The VLDB Journal (2011) 20:183–207
DOI 10.1007/s00778-011-0219-9

SPECIAL ISSUE PAPER

The SHARC framework for data quality in Web archiving

Dimitar Denev · Arturas Mazeika · Marc Spaniol · Gerhard Weikum

Received: 27 August 2010 / Accepted: 3 February 2011 / Published online: 2 March 2011

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 coherent 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 toward better quality with given resources. We define data quality measures, characterize their properties, and develop a suite of quality-conscious scheduling strategies for archive crawling. Our framework includes single-visit and visit–revisit crawls. Single-visit crawls download every page of a site exactly once in an order that aims to minimize the "blur" in capturing the site. Visit–revisit strategies revisit pages after their initial downloads to check for intermediate changes. The revisiting order aims to maximize the "coherence" of the site capture(number pages that did not change during the capture). The quality notions of blur and coherence are formalized in the paper. Blur is a stochastic notion that reflects the expected number of page changes that a time-travel access to a site capture would accidentally see, instead of the ideal view of a instantaneously captured, "sharp" site. Coherence is a deterministic quality measure that counts the number of unchanged and thus coherently captured pages in a site snapshot. Strategies that aim to either minimize blur or maximize coherence are based on prior knowledge of or predictions for the change rates of individual pages. Our framework includes fairly accurate classifiers for change predictions. All strategies are fully implemented in a testbed and shown to be effective by experiments with both synthetically generated sites and a periodic crawl series for different Web sites.

Keywords Web archiving · Data quality · Blur · Coherence · Crawls strategies

D. Denev (✉) · A. Mazeika · M. Spaniol · G. Weikum
Max Planck Institute for Informatics, Campus E1.4,
66123 Saarbrücken, Germany
e-mail: ddenev@mpi-inf.mpg.de

A. Mazeika
e-mail: amazeika@mpi-inf.mpg.de

M. Spaniol
e-mail: mspaniol@mpi-inf.mpg.de

G. Weikum
e-mail: weikum@mpi-inf.mpg.de

1 Introduction

1.1 Motivation

The Web is in constant flux. Eighty percentage of the pages change within a half a year. To prevent the content from disappearing, national libraries (e.g., www.loc.gov, www.webarchive.org.uk, netarkivet.dk, www.bnf.fr, www.webarchiv.cz, etc.,) 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. The captured Web content 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, these scenarios entail 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 timepoints (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* from photography to denote the quality of the snapshot. Similarly to photography, the longer the exposure time (time span 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 insignificantly) during the crawl, we say that the pages are sharp and the snapshot is coherent (or almost coherent).

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 cannot be fully assured, there should ideally be guarantees about data quality. Consider an analyst who studies a politician's behavior and success during an election campaign based on weekly or daily crawls of the corresponding party's Web site (which may include user-provided contents in associated wikis or blogs). Suppose one page of the site covers a television debate with the politician, pointing to other pages with "opinion barometers" for the politician and her opponents in the debate. Each of these "barometer pages" in turn points to recent public appearances of the featured politician. As these pages frequently change, the archived snapshot contains page versions as of different timepoints. Now, already an incoherence by a few hours difference between the captures of these interrelated pages can lead to misinterpretations and wrong conclusions by the analyst. For example, the analyst may see a brilliant performance of the politician on the debate page and then follow the pointer to a barometer page with unfavorable public opinions, simply because the barometer was captured earlier.

As a real-life case in point, an archive of a Web site was disapproved as evidence in a lawsuit about intellectual property rights [32] because the judge considered the archive as having insufficient quality and no guarantees about the consistency of its content. In such cases, a strategy for getting coherent site captures or precisely stating the level of consistency would make a big difference.

The simplest strategy to obtain a coherent capture of a Web site and avoid anomalies would be to freeze the entire site during the crawl period. This naïve approach 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. Long crawl duration 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 coherent capture and keep repeating it until eventually a blur-free snapshot can be obtained. But these repetitions are 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.

Web site masters can help to make archiving easier by providing additional information about the site: its structure, its typical change patterns, hints about where and when changes occur, and so on. The recently introduced *sitemaps* protocol [35] can provide a list of URLs and metadata about page (or sub-directory-level) modifications so that crawlers can more intelligently process the site. However, sitemaps alone are merely hints that can guide a crawl strategy (e.g., toward pages with high likelihood of having changed so that, for example, a search engine robot should obtain a fresh version). Sitemaps are not a solution for ensuring data quality. The framework developed in this paper integrates sitemaps for quality-conscious Web archiving.

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 provides a framework for quality assurance of Web archives. It develops a model of quality properties as well as a suite of algorithms for crawling that allow us to assess and optimize site-capturing strategies.

Our framework, coined SHARC for Sharp Archiving of Web-site Captures, introduces two measures of data quality for site captures:

– *Blur* is a stochastic notion that reflects the expected number of page changes that a time-travel access to a site capture would accidentally see, instead of the ideal view of a instantaneously captured, "sharp" site. We assume that

the timepoints to which analysts will later refer for snapshot analysis are uniformly distributed over time; hence, the stochastic approach.
– *Coherence* is a deterministic quality measure that counts the number of unchanged and thus coherently captured pages in a site snapshot. Here, "unchanged" denotes pages that are definitely known to be invariant throughout some time window, ideally the entire crawl. The setting allows us to guarantee mutual consistency across several pages in a snapshot that are logically interrelated (e.g., a television debate and the "barometer" pages about the participating politicians).

The SHARC framework needs estimates for the frequencies at which changes occur in a Web site. In line with the prior literature [1,11,30], we model site changes by Poisson processes with page-specific change rates. We show that these rates can be statistically predicted based on page types (e.g., MIME types), depths within the site (e.g., distance to site index pages), and URLs (e.g., manually edited user homepages vs. pages generated by content management systems).

We then devise crawl strategies that aim to optimize our archive quality measures. While stochastic guarantees like blur are good enough for explorative use of the archive (while keeping crawl costs low), access that aims to prove or disprove claims about interrelated contents in site snapshots needs deterministic guarantees like coherence and would accept higher crawl costs. For explorative use of the archive, it is sufficient to visit each page once. The order of the downloads is a degree of freedom for the crawl scheduler. For deterministic guarantees, we introduce 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. For very large Web sites, it is unrealistic to obtain a coherent capture for the entire site. In this case, we opt for smaller subsites of interrelated pages and derive these via the sitemaps protocol.

SHARC provides a suite of novel algorithms for archive crawling:

– *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 minimizes blur and provides stochastic guarantees about data quality.
– *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.
– *SHARC-revisits* visits pages twice aiming to minimize the stochastic blur and allowing change detection during the crawl.
– *SHARC-selective*, similarly to SHARC-revisits, visits pages twice to detect changes during the crawl, ordering the visits, and revisits in such a way that maximizes the coherence. The algorithm automatically detects very hot pages that are almost continuously changing and adjusts the scheduling of visits and revisits so that the other, not so hot, pages have a higher chance of getting coherently captured.

A preliminary version of the SHARC framework has been presented in the conference paper [19]. This paper extends the framework and provides additional contributions: (1) the new SHARC-selective strategy, which can cope with sites that contain very hot pages in a self-adapting manner (making the SHARC-threshold strategy of [19,36] obsolete); (2) a fairly accurate classifier for predicting change rates of pages from features like URL, MIME type, etc.; (3) the integration of sitemaps in our architecture; (4) a full-fledged implementation based on the Heritrix crawler and comprehensive experiments on additional, richer datasets.

Our experimental studies are carried out with both synthetically generated Web sites and repeated crawls of different-sized domains, including mpi-inf.mpg.de (MPII), dmoz.org (DMOZ), and sites from the .uk.gov collection (UKGOV). For change prediction, we use standard machine learning algorithms, such as Naive Bayes and C4.5 classifiers. The experiments demonstrate the practical viability of our approach and the advantages of our algorithms compared to more traditional crawl strategies.

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. Section 4 through 7 present our crawl algorithms: the single-visit offline, the single-visit online, and the visit–revisit strategies that minimize stochastic blur; the visit–revisit strategy minimizes the deterministic coherence metric. Section 8 presents our system architecture, a prototype implementation based on the open source archive crawler Heritrix, and the integration of sitemaps. Section 9 presents our experimental evaluation.

2 Related work

The book on Web archiving [20] 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. [27] 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 coherence. We integrate applicable techniques from the SHARC framework into the Heritrix crawler and use the implementation in the experiments.

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 [21,37], efficiency issues in search engines [4], performance-effective cache synchronization [16,31]. Latest research in the area is caching in cloud databases [18]. 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, and archives should optimize for coherence while the perfect consistency is a prerequisite in data caching.

Crawling the Web for search and indexing received a lot of attention. Key issues here are efficiency [12,25], freshness [7,30], temporal analysis [17,34], importance [28,39], relevance to keyword queries [8,10,13,14], seed selection [2], focused crawling [5], page collections that have large coverage and little redundancy [40]. 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.

Metrics to measure when and how much of the individual pages has been changed have been proposed as well [29,1]. Web change models characterizing the dynamics of Web pages have been developed [26]. Typically, the changes of the page p_i are modeled with a Poisson process [10] 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 iff

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

It is equivalent to postulate that the time between two successive changes of 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. We use standard machine learning techniques to predict change rates with URL features [6,23] and Web page features [33].

Olston and Pandey [30] have designed a crawling strategy optimized for freshness. In order to determine which page to download at timepoint t, Olston and Pandey compute the utility function $U_{p_i}(t)$ for each page p_i based on its full change history. 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 time span. We optimize the coherence of entire captures and not the freshness of individual pages.

Chen et al. [9] investigated how to reduce the overall data transfer without significantly affecting the freshness of the archive. A history of Web archives of a site is assumed, and association rules are computed to predict changes of Web pages and optimize a crawler for freshness of a partial site capture. Our goal is quite different though. We aim to download the *entire* Web site and optimize quality measures like sharpness (absence of blur) or coherence.

3 Web archiving model and terminology

The Web archive, or archive for short, periodically crawls a large number of sites, e.g., on a weekly or monthly basis. 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 must 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 for high throughput.) When a new site crawl starts, we assume that either the URLs of all pages are known upfront (from a previous crawl or a sitemap) 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 offline strategies and is relaxed by the online strategies. 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.

Note that pages may change during a crawl. The longer the crawl duration, the more likely it is that a non-negligible fraction of the site's pages changes once or several times. Figure 1 illustrates this situation for *single-visit* crawls. The timepoints when a page is downloaded are marked by a bullet, the timepoints when a page changes are marked by a cross. The figure shows two crawls, each yielding a separate capture

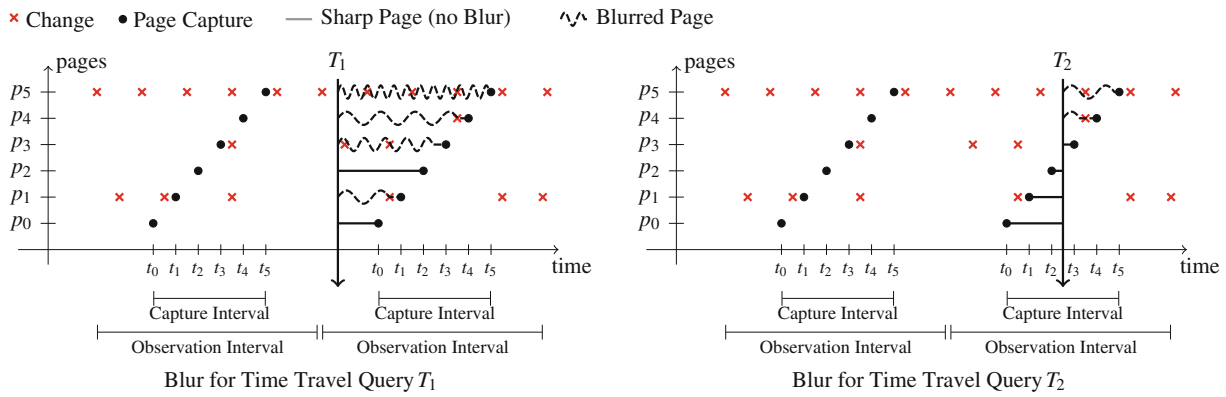


Fig. 1 Sharp versus blurred pages with single-visit crawling

that will be stored in the Web archive for later retrieval by analysts and other users. Note that the figure shows *"blurred" captures*, as the page versions in a capture refer to to different states of the Web site.

