ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Link Analysis

Yannis Kotidis

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

Acknowledgments

- Material adapted from the textbook "*Mining Massive Datasets*" (http://infolab.stanford.edu/~ullman/mmds.html)

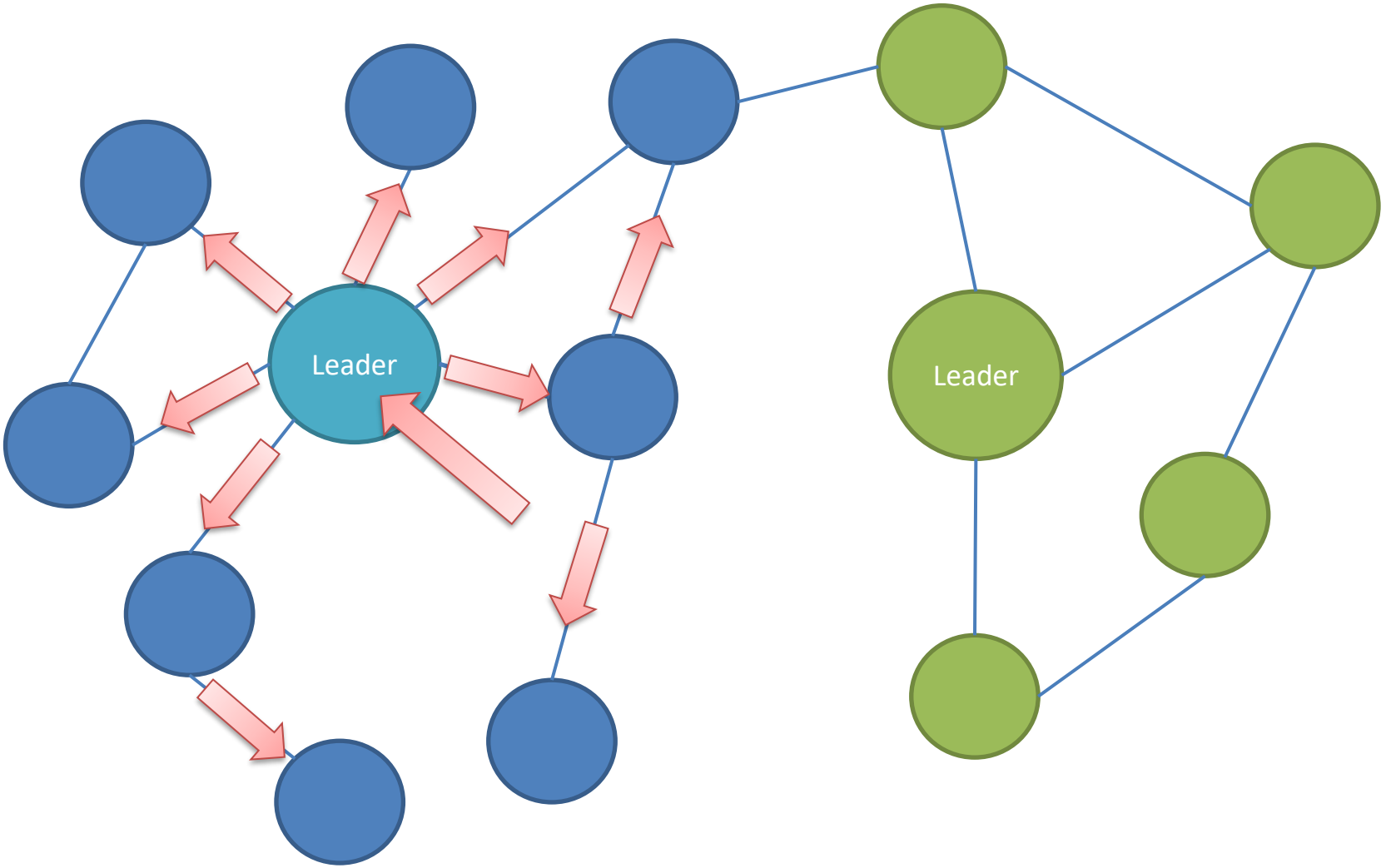
Roadmap

- 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"

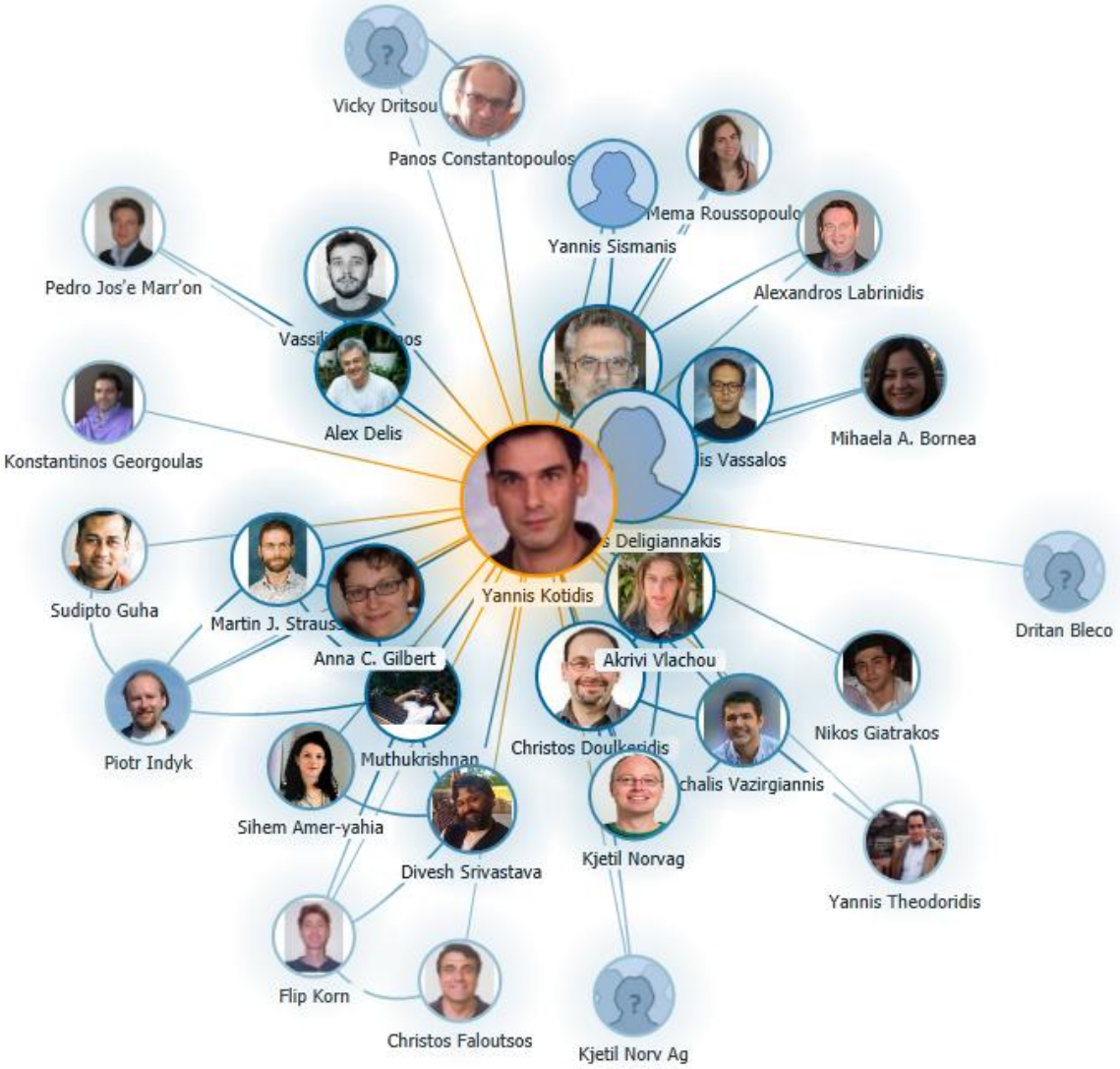
Social Graphs

http://www.socialmediaexaminer.com/what-your-business-needs-to-know-about-social-graphs/

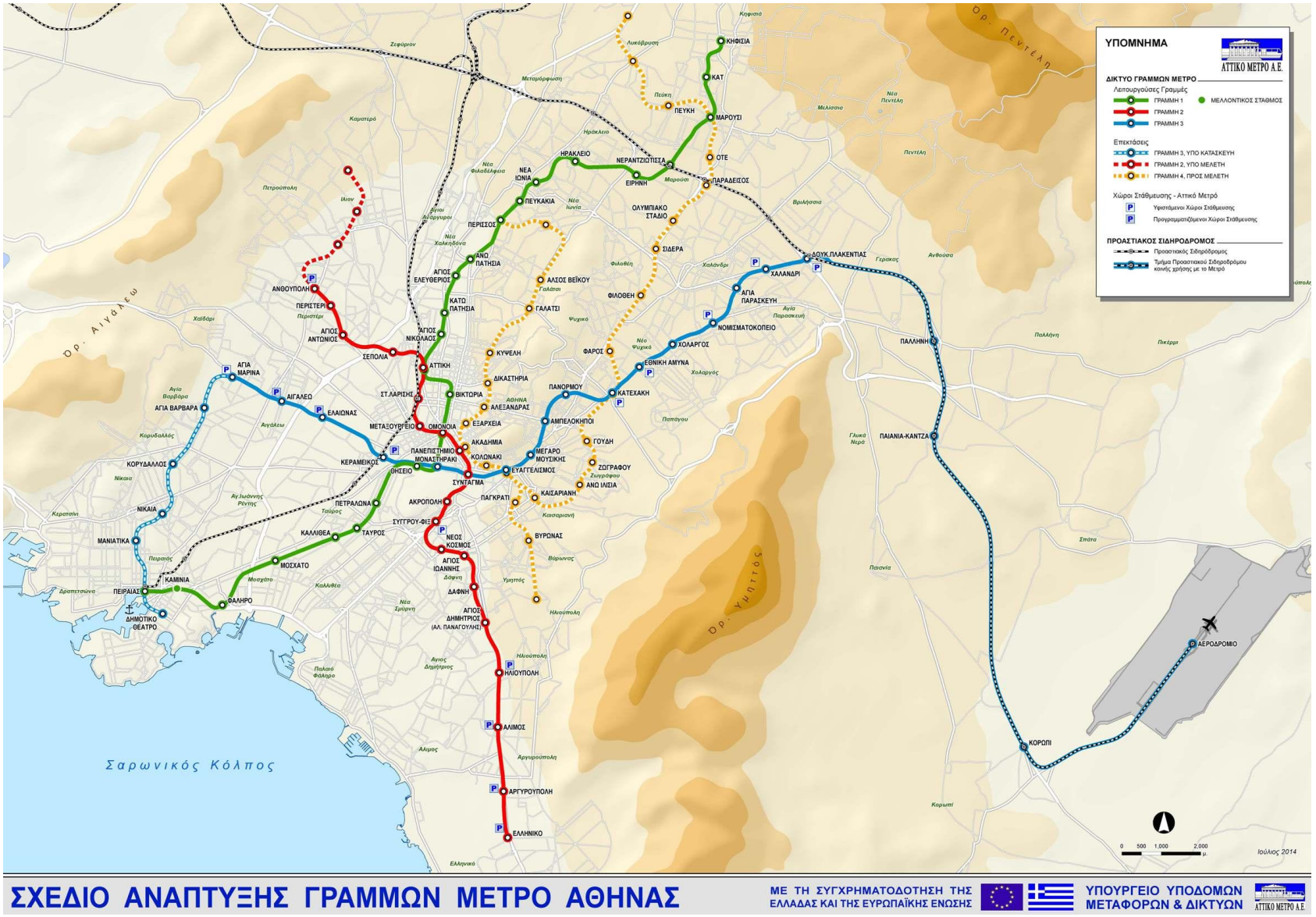
SNA: Marketing campaign



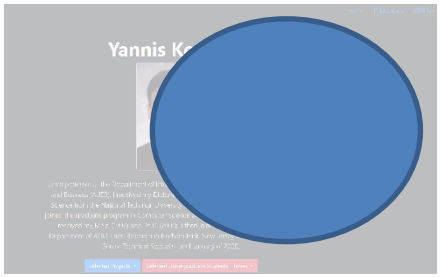
Citation Networks

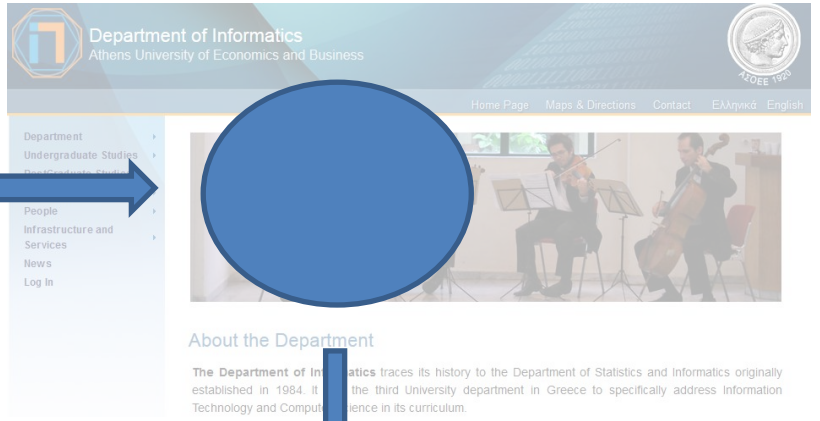


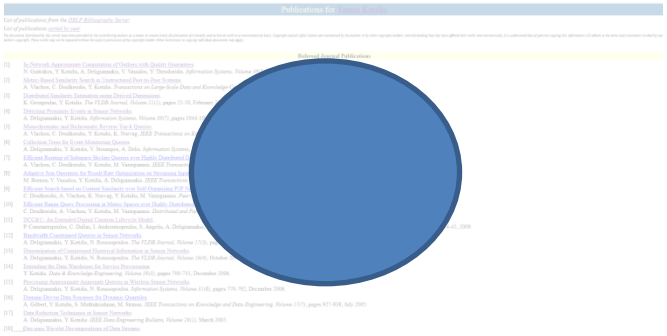
Transport Networks

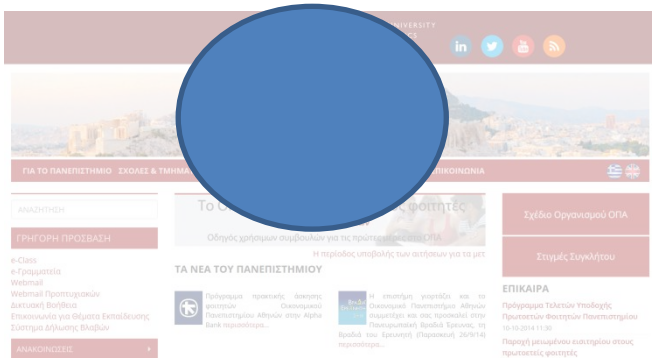


The WWW




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





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 searches for movies
- 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 (Google's idea)

