SHARC: Framework for Quality-Conscious Web Archiving

Dimitar Denev Arturas Mazeika Marc Spaniol Gerhard Weikum
Max Planck Institute for Informatics
Campus E1.4, 66123 Saarbrücken, Germany
{ddenev,amazeika,mspaniol,weikum}@mpi-inf.mpg.de

ABSTRACT

Web archives preserve the history of born-digital content and offer great potential for sociologists, business analysts, and legal experts on intellectual property and compliance issues. Data quality is crucial for these purposes. Ideally, crawlers should gather sharp captures of entire Web sites, but the politeness etiquette and completeness requirement mandate very slow, long-duration crawling while Web sites undergo changes.

This paper presents the SHARC framework for assessing the data quality in Web archives and for tuning capturing strategies towards better quality with given resources. We define quality measures, characterize their properties, and derive a suite of quality-conscious scheduling strategies for archive crawling. It is assumed that change rates of Web pages can be statistically predicted based on page types, directory depths, and URL names. We develop a stochastically optimal crawl algorithm for the offline case where all change rates are known. We generalize the approach into an online algorithm that detect information on a Web site while it is crawled. For dating a site capture and for assessing its quality, we propose several strategies that revisit pages after their initial downloads in a judiciously chosen order. All strategies are fully implemented in a testbed, and shown to be effective by experiments with both synthetically generated sites and a daily crawl series for a medium-sized site.

Keywords

Web Archiving, Data Quality

1. INTRODUCTION

1.1 Motivation

The Web is in constant flux. 80% of the pages change within a half a year. To prevent the content from disappearing, national libraries and organizations like the Internet Archive (`archive.org`) and the European Archive (`europarchive.org`) are collecting and preserving the ever changing Web. These archives not only capture the history of born-digital content but also reflect the zeitgeist of different time periods over more than a decade. This is a gold mine for sociologists, politologists, media and market analysts, as well as experts on intellectual property (IP, e.g., at patent offices) and compliance with Internet legislation (e.g., for consumer services). For example, when a company is accused of violating IP rights (regarding inventions or trademarks), it may want to prove the existence of certain phrases on its Web pages as of a certain timepoint in the past. Conversely, an Internet-fraud investigation may aim at proving the absence of certain phrases (e.g., proper pricing statements or rights of withdrawal) on a Web site. Clearly, this entails that Web archives need to be maintained with a high standard of data quality.

Crawling the Web for archiving substantially differs from crawling performed by major search providers. Search engines aim at broad coverage of the Web, target the most important pages, and schedule revisits of the pages based on their individual freshness and importance. It may even be sufficient to see the `href` anchor in order to index a page without ever visiting the page itself. In contrast, archive curators are interested in a complete capture of a site either at reasonably regular time points (weekly, monthly, quarterly) or in aftermaths (natural disasters, political scandals, research projects).

During a site crawl, pages may undergo changes, resulting in a blurred snapshot of the site. We borrow the terms *blur* and *sharpness* (the opposite of blur) from photography to denote the quality of the snapshot. Similarly to photography, the longer the exposure time (timespan of the entire site crawl), the higher the risk of blurring the capture (archiving pages in different states of the site). In contrast, if the site's pages did not change at all (or changed little) during the crawl, we say that the snapshot is sharp (or almost sharp).

Avoiding blurred captures is important for the quality assurance of the Web archive and its professional usage. Ideally, a user should get mutually consistent pages. In case mutual consistency of pages can not be fully assured, there should be at least guarantees (deterministic or stochastic) for the quality of the Web archive. For example, a journalist can hardly use a web archive of a soccer team because of its inconsistent content. She finds the page of a soccer match of April 18th pointing to the details of the match of April 24th. The pages are inconsistent and not helpful to the journalist. Similarly, an archive of a Web site was disapproved as evidence in a lawsuit about intellectual property rights [23]. The archive was of insufficient quality and no guarantees could be made

VLDB '09, August 24-28, 2009, Lyon, France

about the consistency of its content. In these cases a strategy for getting a sharp capture or stating the level of consistency for the capture could have made a difference.

The simplest strategy to obtain a sharp capture of a Web site and avoid anomalies would be to freeze the entire site during the crawl period. This is impractical as an external crawler cannot prevent the site from posting new information on its pages or changing its link structure. On the contrary, the politeness etiquette for Internet robots forces the crawler to pause between subsequent HTTP requests, so that the entire capturing of a medium-sized site (e.g., a university) may take many hours or several days. This is an issue for search-engine crawlers, too, but it is more severe for archive crawlers as they cannot stop a breadth-first site exploration once they have seen enough `href` anchors. So slow but complete site crawls drastically increase the risk of blurred captures.

An alternative strategy that may come to mind would be to repeat a crawl that fails to yield a sharp capture and keep repeating it until eventually a blur-free snapshot can be obtained. But this is an unacceptably high price for data quality as the crawler operates with limited resources (servers and network bandwidth) and needs to carefully assign these to as many different Web sites as possible.

1.2 Contribution

While the issue of web-archive quality is obvious, it is unclear how to formalize the problem and address it technically. This paper is the first to provide framework for quality assurance of Web archives. It develops a model of quality properties as well as a suite of algorithms for crawls that allows us to assess and optimize site-capturing strategies.

Our framework, coined SHARC for Sharp Archiving of Web-Site Captures, is based on a *stochastic notion of sharpness*. In line with the prior literature [1, 7, 20], we model site changes by Poisson processes with page-specific change rates. We assume that these rates can be statistically predicted based on page types (e.g., MIME types), depths within the site (e.g., distance to site entry points), and URLs (e.g., manually edited user homepages vs. pages generated by content management systems). The user-perceived blur by subsequent access to site captures is derived from a random-observer model: the user is interested in a former timepoint uniformly drawn within a large interval of potential access points.

Within this model, we can reason about the expected sharpness of a site capture or the probabilistic risk of obtaining a blurred (i.e., not perfectly sharp) capture during a crawl. This in turn allows us to devise crawl strategies that aim to optimize these measures. While *stochastic guarantees* of this kind are good enough for explorative use of the archive, access that aims to prove or disprove claims about former site versions needs *deterministic guarantees*. To this end, SHARC also introduces crawl strategies that visit pages twice: a first visit to fetch the page and a later revisit to validate that the page has not changed. The order of visiting and revisiting pages is a degree of freedom for the crawl scheduler. We propose strategies that strive for both deterministic and stochastic sharpness (absence of changes during the site capturing).

SHARC includes a suite of novel algorithms:

- *SHARC-offline* assumes a-priori knowledge of all URLs and their specific change rates, and arranges downloads in an organ-pipe manner with the hottest pages in the middle. It comes with a stochastic guarantee about the expected sharpness.
- *SHARC-online* drops these assumptions and operates with an estimate of the number of pages on the site but without prior knowledge of any URLs other than the crawl's entry point. The algorithm aims to approximate the organ-pipe shape, but can lead to suboptimal schedules.
- For deterministic guarantees, *SHARC-revisits* visits page twice to detect changes during the crawl and ensure exact dating. It provides an easily measurable notion of sharpness.
- Finally, when (parts of) sites change at a high rate so that no acceptable crawl strategy would be able to guarantee sharp captures, *SHARC-threshold* uses a tunable threshold for disregarding hot pages that are "beyond hope".

All SHARC strategies are fully implemented in our testbed. We present experimental evaluation studies with both synthetically generated Web sites and repeated crawls of the medium-sized domain (`www.mpi-inf.mpg.de`) (our institute) over an extended time period. The experiments demonstrate the practical feasibility of our approach and the advantages of our strategies compared to more traditional crawl methods.

The paper is organized as follows. Section 2 reviews related work and the state of the art in archive crawling. Section 3 introduces our computational model for Web archiving and site capturing. Sections 4 through 7 present our crawl algorithms: the offline strategy, the online strategy, the strategy with page revisits, and the strategy with thresholding for hopeless pages. Finally, Section 8 presents an experimental evaluation of all strategies.

2. RELATED WORK

The book on Web archiving [12] gives a thorough overview on issues, open problems, and techniques related to Web archiving. The most typical Web archiving scenario is a crawl of all pages for a given site done once (or periodically) for a given starting time (or given periodic starting times). The book draws a parallel between a photograph of a moving scene and the quality of the Web archive, however the issue is left as an open problem. Mohr et al. [17] describe the Heritrix crawler, an extensible crawler used by European Archive and other Web archive institutions. By default Heritrix archives sites in the breadth-first order of discovery, and is highly extensible in scheduling, restriction of scope, protocol based fetch processors, resource usage, filtering, etc. The system does not offer any tools or techniques to measure and optimize crawling for sharpness.