The archive is accessed by *time-travel queries*, asking for the site *as of* a given timepoint of interest to the user. Figure 1 shows two such requests, denoted by vertical arrows, for timepoints T_1 and T_2 which fall before the capture and the interval between page-capturing timepoints t_2 and t_3, respectively. The archive may not have versions of the pages as of the exact requested time. The user's request for time T is then 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). This mapping defines for each capture an *observation interval*: the capture is returned for all time-travel queries that fall into the observation interval. Figure 1 shows a possible choice for the observation intervals of the two captures. Observation intervals based on the most recent available capture correspond to the simplest standard semantics in temporal database systems [3]. Observation intervals based on the closest capture may appear non-standard, but make 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 individual time-travel queries, some of the pages are "blurred" and some are "incoherent". By *blur*, we refer to the magnitude of change in the timespan between the query timestamp and the time of the actual page capture. The farther the two points are apart, and the more the page changes (on average), the more blurred the page would appear to the user. For example, if the time-travel request is for May 15, a page that was captured on May 10 would appear less blurred than a page that was captured on May 1 or May 31. Figure 1 illustrates this notion of blur by waving lines (more waving means higher blur). We will formalize this measure in Sect. 4. By *incoherence*, we refer to the page changes between the query timestamp and the page capture. If a page has changed between these two timepoints, it may appear incoherent with respect to other pages. Conversely, if we manage to capture a set of potentially interrelated pages such that there is no change at all between their capturing points and the timestamp of user request, then this set would be perfectly coherent. The pages jointly appear as if they were instantaneously captured in the very same state of the Web site. Figure 1 illustrates this notion of coherence by solid lines. Pages without any changes in the critical timespan are coherent and indicated by solid lines. We will formalize the notion of coherence in Sect. 7.

In the SHARC framework, we want to either minimize the blur or maximize the coherence of captures. This goal entails two difficulties:

- First, the timepoints of the users' time-travel queries are not known at the time of crawl.
- Second, the exact timepoints of page changes are not known. We may not know whether there was a page change between the timestamps of a user's access request and a page capture.

To overcome the first difficulty, we assume that the timestamps of user requests are uniformly distributed in the observation interval, and we will aim for a stochastic notion of archive quality.

To overcome the second difficulty, we visit each page twice, where the second visit serves to check for changes. This *visit–revisit* approach is illustrated in Fig. 2. The figure shows a strategy where all the revisits of pages follow all visits. Even if HTTP protocol information does not reliably indicate whether a page is modified or not, we can now easily compare two versions of the same page and test for invariance

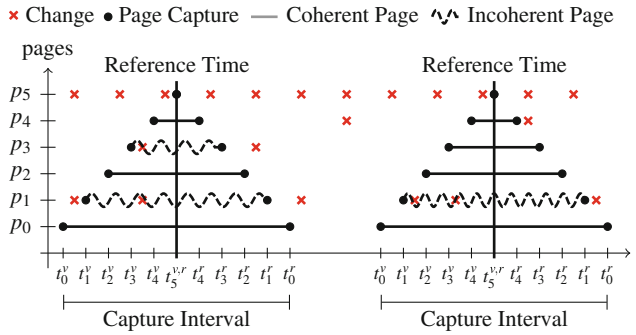
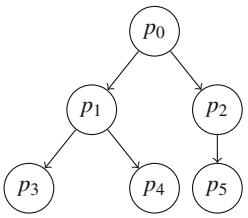


Fig. 2 Coherent versus incoherent pages with visit-revisit crawling

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

Fig. 3 Example of a Web site with change rates

(perhaps ignoring insignificant changes such as banner ads or auto-generated footers). If for a given set of pages (ideally the entire site), none shows any changes between visit and revisit, then this entire set is coherent—as if it were instantaneously captured in the middle of the crawl—denoted as "reference time" in the figure. In this paper, we will consider only strategies with this two-phase structure: all revisits following all visits. Note, however, that the order in which individual pages are visited in each phase is an important degree of freedom in optimizing the crawl schedule.

Single-visit strategies have lower crawl cost than visit–revisit strategies and are good enough to minimize the stochastic notion of blur. However, an archive user cannot be sure if an individual page is sharp or not (i.e., did not or did change between query timestamp and page capture). This uncertainty is not a problem for exploratory usage of the archive. However, in use cases where we need to be sure that an archived snapshot reflects a Web site as of a specific time in the past (e.g., for legal purposes), we need to employ the visit–revisit method.

In devising suitable strategies for scheduling the visits of a site's pages, we need to have some information about how often, and perhaps even when, a page typically changes. To this end, we employ statistical prediction models. Following the state of the art [10,15,24,38], we assume that pages undergo changes according to a *Poisson process* with *change rate* λ. We can then train a classifier to predict the specific rate of a page, based on features of the page: its MIME type, depth in the site graph relative to the entry point, URL string, and so on. The classifier allows us to predict the probability or "risk" that there will be a change of a page in the timespan between capturing it and a time-travel access, and also the expected number of changes in that interval or a quantile for the number of changes.

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 as of now). In contrast, Web archives should capture all (or specifically selected) pages (at regular crawl timepoints) of the entire site and return the most appropriate version of a page (or page set) for a given time-travel query. Optimization criteria for crawling are also different. Search engines optimize for up-to-date indexes that reflect the freshest version of the important pages. In contrast, Web archives aim to obtain and present *coherent* snapshots as of the requested timepoint.

4 SHARC-offline

In this section, we establish the stochastic metric blur for a given site capture 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 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 change rates $\lambda_0, \ldots, \lambda_n$ (the mean number of changes in a time unit). For ease of presentation, we assume that the identifiers (subscripts) of the pages are chosen so that $\lambda_0 \leq \cdots \leq \lambda_n$. We denote the least frequently changing page p_0 as the *coldest page*, and the most frequently changing page p_n as the *hottest page*. We assume that the download timepoints of the pages are equidistant with politeness delay Δ in between the downloads (the most typical scenario). To simplify mathematical expressions, we assume that the crawl 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 Sect. 4.4, we generalize the equations and omit the assumption.

Figure 3 presents an example that we use throughout the section. Change rate $\lambda = 0$ means virtually no changes for p_0. Strictly mathematically, this is not possible, since Poisson model requires $\lambda > 0$. Here, we assume a negligible small change rate $0 < \varepsilon \ll 1$.

4.1 Blur

The blur of a page and the blur of an entire site capture are key measures for assessing the quality of a one-visit Web archive.

Definition 1 (BLUR) Let p_i be a Web page captured 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 captured at times $T = (t_0, t_1, \ldots, t_n)$. The *blur of the archived capture* is the sum of the blur values 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 indicates how many expected changes the explorer of the archive sees if she visits all the pages in the archive. We define the average blur as the average number of the expected changes explorer sees per page:

$$\bar{B}(P, T, n, \Delta) = \frac{1}{n(n+1)\Delta}\sum_{i=0}^{n} \lambda_i \omega(t_i, n, \Delta). \quad (4)$$

The blur of a Web page in the capture is the product of its 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 the penalty of downloading page p_i at time t_i.

Example 1 (Blur) Consider the Web site in Fig. 3 with download time t_i for page p_i (for example page p_3 is downloaded at time $t_3 = 3$). The blur of p_1 is

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

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

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

The average blur per page is $\bar{B}(P, T, 5, 1) \approx 4.58$.

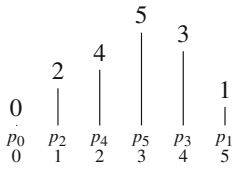
Properties of the download schedule penalty allow us to reason about the blur of the capture for different delay intervals.

Theorem 1 (PROPERTIES OF THE SCHEDULE PENALTY) *The blur is proportional to download delay Δ, i.e.,*

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

Proof The proof follows from the definitions of schedule penalty, blur, and quadratic function of penalty. □

Fig. 4 Organ-pipes arrangement



4.2 Optimal download schedule

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

Example 2 (OPTIMAL DOWNLOAD SCHEDULE) Consider again Example 1. The optimal download schedule is $t_0 = 0$ and $t_1 = 5$ (the outermost points of the interval) for the coldest (least changing) pages p_0 and p_1, $t_2 = 1$ and $t_3 = 4$ (the next outermost points) for the second and third least changing pages p_2 and p_3, followed by $t_4 = 2$ and $t_5 = 3$ for the hottest pages p_4 and p_5. The blur of the capture with optimal download schedule is $B(P, T', 5, 1) = 22.7$.

Figure 4 illustrates the optimal download schedule where the change rate of the scheduled download is visualized as a line of length proportional to the change rate. The visualization resembles an *organ-pipes arrangement* with the highest pipes allocated in middle.

Theorem 2 (OPTIMAL DOWNLOAD SCHEDULE) *Let p_0, $p_1, \ldots, p_n$ be the Web site such that $\lambda_0 \le \lambda_1 \le \cdots \le \lambda_n$. 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 2 is based on three observations:

(i) λ_i are ordered increasingly: $\lambda_i \le \lambda_{i+1}$,
(ii) Equation (6) orders $\omega(t_i, n, \Delta)$ decreasingly: $\omega(t_i, n, \Delta) \ge \omega(t_i, n, \Delta)$,
(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 in 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 1 below. □

Lemma 1 *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*

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

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 Eq. (7).

The largest element of λs must be multiplied 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 schedule; 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 Eq. (7). Indeed, the sum without the swap is as follows:

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

The sum with the swap is as follows:

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

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

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

The left-hand side (LHS) of Eq. (10) is as follows:

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

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 notion of blur and the optimal download schedule for the case when the observation interval $[o_s, o_e]$ does not coincide with the capture interval. Then, the *blur of a 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 schedule penalty*, and the *blur of the archived capture* is the sum of the blur values of the individual pages (cf. Eq. (3)).

Theorem 2 schedules the hottest 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 following theorem.

Theorem 3 *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.*

When the observation interval and the capture intervals are fixed, the hottest pages should be allocated as close as possible to the middle point of the observation interval.

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 blur is minimal. We organize this section as follows. First, we explain the incremental detection of the Web site structure and discuss most common crawl strategies in Sect. 5.1. We develop the SHARC-online strategy by example in Sect. 5.2. Finally, we formally define the SHARC-online strategy and present the algorithm of the strategy in Sects. 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 Downloaded-Detected lists (DD-lists) of URLs. The downloaded list of URLs consists of all URLs that are already crawled, while the detected list comprises 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 (DFS)* and *breadth-first (BFS)* on the example Web graph in Fig. 3.

