# Recrawl Scheduling Based on Information Longevity

Christopher Olston
Yahoo! Research
Santa Clara, California
olston@yahoo-inc.com

Sandeep Pandey
Carnegie Mellon University
Pittsburgh, Pennsylvania
spandey@cs.cmu.edu

## ABSTRACT

It is crucial for a web crawler to distinguish between ephemeral and persistent content. Ephemeral content (e.g., quote of the day) is usually not worth crawling, because by the time it reaches the index it is no longer representative of the web page from which it was acquired. On the other hand, content that persists across multiple page updates (e.g., recent blog postings) may be worth acquiring, because it matches the page's true content for a sustained period of time.

In this paper we characterize the longevity of information found on the web, via both empirical measurements and a generative model that coincides with these measurements. We then develop new recrawl scheduling policies that take longevity into account. As we show via experiments over real web data, our policies obtain better freshness at lower cost, compared with previous approaches.

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

## General Terms

Algorithms, Experimentation, Measurement, Theory

## 1. INTRODUCTION

Modern search engines rely on *incremental web crawlers* [3] to feed content into various indexing and analysis layers, which in turn feed a ranking layer that handles user search queries. The crawling layer has two responsibilities: downloading new pages, and keeping previously-downloaded pages fresh. In this paper we study the latter responsibility, and consider what *page revisitation policy* a crawler should employ to achieve good freshness.

Good freshness can be guaranteed trivially by simply revisiting all pages very frequently. However, doing so would place unacceptable burden on the crawler and leave few resources for discovering and downloading new pages. Hence we seek revisitation policies that ensure adequate freshness while incurring as little overhead as possible.

Prior work on this problem has focused on two factors when determining how often to revisit each web page:


*WWW 2008*, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

1. *Change frequency* (how often the content of a page is updated by its owner) [4, 6, 7].
2. *Relevance* (how much influence the new/old page content has on search results) [12, 13].

A third important factor that has thus far been ignored is *information longevity*: the lifetimes of content fragments that appear and disappear from web pages over time. Information longevity is not strongly correlated with change frequency, as we show in Section 3.2.

It is crucial for a crawler to focus on acquiring longevous (i.e., persistent) content, because ephemeral content such as advertisements or the "quote of the day" is invalid before it hits the index. Besides, a search engine typically has little interest in tracking ephemeral content because it generally contributes little to understanding the main topic of a page.

Real web pages differ substantially in the longevity of their content. Figure 1 shows the temporal evolution of two web pages, in terms of the presence or absence of individual *content fragments*. (The manner in which we divide pages into fragments is not important for the present discussion.) In each graph the horizontal axis plots time, and the vertical axis shows unique fragments, sorted by the order in which they first appear on the page. Each horizontal stripe represents a fragment; the left end-point is the time at which the fragment first appears on the page; the right end-point is the time at which the fragment disappears.

Page A has a small amount of static content, and a large amount of highly volatile content that consists of dynamically generated text and links apparently used to promote other web pages owned by the same organization. Page B contains a mixture of static content, volatile advertisements, and a third kind of content manifest as horizontal stripes roughly 30 to 60 days long. Page B is part of a cooking site that displays a sliding window of recent recipes.

Contrasting Pages A and B, we appreciate the importance of considering the lifetime of information when crafting a page revisitation policy. Pages A and B both undergo frequent updates. However Page A is not worth revisiting often, as most of its updates simply replace old ephemeral information with new ephemeral information that would be pointless for the search engine to index. Page B, on the other hand, is adding information that sticks around for one to two months (i.e., recipes) and might be worthwhile to index, making Page B worth revisiting often. (We estimate the prevalence on the web of pages like A and B in Section 3.3.)

Information longevity has been measured in various ways in previous work [1, 8, 9]. However we are not aware of any work that considers its role in recrawl effectiveness.

