The SOLAR System for Sharp Web Archiving

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

ABSTRACT

Web archives preserve the history of digital information on the Internet. They are a great asset for media and business analysts and also for experts on intellectual property (e.g, patent offices, IP lawyers) and Internet legislators (e.g. consumer services) to prove or disprove certain allegations. To fulfill this purpose, archives should not only periodically crawl a Web site's pages but should also assure that the captured pages are a precise representation of the Web site as of a *single timepoint*. This is a hard problem, since the politeness etiquette and completeness requirement of archive crawlers mandate slow, long-duration crawling while the Web site is changing.

This paper presents the SOLAR (Scheduling of Downloads for Archiving of Web Sites) system for sharp Web archiving. SOLAR crawls all pages of a Web site and then re-crawls the visited pages forming visit-revisit intervals. If all visit-revisit intervals overlap and no page changed between its visit and revisit then all pages are "sharp" and captured as if the entire site were downloaded instantaneously. SOLAR judiciously schedules visits and revisits to maximize the number of sharp pages based on the predictions of page-specific change rates. Experiments with synthetic date show SOLAR outperforms existing techniques and captures the sites as sharp as possible.

1. INTRODUCTION

1.1 Problem

National libraries, the Internet Archive (`archive.org`), the European Archive (`europarchive.org`), and other archival institutions collect and preserve the ever changing Web. These archives reflect the zeitgeist of the society for well over a decade and are a gold mine for sociologists, politologists, media and market analysts. Experts on intellectual property (IP, e.g., at patent offices) and compliance with Internet legislation (e.g., for consumer services) started turning their attention to Web archives, too. 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.

To live up to this potential, Web archives need a strong notion of data quality. When capturing a Web site for archival, it would be ideal if all pages could be collected together instantaneously. Then, we could refer to this archived version as the true state of the Web site as of that time instant. Unfortunately, this ideal is infeasible; crawling a large Web site needs to obey the politeness etiquette and thus spans an extended time period of several hours if not even days. During this time, pages are changing so that the archived version may exhibit incoherent states across different pages. For example, one page may show a legally required disclaimer, and a related page does not mention this although it would be expected to do so. We call a site capture *sharp* (or "coherent") if it is equivalent to the result of a fictitious instantaneous crawl. Our goal in this paper is to develop crawling strategies that aim at sharp site captures, or at least maximize the parts of a site that can be sharply archived and collectively timestamped with a tangible point in time. Note that this objective implies that the crawl scheduling is very different from the crawling strategies that are used for search engines. The latter aim to see as many pages or links to pages as possible, within the resource budget of the crawler, and do not care about incoherent views of different pages.

There has been very little prior work along these lines, because we are only now seeing this great potential of Web archives. [20, 9] introduced a model for quality-conscious Web archiving. They considered both one-visit and visit-revisit strategies. In visit-revisit strategies each page is visited twice: first downloaded and later revisited to check that the page did not change in the meantime. A single page is called *sharp* if the page did not change between its visit and revisit. Additional measures are needed to ensure that all (or almost all) pages of the site are collectively sharp as of the same timepoint. [20, 9] addressed from a stochastic viewpoint, by aiming to minimize the "expected blur" with regard to probability distributions of changes and observation points. While this approach strengthens the overall data quality of a Web archive, it does not yield any tangible sharpness guarantees for a given time point (e.g., a concrete date that a lawsuit would want to refer to).

1.2 Contribution

The contribution of this paper is the SOLAR system: Scheduling of dOwnLoads for the ARchiving of Web sites. Similarly to the framework in [20, 9], we visit and revisit each page and we ensure that the *visit-revisit intervals* of all pages intersect. If all pages are sharp (unchanged between visit and revisit), then any timepoint in the intersection of the intervals can serve as a tangible reference for the true state of the site as of that time. If some parts of the site have frequent changes and the crawl is too slow relative to this change rate, we may not be able to capture the entire site sharply. But we can detect this situation, and we can identify large parts of the site that are collectively sharp as of the same timestamp. Therefore, our goal is to schedule the visits and revisits of pages so as to maximize the total number of sharp pages for some joint timepoint, denoted as ##pages in this paper.