Data caching is the most related work in the field of databases. Data caching stores copies of the most important data items to decrease the cost of subsequent retrievals of the item. Key issues are distribution of the load of data-intensive Web applications [22, 13], efficiency issues in search engines [3], performance-effective cache synchronization [21, 11]. It is realistic and typical to assume notifications of change. Data quality for Web archiving raises different issues. The Web site cannot notify about the changes of Web pages, the archive does not synchronize changed pages, archives

should optimize for sharpness while the perfect consistency is a prerequisite in data caching.

Crawling of the Web for the purpose of search engines received a lot of attention. Key issues here are efficiency [15, 8], freshness [4], importance [18], relevance to keyword queries [6, 9, 5, 10]. Different weights of importance are assigned to the pages on the Web and resources are reserved (frequency of crawls, crawling priority, etc). The freshness, age, PageRank, BackLink, and other properties are used to compute the weights. Other metrics to measure when and how much of the individual pages has been changed have been proposed as well [19, 1]. Web change models characterizing the dynamics of Web pages have been developed [16, 14]. Typically the changes of page p_i are modeled with a Poisson process [6] with the average *change rate* λ_i. The number of changes per time unit Δ is distributed according to Poisson distribution with parameter λ_i if

$$P[\#\text{changes of } p_i \text{ in } \Delta \text{ is } k] = \frac{e^{-\lambda_i \Delta}(\Delta\lambda_i)^k}{k!}.$$

This is equivalent to postulating that the time between two successive changes of the page p_i is exponentially distributed with parameter λ_i:

$$P[\text{time between changes of } p_i \text{ is less than } \Delta] = 1 - e^{-\lambda_i \Delta}.$$

Mathematically, the change rate λ_i is the limit of the number of changes per time unit Δ as Δ approaches to zero.

Olston and Pandey have designed a crawling strategy optimized for freshness [20]. In order to determine which page to download at time point t, Olston and Pandey compute the utility function $U_{p_i}(t)$ for each page p_i and its time points of changes. The utility function is defined such that it gives priority to those pages whose changes will not be overwritten by subsequent changes for the longest timespan. Our setup is very different. We optimize the sharpness of entire captures and not the freshness of individual pages.

3. WEB ARCHIVING MODEL

The Web archive, or archive for short, periodically crawls a large number of sites, e.g., on a weekly or monthly basis. So each site is covered by a series of versions, called (site) *captures*. Each crawl aims to obtain a complete capture of the entire site. Crawling needs to observe the politeness requirements of a site, with pauses of several seconds or even a minute between successive HTTP requests. Thus, an entire site capture may span several days. (The crawler may crawl many sites in parallel to utilize the throughput.) When a new site crawl starts we assume that either the URLs of all pages are known upfront or at least one entry page is known from which the crawl can explore the site's page graph. The former is an assumption made by the SHARC-offline strategy, and is relaxed by the SHARC-online strategy. We may assume that the total number of pages in a site can be estimated when a crawl starts, based on site properties such as domain name or attributes obtained by the HTTP reply when fetching the site's entry page.

Recalling the related work presented in Section 2, we assume that pages undergo changes according to a *Poisson process* and a *change rate* λ_i, and that λ_i can be statistically predicted by regression models. In practice, good quality predictors can be developed for classes of similar pages based on 10–20 of features including its MIME type (e.g., html vs. pdf), depth in the site graph relative to the entry point, URL (user homepages versus pages generated by the content management system), presence of javascript [1, 7].

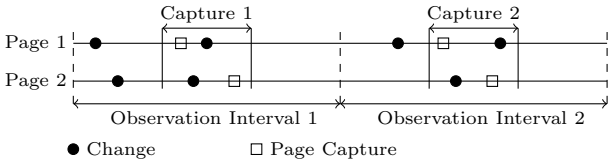


Figure 1: Web Archiving Model

The archive is typically accessed by time-travel queries for a user specified time point t. This query is mapped to an existing capture. When the archive does not have a capture that matches t, the user's request is mapped either to the most recent available capture whose timestamp does not exceed t or to the nearest capture in the past or future (whichever is closer to t). The first variant corresponds to the simplest standard semantics in temporal database systems [2]. The second variant may appear non-standard, but makes sense in our Web archive setting because the user's timepoint of interest may often be fuzzy or a crude estimate for exploration purposes. For example, when a sociologist wants to investigate opinions of a social group on a particular topic using the content of a site as of May 2001 (which could technically be interpreted as mid May, i.e., May 15, if a real timepoint is needed), she may be equally happy with a capture from April 28 or June 3 if the site was not captured during May. For use cases that entail proving the absence or presence of certain contents on the site, this observation-mapping mode works as long as the user's access request specifies an interval (e.g., April 2001, meaning April 1 through April 30) and any timepoint from the interval is acceptable.

Figure 1 illustrates our web archiving model. All pages of the Web site are captured periodically (cf. rectangles in the figure) defining capture intervals. Each capture corresponds to an observation interval (cf. Capture and Observation Intervals in the figure). All temporal queries falling into the interval are mapped to the capture. The changes of the pages (indicated as black circles on the figure) during the observation interval introduces blur. If no changes occurred in the observation interval we call the capture sharp.

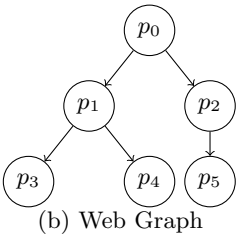
The Web archiving model is quite different from the crawl models of search engines. Search engines aim at broad coverage of the most important pages in the Web (not necessarily entire sites), recrawling the pages and optimizing their freshness (the most recent version compared to now timepoint). In contrast, Web archives should archive all (or specifically selected) versions (at regular crawl time points) of the entire site and return the most appropriate version of the page for a given time-travel query, which is not necessarily the freshest version.

4. SHARC-OFFLINE

In this section we establish the metric of blur (sharpness) for a given archive and develop SHARC-offline, the optimal crawling strategy for Web archiving in an ideal environment. SHARC-offline is not a feasible solution in a realistic run-time setting, but it is a useful baseline for developing practically viable algorithms and assessing their quality. We assume

Page	Change Rate λ
p_0	0
p_1	1
p_2	2
p_3	3
p_4	4
p_5	5

(a) Change Rates



(b) Web Graph

Figure 2: Example of a Web Site with Change Rates

that the Web archive consists of Web pages $p_0, \ldots, p_n$ (all URLs are known in advance), which change according to the Poisson distribution with average change rates $\lambda_0, \ldots, \lambda_n$ (the number of changes in a time unit). For convenience we assume that the indexes of the pages are chosen so that $\lambda_0 \leq \cdots \leq \lambda_n$. This simplifies and shortens the indexes and the algorithms without the loss of generality. We assume that the download times of the pages are periodic with politeness delay Δ in between the downloads (the most typical scenario). To simplify mathematical expressions we assume that the capture interval starts at time 0 and the observation interval coincides with the capture interval: $[o_s, o_e] = [c_s, c_e] = [0, n\Delta]$. Later in Section 4.4 we generalize the equations and omit the assumption. Figure 2 presents an example that we use throughout the paper.

4.1 Blur

Blur of a page and the archive are the key measures to assess the quality of the Web archives.

DEFINITION 4.1 (BLUR). *Let p_i be a Web page archived at time t_i. The blur of the page is the expected number of changes between t_i and query time t, averaged through observation interval $[0, n\Delta]$:*

$$B(p_i, t_i, n, \Delta) = \frac{1}{n\Delta} \int_0^{n\Delta} \lambda_i \cdot |t - t_i| dt = \frac{\lambda_i \omega(t_i, n, \Delta)}{n\Delta}, \quad (1)$$

where

$$\omega(t_i, n, \Delta) = t_i^2 - t_i n\Delta + \frac{(n\Delta)^2}{2}. \quad (2)$$

is the download schedule penalty.

Let $P = (p_0, \ldots, p_n)$ be Web pages archived at times $T = (t_0, t_1, \ldots, t_n)$. The blur of the archive is the sum of the blurs of the individual pages:

$$B(P, T, n, \Delta) = \frac{1}{n\Delta} \sum_{i=0}^{n} \lambda_i \omega(t_i, n, \Delta). \quad (3)$$

The blur of a Web page in the capture is the multiplication of its average change rate and $\omega(t_i, n, \Delta)$. $\omega(t_i, n, \Delta)$ depends on the download time and the length of the capture interval $n\Delta$ and does not depend on the page. Therefore $\omega(t_i, n, \Delta)$ can be interpreted as penalty of downloading page p_i at time t_i.

