Web Search Engine

Serge Abiteboul

INRIA Saclay, Collège de France, ENS Cachan

Goal

- Use the Web to answer queries
  - Input: some keywords
  - Output: pages that contain these keywords
- Main tasks
  1. Crawl the Web to retrieve pages
  2. Build an index for these pages
  3. Rank the pages (to sort results)
  4. Fight the bad guys

Organization

Introduction

1. Crawling the Web
2. Indexing Web pages
3. Ranking Web pages
4. Fighting the bad guys

Jewel: Distributed page ranking

Conclusion

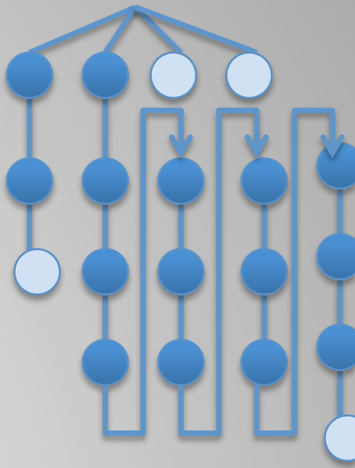
Crawling the pages

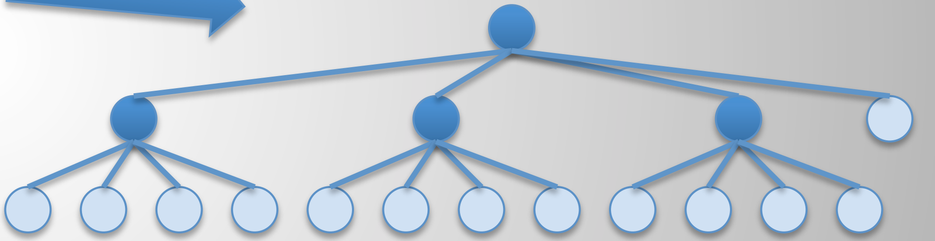
Web crawler

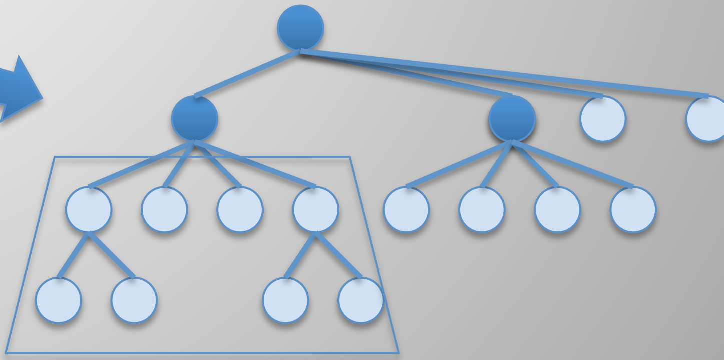
- Crawlers , (Web) spiders , (Web) robots
  - Autonomous agents that retrieve pages from the Web
- Basics of crawling:
  1. Start from a given URL or set of URLs
  2. Retrieve and index the corresponding pages
  3. Discover hyperlinks (**<a>** elements)
  4. Repeat on each link that has been found
- No real termination condition
  - Virtually unlimited number of Web pages
  - Continuous crawl: reread pages that may have changed

Web crawler: graph traversal

- Depth-first: not very adapted, possibility of being lost in robot traps
- Breadth-first: better adapted
- Combination of both: breadth-first but when a site is discovered, crawl it up to certain depth
- Consider also the popularity of pages







Duplicate pages

- Identify (near-duplicates) pages/sites to index less
- Trivial duplicates: same resource at the same canonized URL:

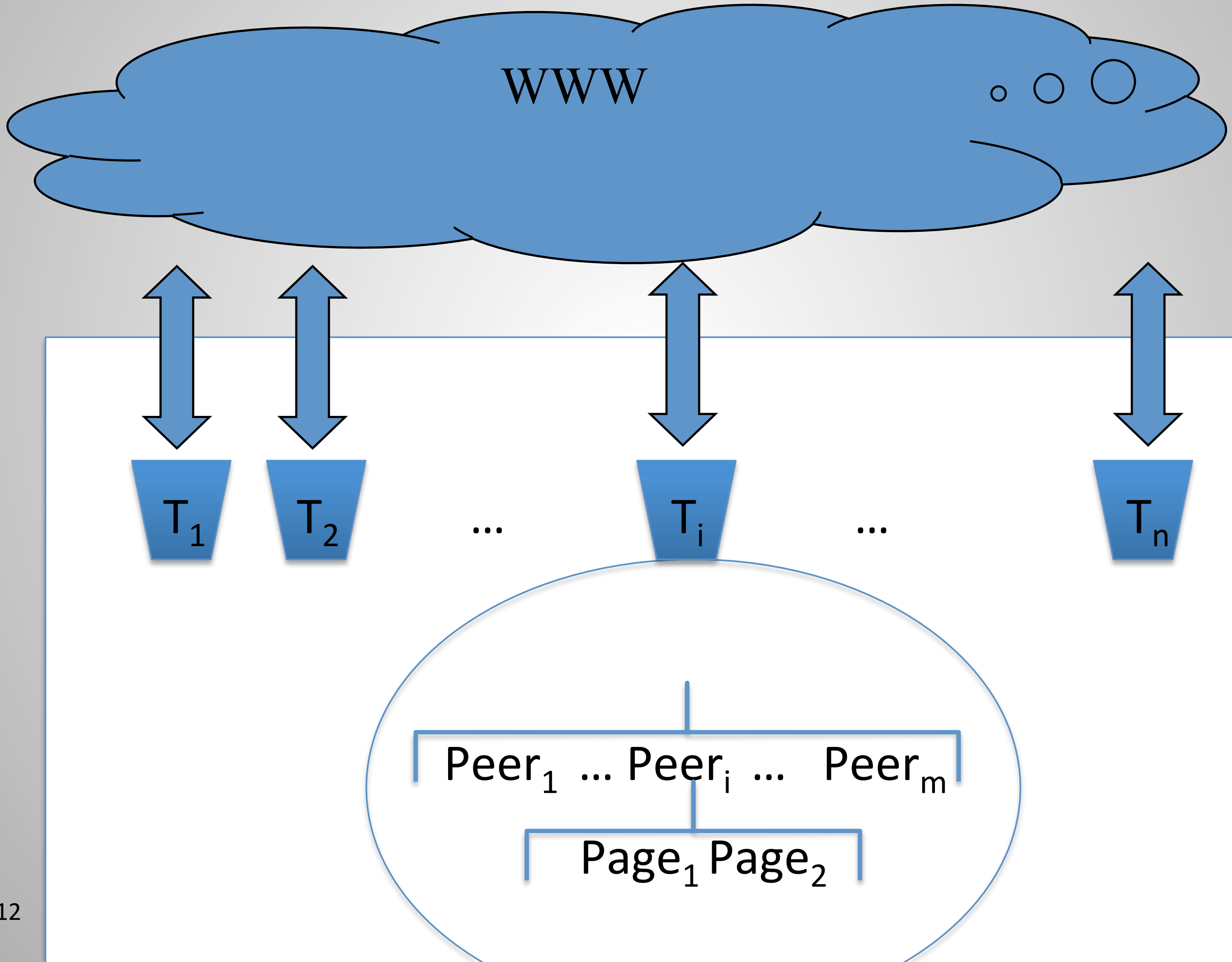
  http://example.com:80/toto

  http://example.com/titi/../toto
- Exact duplicates: identification by hashing
- Near-duplicates: (timestamps, tip of the day, etc.) identification by hashing of sequences of n successive tokens (n-grams)

