Crawling a Country: Better Strategies than Breadth-First for Web Page Ordering

Ricardo Baeza-Yates
Center for Web Research
Universidad de Chile
rbaeza@dcc.uchile.cl

Carlos Castillo
Center for Web Research
Universidad de Chile
ccastillo@dcc.uchile.cl

Mauricio Marin
Center for Web Research
Universidad de Magallanes
mauricio.marin@umag.cl

Andrea Rodriguez
Center for Web Research
Universidad de Concepción
andrea@udec.cl

ABSTRACT

This article compares several page ordering strategies for Web crawling under several metrics. The objective of these strategies is to download the most "important" pages "early" during the crawl. As the coverage of modern search engines is small compared to the size of the Web, and it is impossible to index all of the Web for both theoretical and practical reasons, it is relevant to index at least the most important pages.

We use data from actual Web pages to build Web graphs and execute a crawler simulator on those graphs. As the Web is very dynamic, crawling simulation is the only way to ensure that all the strategies considered are compared under the same conditions. We propose several page ordering strategies that are more efficient than breadth-first search and strategies based on partial Pagerank calculations.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*performance evaluation*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*web-based services*

General Terms

Measurement, Performance, Experimentation

Keywords

Web crawler, Scheduling policy, Web page importance

1. INTRODUCTION

Web crawlers work by downloading millions or billions of Web pages, recursively following links to create a repository of pages. This repository of pages is indexed and used for Web search, or it can be analyzed for Web characterization.

If a crawler is able to download completely a finite set of pages, then any crawling order is good, because at the end all the pages will be downloaded. This is not the case for real Web crawlers, which usually are not able to download all of the Web, mainly due to the following reasons:

1. Network bandwidth and disk space of the Web crawler are neither infinite nor free. The current size of Google's index is of more than 8×10^9 pages. If the average page size is about 15 kB, then the crawling cost could be around 4 million dollars for a full crawl [24] if we only consider network costs. Given that not all pages are of equal quality, there is a motivation for downloading just the most important ones.

2. Pages change over time, and a Web crawler's copy of the Web becomes quickly obsolete. Crawling the Web resembles, to some extent, watching the sky on a clear night [8]. The stars we see never existed simultaneously as we are seeing them, as the light takes different times to reach our eyes. In the same way, pages are crawled at different times, and the search engine's collection is not a snapshot of the Web. Some pages can disappear or change by the time they are presented to a user as search results, in the same way as some stars we see in the sky were gone long ago.

In the case of stars we are limited by the speed of light and the large distances, but in Web crawling we can re-crawl Web pages if we are uncertain about their current status. So, at some point we can no longer continue downloading only new pages and we have to start downloading –or at least verifying– some of the pages we have already seen. Also, this is the only way to find new URLs.

3. The amount of information on the Web is finite, but the number of pages is infinite.

If we define "Web page" as everything that has an URL, then the number of Web pages is infinite [7], due to dynamic pages, and then a crawling process never ends. What we do is to limit the amount of dynamic pages downloaded from any given server in an attempt to make the Web finite.

Our motivation is to develop a scheduling policy for downloading pages from the Web which guarantees that, even if we do not download all of the known pages, we still download the most "valuable" ones. As the number of pages grows, it will be increasingly important to download the "better" ones first, as it will be impossible to download them all. This not only tries to minimize crawling time with respect

WWW2005, May 10–14, 2005, Chiba, Japan.
.

to some measure of page quality, but also reduces the real cost of crawling.

The main contributions of this paper are the following:

- To the best of our knowledge, this is the most comprehensive comparison of crawling strategies published so far.
- It proposes new strategies with better performance than the well-known breadth-first and partial Pagerank ones.
- It proposes strategies which use historical information to conduct the crawling process. By "historical" we mean global Pagerank information calculated from the previous crawling of the same Web.

The next section outlines previous work in the subject of Web crawlers and Web crawling scheduling. Section 3 describes the importance metric and the crawling framework we used in our experiments. Section 4 introduces the page ordering strategies, which are then tested in Section 5. Finally, Section 6 presents our conclusions and directions for future work.

2. PREVIOUS WORK

Web crawlers are a central part of search engines, and details on their crawling algorithms are kept as business secrets. When algorithms are published, there is often an important lack of details that prevents others from reproducing the work. There are also concerns about "search engine spamming", which prevent major search engines from publishing their ranking and crawling algorithms.

2.1 Web crawler architectures

Some descriptions of crawlers (in chronological order) include: RBSE [32], the WebCrawler [49], the World Wide Web Worm [41], the crawler of the Internet Archive [15], the personal search agent SPHINX [44], an early version of the Google crawler [13], the CobWeb [26], Mercator, a modular crawler [38], Salticus [14], the WebFountain incremental crawler [31], the PolyBot parallel crawler [52], a crawler module implemented in WebRACE [58], the decentralized Ubicrawler [10], the crawler of the FAST search engine [51], the WIRE crawler [6] which was used for this research and the Dominos crawler [35].

In addition to the specific crawling architectures listed above, there are general crawler architectures in the literature, including a parallel crawler architecture by Cho [21] and a general crawler architecture described by Chakrabarti [17]. Finally, some crawlers released under the GNU public license include Larbin [4], WebBase [27] and HT://Dig [2].

Besides architectural issues, studies about Web crawling have focused on parallelism [21, 52], discovery and control of crawlers for Web site administrators [54, 53, 1, 40], accessing content behind forms (the "hidden" web) [50], detecting mirrors [23], keeping the freshness of the search engine copy high [30, 20, 25, 5], focused crawling [37, 18, 42, 28, 43] and Web server cooperation [12, 48].

