by page } i$

Simple Example*

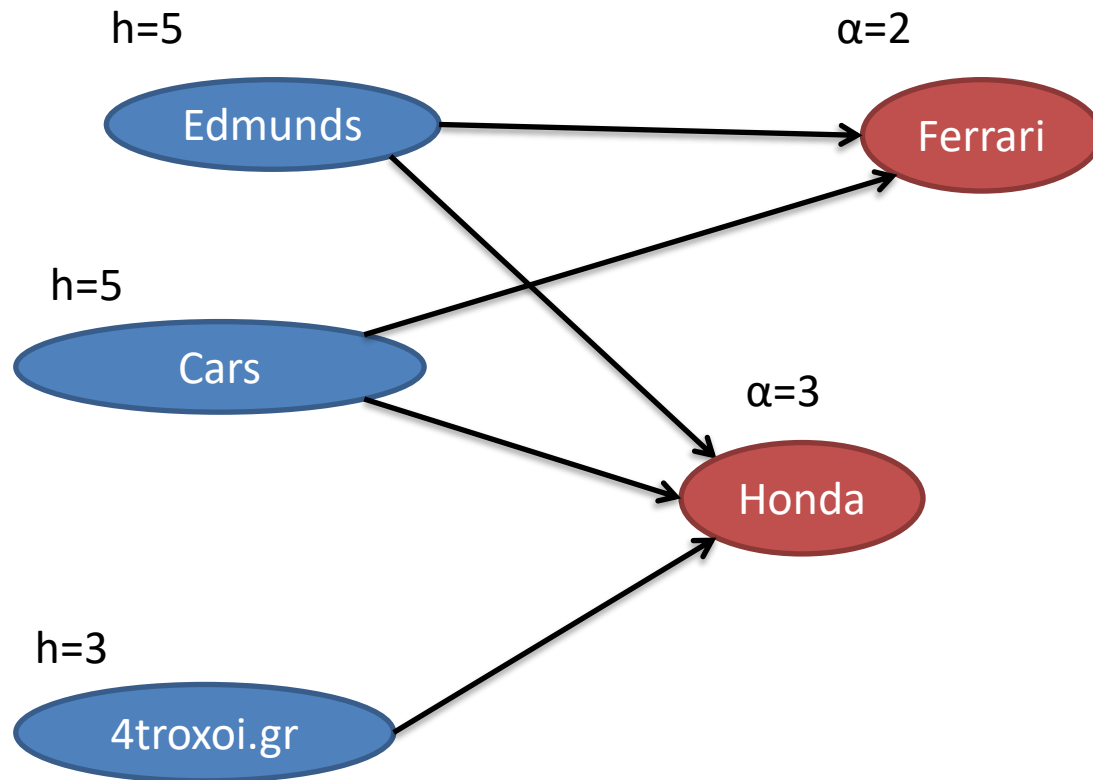


*In practice most pages are both hubs and authorities
In the example, we do not normalize weights