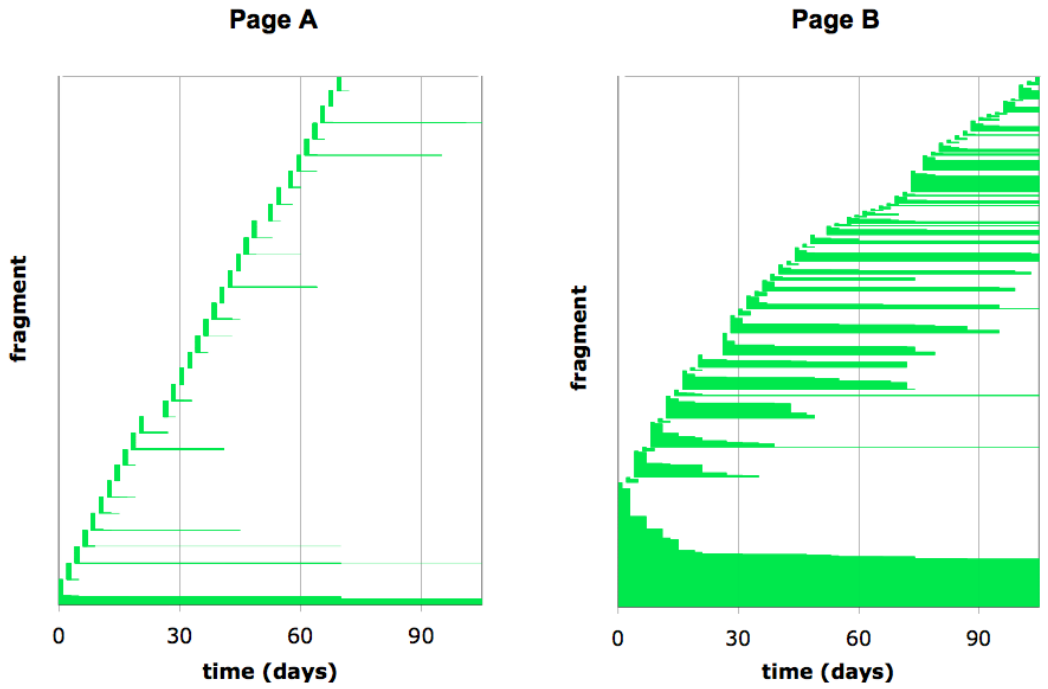


**Figure 1: Temporal behavior of two web pages.**

## 1.1 Contributions

This paper makes the following contributions:

- Identification of information longevity as a key factor in crawler performance.
- Longevity measurements of real web content, and a generative model that accounts for the observed characteristics (Section 3).
- New page revisitation policies that take into account information longevity in addition to the usual factors, and avoid wastefully downloading ephemeral content (Section 4).
- Empirical study of the *online* revisitation problem, where policies must sample page update behavior on the fly, in order to learn how pages change and use the learned information to schedule future revisitations (Section 5).

The revisitation policies we propose are highly practical. They incur very little per-page space and time overhead. Furthermore, unlike some previously-proposed policies, ours do not rely on global optimization methods, making them suitable for use in a large-scale parallel crawler. Lastly, our policies automatically adapt to shifts in page change behavior.

Our revisitation policies are based on an underlying theory of optimal page revisitation, presented next.

# 2. THEORETICAL FRAMEWORK

The scenario we consider is as follows. A crawler has acquired the content associated with a set of pages $\mathcal{P}$. Each page $P \in \mathcal{P}$ may undergo autonomous updates after the crawler's first visit, causing the web-resident copy of $P$ to drift from the crawler's copy. A *page revisitation policy* governs the schedule with which the crawler *refreshes* its local copy of $P$, by revisiting the web-resident copy and reacquiring its content, in order to bring the local copy in sync with the web copy, if only temporarily.

## 2.1 Metrics

The following metrics are of interest:

- **Refresh cost:** the total resources spent refreshing web pages. Following previous work we make the approximation that every refresh event incurs equal resource utilization, and measure refresh cost as the total number of refresh events during in a given time period.
- **Divergence:** the degree to which the crawler's local cache of page content differs from the true web-resident content, averaged across pages and across time.

Mathematically, we have a *page divergence function* $D(P, P')$ of two copies of a page: the true copy $P$ and cached copy $P'$. Immediately following a refresh event, $P = P'$ and $D(P, P') = 0$. Later, if $P$ undergoes updates such that $P \neq P'$, $D(P, P') > 0$. (We give the exact form of the $D(\cdot)$ function in Section 2.1.1.)

