Effective Page Refresh Policies for Web Crawlers

and a Semantic Web Document Ranking Model

Roger-Alekos Berkley
IMSE 2012/2014

For CS561 Web Data Management
Spring 2013
University of Crete

Paper 1: Main Focus

"Effective Page Refresh Policies for Web Crawlers".

-Junghoo Cho, University of California, Los Angeles.
-Hector Garcia-Molina, Stanford University.

- How to maintain local copies of remote data sources "fresh" when the data source is updated autonomously and independently.
- Special attention is given to the problem of Web crawlers that maintain local copies of remote Web pages for Web search engines.
- This paper proposes various page refresh policies and studies their effectiveness.

Paper 2: Main Focus

"Swoogle: A Search and Metadata Engine for the Semantic Web".

-Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, Joel Sachs, University of Maryland Baltimore County.

- Rational random surfer ranking model for Semantic Web Documents (in comparison with Google PageRank).

Web Crawler (1)

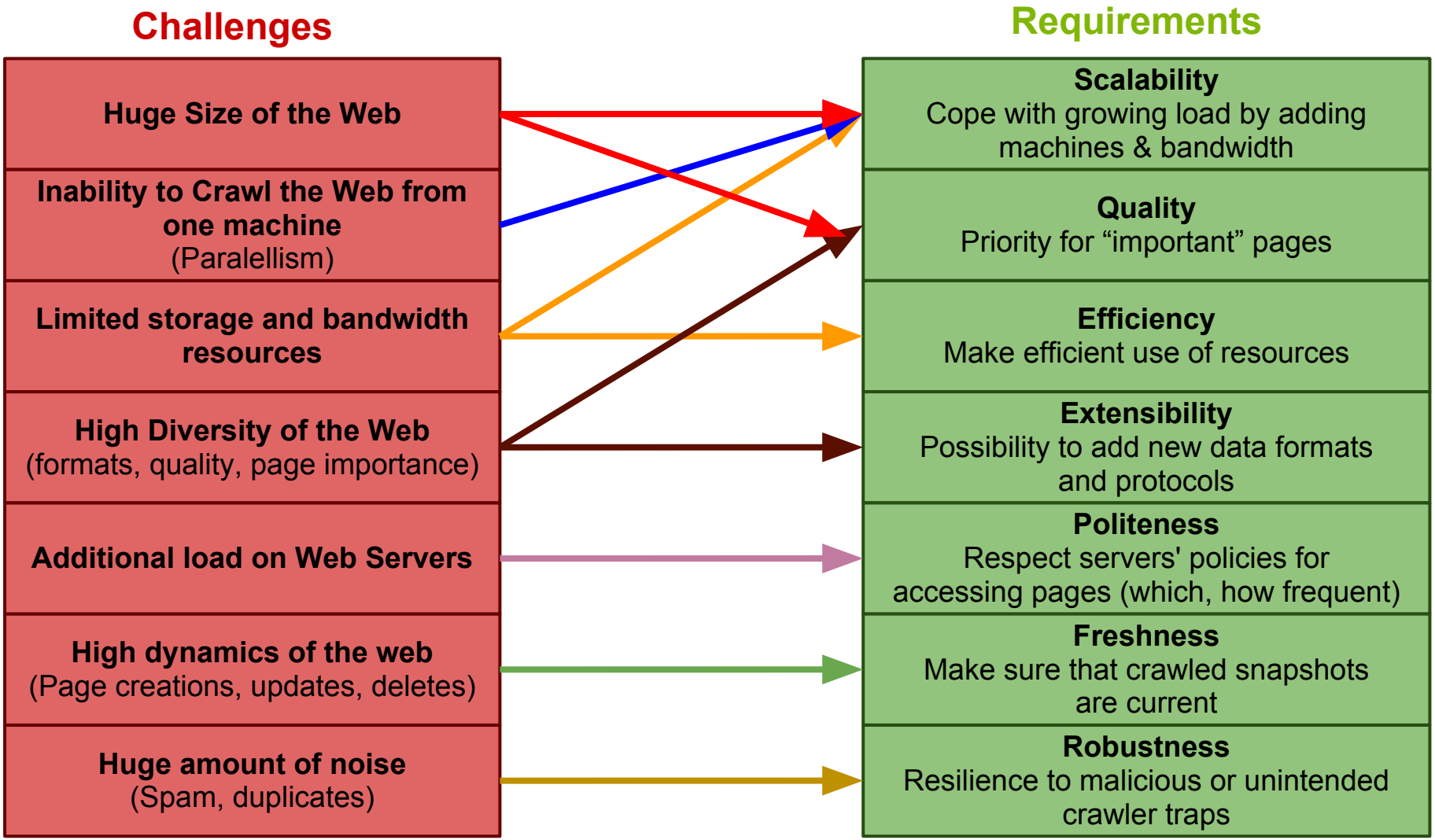
What is a web crawler?
A software system for the bulk and automatic downloading of web pages.

What is it used for?

- Search Engines (main component).
- Mirroring/Web Archiving.
- Web Data Mining.
- Web Monitoring.

Reference: 1,2.

Web Crawler (2)



Reference: 3,4.

Web Crawler (3)