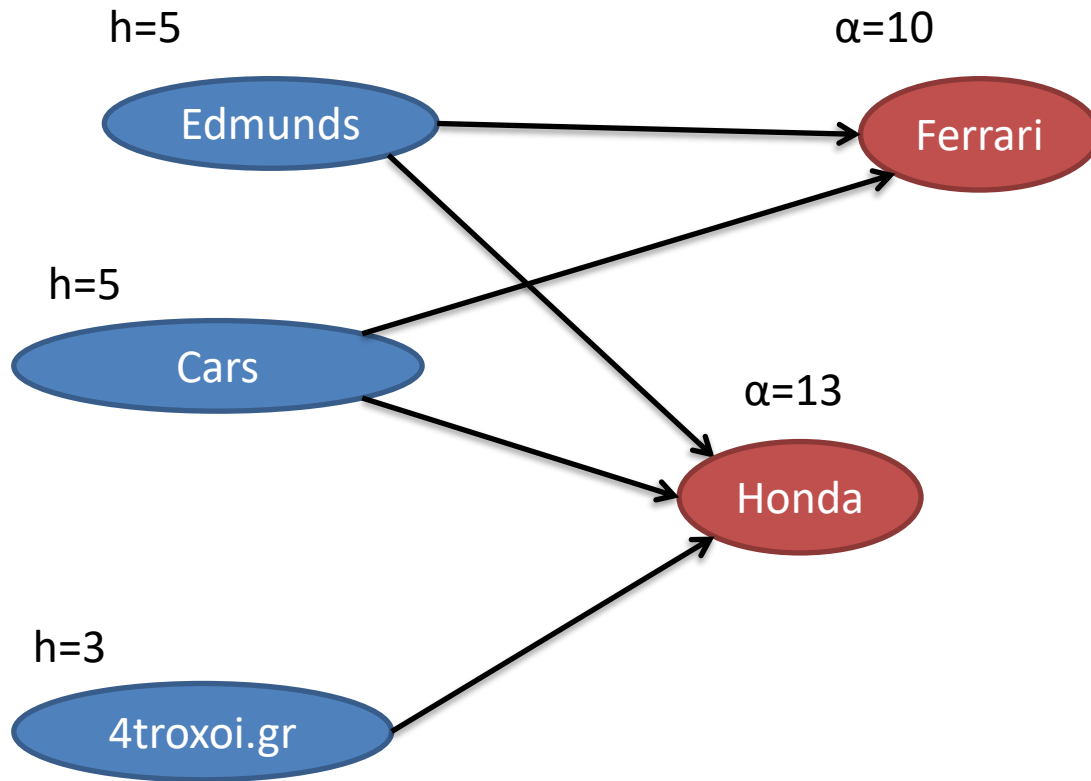
Update α



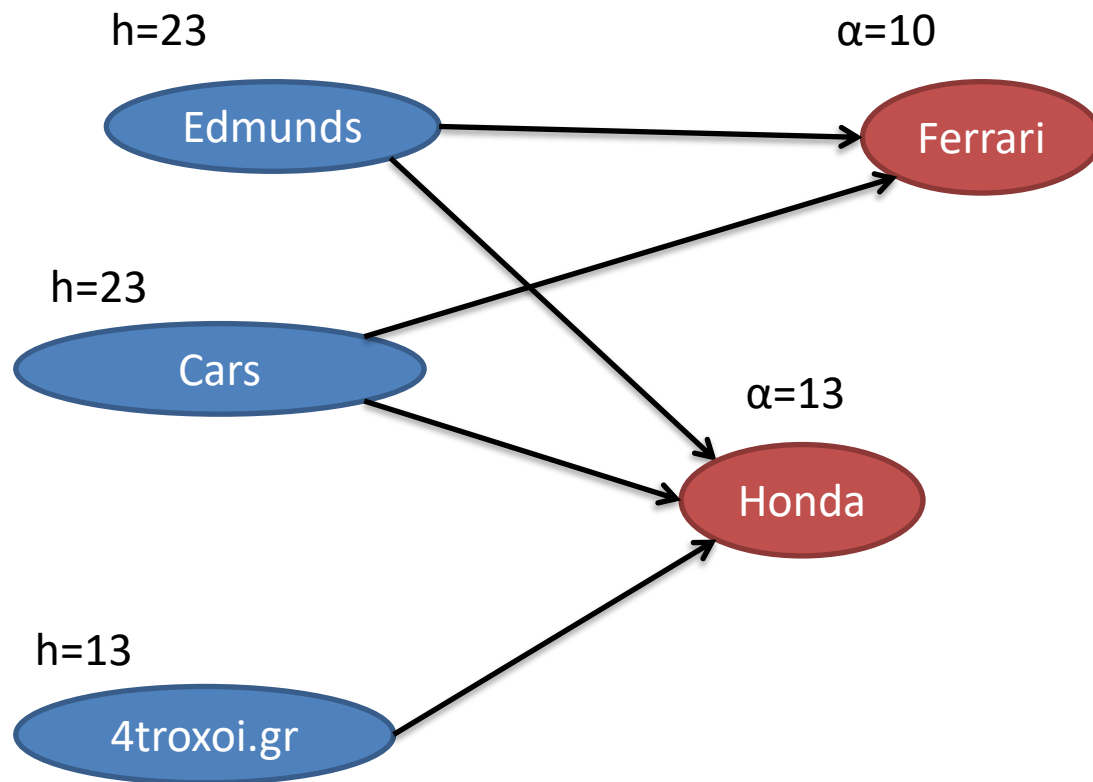
Update h



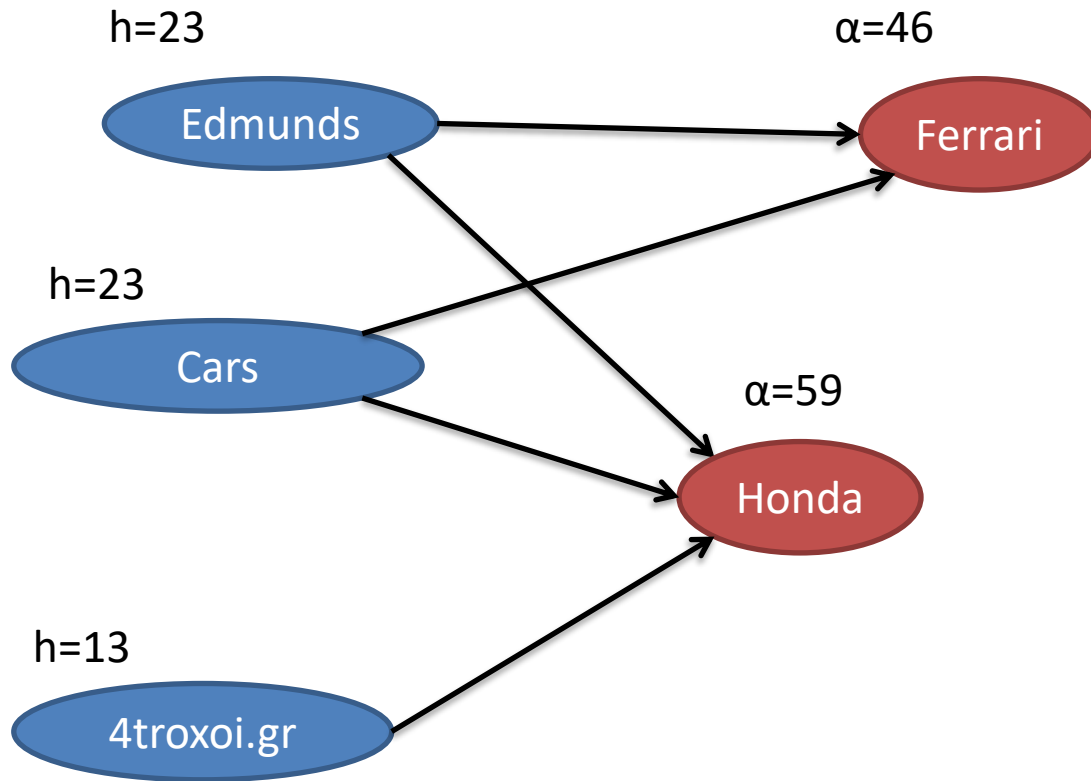
Update α



Update h



Update α



HITS computation

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

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

- Let vector $a = (a_1, \dots, a_n)$
vector $h = (h_1, \dots, h_n)$
 $n \times n$ matrix $A: A_{ij} = 1$ if $i \rightarrow j$
- Then $\alpha = A^T * h$, $h = A * \alpha$

Convergence

- $\alpha = A^T * h, h = A * \alpha$
- Thus, $\alpha = A^T * (A * \alpha) = (A^T * A) * \alpha$
 - $(A^T * A) * \alpha = 1 * \alpha$
 - α is the principal eigenvector of $(A^T * A)$
- Similarly, $h = (A * A^T) * h$
 - h is the principal eigenvector of $(A * A^T)$

PageRank vs HITS

- Both use links to compute importance of a web page
- In PageRank the value of an incoming link $u \rightarrow v$ to page v depends on the links $k \rightarrow u$ into page u
- In HITS, the value of the in-link $u \rightarrow v$ depends on the value of other links $u \rightarrow k$ out of u