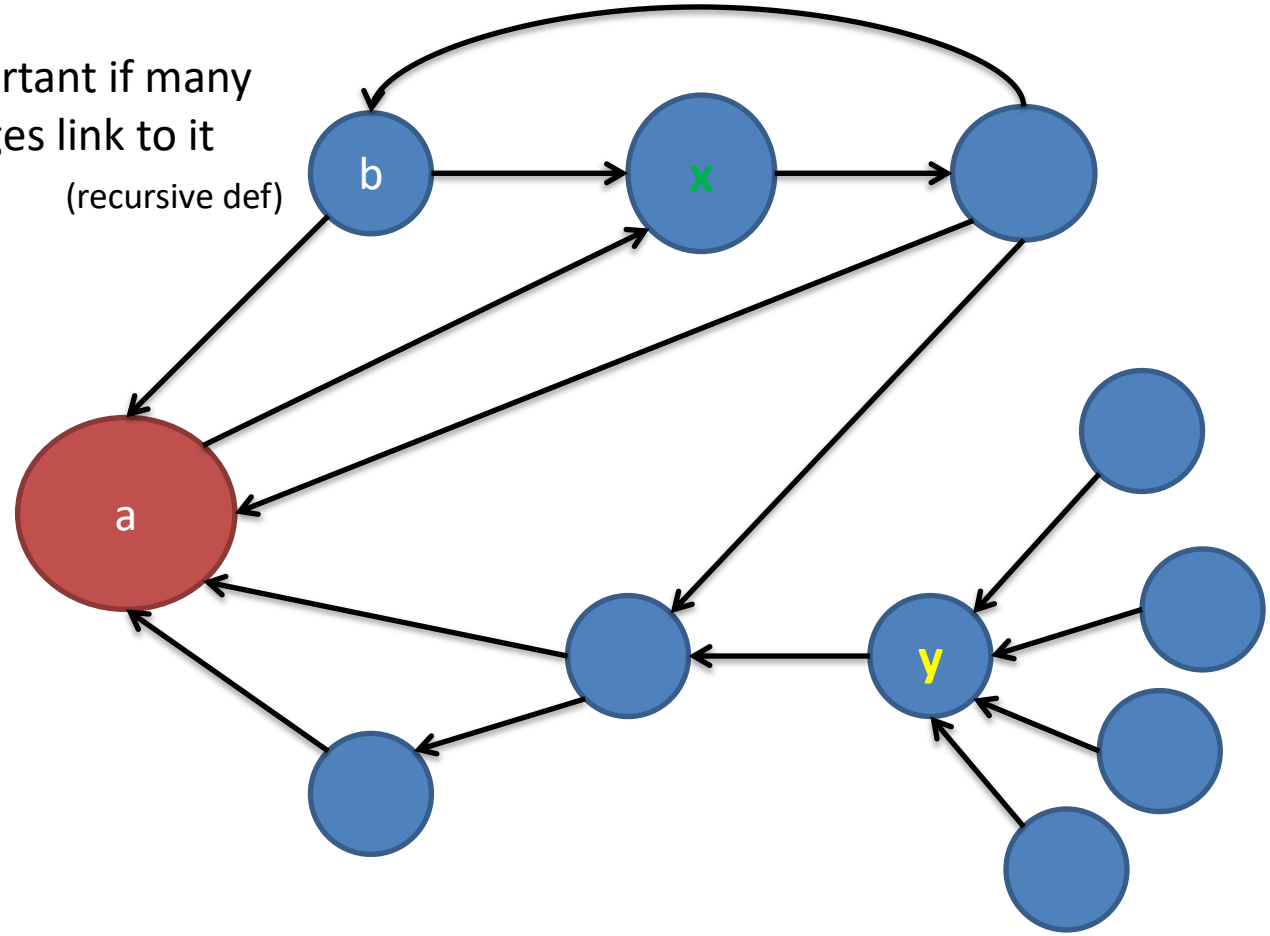
- Believe what people say about you, rather than what you say about yourself
  - Use information from the web link structure in order to measure the importance of web pages, when ranking the results of a query

- Assumption: trustworthy/authoritative sources may link to each other

PageRank* (informal)

Page x is important if many important pages link to it

(recursive def)

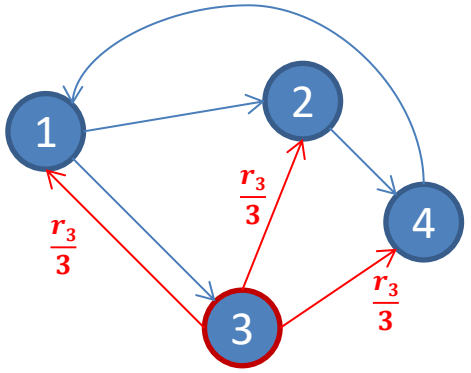


Page y has more incoming links but lower PageRank than page x

*Larry Page and Sergey Brin

How to measure importance

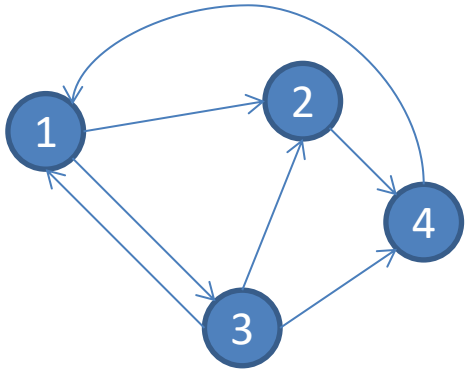
- An incoming link acts as a positive testimony (a "vote")
  - Votes from important pages count more
- Weight of a vote is equally split between all outgoing edges
  - Fig: node 3 has three outgoing edges
- Votes received by node 1:
  - $r_1 = r_3/3 + r_4$



$r_1 = r_3/3 + r_4$
$r_2 = r_1/2 + r_3/3$
$r_3 = r_1/2$
$r_4 = r_2 + r_3/3$

Solution

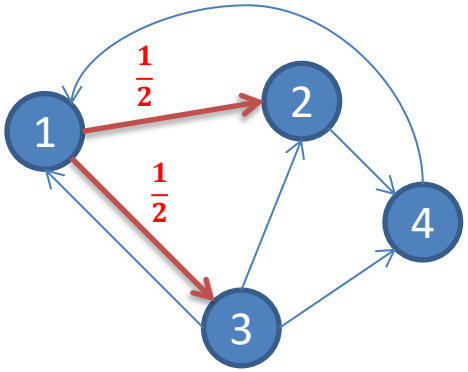
- 4 equations/4 unknowns/no constants
  - No unique solution
- Add another constraint:
  - All ranks sum up to 1
- Solution:
  - $r_1 = 6/18$, $r_2 = 4/18$
  - $r_3 = 3/18$, $r_4 = 5/18$



$r_1 = r_3/3 + r_4$
$r_2 = r_1/2 + r_3/3$
$r_3 = r_1/2$
$r_4 = r_2 + r_3/3$
$r_1 + r_2 + r_3 + r_4 = 1$

Matrix representation of the WWW

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



From:

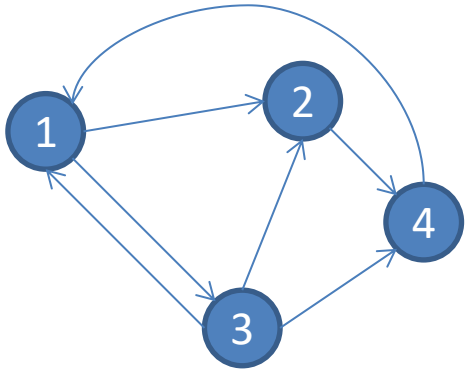
To:

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

columns summing to 1

Store ranks (that we seek to compute) in a vector **r**

- $r=(r_1,r_2,r_3,r_4)^T$

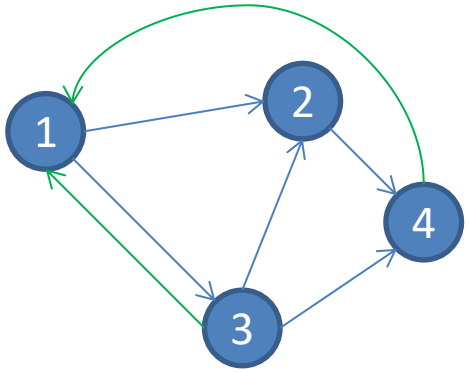


- Constraint: $r_1+r_2+r_3+r_4=1$

$$r = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix}$$

Equivalent matrix formulation

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



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

$\cdot$

r_1
r_2
r_3
r_4

$=$

r_1
r_2
r_3
r_4

$\Leftrightarrow$

$r_1 = r_3/3 + r_4$

$r_2 = r_1/2 + r_3/3$

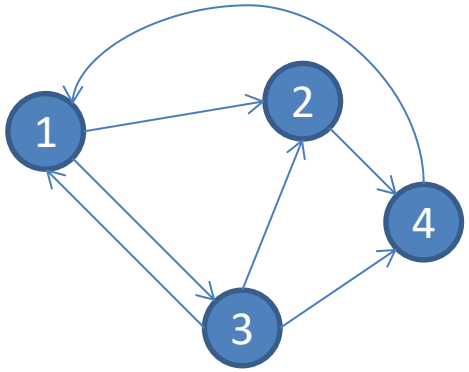
$r_3 = r_1/2$

$r_4 = r_2 + r_3/3$

$r_1 + r_2 + r_3 + r_4 = 1$

Page Ranks as eigenvectors

- A*x = λ*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 λ=1)



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

·

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

=

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

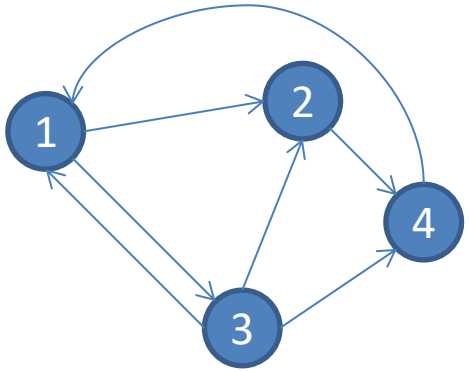
Power Iteration Method

- 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. $|r(i)-r(i-1)| < \varepsilon$, 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	

$\cdot$

¼
¼
¼
¼

$=$

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

$r_1 = r_3/3 + r_4$

$r_2 = r_1/2 + r_3/3$

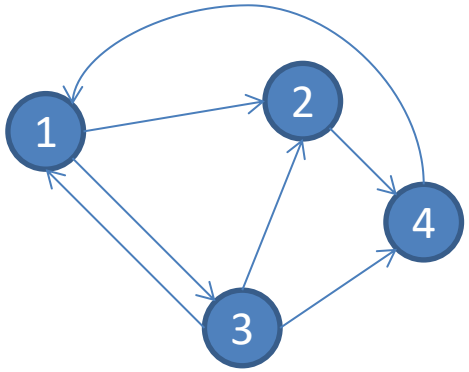
$r_3 = r_1/2$

$r_4 = r_2 + r_3/3$

$r_1 + r_2 + r_3 + r_4 = 1$

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	

$\cdot$

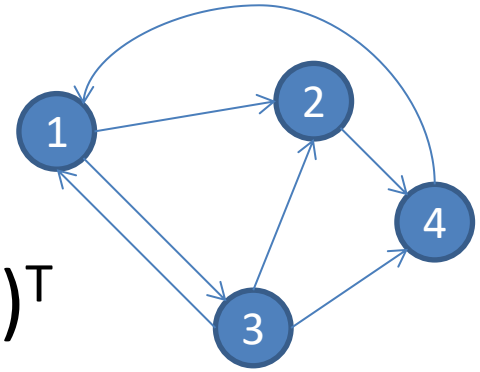
1/3
5/24
1/8
1/3

$=$

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	

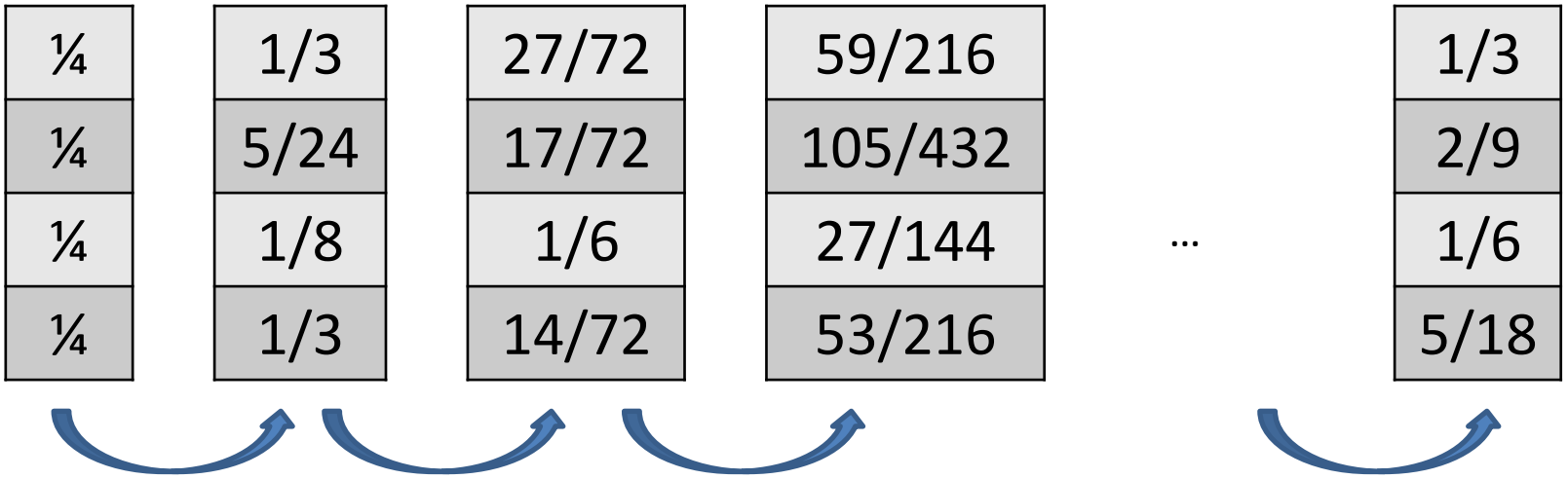
$\cdot$

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

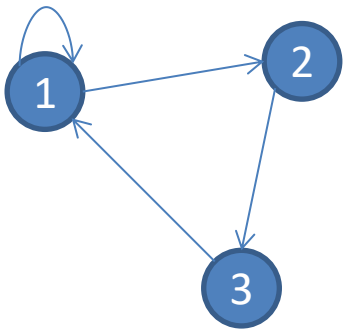
$=$

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

Convergence



Example 2



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

- Can you guess the resulting Page Ranks?