Web crawler

- Follow robot exclusion rules of servers (robots.txt)
- Follow robot ethics:
  - Avoid Denial Of Service (DOS), wait 100ms/1s between two repeated requests to the same Web server
- Parallelism
  - Many crawler machines – each is multithreaded:

One crawler machine – multithreads



Indexing Web pages

Inverted index

Mot	Numéro de page
…	
Abiteboul	34,356,223,9900 …
…	
INRIA	56,356,777,6560, …
…	
Rocquencourt	27,777,9890,22290 …
…	

num	url
…	…
356	abiteboul.com
…	…
777	www.inria.fr…
…	…

“index inversé”

Preliminary tasks

- Tokenization: separate text into lexical units called *tokens*

  tokenisation

- Stemming: merge different forms of the same word

  lemmatisation

  – Also possible: Phonetic stemming; merge phonetically related words; handle spelling errors

- Stop word removal

  retrait des mots vides

Example

- **d**1 The jaguar is a New World mammal of the Felidae family.
- **d**2 Jaguar has designed four new engines.
- **d**3 For Jaguar, Atari was keen to use a 68K family device.
- **d**4 The Jacksonville Jaguars are a professional US football team.
- **d**5 Mac OS X Jaguar is available at a price of US $199 for Apple’s new “family pack”.
- **d**6 One such ruling family to incorporate the jaguar into their name is Jaguar Paw.
- **d**7 It is a big cat.

Difficult in some languages

Tokenization

Separate text into tokens

Remove punctuation, normalize case

- **d**1 the1 jaguar2 is3 a4 new5 world6 mammal7 of8 the9 felidae10 family11
- **d**2 jaguar1 has2 designed3 four4 new5 engines6
- **d**3 for1 jaguar2 atari3 was4 keen5 to6 use7 a8 68k9 family10 device11
- **d**4 the1 jacksonville2 jaguars3 are4 a5 professional6 us7 football8 team9
- **d**5 mac1 os2 x3 jaguar4 is5 available6 at7 a8 price9 of10 us11 $19912 for13 apple's14 new15 family16 pack17
- **d**6 one1 such2 ruling3 family4 to5 incorporate6 the7 jaguar8 into9 their10 name11 is12 jaguar13 paw14
- **d**7 it1 is2 a3 big4 cat5

Stemming (“lemmatisation”)

is ☞ be

Stemming: merge different forms of the same word

- **d**1 the1 jaguar2 be3 a4 new5 world6 mammal7 of8 the9 felidae10 family11
- **d**2 jaguar1 have2 design3 four4 new5 engine6
- **d**3 for1 jaguar2 atari3 be4 keen5 to6 use7 a8 68k9 family10 device11
- **d**4 the1 jacksonville2 jaguar3 be4 a5 professional6 us7 football8 team9
- **d**5 mac1 os2 x3 jaguar4 be5 available6 at7 a8 price9 of10 us11 $19912 for13 apple14 new15 family16 pack17
- **d**6 one1 such2 rule3 family4 to5 incorporate6 the7 jaguar8 into9 their10 name11 be12 jaguar13 paw14
- **d**7 it1 be2 a3 big4 cat5

Stop word removal

Remove too frequent words that bring no information

- **d**1 jaguar2 new5 world6 mammal7 felidae10 family11
- **d**2 jaguar1 design3 four4 new5 engine6
- **d**3 jaguar2 atari3 keen5 68k9 family10 device11
- **d**4 jacksonville2 jaguar3 professional6 us7 football8 team9
- **d**5 mac1 os2 x3 jaguar4 available6 price9 us11 $19912 apple14 new15 family16 pack17
- **d**6 one1 such2 rule3 family4 incorporate6 jaguar8 their10 name11 jaguar13 paw14
- **d**7 ~~it1 be2 a3~~ big4 cat5

Inverted index construction

- family **d**1 , **d**3 , **d**5 , **d**6
- football **d**4
- jaguar **d**1 , **d**2 , **d**3 , **d**4 , **d**5 , **d**6
- new **d**1 , **d**2 , **d**5
- rule **d**6
- us **d**4 , **d**5
- world **d**1
- . . .

Indexing: scaling

A query should require zero or very few disk accesses

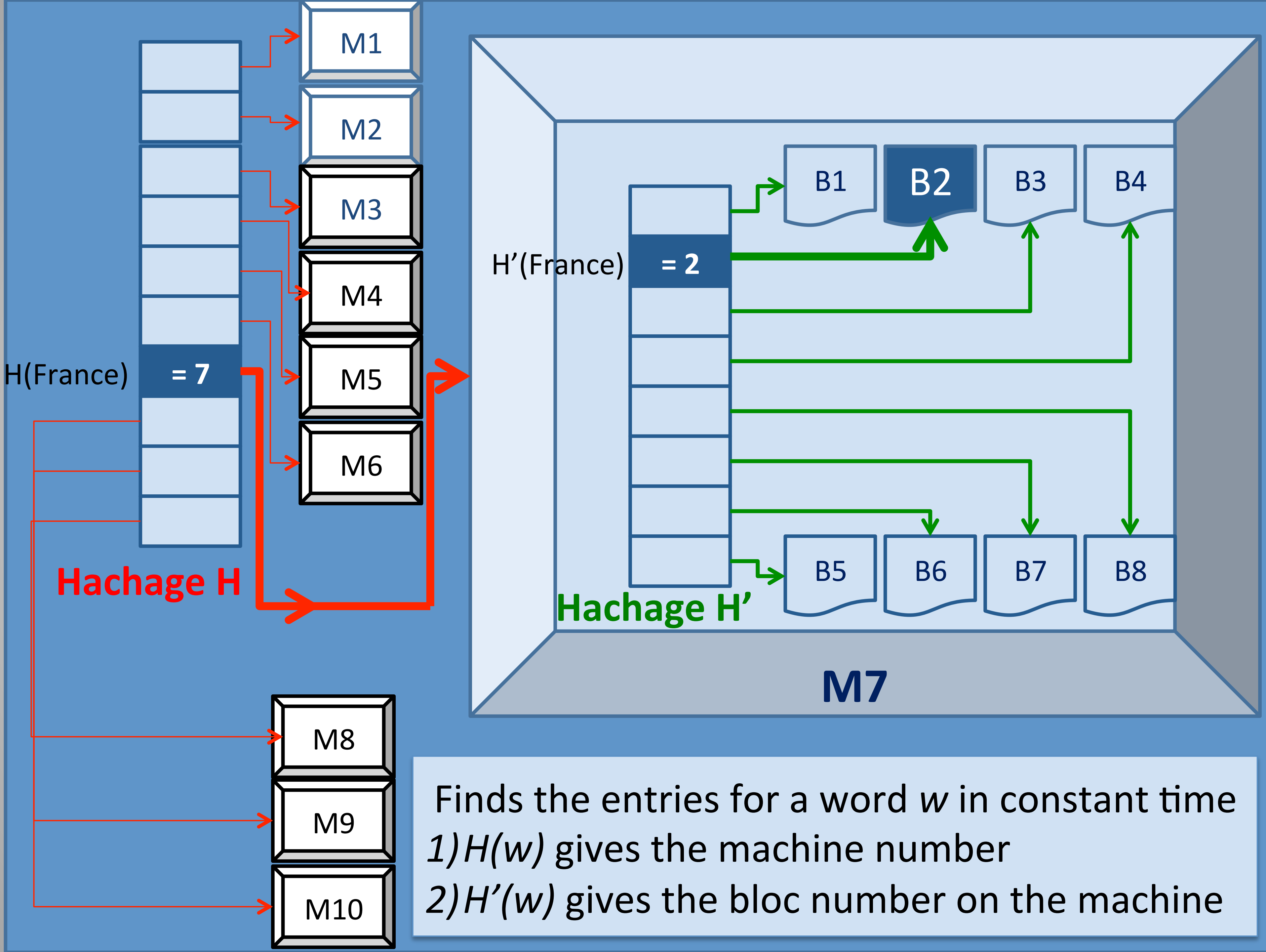
If the number of indexed pages grows, the server needs more storage to keep the index, and each query is becoming more expensive to evaluate