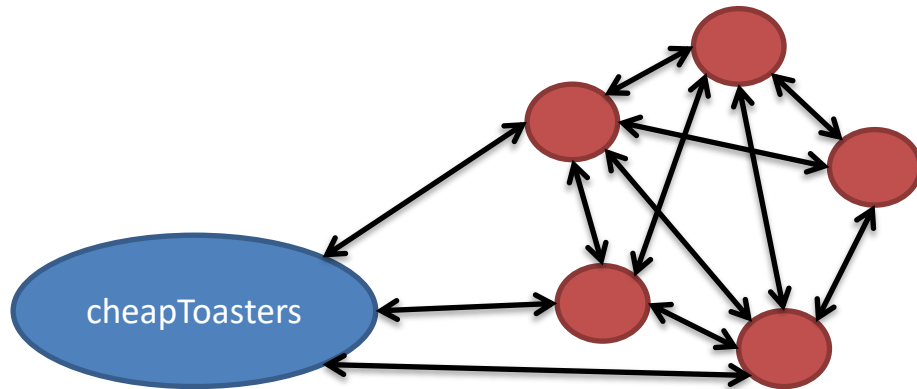
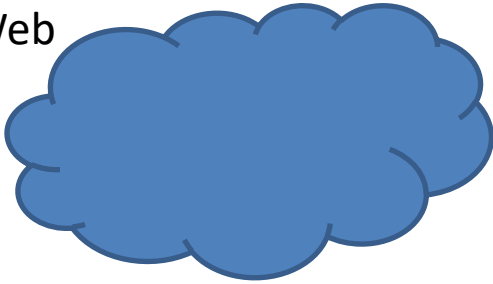
FIGHTING SPAM

Spam

- Recall that the idea behind link analysis in e.g. PageRank, HITS is to look at who is talking about your page rather than what you claim about it.
- Can these algorithms be fooled?
 - yes !

Boost my cheapToasters.com page!

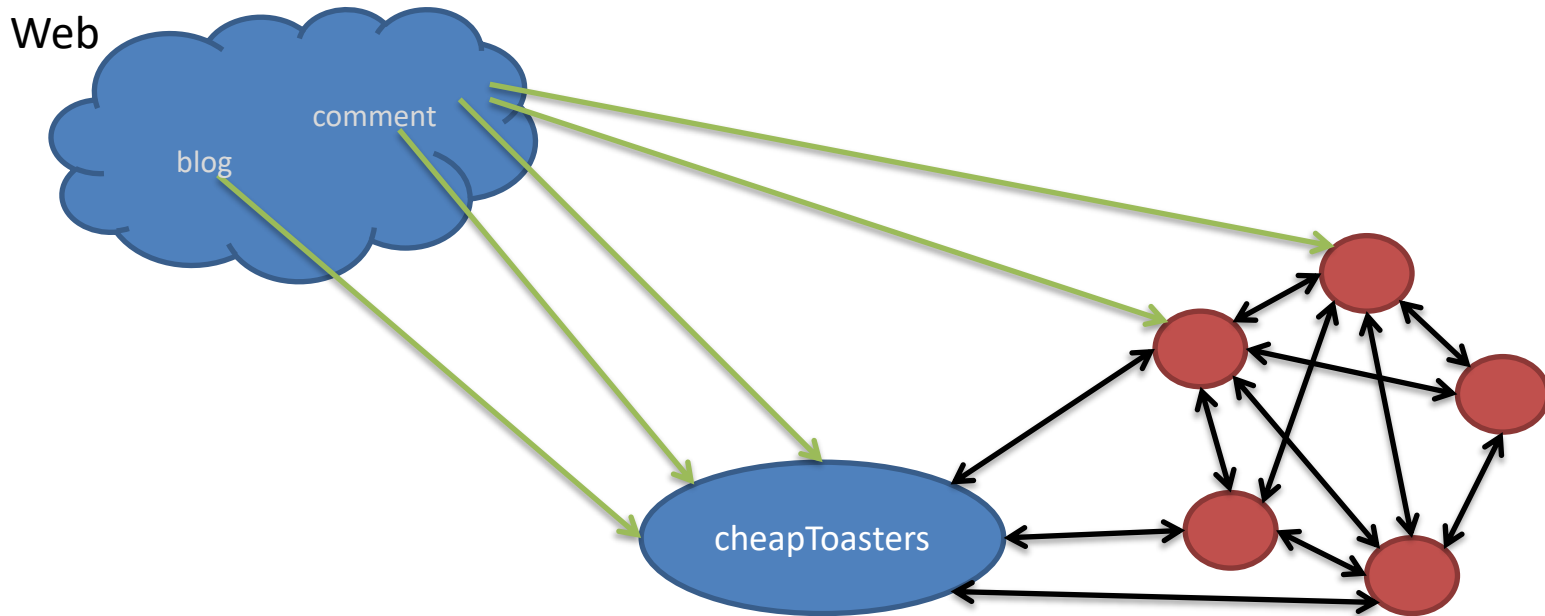
Web



Create "link farm" of e.g. 100K pages pointing to cheapToasters.com

WON'T WORK!