EXAMPLE 4.2 (BLUR). *Consider the Web site in Figure 2 with download time i for page p_i (for example page p_3 is downloaded at time $t_3 = 3$). The blur of p_1 is*

$$B(p_1, 1, 5, 1) = \frac{1 \cdot (1^2 - 1 \cdot 5 \cdot 1 + (5 \cdot 1)^2/2)}{5 \cdot 1} = 1.7.$$

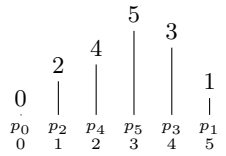


Figure 3: Organ Pipes

Similarly $B(p_0, 0, 5, 1) = 0$, $B(p_2, 2, 5, 1) = 2.6$, $B(p_3, 3, 5, 1) = 3.9$, $B(p_4, 4, 5, 1) = 6.8$, $B(p_5, 5, 5, 1) = 12.5$. The blur of the archive is

$$B(P, T, 5, 1) = 0 + 1.7 + 2.6 + 3.9 + 6.8 + 12.5 = 27.5$$

Properties of the download schedule penalty immediately allows to reason about the blur of the capture for different delay intervals.

THEOREM 4.3. (PROPERTIES OF THE SCHEDULE PENALTY) *Doubling download delay quadruples the schedule penalty and doubles the blur of the archive:*

$$\omega(i\Delta, n, \Delta) = \Delta^2 \omega(i, n, 1), \quad (4)$$

$$B(P, T, n, \Delta) = \Delta B(P, T, n, 1). \quad (5)$$

PROOF. The proof follows from the definitions of the schedule penalty and the blur. □

The result in Theorem 4.3 comes at no surprise: the schedule penalty is a quadratic function and doubling the delay between the downloads increases the blur four times.

4.2 Optimal Download Schedule

Different download schedules results in different levels of blur. In the rest of the section we investigate the optimal download schedule of the archive. Mathematically, for the given Web site $p_0, \ldots, p_n$ we will identify the optimal sequence $t_0, \ldots, t_n$, (a permutation of $0, \Delta, \ldots, n\Delta$) that minimizes the blur of the archive (cf. Equation (3)). In particular we show that the pages that change most should be downloaded in the middle of the crawl.

EXAMPLE 4.4 (OPTIMAL DOWNLOAD SCHEDULE). *Lets continue Example 4.2. The optimal download schedule is $t_0 = 0$ and $t_1 = 5$ (the most outer points of the interval) for the least changing pages p_0 and p_1, $t_1 = 1$ and $t_2 = 4$ (the next outer points) for the second and third least changing pages p_2 and p_3, followed by $t_3 = 2$ and $t_4 = 3$ for pages that change most p_4 and p_5. The blur of the capture with optimal download schedule is $B(P, T', 5, 1) = 22.7$.*

Figure 3 illustrates the optimal download schedule where the change rate of the scheduled download is visualized as a line of length proportional to the download rate. The visualization resembles the organ-pipes with the highest pipes allocated as much in middle as possible.

THEOREM 4.5. (OPTIMAL DOWNLOAD SCHEDULE) *Let p_0, $p_1, \ldots, p_n$ be the Web site such that $\lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_n$. Then the optimal download schedule $t_0, \ldots, t_n$ is defined by the following*

$$t_i = \frac{i}{2} \quad \text{if } i \text{ is even and} \quad t_i = n - \frac{i-1}{2} \quad \text{otherwise} \quad (6)$$

for $i = 0, 1, \ldots, n$.

PROOF. The proof of Theorem 4.5 is based on three observations:

(i) λ_i are ordered increasingly: $\lambda_i \leq \lambda_{i+1}$,

(ii) Equation (6) orders $\omega(t_i, n, \Delta)$ decreasingly: $\omega(t_i, n, \Delta) \geq \omega(t_i, n, \Delta)$,

(iii) Equation (3) is minimized when λ_i are ordered decreasingly and $\omega(t_i, n, \Delta)$ are ordered increasingly.

λ_i are scheduled increasingly because of the assumption of the theorem, and therefore case (i) is true.

Function $\omega(t, n, \Delta)$ is quadratic wrt t with its minimum at $t_{min} = (n\Delta)/2$. Equation (6) schedules t_is in such a way that $\omega(t_n, n, \Delta)$ is smallest. The greater the index i, the closer to the middle t_i is allocated. Therefore $\omega(t_i, n, \Delta)$ are scheduled decreasingly and Case (ii) is true.

The proof of Case (iii) is given in Lemma 4.6 below. □

LEMMA 4.6. *Let* $\lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_n$ *and* $\omega(t_0, n, \Delta) \geq \omega(t_1, n, \Delta) \geq \cdots \geq \omega(t_n, n, \Delta)$. *Let* $j_0, \ldots, j_n$ *be a permutation of* $0, \ldots, n$. *Then*

$$\sum_{m=0}^{n} \lambda_m \omega(t_m, n, \Delta) \leq \sum_{m=0}^{n} \lambda_{t_{i_m}} \omega(t_{j_m}, n, \Delta). \tag{7}$$

PROOF. Indeed, let $i_0, \ldots, i_n$ be the optimal permutation of $0, \ldots, n$ such that $(\lambda_{i_0}, \ldots, \lambda_{i_n})$ and $(\omega(t_{j_0}, n, \Delta), \ldots, \omega(t_{j_n}, n, \Delta))$ minimize the right hand side of Equation (7).

The largest element of λ-s must be multiplied by the smallest element of $\omega(t_{i_l})$, otherwise the solution is not optimal. We prove this step by contradiction. Let

$$\begin{matrix}(\lambda_{i_0}, & \ldots, & \lambda_{i_k}, & \ldots, & \lambda_{i_l}, & \ldots, & \lambda_{i_n}) \\ (\omega(t_{j_0}, n, \Delta), & \ldots, & \omega(t_{j_k}, n, \Delta), & \ldots, & \omega(t_{j_l}, n, \Delta), & \ldots, & \omega(t_{j_n}, n, \Delta))\end{matrix}$$

be the optimal sequence, however $\lambda_{i_k} = \lambda_0$ (the smallest among all λ_is) and $\omega(t_{j_l}, n, \Delta) = \omega(t_0, n, \Delta)$ (the largest among all $\omega(t_i, n, \Delta)$s). Then we can show that the by swapping λ_{i_k} with λ_{i_l} (or alternatively $\omega(t_{i_k}, n, \Delta)$ with $\omega(t_{i_l}, n, \Delta)$) we can further decrease the sum in Equation (7). Indeed, the sum without the swap is:

$$\sum_{\substack{m=1,\ldots,n \\ m\neq l, m\neq k}} \lambda_{i_m} \omega(t_{j_m}, n, \Delta) + \lambda_0 \omega(t_{j_k}, n, \Delta) + \lambda_{i_l} \omega(t_0, n, \Delta) \tag{8}$$

The sum with the swap is:

$$\sum_{\substack{m=0,\ldots,n \\ m\neq l, m\neq k}} \lambda_{i_m} \omega(t_{j_m}, n, \Delta) + \lambda_0 \omega(t_0, n, \Delta) + \lambda_{i_k} \omega(t_{j_l}, n, \Delta) \tag{9}$$

Since the first sums in Equations (8) and (9) are the same we reach the contradiction if we prove that

$$\lambda_0 \omega(t_{j_k}, n, \Delta) + \lambda_{i_l} \omega(t_0, n, \Delta) > \lambda_0 \omega(t_0, n, \Delta) + \lambda_{i_k} \omega(t_{j_l}, n, \Delta). \tag{10}$$

The left hand side (LHS) of Equation (10) is:

$$\begin{aligned} LHS &= \lambda_0 \omega(t_{j_k}, n, \Delta) + \lambda_{i_l} \omega(t_0, n, \Delta) \\ &= \lambda_0 \Big(\omega(t_0, n, \Delta) + \big(\omega(t_{j_k}, n, \Delta) - \omega(t_0, n, \Delta) \big) \Big) \\ &\quad + \lambda_{i_l} \omega(t_0, n, \Delta) \\ &= \lambda_0 \omega(t_0, n, \Delta) + \big(\lambda_{i_l} + (\lambda_0 - \lambda_{i_l}) \big) \\ &\quad \times \big(\omega(t_{j_k}, n, \Delta) - \omega(t_0, n, \Delta) \big) + \lambda_{i_l} \omega(t_0, n, \Delta) \\ &= \lambda_0 \omega(t_0, n, \Delta) + \lambda_{i_l} \omega(t_{j_k}, n, \Delta) \\ &\quad + \big(\lambda_{i_l} - \lambda_0 \big) \big(\omega(t_0, n, \Delta) - \omega(t_{j_k}, n, \Delta) \big) \\ &= RHS + \text{strictly positive number}, \end{aligned}$$

since λ_0 is the smallest among λ_is and $\omega(t_0, n, \Delta)$ is the largest among $\omega(t_j, n, \Delta)$s.