1/2		1
1/2		
	1	

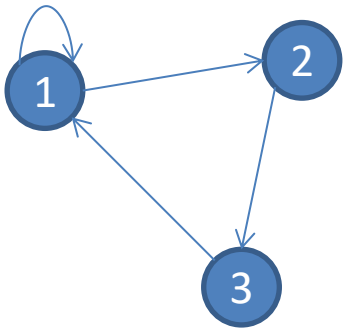
$\cdot$

1/3
1/3
1/3

$=$

1/2
1/6
1/3

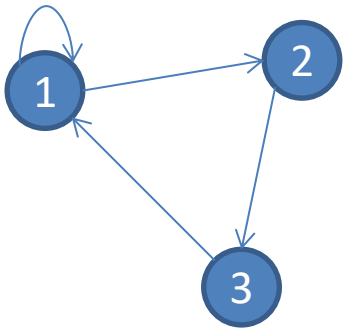
Example 2



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

1/2		1		1/2		7/12
1/2			·	1/6	=	1/4
	1			1/3		1/6

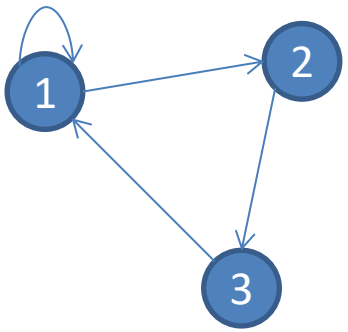
Example 2



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

1/2		1		7/12		11/24
1/2			·	1/4	=	7/24
	1			1/6		¼

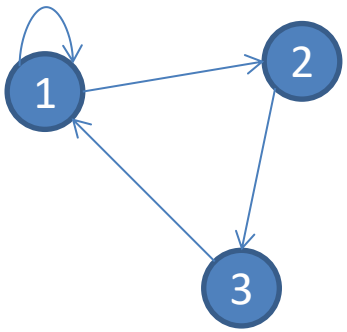
Example 2



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

1/2		1		11/24		23/48
1/2			·	7/24	=	11/48
	1			1/4		7/24

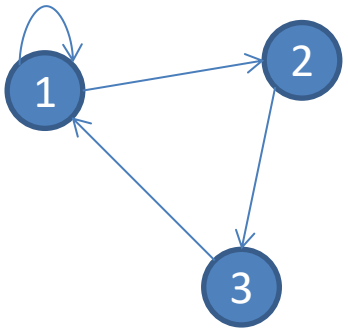
Example 2



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

1/2		1		23/48		51/96
1/2			·	11/48	=	23/96
	1			7/24		11/48

Example 2



- Next iteration

1/2		1
1/2		
	1	

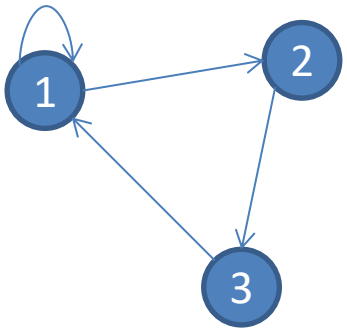
.

51/96
23/96
11/48

=

95/192
51/192
23/96

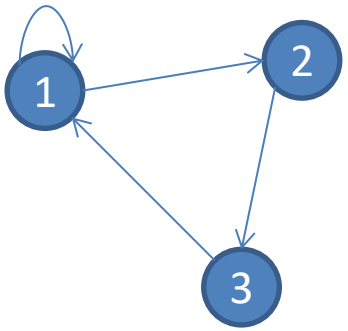
Example 2



- Next iteration

1/2		1		95/192		1/2
1/2			.	51/192	=	95/384
	1			23/96		51/192

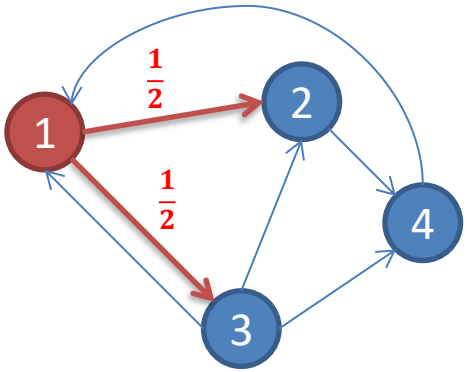
Example 2



- In the end…

1/2		1		1/2	=	1/2
1/2			.	1/4		1/4
	1			1/4		1/4

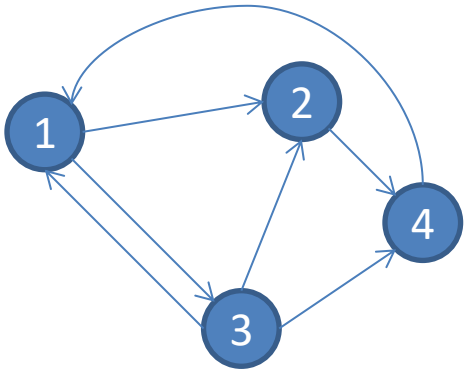
Random Web Surfer



- Surfer starts at arbitrary node
- Picks one outgoing link at random and follows it
- E.g. assume surfer is at node 1
  - with probability ½ she jumps to node 2, OR
  - with probability ½ she jumps to node 3

Matrix formulation

- Let vector $\mathbf{p^t}=(p^t_1, p^t_2, \ldots p^t_n)$
- 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?



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

Markov process/random walk

What happens in the end

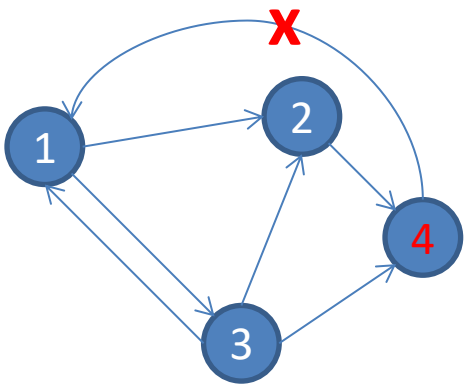
- Assume that after some (possibly long) time, the probability distribution on the location of the surfer reaches a state where $p^t = W*p^t$
- Then, p^t is **stationary distribution** of a random walk
- Recall that in the power iteration method we where looking for ranks such that r = W*r
- Thus, the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

Convergence

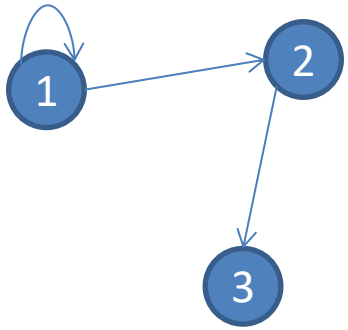
- For graphs that satisfy certain conditions, the random walk always reaches a stationary distribution, no matter the initial conditions
  - What can go wrong?

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

$=$

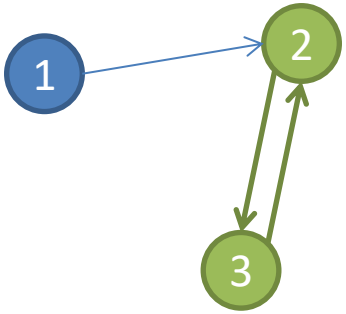
1/24
1/24
1/12

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

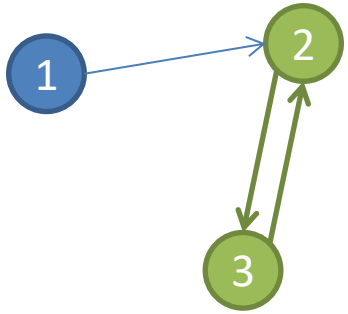
1		1
	1	

·

1/3
1/3
1/3

=

0
2/3
1/3



1		1
	1	

·

0
2/3
1/3

=

0
1/3
2/3

1		1
	1	

·

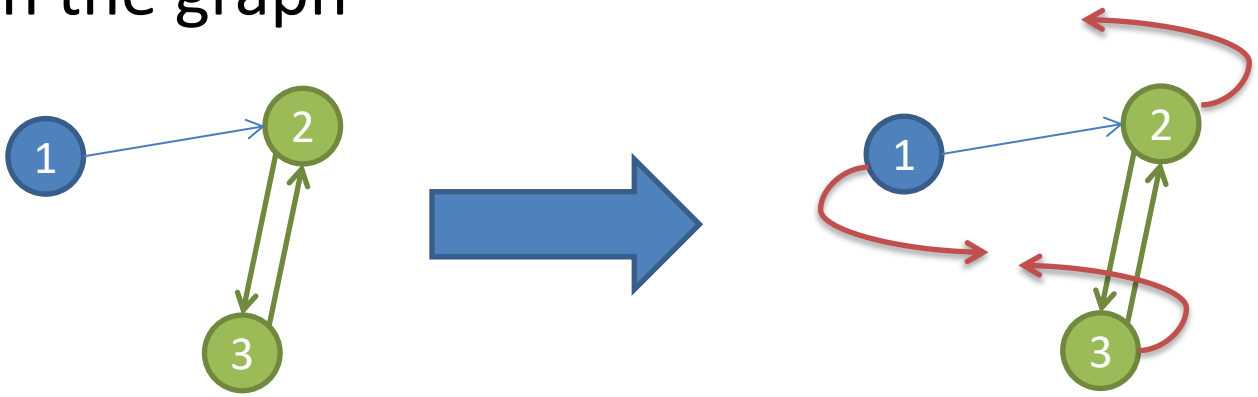
0
1/3
2/3

=

0
2/3
1/3

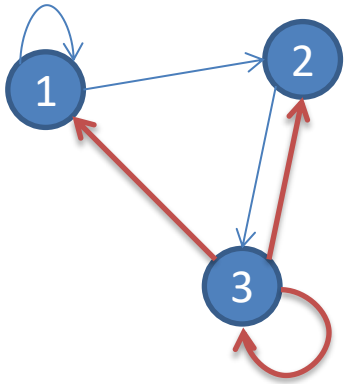
Google's solution: add teleports

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



Dead ends

- Always perform a random teleport from dead-ends



Google's Page Rank formulation*

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

Remember:

- r_i : page rank of node i
- d_i : out-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)

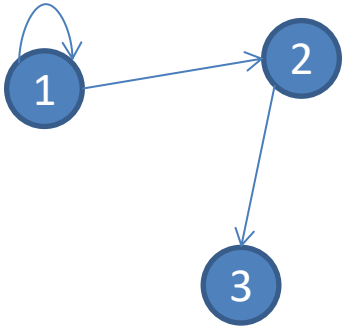
Does this converge?

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

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

Is W stochastic?

- Stochastic ⇔ columns add to 1

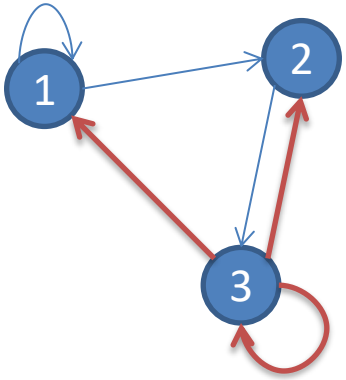


1/2		
1/2		
	1	

Sum = 0 for dead-ends

Is W stochastic?

- Stochastic ⇔ columns add to 1



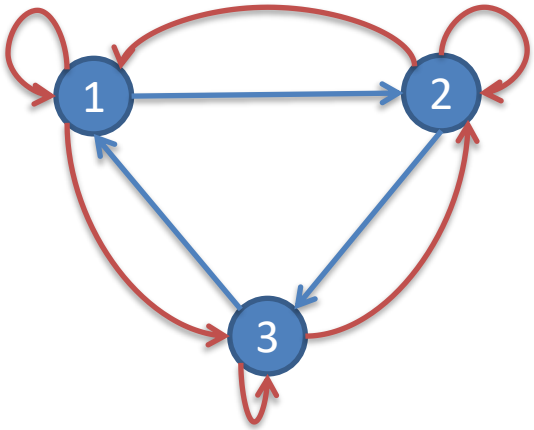
1/2		1/3
1/2		1/3
	1	1/3

Make column stochastic

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

Is W aperiodic?

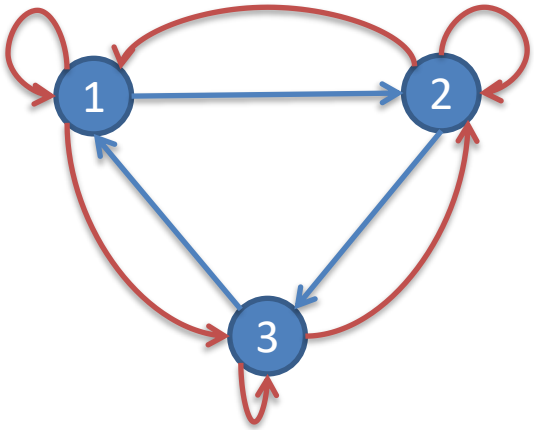
- A Markov-chain is periodic if there exists k > 1 such that the interval between two visits to some state s is always a multiple of k



Note: not all teleports required for making W aperiodic

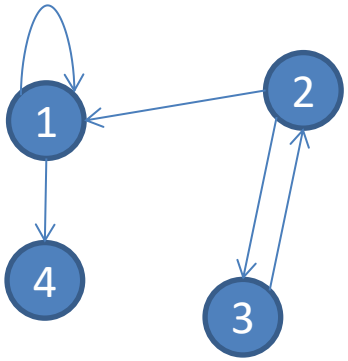
Is W irreductible?

- A Markov-chain is irreductible if there is a non-zero probability of going from node x to node y for any such pair.

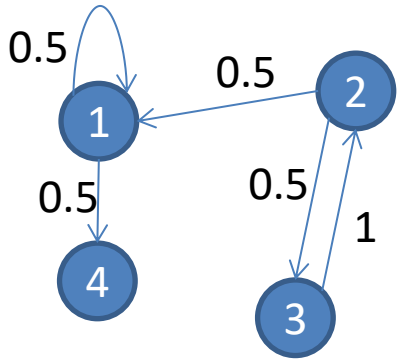


Google Matrix (no dead-ends)

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



Example

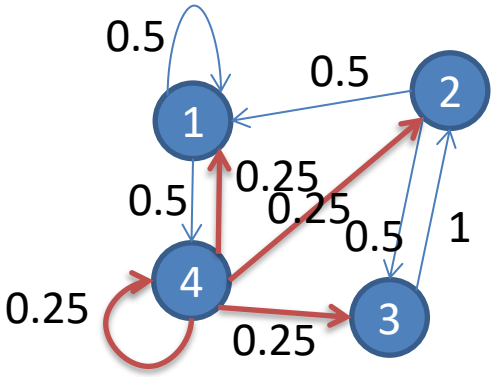


- Original matrix W:

0.5	0.5		
		1	
	0.5		
0.5			

- Node 4 is a dead-end (fix). 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 β=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 (→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,….,1/n);

t=0;

Do {

t++;

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

Let L = $\Sigma r^t(i)$; // L≤β 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 $|r^t(i)-r^{t-1}(i)|>e$;

Note

- Assume n=1 billion web nodes
- Assume ranks are stored as 4 bytes numbers
- Dense matrix would require:

  1Bx1B 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 spare: few links per node
  - Store W row-wise (=for each node, keep list of incoming edges)
    - Inverted web-graph computed easily via map-reduce (next slides)
  - We need 14 * 1 billion ids =~ 56GB
  - r(t), r(t-1) vectors need ~4GB each
  - Thus, computation requires a server with 64GB available memory

Reverse web-link graph

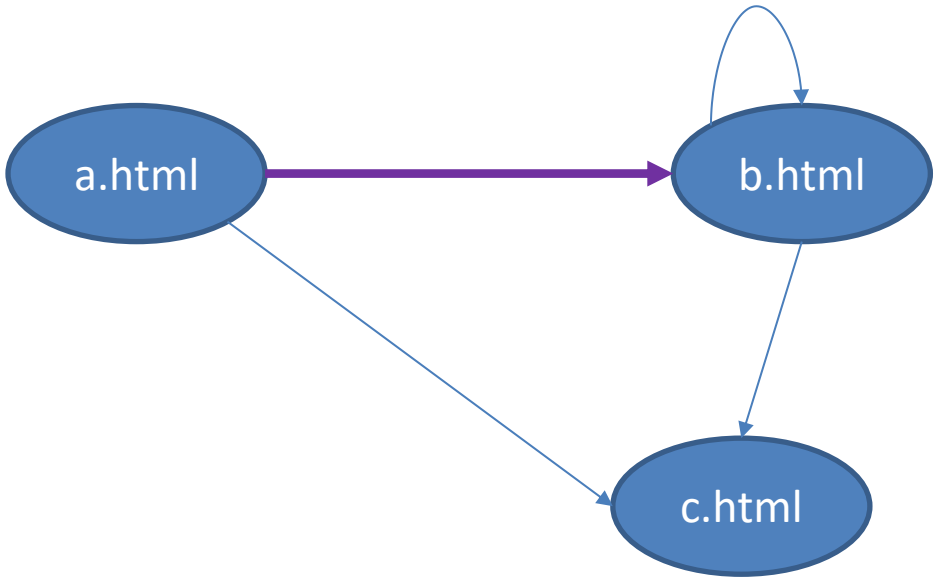
(find where page links come from)

a.html

```
<html>
 <a href="b.html">....</a>
 ...
 <a href="c.html">....</a>
```