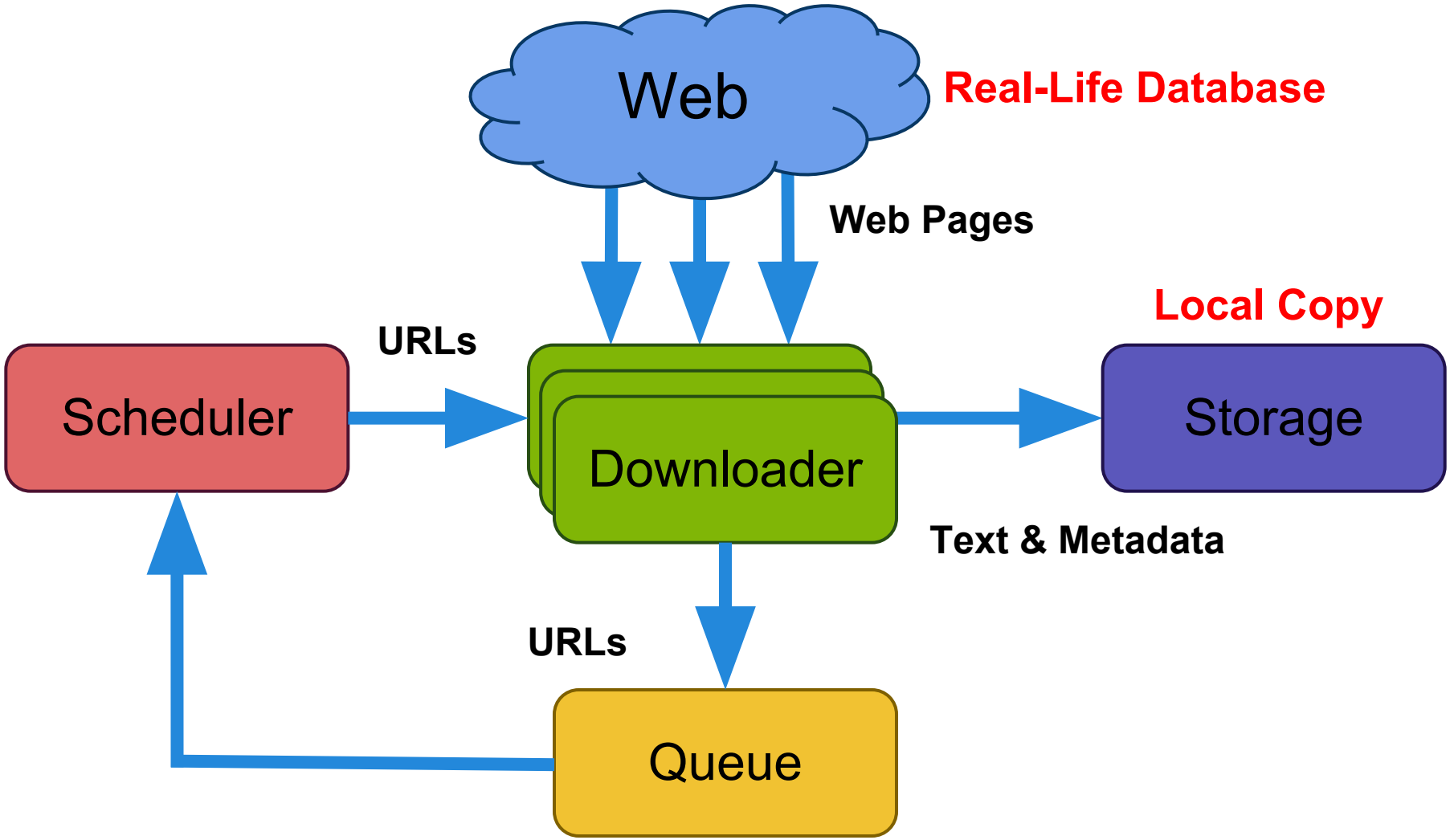
All types of crawlers usually aim to have coverage that is as complete as possible.

Types of Crawlers:

- **Universal Crawler**: supports universal search engines.
- **Focused Crawler**: collects web-pages according to a criteria (e.g. certain domain).
- **Topical Crawler**: collects web-pages about a specific topic (e.g. solar energy).
- **Batch-mode Crawler**: revisits a fixed collection of web-pages periodically.
- **Incremental Crawler**: continuously revisits known pages and increases crawl quality by finding new good pages.
- **Archive/Snapshot crawler**: gets at most a snapshot of a web page.
- **Mirroring/Caching System**: useful for “offline” access to web pages.

Reference: 3,4.

General Web Crawler Architecture



Reference: 1.

Paper 1

"Effective Page Refresh Policies for Web Crawlers".

-Junghoo Cho, University of California, Los Angeles.
-Hector Garcia-Molina, Stanford University.

1. Freshness and Age.
2. Evolution of real-world elements.
3. Evolution of real-world database.
4. Synchronization policies.
5. Weighted Freshness and Age.
6. Experiments.

1. Freshness and Age

- Data sources do not push changes to local copies.
- Local copies get out of date.
- Local copies have to periodically "poll" data sources to keep the copy "fresh".

Freshness (F): the fraction of the local copy that is up-to-date.

Freshness of a local element at time (t)

$$F(e_i;t) = \begin{cases} 1 & \text{if } e_i \text{ is up-to-date at time } t \\ 0 & \text{otherwise.} \end{cases}$$

Freshness of a local database at time (t)

$$F(S;t) = \frac{1}{N}\sum_{i=1}^{N} F(e_i;t).$$

Age (A): measures "how old" the local copy is.

Age of a local element at time (t)

$$A(e_i;t) = \begin{cases} 0 & \text{if } e_i \text{ is up-to-date at time } t \\ t - t_m(e_i) & \text{otherwise.} \end{cases}$$

Age of a local database at time (t)