The proof of lemma follows with the help of mathematical induction. The induction basis is trivial. The optimal solution for n reduces to the optimal solution for $n-1$ elements, since the λ_0 must be multiplied with $\omega(t_0)$ in the optimal solution. □

4.3 SHARC-offline Algorithm

Algorithm 1 depicts the algorithm of SHARC-offline. Since all the pages are known and sorted in advance we need to scan all the pages only once to schedule the downloads.

```
    input  : sorted pages p_0, ..., p_n
    output: download schedule p_0^D, ..., p_n^D
1   begin
2       for i = 0, 1, ..., n do
3           if i is even then p_i^D = p_{i/2}
4           else p_i^D = p_{n-(i-1)/2}
5       end
6   end
```

Algorithm 1: SHARC-offline

4.4 General Observation Interval

In this section we generalize the blur and the optimal download sequence for the case when observation interval $[o_s, o_e]$ does not coincide with the capture interval. Then the blur of a Web page is

$$B(p_i, t_i, n, \Delta) = \frac{1}{o_e - o_s} \int_{o_s}^{o_e} \lambda_i \cdot |t - t_i| dt = \frac{\lambda_i \omega(t_i, o_s, o_e)}{o_e - o_s},$$

where

$$\omega(t_i, o_s, o_e) = t_i^2 - t_i(o_s + o_e) + \frac{o_s^2 + o_e^2}{2}$$

is the generalized download sequence penalty and the blur of the capture is the sum of the blurs of the individual pages (cf. Equation (3)).

Theorem 4.5 schedules the most changing pages in the middle of the capture interval (point $n\Delta/2$). In case the observation interval does not coincide with the capture interval and there are no restrictions for the start of the capture interval we should schedule the most changing pages around the middle of the observation interval (point $(o_s + o_e)/2$). We formalize it in the theorem bellow.

THEOREM 4.7. *Let $t_0, \ldots, t_n$ be the optimal download schedule for Web site with $[0, n\Delta]$ observation interval. Then*

$$t_i + \frac{o_e + o_s - n\Delta}{2}$$

is the optimal download position for page p_i with $[o_s, o_e]$ observation interval.

In case the observation interval and the capture intervals are fixed, the most changing pages should be allocated as close to the middle point of the observation interval as possible.

5. SHARC-ONLINE

The SHARC-offline strategy assumes that all URLs of the Web site are known in advance. In this section we relax the assumption and introduce SHARC-online, the archive crawl optimization strategy with no or limited knowledge of the web site. Starting with a given set of seeds SHARC-online *incrementally* extract the urls of other pages from the downloaded pages and schedules the pages for download so the archive is optimized for sharpness. We organize this section as follows. First, we explain the incremental detection of the Web site structure and discuss most common crawl strategies in Section 5.1. We develop the SHARC-online strategy by example in Section 5.2. Finally, we formally define the SHARC-online strategy and present the algorithm of the strategy in Sections 5.3 and 5.4.

5.1 Discovery of the Web Graph

Typically crawlers do not know the URLs of the pages in the crawled site. The archive crawlers start with the download of a given set of URLs (seeds of the crawl), extract the URLs of the downloaded pages, and continue the process until all the documents are downloaded and no new URLs are detected. At any iteration the crawler keeps a DD-list (Downloaded-Detected list) of URLs. The downloaded list of URLs consists of all URLs that are already crawled, while the detected URLs comprise the extracted from the downloaded pages but not yet downloaded URLs. Different crawl strategies schedule the URLs in a different manner. Below we demonstrate the most popular crawl strategies: depth-first and breadth-first on the example Web graph in Figure 2.

Table 1 depicts the detection and download of Web pages of the depth-first strategy. The strategy starts with the seed page p_0 and inserts it into the detected part of the DD-list (cf. p_0 in the iteration $I = 0$ in Table 1). Then it downloads the page (p_0 is moved to the downloaded part of the DD-list, cf. $I = 1$ in the table) parses the html page and inserts detected URLs p_1, p_2 into the detected part of the DD-list. The depth first strategy inserts newly detected pages at the beginning of the detected list, thus the depth-first pages have higher priority for download (cf iteration $I = 2$ in the table). In contrast, breadth-first strategy appends newly discovered pages, assigning a higher priority for breadth-first pages (cf. Table 2).

5.2 SHARC-online Strategy by Example

At any given iteration the crawler does not know all pages, but only the pages of the Web site in the DD-list. Our SHARC-online strategy optimizes the download and detection of the Web pages incrementally. Given the (estimated) size of the Web site, the SHARC-online produces a download schedule that resembles the schedule of the SHARC-offline strategy.

I	DD-list	
	Downloaded	Detected
0		p_0
1	p_0	p_1, p_2
2	p_0, p_1	p_3, p_4, p_2
3	p_0, p_1, p_3	p_4, p_2
4	p_0, p_1, p_3, p_4	p_2
5	p_0, p_1, p_3, p_4, p_2	p_5
6	$p_0, p_1, p_3, p_4, p_2, p_5$	

Table 1: Depth-First Crawl Strategy

I	DD-list	
	Downloaded	Detected
0		p_0
1	p_0	p_1, p_2
2	p_0, p_1	p_2, p_3, p_4
3	p_0, p_1, p_2	p_3, p_4, p_5, p_5
	...	...
6	$p_0, p_1, p_2, p_3, p_4, p_5$	

Table 2: Breadth-First Crawl Strategy

Table 3 illustrates the SHARC-online strategy for the running example. The SHARC-online crawl starts with p_0 page as a seed and the estimated number of pages in the site $n + 1 = 6$. The crawl downloads page p_0 and detects another two pages p_1, p_2. The algorithm is in its ascending phase, and therefore it schedules the downloads in increasing schedule of the change rate λ_i. In the $I = 2$ iteration the algorithm downloads p_1 and detects additional pages p_3 and p_4. The number of detected and downloaded pages exceeds the middle of the interval and the algorithm switches to the middle phase to preserve the middle of the organ-pipes. The algorithm downloads p_4 and p_3 in the middle phase. Then the number of downloaded pages exceeds the middle the algorithm finishes in the descending phase with the downloads of p_2 and p_5.

5.3 Formalization of SHARC-online

SHARC online schedules the detected pages of the DD-list for download. The strategy aims to resemble the schedule of the SHARC offline strategy. Due to the limited knowledge of the detected pages the algorithm has three phases: ascending, middle, and descending phases.

I	DD-list	
	Downloaded	Detected
0		p_0
1	p_0	p_1, p_2
2	p_0, p_1	p_2, p_3, p_4
3	p_0, p_1, p_4	p_2, p_3
4	p_0, p_1, p_4, p_3	p_2
5	p_0, p_1, p_4, p_3, p_2	p_5
6	$p_0, p_1, p_4, p_3, p_2, p_5$	

Table 3: SHARC-online Crawl Strategy

The SHARC-online strategy maintains the list of detected pages $(p_0^E, p_1^E, \ldots, p_{n^E-1}^E)$ (sorted in ascending order according to the change rates), the number of downloaded pages n^D, the number of detected pages n^E, and an approximated overall number of the pages $n+1$. The SHARC-online strategy expresses the next page to be downloaded $p_{n^D}^D$ in terms of these three variables.

5.3.1 Ascending Phase

The ascending phase resembles the beginning of the organ-pipes and is applied when the number of downloaded and detected pages is below the estimated middle point of the crawl. During this phase the algorithm implements the cheapest first strategy. Equation (11) formalizes the ascending strategy.

$$p_{n^D}^D = p_0^E \tag{11}$$

The ascending strategy is executed as long as the number of downloaded and detected pages is less than half of the size of the site:

$$n^D + n^E \leq \frac{n+1}{2}. \tag{12}$$

Example 5.1. (Ascending Phase) *Consider $I = 1$ step in Table 3. The number of downloaded pages $n^D = 1$, the number of detected pages $n^E = 2$, and the list of detected pages sorted in ascending order according to the λs is $(p_0^E, p_1^E) = (p_1, p_2)$ Lets assume that the estimated number of pages in the crawl is $n+1 = 6$. Since $n^D + n^E = 1 + 2 \leq 3 = \frac{n+1}{2}$, therefore the algorithm is in the ascending phase and the next download element is $p_1^D = p_1^E = p_2$.*

5.3.2 Middle Phase

The middle phase schedules the next download so the symmetry around the middle of the organ-pipes is preserved as much as possible. For each downloaded page on the ascending part we reserve an appropriate page on the descending part of the organ-pipes. The strategy is applied when the overall number of downloaded and detected pages exceeds the half of the number of the pages, but the number of downloaded pages has not reached the middle of the crawl. Equation (13) formalizes the phase.

$$p_{n^D}^D = \begin{cases} p_{n^D}^E & \text{if } n^D < n^E, \\ p_{n^E-1}^E & \text{otherwise.} \end{cases} \tag{13}$$

Equation (14) formalizes the conditions when the middle phase is applied.