If the number of users grows, the server receives more requests

In both cases, the server is quickly overwhelmed

Solution

- The technique of hashing and
- Parallelism



Finds the entries for a word *w* in constant time

1) *H(w)* gives the machine number

2) *H'(w)* gives the bloc number on the machine

The scale, e.g., for Google search

The size of the index

- Billions of pages
- The size of the index is roughly that of the indexed text

Dozens of billions of queries per month

Farms everywhere in the world with thousands of machines

Ranking Web pages

Main issue: how to select the pages proposed in first pages of answer?

The witchery of search engines

There may be hundreds of **millions** of result-pages that contain the keywords of the query

A user will look at the first pages of results – 10 result-pages

- Possibly a few pages of pages – rarely **one hundred**

The witchery of search: place the result-pages the user wants in the first pages of results

- Information retrieval measures: e.g., TFIDF
- Measures based on the analysis of Web graph: e.g., PageRank

TF-IDF

Weight term occurrences: E.g., TF-IDF

Terms occurring frequently in a given document : more relevant

Terms occurring rarely in the collection : more informative

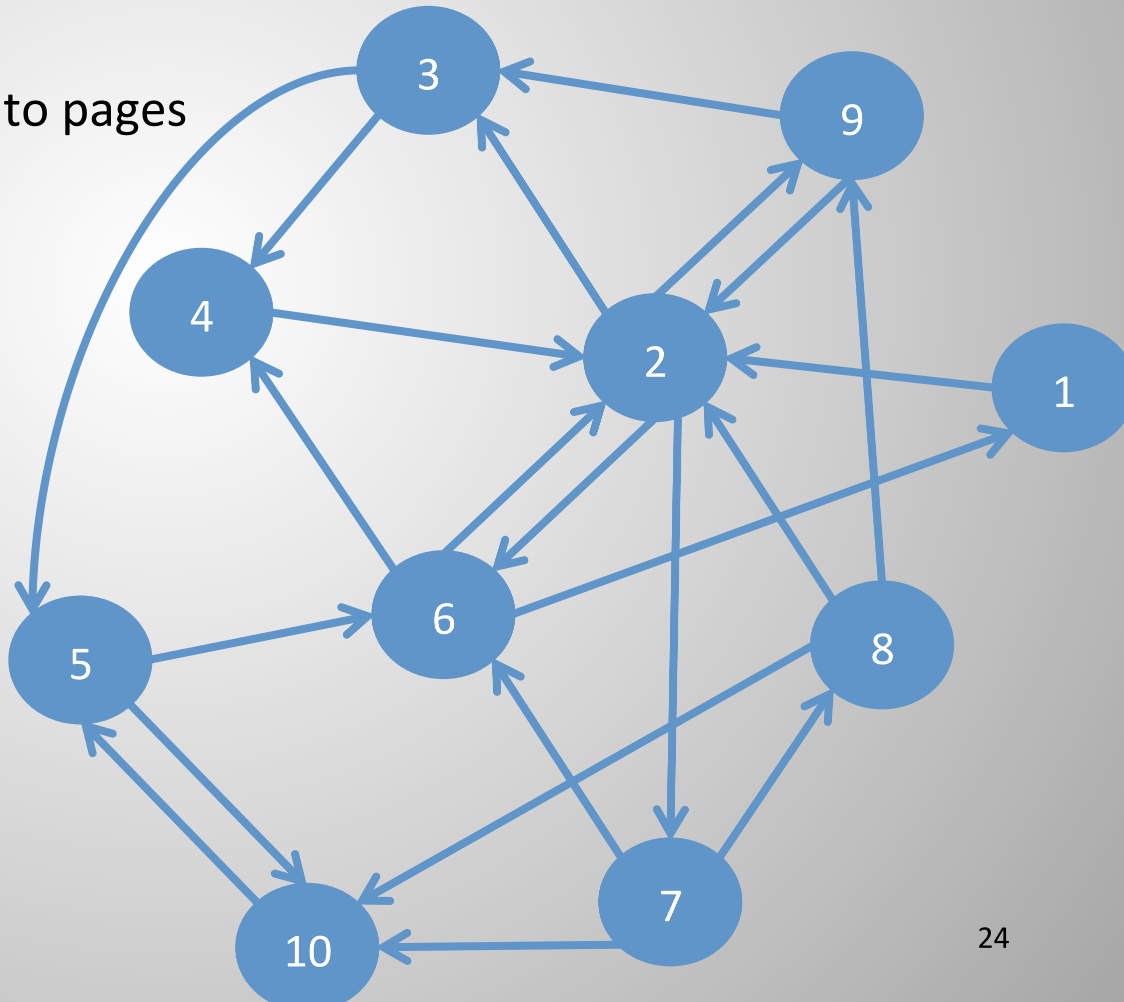
The Web is a graph

A directed graph

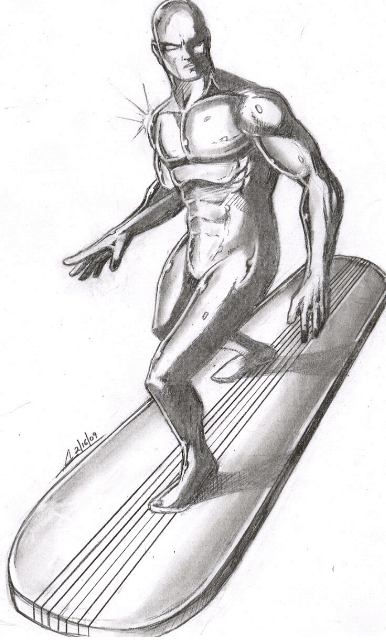
- Vertices: Web pages
- Edges: Links from pages to pages

Many other fun graphs

- Social networks
- Scientific publications
- Encyclopedias …



The random surfer of the Web



Random surfer's algorithm

- Start: choose a random page on the Web
- Follow links: choose between them randomly
- Keep going on for ever
- In a dead-end: choose a random page on the Web

Popularity of a page: probability for a surfer to be in a page after an infinite time

High probability for popular pages

How can we compute this popularity?