2.2 Web crawling ordering

Cho *et al.* [22] made the first study on policies for crawling scheduling. Their data set was a 180,000-pages crawl from the `stanford.edu` domain, in which a crawling simulation was done with different strategies. The ordering metrics tested were breadth-first, backlink-count and partial Pagerank, which are explained later in this article. One of the conclusions was that if the crawler wants to download pages with high Pagerank early during the crawling process, then the partial Pagerank strategy is the better, followed by breadth-first and backlink-count. However, these results are for just a single domain.

Najork and Wiener [45] performed an actual crawl on 328 million pages, using breadth-first ordering. They found that a breadth-first crawl captures pages with high Pagerank early in the crawl, but they did not attempt to compare it to other strategies.

Abiteboul *et al.* [3] designed a crawling strategy based on an algorithm called OPIC (On-line Page Importance Computation). In OPIC, each page is given an initial sum of "cash" which is distributed equally among the pages it points to. It is similar to a Pagerank computation, but it is faster and is done just in one step. An OPIC-driven crawler downloads first the pages in the crawling frontier with higher amounts of "cash". Experiments were carried in a 100,000-pages synthetic graph with a power-law distribution of inlinks. However, there was no comparison with other strategies nor experiments in the real Web.

Boldi *et al.* [11] used simulation on subsets of the Web of 40 million pages from the `.it` domain and 100 million pages from the WebBase crawl, testing breadth-first against random ordering and an omniscient strategy. The winning strategy was breadth-first, although a random ordering also performed surprisingly well. One problem is that the WebBase crawl is biased to the crawler used to gather the data. They also showed that Pagerank calculations carried on partial subgraphs of the Web give a bad approximation of the actual Pagerank values.

This paper complements the results presented in [16] – which include an evaluation of some crawling strategies on the Chilean Web– by comparing a wider range of strategies in a larger dataset and by including time in the analysis, proposing strategies which use historical information, as well as the Kendall's τ metric in the evaluation.

3. METHODOLOGY

We downloaded subsets of the Web graph (complete country domains to avoid biasing the crawled data) using a Web crawler, and then simulated different strategies. In this section, we define the importance metric and the crawling architecture.

3.1 Page importance metric: Pagerank

Our goal is to download "important" pages first. We use the Pagerank score as a metric of importance of Web pages. The Pagerank algorithm was introduced by Page *et al.* [47], and has a recursive definition stating in simple terms that "a page with high Pagerank is a page referenced by many pages with high Pagerank".

To calculate the Pagerank, each page on the Web is modeled as a state in a system, and each hyperlink as a transition between two states. The Pagerank value of a page is the probability of being in a given page when this system reaches its stationary state. A good metaphor for understanding this is to think in terms of a "random surfer", a person who visits pages at random, and upon arrival to each page, chooses an outgoing link uniformly at random from the links in that page. The Pagerank of a page is the fraction

of time the random surfer spends at each page.

However, actual Web graphs include many pages with no out-links, which act as "rank sinks" as they accumulate rank but never distribute it to other pages. Also, we would not like pages to accumulate ranking, by not passing all of their score to other pages. For these reasons, most of the implementations of Pagerank add "random jumps" from every page to all pages in the collection, including itself.

In terms of the random surfer model, we can state that when choosing the next step, the random surfer either chooses a page at random from the collection with probability ϵ, or chooses to follow a link from the current page with probability $1 - \epsilon$. This is the model used for calculating Pagerank in practice, and it is described by the following equation:

$$Pagerank(p) = \frac{\epsilon}{N} + (1 - \epsilon) \sum_{x \in \Gamma^-(p)} \frac{Pagerank(x)}{|\Gamma^+(x)|} \quad (1)$$

where N is the number of pages in the collection, and the parameter ϵ is typically between 0.1 and 0.2 (we used 0.15), based on empirical evidence. Pagerank is a global, static measure of quality of a Web page, very efficient in terms of computation time, as it only has to be calculated once at indexing time and is later used repeatedly at query time.

Note that Pagerank can be maliciously manipulated and in fact there are thousands or millions of Web pages created specifically for the objective of deceiving the overall ranking function [33]. Pagerank by itself cannot be used as the only quality metric in real Web search, and some modifications to the algorithm can be made to reduce the impact of collusion [59]. Also, Pagerank might miss some high quality pages as it is biased towards older pages [9]. However, Pagerank is a good measure of page quality, better than just counting the number of in-links, is used in global-scale search engines, and in our case, and in general, it was impractical to get human judgments (or reliable page traffic statistics) for a significant fraction of the pages.

3.2 Crawling simulator

A valid download order for a Web crawler must obey some restrictions. The most important self-imposed restriction for Web crawlers is that they must not overload the Web sites they visit [40].

Web crawlers operate with a high degree of parallelism, opening hundreds or thousands of HTTP connections simultaneously, so a Web crawler could potentially overload a Web site with requests. To avoid overloading Web sites, Web crawlers usually download at most one page from a given Web site at a given time, and wait a certain period of time between two accesses to the same site. The exact waiting time w is usually a few seconds. Anecdotal evidence from the crawlers of known search engines in access log files shows varying access intervals, from 20 seconds to 3–4 minutes, and we use $w = 15$ seconds as a default in both simulation and actual Web crawls.

To comply with these restrictions, it is customary to maintain a queue of Web sites, and for each Web site a queue of pages to download. This is depicted in Figure 1.