The overall divergence of the web cache at a given time is defined as a weighted average across pages:

$$D(\mathcal{P}) = \frac{1}{|\mathcal{P}|} \sum_{P \in \mathcal{P}} W_P \cdot D(P, P') \qquad (1)$$

where $W_P$ denotes the importance or relevance weight of page $P$, e.g., $P$'s pagerank or embarrassment coefficient [13].

Averaging across a given time interval $(t_1, t_2)$, we have:

$$D(\mathcal{P}, t_1, t_2) = \int_{t=t_1}^{t_2} \frac{1}{|\mathcal{P}|} \sum_{P \in \mathcal{P}} W_P \cdot D(P(t), P'(t))\, dt \qquad (2)$$

where $P(t)$ denotes the true content of page $P$ at time $t$, and $P'(t)$ denotes the (possibly out-of-date) content cached for $P$ as of time $t$.

### 2.1.1 Page Divergence Metric

Modern web pages tend to consist of multiple *content regions* stitched together, e.g., static logos and navigation bars, dynamic advertisements, and a central region containing the main content of the page, which is also dynamic but behaves quite differently from advertisements. Consequently, page divergence metrics that model a page as an atomic unit of content are no longer a good fit.

The page divergence metric we propose is called *fragment staleness*. It measures the fraction of content fragments that differ between two versions of a page. Mathematically, we treat each page version as a set of fragments and use the well-known Jaccard formula for comparing two sets:

$$D(P, P') = 1 - \frac{|F(P) \cap F(P')|}{|F(P) \cup F(P')|} \quad (3)$$

where $F(P)$ denotes the set of fragments that comprise $P$.

As far as how we divide a page into fragments, we require a method that is amenable to efficient automated computation, because we will use it in our crawling algorithms. Hence we rule out sophisticated semantic approaches to page segmentation, which are likely to be too heavyweight and noisy for this purpose. A simple and robust alternative that is well aligned with the way search engines tend to treat documents is to treat sequences of consecutive words as coherent fragments. We adopt the well-known *shingling method* [2], which emits a set of content fragments, one for each word-level $k$-gram ($k \geq 1$ is a parameter).

Our rationale for adopting the fragment staleness metric is as follows. Consider what would happen if the cached copy of a page contains some fragments that do not appear in the web-resident version: The search engine might display the page's URL in response to a user's search query string that matches the fragment; yet if the user clicks through she may find no relevant content. Conversely, if a query matches some fragment in the web-resident copy that does not appear in the cached copy, the search engine would overlook a potentially relevant result. Fragment staleness gives the likelihood of introducing false positives and false negatives into search results.[1]

Fragment staleness generalizes the "freshness" metric of [4, 6], which we refer to in this paper as *holistic staleness* (staleness is the inverse of freshness). Holistic staleness implicitly assumes one content fragment per page and yields a Boolean response: each page is either "fresh" or "stale."

## 2.2 Optimal Recrawling

A theory of optimal recrawling[2] can be developed in a manner that is independent of any particular choice of per-page divergence metric. The goal is a page revisitation policy that achieves minimal time-averaged divergence within a given refresh cost budget. For the sake of the theory let us assume that for each page $P$, divergence depends only on the time $t_P$ since the last refresh of $P$. Assume furthermore that, other than in the case of a refresh event, divergence does not decrease. Under these assumptions we can re-express divergence as $D(P(t), P'(t)) = D^*_P(t - t_P)$, for some monotonic function $D^*_P(\cdot)$.

Let $T$ be any nonnegative constant. It can be shown using the method of Lagrange Multipliers that the following revisitation policy achieves minimal divergence compared with all policies that perform an equal number of refreshes:

*At each point in time $t$, refresh exactly those pages that have $U_P(t - t_P) \geq T$, where*

$$U_P(t) = t \cdot D^*_P(t) - \int_0^t D^*_P(x)\, dx \quad (4)$$

Intuitively speaking, $U_P(t)$ gives the *utility* of refreshing page $P$ at time $t$ (relative to the last refresh time of $P$). The constant $T$ represents a *utility threshold*; we are to refresh only those pages for which the utility of refreshing it is at least $T$. The unit of utility is **divergence** $\times$ **time**.