Table 1 depicts the detection and downloads 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-lists. The depth-first strategy inserts newly detected pages at the beginning of the detected list; thus, the newly detected 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 early detected pages (cf. Table 2).

Table 1 Depth-first crawl strategy (DFS)

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 2 Breadth-first crawl strategy (BFS)

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 3 SHARC-online 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_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	$p_0, p_1, p_4, p_3, p_2, p_5$	|

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.

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 arrangement. 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-lists for downloading. 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.

The SHARC-online strategy maintains the list of detected pages $\left(p_0^E, p_1^E, \ldots, p_{n^E-1}^E\right)$ (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 coldest-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 3 (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 $\left(p_0^E, p_1^E\right) = (p_1, p_2)$. 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.

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

Example 4 (MIDDLE PHASE) Lets continue Example 3 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

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

Since

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

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) number of pages. During this phase, the algorithm implements the hottest-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 arrangement and until all detected URLs are downloaded:

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

Example 5 (Descending Phase) Lets continue Example 4 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 $(p_0, \ldots, p_m)$,
estimated size of the crawl n
output: download schedule $(p_0^D, \ldots, p_n^D)$

1 **begin**
2 $\quad \mathbf{P}^D = (p_0^D, \ldots, p_{n^D}^D) = (), n^D = 0$
3 $\quad \mathbf{P}^E = (p_0^E, \ldots, p_{n^E}^E) = (p_0, \ldots, p_m), n^E = m$
4 $\quad$ **while** $\mathbf{P}^E \neq \emptyset$ **do**
5 $\quad\quad$ **if** $n^D + n^E \leq (n+1)/2$ **then** $pos = 0$
6 $\quad\quad$ **else if** $n^D \leq (n+1)/2$ **then**
7 $\quad\quad\quad$ $pos = n^D < n^E ? p_{n^D}^E : p_{n^E-1}^E$
8 $\quad\quad$ **else** $pos = n^E$
9 $\quad\quad$ $\texttt{append}(\mathbf{P}^D, p_{pos}^E), \texttt{remove}(\mathbf{P}^E, p_{pos}^E)$
10 $\quad\quad$ $\texttt{add\_sort}(\mathbf{P}^E, \texttt{urls}(p_{pos}^E))$
11 $\quad\quad$ $n^D{+}{+}, n^E{-}{-}, n^E = n^E + |\texttt{urls}(p_{pos}^E)|$
12 $\quad$ **end**
13 **end**

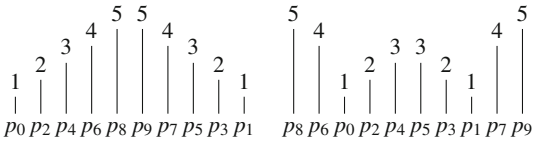
Algorithm 2: SHARC-online

6 Worst-case analysis

In this section, we investigate the worst-case scenario for the SHARC-online strategy. In SHARC-offline, the hottest pages are scheduled in the middle of the crawl interval. Since SHARC-online does not possess the full knowledge about the URLs of the site, it may download pages at positions different from those of the SHARC-offline strategy. To analyze 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 if we placed the k hottest pages in the k outermost positions. Example 6 illustrates this case.

Example 6 (WORST- CASE BLUR) 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 Fig. 5a with the worst-case scenario in Fig. 5b.

The highest schedule penalty positions in the crawl are the first and the last download slots: $\omega(0, 10, 1) = \omega(9, 10, 1) = 40.5$, and downloads of the hottest pages (p_8 and p_9 with $\lambda_8 = \lambda_9 = 5$) at these positions maximize 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 hottest pages p_6 and p_7 are scheduled there. The remaining positions are



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

Fig. 5 Example of worst-case blur scenario with $n = 9$ and $k = 4$

scheduled according to SHARC-offline strategy resulting in an organ-pipes-like middle part of the download schedule.

The increase in 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 $2 \cdot \sum_{i=0}^{4} i(i) = 755$, and the relative increase in the blur of the worst case is $145/755 \approx 20\%$.

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

Theorem 4 *Let the number of pages in the site $n+1$ be even with such change rates of the pages: $\lambda_0 = \lambda_1 \leq \lambda_2 = \lambda_3 \leq \cdots \leq \lambda_{n-1} = \lambda_n$. 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:*

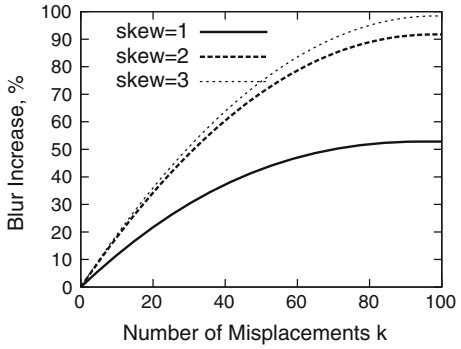
$$2 \sum_{i=0}^{k/2-1} (\lambda_{n-2i} - \lambda_{2i})\omega(i\Delta, n, \Delta) - 2 \sum_{i=0}^{(n+1-k)/2} (\lambda_{k+2i} - \lambda_{2i})\omega\left(\left(\frac{k}{2} + i\right)\Delta, n, \Delta\right). \quad (17)$$

Proof The proof follows from similar arguments as in Theorem 2. □

The actual increase in the blur that SHARC-online exhibits over SHARC-offline depends on the characteristics of the Web site, especially on the size of the site (number of pages) and the skew in the distribution of the pages' change rates.

Skew has the larger impact, as illustrated in (Fig. 6a). Here, we modeled the skew with $\lambda_{2i-1} = \lambda_{2i} = 100/i^{skew}, i = 0, \ldots, n/2, n = 101$. The increase in the skew by one increases the blur by an order of magnitude. The misplacement of the hottest pages incurs the highest amount of additional blur (see the steep increase for k =0–10 in Fig. 6a). As the more and more pages are misplaced ($k = 20, \ldots, 100$), the increase slows down.

Increasing the size $n+1$ of the Web site, illustrated in Fig. 6b, leads to high additional blur for small n (cf. $n = 10^1$



(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

Fig. 6 Worst-case blur for different values of skew and size

and $n = 10^2$ in the figure), while further increasing the number of pages changes the resulting blur only slightly ($n = 10^2$ and $n = 10^3$ in the figure).

7 Visit–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 approach allows us to deterministically reason about the state of the archive as of the middle point of the crawl. We call a page *coherent* if its versions at visit time and revisit time are identical. The coherence of an entire site capture is the number of coherent pages. In the following, we present two strategies: SHARC-revisits, which aims at minimizing blur, and SHARC-selective which aims at maximizing coherence.

Definition 2 (COHERENCE) A page in a site capture is coherent if it did not change between its visit and revisit. The coherence of a site capture is the number of pages in the archived capture that did not change between their visit and revisit timepoints.

7.1 SHARC-revisits

Given Web pages $p_0, p_1, \ldots, p_n$ and the change rates $\lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_n$, the task is to find the timepoints (download slots) for the initial visits $t_0^v, t_1^v, \ldots, t_n^v$ and for the revisits $t_0^r, t_1^r, \ldots, t_n^r$ such that the blur of the site capture 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 3 (BLUR OF PAGE WITH REVISITS.) Let p_i be a Web page with visit time t_i^v and revisit time t_i^r. The *blur of page* p_i is

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

where

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

is the download schedule penalty for downloads with revisits. The *blur of an archived capture* with revisits is the sum of the blur values of the individual pages:

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

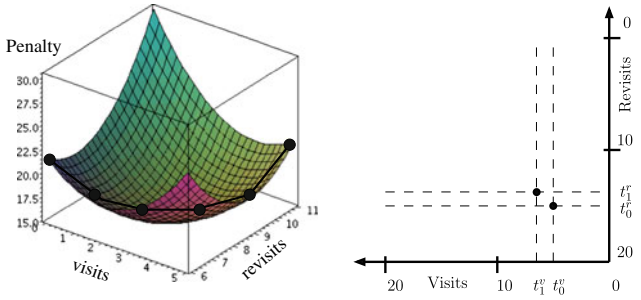
where $T^v = \left(t_0^v, \ldots, t_n^v\right)$ are the visit, and $T^r = \left(t_0^r, \ldots, t_n^r\right)$ are the revisit times of pages $P = (p_0, \ldots, p_n)$. The average blur is $\bar{B}\left(P, T^v, T^r, n, \Delta\right) = 1/n \sum_{i=0}^{n} B\left(p_i, t_i^v, t_i^r, n, \Delta\right)$.

Example 7 (BLUR WITH REVISITS) Consider the Web site in Fig. 3 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

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

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

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



(a) Penalty **(b)** Greedy Approach

Fig. 7 Optimization of crawling with revisits

The derivation of the optimal visits $t_0^v, \ldots, t_n^v$ and revisits $t_0^r, \ldots, t_n^r$ for minimum blur is similar to the analysis of the optimal download schedule without revisits (cf. Theorem 2).

Again, we need to schedule pages in ascending order of λ values and descending order of download schedule penalties $\omega(p_i, t_i^v, t_i^r, n, \Delta)$, so that the product of these factors minimizes the overall sum in Eq. 18. Similarly to the download schedule penalty without revisits, the penalty with revisits is always an elliptic paraboloid w.r.t. t_i^v, t_i^r, with one minimum (cf. Fig. 7a). The equation suggests the following strategy toward minimizing blur. We schedule the visit and revisit of the hottest page in the download slots with the smallest penalty (t_0^v, t_0^r) (cf. Fig. 7b). Then, we mark all points (t_0^v, t) and (s, t_0^r) as invalid and search the next valid position with smallest penalty, and so on. This strategy results in visit–revisits forming a diagonal line in the visit–revisit plane (cf. filled circles in Fig. 7a.

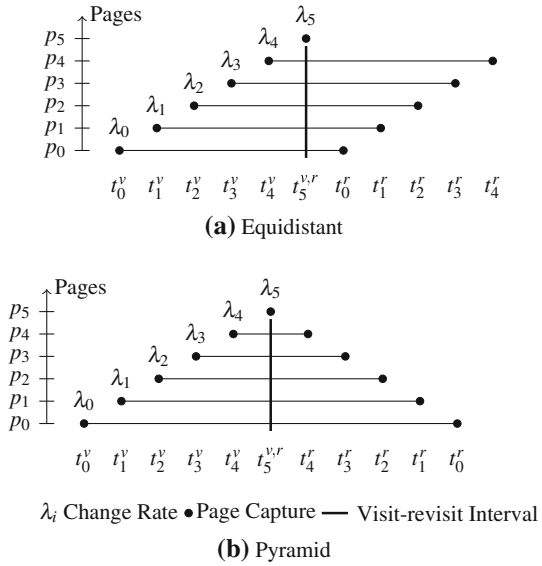
The strategy is greedy: at each step, we aim to assign the hottest change rate at the lowest penalty position. While the strategy yielded an optimum in the single-visit case, this is not necessarily the optimum for the visit–revisit case. To obtain an optimum schedule, we would need to scan all possible parabola of visits–revisits in the elliptic paraboloid and check for which the sum of the factors of change rates and penalty positions yield the smallest blur.

Definition 4 (SHARC- REVISITS) Let $P = (p_0, \ldots, p_n)$ be the Web site such that $\lambda_0 \leq \cdots \leq \lambda_n$. The pair

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

defines the *greedy strategy* for the visit and revisit times of page p_i.