The crawling simulator receives as an input a Web graph representing links between pages; URLs pointing to non existent pages, or to pages with errors are not included in this graph. We have observed that they are about 15% of the links found. The crawling simulator uses this information to build a heap priority queue with nodes representing sites.

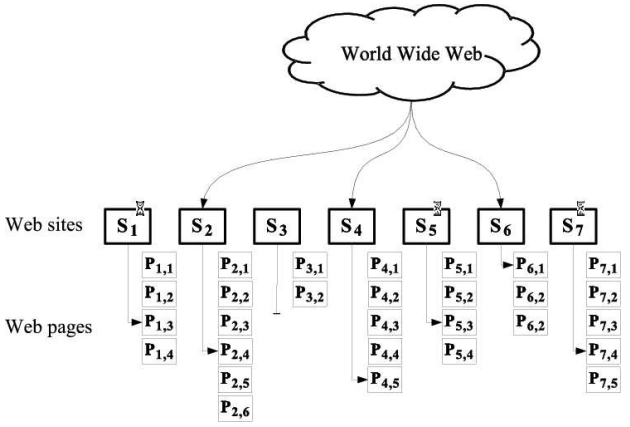

Figure 1: Operation of a crawler. There is a queue for each Web site with at most one connection to each active Web site (sites 2, 4, and 6). Some Web sites are "idle", because they have transfered pages too recently (sites 1, 5, and 7) or because they have exhausted all of their pages to download (3).

At each simulation step, the scheduler chooses the topmost Web site from the queue of Web sites and sends this site's information to a module that will simulate downloading pages from the Web site. The page sizes, distribution of speeds and latencies can also be provided as an input to the simulator, although for the experiments with the simulator described in this paper we were only interested in the crawling order, and not in the total crawling time.

The crawler keeps a list of at most r actuve connections to Web sites, and simulates (possible) bandwidth saturation using the number of current transfers, their speeds and the bandwidth of the simulated link. Note that, as shown in Figure 1, r is bound by the number of active Web sites, so crawling performance can drop dramatically if the latter is too small. Being careful about this leads to one of the best heuristics, as shown later.

4. PAGE ORDERING STRATEGIES

The most important problem of a page ordering strategy is that it must work with partial information, because the "complete" graph is only known by the end of the crawling. However, if we are repeating a crawl, that is, if we are crawling a portion of the Web which was already crawled a few weeks or a few months ago, we have historical information available. This is typically the case for a search engine and the historical information can be used to direct the crawl towards pages which had a high Pagerank in the last crawl.

The schedule is done using an estimation of page quality. This estimation can also be combined with page changes information, because in some cases a crawler might want to bias the download schedule towards both high-quality and fast-changing pages [6].

We consider three types of strategies regarding how much information they can use: no extra information, historical information, and all the information. A random ordering can be considered a baseline for comparison. In that case, the Pagerank is expected to grow approximately linearly with the number of pages crawled.

4.1 Strategies with no extra information

Breadth-first Under this strategy, the crawler visits the pages in breadth-first ordering. It starts by visiting all the home pages of all the "seed" Web sites, and Web page heaps are kept in such a way that new pages added go at the end. This is the same strategy tested by Najork and Wiener [45], which in their experiments showed to capture high-quality pages first.

Backlink-count This strategy crawls first the pages with the highest number of links pointing to it, so the next page to be crawled is the most linked from the pages already downloaded. This strategy was described by Cho *et al.* [22].

Batch-pagerank This strategy calculates an estimation of Pagerank, using the pages seen so far, every K pages downloaded. The next K pages to download are the pages with the highest estimated Pagerank. We used $K = 100,000$ pages, which in our case gives about 30 to 40 Pagerank calculations during the crawl. This strategy was also studied by Cho *et al.* [22], and it was found to be better than backlink-count. However, Boldi *et al.* [11] showed that the approximations of Pagerank using partial graphs can be very inexact.

Partial-pagerank This is like *batch-pagerank*, but in between Pagerank re-calculations, a temporary pagerank is assigned to new pages using the sum of the Pagerank of the pages pointing to it divided by the number of out-links of those pages.

OPIC This strategy is based on OPIC [3], which can be seen as a weighted backlink-count strategy. All pages start with the same amount of "cash". Every time a page is crawled, its "cash" is split among the pages it links to. The priority of an uncrawled page is the sum of the "cash" it has received from the pages pointing to it. This strategy is similar to Pagerank, but has no random links and the calculation is not iterative – so it is much faster.

Larger-sites-first The goal of this strategy is to avoid having too many pending pages in any Web site, to avoid having at the end only a small number of large Web sites that may lead to spare time due to the "do not overload" rule. The crawler uses the number of un-crawled pages found so far as the priority for picking a Web site, and starts with the sites with the larger number of pending pages. We introduced this strategy in [16] and was found to be better than breadth-first.

4.2 Strategies with historical information

These strategies use the Pagerank of a previous crawl as an estimation of the Pagerank in this crawl, and start in the pages with a high Pagerank in the last crawl. This is only an approximation because Pagerank can change: Cho and Adams [19] report that the average relative error for estimating the Pagerank four months ahead is about 78%. Also, a study by Ntoulas *et. al* [46] reports that "the link structure of the Web is significantly more dynamic than the contents on the Web. Every week, about 25% new links are created".

We explore a number of strategies to deal with the pages found in the current crawl which were not found in the previous one:

Historical-pagerank-omniscient New pages are assigned a Pagerank taken from an oracle that knows the full graph.