Introduced by Brin and Page: PageRank

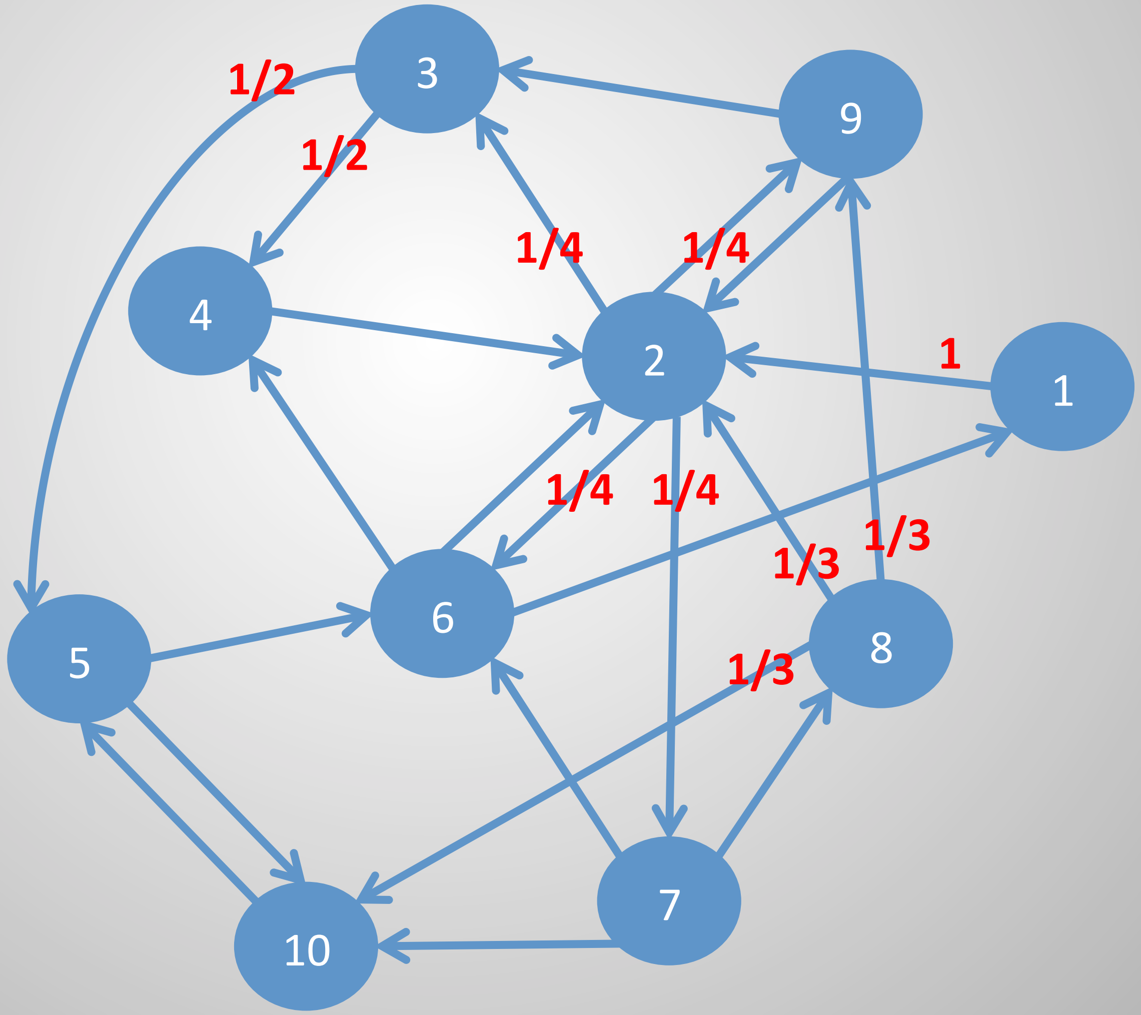
Transfer matrix

$G(i,j) = 0$ if there is no link from page i to page j;

$G(i,j) = 1/n_i$ otherwise, with *n_i the number of outgoing links of page i*

$$G = \begin{matrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 0 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \\ 0 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Popularity transfers