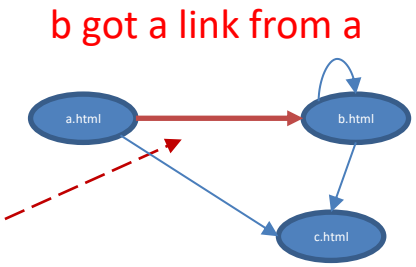
b.html

```
<html>
 <a href="b.html">....</a>
 ...
 <a href="c.html">....</a>
```



Reverse web-link graph

- Use href as a key, source url as value!



a.htm

```
<html>
 <a href="b.htm">....</a>
 ...
 <a href="c.html">....</a>
```

b.htm

```
<html>
 <a href="b.htm">....</a>
 ...
 <a href="c.htm">....</a>
```

Mappers

b.htm, a.htm

c.htm, a.htm

b.htm, b.htm

c.htm, b.htm

Reducers

b.htm, (a.htm,b.htm)

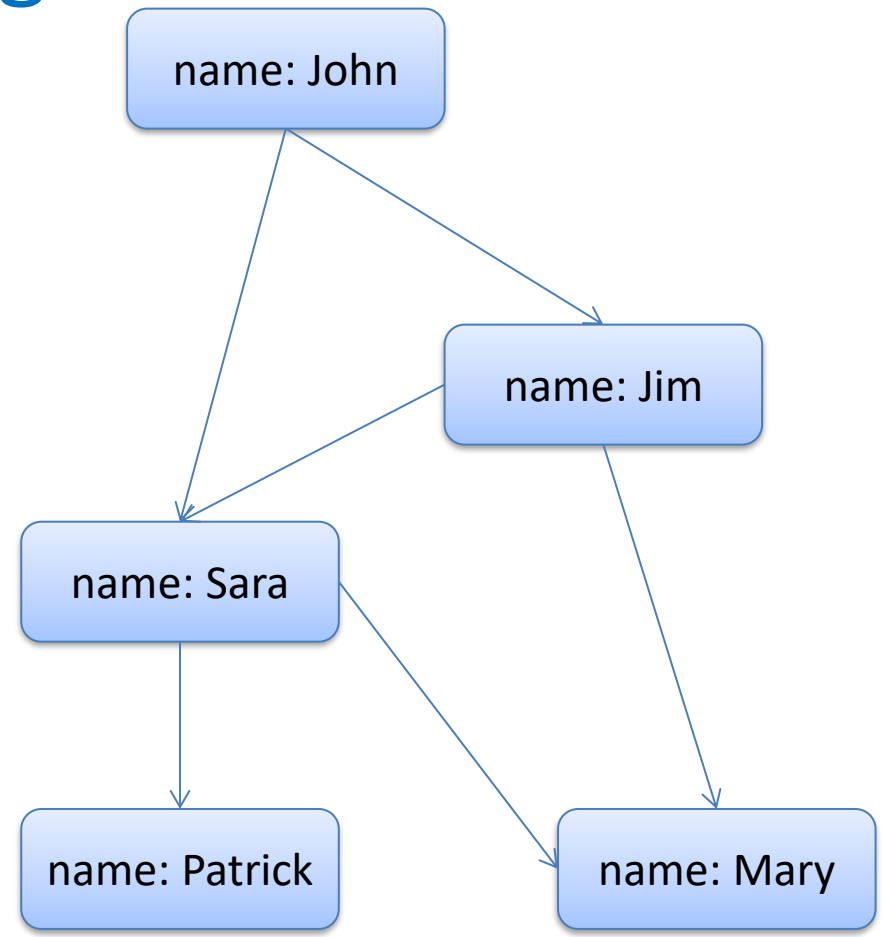
c.htm, (a.htm,b.htm)

Reverse web-link graph Recap

- Find where page links come from
  - Map: output <target, source> for each link to target in a page source
  - Reduce: concatenate the list of all source URLs associated with a target
    - Output <target, list(source)>

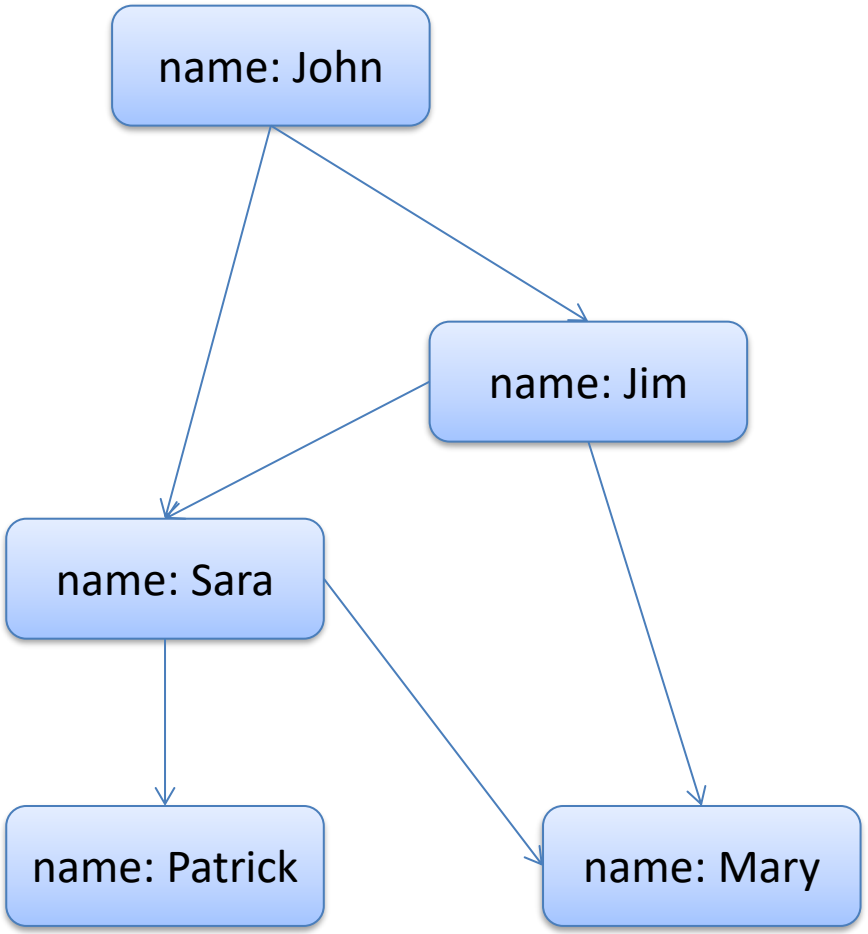
GraphX:
Define nodes using a DataFrame

```
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")
```



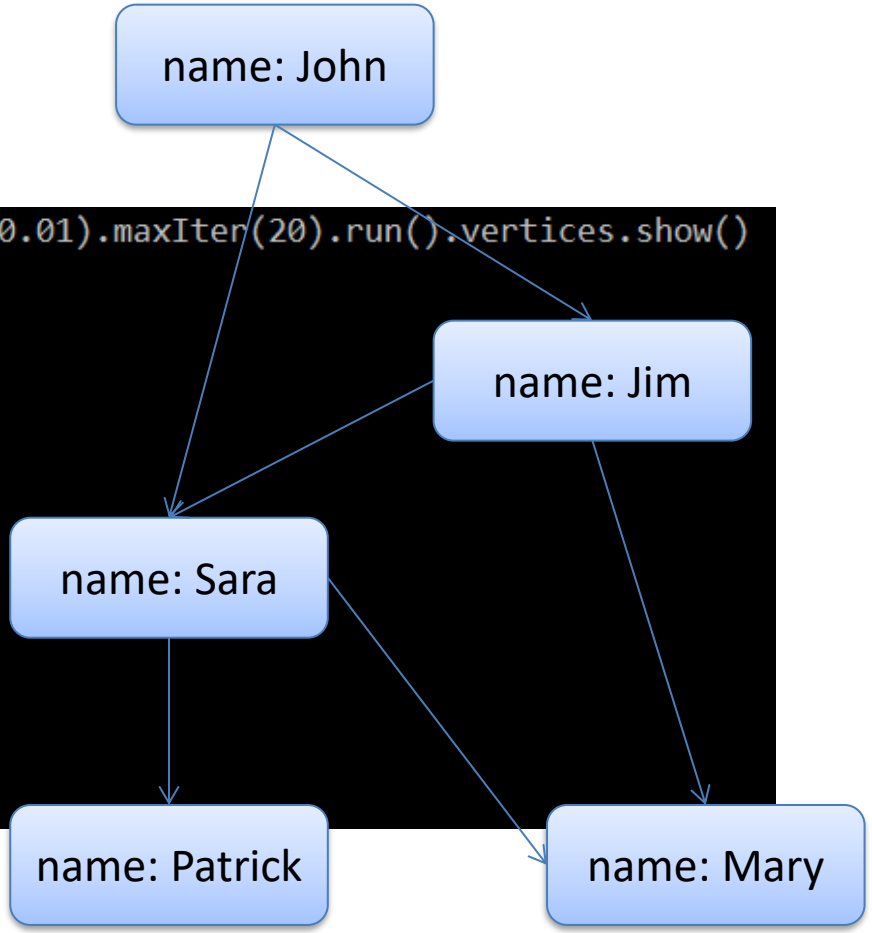
Now define edges & GraphFrame

```
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)
```



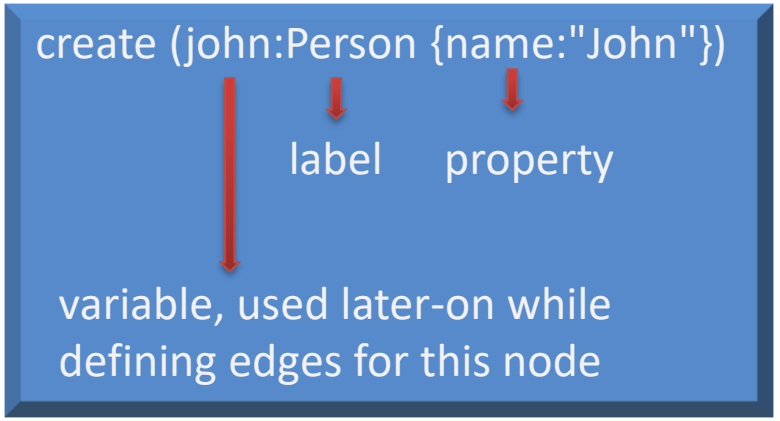
Computing PageRank: Spark/GraphX

```
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>
```



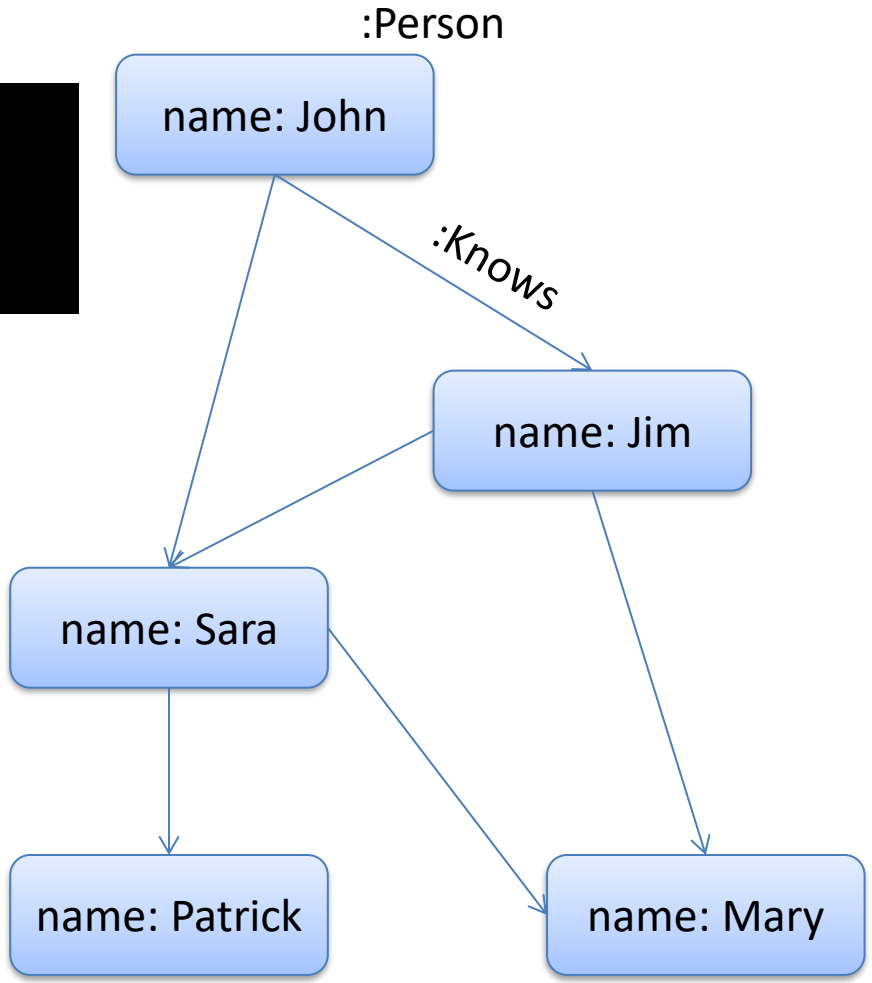
Create Social Graph script: Neo4j

```
//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);
```