We assume that we have a statistical predictor for each page's change rate, adopting the Possion-process model of [1, 6] and based on previous observations and page features like MIME type, depth within the site, URL, manually edited vs. CMS generated, etc. We use these predictors to construct a *confidence interval* for each page: a timespan during which the page is very likely to remain unchanged. Then, the actual scheduling of visits and revisits is treated as an assignment problem for the confidence intervals of the site's pages. As we will see later, this entails difficult combinatorial issues. In particular, we may be unable to accommodate all pages and need to identify "hopeless" pages in order to maximize ##pages within the crawl's resource limitations.

SOLAR makes the following novel contributions:

- We develop a family of scheduling strategies that aim to maximize ## pages, based on visit-revisit intervals and the statistically estimated confidence intervals.
- The SOLAR system is fully implemented in our testbed. We present experimental studies with synthetically generated Web sites. The experiments demonstrate the practical feasibility of our approach and the advantages of SOLAR compared to other state-of-the art sharp crawling strategies.

The paper is organized as follows. Section 2 reviews related work. Section 3 introduces our computational model for site capturing. Section 4 and 5 present our crawl algorithms: first an offline strategy where information about the site and the pages' confidence intervals are known in advance, then an online strategy where pages and confidence intervals are discovered as the crawl is proceeding. Section 6 presents our experimental evaluation.

2. RELATED WORK

The most comprehensive overview on Web archiving is given by Masanès [14]. The author describes the various aspects involved in the archiving as well as the subsequent accessing process. The issue of coherence is introduced as well, but only some heuristics how to measure and improve the archiving crawler's frontier are suggested.

From a technical point of view, the closest related work to crawling strategies for high-quality Web archives are by Spaniol et al. [20] and Denev et al. [9]. Spaniol et al [20] assumes that all pages are known a-priori and allocates visits and revisits of the pages so the visits-revisits form symmetric intervals of length $2i$. In this paper we generalize the approach: the visit-revisit intervals are based on the confidence interval of no change of the pages; scheduling of the pages can be done dynamically as the Web site is discovered during the crawl. The main focus of the SHARC framework [9] is the stochastic one-visit strategies (where the average number of changes of pages are minimized). In this paper we optimize the deterministic ##pages.

Our solution assumes that a confidence interval of no change can be computed for a given page. Such intervals can be computed using the Poisson model, the de facto model of changing Web [1, 6, 5, 13, 11]. In the Poisson model, the time between two successive changes of page p_i is exponentially distributed with the average change rate λ_i. Statistical prediction models are suggested to estimate change rates λ_i based on a small number of features [1, 6, 22]. Also, recently new protocols were developed allowing Web masters to supply additional information about the Web pages of the site for crawlers including the URLs of the site, their last update, frequency of change, etc [10, 19].

Crawling of the Web for the purpose of search engines has received a lot of attention. Key issues here are efficiency [12, 7], freshness [3, 17], importance [16], relevance to keyword queries [4, 8, 21] and broad coverage of the Web [19]. In contrast, archive curators are interested in a complete capture of a site. In addition, a visit and then a revisit of all pages of the site enables them to reason about the quality of the archive.

Scheduling of jobs to available processors in real-time systems [2, 18, 15] is similar to scheduling of confidence intervals. Scheduling in real-time systems maximizes utilization of processors, requiring that the time intervals of execution of individual jobs on the same processor do not intersect. In contrast, SOLAR requires that all intervals intersect. This changes the settings and application of real-time scheduling algorithms is not straight forward.