Computing the popularity of pages

We consider the **fixpoint equation**

$$\mathbf{Pop = G^T \cdot Pop}$$

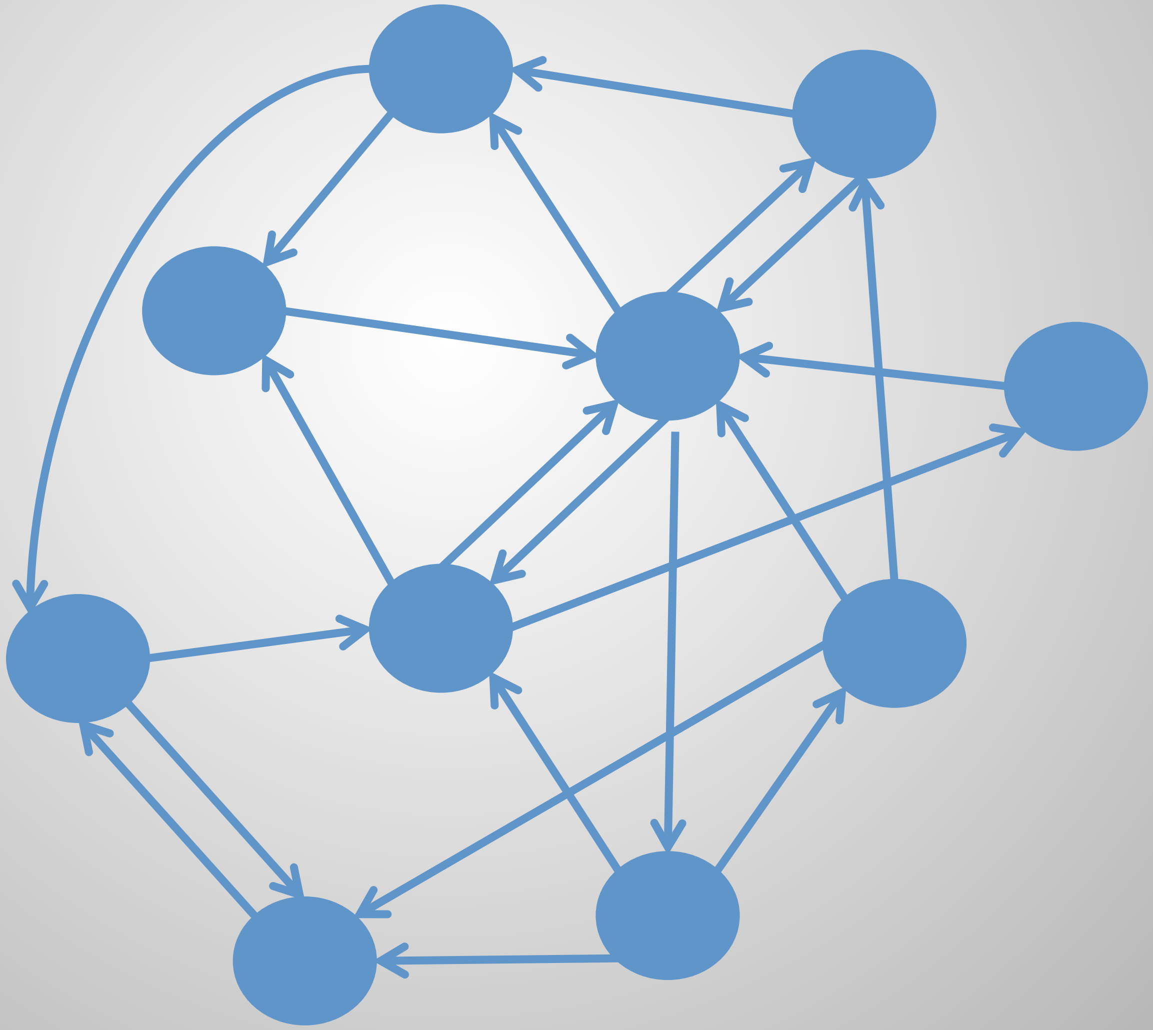
If the graph is strongly connected and aperiodic, this can be computed as the limit when k goes to ∞ of