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



(a) Equidistant

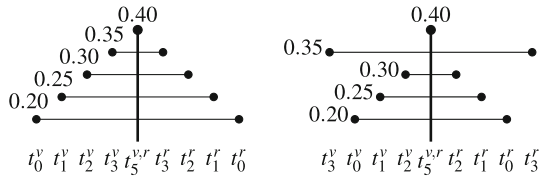
(b) Pyramid

Fig. 8 Shapes of schedules

7.2 SHARC-selective

Overview of the analysis and outline. SHARC-selective schedules all visits before the revisits so that intervals between visit and revisit have a non-empty intersection. With this approach, the ideal outcome would be that all pages are mutually coherent if they individually did not change between their visits and revisits.

Mathematical analysis of the optimal strategy that maximizes coherence is hard. Ultimately, one needs to try out all possible schedules of visit–revisit intervals ($((n+1)!)^2$ in total) and opt for the strategy that has the highest *expected coherence*. Two extreme choices of visit–revisit intervals are equidistant schedule where all intervals have the same lengths, as shown in Fig. 8a), and pyramid-like schedule, shown in Fig. 8b, where the intervals are centered around a joint "axis". To reduce the complexity of the problem, we consider only the family of *pyramid-like* visit–revisit schedules centered around the middle point. The rationale behind this choice is the higher expected coherence for pyramid-like compared to equidistant schedule where the change rates of the pages are the same. Allocation of pages (change rates) to the intervals is the degree of freedom of the SHARC-selective algorithm. Intuitively, one could allocate the hottest pages to the shortest intervals greedily maximizing each page expected coherence. However, in certain cases, it is better to "give up" extremely hot (*hopeless*) pages, by assigning them to longer visit–revisit intervals so that other (*hopeful*) pages get shorter visit–revisit intervals and, therefore, have higher chances of getting coherently captured.



(a) Hottest to Shortest **(b)** $\lambda_3 = 0.35$ is Hopeless

Fig. 9 Example with hopeless pages

In summary, SHARC-selective employs three principles:

1. Visit–revisit intervals form a pyramid.
2. Greedily assign the hottest hopeful pages to the shortest intervals.
3. Greedily assign the hottest hopeless pages to the longest intervals.

We organize the section in the following. First, we define the expected coherence of a schedule. Second, we show that for the same change rates, pyramid is better than equidistant schedule. Third, we define the hopeless pages and give the SHARC-selective offline and online algorithms.

Expected coherence of a schedule is the key concept to define hopeless page and SHARC-selective schedule.

Definition 5 (EXPECTED COHERENCE OF A SCHEDULE) Let $\lambda_i, \ldots, \lambda_k$ be the change rates scheduled so the lengths of their visit–revisit intervals are $I_i, \ldots, I_k$. Then, the expected coherence is

$$EC\left((\lambda_i, I_i), (\lambda_{i+1}, I_{i+1}), \ldots, (\lambda_k, I_k)\right) = e^{-\lambda_i I_i} + \cdots + e^{-\lambda_k I_k}.$$

Example 9 (EXPECTED COHERENCE OF A SCHEDULE) Consider the schedule in Fig. 9a. There the change rates 0.40, 0.35, 0.30, 0.25, 0.20 are scheduled on the intervals 0, 2, 4, 6, 8 (hottest to shortest), and the expected coherence is

$$EC\left((0.40, 0), (0.35, 2), (0.30, 4), (0.25, 6), (0.20, 8)\right) = e^{-0.40 \cdot 0} + e^{-0.35 \cdot 2} + e^{-0.30 \cdot 4} + e^{-0.25 \cdot 6} + e^{-0.20 \cdot 8} \approx 2.22.$$

Therefore, expected coherence is 2.22 pages (out of maximum five coherent pages).

Pyramid is better than equidistant schedule in terms of expected coherence for the case when all pages have the same change rates. Below we formalize this result.

Theorem 5 (PYRAMID IS BETTER THAN EQUIDISTANT) *Let $\lambda_0 = \cdots = \lambda_n = \lambda$. Then, the expected coherence is higher for the pyramid schedule compared to the equidistant schedule for large enough $n (n \geq 1 + 1/(e^{\lambda\Delta} - 1))$.*

Proof The proof is by induction. Assume that the theorem is true for the schedules of length n:

$$\begin{aligned}
&\text{expected-coherence(pyramid}(n)) \\
&\quad = EC\left((\lambda, 0) \ldots, (\lambda, 2(n-1))\right) \\
&\quad = 1 + e^{-2\lambda\Delta} + e^{-4\lambda\Delta} + \cdots e^{-2(n-1)\lambda\Delta} \\
&\quad \geq 1 + (n-1)e^{-n\lambda\Delta} \\
&\quad = EC\left((\lambda, n) \ldots, (\lambda, n)\right) \\
&\quad = \text{expected-coherence(equidist}(n)).
\end{aligned}$$

We need to prove that

$$\begin{aligned}
&\text{expected-coherence(pyramid(n+1))} \\
&\quad \geq \text{expected-coherence(equidist(n+1)).}
\end{aligned}$$

Since

$$\begin{aligned}
&\text{expected-coherence(pyramid(n+1))} \\
&\quad = \text{expected-coherence(pyramid(n))} + e^{-2n\lambda\Delta}
\end{aligned}$$

and

$$\begin{aligned}
&\text{expected-coherence(equidist(n))} \\
&\quad = \text{expected-coherence(equidist(n-1))} \\
&\qquad + ne^{-(n+1)\lambda\Delta} - (n-1)e^{-n\lambda\Delta},
\end{aligned}$$

it suffices to show that

$$e^{-2n\lambda\Delta} \geq ne^{-(n+1)\lambda\Delta} - (n-1)e^{-n\lambda\Delta}.$$

This follows from the fact that

$$ne^{-(n+1)\lambda\Delta} \leq (n-1)e^{-n\lambda\Delta}. \tag{22}$$

Indeed, taking the logarithm of both sides in Eq. (22):

$$\begin{aligned}
&\log(n) - (n+1)\lambda\Delta \leq \log(n-1) - n\lambda\Delta \\
&\quad \Leftrightarrow \log(n) - \log(n-1) \leq \lambda\Delta,
\end{aligned}$$

which is true for all large enough n. □

A hopeless page is such an extremely hot page that it pays off to sacrifice the page and assign a long visit–revisit interval to it in order for the other pages to enjoy shorter intervals and increase the overall expected coherence of the capture.

Definition 6 (HOPELESS PAGE) Let $\lambda_i \leq \cdots \leq \lambda_k$ be the change rates and $I_i < \cdots < I_k$ be the lengths of the visit–revisit intervals. Page p_i is hopeless (change rate λ_i is hopeless) iff

$$\begin{aligned}
&EC\left((\lambda_k, I_i), (\lambda_{k-1}, I_{i+1}), \ldots, (\lambda_i, I_k)\right) \\
&\quad = e^{-\lambda_k I_i} + e^{-\lambda_{k-1} I_{i+1}} + \cdots + e^{-\lambda_i I_k} \\
&\quad \leq e^{-\lambda_{k+1} I_i} + e^{-\lambda_{k+2} I_{i+1}} \ldots + e^{-\lambda_i I_{k-1}} + e^{-\lambda_k I_k} \\
&\quad = EC\left((\lambda_{k-1}, I_i), (\lambda_{k-2}, I_{i+1}), \ldots, (\lambda_i, I_{k-1}), (\lambda_k, I_k)\right).
\end{aligned}$$

Otherwise, we call page p_i (change rate λ_i) hopeful.

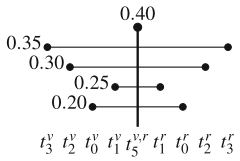


Fig. 10 SHARC-selective schedule for change rates in Fig. 9a

Example 10 (HOPELESS PAGE) Let $\lambda_1 = 0.20, \lambda_2 = 0.25, \lambda_3 = 0.30, \lambda_4 = 0.35$ and $I_1 = 2, I_2 = 4, I_3 = 6, I_4 = 8$ (cf. Example 9). Then, change rate λ_1 is hopeless. In order to verify this statement, we need to compare

$$EC\left((0.35, 2), (0.30, 4), (0.25, 6), (0.20, 8)\right) \approx 1.22$$

(expected coherence for schedule in Fig. 9a) with

$$EC\left((0.30, 2), (0.25, 4), (0.20, 6), (0.35, 8)\right) \approx 1.27$$

(the expected coherence for schedule in Fig. 9b). Since $2.22 < 2.27$, therefore, λ_3 is hopeless.

Algorithms. The SHARC-selective (offline) employs the three principles to schedule the pages: (i) pyramid intervals, (ii) hottest hopeful pages to shortest interval, and (iii) hottest hopeless pages to the longest intervals.

Example 11 (SHARC- SELECTIVE.) Let us continue Examples 9 and 10.

Any page allocated the zero-length interval is always hopeful. Therefore, $\lambda_5 = 0.4$ is hopeful and allocated at the zero-length interval. This completes the first iteration.

Example 9 shows that $\lambda_3 = 0.35$ is a hopeless page for change rates $(\lambda_3, \lambda_2, \lambda_1, \lambda_0) = (0.35, 0.30, 0.25, 0.20)$ and intervals $(I_1, I_2, I_3, I_4) = (2, 4, 6, 8)$ and is allocated for interval $I_4 = 8$. This completes the second iteration.

Since

$$EC\left((\lambda_2, I_1), (\lambda_1, I_2), (\lambda_0, I_3)\right) \approx 1.21 < 1.22 \approx EC\left((\lambda_1, I_1), (\lambda_0, I_2), (\lambda_3, I_3)\right)$$

therefore λ_2 is also hopeless and is scheduled for the second largest interval I_3. This completes the third iteration.

Since

$$EC\left((\lambda_1, I_1), (\lambda_0, I_2)\right) \approx 1.06 > 1.04 \approx EC\left((\lambda_2, I_1), (\lambda_1, I_2)\right),$$

therefore, λ_1 is a hopeful change rate and is allocated for the shortest interval I_1. This allocates the last (hopeful) change rate λ_2 to I_2. The schedule is illustrated in Fig. 10.

Algorithm 3 presents the SHARC-selective offline algorithm. We check the page whether it is hopeless or hopeful, appending it to the corresponding queue (cf. Lines 5–6). Once all hopeless pages $\mathbb{H}$ and hopeful pages $\mathbb{F}$ are identified, we visit all the hopeless pages $\mathbb{H}$ to allocate the longest intervals to $\mathbb{H}$. Then, hopeful page follows and gets shorter intervals. The revisits are in the reverse order.

```
   input  : sorted pages p_0, ..., p_n
            sorted intervals I_0, ..., I_n
   output: visit–revisit schedule p_0^v, ..., p_n^v, p_0^r, ..., p_n^r
            hopeless pages ℍ
            hopeful pages 𝔽
 1 Initialize hopeless pages ℍ = ()
 2 Initialize hopeful pages 𝔽 = ()
 3 begin
 4     for i = 0, 1, ..., n do
 5         if EC((λ_{n−i}, I_{|𝔽|}), ..., (λ_0, I_{n−|ℍ|}))) ≥
           EC((λ_{n−i−1}, I_{|𝔽|}), ..., (λ_{n−1}, I_{n−|ℍ|}))) then
           𝔽 = 𝔽 ∪ p_i
 6         else ℍ = ℍ ∪ p_i
 7     end
 8     visit ℍ; visit 𝔽
 9     revisit 𝔽; revisit ℍ
10 end
```

Algorithm 3: SHARC-selective Offline

SHARC-selective offline schedules the pages from hottest to coldest around the middle point of the crawl. Consequently, the first Web page in the visit–revisit schedule is either the first detected hopeless page or the coldest page if there are no hopeless pages.