3. MODEL AND NOTATION

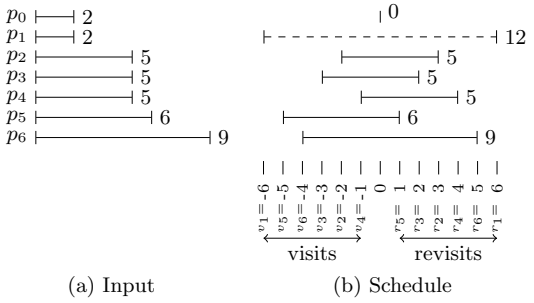
A Web archive, archive for short, periodically (e.g., on a weekly or monthly basis) crawls a large number of sites, or more intensively in aftermaths of big sports events, natural disasters, political scandals, etc. Each crawl aims to fetch *all* pages $\mathbb{P} = (p_0, \ldots, p_n)$ of the site. Crawling needs to observe the politeness requirements of a site with pauses of several seconds or even a minute between successive HTTP requests. Thus, an entire site capture may span hours or days. (The crawler crawls many sites in parallel for throughput.) When a new site crawl starts, we assume that (i) either the URLs of all pages are known upfront or (ii) at least one entry page is known from which the crawl can explore the site's link graph. The former is an assumption made by the offline version and analytical investigation of the SOLAR model (Sections 4.1–4.2). The condition is then relaxed in the online version of the SOLAR algorithm (Section 5).

We optimize crawling for an archive that can be viewed as a precise representation of the Web site as of a *single timepoint*.

We visit every page of the site twice: the first *visit* to fetch the page, and the *revisit* to test if the page is sharp (i.e., did not change). A schedule assigns visit and revisit positions to each page. We require that all visits are before any revisits. Therefore, the visit-revisit intervals of all pages must have a non-empty intersection for which, in the ideal case, all pages are collectively sharp as of the same timepoint.

Testing for the absence of changes could be implemented by the HTTP If-Modified-Since header, but this is not a reliable method for many kinds of Web pages and its cost is not that much lower than for a full get. Therefore, we always download pages at revisits and compare their contents against the previously seen ones, using the usual stages from hash-signature comparisons to full-contents testing. This way, we have explicit control over what exactly we want to view as a true change of a page. For example, we could ignore irrelevant modifications of page footers (e.g., a CMS-generated timestamp) or banner ads; but we consider these if we care (e.g., if banner ads are legally relevant).

As total sharpness of an entire site capture is often not feasible at all, SOLAR aims to maximize the number of sharp pages in a site capture by judicious scheduling of visit-revisit intervals based on estimated *confidence intervals* for the absence of changes in a page (explained below). Figure 1 illustrates this by example. The algorithm considers the lengths $\mathbb{I} = (I_0, \ldots, I_n)$ of the confidence intervals (the numbers on the right end of the intervals in Figure 1(a)) to schedule the visits $\mathbb{V} = (v_0, \ldots, v_n)$ and revisits $\mathbb{R} = (r_0, \ldots, r_n)$ so that (i) v_i and r_i get unique positions, except for one page for which visit and revisit collapse onto the same timepoint, (ii) there is at least a minimum politeness delay Δ between any two download positions, (iii) all intervals overlap, and (iv) the pair v_i, r_i forms an interval of length less than or equal to I_i. An equivalence relation between scheduling of pages and scheduling of confidence intervals immediately follows Condition (iv).



(a) Input (b) Schedule

Figure 1: Scheduling of Confidence Intervals

Conditions (i) and (ii) arise from the politeness etiquette and must be obeyed by any schedule. Violation of Condition (iii) for page p_i means that page p_i cannot be guaranteed to be captured as of the same timepoint as the other pages. This situation should be tolerated only if unavoidable (e.g., because of late discovery of pages in the online case, or because the page is changing extremely frequently). Violation of Condition (iv) would entail that a page has a high risk of being changed between its visit and revisit. We can always shorten the timespan between visit and revisit (if the other conditions can still be ensured), but should avoid making it too long. We may face a situation, though, where some pages have very short confidence intervals or the appropriate download slots are already used to schedule other pages. Then we may decide to postpone scheduling these pages, which we call *hopeless*, and will arrive at a (still large) fraction of the site for which sharpness is statistically guaranteed. Figure 1(b) shows the output by SOLAR for the input in Figure 1(a). Here SOLAR successfully schedules six intervals: five intervals (pages $p_2, \ldots, p_6$) are of maximal length, interval of page p_0 is shortened, and page p_1 is hopeless.