## 2.3 Discussion

Figure 2 shows divergence curves for the two web pages illustrated in Figure 1. The horizontal axis of each graph plots time, assuming the crawler refreshes both pages at time 0. The vertical axis plots divergence, under the fragment staleness metric introduced in Section 2.1.1.[3]

If we expect the divergence curve of a page to behave similarly after the next refresh as it did after the $t = 0$ refresh, then we should favor refreshing Page B over refreshing Page A at the present time. The reason is that refreshing Page B gives us more "bang for our buck," so to speak, than Page A. Utility at time $t$ is given by the area of the shaded region. Page B has higher utility than Page A, which matches our intuition of "bang for buck."

In addition to giving the intuition behind the utility formula derived in Section 2.2, this example also illustrates the appropriateness of fragment staleness as a divergence metric. Although our generic theory of optimal revisitation of Section 2.2 can be applied to holistic staleness as well (see Appendix A; our result matches that of [4]), the nature of the holistic staleness metric can lead to undesirable behavior: Under a revisitation policy that optimizes holistic staleness, Pages A and B, which undergo equally frequent changes, are treated identically—either both ignored or both refreshed with the same frequency.

One may wonder whether a simpler utility function such as $U_p(t) = D^*_P(t)$ (i.e., prioritize refreshing based on current divergence) may work nearly as well as the theoretically optimal formula given in Equation 4. We have verified empirically via simulation over real web data (see Section 3.1) that refreshing according to Equation 4 yields significantly improved staleness for the same refresh cost, compared with using the naive form $U_p(t) = D_P(t)$. In fact, the naive form even performs worse than uniform refreshing, which is to be expected in light of the findings of [4].

[1] To account for nonuniform importance or relevancy of fragments, one can extend our formula to associate a numeric weight with each fragment, along the lines of [12]. The techniques presented in this paper are not tied to uniform weighting of fragments.

[2] This theory was first presented in [10], in a somewhat different context.

[3] In Figure 2 divergence usually increases or remains constant over time, but occasionally undergoes a slight decrease. These decreases are due to a few fragments reappearing after they had previously been removed from the page. (The visualizations of Figure 1 do not show lapses in fragment presence.)