$$n^D + n^E > \frac{n+1}{2}, \qquad n^D \leq \frac{n+1}{2}. \tag{14}$$

Example 5.2. (Middle Phase) *Lets continue Example 5.1 with $I = 2$ step. The number of downloaded pages $n^D = 2$, the number of detected pages $n^E = 3$, and the list of detected pages sorted in ascending order according to the λs is*

$$(p_0^E, p_1^E, p_2^E) = (p_2, p_3, p_4).$$

Since

$$n^D + n^E = 2 + 3 > 3 = \frac{n+1}{2} \text{ and } n^D = 2 < 3 = \frac{n+1}{2},$$

therefore the algorithm is in the middle phase and the next download element is

$$p_2^D = p_2^E = p_4.$$

5.3.3 Descending Phase

The descending phase resembles the ending of the organ-pipes and is applied when the number of downloaded pages is more than the half of the (estimated) page number. During this phase the algorithm implements the most expensive first strategy. Equation (15) formalizes the descending strategy.

$$p_{n^D}^D = p_{n^E-1}^E \tag{15}$$

The descending phase is executed as soon as the number of downloaded pages exceeds the middle of the organ-pipes and until all detected URLs are downloaded:

$$n^D > \frac{n+1}{2}, \qquad n^E \neq 0. \tag{16}$$

Example 5.3 (Descending Phase). *Lets continue Example 5.2 with $I = 4$ step. The number of downloaded pages $n^D = 4$ and the number of detected pages $n^E = 1$. Since $n^D = 4 > 3 = \frac{n+1}{2}$, therefore the algorithm is in the descending phase and the next download element is $p_5^D = p_0^E = p_2$.*

5.4 SHARC-online Algorithm

Algorithm 2 depicts the SHARC-online algorithm. At each iteration (lines 4–12) the algorithm inspects the sizes of downloaded and detected lists and identifies whether the algorithm is in the ascending (line 5), middle (line 6) or descending (line 8) phase and computes the index of the next download.

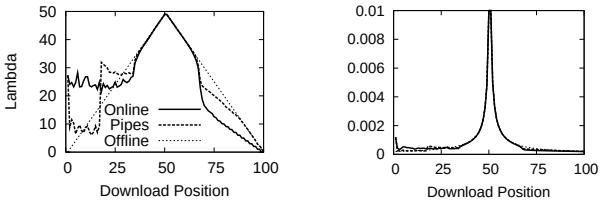
```
    input : sorted seeds (p_0, ..., p_m),
            estimated size of the crawl n
    output: download sequence (p_0^D, ..., p_n^D)
 1  begin
 2      P^D = (p_0^D, ..., p_{n^D}^D) = (), n^D = 0
 3      P^E = (p_0^E, ..., p_{n^E}^E) = (p_0, ..., p_m), n^E = m
 4      while P^E ≠ ∅ do
 5          if n^D + n^E ≤ (n+1)/2 then pos = 0
 6          else if n^D ≤ (n+1)/2 then
 7              pos = n^D < n^E ? p_{n^D}^E : p_{n^E-1}^E
 8          else pos = n^E
 9          append(P^D, p_pos^E), remove(P^E, p_pos^E)
10          add_sort(P^E, urls(p_pos^E))
11          n^D++, n^E--, n^E = n^E + |urls(p_pos^E)|
12      end
13  end
```

Algorithm 2: SHARC-online

5.5 Other Online Strategies

SHARC-online is an online strategy that resembles the organ-pipes of the SHARC-offline. Other online strategies are possible. Consider, for example, the greedy strategy where the smallest elements are allocated at the beginning and largest elements are allocated at the end without the middle phase (cf. Pipes in Figures 4). This strategy performs around two times worse than SHARC-online (cf. the average distribution of λs in Figures 4 for the skewed and very-skewed cases). If the data is less skewed (not that typical in real world cases), SHARC-online resembles the ending and beginning parts of the organ-pipes better with almost the same middle part (cf. Figure 4(a)). If the data is heavily skewed (very

typical in real world) SHARC-online is still two times better on average, however this is almost not visible, since both techniques are very close to the theoretical minimum (cf. Figure 4(b)).



(a) Skew of λs is Low (b) Skew of λs is High

Figure 4: Other Online Strategies

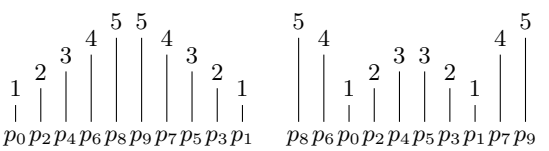
6. WORST-CASE ANALYSIS

In this section we investigate the worst case scenario for the SHARC-online strategy. In SHARC-offline the most changing pages are in the middle of the download interval. Since SHARC-online does not possess the full knowledge about the URLs of the site it may require to download pages that do not follow the SHARC-offline strategy at certain download positions. To cope with the complexity of the task we assume that SHARC-online can schedule $(n+1)-k$ downloads optimally. However, k downloads do not follow the SHARC-offline (k-misplacements).

The worst k-misplacements are the most outwards download positions with the most changing pages. Example 6.1 illustrates the strategy.

EXAMPLE 6.1 (WORST CASE COHERENCE). *Consider a Web site of $n+1=10$ pages with $\lambda_0=\lambda_1=1$, $\lambda_2=\lambda_3=2$, $\lambda_4=\lambda_5=3$, $\lambda_6=\lambda_7=4$, $\lambda_8=\lambda_9=5$. Let $k=4$ be the number of pages that may not follow the SHARC-offline strategy. The optimal SHARC-offline strategy of this site is illustrated in Figure 5(a) with the worst case scenario in Figure 5(b).*

The highest schedule penalty positions in the crawl are the first and the last downloads: $\omega(0,10,1)=\omega(9,10,1)=40.5$, and downloads of the most changing pages (p_8 and p_9 with $\lambda_8=\lambda_9=5$) at these positions maximizes the blur of the archive. The next two highest schedule penalty positions are $\omega(1,10,1)=\omega(8,10,1)=32.5$ and the next two most changing pages p_6 and p_7 are scheduled there. The remaining positions are scheduled according to SHARC-offline strategy resulting in an organ-pipes-like middle part of the download schedule.



(a) SHARC-Offline (b) Worst Case with $k=4$

Figure 5: Example of Worst Case Coherence Scenario with $n=9$ and $k=4$

The increase of the blur for the worst-case scenario is

$$2(5-1)\omega(0,10,1)+2(4-2)\omega(1,10,1)-2(3-1)\omega(2,10,1) - 2(4-2)\omega(3,10,1)-2(5-3)\omega(4,10,1)=145.$$

Since now the blur of the SHARC-offline is $2\cdot\sum_{i=0}^{4} i(i) = 755$, the relative increase of the blur of the worst case is $145/755\approx 20\%$.

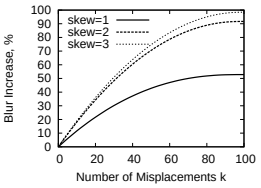
The following theorem summarizes the increase of the worst case scenario with k number of misplacements. For simplicity, we assume that the number of pages $n+1$ and the number of misplacements k are even numbers.

THEOREM 6.2. *Let the number of pages in the site $n+1$ be even with such change rates of the pages: $\lambda_0=\lambda_1\le\lambda_2=\lambda_3\le\cdots\le\lambda_{n-1}=\lambda_n$. Let k be the even number of pages that can be misplaced in the optimal SHARC-offline strategy and Δ be the delay between two downloads. In the worst case the blur of the crawl increases by:*

$$2\sum_{i=0}^{k/2-1}(\lambda_{n-2i}-\lambda_{2i})\omega(i\Delta,n,\Delta) - 2\sum_{i=0}^{(n+1-k)/2}(\lambda_{k+2i}-\lambda_{2i})\omega\Big(\big(\frac{k}{2}+i\big)\Delta,n,\Delta\Big). \quad (17)$$

PROOF. The proof follows from similar arguments as in Theorem 4.5. □

Skew has the largest impact to the increase of blur (cf. Figure 6(a)). The increase of the skew by one increases the blur by an order of magnitude. The misplacement of the most changing pages (cf. the steep increase for k =0–10 in the figure) are the costliest. As the more and more pages are misplaced ($k=20,\ldots,100$) the increase slows down. The number of pages in the figure is $n+1=100$ and the change rates of the pages are set in the following way: $\lambda_{2i-1}=\lambda_{2i}=100/i^{skew}$, $i=0,\ldots,n/2$.



(a) Skew Varies

	n		
k	10^1	10^2	10^3
00%	00%	00%	00%
20%	28%	34%	36%
40%	48%	60%	63%
60%	60%	79%	83%
80%	64%	89%	95%
100 %	64%	92%	99%

(b) n Varies

Figure 6: Worst-Case Analysis

The increase of the size of the Web site $n+1$ (cf. Figure 6(b)) changes the increase substantially for small n (cf. $n=10^1$ and $n=10^2$ in the figure), while the further increase of the number of pages changes the increase only slightly (cf. $n=10^2$ and $n=10^3$ in the figure).