The lengths I_i of the confidence intervals can be computed by the commonly used Poisson model for changes of the Web. Given the average change rate λ_i of page p_i the probability that page p_i does not change in an interval of length I_i is

$$P(\text{no change of } p_i \text{ in interval of length } I_i) = \frac{e^{-\lambda_i I_i}(\lambda_i I_i)^0}{0!}.$$

If we require that this probability is above a treshold τ (e.g., 90 percent), then the length of the confidence interval becomes

$$I_i = \lambda_i^{-1} \log \tau^{-1}. \quad (1)$$

The change rates can be statistically predicted based on MIME types, depths within the site, and other page properties. Alternatively, webmasters can provide change rates through their sitemap.xml files. For brevity, we will omit the word "confidence" and call the confidence intervals just intervals and the lengths of the confidence intervals as just "lengths".

For simplicity, we focus on a discretized version of the problem: we assume that all intervals are of integer lengths ($\mathbb{I} \subset \mathbb{Z}^+ \cup 0$), and the delay between two download positions is exactly Δ. To simplify mathematical equations we set Δ to 1, and denote the visit slots with negative integers ($\mathbb{V} \subset \mathbb{Z}^-$) and the revisit slots with positive integers ($\mathbb{R} \subset \mathbb{Z}^+$).

4. SOLAR SYSTEM

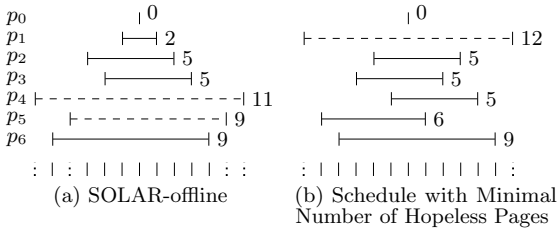
4.1 Schedules and SOLAR-offline

SOLAR-offline takes the lengths $\mathbb{I}$ as input and arranges the largest possible number of non-hopeless intervals. The start and end points of the intervals get unique visit and revisit positions. SOLAR-offline schedules the revisit positions first and the visit positions fall out automatically.

Definition 4.1. (Schedule, Hopeless and Hopeful Pages.) *Let $\mathbb{I} = (I_0, \ldots, I_n)$ be the lengths of intervals. Let $\mathbb{V} = (v_0, \ldots, v_n)$ ($v_i \leq 0, i = 0, \ldots, n$) be the positions of visits and $\mathbb{R} = (r_0, \ldots, r_n)$ ($r_i > 0, i = 0, \ldots, n$) be the positions of revisits such that: (i) all download positions are unique: $v_i \neq v_j$ for all $i \neq j$, $i, j \in \{0, \ldots, n\}$; $v_i \neq r_j$ for all $i, j \in \{0, \ldots, n\}$ except one pair (v', r'), for which $v' = r' = 0$, and (ii) all intervals are scheduled: $\cup_i\{-v_i\} = \cup_i\{r_i\} = \{0, 1, \ldots, n-1\}$.*

Then $\mathbb{V}$ and $\mathbb{R}$ is a schedule of $\mathbb{I}$. Length I_i in schedule $\mathbb{V}$ and $\mathbb{R}$ is hopeless if the corresponding visit-revisit pair is longer than the length: $r_i - v_i > I_i$. Otherwise the interval I_i is hopeful. Correspondingly, we call the page p_i hopeless or hopeful too.