Historical-pagerank-random New pages are assigned a Pagerank value selected uniformly at random among the values obtained in previous crawl.

Historical-pagerank-zero New pages are assigned Pagerank zero, i.e., old pages are crawled first, then new pages are crawled.

Historical-pagerank-parent New pages are assigned the Pagerank of the parent page (the page in which the link was found) divided by the number of out-links of the parent page.

It is likely that modern Web search engines use historical information to some extent, but to the best of our knowledge neither the exact mechanism nor the performance have been published so far.

4.3 Strategy with all the information

Omniscient: this strategy can query an "oracle" which knows the complete Web graph and has calculated the actual Pagerank of each page. Every time the *omniscient* strategy needs to prioritize a download, it asks the oracle and downloads the page with the highest ranking in its frontier. Note that this strategy is bound to the same restrictions as the others, and can only download a page if it has already downloaded a page that points to it.

5. EXPERIMENTS

In this section we describe the experiments done with the crawler simulator and the ordering strategies described in the previous section.

5.1 Datasets: .cl and .gr

We worked with two datasets that correspond to pages under the `.cl` (Chile) and `.gr` (Greek) top-level domains. We downloaded pages using the WIRE crawler [6] in breadth-first mode, including both static and dynamic pages. While following links, we stopped at depth 5 for dynamic pages and 15 for static pages – we use an heuristic to detect most of the dynamic pages based on known filename extensions and searching for a question mark in the URL. We also limited the crawler to download at most 25,000 pages from each Web site.

We made two complete crawls on each domain, as we said before, to avoid biasing the data to the crawling order. We did it in April and May for Chile, and in May and September for Greece. The summary of the characteristics of the collection, as well as some demographic information about the two countries are presented in Table 5.1.

Table 1: Summary of characteristics.

	Greece	Chile
Population [56]	10.9 Million	15.2 Million
GDP [55]	133 US$ bn.	66 US$ bn.
Per-capita GDP [55]	17,697 US$	10,373 US$
Human development [57]	24^{th}	43^{th}
Web servers contacted	29,000-31,000	50,000-51,000
Pages with HTTP OK	3.4-3.6 Mill.	2.4-2.8 Mill

Both datasets are comparable in terms of the number of Web pages, but there are wide differences in terms of geography, language, demographics, history, etc. Dill *et al.* [29] studied several sub-sets of the Web, and found that the Web graph is self-similar in several senses and at several scales, and that this self-similarity is pervasive, as it holds for a

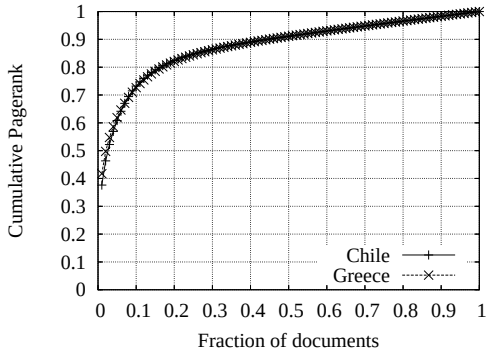

Figure 2: Cumulative Pagerank in the .CL and .GR domain, showing almost exactly the same distribution; these curves represents an upper bound on the cumulative Pagerank of any crawling strategy.

number of different parameters. Note than a large sub-set of the whole Web (and any non-closed subset of the Web) is always biased by the strategy used to crawl it. Top-level domains are useful because they represent pages sharing a common cultural context; we consider that they are more useful than large Web sites because pages in a Web site are more homogeneous. Obviously, different top-level domains differ in some aspects, for instance, related to the prevalence of spam pages, but while spam pages appear as anomalies in the histogram of in-degree or out-degree, the overall distribution still follows a power-law [34].

5.2 Performance metrics

Our importance metric is Pagerank. Thus, for evaluating different strategies, we calculated the Pagerank value of every page in each Web graph and used those values to calculate the evolution of the Pagerank as the simulated crawl goes by. We recall that although Pagerank is not the best measure, there is no other practical choice.

We used three measures of performance: cumulative Pagerank, average Pagerank and Kendall's τ.

Cumulative Pagerank: we plotted the sum of the Pagerank of downloaded pages at different points of the crawling process. The strategies which are able to reach values close to the target total value 1.0 faster are considered the most efficient ones. A strategy which selects random pages to crawl will produce a diagonal line in this graph.

There is an upper bound on how well this can be done, and it is given by the distribution of Pagerank, which is shown in Figure 2.

The results for the different strategies are shown in Figure 3; in this simulation we are using $r = 1$, one robot at a time, because we are not interested in the time for downloading the full Web, but just in the crawling order.

Obviously the *omniscient* has the best performance, but it is in some sense too greedy because by the last stages of the crawl it performs close to random. Note that other strategies can perform better than the *omniscient* strategy at the end of the crawl, because all strategies are bound to the same restrictions about being polite with Web sites, and therefore, downloading all the "good" pages too early leads to reducing too much the list of available Web sites and reduces the performance in the last stages of the crawl.