To turn the offline strategy into an online strategy, instead of scanning the pages from hottest to coldest, we scan from the coldest page to the hottest one. The strategy detects on-the-fly new Web pages and schedules the visits of the hopeless pages as early as possible (so these pages get the longest available visit–revisit intervals). This is shown in Algorithm 4. We start with the set of seeds (Line 2) and identify hopeful and hopeless pages. If there are hopeless pages (Line 4), we download the hottest hopeless page so it gets the longest visit–revisit interval (Line 5–6). Otherwise, we download the coldest hopeful page (we are at the bottom of the pyramid). Then, we detect new pages and again identify the hopeful and hopeless pages (Line 11). The process is continued until all pages are downloaded. The algorithm concludes with downloads of all pages according to their scheduled revisits.

8 Prototype implementation

8.1 Architecture

We have implemented all SHARC strategies in an experimental testbed, and we have integrated selected SHARC strategies into the Heritrix archive crawler [27]. The SHARC prototype consists of three main components: scheduler, multi-threaded downloader, and database. The *scheduler* dispatches pages for downloading, driven by configurable options for our SHARC strategies. In the original Heritrix crawler, the scheduling is based on a breadth-first strategy;

input : sorted seeds $(p_0, \ldots, p_m)$
predicted Web size n
output: visit–revisit sequence $(p_0^v, \ldots, p_n^v, p_0^r, \ldots, p_n^r)$

```
1  begin
2      (ℍ, 𝔽)=Selective-Offline (p_0, ..., p_m, 2m, ..., 2n)
3      while 𝔽 ≠ ∅ and ℍ ≠ ∅ do
4          if ℍ ≠ ∅ then
5              Download the hottest p ∈ ℍ
6              ℍ = ℍ \ p
7          else
8              Download the coldest p ∈ 𝔽
9              𝔽 = 𝔽 \ p
10         end
11         D = urls(p) ∪ 𝔽 ∪ ℍ
12         (ℍ, 𝔽) = Selective-Offline (D, 2|D|, ..., 2n)
13     end
14     revisit the pages in the opposite order
15 end
```

Algorithm 4: SHARC-selective Online

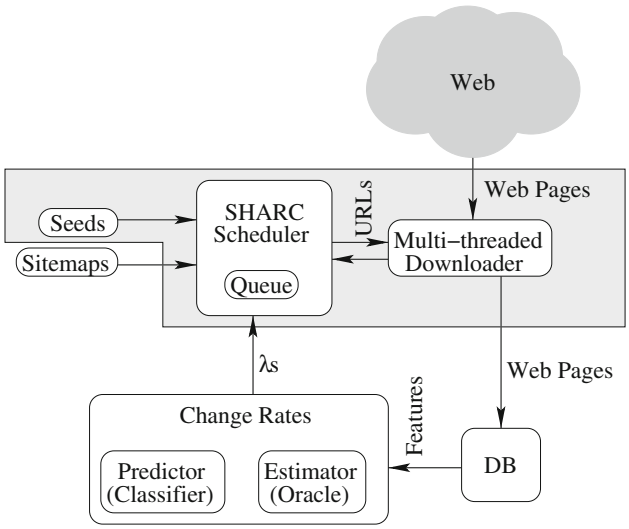


Fig. 11 SHARC architecture

search engines, on the other hand, employ techniques that optimize for freshness, importance of pages, and scope (news, blogs, Deep Web). Given the schedule, the *downloader* aims to fetch as many pages as possible, within the limits of the available network bandwidth. The downloader runs multiple threads in parallel, one for each crawled site, and parses the downloaded pages to discover new URLs. To avoid downloading, the same page more than once, the downloader normalizes the URLs and employs de-duplication techniques. Newly found URLs are returned back to the scheduler for planning, while fully downloaded Web pages are stored and indexed in a *database* (PostgreSQL in our case).

We have integrated selected SHARC strategies into the Heritrix software (within the limitations given by the original Heritrix architecture, hence not all strategies). Figure 11 depicts the integrated architecture. We reused most of the modules of Heritrix (the shaded region), and we added the *Change-Rates module* and replaced the Heritrix scheduler with the SHARC scheduler. Like standard Web crawlers, the SHARC crawler usually starts with a set of seed URLs for a site to be captured. In addition, we have implemented the *sitemaps* protocol, which allows us to load URLs of a site and their typical change rates from published sitemaps. The SHARC scheduler retrieves this information and schedules the pages for visits and then for revisits.

8.2 Determining change rates

Change rates can be determined from three sources: (1) extracted from *sitemaps*, (2) *estimated from previous crawls* of a site, (3) *predicted by machine learning methods* (classifiers or regression models) based on easily observable features. We discuss all these issues below in turn.

```
<sitemapindex xmlns=
"http://www.sitemaps.org/schemas/sitemap/0.9">
<sitemap>
  <loc>http://www.cnn.com/sitemap_specials.xml</loc>
  <lastmod>2007-04-18T12:05:20-04:00</lastmod>
</sitemap>
<sitemap>
  <loc>http://www.cnn.com/sitemap_elections.xml</loc>
  <lastmod>2008-01-08T20:17:50-05:00</lastmod>
</sitemap>
```

Fig. 12 Example of sitemap with sitemap indexes

Sitemaps are an easy way for webmasters to inform robots about pages on their sites that are available at the Web site for crawling. Sitemaps are XML files that contain URLs pointing to other sitemaps (see Fig. 12) or a list of URLs available at the site (see Fig. 13). The compressed size of the sitemap is limited to 10MB and can contain up to 50K URLs. These limitations are introduced so that the Web server does not need to serve very large files. If a sitemap exceeds the limit, then multiple sitemap files and a sitemap index file must be created. However, it has become practice that webmasters create several sitemaps even for small Web sites, grouping the URLs into conceptual partitions of interrelated URLs on a site, *sub-sites* so to speak. Our framework can harness information about subsites that site owners want to be crawled and archived as coherently as possible.

A sitemap file consists of a list of URLs with the following metadata (cf. Fig. 13):

- *loc* is a mandatory field indicating the URL of a Web page.
- *lastmod* is an optional field indicating the last modified date and time of the page.

```
<urlset xmlns=
"http://www.sitemaps.org/schemas/sitemap/0.9">
<url>
  <loc>http://www.dw-world.de</loc>
  <lastmod>2009-02-11</lastmod>
  <changefreq>hourly</changefreq>
  <priority>1.0</priority>
</url>
<url>
  <loc>http://www.dw-world.de/dw/0,,265,00.html</loc>
  <lastmod>2008-11-11</lastmod>
  <changefreq>hourly</changefreq>
  <priority>1.0</priority>
</url>
</urlset>
```

Fig. 13 Example of sitemap with URLs

- *changefreq* is an optional field indicating the typical frequency of change. Valid values include *always, hourly, daily, weekly, monthly, never*. This information can be mapped onto (bins of) change rates for the page-specific parameter of the Poisson-process model.
- *priority* is an optional field indicating the relative importance or weight of the page on the Web site.

Currently, approximately 35 million Web sites publish sitemaps, providing metadata for several billion URLs. Top domains using sitemaps are in .com, .net, .cn, .org, .jp, .de, .cz, .ru, .uk, .nl domains including www.cnn.com, www.nytimes.com, www.bbc.co.uk, www.dw-world.de [35].

The oracle of change rates returns the best estimate for the change rate of a given page. We call it an *oracle*, since it needs to know the full history of changes. This is in contrast to a change rate *predictor* (see below) where the change history is known only for a *sample* of pages and is used to learn a prediction model.

We use the standard maximum likelihood estimator (MLE) for a Poisson distribution to compute the change rate λ. We postulate that the history of timepoints of changes for a time period T is available and the changes of a page follow a Poisson process. We split the period T into n intervals of length τ, such that there is either 1 or 0 changes in each interval. For each page p, we define sample data $x_1, x_2, \ldots, x_n$, where x_i is the number of observed changes of p in the i-th interval. Then, $n = T/\tau$, and the MLE for change rate λ is

$$\hat{\lambda} = \frac{1}{n} \sum_{i=1}^{n} x_i .$$

The predictor of change rates uses standard classification techniques (Naive Bayes and the C4.5 decision-tree classifier) to predict change rates from given features of a page. (We also tried linear regression. It was poorer because of the dominating non-changing pages.) Since classifiers work with categorical output data (labeled classes), we discretize the change rates using equal-frequency binning [22] with ten bins. Equal-frequency binning aims to partition the domain of change rates into bins (intervals) so that each bin contains the same (or nearly the same) number of observations (individual pages) from the dataset. As for the features of the pages, we have investigated two different sets: features that are only available in online settings (the Web page itself is not available, but only its URL and its metadata) and offline settings (where the Web page is available as well):

- *online features:* features from the URL string: domain name, MIME type, depth of the URL path (number of slashes), length of the URL, the first three word-segments of the URL path, the presence/absence of special symbols: tilde (~), underline(_), question mark(?), semicolumn(;), column(:), comma(,), and percentage sign (%) .
- *offline features:* all online features and the number of days since the last change, number of images, number of tables, number of outlinks, number of inlinks in the Web page.

We tested the classifiers on the available datasets (see Sect. 9.3) using 10-fold cross-validation. The results are shown in Table 4. To emphasize the results of the best techniques, we have set these numbers in bold.

The change rate predictors are indeed practically viable. Not surprisingly, the overall winners use offline features, but the online features predictors are also fairly accurate.

C4.5 is slightly more accurate than the Naive Bayesian classifier (see Table 4); therefore, we use C4.5 in our main experiments presented in Sect. 9. For most datasets, the classifier achieved about 70% accuracy. However, even when the classifier is only 40% accurate, the mispredicted change rates are typically close to the actual values and still useful for the scheduling algorithms. This is because we are using ten bins and when we do not classify the change rate into the correct bin (for example the bin with hourly change rates), it is often assigned to an adjacent bin (e.g., the bin of daily changing rates). Figure 14 assesses such mispredictions for the DH

Table 4 Classification precision for MPII, DMOZ, and UKGOV Datasets

Dataset	Online features		Offline features	
	Bayesian	C4.5	Bayesian	C4.5
MPII	90.857	**97.951**	97.502	**97.951**
DMOZ	27.398	**43.520**	48.670	**43.753**
MOD	79.379	**85.995**	80.794	**86.886**
DFID	**70.653**	67.437	**74.020**	71.256
ARMY	78.423	**80.660**	79.312	**82.406**
RAF	85.848	**86.561**	89.453	**91.902**
DH	37.011	**41.658**	41.798	**46.002**

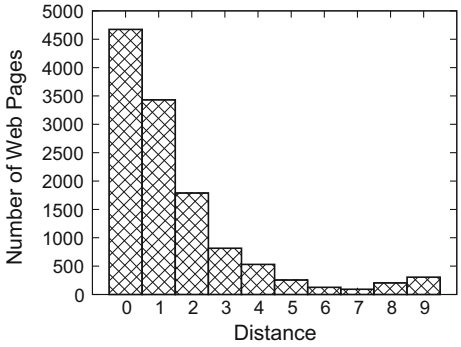


Fig. 14 Assessment of C4.5 classification for the DH dataset

Table 5 Classification precision for sitemap datasets

Dataset	Naive Bayesian	C4.5
dw-world.de	**100.00**	99.981
intereconomia.com	98.464	**99.889**
fr.euronews.net	**99.977**	**99.977**

dataset (only 41.658% precision with online features). Over 4,500 of the pages are predicted correctly (see the first column in the figure), and additional 3,500 ones are assigned to adjacent bins. The number of pages that are mispredicted by two bins "away" is about 2,000 (third column).