Example 4.1. (Schedule, Hopeless and Hopeful Pages.) *Assume that the lengths are as in Figure 1(a). Then $v_0 = r_0 = 0$, $v_1 = -1, r_1 = 1$, $v_2 = -3, r_2 = 2$, $v_3 = -2, r_3 = 3$ $v_6 = -5, r_6 = 4$, (hopeful pages), $v_4 = -4, r_4 = 5$ and $v_5 = -6, r_5 = 6$ (hopeless pages) is a schedule of the intervals. The schedule is illustrated in Figure 2(a).*



(a) SOLAR-offline (b) Schedule with Minimal Number of Hopeless Pages

Figure 2: SOLAR schedules

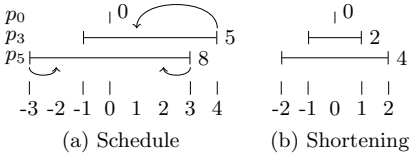
Given all intervals in advance, the SOLAR-offline algorithm schedules the pages based on the following principles: First, the algorithm is greedy. It schedules short intervals first in ascending order of length, since there is less leeway for them. Second, the scheduling is iterative: given $i-1$ scheduled hopeful pages, the algorithm looks for the visit-revisit pair so the interval does not exceed the given length. Third, if no visit-revisit pair can be found in the ith planning step the page is declared hopeless. All hopeless pages are scheduled at the end (after all hopeful pages), in descending order of their interval lengths.Pseudocode for the SOLAR-offline method is given in Algorithm 1.

input : lengths sorted descending: $I_0, \ldots, I_n$
output: visit-revisit schedule: $\mathbb{V}$ and $\mathbb{R}$
list of hopeless pages: $\mathbb{H}$
begin
  Initialize the schedule $\mathbb{V} = \mathbb{R} = (0)$
  Initialize the list of hopeless pages $\mathbb{H} = \emptyset$
  // Identify and schedule hopeful pages
  for $i = 1, \ldots, n$ **do**
    Find smallest $r_i \in \mathbb{Z}^+ \setminus \mathbb{R} : v_i \in \mathbb{Z}^- \setminus \mathbb{V}$
    if *such* (v_i, r_i) *exists* **then**
      $\mathbb{V} = \mathbb{V} \oplus v_i \qquad \mathbb{R} = \mathbb{R} \oplus r_i$
    else
      $\mathbb{H} = \mathbb{H} \oplus I_i$
  end
  // Schedule hopeless pages
  Sort $\mathbb{H}$ ascending
  foreach $h \in \mathbb{H}$ **do**
    Find largest $v \in \mathbb{Z}^- \setminus \mathbb{V}$ and smallest $r \in \mathbb{Z}^+ \setminus \mathbb{R}$
    $\mathbb{V} = \mathbb{V} \oplus v \qquad \mathbb{R} = \mathbb{R} \oplus r$
  end
end

Algorithm 1: SOLAR-offline

4.2 Shrinking a Schedule

During the iterative construction of the SOLAR, we always use the full lengths of the pages' confidence intervals. That is, the lengths are never shortened for possibly achieving higher number of hopeful pages in later iterations. This may produce a much wider schedule than the number of download positions $[-n, n]$ (cf. Figure 3(a)), or it may schedule some hopeless pages on visit-revisit positions in between the hopeful ones (cf. page p_5 in Figure 2(a)). In both cases we can improve the schedule: the schedule of hopeful pages should be shortened and the hopeless pages should be scheduled afterwards.



(a) Schedule (b) Shortening

Figure 3: Shortening of Intervals

Example 4.2. (Shrinking of a Schedule.) *Consider the schedule in Figure 3(a). The shrink schedule is illustrated in Figure 3(b).*

5. SOLAR-ONLINE

Section 4 assume that all pages of a site are known to the archival crawler in advance. In this section we relax the assumption and build a schedule iteratively as we discover pages on the Web site. We still assume that the interval lengths can be predicted for each page, using a previously trained predictor based on page type and other features.