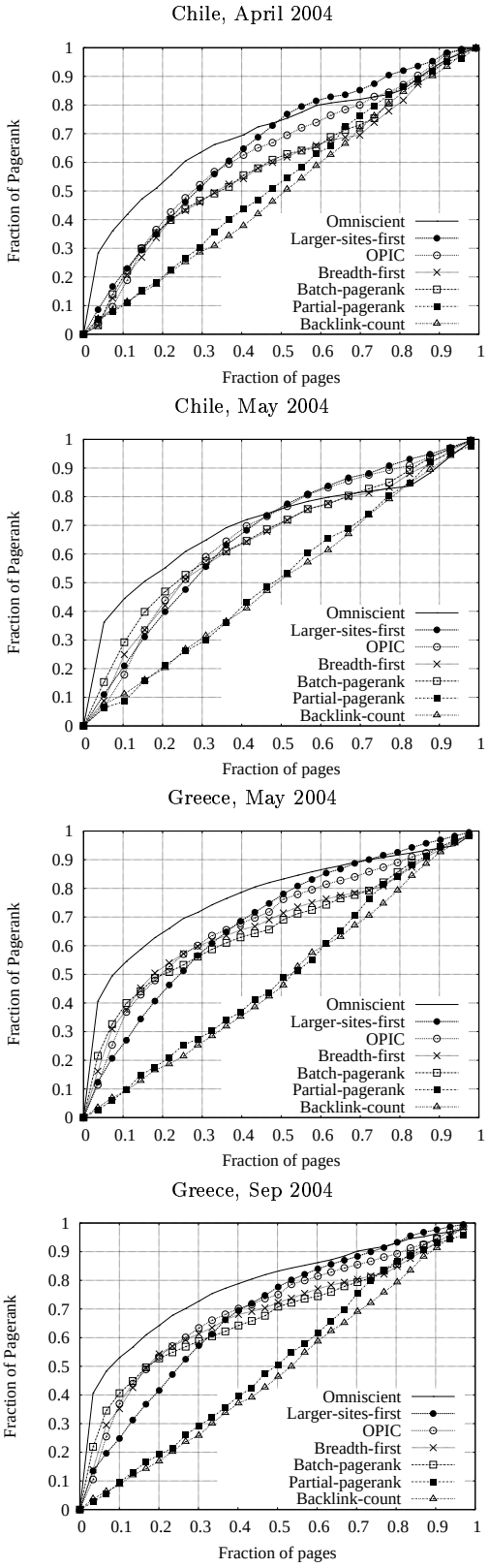

Figure 3: Comparison of cumulative Pagerank vs retrieved pages with the different strategies, excluding the historical strategies.

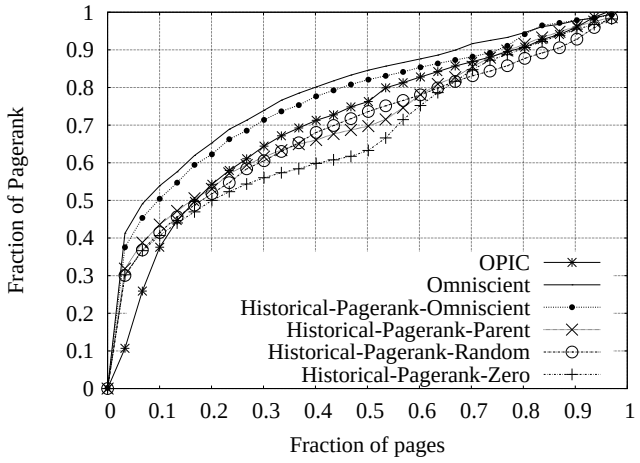

Figure 4: Comparison of cumulative Pagerank using the historical strategies against the *omniscient* and *OPIC* strategies, for a crawl of the Greek Web in September 2004, using Pagerank information from May 2004.

On the other end, *backlink-count* and *partial-pagerank* are the worst strategy according to cumulative Pagerank, and perform worser than a random crawl. They both tend to get stuck in pages that are locally optimal, and then fail to discover other pages.

Breadth-first is close to the best strategies for the first 20-30% of pages, but after that it becomes less efficient.

The strategies *batch-pagerank*, *larger-sites-first* and *OPIC* have better performance than the other strategies, with an advantage towards *larger-sites-first* when the desired coverage is high. These strategies can retrieve about half of the Pagerank value of their domains downloading only around 20-30% of the pages.

We tested the *historical-pagerank* strategies in the Greek Web graph of September, using the Pagerank calculated in May for guiding the crawl – we are using Pagerank that is 4 months old. We were able to use the Pagerank of the old crawl (May) for only 55% of the pages, as the other 45% of pages were new pages, or were not crawled in May.

Figure 4 shows results for a number of ways of dealing with the above 45% of pages along with results for the same Web graph but using the *OPIC* strategy for comparison. These results show that the May Pagerank values are not detrimental to the crawl of September.

The *OPIC* has a bad performance at the beginning of the crawl, but it improves as more link information is gathered as the crawl goes by. The *historical-pagerank-random* strategy has a good performance, despite of the fact that the Web graph is very dynamic [46], and than on average it is difficult to estimate the Pagerank using historical information [19]. A possible explanation is that the ordering of pages by Pagerank changes more slowly and in particular the pages with high ranking have a more stable position in the ranking than the pages with low ranking, which exhibit a larger variability. Also, as Pagerank is biased towards old pages [9], 55% of pages that already existed in May account for 72% of the total Pagerank in September.

Table 2: Comparison of the scheduling strategies, considering average cumulative Pagerank during the crawl and Kendall's τ of the page ordering against the optimal ordering.