We also tested the classifiers for sites with sitemap data. The setup of the experiment is similar to the previous setting; however, we did not need to discretize the change rates, because change frequency is already a categorical attribute in sitemaps (see sitemap example at the beginning of the section). As input features of pages, we have used the domain name, MIME type, depth within the URL path, and number of days since the last update. The results of 10-fold cross-validation are shown in Table 5. To emphasize the results of the best techniques, we have set these numbers in bold. Change rate predictors for sitemaps are extremely precise.

9 Experimental evaluation

We evaluated all SHARC methods in terms of the blur and coherence quality metrics. We first present the competitors under comparison in Subsect. 9.1. The experiments were designed so that we could use the exact history of changes as a reference for computing the actual blur and actual coherence of captures. These metrics are defined in Subsect. 9.2. Datasets on which the SHARC framework is evaluated are described in Subsect. 9.3. Our main experimental findings on the blur and coherence measures are presented in Sects. 9.4 and 9.5. Finally, we present sensitivity studies in Sect. 9.6 where we vary the Web site properties like the size, change skew, and crawl duration. Throughout the section, to emphasize the results of the best strategy, we have set these numbers in bold.

9.1 Methods under comparison

We experimentally evaluate our own techniques—SHARC-offline, SHARC-online, SHARC-revisits, and SHARC-selective—against a variety of baseline strategies: breadth-first search (BFS) and depth-first search (DFS) (most typical techniques by archive crawlers), hottest-first (HF), hottest-last (HL) (most promising simple crawlers, where heat refers to change rates of pages), and the method of Olston and Pandey (OP) [30] (the best freshness-optimized crawling strategy). SHARC-offline requires the knowledge of all URLs of the site in advance. The setup of our experiments allowed us to study this idealized strategy as a reference. In practice, it is unrealistic to assume such knowledge, and SHARC-online will be used instead. The BFS and DFS strategies schedule downloads based on the graph structure of the site as it is dynamically traversed by the crawler (the crawl tree). BFS stores the detected URLs in a FIFO queue, while DFS stores the detected URLs in a stack. HF and HL download the hottest and the coldest page from the list of detected pages. All online strategies (SHARC-online, SHARC-selective online, BFS, DFS, HF, HL) are dynamic and work incrementally: in each iteration, we schedule only one page (from among the so far detected pages). The OP strategy sorts the pages in each iteration according to the values of the utility function $U_{p_i}(i)$ and downloads the one with the highest value. The utility function $U_{p_i}(i)$ assumes knowledge of the full change history. This assumption is not practical; we give these optimistic performance numbers for comparison.

For incorporating change rate estimates, we include two versions: the change rate is given either by the oracle or by the predictor. We trained the predictors only with the online features. A random sample of 10% of the size of each Web site was used to train the classifiers.

The incoherence measure (in contrast to blur) assumes the visit–revisit crawl model where all pages are accessed twice. The classical BFS, DFS, HF, and HL strategies do not have revisits, but for fair comparison, we simulated the revisits using FIFO (first-in-first-out) or LIFO (last-in-first-out) strategies. We did not run OP for the visit–revisit case, since there is not obvious generalization of this technique.

9.2 Quality metrics

We use the actual blur (Eq. (23)) and actual incoherence to assess single-visit and visit–revisit strategies, respectively.

The intuition behind the actual blur is the same as the one behind the stochastic blur (Definitions 1 and 3): the average number of changes that an explorer of the archive will encounter. The stochastic blur uses change rates λ_i while the actual blur uses the history of actual timepoints of changes

$(h_0, \ldots, h_m)$. In combination with the download time t of page p and the observation interval $[o_s, o_e]$, this allows us to define the actual blur of p:

$$B(p) = \frac{1}{o_e - o_s} \left(\sum_{o_s \le h_j \le t} (h_j - o_s) + \sum_{t < h_j \le o_e} (o_e - h_j) \right). \tag{23}$$

The actual blur B of an entire site capture with pages $(p_0, \ldots, p_n)$ is the sum of the actual blur values of all pages:

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

The actual blur with revisits can be defined similarly. Let t^v be the visit and t^r be the revisit time of page p. Then, the *actual blur with revisits* for observation interval $[o_s, o_e]$ is as follows:

$$B(p) = \frac{1}{o_e - o_s} \left(\sum_{o_s < h_j \le t^v} (h_j - o_s) + \sum_{t^v < h_j \le \frac{t^v + t^r}{2}} \left(\frac{t^v + t^r}{2} - h_j \right) + \sum_{\frac{t^v + t^r}{2} < h_j \le t^r} (h_j - \frac{t^v + t^r}{2} + \sum_{t^r < h_j \le o_e} (o_e - h_j) \right). \tag{24}$$

The intuition behind this formula is that for each page that should be accessed as of observation time t, we can choose either the version as of the visit time or the version as of the revisit time, whichever is closer to t.

The actual blur with revisits of a site capture with pages $(p_0, \ldots, p_n)$ is the sum of the corresponding values of all pages:

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

The OP utility function U_p can be expressed in terms of the actual blur. U_p is the actual 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.

In Sect. 7, we defined the coherence measure as the number of pages that did not change during their visit–revisit intervals. In the experiments, we measure and report the *actual incoherence*: the number of pages that changed during their visit–revisit intervals. So just like with blur, the lower the reported numbers are the better it is for Web archive quality.

9.3 Datasets

We tested all methods on a variety of real-world datasets and also on synthetically generated Web sites for systematic variation of site properties. The real-world datasets are summarized in Table 6. They are used for our main experiments about blur and incoherence, presented in Sects. 9.4 and 9.5. The datasets span long periods (a year or longer) and consist of captures (weekly or even daily) of entire sites. The datasets consist of an academic Web site (Max Planck Institute for Informatics MPII), a number of governmental Web sites (UK government), and a Web directory (DMOZ). The MPII dataset constitutes crawls of our Web server. The "home" Web server allowed us to crawl it frequently and aggressively (without respecting the politeness delays). The DMOZ dataset represents a large Web site with subsites (topic categories) that change frequently and subsites that change infrequently. The UKGOV dataset spans a long period and is, to the best of our knowledge, the most comprehensive, freely available Web archive reference collection.

We used the available datasets to simulate crawls and evaluate the SHARC framework. As a stress test for politeness-aware archive crawling, we artificially slowed down the crawls so that they would take as long as the entire time period covered by the dataset. This is done by replaying crawls from our stored data with virtual time. Obviously, slowing down a crawl so that it would take months or even a whole year is quite extreme and does not correspond to what you would do in reality. But in this way, we impose a large number of page changes on each crawl and turn our experiments into more informative stress-test studies.

The synthetic datasets simulate changes according to the Poisson process and are used in sensitivity experiments (see Sect. 9.6). The change rates are modeled with a skewed distribution: $\lambda_i = 1/((i+1)^{\text{skew}})$. The Web graph of a synthetic site is generated to form a tree with n pages;

Table 6 Datasets

Dataset	Abbreviation	Web site	Periodicity	Time range	Pages	Changing pages
MPII	MPII	mpi-inf.mpg.de	daily	08.09–07.10	72,071	1,356
DMOZ	DMOZ	dmoz.org	weekly	10.09–07.10	177,446	50,855
UKGOV	MOD	mod.uk	weekly	08.03–02.06	10,047	5,988
UKGOV	DFID	dfid.gov.uk	weekly	08.03–02.06	2,186	1,131
UKGOV	ARMY	army.mod.uk	weekly	08.03–02.06	37,330	15,259
UKGOV	RAF	raf.mod.uk	weekly	08.03–02.06	27,836	4,286
UKGOV	DH	dh.gov.uk	weekly	08.03–02.06	15,884	12,203

each page p_i has outdegree children. The default values are skew $= 1.2$, outdegree $= 400$, and $n = 10,000$. To correlate site structure with skewed change rates, we generated Web sites in two flavors. The first flavor has the hottest page at the root of the site tree, and the pages are gradually "cooling down" toward the leaves (denoted by "Cold Leaves" in Tables 11, 12, 13, 14 , 15, 16, 17, 18, 19, 20, 21). The second flavor consists of sites where the root is coldest, and the pages are gradually "heating up" toward the leaves (denoted by "Hot Leaves" in the captions).

9.4 Blur experiments with single-visit strategies

The results of the experiments on blur with single-visit strategies over real-world datasets are shown in Table 7. The best values for each dataset are highlighted in boldface. SHARC-offline outperforms all competitors by a large margin. SHARC-online with change-rate oracle performs nearly as well as the optimal SHARC-offline method. Its additional burden, compared to SHARC-offline, is that it needs to incrementally detect the pages of a Web site. The experiments show that the penalty of this page discovery process is low.

SHARC-online with predicted change rates leads a more pronounced increase in the blur metric. Obviously, estimation errors about which pages are hot and which ones are not so hot have a notable influence on the overall quality of a site capture. As a consequence, SHARC-online may even be slightly inferior to more traditional baselines on some data, but it wins by a large margin for most datasets. Note that the traditional baselines used a change-rate oracle; so the slight losses of SHARC-online are mostly due to the facts that the opponents had better a priori knowledge about changes.

Overall, this experiment showed that our blur-optimizing strategies do indeed provide what our theory suggested: they clearly improve the quality of site captures for web archiving.

9.5 Incoherence experiments with visit–revisit strategies

The results of the experiments on incoherence with visit–revisit strategies over real-world datasets are shown in Table 8 with change rate oracle and Table 9 with change rate predictors, respectively.