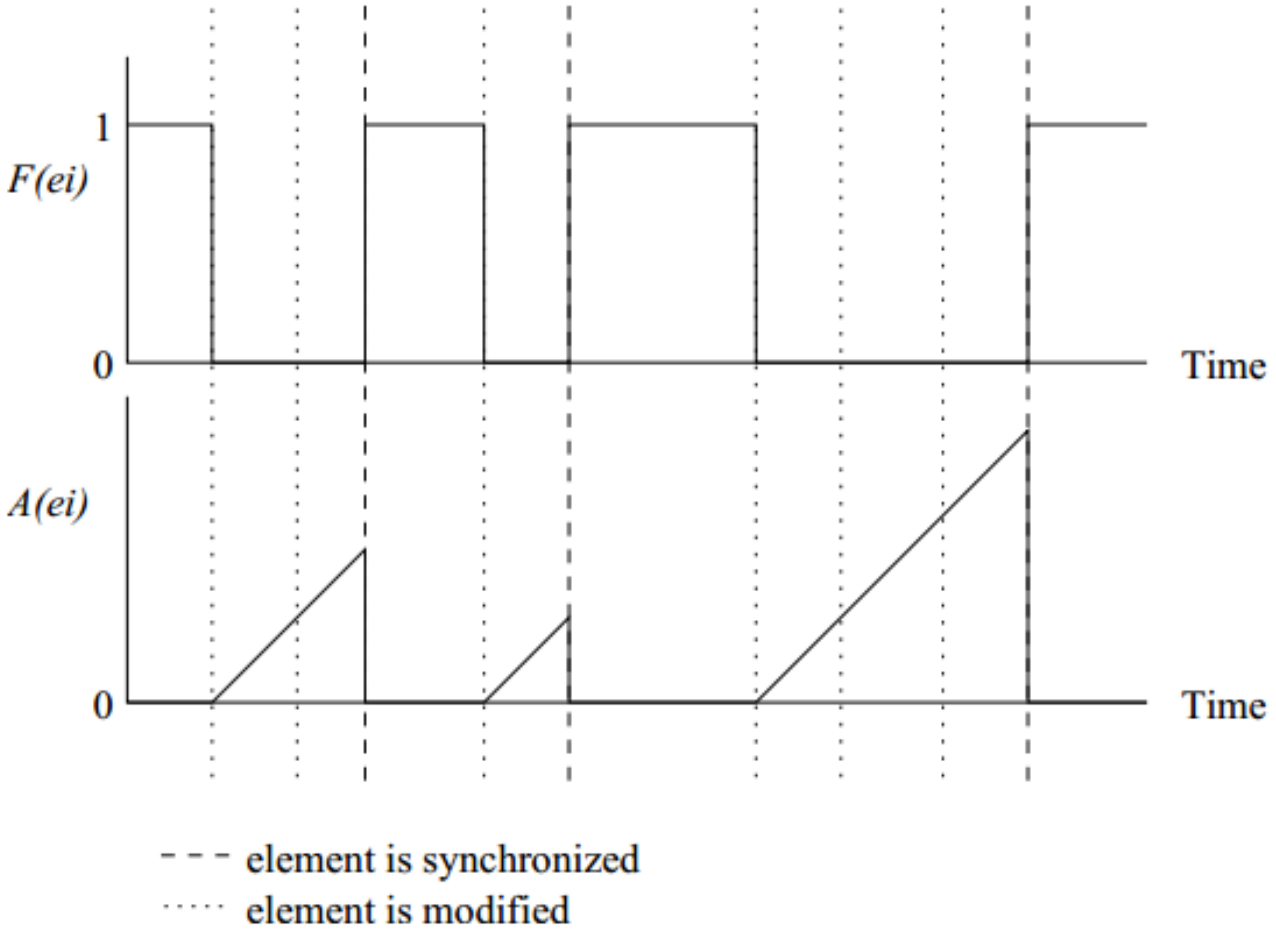
$$A(S;t) = \frac{1}{N}\sum_{i=1}^{N} A(e_i;t).$$

tm(e1) is the time of the first modification of element (e1) after its last synchronization.

Reference: 5.

1.a Freshness and Age (Evolution)

Time evolution for Freshness and Age of an element:



Reference: 5.

1.b Freshness and Age (Examples)

Example 1
Database (**S**) with 100 elements, **S**= {e1, e2, e3.... e100}.
At time (**t**), 65 elements are out-of-date and 35 elements fresh.

F(S,t) = 1/100(1*35) = **0.35** → (1: database fully fresh)
(0: database fully out-of-date)

Example 2
Database with 3 elements, **S**= {e1, e2, e3}.
At time (**t**), all 3 elements out-of-date.
F(S,t) = 0

A(e1) = t - tm(e1) = 3 hours.
A(e2) = t - tm(e2) = 7 hours.
A(e3) = t - tm(e3) = 10 hours.

A(S,t) = (3+7+10)/3 = **6.66** hours.

Reference: 5.

2. Evolution of Real-World Elements

- Freshness is hard to measure exactly in practice since we need to instantaneously compare the real-world data to the local copy.
- But it is possible to estimate freshness given some information about how the real-world data changes.
- A **Poisson Process** is often used to model events that happen *randomly* and *independently* with a fixed rate of change (**λ**) over time.
- In this paper, it is assumed that the Real-World elements are modified by a Poisson Process.

Reference: 5.

3. Evolution of Real-World Database

We can model the change of the **real-world** database as a whole by one of the following:

1) Uniform change frequency model

All real-world database elements change at the same rate (λ).

2) Non-uniform change frequency model

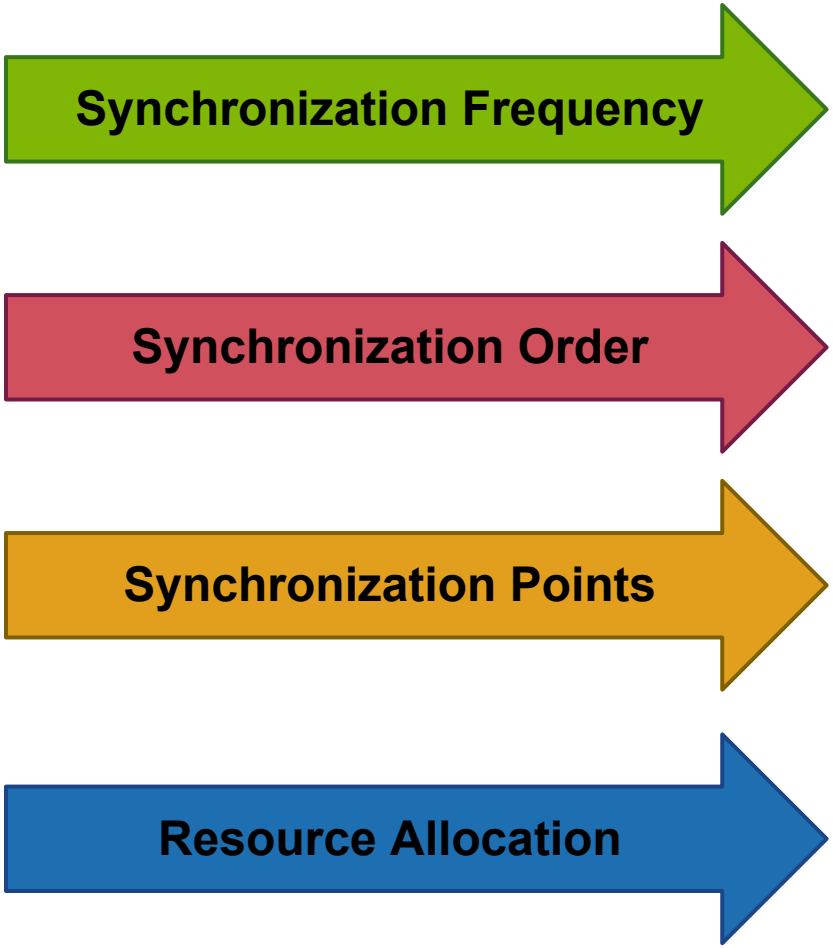
Elements change at different rates.