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

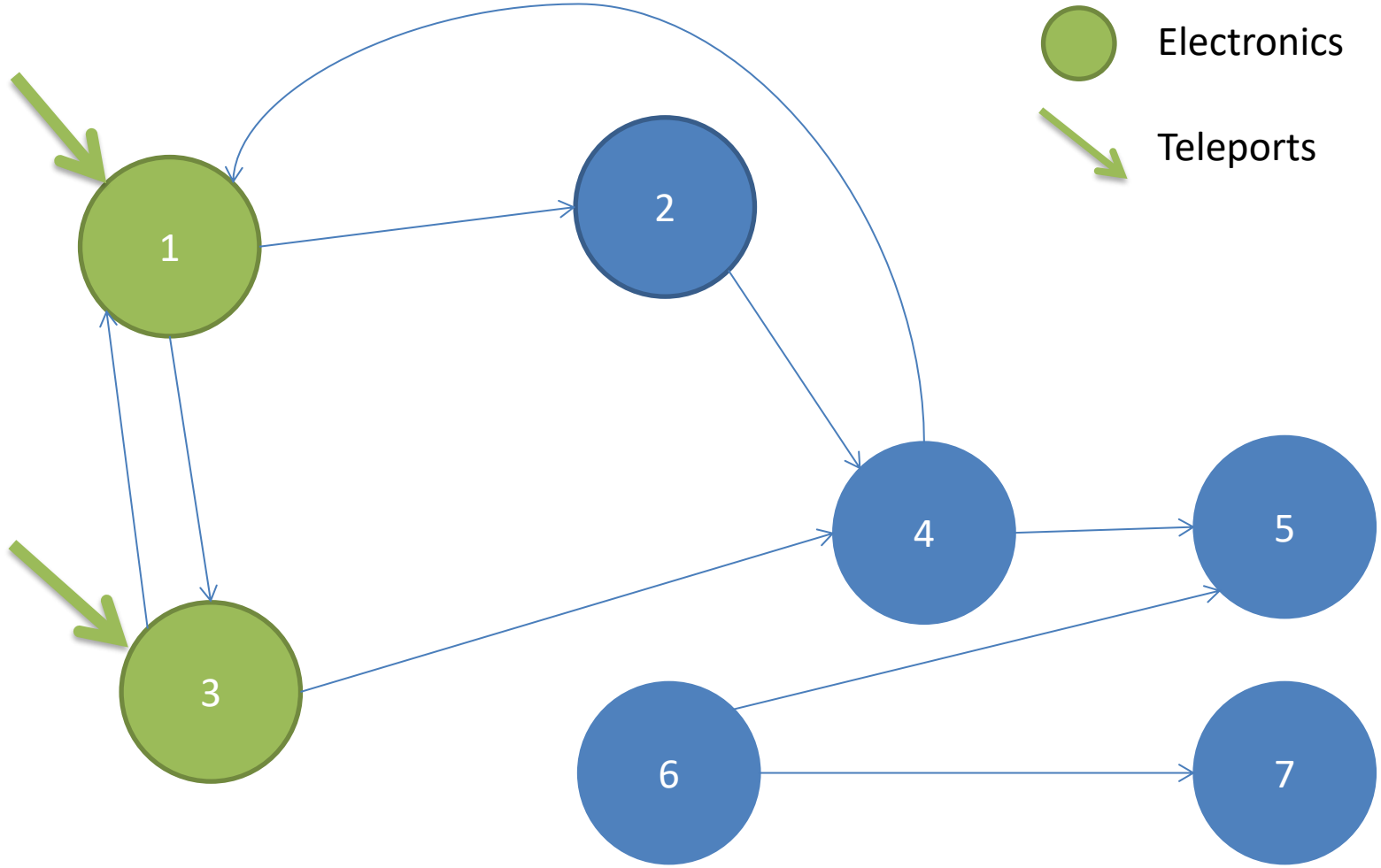
- "The significance of an electronics website is determined by the presence of inbound links from other noteworthy electronics websites."

- Web-surfer: initiate random walks from a selected pool of noteworthy web-sites on electronics

  = make restarts from these 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

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

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

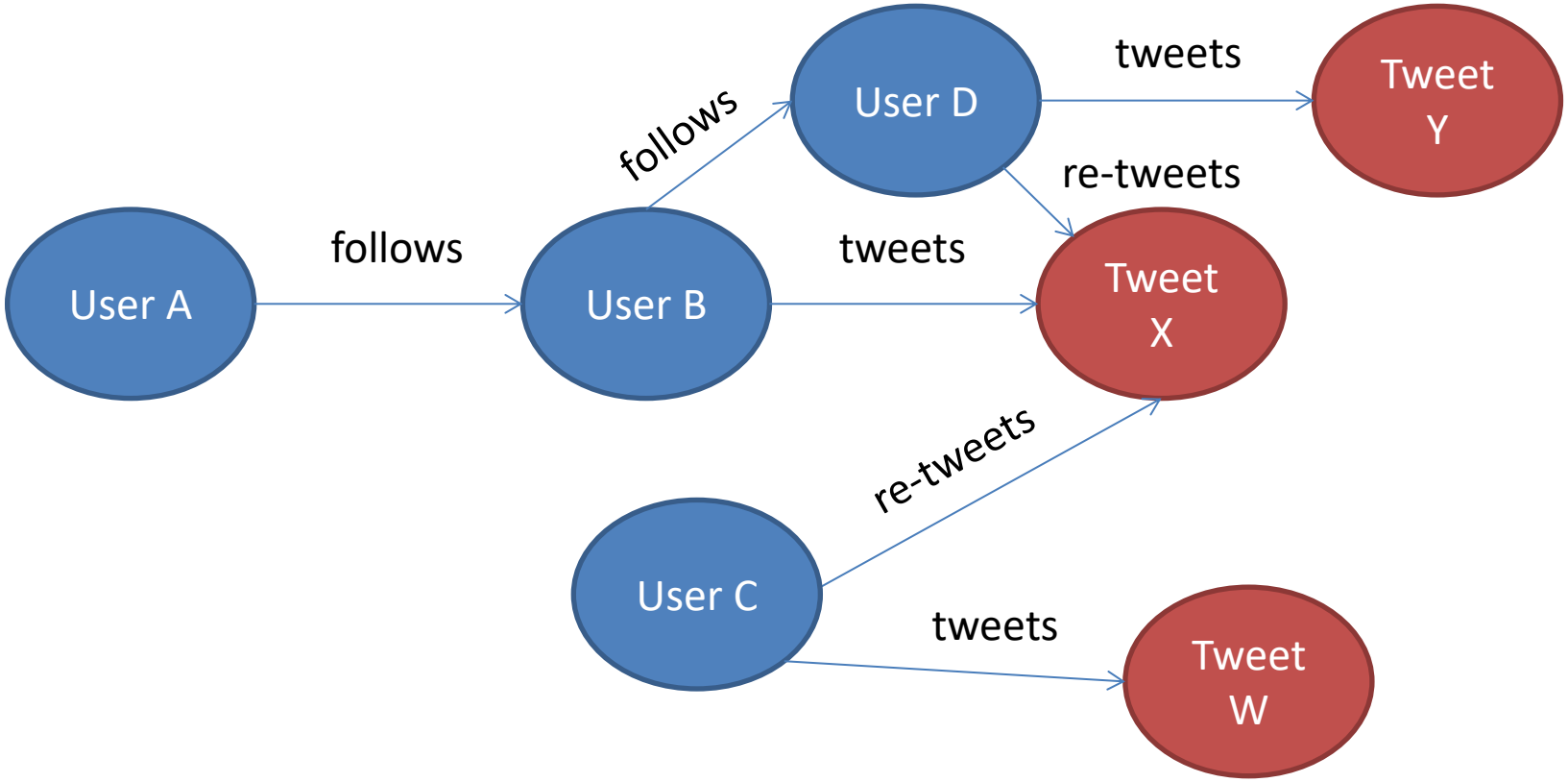
Who defines the topic on a given query?

- 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

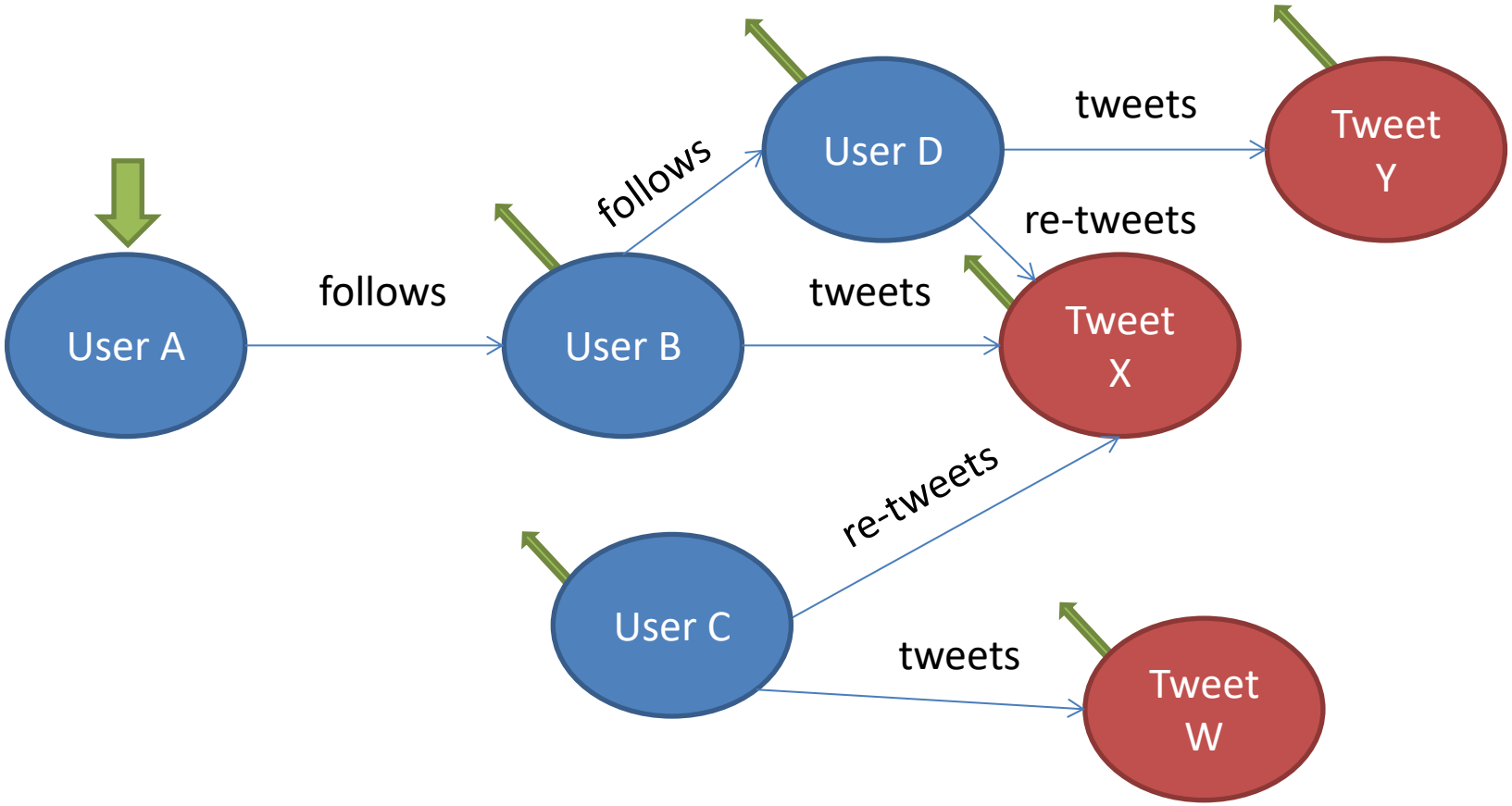
- PageRank assesses the importance of network nodes by considering incoming links as a form of positive endorsement for the credibility of a node.
- Many problems can be modeled in a similar setting.
- Note difference between global and personalized ranking.

Ranking tweets for a specific user



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

Personalized pageRank (set T={User A})

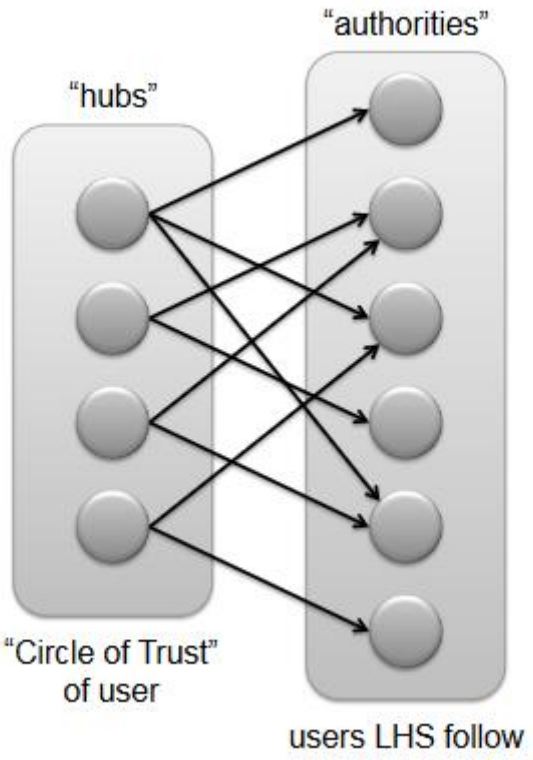


Note: equivalent to random walk with restart (at node User A)

Circle-of-Trust (Twitter)

source: "**WTF**: The Who to Follow Service at Twitter"

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



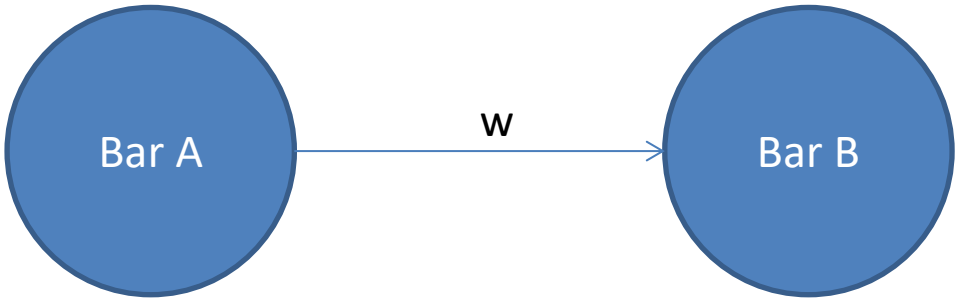
*SALSA=Stochastic Approach for Link-Structure Analysis

Bar Recommendations

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

- Ask for nice bars in Athens
- System pulls 1000 bars from database, must present top-10 to the user
- Idea: form a graph at runtime to rank bars

Bar Preference Graph



Create a link if a past user ranked/rated Bar B higher than bar 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**: Nodes that link to authorities (e.g. cars.com)*

***J. Kleinberg**: *Authoritative sources in a hyperlinked environment*, Journal of the ACM 46:604-632, 1999

Hubs

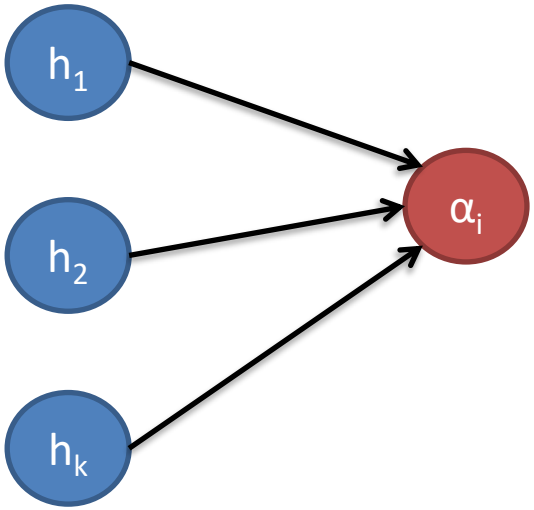
- Hubs are pages that helps you find relevant information by providing links towards the authoritative pages
  - E.g. cars.com, edmunds.com, 4troixoi.gr point to car manufacturers' pages
- A high-quality hub directs users to valuable authoritative pages
- A reputable authoritative page is linked to by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

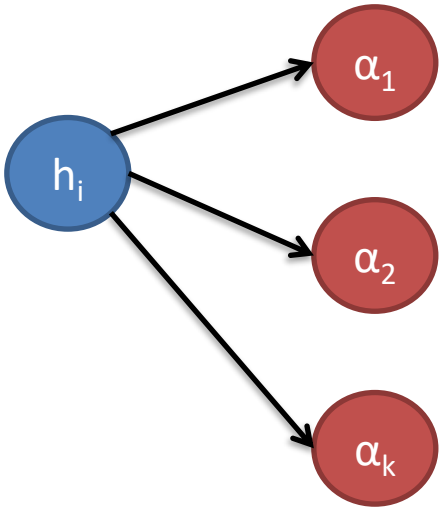
In a nutshell

- A web page is
  - a good authority if many good hubs link to it
  - a good hub if it links to many good authorities