For the tests with change rate oracle, the SHARC-selective strategies outperformed all baseline opponents by a substantial margin. SHARC-selective offline exhibits incoherence values that are lower than those of the competitors by more than a factor of 3. SHARC-online did not perform quite as well as its offline counterpart, but it is not much worse and still much better than all online competitors. On one specific dataset, SHARC-selective online outperformed the offline variant, but it is due to "random" effects regarding lucky situations by the order in which pages are detected. SHARC-revisits are designed to minimize the blur metric (not shown here) and performed moderately on incoherence. The flexibility that the SHARC-selective strategies have in

Table 7 Average blur per page

Site	SHARC			HF	HL	BFS	DFS	OP
	Offline Oracle	Online Oracle	Online Predicted	Oracle	Oracle			Oracle
MPII	**0.12**	0.13	0.16	0.26	0.15	0.24	0.15	0.18
DMOZ	**0.18**	0.19	0.23	0.24	0.31	0.25	0.25	0.24
MOD	**2.14**	2.16	2.17	2.48	2.98	2.41	2.46	3.15
DFID	**2.11**	2.17	2.17	3.35	2.20	2.51	2.19	2.17
ARMY	**1.23**	1.26	1.29	1.55	1.65	1.56	1.45	1.81
RAF	**0.11**	**0.11**	0.14	0.14	0.15	0.14	0.15	0.15
DH	**2.82**	2.83	2.94	4.00	3.08	3.08	3.08	3.27

Table 8 Incoherence with oracle of change rates

Site	SHARC-selective		SHARC	HF	HF	HL	HL	BFS	BFS	DFS	DFS
	Offline	Online	Revisits	FIFO	LIFO	FIFO	LIFO	FIFO	LIFO	FIFO	LIFO
MPII	**145**	453	837	1,013	1,252	751	508	935	1,091	908	726
DMOZ	**15,914**	25,206	34,233	37,405	42,240	33,413	31,720	35,672	36,176	34,357	33,048
MOD	**4,449**	4,701	5,803	5,903	5,769	5,766	5,452	5,840	5,475	5,767	4,890
DFID	**959**	987	1,113	1,110	1,115	1,109	1,104	1,097	1,038	1,079	1,049
ARMY	**10,947**	12,479	14,052	13,865	14,528	14,008	13,901	14,104	13,996	14,172	13,120
RAF	**2,120**	3,284	4,096	3,487	4,243	4,077	3,676	3,242	3,192	4,006	3,941
DH	11,376	**11,368**	11,589	11,514	11,494	11,546	11,457	11,625	11,483	11,625	11,482

Table 9 Incoherence with predictor

Site	SHARC-selective		SHARC	HF	HF	HL	HL
	Offline	Online	Revisits	FIFO	LIFO	FIFO	LIFO
MPII	558	**504**	720	835	931	808	**504**
DMOZ	**26,016**	31,847	34,098	36,553	38,396	34,447	31,847
MOD	**4,516**	5,089	5,776	5,867	5,662	5,748	5,262
DFID	**961**	985	1,076	1,102	1,076	1,088	1,054
ARMY	**11,774**	12,876	14,305	14,290	14,245	14,113	13,796
RAF	**3,520**	3,745	3,891	4,030	3,615	3,914	3,745
DH	11,360	**11,305**	11,545	11,596	11,517	11,681	11,612

dealing with hopeless pages pays off well and leads to the highest gains on sites with a large number of changing pages like DMOZ, MOD, DH, and ARMY.

When all methods are limited to the realistic case of relying on a change rate predictor, the SHARC-selective strategies again win by a substantial margin, as shown in Table 9. We did no longer include the BFS and DFS strategies in this comparison, as they were already clear losers in the experiments with change rate oracle. For some datasets, SHARC-selective online even performed better than the offline variant. The explanation lies in the nature of the corresponding Web sites. For example, the MPII dataset is very skewed: there are about 1,000 pages, out of 70,000, with very high change rates, while the other 69,000 hardly ever changed. This reduced the quality of the change rate predictor and led to suboptimal behavior. SHARC-selective online, our natural candidate for deployment in a real system, performed very well across the suite of datasets: usually not much worse than the offline variant, and sometimes even better.

Compared to the previous experiment with change rate oracle (see Table 8), the incoherence values of the predictor-based methods increased considerably. Although our predictors generally provided decent accuracy, there is room for improvement in this regard.

We also performed live measurements (as opposed to experiments replayed from stored data about former crawls) on a number of sites, using our extended version of the Heritrix crawler. Here, we focused on sites which publish sitemaps, and we limit ourselves to the three most interesting competitors because each strategy requires a separate crawl in real time. Running more simultaneous crawls on the same Web site would influence the crawls themselves, and running different strategies sequentially would make their results incomparable as they no longer see the same state of the site.

We crawled selected subsites of dw-world.de (5,309 p in total), intereconomia.com (2,948 p), and fr.euronews.net (672 p) defined by available sitemaps. The results are shown in Table 10. SHARC-selective clearly wins for all sites.

Table 10 Incoherence with sitemaps and Heritrix crawler

Site	SHARC Selective	HL LIFO	DFS LIFO
dw-world.de	**3,655**	3,705	3,712
intereconomia.com	**2,704**	2,723	2,721
fr.euronews.net	**643**	657	658

Table 11 Scalability: average blur per page (cold leaves)

Size	SHARC		HF	HL	BFS	DFS	OP
	Offline	Online					
$1 \cdot 10^4$	**1.21**	1.70	1.75	2.26	2.00	1.80	2.28
$5 \cdot 10^4$	**1.25**	1.75	1.80	2.38	2.07	1.83	2.42
$1 \cdot 10^5$	**1.27**	1.77	1.82	2.44	2.13	1.87	2.48
$5 \cdot 10^5$	**1.30**	1.81	1.86	2.52	2.18	2.00	2.56
$1 \cdot 10^6$	**1.31**	1.83	1.87	2.55	2.19	2.01	2.59

Table 12 Scalability: average blur per page (hot leaves)

Size	SHARC		HF	HL	BFS	DFS	OP
	Offline	Online					
$1 \cdot 10^4$	**1.19**	1.21	2.24	1.44	1.28	1.22	2.25
$5 \cdot 10^4$	**1.25**	1.26	2.39	1.53	1.92	1.81	2.42
$1 \cdot 10^5$	**1.26**	1.27	2.43	1.55	1.36	1.30	2.47
$6 \cdot 10^5$	**1.29**	1.30	2.52	1.60	1.39	1.96	2.56
$1 \cdot 10^6$	**1.31**	**1.31**	2.55	1.61	2.21	1.94	2.59

9.6 Sensitivity studies

In this section, we studied the sensitivity of our crawl strategies with regard to the scale (size) of a Web site, the skew in the change rate distribution of a site's pages, and the politeness-driven duration of the crawl. As we wanted to vary these parameters systematically, we performed these experiments with synthetically generated Web sites.

Scalability. Tables 11 and 12 show blur results for Web sites with hotter pages closer to the root and hotter pages closer

Table 13 Scalability: incoherence (cold leaves)

Size	SHARC-selective		SHARC	HF	HF	HL	HL	BFS	BFS	DFS	DFS
	Offline	Online	Revisits	FIFO	LIFO	FIFO	LIFO	FIFO	LIFO	FIFO	LIFO
$1 \cdot 10^4$	**3,075**	**3,075**	4,104	4,184	4,140	4,177	3,079	4,170	3,719	4,163	3,995
$5 \cdot 10^4$	**11,940**	**11,940**	17,327	17,394	17,959	17,397	12,100	17,368	16,072	17,418	16,366
$1 \cdot 10^5$	**21,378**	**21,378**	32,051	32,142	33,705	32,110	21,533	32,015	29,033	32,129	29,395

Table 14 Scalability: incoherence (hot leaves)

Size	SHARC-selective		SHARC	HF	HF	HL	HL	BFS	BFS	DFS	DFS
	Offline	Online	Revisits	FIFO	LIFO	FIFO	LIFO	FIFO	LIFO	FIFO	LIFO
$1 \cdot 10^4$	**3,019**	3,039	4,143	4,140	4,057	4,165	3,238	4,136	3,785	4,175	3,999
$5 \cdot 10^4$	**11,830**	11,997	17,377	17,321	17,925	17,330	12,364	17,288	16,582	17,366	15,397
$1 \cdot 10^5$	**21,417**	21,630	32,295	32,050	33,776	32,313	22,165	32,028	30,618	32,181	29,664

Table 15 Crawl duration: average blur per page (cold leaves)

Slow-down	SHARC		HF	HL	BFS	DFS	OP
	Offline	Online					
1	**1.20**	1.68	1.97	2.25	2.21	1.78	2.28
2	**2.41**	3.38	3.96	4.50	4.36	3.46	4.41
3	**3.58**	5.02	5.89	6.70	6.48	5.07	6.54
4	**4.79**	6.72	7.88	8.98	8.81	7.28	8.77
5	**5.99**	8.38	9.83	11.21	10.77	8.58	10.95
10	**12.00**	16.83	19.75	22.48	21.62	16.75	22.00

to the leaves, respectively. While SHARC-online is only slightly worse than SHARC-offline in Table 12, the difference between the strategies is more prominent in Table 11. This difference appears because we discover cold pages much later in Table 12 and misguide the schedule for page downloads by seeing mostly hot pages early on.

SHARC-online consistently outperforms all baseline opponents. Schedules of HF for the dataset with cold leaves and HL for the hot leaves are very similar to the schedule of SHARC-online. Consequently, HF and HL perform almost as well as SHARC-online in some cases, but strongly deteriorate in the other cases. Moreover, they are much more sensitive to the size of a Web site. The sensitivity comes from the strong influence of the different sizes on the number of leaves and consequently on the placement of the hot and cold pages in the schedule.

Tables 13 and 14 show scalability results for the incoherence measure with visit–revisit strategies. SHARC-selective strategies outperform all competitors, with increasing gains as the size of the site increased. SHARC-selective online produces schedules identical to those of the offline variant for hot pages near the root (Table 13), as it detects these pages early. For the dual case of hot pages near the leaves (Table 14), SHARC-selective online lost against its offline counterpart, but still outperformed all other opponents by a large margin.

Table 16 Crawl duration: average blur per page (hot leaves)

Slow-down	SHARC		HF	HL	BFS	DFS	OP
	Offline	Online					
1	**1.20**	1.22	2.27	2.10	1.86	1.85	2.27
2	**2.39**	2.42	4.48	4.17	2.47	2.46	4.40
3	**3.60**	3.65	6.73	6.25	3.92	4.02	6.56
4	**4.80**	4.86	8.99	8.37	7.55	7.47	8.77
5	**5.98**	6.07	11.19	10.42	8.29	8.24	10.94
10	**12.01**	12.18	22.47	20.92	17.67	18.79	22.03

Crawl Duration. In this experiment, we increased the politeness delay by a specified slow-down factor. Tables 15 and 16 show the resulting blur values. For all strategies, the capture interval increases, and in turn, the blur increases as well. For short captures, all competitors perform similarly, as crawling is almost "instantaneous". For longer captures, our SHARC strategies become increasingly advantageous over the competitors.