In this paper: each element in the real-world database (web page) is assumed to change independently and **at its own rate**.

one element may change once a day, and another element may change once a year.

Reference: 5.

4. Synchronization Policies Dimensions



Reference: 5.

4.a. Synchronization Frequency

Parameters to be decided:

$\mathcal{N}$: number of elements to be synchronized per time unit.

I: time unit.

By changing "I" we adjust how often the elements are synchronized.

$f = 1/I$ represents the average frequency at which an element in the database is synchronized.

Reference: 5.

4.b. Synchronization Order Policies

Example: we maintain a local database of 10,000 Web pages from a certain website. We keep our copy updated by revisiting the pages in the site. In performing the crawl, we may adopt one of the following options:

Fixed synchronization order policy

1. We maintain a list of all 10,000 URLs and we visit the URLs repeatedly in the same order.

Random synchronization order policy

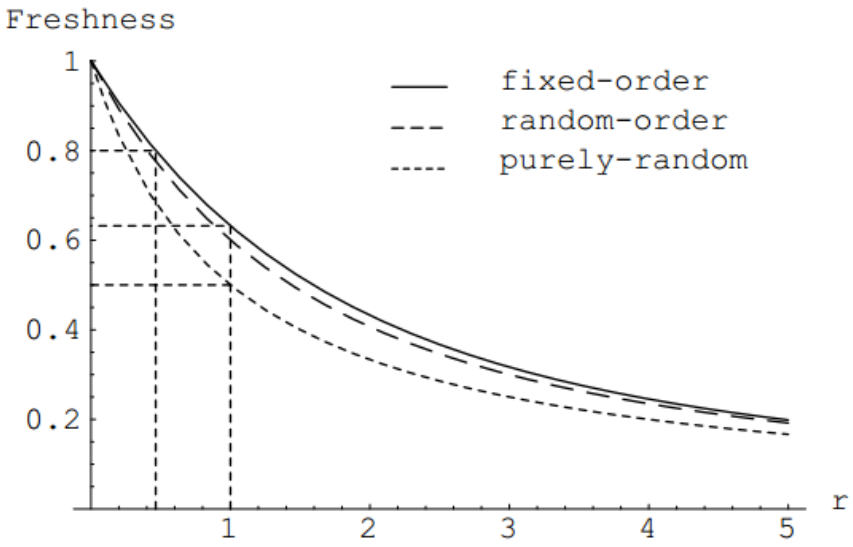
2. We only maintain the URL of the root page of the site, and whenever we crawl the site, we start from the root page, following links. The link structure and order at a particular crawl determines the page visit order, the synchronization order may change from one crawl to the next.

Purely random synchronization order policy

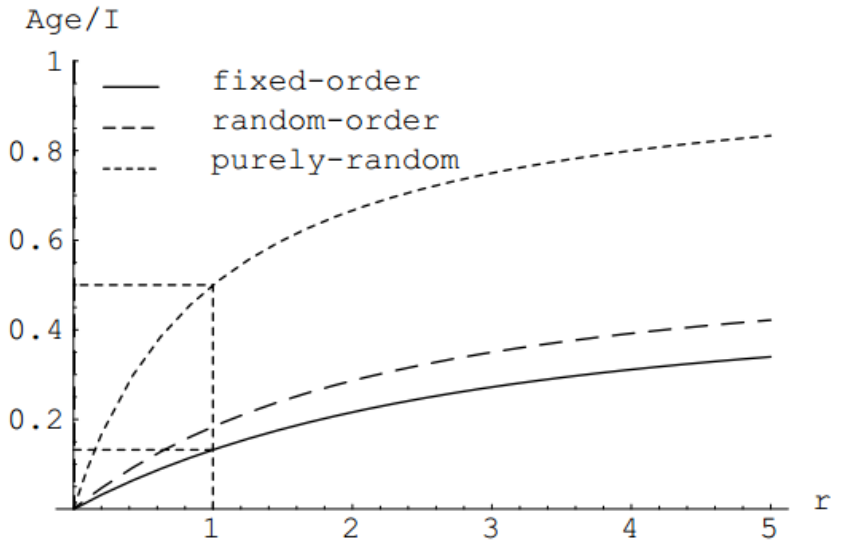
3. At each synchronization point, we select a random element from the database and synchronize it.

Reference: 5.

4.b. Comparison of Sync. Order Policies



(a) Freshness graph over $r = \lambda/f$



(b) Age graph over $r = \lambda/f$

r = λ/f

λ = Rate at which a real-world element changes.

f = Frequency at which a local element is synchronized.

When r < 1, we synchronize the elements more often than they change.

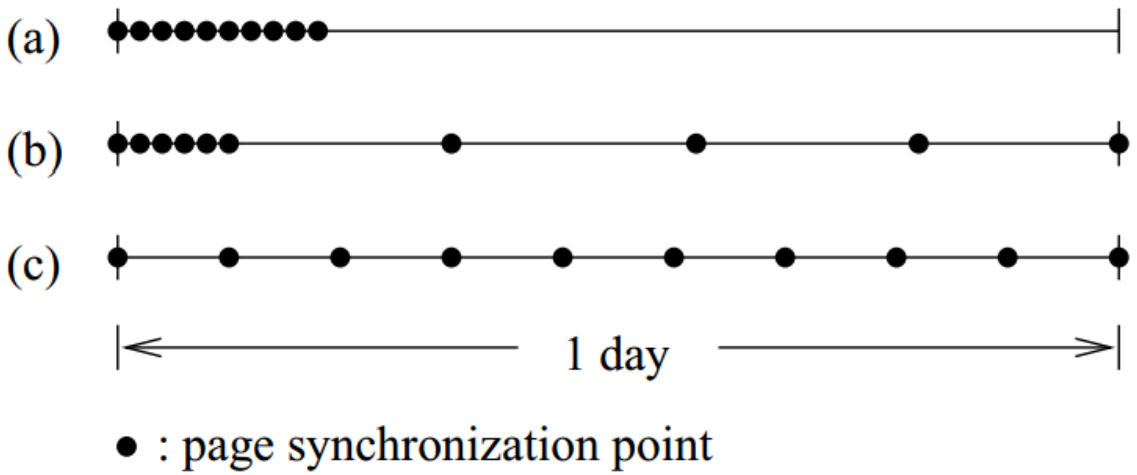
When r > 1, the elements change more often than we synchronize them.