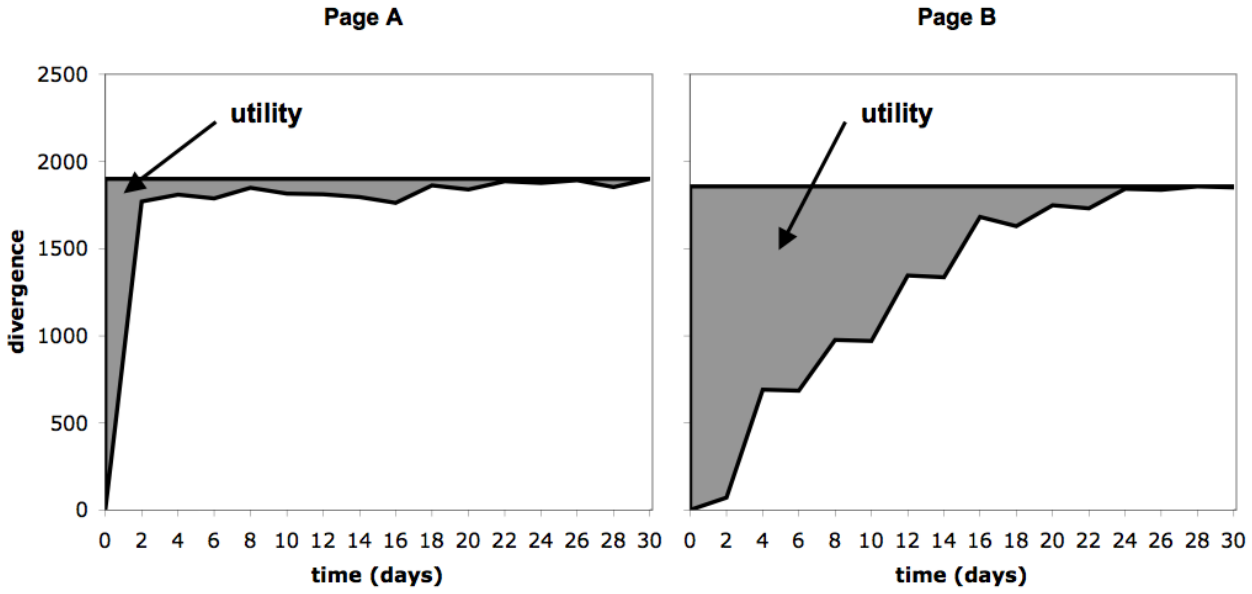


**Figure 2: Two divergence graphs, with utility.**

# 3. ANALYSIS OF WEB DATA

In this section we study the information longevity characteristics of real web pages. After describing our data sets in Section 3.1 we measure the (lack of) correlation between information longevity and change frequency in Section 3.2. Then in Section 3.3 we propose and validate a generative model for dynamic web content. Lastly in Section 3.4 we measure the potential performance gain from use of a longevity-aware revisitation policy.

## 3.1 Data Sets

We use two real web data sets:

- **Random.** We obtained a random sample of 10,000 URLs from Yahoo's crawled collection, and configured a crawler to download the content of each URL once every two days over the course of several months in 2006. A total of 50 snapshots were obtained.
- **High-quality.** Our second data set is based on a random sample of 10,000 URLs from the OpenDirectory [11]. We consider this data set to consist of much higher quality pages than random ones, on average. For this data set we obtained 30 temporal snapshots, again by downloading each page every second day.

Within each data set we assign uniform importance weights ($W_P = 1$ for every page).

The high-quality data set is of significantly more interest to study than the random data set, because crawlers typically avoid recrawling low-quality pages frequently. The interesting question is how frequently to recrawl each high-quality page. For most of our experiments we only report results on the high-quality data set, due to space constraints. In general the two data sets yield similar results.

Unfortunately a few page snapshots are missing from the data, because in some cases the server hosting a given page was unreachable, even after several retries. In these rare cases we substituted the content from the previous snapshot.

To evaluate page revisitation policies (Sections 3.4 and 5), we need a notion of "ground truth." Since we are not the originators of the web pages in our data sets, we do not have access to complete update histories. The only information available to us comes from our bi-daily snapshots, and we need to interpolate. Our method of interpolation is described in Appendix B.

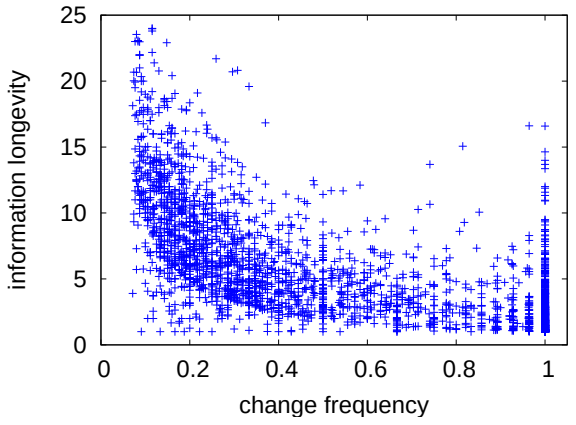


**Figure 3: Change frequency versus information longevity.**

## 3.2 Information Longevity Distribution

Figure 3 plots change frequency versus information longevity for the high-quality data set. Each point denotes a page. Change frequency is computed as the number of snapshots that differ from the previous snapshot, normalized to $[0, 1]$. Information longevity is the average lifetime of fragments (shingles) on the page, in terms of the number of contiguous snapshots in which a given shingle occurred (shingles that were present in the initial or final snapshots are not included, since we are unable to determine their lifetimes).

The fact that the points in Figure 3 are substantially spread out indicates that information longevity is not strongly correlated with change frequency (the correlation coefficient is 0.67). In other words, if we are told the change frequency

of a page, we cannot accurately predict the longevity of its content. This observation, combined with the fact that information longevity is a major determinant of effective page revisitation strategy (Section 1), motivates the study of information longevity on the web.