Formulation

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





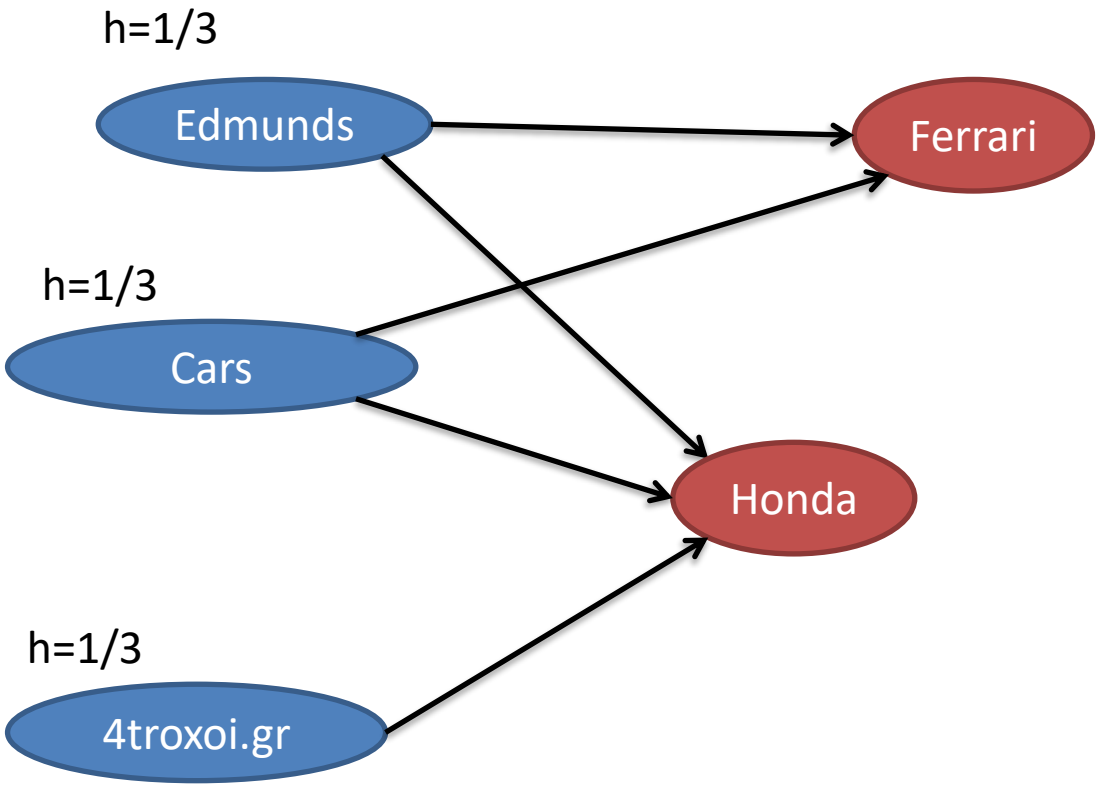
α_i = sum of h_j for all pages j pointing to page i

h_i = sum of α_j for all pages j pointed to by page i

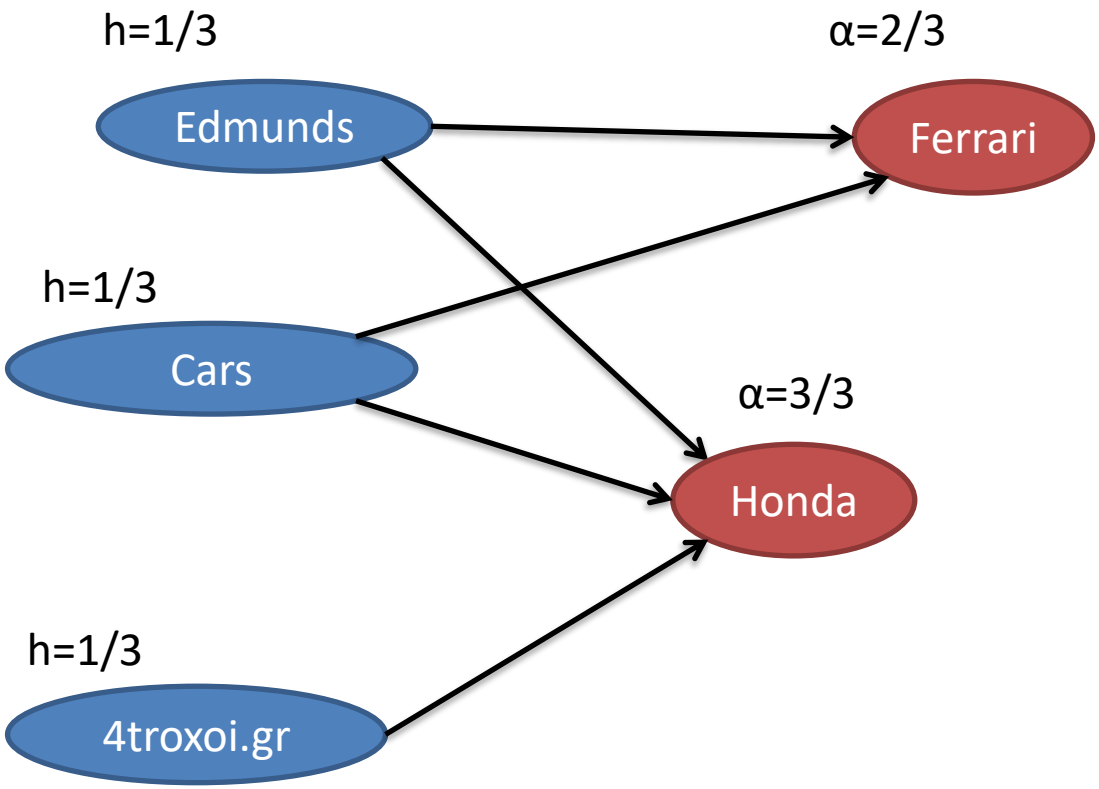
HITS Algorithm

- Initialize h_i scores (sum $h_i = 1$)
- Compute α_i scores from h_j scores of nodes j pointing to node i
  - Normalize α_i scores (divide each with sum(α_i))
- Compute h_i scores from α_j scores of nodes j pointed by node i
  - Normalize h_i scores (divide each with sum(h_i))
- Repeat until convergence

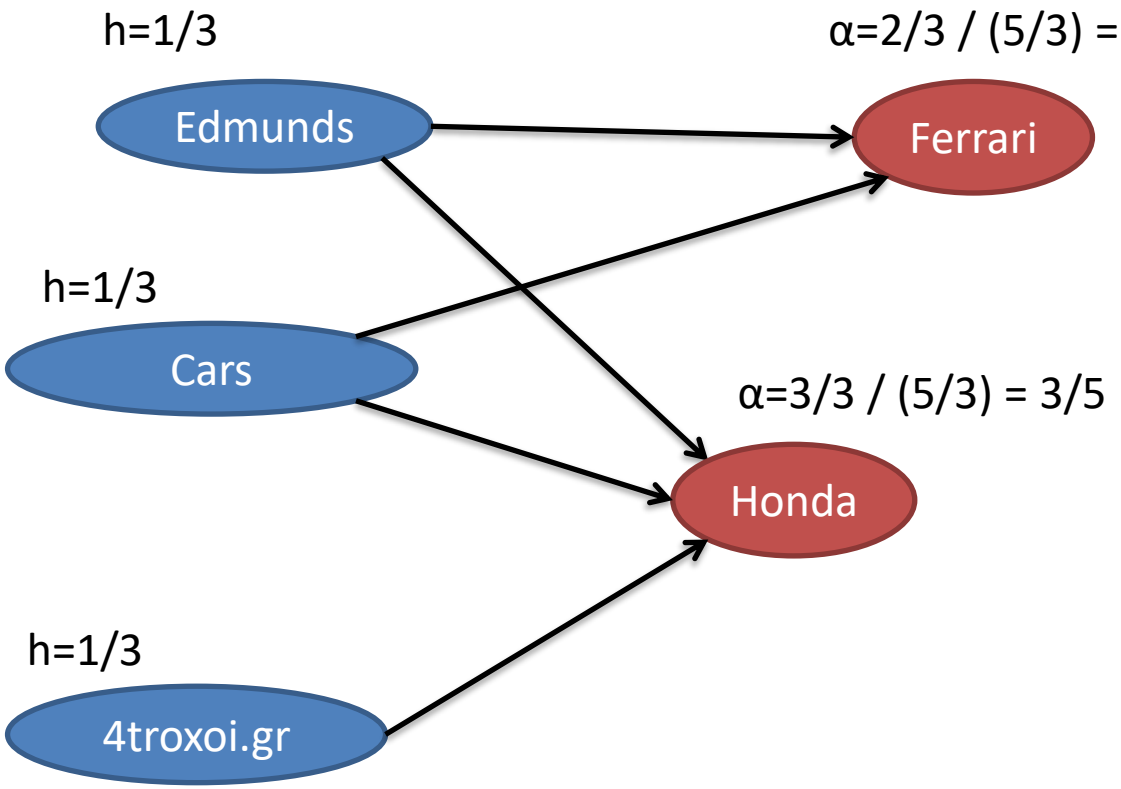
Simple Example



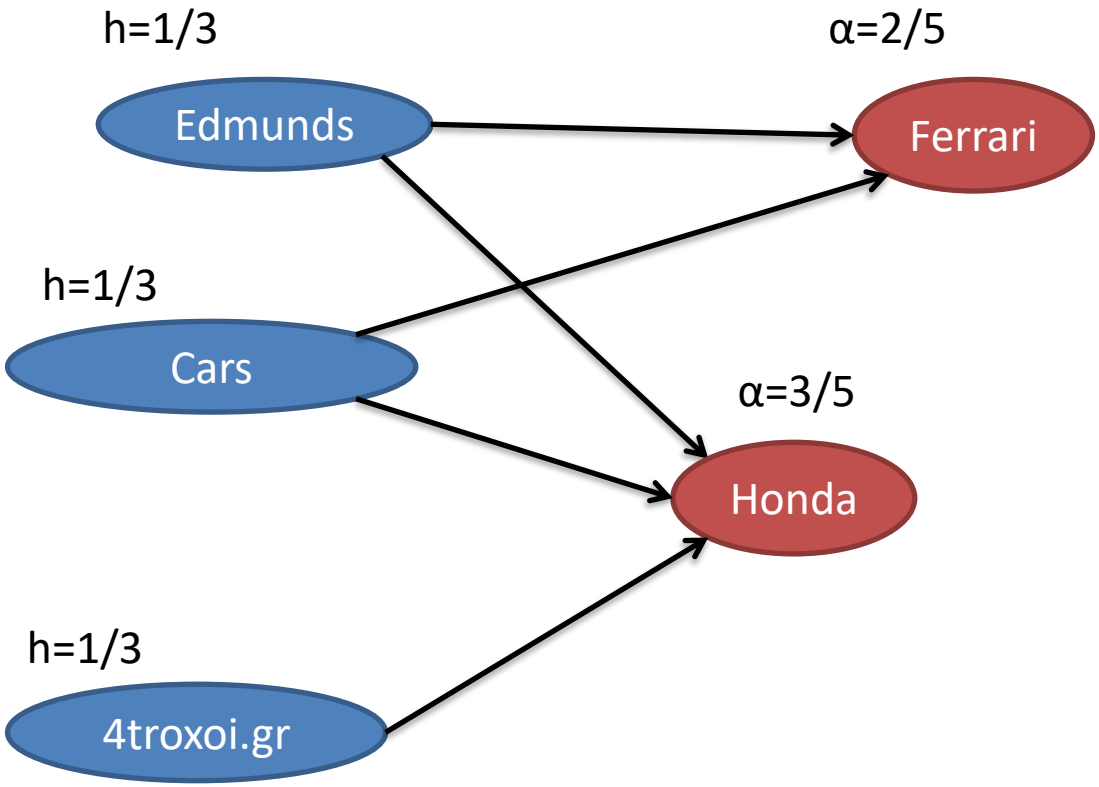
Update α



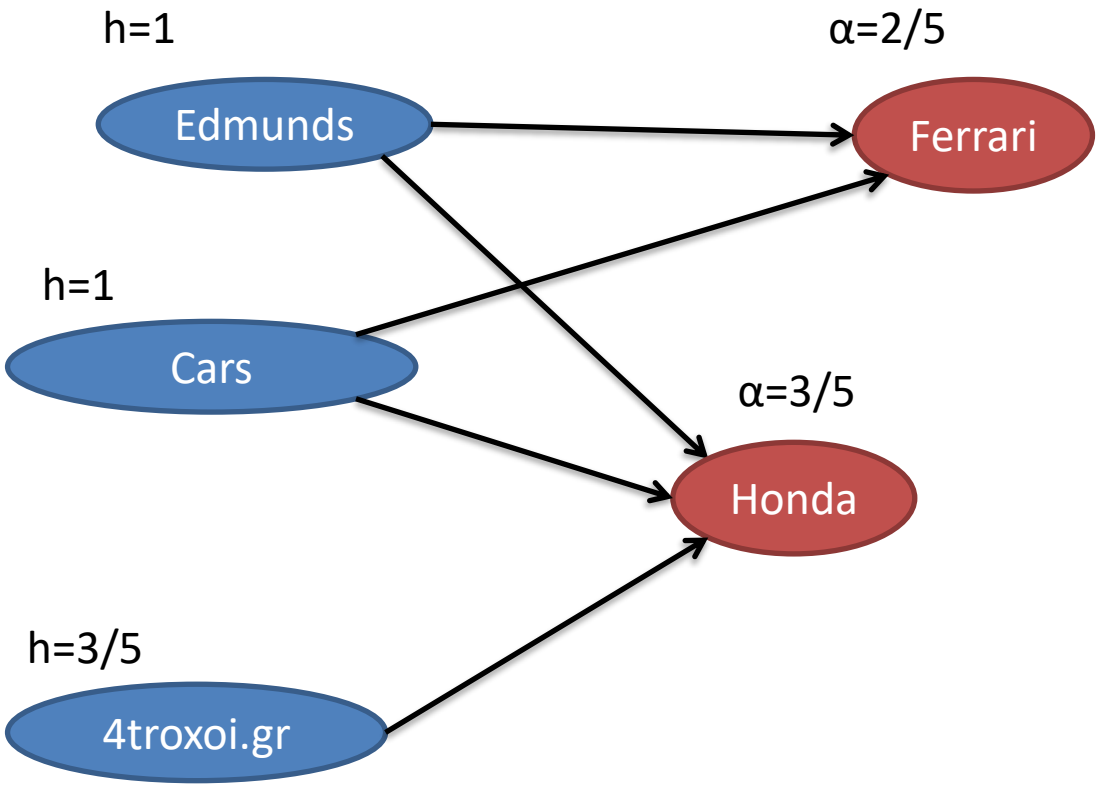
Normalize α values: divide by sum(α_i)