Boost my cheapToasters.com page!



Add links from accessible pages to cheapToasters.com and the farm

Now PageRank leaks into cheapToaster.com and the farm

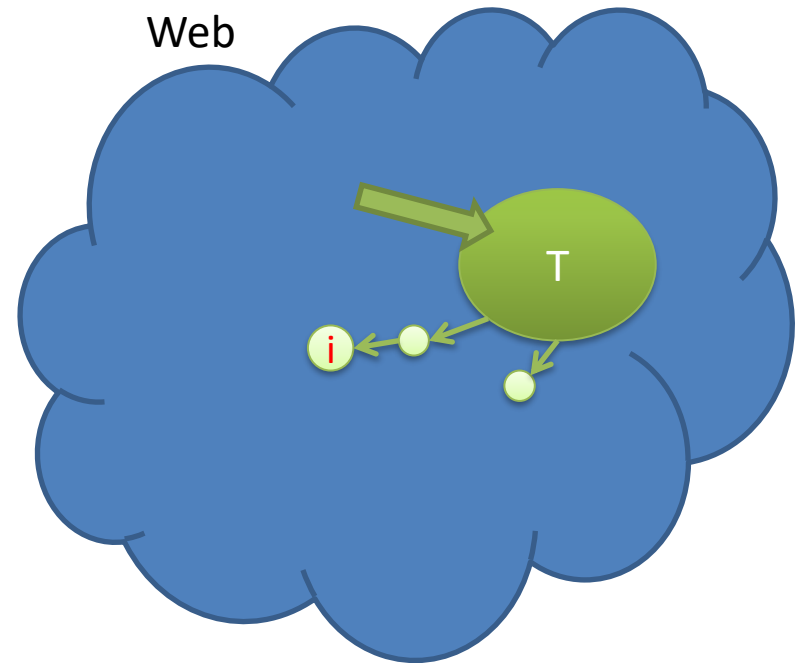
However, search engines are getting increasingly better at detecting these farms

Fighting spam

- Similar idea to topic-specific PageRank
- Assemble a list **T** to trusted pages from the web
 - Use trusted domains such as .edu, .gov, etc whose content is curated, OR
 - Get pages with highest PageRank values. These are hard to compromise. Have humans examine them and decide which of them are trustworthy.

TrustRank

- Compute PageRank using teleports towards those pages in T only
 - TrustRank = computed PageRank
- If a page is linked from or is in a short distance from pages in T, it gets a high value of TrustRank