Strategy	Avg. Pagerank $\pm\sigma$	τ
Backlink-count	0.50 ± 0.01	0.0157
Partial-pagerank	0.52 ± 0.01	0.0236
Breadth-first	0.64 ± 0.04	0.1293
Batch-pagerank	0.63 ± 0.03	0.1961
OPIC	0.67 ± 0.03	0.2229
Larger-sites-first	0.67 ± 0.01	0.2498
Hist.-pr-zero	0.68	0.3573
Hist.-pr-random	0.70	0.3689
Hist.-pr-parent	0.70	0.3520
Hist.-pr-omni.	0.77	0.6385
Omniscient	0.74 ± 0.04	0.6504

Average Cumulative Pagerank: this is the average across the entire crawl. As we have normalized the cumulative Pagerank as a fraction of documents, it is equivalent to the area under the curves shown in Figure 3. The historical strategies are better than OPIC, mostly because of OPIC's poorer performance at the beginning of the crawl. The result is presented in Table 2, in which we have calculated the average of the strategies across the four collections (note that the *historical-pagerank* strategies were tested in a single pair of collections, so their values are not average values). We can see that the strategies with history marginally improve (at most 3%) *OPIC* and *larger-sites-first* (leaving out the *omniscient* ones).

Kendall's Tau: this is a metric for the correlation between two ranked lists, which basically measures the number of pairwise inversions in the two lists [39]. Two identical lists have $\tau = 1$, while two totally uncorrelated lists have $\tau = 0$ and reversed lists have $\tau = -1$. We calculated this coefficient for a 5000-page sample of the page ordering in each strategy, against the same pages when pages were ordered by Pagerank. The results are shown in Table 2. In this case *larger-sites-first* is the best of the strategies without history, and *historical-pagerank-random* otherwise (without taking in account the *omniscient* ones).

We attempted to measure estimated search length for different page percentiles, for instance, measuring how many page downloads are necessary to get the top 10% of pages. However, this kind of measure is very sensitive to small variations, such as having a single high-quality page downloaded by the end of the crawl.

5.3 Validation

We performed two actual crawls using WIRE [6] in two consecutive weeks in the `.GR` domain, using *Breadth-first* and *Larger-sites-first*. We ran the crawler in a single Intel PC of 3.06GHz with 1Gb of RAM under Linux, in batches of up to 200,000 pages, using up to $r = 1000$ simultaneous network connections, with $w = 5$ seconds between accesses to the same Web site, and increasing to $w = 15$ for sites with less than 100 pages.

In the case of an actual Web crawl, we are interested in the time variable, as it is worthless to download pages in the

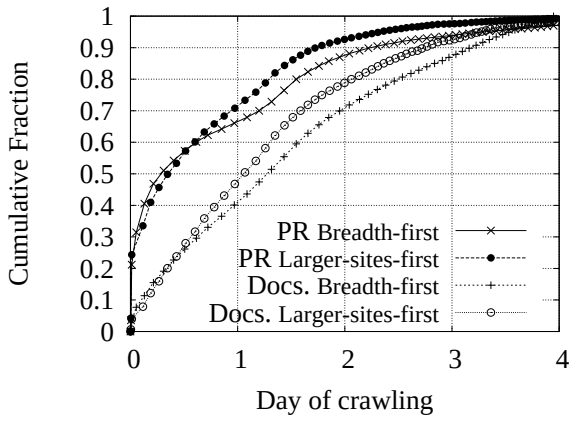

Figure 5: Cumulative Pagerank (PR) and cumulative fraction of documents (Docs.) of an actual crawl of the *.GR* domain using two strategies: *Breadth-first* and *Larger-sites-first*.

right order if they cannot be downloaded fast. We calculated the Pagerank of all the pages in the collection when the crawling was completed and then measured how much of the total Pagerank was covered during each batch. The results are shown in Figure 5.

Both crawling strategies are efficient in terms of downloading the valuable pages early, but *larger-sites-first* is faster in both downloading documents and downloading Pagerank. This strategy "saves" several small Web sites for the middle and end part of the crawl.

6. CONCLUSIONS

Most of the strategies tested were able to download important pages first. As shown in [11], even a random strategy can perform well on the Web, in the sense that a random walk on the Web is biased towards pages with high Pagerank [36]. However, there are differences in how quickly high-quality pages are found depending on the ordering of pages.

The *historical-pagerank* family of strategies were very good, and in case of no historical information available, both *OPIC* and *larger-sites-first* are our recommendations and their performance is similar. *Breadth-first* has a bad performance compared with these strategies; *batch-pagerank* requires to do a full Pagerank computation several times during the crawl, which is computationally very expensive, and the performance is not better than simpler strategies.

Notice that the *larger-sites-first* strategy has practical advantages over the *OPIC* strategy. First, it requires less computation time, and also does not require keeping a count of weighted in-links to a given page as *OPIC* does. The latter is relevant when we think of distributed crawlers as no communication between computers is required to exchange these data during the crawling process. Thus *larger-sites-first* has better scalability making it more suitable for large scale distributed crawlers. In a real setting, this strategy should include mechanisms to avoid spam pages, such as checking for near-duplicate pages to avoid giving too much ranking to sites that create artificial loops with dynamic pages to inflate their page count.

Our results are more useful for crawls in the order of 10-100 million pages –very large intranets, national domains, and other subsets of the Web. In a large-scale setting of billions of pages, keeping a heap of Web pages in main memory is impossible, and keeping a heap of Web sites might be very expensive. In those cases, using historical information to guide the crawl might be a good alternative to avoid downloading pages in random order.

One remaining open problem is how to compare the time efficiency of crawlers. This has been done for the computational overhead of different focused crawling strategies [43], but the downloading module has been explicitly left out. For example, it is not enough to define a measure of bytes downloaded per second and Mbs of bandwidth, for some standardized PC configuration (GHz and RAM). We also need a standardized set of sites and a fixed standard server, which not only allows to compare crawlers in the same machine, but also with exactly the same connectivity.