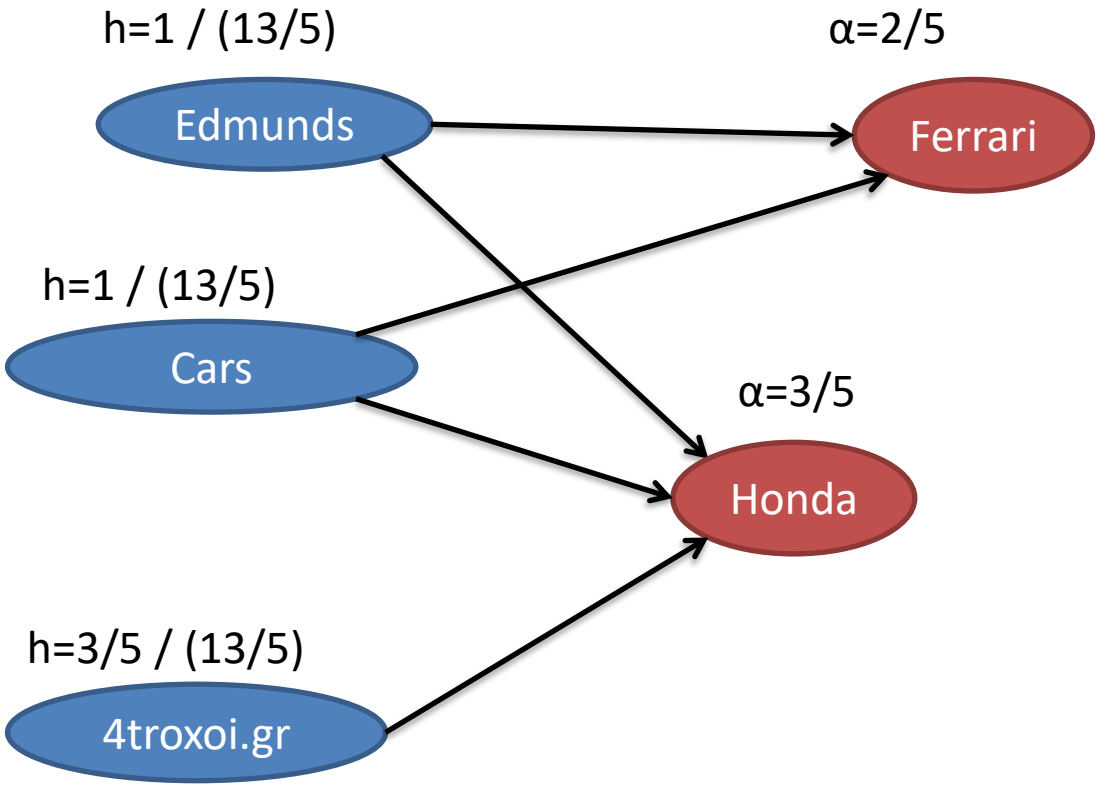
Normalized α values



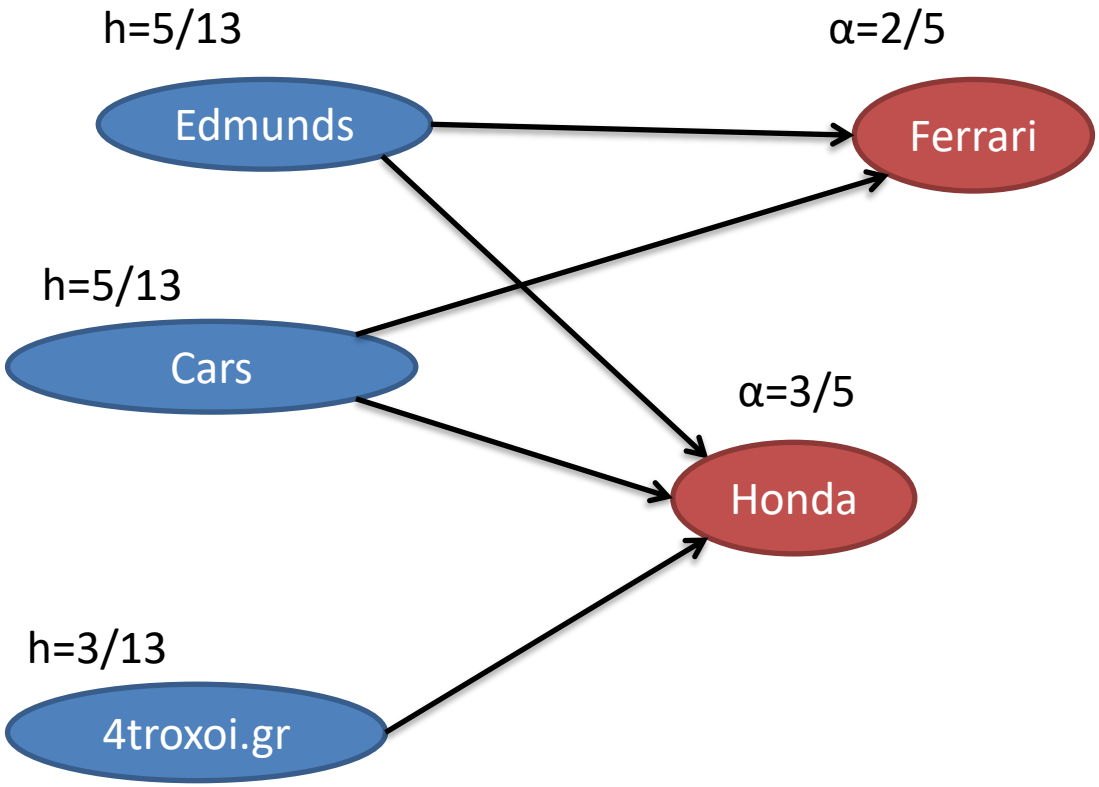
Update h



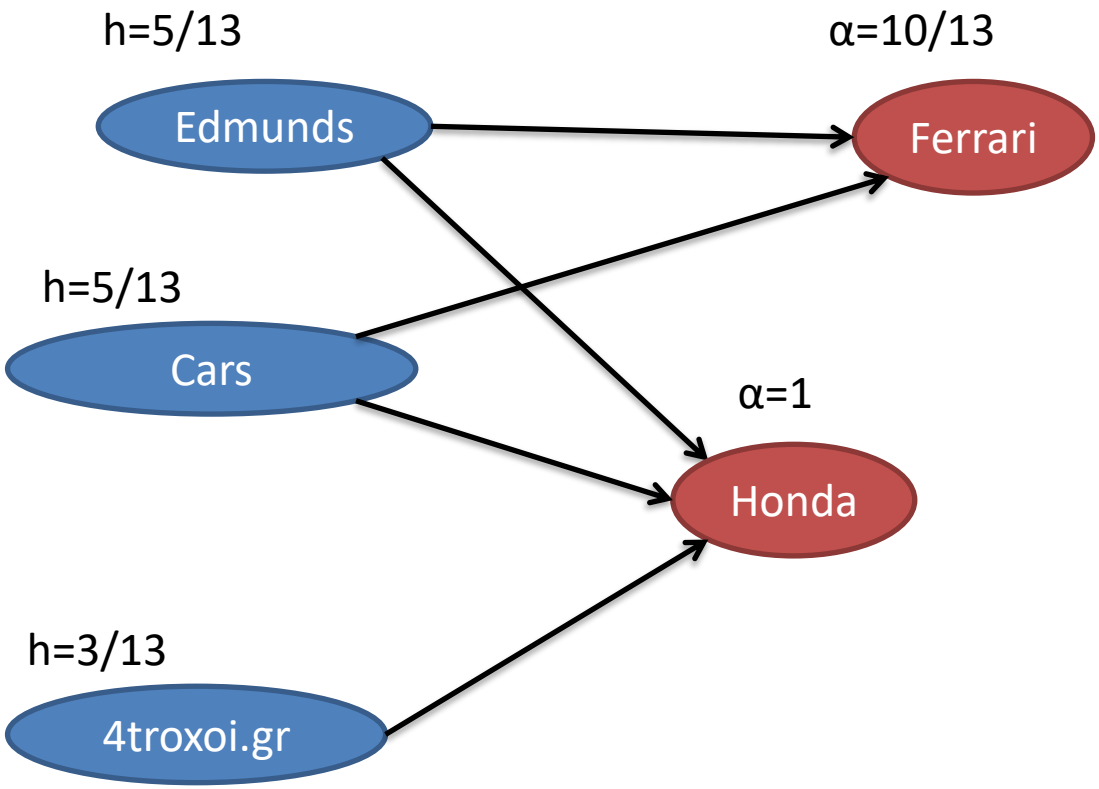
Normalize h values: divide by sum(h_i)



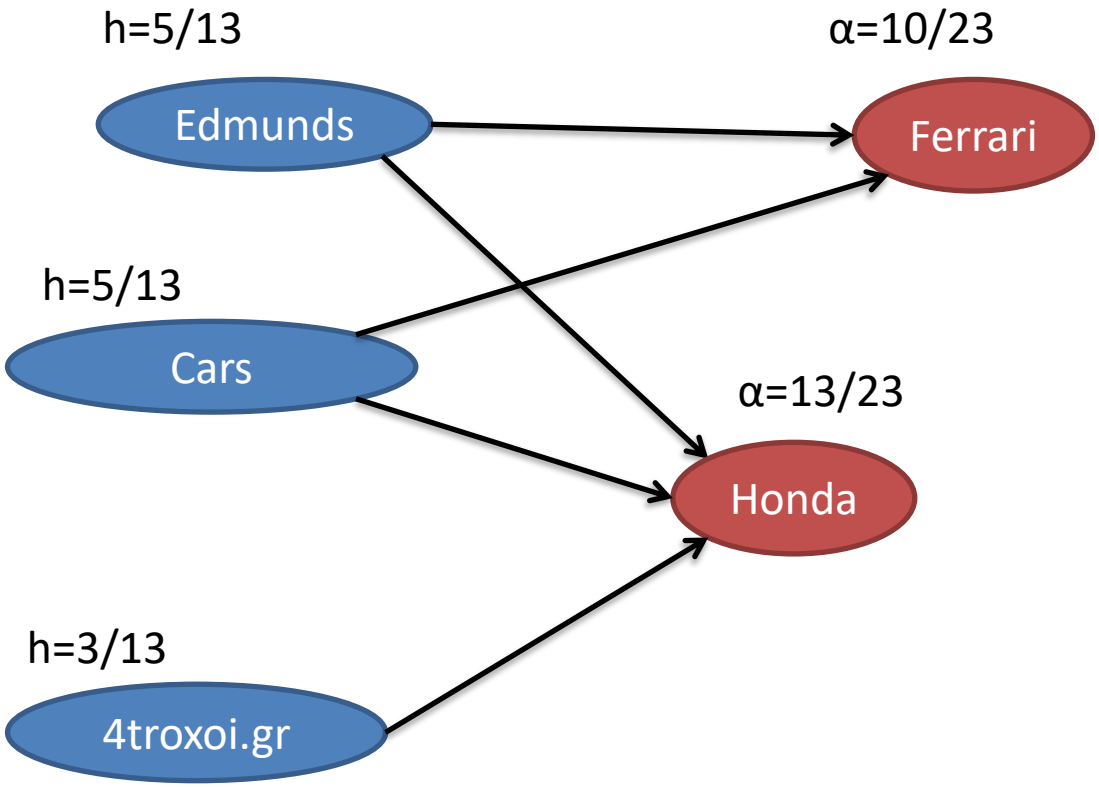
Normalized h values



Update α


h=5/13
α=10/13
Edmunds
Ferrari
h=5/13
Cars
α=1
Honda
h=3/13
4troxoi.gr

Normalized α values



HITS computation

$$a_i = \sum_{i \to j} h_j \qquad h_i = \sum_{i \to j} a_j$$

- Let vector a = (a_1,….a_n)
  vector h = (h_1,…h_n)
  nxn matrix A: A_{ij} = 1 if i→j

- Then $\alpha = A^T * h$, $h = A * \alpha$

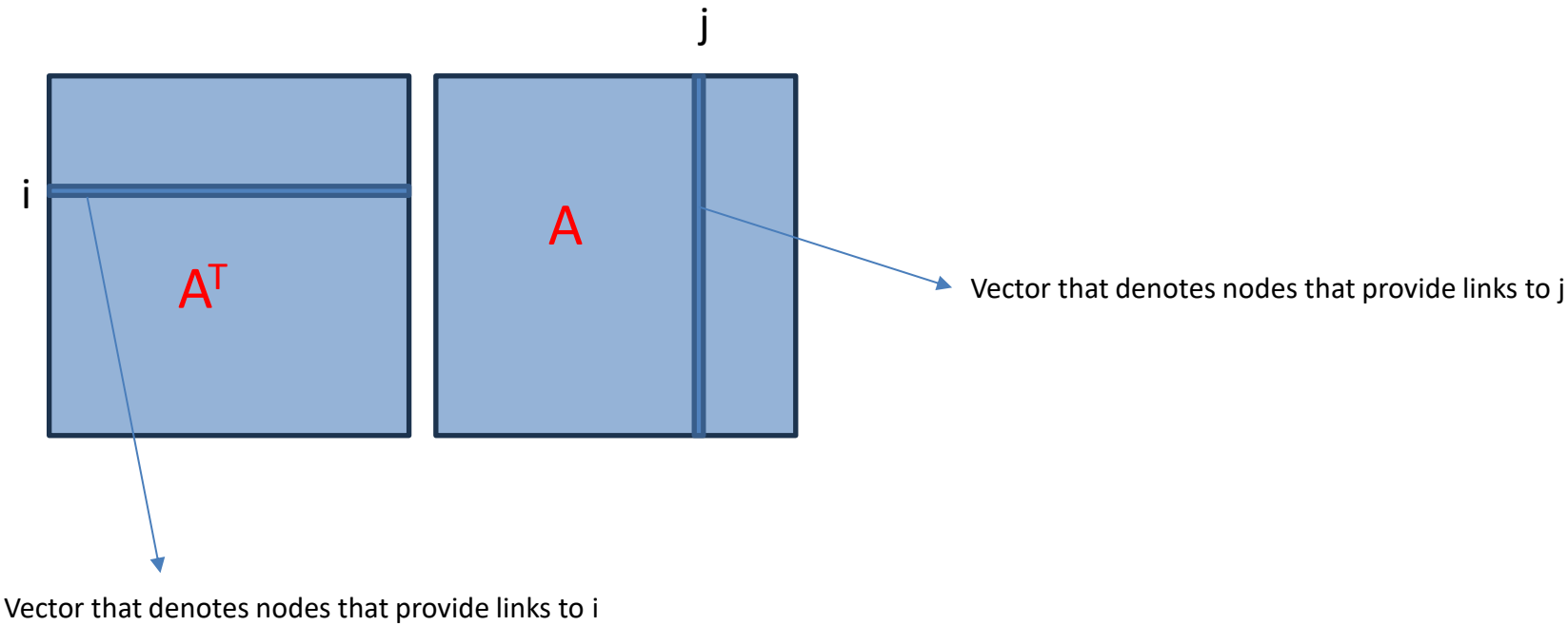
Convergence

- $\alpha = A^T * h$, $h = A * \alpha$
- 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 $(A * A^T)$