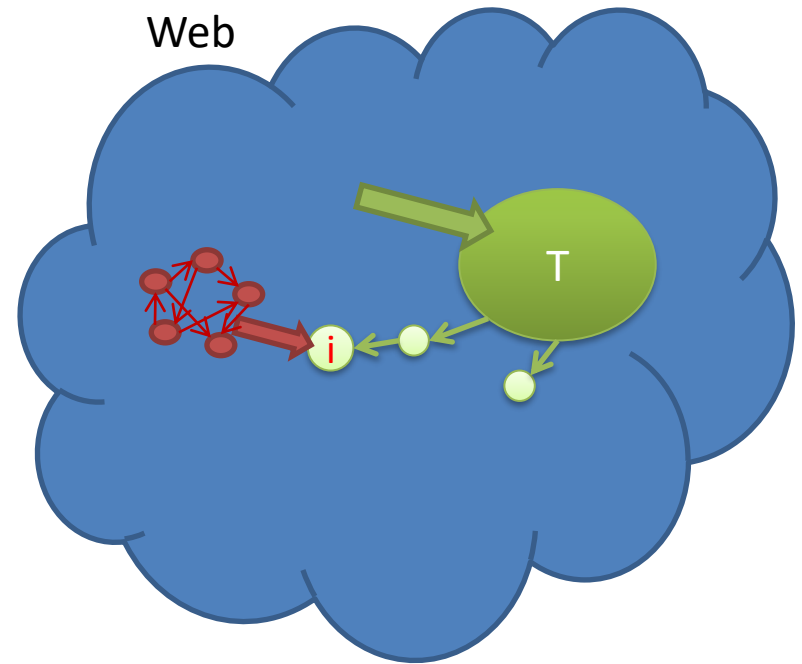
t_i = TrustRank of page i

p_i = PageRank of page i

$$\text{spam} - \text{mass}_i = \frac{p_i - t_i}{p_i}$$

Spam-mass

- Portion of PageRank that comes from spam
 - Small or negative values mean that page is most likely not spam
 - A value close to 1 means that the page is probably spam

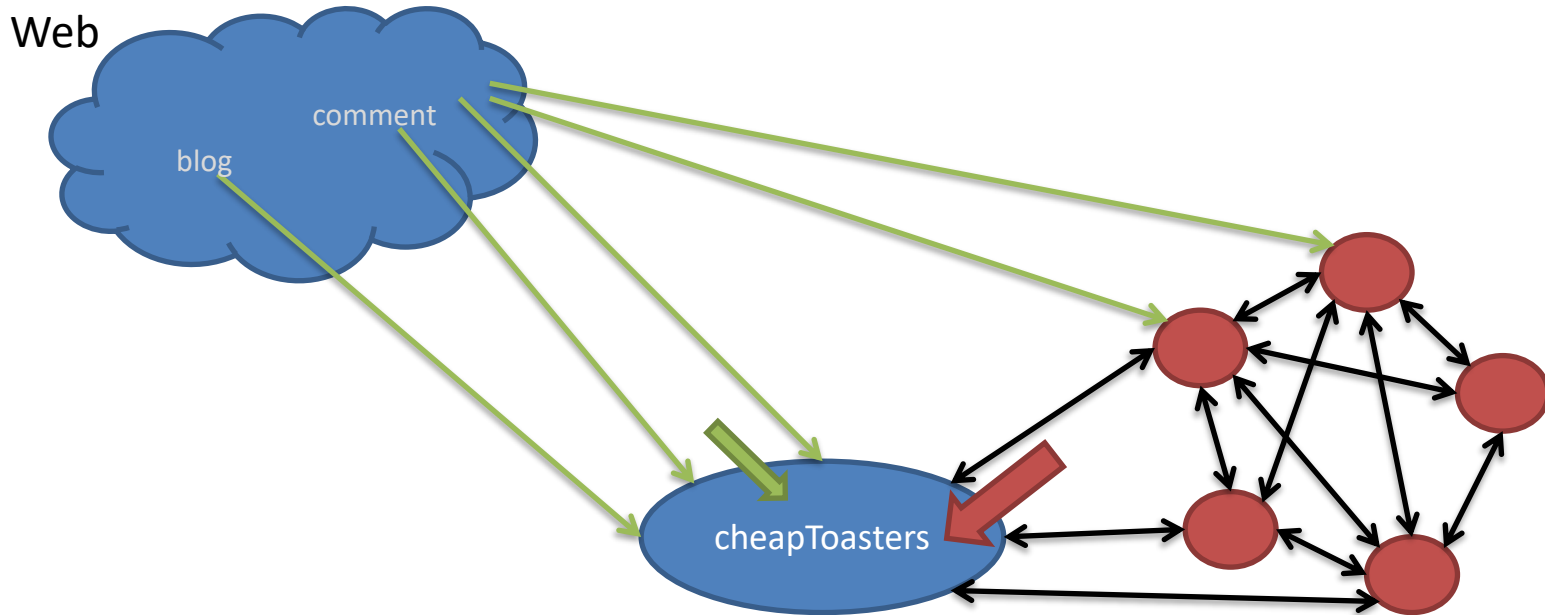


t_i = TrustRank of page i

p_i = PageRank of page i

$$spam-mass_i = \frac{p_i - t_i}{p_i}$$

Spam-mass of cheapToasters.com



If no trustworthy page links to my website (directly or indirectly), then its spam-mass will be close to 1

$$\text{spam-mass}_i = \frac{p_i - t_i}{p_i}$$