7. REVISIT STRATEGIES

In this section we investigate the crawling strategies where each page is downloaded twice: after all the pages are visited (first download at t_i^v), we revisit them (second download at t_i^r). This allows us to deterministically reason about the state of archive in the interval between the last visit and the first

revisit. We call a page sharp (*#page*) if its versions at visit time and revisit time are the same. Then the number of the sharp pages (*##pages*) deterministically characterizes the quality of the Web archive for the above mentioned interval. Below we present two strategies: SHARC-revisits, which aims at minimizing the blur, and SHARC-threshold which maximizes *##pages*.

7.1 SHARC-revisits

Given Web pages $p_0, p_1, \ldots, p_n$ and the average change rates $\lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_n$ the task is to find the first visits $t_0^v, t_1^v, \ldots, t_n^v$ and the second revisits $t_0^r, t_1^r, \ldots, t_n^r$ so that the blur of the archive is minimized. Since the archive now consists of two versions of the page we return the version of the page that is closer to the given query time t (cf. $\min\{|t_i^v - t|, |t_i^r - t|\}$ in the definition below).

Definition 7.1. (Blur of page and archive with revisits) *Let p_i be a Web page with t_i^v visit and t_i^r revisit times. The blur of Web page p_i is*

$$\begin{aligned} B(p_i, t_i^v, t_i^r, n, \Delta) &= \frac{1}{(2n+1)\Delta} \int_0^{2n\Delta+1} \lambda_i \min\{|t_i^v - t|, |t_i^r - t|\} \\ &= \frac{1}{(2n+1)\Delta} \lambda_i \Big(\int_0^{(t_i^v+t_i^r)/2} |t_i^v - t| \, dt + \int_{(t_i^v+t_i^r)/2}^{(2n+1)\Delta} |t_i^r - t| \, dt \Big) \\ &= \frac{1}{(2n+1)\Delta} \lambda_i \omega(t_i^v, t_i^r, n, \Delta), \end{aligned} \tag{18}$$

where

$$\begin{aligned} \omega(t_i^v, t_i^r, n, \Delta) &= (t_i^v)^2 - \frac{(t_i^v + t_i^r)^2}{4} \\ &\quad + (t_i^r)^2 - t_i^r(2n+1)\Delta + \frac{(2n+1)^2\Delta^2}{2} \end{aligned} \tag{19}$$

is the download schedule penalty for downloads with revisits. The blur of the archive with revisits is the sum of the blurs of the individual pages:

$$B(P, T^v, T^r, n, \Delta) = \sum_{i=0}^{n} B(p_i, t_i^v, t_i^r, n, \Delta), \tag{20}$$

where $T^v = (t_0^v, \ldots, t_n^v)$ are the visit and $T^r = (t_0^r, \ldots, t_n^r)$ are the revisit times of pages $P = (p_0, \ldots, p_n)$.

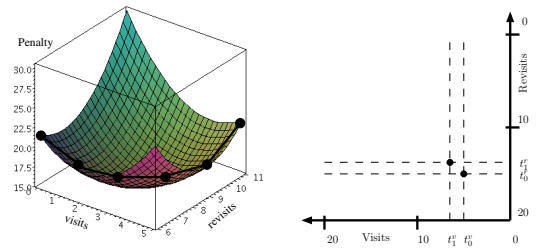
Example 7.2. (Blur with Revisits) *Consider the Web site in Figure 2 with $t_0^v = 0, t_1^v = 1, \ldots, t_5^v = 5$ visit and $t_0^r = 6, t_1^r = 7, \ldots, t_{11}^r = 11$ revisit times. The blur of page p_1 is*

$$\begin{aligned} B(p_1, 1, 7, 5, 1) &= \frac{1 \cdot \omega(1, 7, 5, 1)}{2 \cdot 5 + 1} = 1^2 - \frac{(1+7)^2}{4} + 7^2 \\ &\quad - 7(2 \cdot 5 + 1) + \frac{(2 \cdot 5 + 1)^2}{2} = 35/22. \end{aligned} \tag{21}$$

Similarly, $B(p_0, 0, 6, 5, 1) = 0$, $B(p_2, 2, 8, 5, 1) = 62/22$, $B(p_3, 3, 9, 5, 1) = 93/22$, $B(p_4, 4, 10, 5, 1) = 140/22$, $B(p_5, 5, 11, 5, 1) = 215/22$, and the blur of the archive is

$$B(P, T^v, T^r, 4, 1) = 0 + \frac{35}{22} + \frac{62}{22} + \frac{93}{22} + \frac{140}{22} + \frac{215}{22} \approx 24.77.$$

Analysis of the optimal visits $t_0^v, \ldots, t_n^v$ and revisits $t_0^r, \ldots, t_n^r$ that minimize the blur of the archive in Equations (18)–(20) is similar to the analysis of the optimal download schedule without revisits (cf. Theorem 4.5). Again, we need to schedule all λs ascendingly and all schedule penalties $\omega(p_i, t_i^v, t_i^r, n, \Delta)$ descendingly so the multiplication of them (cf. Equation (18)) minimizes the overall sum. Similarly to the schedule penalty without revisits, the schedule penalty with revisits is an elliptic paraboloid wrt t_i^v, t_i^r with one minimum value (cf. Figure 7(a)). This suggests a strategy to optimize the sharpness. We schedule the visit and revisit of the most changing page at the coordinates of the smallest penalty (t_0^v, t_0^r) (cf. Figure 7(b)). Then we mark all points (t_0^v, t) and (s, t_0^r) as invalid and search the next valid position of the smallest penalty, etc. This results in the visit-revisit pairs on the diagonal parabola in the elliptic paraboloid (cf. filled circles in Figure 7(a)).



(a) Penalty (b) Greedy Approach

Figure 7: Optimization of Archiving with Revisits

Definition 7.3. (SHARC-revisits) *Let $P = (p_0, \ldots, p_n)$ be the Web site such that $\lambda_0 \leq \cdots \leq \lambda_n$. The pair*

$$(t_i^v, t_i^r) = \begin{cases} (\frac{i}{2}, n + 1 + \frac{i}{2}) & \text{if } i \text{ is even and} \\ (n - \frac{i-1}{2}, 2n + 1 - \frac{i-1}{2}) & \text{otherwise} \end{cases}$$

defines the greedy visit and revisit times of page p_i.

Example 7.4. (SHARC-revisits) *Let us continue Example 7.2. The greedy visit and revisit times for pages $p_0, \ldots, p_5$ are $(t_0^v, t_0^r) = (0, 6)$, $(t_1^v, t_1^r) = (5, 11)$, $(t_2^v, t_2^r) = (1, 7)$, $(t_3^v, t_3^r) = (4, 10)$, $(t_4^v, t_4^r) = (2, 8)$, $(t_5^v, t_5^r) = (3, 9)$. The blur of the greedy schedule is approximately 22.59.*

Theorem 7.5. (Greedy download schedule with revisits) *The SHARC-revisits strategy downloads the most changing page at the position with the smallest schedule penalty at each iteration.*

Proof. $(t_{min}^v, t_{min}^r) = \Big(\Big[\frac{2n+1}{4} \Big] \Delta, \Big[3\frac{2n+1}{4} \Big] \Delta \Big)$ minimizes Equation (19). (t_i^v, t_i^r) in Definition 7.3 is the closest (visit, revisit) pair to (t_{min}^v, t_{min}^r). Therefore, SHARC-revisits yield the smallest schedule penalty at each iteration. □

7.2 SHARC-threshold

The easiest way to guarantee page sharpness in the archive is to revisit a page immediately after its visit, simply because this shortens the visit-revisit interval. However, doing this for each page separately does not help towards better archive quality. Instead, we we want to reason on the combined sharpness of all pages in an entire site capture together. If the visit-revisit intervals of the pages in a site have a non-empty intersection, then we have a sharp capture of the complete site and can effectively date the capture with any

timepoint in the overlap of the visit-revisit intervals. This would be the gold standard of archive quality and we could use such a capture in a hard business cases (cf. Section 1).

The following theorem formalizes the conditions for the non-empty overlap of visit-revisit intervals of pages $p_0, \ldots, p_n$.

THEOREM 7.6. (THE MIDDLE POINT) *Let $[t_i^v, t_i^r]$ be the visit-revisit times of p_i. The visit-revisit interval has a non-empty overlap if the middle point of the capture interval is in all visit-revisit intervals $(c_s + c_e)/2 \in [t_i^v, t_i^r]$.*

PROOF. The theorem is proved by contradiction: if one of the intervals does not have the middle point then both visit and revisit times are above (below) the middle point. However, one can find another interval with its visit and revisit times below (above) the middle points, and the intersection of the two is empty. □

Two conclusions can be made from Theorem 7.6. First, all intervals should contain the middle point. Second, the closer the change time to the middle the less chances for all pages to be sharp. In fact, if there is a page with a change at the exact middle of the crawl interval the whole capture cannot be completely sharp.