SHARC-online outperforms all other online strategies. The only exception is the DFS strategy for the longest crawl and the dataset with cold leaves. The specific placement of the changes in the tree of the Web site made the schedule of the DFS slightly closer to the schedule of SHARC-offline. However, as the placement of the hot pages changes (Table 16),

Table 17 Crawl duration: incoherence (cold leaves)

Slow-down	SHARC-selective		SHARC	HF	HF	HL	HL	BFS	BFS	DFS	DFS
	Offline	Online	Revisits	FIFO	LIFO	FIFO	LIFO	FIFO	LIFO	FIFO	LIFO
1	**3,058**	**3,058**	4,215	4,172	4,118	4,212	3,062	4,176	3,931	4,222	3,695
2	4,608	**4,607**	58,65	5,875	5,359	5,857	4,919	5,811	5,317	5,887	5,542
3	**5,672**	5,673	6,945	7,009	6,193	6,984	6,301	6,984	6,215	6,975	6,336
4	**6,255**	6,286	7,695	7,678	6,646	7,711	7,287	7,701	6,814	7,703	6,842
5	**6,721**	6,753	8,207	8,214	7,009	8,224	7,985	8,224	7,221	8,195	7,414
10	7,906	**7,903**	9,443	9,419	8,003	9,432	9,488	9,415	8,561	9,425	8,619

Table 18 Crawl duration: incoherence (hot leaves)

Slow-down	SHARC-selective		SHARC	HF	HF	HL	HL	BFS	BFS	DFS	DFS
	Offline	Online	Revisits	FIFO	LIFO	FIFO	LIFO	FIFO	LIFO	FIFO	LIFO
1	**3,002**	3,006	4,102	4,115	4,083	4,150	3,216	4,091	3,735	4,170	4,006
2	**4,547**	4,859	5,778	5,767	5,281	5,741	5,288	5,733	5,598	5,818	5,305
3	**5,552**	6,387	6,835	6,894	6,066	6,833	6,774	6,896	6,531	6,887	6,260
4	**6,221**	6,466	7,697	7,689	6,696	7,688	7,816	7,690	6,856	7,681	6,953
5	**6,744**	6,847	8,208	8,215	7,092	8,210	8,541	8,222	7,379	8,195	7,634
10	**7,959**	8,075	9,431	9,436	8,118	9,414	9,760	9,403	8,708	9,446	8,309

Table 19 Skew: average blur per page

Skew	SHARC		HF	HL	BFS	DFS	OP
	Offline	Online					
(a) Cold leaves							
0.50	**60.49**	67.00	73.53	73.84	71.79	67.63	69.23
1.00	**2.57**	3.35	3.99	4.40	4.29	3.54	4.20
1.50	**0.61**	0.88	1.08	1.21	1.18	1.00	1.22
2.00	**0.37**	0.58	0.70	0.74	0.74	0.63	0.74
(b) Hot leaves							
0.50	**60.38**	62.38	73.71	72.02	69.01	69.56	69.23
1.00	**2.55**	2.61	4.39	4.17	3.10	3.15	4.16
1.50	**0.62**	**0.62**	1.22	1.16	1.06	1.11	1.22
2.00	**0.38**	**0.38**	0.75	0.71	**0.38**	**0.38**	0.75

the cold pages are discovered earlier and SHARC-online substantially outperforms DFS.

For visit–revisit strategies aiming at low incoherence, the results are shown in Tables 17 and 18. The SHARC-selective methods outperform all competitors. The online method is very close to the offline variant, and sometimes even better (see Table 17). This is due to "random" effects: early discovery of hopeless (very hot) pages resulted in almost identical schedules for online and offline methods. But for the dual case with hot pages closer to leaves (Table 18), the offline strategy consistently outperforms the online variant as the latter discovers hopeless pages much later and thus places them in suboptimal slots.

Skew. Skew controls how uniformly the changes are distributed among the pages. High skew allocates most of the changes to very few pages, while low skew keeps the changes uniformly distributed. In absolute numbers, this results in high blur for low skew and low blur for high skew for all strategies. Table 19 shows the results for this sensitivity study with the single-visit strategies aiming at low blur. Relative to the baseline opponents, the SHARC strategies cope best with high skew. Their relative gains increase with increasing skew.

Tables 20 and 21 show the result of the study with visit–revisit strategies aiming at low incoherence. SHARC-selective offline is the best strategy, and SHARC-selective online is second best. For large skew values (skew $>$ 1) and hot pages closer to leaves, the online method does not differ from the offline variant, since very few pages are very hot and their late discovery by the online crawler does not influence the schedule anymore.

10 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 coherence. The blur measure is appropriate for explorative use of archives. The coherence measure is appropriate for legally tangible use of archives. For each of the measures, we presented strategies applicable in practice. SHARC-online minimizes the blur, and SHARC-selective maximizes coherence. The experiments confirm that SHARC-online and SHARC-selective outperform their competitors.

Directions for future work include the following. First, experiments with sitemaps revealed that some sites change

Table 20 Skew: incoherence (cold leaves)

Skew	SHARC-selective		SHARC	HF	HF	HL	HL	BFS	BFS	DFS	DFS
	Offline	Online	Revisits	FIFO	LIFO	FIFO	LIFO	FIFO	LIFO	FIFO	LIFO
1.00	**7,061**	7,073	8,518	8,517	7,257	8,504	8,110	8,540	7,623	8,509	7,567
1.50	**365**	367	1,064	1,037	1,308	1,060	710	1,056	1,119	1,035	935
2.00	**11**	35	169	174	246	176	210	178	223	170	175

Table 21 Skew: incoherence (hot leaves)

Skew	SHARC-selective		SHARC	HF	HF	HL	HL	BFS	BFS	DFS	DFS
	Offline	Online	Revisits	FIFO	LIFO	FIFO	LIFO	FIFO	LIFO	FIFO	LIFO
1.00	**7,012**	7,151	8,455	8,479	7,283	8,475	8,587	8,516	7,790	8,486	7,316
1.50	**355**	**355**	1,018	1,010	1,237	1,033	362	1,035	714	1,022	1,157
2.00	**20**	**20**	182	174	225	171	22	177	186	181	169

very frequently, and it is not possible to capture them entirely coherently. Instead, the large subsites for coherent captures should be identified. Second, it would be interesting to incorporate the importance (priority weight) of pages into the framework. The importance weights may depend on the freshness requirements on the pages, access frequency as estimated, for example, from logs, and so on. Third, it would be interesting to develop strategies for continuous and incremental crawling. In such settings, we would no longer distinguish individual captures but would continuously visit pages driven by archiving priorities.

Acknowledgments This work is supported by the 7th Framework IST programme of the European Union through the Living Web Archives (LiWA) project. Data provisioning by the European Archive (europearchive.org) is gratefully acknowledged.

References

1. Adar, E., Teevan, J., Dumais, S.T., Elsas, J.L.: The Web changes everything: understanding the dynamics of Web content. In: WSDM'09, pp. 282–291 (2009)
2. Alam, Md.H., Ha, J., Lee, S.: Fractional pagerank crawler: Prioritizing URLs efficiently for crawling important pages early. In: DASFAA'09, pp. 590–594 (2009)
3. Segev, A., Shoshani, A.: Logical modeling of temporal data. SIGMOD Rec. **16**(3), 454–466 (1987)
4. Baeza-Yates, R., Gionis, A., Junqueira, F., Murdock, V., Plachoura, V., Silvestri, F.: Design trade-offs for search engine caching. ACM Trans. Web **2**(4), 1–28 (2008)
5. Batsakis, S., Petrakis, E.G.M., Milios, E.E.: Improving the performance of focused Web crawlers. Data Knowl. Eng. **68**(10), 1001–1013 (2009)
6. Baykan, E., Henzinger, M., Marian, L., Weber, I.: Purely URL-based topic classification. In: WWW'09, pp. 1109–1110 (2009)
7. Brewington, B.E., Cybenko, G.: Keeping up with the changing Web. Computer **33**(5), 52–58 (2000)
8. Castillo, C., Marin, M., Rodriguez, A., Baeza-Yates, R.: Scheduling algorithms for Web crawling. In: LA-WEBMEDIA'04, pp. 10–17 (2004)
9. Chen, L., Bhowmick, S.S., Nejdl, W.: Near-miner: mining evolution associations of Web site directories for efficient maintenance of Web archives. PVLDB **2**(1), 1150–1161 (2009)
10. Cho, J., Garcia-Molina, H.: Synchronizing a database to improve freshness. SIGMOD Rec. **29**(2), 117–128 (2000)
11. Cho, J., Garcia-Molina, H.: Estimating frequency of change. ACM Trans. Inter. Tech. **3**(3), 256–290 (2003)
12. Cho J., Garcia-Molina H., Page L. (2007) Efficient crawling through URL ordering. In: WWW'07, pp. 161–172. (2007)
13. Cho J., Ntoulas A. (2002) Effective change detection using sampling. In: VLDB'02, pp. 514–525. (2002)
14. Cho J., Schonfeld U. (2007) Rankmass crawler: a crawler with high personalized pagerank coverage guarantee. In: VLDB'07, pp. 375–386. (2007)
15. Cho, J., Garcia-Molina, H.: Estimating frequency of change. ACM Trans. Internet Technol. **3**(3), 256–290 (2003)
16. Colby, L.S., Kawaguchi, A., Lieuwen, D.F., Mumick, I.S., Ross, K.A.: Supporting multiple view maintenance policies. SIGMOD Rec. **26**(2), 405–416 (1997)
17. Dai, N., Davison, B.D.: Freshness matters: in flowers, food, and Web authority. In: SIGIR'10, pp. 114–121 (2010)
18. Dash, D., Kantere, V., Ailamaki, A.: An economic model for self-tuned cloud caching. In: ICDE'09, pp. 1687–1693 (2009)
19. Denev, D., Mazeika, A., Spaniol, M., Weikum, G.: Sharc: framework for quality-conscious Web archiving. PVLDB **2**(1), 586–597 (2009)
20. Masanès, J. (ed.): Web Archiving. Springer, UK (2006)
21. Härder, T., Bühmann, A.: Value complete, column complete, predicate complete. In: VLDBJ **17**(4), pp. 805–826 (2008)
22. Jiawei, M., Han, J.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2005)
23. Kan, M.-Y., Thi, H.O.N.: Fast Webpage classification using URL features. In: CIKM'05, pp. 325–326 (2005)
24. Kim, S., Lee, S.: Estimating the change of Web pages. In: ICCS'07, Vol. 4489 of LNCS, pp. 798–805 (2007)
25. Lee, H.-T., Leonard, D., Wang, X., Loguinov, D.: Irlbot: scaling to 6 billion pages and beyond. In: WWW'08, pp. 427–436 (2008)
26. Levene, M., Poulovassilis, A. (eds.): Web Dynamics—Adapting to Change in Content, Size, Topology and Use. Springer, Berlin (2004)

27. Mohr, G., Kimpton, M., Stack, M., Ranitovic, I.: Introduction to Heritrix, an archival quality Web crawler. In: IWAW'04 (2004)
28. Najork, M., Wiener, J.L.: Breadth-first search crawling yields high-quality pages. In: WWW'01, pp. 114–118 (2001)
29. Ntoulas, A., Cho, J., Olston, C.: What's new on the Web?: the evolution of the Web from a search engine perspective. In: WWW'04, pp. 1–12 (2004)
30. Olston, C., Pandey, S.: Recrawl scheduling based on information longevity. In: WWW'08, pp. 437–446 (2008)
31. Olston, C., Widom, J.: Best-effort cache synchronization with source cooperation. In: In SIGMOD'02, pp. 73–84 (2002)
32. Practice.com. Debunking the wayback machine. http://practice.com/2008/12/29/debunking-the-wayback-machine
33. Qi, X., Davison, B.D.: Web page classification: features and algorithms. ACM Comput. Surv. **41**(2), 1–31 (2009)
34. Schenkel, R.: Temporal shingling for version identification in Web archives. In: ECIR'10, pp. 508–519 (2010)
35. Schonfeld, U., Shivakumar, N.: Sitemaps: above and beyond the crawl of duty. In: WWW'09, pp. 991–1000 (2009)
36. Spaniol, M., Denev, D., Mazeika, A., Weikum, G., Senellart, P.: Data quality in Web archiving. In: WICOW'09, pp. 19–26 (2009)
37. Tolia, N., Satyanarayanan, M.: Consistency-preserving caching of dynamic database content. In: WWW'07, pp. 311–320 (2007)
38. Singh, S.R. (2007) Estimating the rate of Web page updates. In: IJCAI'07, pp. 2874–2879 (2007)
39. Zheng, S., Dmitriev, P., Giles, C.L.: Graph-based seed selection for Web-scale crawlers. In: CIKM'09, pp. 1967–1970 (2009)
40. Zhou, Y., Jiang, M., Zhang, Q., Huang, X., Wu, L.: Selective re-crawling for object-level vertical search. In: WWW'10, pp. 1221–1222 (2010)