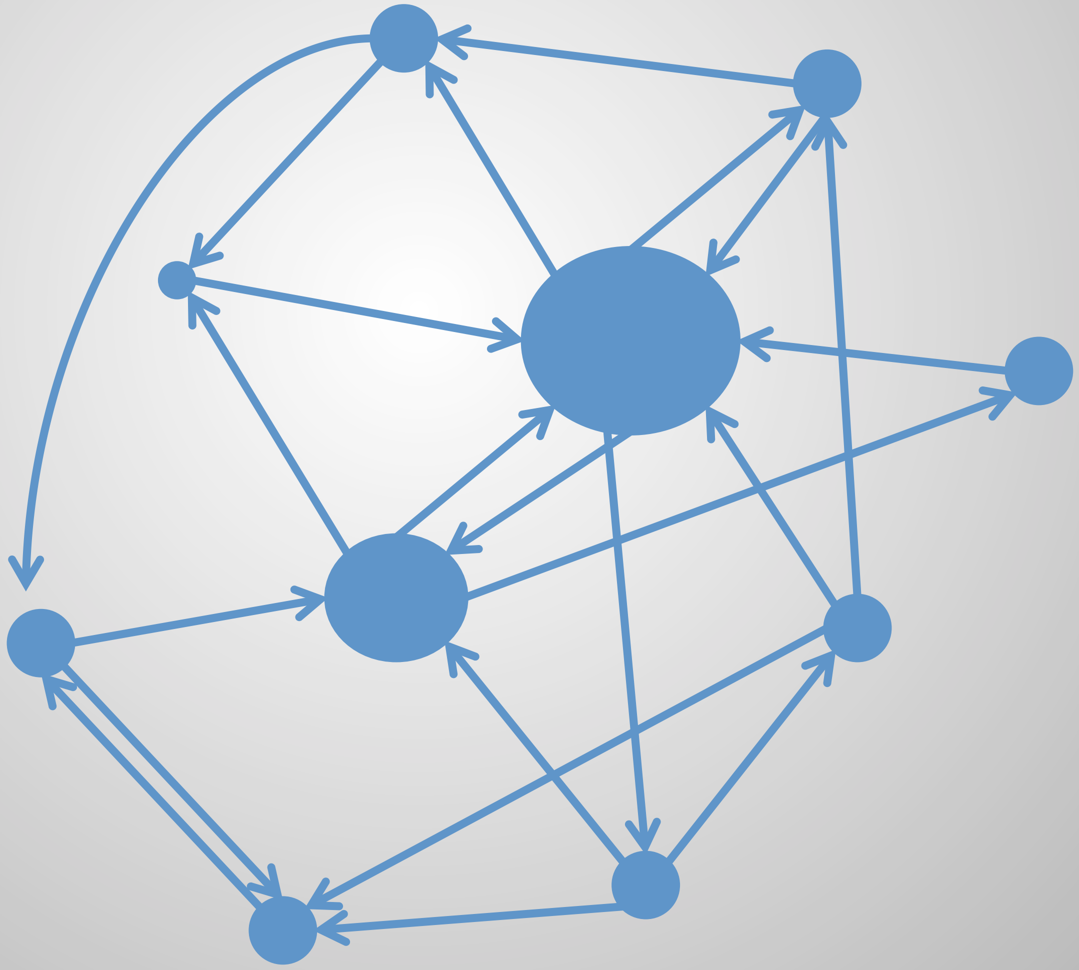
$$\mathbf{Pop_k = (G^T)^k \cdot Init}$$

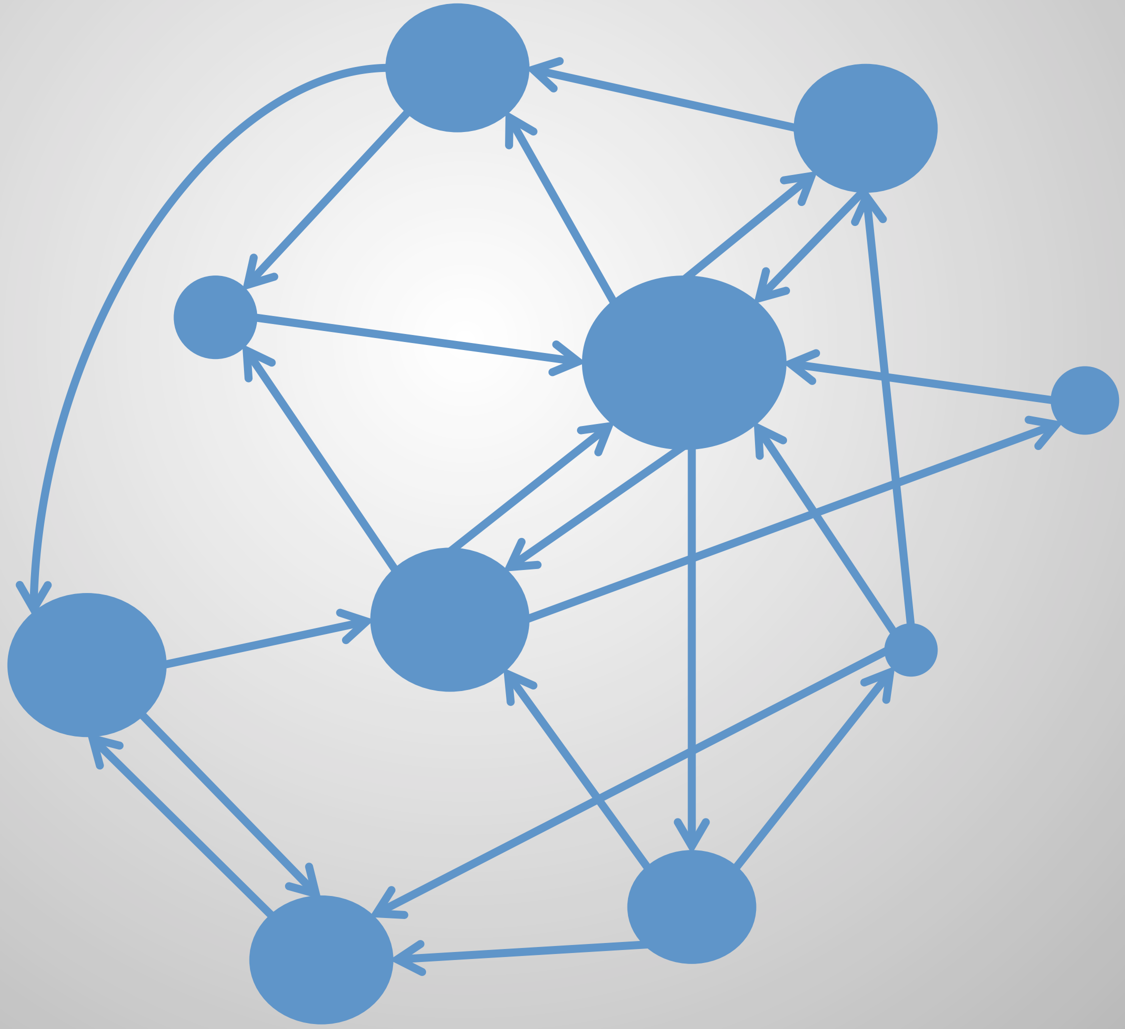
for some initial column vector Init, say

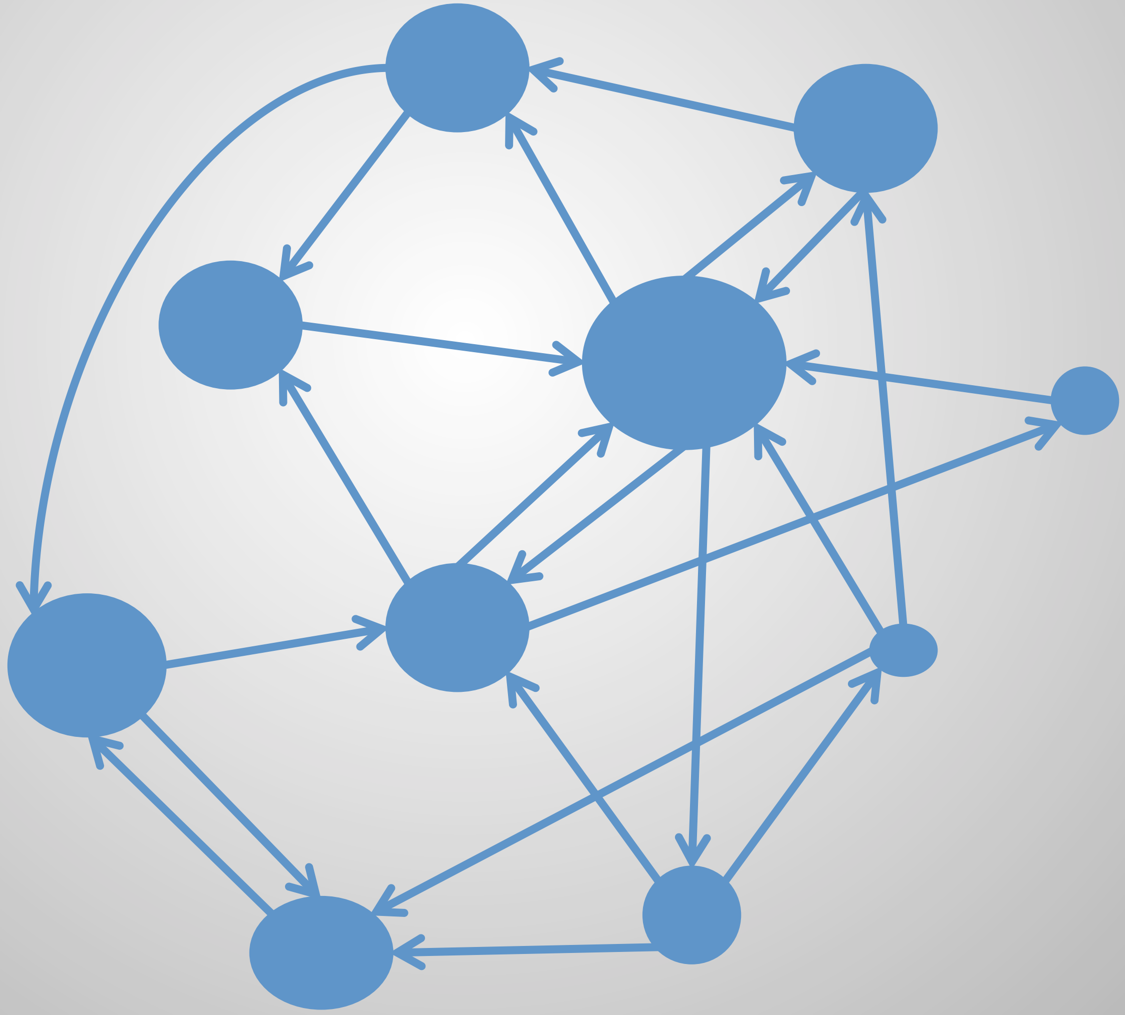
Init = [1/10, 1/10, 1/10, 1/10, 1/10, 1/10, 1/10, 1/10, 1/10, 1/10]

Then, Pop(i) is the probability that the surfer is in page i after an infinite time