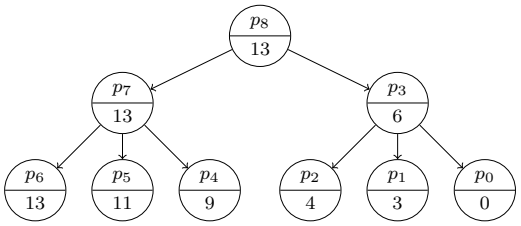
SOLAR-online is based on SOLAR-offline. We need some adjustments to turn SOLAR-offline into an incremental SOLAR-online strategy. SOLAR-offline aims to place pages with short intervals close to the middle point of the entire visit-revisit schedule (i.e., left-most revisits) and replaces the one short with two long afterwards. SOLAR-online has a similar rationale, but has the handicap that it learns about interval lengths only as it detects new pages. This entails difficulties, since SOLAR-offline uses the positions close to the middle, however the schedule starts with the left-most revisit positions.

Our SOLAR-online algorithm integrates the SOLAR-Offline and shrinking incrementally with the following principles. SOLAR-online starts with a set of seeds and aims to schedule the longest available interval. Selection of the longest interval gives as much room as possible for scheduling of shorter intervals around the middle point. If the longest interval is not possible to schedule, due to unavailable revisit position, we continue with the second longest interval. If no schedulable interval exists we schedule the shortest interval and declare it to be hopeless. The revisit positions of the hopeless pages are assigned at the end of all pages in reverse order. Once all pages are visited, we shrink the revisits. Individual steps of the algorithm are given in Algorithm 2.

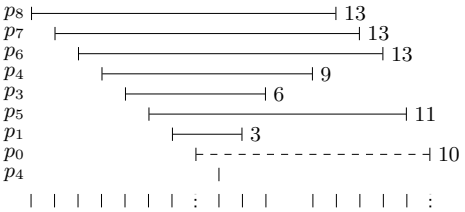
Example 5.1. (Online Schedule.) *Consider the Web Site as in Figure 4(a) (here the lengths of the intervals are pointed out in the bottom part of the nodes) and the seed p_9. SOLAR-online downloads the seed and discovers pages p_7 and p_3. p_7 has the longest interval, and therefore is scheduled and downloaded next. The list of detected pages increases to $\mathbb{I}^E = (p_6, p_5, p_4, p_3)$. Next p_7, and afterwards p_6 are the longest and schedulable pages, and therefore they*

```
input  : Set of seeds p_0^s, ..., p_m^s
         Number of pages in site n
         Confidence threshold τ
output: Downloaded sequence (p_0^D, ..., p_n^D)
begin
    Initialize the detected lengths sorted descending
    I^E = (I_0^s, ..., I_m^s)
    Initialize the revisits R = ()
    Initialize current visit position v = −n
    while I^E ≠ ∅ do
        foreach I_i ∈ I^E do
            Let r = v + I_i
            if r > 0 and r ∉ R then
                // page p_i is hopeful
                Select page p = p_i for download
                Exit the foreach loop
            end
        end
        if no page was downloaded in the foreach loop then
            // page p_j is hopeless
            Let I_j ∈ I^E be the shortest interval
            Select page p = p_j for download
        end
        Download p
        Update detected pages I^E = I^E ∪ I^D(p_i)
        if r > 0 and r ∉ R then
            Use the revisit position R = R ∪ r
        end
    end
    shrink (R)
    Schedule revisits of hopeless pages in reverse order
end
```

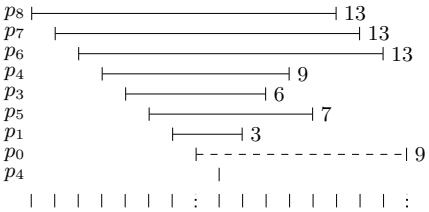
Algorithm 2: SOLAR-online



(a) Web Site



(b) Online Schedule before Shrinking



(c) Online Scheduling after Shrinking

Figure 4: Online Example

are scheduled and visited. Now the next longest page is p_5 (of length 11), however it is not possible to schedule a revisit for the page, and p_3 is scheduled and visited instead. Pages p_5 and p_1 follow in a similar fashion. Since no page can be scheduled at this point, the shortest interval (page p_0) is visited and declared hopeless. Eventually, p_4 is visited and the visit part of the algorithm is completed. Now we shrink the revisits (the shortest intervals are shrunk having the highest priority). This way, pages p_5 (from length 11 to 7) and p_0 (from length 10 to 9) are shortened.