Reference: 5.

4.c. Synchronization Points

In some cases, we may need to synchronize the database only in a limited time window.

Example: we maintain a local database of **10** pages from a certain website. The site is heavily accessed during daytime. We consider several synchronization policies:



(a)	(b)	(c)
Sync. all 10 pages at the beginning of the day.	Sync. most pages at the beginning of the day, some pages during the rest of the day.	Sync. 10 pages uniformly over a day.

Reference: 5.

4.d. Resource Allocation Policies

Example
Database of 3 elements {e1, e2, e3}.

- **e1** changes at the rate λ1=4 (times/day).
- **e2** changes at the rate λ2=3.
- **e3** changes at the rate λ3=2.

Option 1
Uniform allocation policy

Option 2
Non-uniform allocation policy

We decided to synchronize the **database** at the total of **9** elements/day.

In deciding how frequently to synchronize **each element**, we have **2 options**:

1. **Synchronize all elements uniformly at the same rate.**
   Meaning: Sync. (e1, e2, e3) at the same rate of 3 times a day.

OR

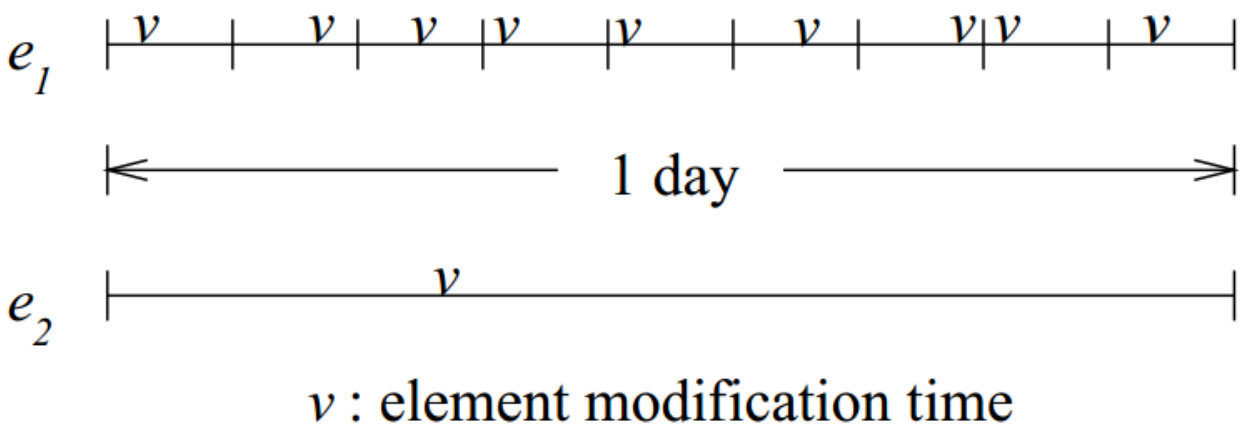
2. **Synchronize an element proportionally more often if it changes more often.**
   Meaning: Sync. e1 at (frequency) f1=4, e2 at f2=3, e3 at f3=2.

Reference: 5.

4.d. Superiority of the Uniform-allocation policy (1/3)

Two-Element Database Example

S= {e1, e2}
f(e1) = 9 times a day.
f(e2) = 1 time a day.
Goal: maximize the freshness of the database.



e1 changes once and only once in each interval, we do not know exactly when it changes within this interval.
e2 changes once and only once a day, we do not know exactly when it changes within this day.

Reference: 5.

4.d. Superiority of the Uniform-allocation policy (2/3)

- Let's assume that we can only synchronize **one** element per day.
- **Question**: which element should we synchronize?

Item	e1	e2
Frequency of change	9 times/day	1 time/day
Point of Change	After/before middle of the period	After/before the middle of the day
Benefit	1/18 (of a day)	1/2 (of a day)
Probability of either point	50%	50%
Expected Benefit	**1/36 (of a day)**	**1/4 (of a day)**

Answer: e2 results in higher expected benefit.

Reference: 5.

4.d. Superiority of the Uniform-allocation policy (3/3)

Using the same logic, we calculate the expected benefit at different sync. levels:

Proportional Policy
- e1 changes 9 times, gets synced 9 times.
- e2 changes once, gets synced once.

Uniform Policy
- No matter how much the elements change, they get the same amount of sync.

Row	Total Sync. Frequency	Allocation f1	Allocation f2	Expected Benefit	Best	
(a)	1	1	0	$\frac{1}{2} \times \frac{1}{18} = \frac{1}{36}$	0	1
(b)		0	1	$\frac{1}{2} \times \frac{1}{2} = \frac{9}{36}$		
(c)	2	2	0	$\frac{1}{2} \times \frac{1}{18} + \frac{1}{2} \times \frac{1}{18} = \frac{2}{36}$	0	2
(d)		1	1	$\frac{1}{2} \times \frac{1}{18} + \frac{1}{2} \times \frac{1}{2} = \frac{10}{36}$		
(e)		0	2	$\frac{1}{3} \times \frac{2}{3} + \frac{1}{3} \times \frac{1}{3} = \frac{12}{36}$		
(f)	5	3	2	$\frac{3}{36} + \frac{12}{36} = \frac{30}{72}$	2	3
(g)		2	3	$\frac{2}{36} + \frac{6}{16} = \frac{31}{72}$		
(h)	10	9	1	$\frac{9}{36} + \frac{1}{4} = \frac{36}{72}$	7	3
(i)		7	3	$\frac{7}{36} + \frac{6}{16} = \frac{41}{72}$		
(j)		5	5	$\frac{5}{36} + \frac{15}{36} = \frac{40}{72}$		

Rows (a) to (e): when the synchronization frequency is smaller than the change frequency, it's better to give up syncing the elements that change too fast (and focus on what we can track).
Rows (h) to (j): Even if the sync. frequency is larger, the uniform policy is more effective. And, the optimal policy is somewhere in between the proportional and the uniform policies.