Acknowledgments

We acknowledge the support of the Millennium Scientific Initiative through Grant P01-029-F of Mideplan. We thank Efthimis Efthimiadis from Univ. of Washington for the initial seeds for the Greek domain.

7. REFERENCES

[1] Robotcop. www.robotcop.org, 2002.

[2] HT://Dig. http://www.htdig.org/, 2004. GPL software.

[3] S. Abiteboul, M. Preda, and G. Cobena. Adaptive on-line page importance computation. In *Proceedings of the twelfth international conference on World Wide Web*, pages 280–290. ACM Press, 2003.

[4] S. Ailleret. Larbin. http://larbin.sourceforge.net/index-eng.html, 2004. GPL software.

[5] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan. Searching the Web. *ACM Transactions on Internet Technology (TOIT)*, 1(1):2–43, August 2001.

[6] R. Baeza-Yates and C. Castillo. Balancing volume, quality and freshness in web crawling. In *Soft Computing Systems - Design, Management and Applications*, pages 565–572, Santiago, Chile, 2002. IOS Press Amsterdam.

[7] R. Baeza-Yates and C. Castillo. Crawling the infinite Web: five levels are enough. In *Proceedings of the third Workshop on Web Graphs (WAW)*, volume 3243 of *Lecture Notes in Computer Science*, pages 156–167, Rome, Italy, October 2004. Springer.

[8] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[9] R. Baeza-Yates, F. Saint-Jean, and C. Castillo. Web structure, dynamics and page quality. In *Proceedings of String Processing and Information Retrieval (SPIRE)*, volume 2476 of *Lecture Notes in Computer Science*, pages 117 – 132, Lisbon, Portugal, 2002. Springer.

[10] P. Boldi, B. Codenotti, M. Santini, and S. Vigna. UbiCrawler: a scalable fully distributed Web crawler. *Software, Practice and Experience*, 34(8):711–726, 2004.

[11] P. Boldi, M. Santini, and S. Vigna. Do your worst to make the best: Paradoxical effects in pagerank incremental computations. In *Proceedings of the third Workshop on Web Graphs (WAW)*, volume 3243 of *Lecture Notes in Computer Science*, pages 168–180, Rome, Italy, October 2004. Springer.

[12] O. Brandman, J. Cho, H. Garcia-Molina, and N. Shivakumar. Crawler-friendly web servers. In *Proceedings of the Workshop on Performance and Architecture of Web Servers (PAWS)*, Santa Clara, California, USA, June 2000.

[13] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, April 1998.

[14] R. D. Burke. Salticus: guided crawling for personal digital libraries. In *Proceedings of the first ACM/IEEE-CS joint conference on Digital Libraries*, pages 88–89, Roanoke, Virginia, June 2001.

[15] M. Burner. Crawling towards eternity - building an archive of the world wide web. *Web Techniques*, 2(5), May 1997.

[16] C. Castillo, M. Marin, A. Rodríguez, and R. Baeza-Yates. Scheduling algorithms for Web crawling. In *Latin American Web Conference (WebMedia/LA-WEB)*, pages 10–17, Riberao Preto, Brazil, October 2004. IEEE CS Press.

[17] S. Chakrabarti. *Mining the Web*. Morgan Kaufmann Publishers, 2003.

[18] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31(11–16):1623–1640, 1999.

[19] J. Cho and R. Adams. Page quality: In search of an unbiased Web ranking. Technical report, UCLA Computer Science, 2004.

[20] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In *Proceedings of ACM International Conference on Management of Data (SIGMOD)*, pages 117–128, Dallas, Texas, USA, May 2000.

[21] J. Cho and H. Garcia-Molina. Parallel crawlers. In *Proceedings of the eleventh international conference on World Wide Web*, pages 124–135, Honolulu, Hawaii, USA, May 2002. ACM Press.

[22] J. Cho, H. García-Molina, and L. Page. Efficient crawling through URL ordering. In *Proceedings of the seventh conference on World Wide Web*. Elsevier Science, April 1998.

[23] J. Cho, N. Shivakumar, and H. Garcia-Molina. Finding replicated web collections. In *ACM SIGMOD*, pages 355–366, 1999.

[24] N. Craswell, F. Crimmins, D. Hawking, and A. Moffat. Performance and cost tradeoffs in web search. In *Proceedings of the 15th Australasian Database Conference*, pages 161–169, Dunedin, New Zealand, January 2004.

[25] A. Czumaj, I. Finch, L. Gasieniec, A. Gibbons, P. Leng, W. Rytter, and M. Zito. Efficient Web searching using temporal factors. *Theoretical Computer Science*, 262(1–2):569–582, 2001.

[26] A. S. da Silva, E. A. Veloso, P. B. Golgher, B. A. Ribeiro-Neto, A. H. F. Laender, and N. Ziviani. Cobweb - a crawler for the brazilian web. In *Proceedings of String Processing and Information Retrieval (SPIRE)*, pages 184–191, Cancun, México, September 1999. IEEE CS Press.

[27] L. Dacharay. WebBase. http://freesoftware.fsf.org/webbase/, 2002. GPL Software.

[28] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. Focused crawling using context graphs. In *Proceedings of 26th International Conference on Very Large Databases (VLDB)*, pages 527–534, Cairo, Egypt, September 2000.