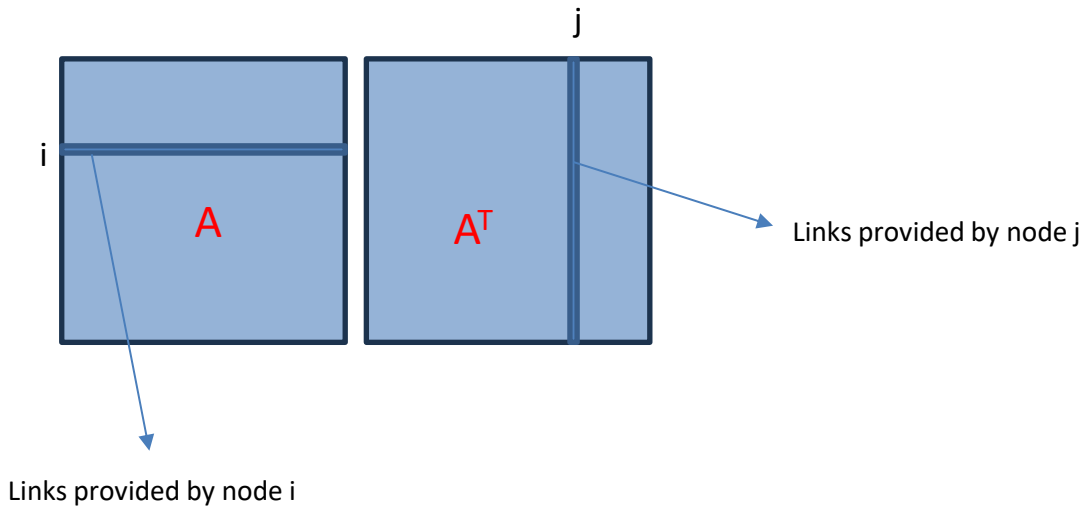
α: principal eigenvector of co-citation matrix

- $(A^T * A) * \alpha = 1 * \alpha$
- $(A^T * A)[i,j] = ?$ ⟹ Inner product of depicted vectors = number of nodes that link to both i and j



h: principal eigenvector of co-reference matrix

- $(A*A^T)*h = 1*h$
- $(A*A^T)[i,j] = ?$ ⇨ Inner product of depicted vectors = number of nodes to which both i and j link



PageRank vs HITS

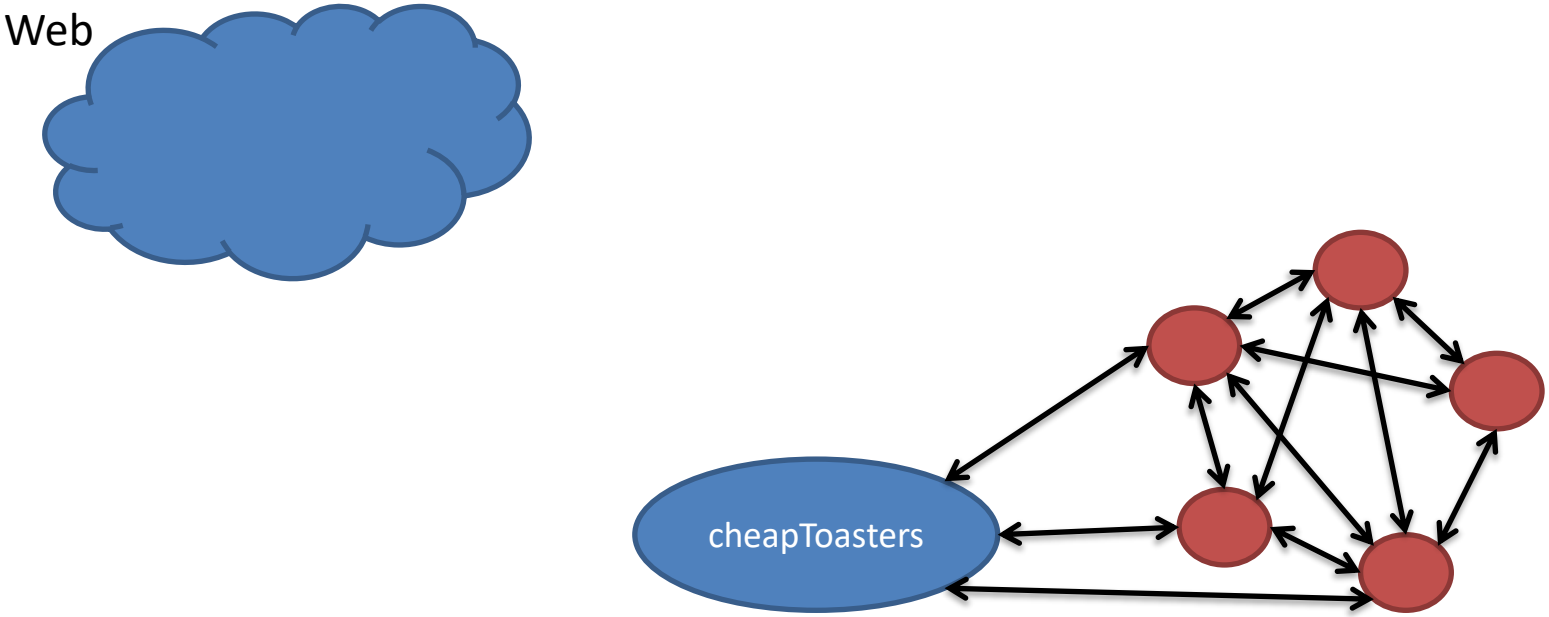
- Both use links to compute importance of a web page
- In PageRank the value of an incoming link u→v to page v depends on the links k→u into page u
  - Seek links from pages that receive links from high quality pages
- In HITS, the value of the in-link u→v depends on the value of other links u→k out of u
  - Seek links from pages that provide links to high quality pages

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 !

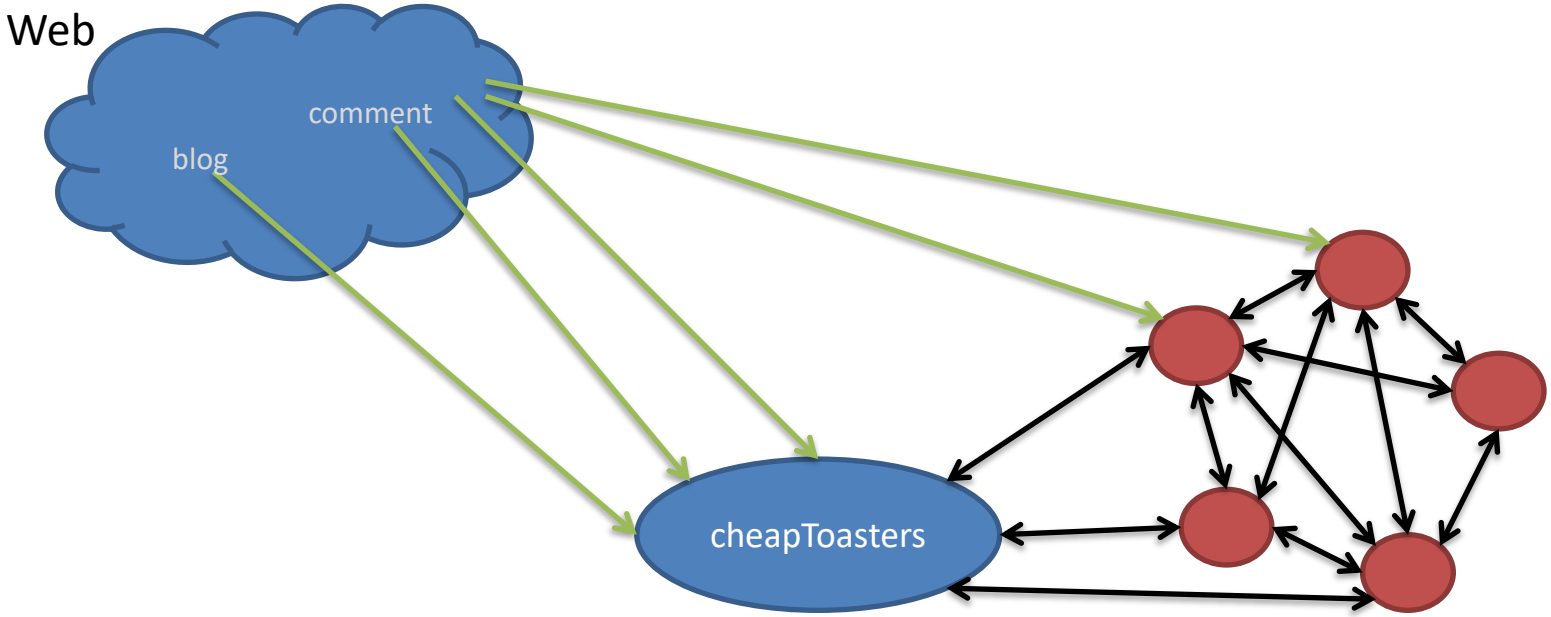
Boost my cheapToasters.com page!



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

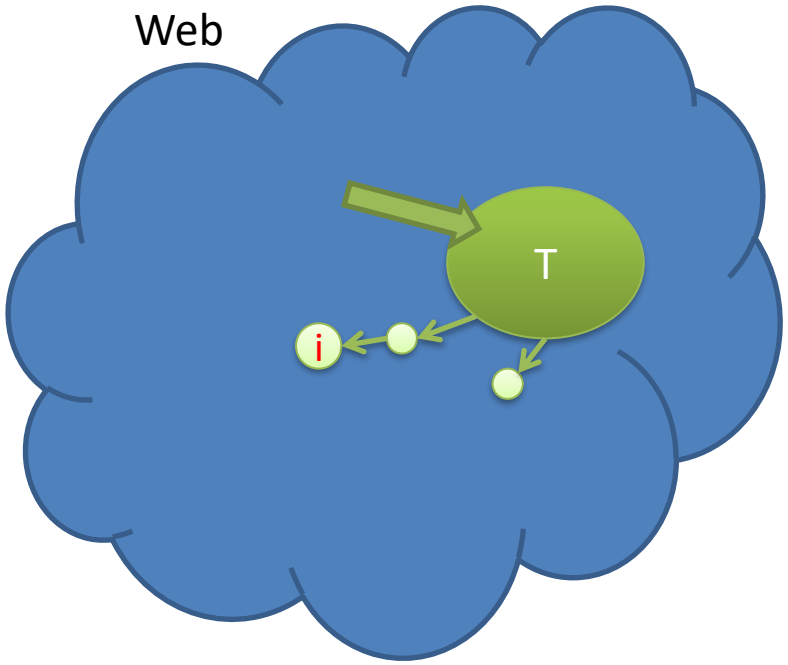
- Similar idea to topic-specific PageRank

- 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

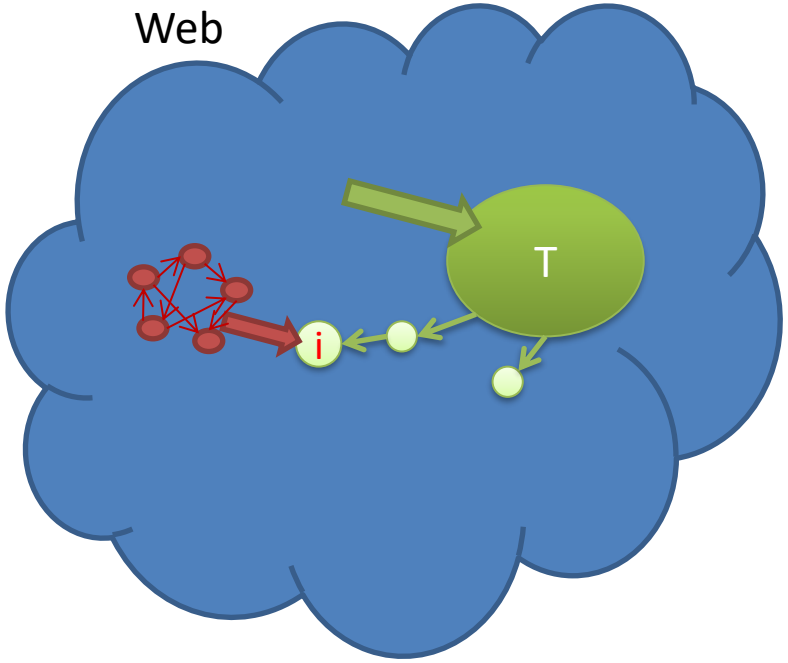


t_i = TrustRank of page I
p_i = PageRank of page i

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

Spam-mass

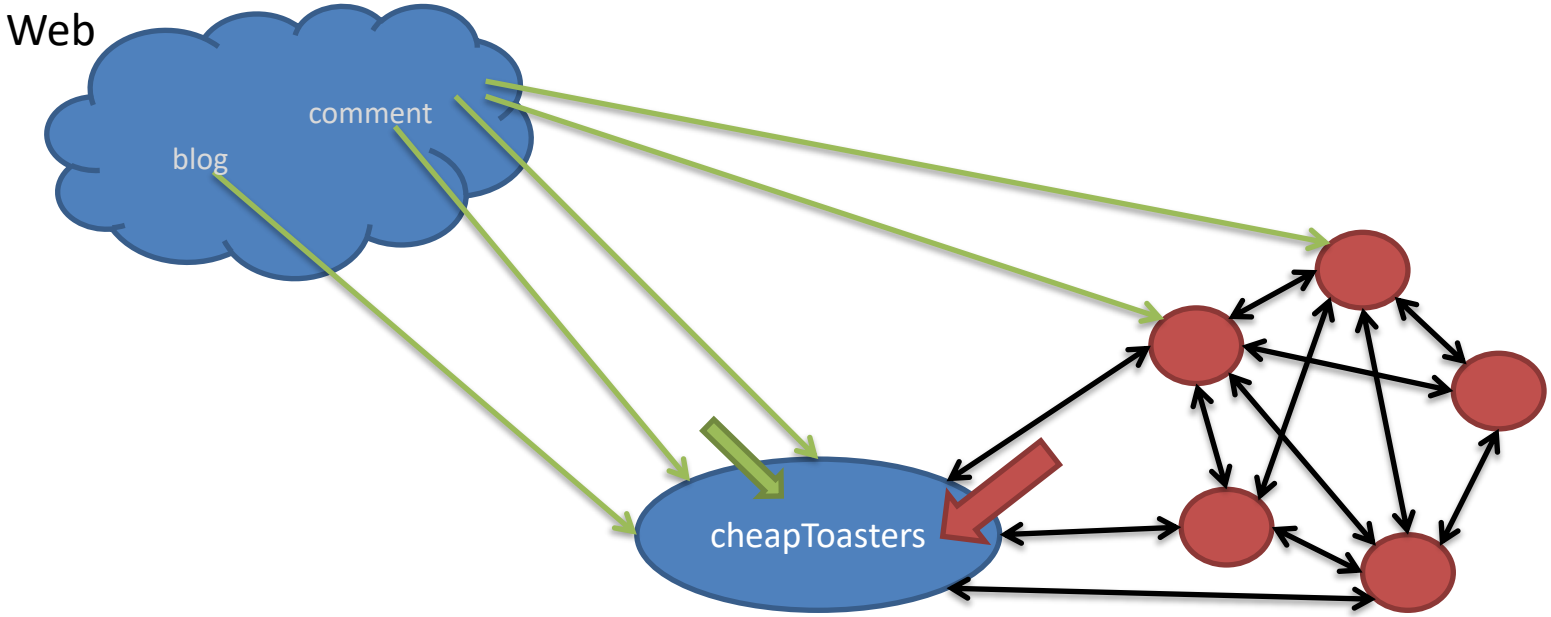
- Portion of PageRank that comes from spam
  - Small or negative values mean that page is most likely not spam
  - A value close to 1 means that the page is probably spam



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 web-site (directly or indirectly), then its spam-mass will be close to 1

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