Reference: 5.

4.d. Optimal Resource-Allocation Policy

Example: 5-element real-world database.

	e_1	e_2	e_3	e_4	e_5
(a) change frequency	1	2	3	4	5
(b) synchronization frequency (freshness)	1.15	1.36	1.35	1.14	0.00
(c) synchronization frequency (age)	0.84	0.97	1.03	1.07	1.09

(b) and (c) are the optimal synchronization frequencies for these elements, based on their frequency of change.

Outcome

1. To improve *freshness*, we should penalize the elements that change too often.
2. To improve *age*, we should synchronize an element more often when it changes more often.

Optimal Policy

Notice the *marginal* differences between the synchronization frequencies?

- Optimal resource-allocation policy is rather closer to the uniform policy than the proportional policy.

Reference: 5.

5. Weighted Freshness and weighted age

So far, we assumed that the *freshness* and the *age* of every element is equally important to the users. But what if the elements have different "importance"?

Element Importance = Traffic

When the users access element *e1* **twice more often** than *e2*, we consider e1 twice as important as e2. By maintaining *e1* up-to-date we can make the users see fresh elements **twice** as often.

wi = weight (importance) of an element.

$$F(S;t) = \left(\sum_{i=1}^{N} w_i \, F(e_i;t)\right) \Big/ \left(\sum_{i=1}^{N} w_i\right)$$

(1) Weighted Freshness

$$A(S;t) = \left(\sum_{i=1}^{N} w_i \, A(e_i;t)\right) \Big/ \left(\sum_{i=1}^{N} w_i\right)$$

(2) Weighted Age

Reference: 5.

5. Weighted Freshness and weighted age

Example: We maintain **6** elements in the local database, their **weights** and change frequencies are different. Assuming that we synchronize **6** elements per day.

	e_{11}	e_{12}	e_{13}	e_{21}	e_{22}	e_{23}
(a) weight (w_i)	1			2		
(b) change frequency (times/day)	1	2	3	1	2	3
(c) synchronization frequency (freshness)	0.78	0.76	0.00	1.28	1.56	1.62
(d) synchronization frequency (age)	0.76	0.88	0.94	0.99	1.17	1.26

(c) and (d) are the optimal synchronization frequencies for these elements, based on their weight and frequency of change.

Outcome

1. We can see that we should visit an element more often when its weight is higher. *Such as: (e11 and e21), (e12 and e22).*

2. But, not necessarily **proportionally** more often. *Such as: (e13 and e23).*

Reference: 5.

6. Experimental Setup

To collect data on how often web pages change (real-life database):

- Crawled 720,000 pages from 270 "popular" websites.
- For a period of 4 months.
- Using Stanford WebBase crawler.
- Indexing speed 60 pages per second.
- Storage 210GB of HTML.

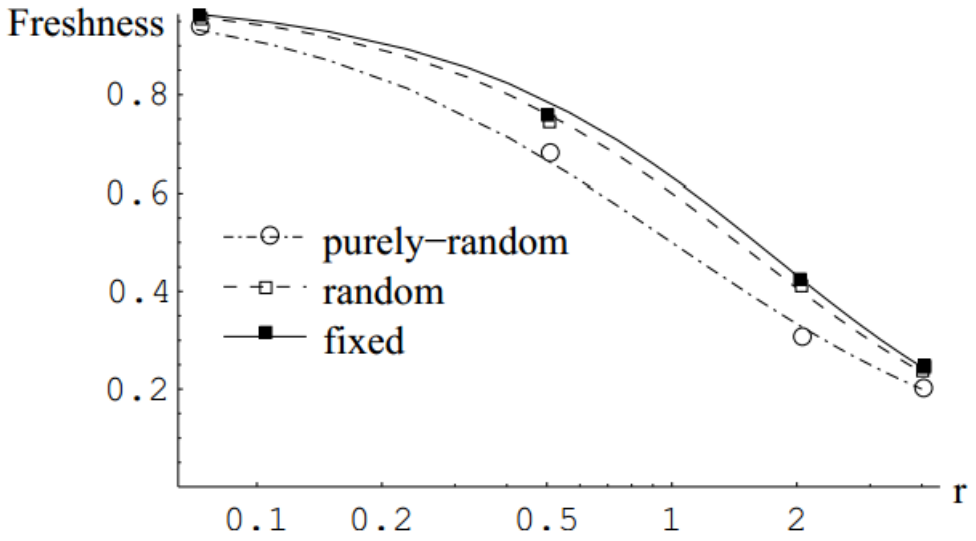
To select sites for the local copy:

- Used a snapshot of the web available from Stanford WebBase crawler.
- Selected 400 most popular websites and contacted them.
- 270 websites agreed to be experimented.
- Starting at the root URL, 3000 pages were crawled daily from each website.

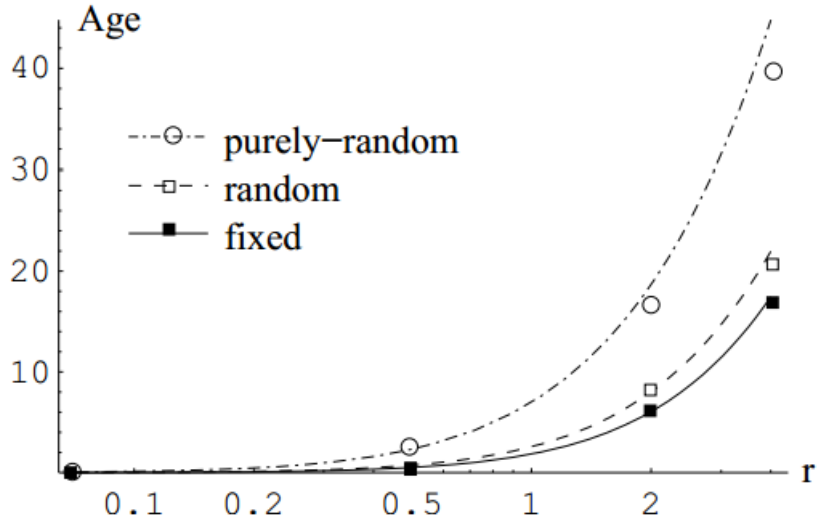
Reference: 5.