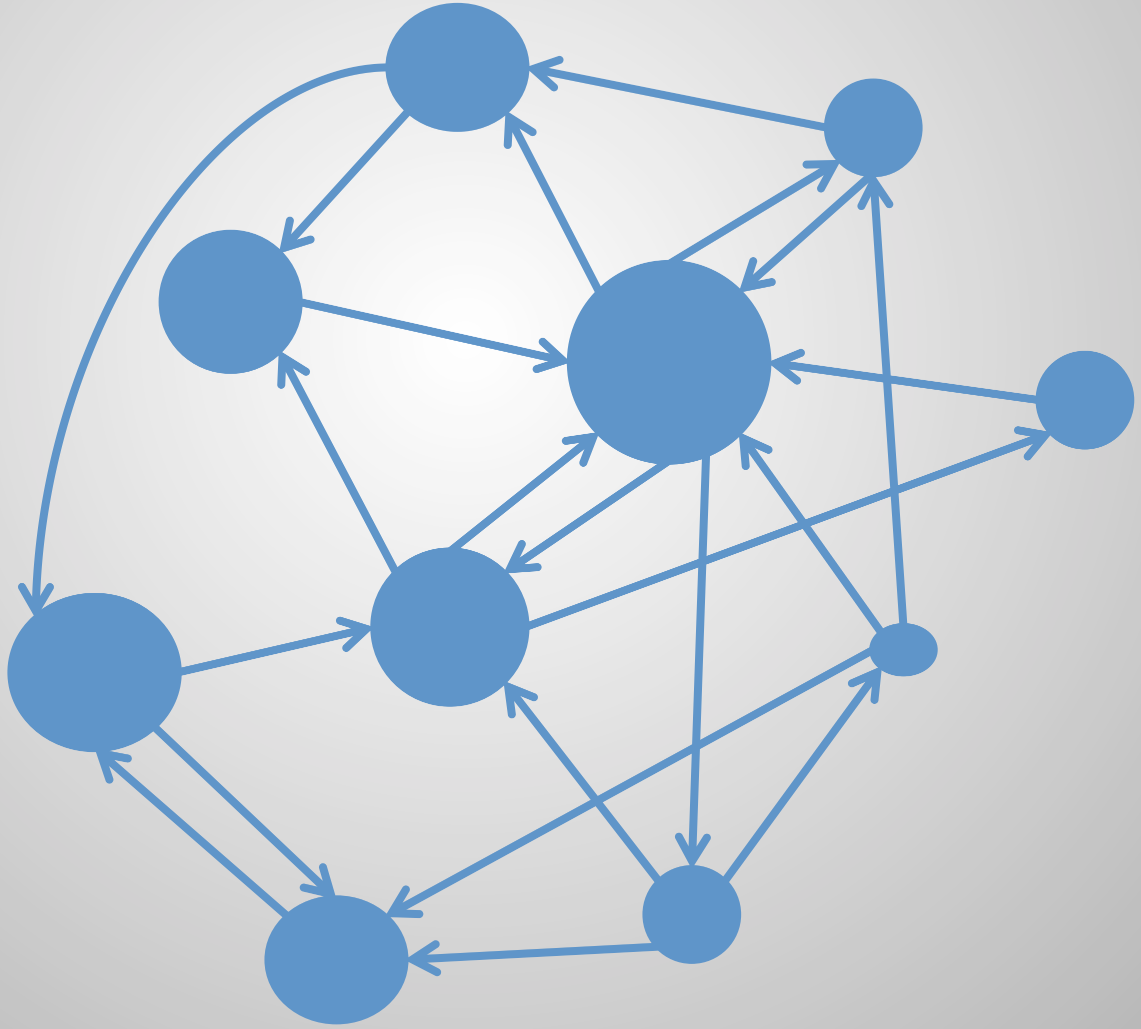








The fixpoint is reached



Issues

The graph is not strongly connected

- Introduce a **damping factor** to fix this “amortissement”
- Replace $\mathrm{Pop}_k = (G^T)^k \cdot \mathrm{Init}$

  By $\mathrm{Pop}_k = ((1-d)G^T + dU)^k \cdot \mathrm{Init}$

  where U is a matrix with 1
- Interpretation: add thin edges between all edges of the graph

Ranking pages in practice

Rank by weights based on several criteria

- E.g., weight(*t,d*) *= tfidf(t,d) × Pop(d)*
- Many more criteria in Google

Issue of the bad guys

Fighting the bad guys

Spamdexing the old fashion style

- Lying about the document to bring readers to a page
  - Include words that are not related to the page content
  - Typically make them invisible, e.g. blank on blank
- In meta-information such as
  - <meta name="Aretha Franklin Ray Charles Elvis Presley Sam Cooke John Lennon Marvin Gaye Bob Dylan Otis Redding Stevie Wonder James Brown">
- Countermeasure
  - Detect invisible text and ignore it
  - Ignore meta-information unrelated to the page content

Spamdexing a little old fashioned

Pollute existing websites (typically blogs, wikis) by adding links to the spamdexer website

Countermeasures

- Control who can write in the blog/wiki/etc.

  (require validation)
- Detect pollution and block it
- Disallow indexing

Spamdexing the modern way

- Use **link-farms** to artificially raise the popularity of your page
- Countermeasures:

  detect link-farms

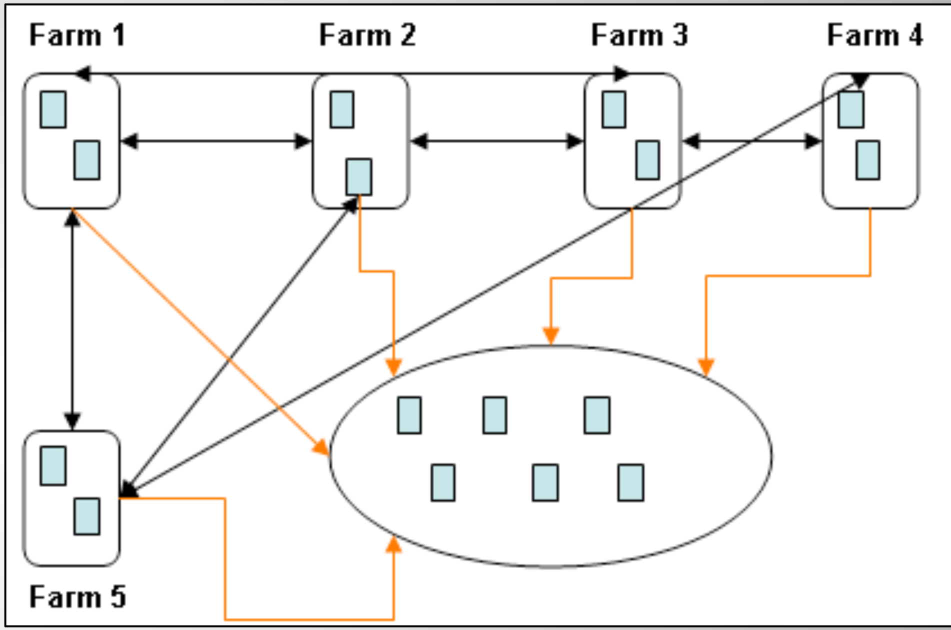
  – E.g. brutal PageRank variations

  – Bizarre content

  This is not easy

  – Use less PageRank

  – Alternative ranking such as TrustRank



Bombing

Try querying "**find Chuck Norris**" in Google, Exalead…

- The first result page contains

Google won't search for **Chuck Norris** because it knows you don't find **Chuck Norris**, he finds you.