## 3.3 Generative Model

We model a page as a set of independent content regions, where the temporal behavior of each region falls into one of three categories:

- **Static behavior:** content essentially remains unchanged throughout the page's lifetime, such as templates.
- **Churn behavior:** content is overwritten repeatedly, such as advertisements or "quote of the day."
- **Scroll behavior:** content is appended periodically, such as blog postings or "what's new" items. Typically, as new items are appended, very old items are removed.

Page A in Figure 1 consists of a small static region (templates) and a larger churn region (advertisements). Page B consists of a large static region (templates), a churn region (advertisements) and a scroll region (recipe postings).

In our generative model, each churn or scroll region $R$ has an associated Poisson update process with rate parameter $\lambda_R$ (in our data sets updates closely follow a Poisson distribution, which is consistent with previous findings). In a churn region, each update completely replaces the previous region content, yielding the fragment lifetime distribution:

$$\mathbb{P}(\textit{lifetime} \leq t) = 1 - e^{(-\lambda_R \cdot t)}$$

In a scroll region each update appends a new content item and evicts the item that was appended $K$ updates previously, such that at any given time there are $K$ items present.[4] The fragment lifetime distribution is:

$$\mathbb{P}(\textit{lifetime} \leq t) = 1 - \sum_{i=0}^{K-1} \frac{(\lambda_R \cdot t)^i \cdot e^{-\lambda_R \cdot t}}{i!}$$

Figure 4 plots the lifetime distributions for a churn region and a scroll region with $K = 10$, where both regions have the same update rate $\lambda_R = 0.25$. The two distributions are quite different. Fragment lifetimes tend to be much longer in the scroll case. In fact, in the scroll case it is unlikely for a fragment to have a short lifetime because it is unlikely for ten updates to occur in rapid succession, relative to $\lambda_R$.

### 3.3.1 Model Validation

To validate our generative model, we analyzed the real fragment lifetime distribution of pages from the high-quality data set. We focused on a set of pages that have the same average change frequency $\lambda_P = 0.25$. We assigned an estimated $K$ value to every non-static fragment, based on the number of page update events the fragment "survives" (i.e., remains on the page). For each page we found the most common $K$ value among its fragments.

We obtained three groups of pages: those whose dominant $K$ value is 1 (churn behavior), those dominated by $K = 2$ (short scroll behavior), and those dominated by scroll behavior with some $K > 2$ (the third category was not large enough to subdivide while still having enough data for smooth lifetime distributions). Figure 5 plots the fragment lifetime distributions for the non-static fragments of the three groups of pages, along with corresponding lifetime distributions obtained from our generative model. Each instantiation of the model reflects a distribution of $K$ values that matches the distribution occurring in the data.

[4]In the case of $K = 1$ the scroll model degenerates to the churn model. Nonetheless it is instructive to consider them as two distinct models.