6. EXPERIMENTS

In this section we experimentally evaluate SOLAR-offline and SOLAR-online and compare them with Hottest-middle and SHARC-threshold [20, 9]. Both Hottest-middle and SHARC-threshold strategies are two-visit strategies: visit-revisits form $[-i, i]$ intervals. In Hottest-middle, the most changing pages are allocated for the shortest intervals. SHARC-threshold uses a similar allocation scheme with a difference that some too much changing pages are declared "hopeless" and allocated to intervals after all "hopeful" pages are scheduled.

SOLAR-offline requires the knowledge of all URLs of the site in advance. This knowledge together with the change rates is available only for Web sites with sitemaps. In case the site does not provide a sitemap, SOLAR-online should be used instead. The careful setup of the synthetic experiments allowed us to generate and keep the history of all changes of the pages and apply both online and offline strategies.

The methods are evaluated on two synthetic datasets: the skewed and the smooth one. The change rates in the skewed dataset decrease quite rapidly. In the dataset the change rates are distributed exponentially with the following change rates: $\lambda_0 = 1.0$, $\lambda_{i+1} = \lambda_i / 1.06$, if $i \bmod 9 < 3$, and $\lambda_{i+1} = \lambda_i / 1.001$, otherwise. This dataset consists of a few very changing pages, with most of the pages changing infrequently and mimics a rather static Web sites (including universities and companies). In the smooth dataset, the changes are again distributed exponentially, however the decrease of the change rate from λ_i to λ_{i+1} is not that extreme: the change rates are generated so that the confidence intervals are of lengths $1, 2, \ldots, n$. The number of pages in each dataset is set to $n = 1000$. This mimics more dynamic Web sites including blogs and discussion forums.

In each experiment we execute the schedules and compute the ##pages (number of pages that did not change between the visit and revisit). This shows the number of pages that are mutually coherent at a virtual timepoint during the crawl.

Table 1 summarizes the offline experiment. The SOLAR (cf. 710 for the skewed and 515 for the smooth ##pages) outperforms both the Hottest-middle (cf. 647 and 493 ##pages)

	Skewed		Smooth	
Algorithm	E(##pages)	##pages	E(##pages)	##pages
SHARC-threshold	684.286	682	492.864	493
Hottest-middle	649.577	647	492.864	493
SOLAR-offline	704.041	710	515.249	515

Table 1: Offline

	Skewed		Smooth	
Algorithm	E(##pages)	##pages	E(##pages)	##pages
Hottest-middle	672.271	669	455.512	449
SOLAR-online	680.421	679	459.799	455

Table 2: Online

and the SHARC-threshold (cf. 682 and 493 ##pages). The careful setup of the experiments allowed us to compute the expected number of sharp pages (cf. the E(##pages) column in the table) and eliminated the effect of randomness. Again, the evaluation shows that SOLAR outperforms the competitors and the difference is statistically significant.

Table 2 summarizes the online experiment. This experiment is based on the same datasets as in Table 1, however we do not assume that all URLs are known in advance. In contrast, we assume that all pages form Web site with a tree-link structure, where p_0 is the root, p_0 has 20 children: $p_1, p_2, \ldots, p_{20}$, and $p_i (0 < i < 99)$ has 10 children each: $p_{i*10+11}, p_{i*10+12}, \ldots, p_{i*10+20}$, and we discover the Web pages as we crawl. The online version of the Hottest-middle visits the coldest discovered page in each iteration, and revisits in change-descending order. SHARC-threshold is not an online strategy and is not included in the experiment. Similarly to the offline experiment, SOLAR outperforms the competitors and the results are statistically significant. SOLAR-online performs slightly worse than the SOLAR-offline counterpart (cf. 679 vs 710 and 455 vs 515 ##pages in Tables 1–2), since all URLs are not known in advance. Controversy, Hottest-middle may enjoy the online settings and the ##pages increase (cf. 647 vs 669 but 493 and 455). Possible reason for that is that the scheduled intervals in the online version of Hottest-middle are less regular and closer to the scheduled intervals of the SOLAR-online.