Analysis of the optimal strategy that maximizes the number of sharp pages is hard and may require assumptions about the distribution of the change time. Ultimately, one should try out all possible visit-revisit intervals ($((n+1)!)^2$ number in total) and choose the schedule that maximizes the number of sharp pages. To cope with the complexity of the problem we use an pyramid organization of the intervals and define a class of promising pages and optimize only them. The pyramid organization places intervals so that the smaller intervals are totally included in the greater ones. This way all intervals are symmetric, are around the middle of the capture, and the shortest interval assigned to the page with the highest change rate. Partitioning of all pages into promising and hopeless (opposite of promising) allows us to sacrifice the download positions of the hopeless pages for a better download positions of the promising ones. The following definition formalizes the SHARC-threshold strategy.

DEFINITION 7.7. (SHARC-THRESHOLD STRATEGY) *Let $p_0, \ldots, p_n$ be the Web site with $\lambda_0 \leq \cdots \leq \lambda_n$. Let τ be the probability threshold. The following defines the visit-revisit intervals iteratively.*

Let p be the hottest page at iteration i such that the probability $P[p$ changes in$[(n-i)\Delta, (n+i+1)\Delta]] = 1-\exp\{-(2(n-i)+1)\Delta\lambda\}$ is less than τ then page p is called promising and its visit-revisit interval is

$$[(n-i)\Delta, (n+i+1)\Delta],$$

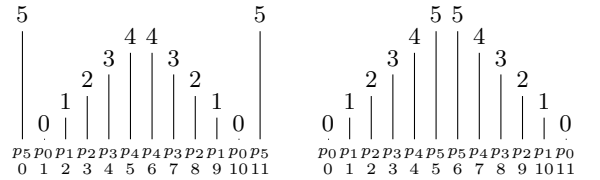
otherwise the page is postponed until the end.

Let $p_0^h, \ldots, p_k^h$ be the set of hopeless pages such that $\lambda_0^h \leq \cdots \leq \lambda_k^h$. Then the visit-revisit interval of p_i^h is $[(k-i)\Delta, (2n-1-k+i)\Delta]$

THEOREM 7.8. (AVERAGE NUMBER OF SHARP PAGES) *Let $p_0^p, \ldots, p_l^p$ and $p_0^h, \ldots, p_k^h$ be the promising and hopeless pages. Let the changes of the pages be distributed according to the independent Poisson processes with $\lambda_0^p \leq \cdots \leq \lambda_k^p$ and $\lambda_0^h \leq \cdots \leq \lambda_l^h$. Let $[k+i+1, 2l+k+2-i]$ be the visit-revisit interval of p_i^p and $[k-j, 2l+k+j+3]$ be the visit-revisit interval of p_j^h of the SHARC-threshold schedule. Then the average number of sharp pages is*

$$\sum_{i=0}^{l} e^{-\lambda_i^p \left(2(l-i)+1\right)\Delta} + \sum_{j=0}^{k} e^{-\lambda_j^h \left(2(l+j)+3\right)\Delta}.$$

PROOF. The proof follows from the properties of the Poisson process. □



(a) p_5 is Hopeless (b) All Pages are Promising

Figure 8: SHARC-threshold

EXAMPLE 7.9. (SHARC-THRESHOLD) *Consider the pages from Figure 2. Let they change in the interval $[0, 11]$ as follows: p_0 does not change, p_1 changes at 1, p_2 at 2 and 9, p_3 at 2, 8, and 10, p_4 at 1, 3, 7, and 11, and p_5 at 2, 3, 4, 5, 6, and 7. Let the visit-revisit intervals follow SHARC-threshold schedule. If the threshold is such that page 5 is a hopeless page and the rest of the pages are promising (c.f. Figure 8(a)), then the number of the sharp pages in 4: 1, 2, 3, and 4. However, if all pages are promising (c.f. Figure 8(b)), then the number of the sharp pages is 1.*

SHARC-threshold increses the number of the sharp pages but increases the blur as well (cf. Example 7.9). SHARC-threshold is similar to the worst case scenario of SHARC-online with k misplacement (cf. Section 6): worst case scenario schedules the hottest and SHARC-threhold schedules the hopeless pages at the ends of the interval. Note though that SHARC-threshold is a revisiting strategy with all pages downloaded twice, on the other hand worst case scenario schedules the pages only once.

8. EXPERIMENTAL EVALUATION

In this section we experimentally evaluate SHARC-offline, SHARC-online, SHARC-revisits, and SHARC-threshold and compare them with the most related techniques: Breadth-first and Depth-first (most typical techniques by archive crawlers), Hottest-first and Coldest-first (most promising naive crawlers), and Olston and Pandey [20] (the best freshness-optimized crawling technique). SHARC-offline requires the knowledge of all URLs of the site in advance. The careful setup of the experiment allowed us to apply this strategy as an ideal baseline. In practice, it is rather unrealistic to assume such knowledge, and SHARC-online should be used instead. Breadth-first and Depth-first schedule the downloads based on the graph (crawl tree) structure of the site. Breadth-first downloads the urls of the pages first and then the urls of the children, while Depth-first visits all urls of the children first and only then the urls of the pages. Hottest-first and Coldest-first sort the detected pages according to the average change rates and schedule the pages in the descending and ascending order respectively. The Olston and Pandey strategy at each step orders the pages

according to the values of their utility functions $U_{p_i}(i)$ and downloads the one with the highest value.

Since including the revisits changes the settings for the experiments, we run two independent sets of experiments: all (relevant) strategies without revisits and all (relevant) strategies with revisits independently. The classical Breadth-first, Depth-first, Hottest-first, Coldest-first and Olston and Pandey do not have revisits but for a fair comparison we added the second run of the strategies to simulate the revisits.

In the experiments without revisits we compute the exact blur (as opposed to the stochastic-average in Sections 4–7). Let $(h_0, \ldots, h_m)$ be the history of changes and t be the download time of page p. Then the exact blur of page p in the observation interval $[o_s, o_e]$ is:

$$B(p) = \frac{1}{o_e - o_s}\Big(\sum_{o_s \le h_j \le t} (h_j - o_s) + \sum_{t < h_j \le o_e} (o_e - h_j)\Big). \quad (22)$$

The exact blur B of the archive $(p_0, \ldots, p_n)$ is the sum of the exact blur of all pages:

$$B(p_0, \ldots, p_n) = \sum_{i=0}^{n} B(p_i).$$

The exact blur with revisits can be extended similarly. Let t_v be the visit and t_r be the revisit time of the page. Then the exact blur with revisits of page p in the observation interval $[o_s, o_e]$ is:

$$B(p) = \frac{1}{o_e - o_s}\Big(\sum_{o_s < h_j \le t_v} (h_j - o_s) + \sum_{t_v < h_j \le \frac{t_v + t_r}{2}} \Big(\frac{t_v + t_r}{2} - h_j\Big) + \sum_{\frac{t_v + t_r}{2} < h_j \le t_r} \Big(h_j - \frac{t_v + t_r}{2} + \sum_{t_r < h_j \le o_e} (o_e - h_j)\Big)\Big). \quad (23)$$

The exact blur with revisits B of the archive $(p_0, \ldots, p_n)$ is the sum of the exact exact blur with revisits of all pages:

$$B(p_0, \ldots, p_n) = \sum_{i=0}^{n} B(p_i).$$

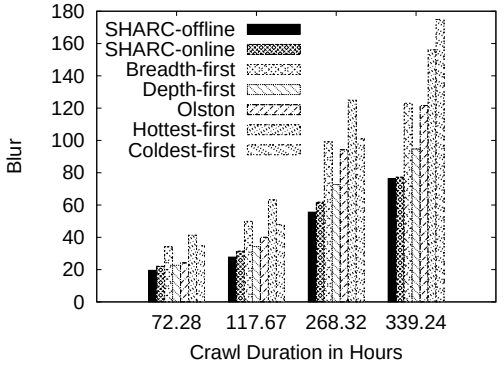
The Olston and Pandey utility function U_p can be expressed in terms of the exact blur. U_p is the exact blur of page p in interval $[o_s, o_e]$ given that the page is downloaded at o_e: $U_p = B(p) = \frac{1}{o_e - o_s} \sum_{j=0}^{m} (h_j - o_s)$. The utility function gives a higher priority to pages with late changes.

We test the approach on a synthetic and a real world datasets. In the synthetic dataset we simulate changes according to Poisson processes with $\{\lambda_0, \ldots, \lambda_n\}$, where $\lambda_i = \lambda_{n-i} = 1/(i^{skew})$. Pages in the graph of the archived site form a tree; each page p_i is pointing to *outdegree* number of children. The default values are $skew = 1.75$, $outdegree = 8$, and $n = 1024$. The spanning tree used by the simulated crawls is such that the pages with high change rates are placed on the top levels. The real world data consists of the pages of the Web site of the Max Planck Institute for Computer Science (daily changes of 60682 pages between 04.09.2008 and 18.09.2008 of the `www.mpi-inf.mpg.de` site).