Your search - **Chuck Norris** - did not match any documents.

Suggestions:

– Run, before he finds you.
– Try a different person.
– Try someone less dangerous.

- To do that: people created many pages pointing to that page with a text "find Chuck Norris"

Jewel: the OPIC Algorithm

Work with Gregory Cobena, Mihai Preda & Laurent Mignet

Used in the INRIA-Xyleme crawler with up to a billion URLs

On-line vs. off-line computation

Off-line algorithm

- Crawl the Web and build a Link-matrix
- Store the link matrix and update it – very expensive
- Start an off-line computation of popularity on a frozen link matrix

On-line algorithm: OPIC

for **On-line Page Importance Computation**

- The storage of the link matrix is not needed
- An estimate of popularity is computed while crawling and continuously updated

Static Graphs: OPIC

Start with a set of pages

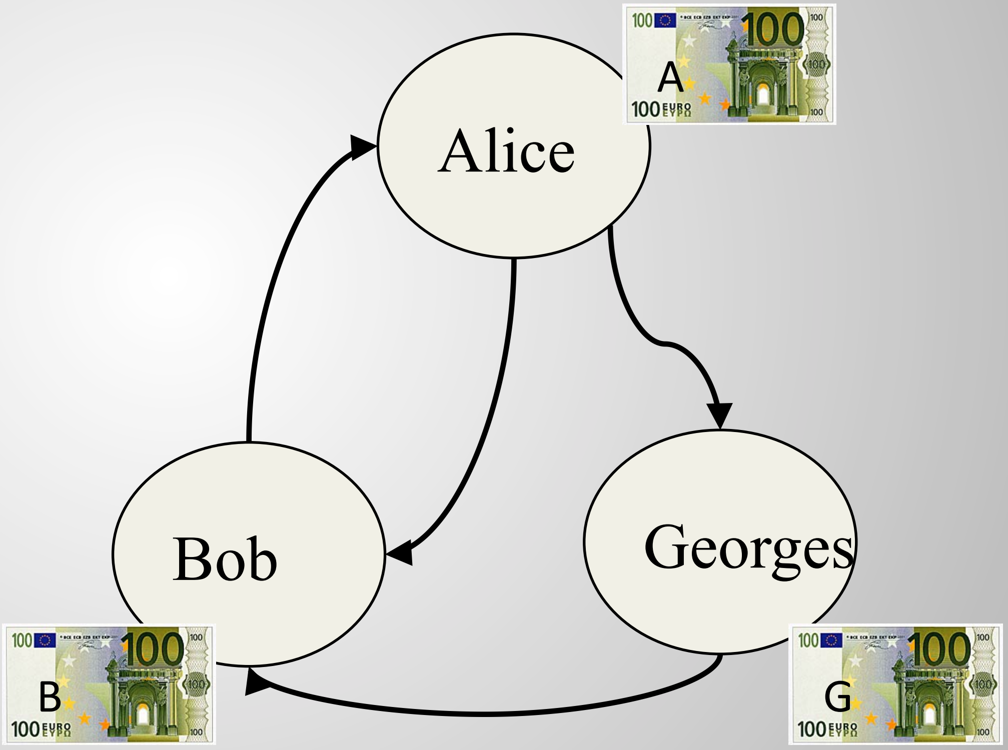
- Assign to each page a small amount of cash

Choose arbitrarily a page

- Read the page (it may have changed on the Web)
- Distribute its cash equally between its children
- If it has no child, distribute it equally between all pages

The total of cash in all pages does not change

Example: Small Web of 3 pages



OPIC-popularity

Definition: the **OPIC-popularity** of a page is the flow of cash in the page

$$\text{OPIC-Pop(page)} = \frac{\text{The sum of the cash a page received}}{\text{The sum of cash all the pages received}}$$

Thm: **OPIC-popularity is the same as PageRank-popularity (at the limit)**

Issue: speed of convergence

Advantages

1. Requires less resources: no need to store the Web graph
2. Computes popularity of newly discovered pages while crawling
   - Use it to decide what to crawl: the most popular ones
3. Continuously **adapt to changes** (new pages, page updates)
   - Consider only the flow in a page "recently" (time window)

Experiments (2)

Crawler

- Up to 100 robots running simultaneously on a single PC
- Average of 50 pages/seconds on an (old)PC (4 millions/day)
- Limiting factor was the number of random disk access

Performance and Politeness

- Pages were grouped by domain to minimize the cost of DNS (Domain Name Server) resolution
- To avoid rapid firing, we maintained a large number of accessible sites in memory (1 million domains).

Knowledge about visited pages: 100 million pages in main memory

- For each page, the exact disk location of the info structure (4 bytes) + a counter that we use for page rank and for the crawling strategy
- One disk access per page read

Conclusion

Web search engine

- The **crawler** to visit the Web
- The **index** with heavy use of parallelism
- The **ranking** based on popularity and other measures

There is much more

Fagin threshold algorithm for combining AND results

- retrieving top-k results with minimal access to the lists for each keyword

Boolean queries And/Or/Not

Clustering answers

Searching images and in general multimedia

Search taking user preferences into account

Search taking time into account: e.g., alerts for new documents

Search taking space into account: e.g., based on user location

And beyond

Negative references

- In recent news: A service company ranked first with mostly negative references

Today's technology is simplistic

Input: some keywords

Output: pages that contain these word

☛ Richer syntax

☛ Precise answers

Move to more semantics and structured data

- Semantic Web: already discussed
- Hidden Web
  - part of Web content that lies in online databases
  - typically queried through HTML forms
  - not usually accessible by following hyperlinks
  - Huge

Non technical issues

Search engine are perhaps becoming less essential

- Social networks: opinions and recommendations

Search engine are perhaps too powerful

- Almost monopoly by a few players
- Enormous power: you disappear if you are not high on the list
- The secrecy over the page ranking functions of search engine is questionable
- P2P search engines?

References

- Sergey Brin and Lawrence Page, « *The anatomy of a large-scale hypertextual Web search engine*. » *Computer Networks*, 1998.

- Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset et Pierre Senellart, « *Web data management* », *Cambridge University Press,* 2011 : www.webdam.inria.fr/Jorge
- Serge Abiteboul, Mihai Preda et Grégory Cobena, « *Adaptive On-Line Page Importance Computation* », *WWW Conf. 2003.*
- Alban Galland, Serge Abiteboul, Amélie Marian et Pierre Senellart, « *Corroborating information from disagreeing views* », *Conf. WSDM 2010.*

Merci !
FRAPAR.

Tova Milo

- Full professor at Tel Aviv university
- Department head
- Research
  - advanced database applications (data integration, semi-structured information…)
  - Web-based applications and Business Processes
- Program Chair of PODS, ICDT, VLDB…
- Member of the VLDB Endowment and the ICDT executive board
- 2010 ACM PODS Test-of-Time Award
- ERC Advanced Investigator grant