7. CONCLUSION

This paper has devised and investigated scheduling methods for capturing Web sites in a way that allows archive curators and analysts to view all or all but a few pages of an archived site as of the very same timepoint. Moreoever, our techniques give guarantees on this jointly sharp state of all pages on the site, or can identify the site fractions with this property and can precisely pinpoint the exceptions (if any). This enables a new dimension of exploiting Web archives, ensuring their data quality, and making tangible claims about the presence or absence of legally relevant contents on (multiple pages of) a site.

We believe that Web archive quality is an underexplored issue of great practical importance. This paper is certainly not the final word on this topic. As crawling for an archive is fundamentally different from crawling for a search engine, entirely new architectures may have to be pursued. For example, instead of visiting each page twice in a periodically repeated capture, we could alternatively pursue a continuous crawl with page-specific visiting rates, and reason about sharp sites at any given timepoint instead of focusing on the best timepoints during one of the periodic crawls. Other aspects to consider in our future work are the possible availability of sitemap information that can provide additional guidance to the scheduling strategies, introduction of weights of pages to the model, and possibilities to repair incoherent pages with re-crawls and post-processing of the archives.

Acknowledgements

This work is supported by the 7^{th} 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.

8. REFERENCES

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

[2] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. (Seffi) Naor, and B. Schieber A unified approach to approximating resource allocation and scheduling. In *J. ACM*, 48(5): pp. 1069–1090, 2001.

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

[4] C. Castillo, M. Marin, A. Rodriguez, and R. Baeza-Yates. Scheduling algorithms for web crawling. In *LA-WEBMEDIA '04*, pp. 10–17, 2004

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

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

[7] J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through url ordering. In *WWW '07*, pp. 161–172, 2007.

[8] J. Cho and U. Schonfeld. Rankmass crawler: a crawler with high personalized pagerank coverage guarantee. In *VLDB '07*, pp. 375–386, 2007.

[9] D. Denev, A. Mazeika, M. Spaniol, and G. Weikum. Sharc: Framework for quality-conscious web archiving. *PVLDB*, 2(1):586–597, 2009.

[10] Google, Yahoo, and Microsoft. Sitemaps. Website, 2008. `http://www.sitemaps.org/`.

[11] R. Klamma and C. Haasler. Wikis as social networks: Evolution and dynamics. In *2nd SNA-KDD*, 2008.

[12] H.-T. Lee, D. Leonard, X. Wang, and D. Loguinov. Irlbot: scaling to 6 billion pages and beyond. In *WWW '08*, pp. 427–436, 2008.

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

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

[15] J. Mestre. Adaptive local ratio. In *SODA '08*, pp. 152–160, 2008.

[16] M. Najork and J. L. Wiener. Breadth-first search crawling yields high-quality pages. In *WWW '01*, pp. 114–118, 2001.

[17] C. Olston and S. Pandey. Recrawl scheduling based on information longevity. In *WWW '08*, pp. 437–446. 2008.

[18] A. Panconesi and M. Sozio. Fast distributed scheduling via primal-dual. In *SPAA '08*, pp. 229–235. 2008.

[19] U. Schonfeld and N. Shivakumar. Sitemaps: above and beyond the crawl of duty. In *WWW '09*, pp. 991–1000, 2009.

[20] M. Spaniol, D. Denev, A. Mazeika, G. Weikum, and P. Senellart. Data quality in web archiving. In *WICOW*, pp. 19–26, 2009.

[21] Q. Xu and W. Zuo. First-order focused crawling. In *WWW '07*, pp. 1159–1160, 2007.

[22] J.-M. Yang, R. Cai, C. Wang, H. Huang, L. Zhang, and W.-Y. Ma. Incorporating site-level knowledge for incremental crawling of web forums: a list-wise strategy. In *KDD '09*, pp. 1375–1384, 2009.