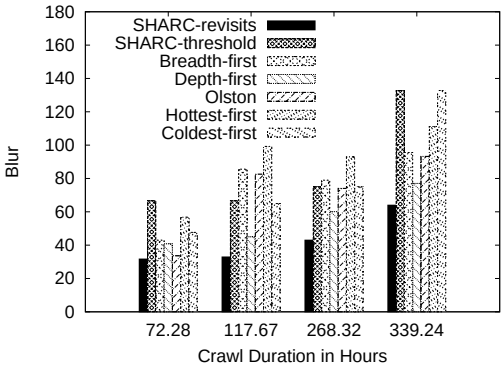
8.1 Impact of the Crawl Duration

We compared the techniques by varying the crawl duration on real world data (cf. Figure 9(a)) in this experiment. Summarizing, SHARC-online outperforms the competitors and it performs as well as SHARC-offline. Short capture intervals encounter the problem that there is not enough data data to estimate the change rates with sufficient accuracy. As a result SHARC-offline does not drastically reduce the blur. With increasing crawl duration and larger amounts of data gathered the problems are overcome and the SHARC-online strategy gets considerable advantage over the competitors.

The blur decreases when revisits are introduced. The greedy SHARC-revisits strategy is the best strategy in all cases (cf. Figure 9(b)). SHARC-threshold optimizes a different metric (the number of sharp pages, cf. Section 7) and results in higher blur.



(a) Crawling Once



(b) Crawling with Revisits

Figure 9: Blur in Varying Crawls

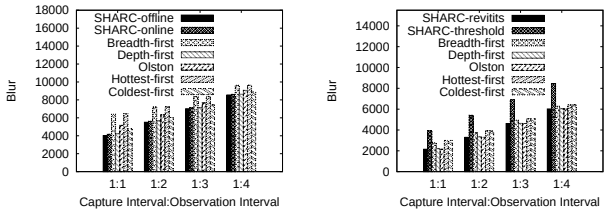
8.2 Impact of the Observation Interval

In this experiment we use the synthetic dataset and vary the observation interval for a fixed crawl interval. We expect that the differences between the techniques disappear as the length of the observation interval increases. The reason are the changes outside the crawl interval which increase the blur substantially for all strategies.

Figure 10(a) confirms our expection. SHARC-offline and SHARC-online outperform other techniques with the largest gain for the same observation and capture interval (cf. Figure 10(a)). The differences between the techniques are less pronounced as the observation interval increases.

The results are similar for the experiment with the revisits (cf. Figure 10(b)). The greedy SHARC-revisits yields the lowest blur, while SHARC-threshold not being designed for blur minimization is outperformed by the other strategies.

Table 4 measures the number of sharp pages for 1:1 case. Here, SHARC-threshold has the highest number of sharp pages, while other techniques perform similarly.



(a) Crawling Once (b) Crawling with Revisits

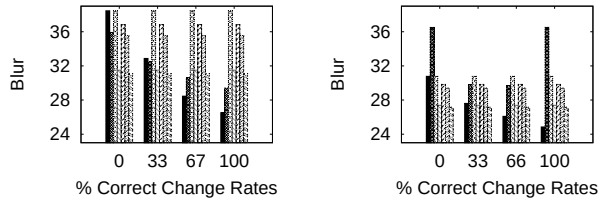
Figure 10: Blur for Observation Periods

Strategy	*##pages*
SHARC-revisits	770
SHARC-threshold	874
Breadth-first	766
Depth-first	776
Olston	782
Hottest-first	765
Coldest-first	776

Table 4: Number of Sharp Pages

8.3 Robustness of SHARC Strategies

In this test we assess the robustness of the SHARC strategies to mis-estimation of change rates λ. In the experiment we used the synthetic dataset and simulated the worst case mis-estimation of change rates for our strategies: first we vary a fraction of mis-estimated pages and then for each such page p_i with λ_i we use the change rate $1 - \lambda_i$. Figures 11(a) and 11(b) show the results. SHARC-online and SHARC-revisits perform better than the competitors if the change rate estimation is accurate for at least 50% of the pages.



(a) Crawling Once (b) Crawling with Revisits

Figure 11: Change Rate Approximation Error (the Legend is the Same as in Figure 10)

9. CONCLUSION AND FUTURE WORK

Data quality is crucial for the future exploitation of Web archives. To this end, the paper defined and investigated two quality measures: blur and number of sharp pages (## pages). The blur measure is appropriate for explorative use of archives. The ##pages measure is appropriate for lawyer-style use of archives. For each of the measures we presented strategies applicable in practice. SHARC-online minimizes the blur and SHARC-threshold maximizes the number of ## pages. The experiments confirm that SHARC-online and SHARC-threshold outperform their competitors.

There are a number of possible research directions. In our stress experiments there was not a single time interval where all pages where sharp. In this case it is a challenge to identify the pages that should be mutually consistent and schedule the downloads of the pages so this consistency is guaranteed. Another research direction is to combine the SHARC-online and SHARC-threshold strategies. This would allow us to have a single archive for both exploration and lawyer style access. Yet another research direction is to develop strategies with multiple revisits or even continuous archiving with a given budget on the number of downloads.

Acknowledgements

This work is supported by the 7^{th} Framework IST programme of the European Union through the small or medium-scale focused research project (STREP) on Living Web Archives (LiWA).

10. REFERENCES

[1] E. Adar, J. Teevan, S. T. Dumais, J. L. Elsas. The Web changes everything: Understanding the dynamics of web content. In *WSDM*, pp. 282–291, 2009.

[2] A. Segev, A. Shoshani, Logical modeling of temporal data. *SIGMOD Rec.*, 16(3):454–466, 1987.

[3] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Murdock, V. Plachoura, F. Silvestri. Design trade-offs for search engine caching. *ACM Trans. Web*, 2(4):1–28, 2008.

[4] B. E. Brewington, G. Cybenko. Keeping up with the changing web. *Computer*, 33(5):52–58, May 2000.

[5] C. Castillo, M. Marin, A. Rodriguez, R. Baeza-Yates. Scheduling algorithms for web crawling. In *WebMedia*, pp. 10–17, 2004.

[6] J. Cho, H. Garcia-Molina. Synchronizing a database to improve freshness. *SIGMOD Rec.*, 29(2):117–128, 2000.

[7] J. Cho, H. Garcia-Molina. Estimating frequency of change. *Trans. Inter. Tech.*, 3(3):256–290, 2003.

[8] J. Cho, H. Garcia-Molina, L. Page. Efficient crawling through url ordering. In *WWW*, pp. 161–172, 1998.

[9] J. Cho, A. Ntoulas. Effective change detection using sampling. In *VLDB*, pp. 514–525. 2002.

[10] J. Cho, U. Schonfeld. Rankmass crawler: a crawler with high personalized pagerank coverage guarantee. In *VLDB*, pp. 375–386. 2007.

[11] L. Colby, A. Kawaguchi, D. Lieuwen, I. Singh Mumick, K. Ross. Supporting multiple view maintenance policies. *SIGMOD Rec.*, 26(2):405–416, 1997.

[12] J. Masanès, editor. *Web Archiving*, Springer, 2006.

[13] T. Härder, A. Bühmann. Value complete, column complete, predicate complete. *The VLDB Journal*, 17(4):805–826, 2008.

[14] R. Klamma, C. Haasler. Wikis as social networks: Evolution and dynamics. In *SNA-KDD*, 2008.

[15] H.-T. Lee, D. Leonard, X. Wang, D. Loguinov. Irlbot: scaling to 6 billion pages and beyond. In *WWW*, pp. 427–436, 2008.

[16] M. Levene, A. Poulovassilis, eds. *Web Dynamics - Adapting to Change in Content, Size, Topology and Use*. Springer, 2004.

[17] G. Mohr, M. Kimpton, M. Stack, I. Ranitovic. Introduction to heritrix, an archival quality web crawler. In *IWAW*, 2004.

[18] M. Najork, J. L. Wiener. Breadth-first search crawling yields high-quality pages. In *WWW*, pp. 114–118, 2001.

[19] A. Ntoulas, J. Cho, C. Olston. What's new on the web?: the evolution of the web from a search engine perspective. In *WWW*, pp. 1–12, 2004.

[20] C. Olston, S. Pandey. Recrawl scheduling based on information longevity. In *WWW*, pp. 437–446, 2008.

[21] C. Olston, J. Widom. Best-effort cache synchronization with source cooperation. In *SIGMOD*, pp. 73–84, 2002.

[22] N. Tolia, M. Satyanarayanan. Consistency-preserving caching of dynamic database content. In *WWW*, 2007.

[23] Debunking the Wayback Machine `http://practice.com/2008/12/29/debunking-the-wayback-machine`