[29] S. Dill, R. Kumar, K. S. Mccurley, S. Rajagopalan, D. Sivakumar, and A. Tomkins. Self-similarity in the web. *ACM Trans. Inter. Tech.*, 2(3):205–223, 2002.

[30] R. W. Edward G. Coffman, Z. Liu. Optimal robot scheduling for web search engines. *Journal of Scheduling*, 1(1):15–29, 1998.

[31] J. Edwards, K. S. McCurley, and J. A. Tomlin. An adaptive model for optimizing performance of an incremental web crawler. In *Proceedings of the Tenth Conference on World Wide Web*, pages 106–113, Hong Kong, May 2001. Elsevier Science.

[32] D. Eichmann. The RBSE spider: balancing effective search against web load. In *Proceedings of the first World Wide Web Conference*, Geneva, Switzerland, May 1994.

[33] N. Eiron, K. S. McCurley, and J. A. Tomlin. Ranking the web frontier. In *Proceedings of the 13th international conference on World Wide Web*, pages 309–318. ACM Press, 2004.

[34] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: Using statistical analysis to locate spam Web pages. In *Proceedings of the seventh workshop on the Web and databases (WebDB)*, June 2004.

[35] Y. Hafri and C. Djeraba. High performance crawling system. In *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 299–306. ACM Press, 2004.

[36] M. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. On near–uniform url sampling. In *Proceedings of the Ninth Conference on World Wide Web*, pages 295–308, Amsterdam, Netherlands, May 2000. Elsevier Science.

[37] M. Hersovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur. The shark-search algorithm. An application: tailored Web site mapping. In *Proceedings of the seventh conference on World Wide Web*, pages 317–326. Elsevier Science, April 1998.

[38] A. Heydon and M. Najork. Mercator: A scalable, extensible web crawler. *World Wide Web Conference*, 2(4):219–229, April 1999.

[39] M. G. Kendall. *Rank Correlation Methods*. Griffin, London, England, 1970.

[40] M. Koster. Robots in the web: threat or treat ? *ConneXions*, 9(4), April 1995.

[41] O. A. McBryan. GENVL and WWWW: Tools for taming the web. In *Proceedings of the first World Wide Web Conference*, Geneva, Switzerland, May 1994.

[42] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[43] F. Menczer, G. Pant, P. Srinivasan, and M. E. Ruiz. Evaluating topic-driven web crawlers. In *Proceedings of the 24th conference on research and development in information retrieval (SIGIR)*, pages 241–249. ACM Press, 2001.

[44] R. Miller and K. Bharat. Sphinx: A framework for creating personal, site-specific web crawlers. In *Proceedings of the seventh conference on World Wide Web*, Brisbane, Australia, April 1998. Elsevier Science.

[45] M. Najork and J. L. Wiener. Breadth-first crawling yields high-quality pages. In *Proceedings of the Tenth Conference on World Wide Web*, pages 114–118, Hong Kong, May 2001. Elsevier Science.

[46] A. Ntoulas, J. Cho, and C. Olston. What's new on the web?: the evolution of the web from a search engine perspective. In *Proceedings of the 13th conference on World Wide Web*, pages 1 – 12, New York, NY, USA, May 2004. ACM Press.

[47] L. Page, S. Brin, R. Motwani, and T. Winograd. The Pagerank citation algorithm: bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[48] G. Pant, S. Bradshaw, and F. Menczer. Search engine-crawler symbiosis. In *Proceedings of the European Conference on Digital Libraries (ECDL)*, volume 2769 of *Lecture Notes in Computer Science*, pages 221–232. Springer, August 2003.

[49] B. Pinkerton. Finding what people want: Experiences with the WebCrawler. In *Proceedings of the first World Wide Web Conference*, Geneva, Switzerland, May 1994.

[50] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *Proceedings of the Twenty-seventh International Conference on Very Large Databases (VLDB)*, pages 129–138, Rome, Italy, 2001. Morgan Kaufmann.

[51] K. M. Risvik and R. Michelsen. Search engines and web dynamics. *Computer Networks*, 39(3), June 2002.

[52] V. Shkapenyuk and T. Suel. Design and implementation of a high-performance distributed web crawler. In *Proceedings of the 18th International Conference on Data Engineering (ICDE)*, pages 357 – 368, San Jose, California, February 2002. IEEE CS Press.

[53] J. Talim, Z. Liu, P. Nain, and E. G. C. Jr. Controlling the robots of web search engines. In *Proceedings of ACM Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS/Performance)*, pages 236–244, Cambridge, Massachusetts, USA, June 2001.

[54] P.-N. Tan and V. Kumar. Discovery of web robots session based on their navigational patterns. *Data Mining and Knowledge discovery*, 6(1):9–35, 2002.

[55] The Economist. Country Profiles, 2002.

[56] United Nations. Population Division, 2002.

[57] United Nations. Human Development Reports, 2003.

[58] D. Zeinalipour-Yazti and M. D. Dikaiakos. Design and implementation of a distributed crawler and filtering processor. In *Proceedings of the fifth Next Generation Information Technologies and Systems (NGITS)*, volume 2382 of *Lecture Notes in Computer Science*, pages 58–74, Caesarea, Israel, June 2002. Springer.

[59] H. Zhang, A. Goel, R. Govindan, K. Mason, and B. V. Roy. Making eigenvector-based reputation systems robust to collusion. In *Proceedings of the third Workshop on Web Graphs (WAW)*, volume 3243 of *Lecture Notes in Computer Science*, pages 92–104, Rome, Italy, October 2004. Springer.