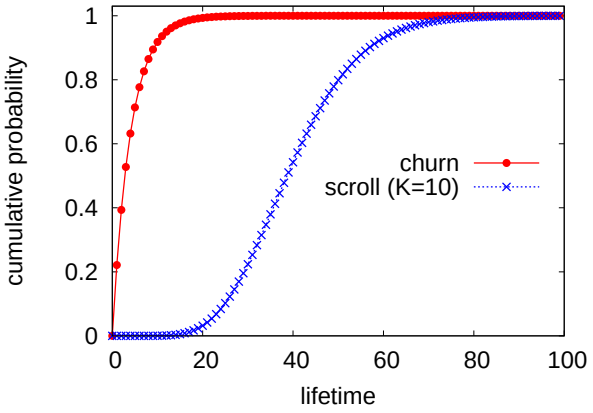


**Figure 4: Cumulative distribution of fragment lifetimes, under two different content evolution models.**

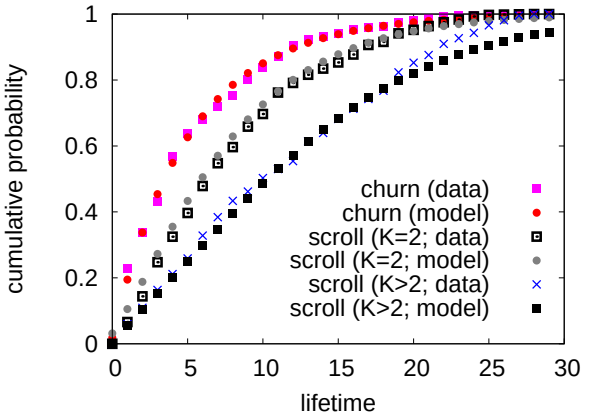


**Figure 5: Actual and modeled lifetime distributions.**

The actual lifetime curves closely match the ones predicted by the model. One exception is that the $K > 2$ curve for the actual data diverges somewhat from the model at the end (top-right corner of Figure 5). This discrepancy is an unfortunate artifact of the somewhat short time duration of our data set: Fragments present in the initial or final snapshot of a page were not included in our analysis because we cannot determine their full lifetimes. Consequently the data is slightly biased against long-lifetime fragments.

Unlike the right-most curve of Figure 4, none of the curves in Figure 5 exhibit the flat component at the beginning. The reason is that in each of our page groups there is at least some $K = 1$ content that churns rapidly.

### 3.3.2 Implications

In our data set, churn behavior accounts for 67% of the non-static fragments, with 33% exhibiting scroll behavior

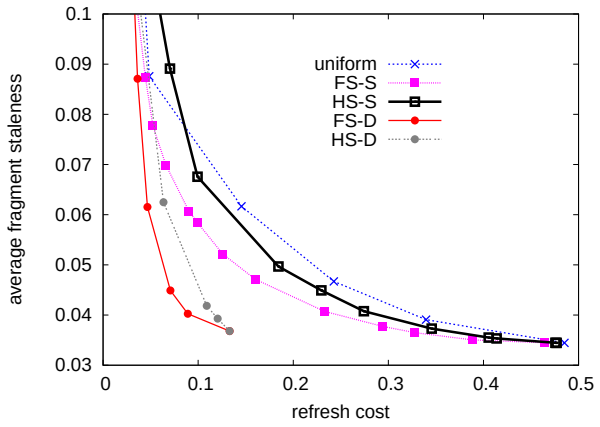


**Figure 6: Performance of offline revisitation policies.**

($K > 1$). Based on these findings we conclude that about a third of the dynamic content on high-quality web sites behaves in a scrolling fashion. The prevalence of scrolling content points to the inadequacy of models that only account for churning content, including the ones used in pervious work on recrawl scheduling.

## 3.4 Offline Page Revisitation Policies

Now that we have developed an understanding of information longevity as a distinct web evolution characteristic from change frequency, we study whether there is any advantage in adopting a longevity-aware web crawling policy. For now we consider *offline* policies, i.e., ones that rely on a-priori measurements of the data set to set the revisitation schedule (we consider online policies in Sections 4 and 5).

The offline policies we consider are:

- **Uniform:** refresh each non-static page with the same frequency. Ignore static pages.
- **Fragment staleness with static input (FS-S):** the optimal policy for fragment staleness as derived in Section 2.2, given a-priori knowledge of the divergence function $D^*_P(\cdot)$. (We estimated $D^*_P(\cdot)$ offline: For each $t$ let $D^*_P(t)$ equal the average divergence between pairs of snapshots separated by $t$ time units.)
- **Holistic staleness with static input (HS-S):** from [4], the optimal policy for holistic staleness, given a-priori knowledge of each page $P$'s change frequency $\lambda_P$. (We estimated $\lambda_P$ offline by dividing the total number of changes by the total time interval.)
- **Fragment staleness with dynamic input (FS-D):** the same as FS-S except instead of using a static, time-averaged divergence function $D^*_P(\cdot)$, at each point in time use the divergence curve that occurred between the time of the previous refresh and the current time.
- **Holistic staleness with dynamic input (HS-D):** the same as FS-D, but substituting holistic staleness as the divergence metric.

Figure 6 shows how these policies perform on the high-quality data set. The x-axis plots normalized refresh cost (1 corresponds to refreshing every snapshot of every page). The y-axis plots average fragment staleness as per Equations 2 and 3 (Section 2.1).[5] On both axes, lower is better.