6. Experimental Results (1)

Verification of the comparison of the **synchronization-order** policies.



(a) Freshness metric



(b) Age metric

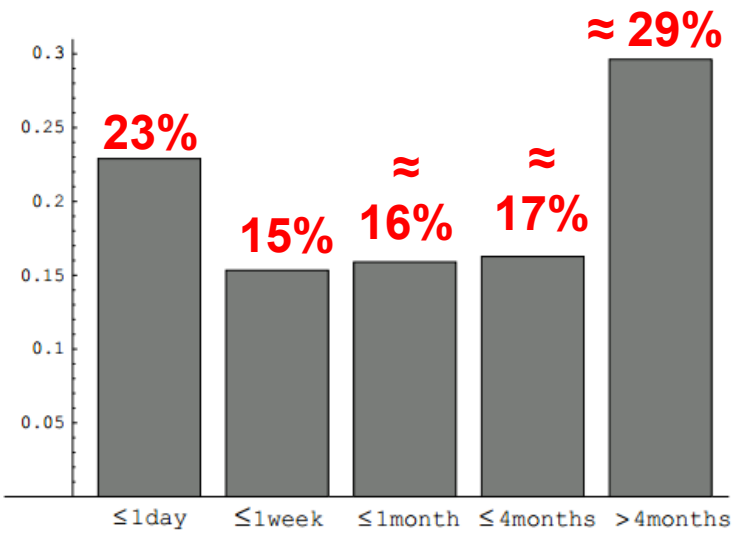
r: change frequency rate for crawls. (r=2, once a month).
Lines: predictions by theoretical analysis.
Points: experimental results.

Reference: 5.

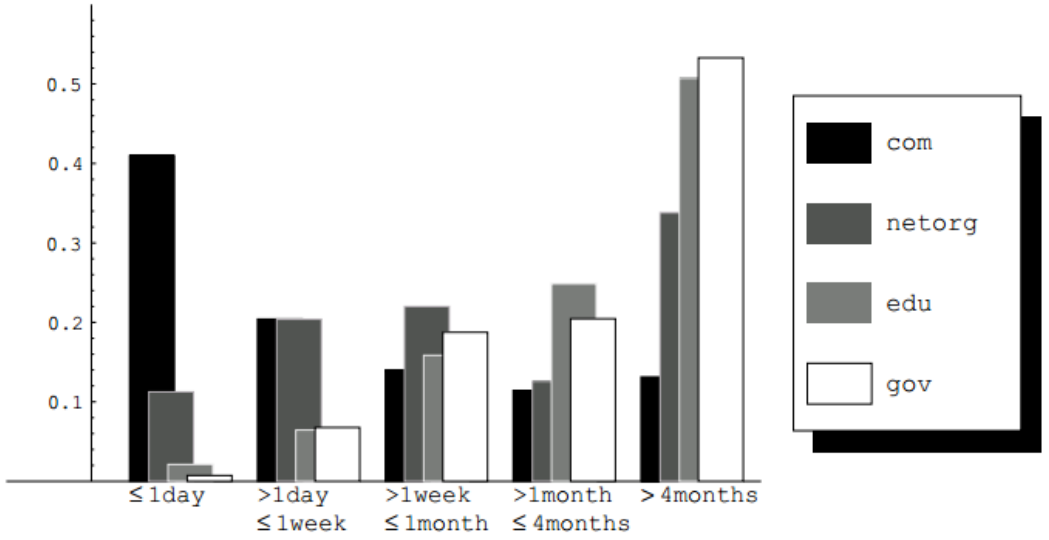
6. Experimental Results (2)

Study of the **frequency of change** of the experimented web pages.

(important for the next step of evaluating the resource-allocation policies).



(a) Over all domains



(b) For each domain

Horizontal: *Average Change Interval.* (total monitoring period divided by number of detected changes).
Vertical: fraction of pages.

Reference: 5.

6. Experimental Results (3)

Evaluation of the three **resource-allocation** policies**.**

	overall		com		gov	
	Freshness	Age	Freshness	Age	Freshness	Age
Proportional	0.12	400 days	0.07	386 days	0.69	19.3 days
Uniform	0.57	5.6 days	0.38	8.5 days	0.82	2.0 days
Optimal	0.62	4.3 days	0.44	7.4 days	0.85	1.3 days

Freshness and Age prediction of a local copy based on real-web data
Fixed-Order policy adopted (best).

Outcome

1. Optimal policy is significantly better than the two other policies.
2. Uniform policy is significantly better than the Proportional policy.

Facts

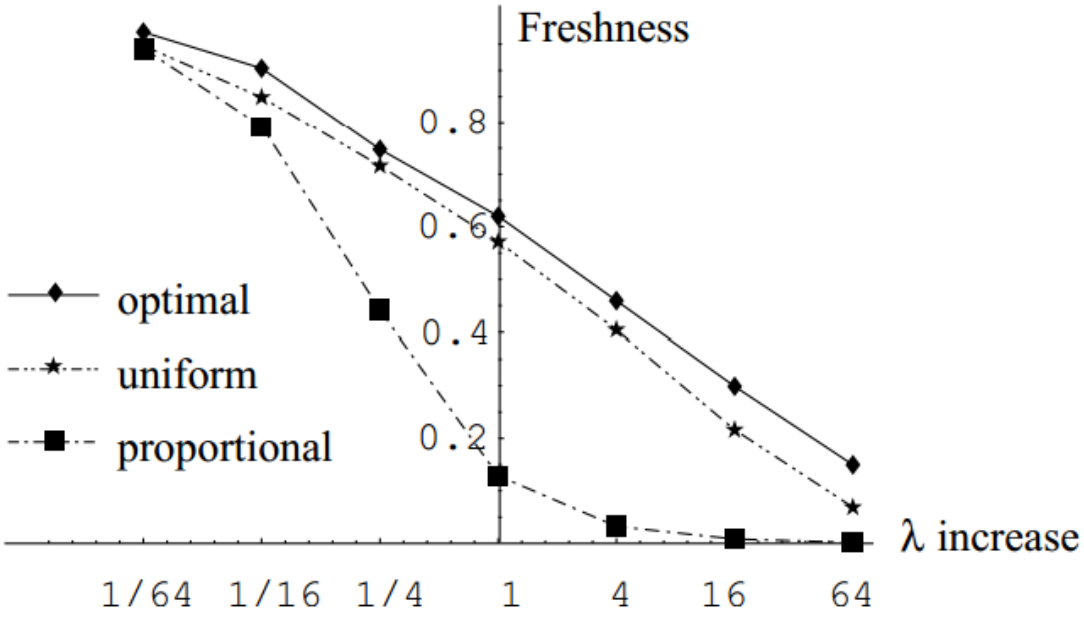
- In a case, *Freshness* increased 500% by using the *Optimal policy* instead of the *Proportional policy*.
- Age decreases by 23% from the *Uniform policy* to the *Optimal policy*.
- Age of the *Proportional policy* is 93 times worse than the *Optimal policy*.

Reference: 5.

6. Experimental Results (4)

Optimal policy is more useful when pages change more often.

Estimated Freshness for different Average Change Frequencies



Facts

- For the (.gov) domain, which has fewer frequently changing pages, the freshness difference is 18% between the proportional and optimal policies.
- Difference becomes 84% for the (.com) domain.
- Same kind of improvement for Age, which increases from 1380% improvement for the (.gov) to 5110% for the (.com).

Reference: 5.

Paper 2

"Swoogle: A Search and Metadata Engine for the Semantic Web".

-Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, Joel Sachs, University of Maryland Baltimore County.

1. Google PageRank.
2. Rational random surfer ranking model for Semantic Web Documents.

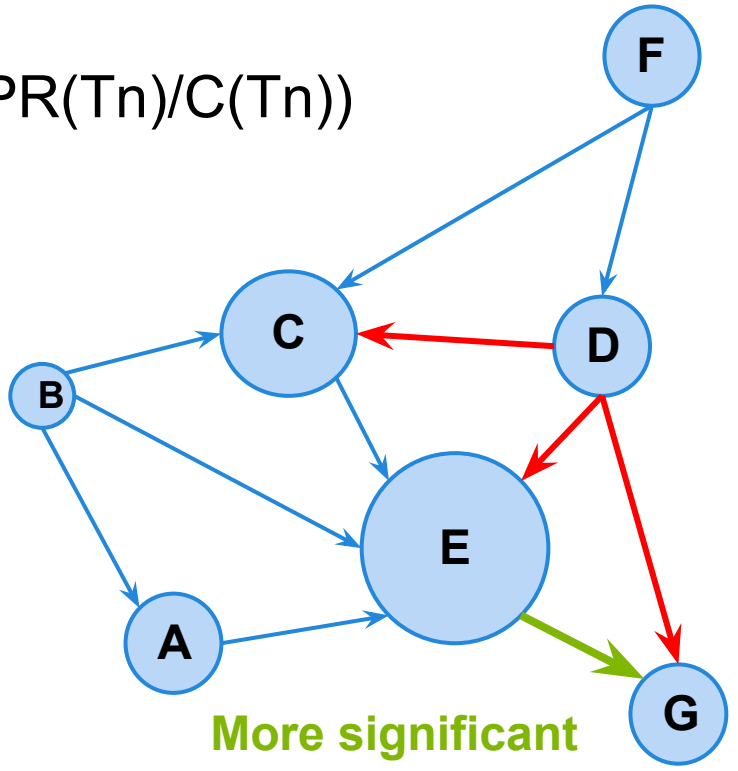
Google PageRank

Random Surfing Model

- More incoming links = Higher PageRank.
- Higher incoming link quality (vote) = Higher PageRank.
- More outgoing links = lower outgoing vote.
- Damping factor = maximizes the amount of outgoing votes (Decay).

$$PR(A) = (1-d) + d\,(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$$

PR(A): PageRank of page A.
d = damping factor, set to 0.85.
T1: a page that points to page A.
Tn: another page that points to page A.
C(T1): number of links going out of Page T1.



Reference: 6.

Swoogle Ranking

The nature of hyperlinks leads to a non-uniform probability of following a particular outgoing link. This is not appropriate for the Semantic Web.

Rational Random Surfing Model: accounts for the various types of links that can exist between SWDs. Types of links:

1. *Imports (A,B):* A imports all content of B.
2. *Uses-term (A,B):* A uses some of terms defined by B, without importing B.
3. *Extends (A,B):* A extends the definitions of terms defined by B.
4. *Asserts (A,B):* A makes assertions about the individuals defined by B.

These relationships account for the ranking and carry different weights.

For instance, when a surfer imports (A,B) while visiting A, it is natural for it to follow this link because B is semantically part of A.

Similarly, the surfer may follow the extends(A,B) relation because it can understand the defined term completely only when it browses both A and B.

Reference: 7.

Bibliography

1. *Web Crawling*. Christopher Olston, Marc Najork. Foundations and Trends in Information Retrieval. 2010. http://homepages.dcc.ufmg.br/~nivio/cursos/ri12/transp/olston-najork@web-crawling10.pdf
2. *Web Crawling*. Universitat Pompeu Fabra, Barcelona, Spain. Department of Technologies, Web Research Group. 2006. http://grupoweb.upf.es/WRG/course/slides/crawling.pdf
3. *Crawling the Web*. Jefferey D. Ullman (http://infolab.stanford.edu/~ullman/). InfoLab, Stanford University, California. (http://infolab.stanford.edu/~ullman/mining/pdf/crawl.pdf).
4. *Web Dynamics: Searching the Dynamic Web*. R. Schenkel, M. Spaniol. Max Planck Institut Informatik. Saarbrücken, Germany. 2009. (http://www.mpi-inf.mpg.de/departments/d5/teaching/ss09/dyn/slides/dyn-kap3.pdf)
5. *Effective Page Refresh Policies for Web Crawlers*. Junghoo Cho, UCLA. Hector Garcia-Molina, Stanford University.
6. *The Anatomy of a Large-Scale Hypertextual Web search Engine.* (The Google Blueprint) Sergey Brin, Lawrence Page.Stanford University and Google.
7. Swoogle: A Search and Metadata Engine for the Semantic Web. Ding, Finin, Joshi, Pan, Cost, Peng, Reddivari, Doshi, Sachs. Department of Computer Science and Electronic Engineering. University of Maryland Baltimore County.

Questions?

Thank you for your attention.