Roughly half of the pages are completely static, so the largest interesting value of refresh cost is 0.5. Even if we do refresh every non-static page at every opportunity (every two days, in our data set), staleness still does not go to zero due to divergence during the two-day period between refreshes. (Recall from Section 3.1 that we interpolate between snapshots.)

The FS-S policy is roughly the analogue of HS-S. Both assume stationary page change behavior (encoded as $D^*_P(\cdot)$ and $\lambda_P$, respectively). Comparing these two, we see that the fragment-based policy (FS-S) performs significantly better, especially if we consider uniform refreshing as the baseline. The fragment-based policy is geared toward refreshing content of high longevity, and avoids wasting resources on ephemeral content.

Turning to the dynamic policies FS-D and HS-D, we again see the fragment-based policy (FS-D) performing better. Also, both dynamic policies vastly outperform their static counterparts, which points to adaptiveness as a big win.

# 4. ONLINE REVISITATION POLICIES

We now turn our attention to *online* page revisitation policies. An online policy is not given any a priori information about page change behavior, and must expend refreshes in order to learn how pages behave. There is little previous work on this topic.

In this section we present two online revisitation policies. We begin by introducing a data structure common to both policies, called a change profile, in Section 4.1. We then present our two online policies in Sections 4.2 and 4.3. Both policies are based on our underlying theory of optimal refreshing (Section 2.2) and are governed by a utility threshold parameter $T$; we discuss how to choose $T$ in Section 4.4. Lastly, in Section 4.5 we give a method of bounding the risk associated with overfitting to past observations.

## 4.1 Change Profiles

A *change profile* of page $P$ consists of a sequence of (time, divergence) pairs, starting with a *base measurement* ($t_B$, 0) and followed by zero or more subsequent measurements in increasing order of time. Time $t_B$ is called the *base time*, and represents the time at which the change profile was initiated. Each subsequent measurement corresponds to a refresh event performed subsequently to $t_B$ by the revisitation policy. All divergence values in the profile are computed relative to the *base version* $P(t_B)$.

An example change profile is: $\langle$ (10, 0), (12, 0.2), (15, 0.2), (23, 0.3) $\rangle$. This profile indicates that the refresh times for this page include 10, 12, 15 and 23, and that $D(P(10), P(12)) = 0.2$, $D(P(10), P(15)) = 0.2$, and $D(P(10), P(23)) = 0.3$. $P(10)$ is the base version; 10 is the base time.

Our online revisitation policies initiate and maintain change profiles in the following manner. Each time a new refresh of $P$ is performed, a new change profile is created with base time $t_B$ set to the time of the refresh. Each subsequent time

[5]We also generated an analogous graph that plots holistic staleness on the y-axis (omitted due to space constraints). As expected the HS policies outperform the FS policies on that metric.

$t$ that $P$ is refreshed, the profile is extended with the pair $(t,\ D(P(t_B), P(t)))$.

Up to $h$ change profiles are maintained simultaneously, each with a different base version. ($h$ is a positive integer parameter that governs the amount of history tracked by the policy.)

## 4.2 Curve-Fitting Policy

Our first policy is based on fitting divergence curves over the sampled divergence values found in the change profiles. It works as follows. Immediately after each refresh of page $P$, the policy updates its change profiles and then computes a new *refresh period* $\phi_P$, i.e., the period of time to wait before refreshing $P$ again. The process for computing $\phi_P$ has three steps:

1. **Combine change profiles.** Combine the observations recorded in the $h$ different change profiles, into a single profile. In particular, first normalize each change profile by subtracting the base version from each recorded timestamp, so that each profile starts from $t = 0$. Then take the union of the $(t,\ D)$ pairs across all profiles, while combining entries that have the same $t$ value by taking the average $D$ value.
2. **Fit divergence curve.** Fit a continuous divergence curve to the set of points in the combined change profile, based on our generative model of Section 3.3. In particular, fit one curve that corresponds to churn behavior, and a second curve for scroll behavior, and select the one that yields the closest fit.
3. **Compute refresh period.** Using Equation 4, set the refresh period $\phi_P$ equal to the $t$ value that causes $U_P(t) = T$, where $T$ is the utility threshold parameter.

To ensure that enough observations are collected at the outset, for each newly-discovered page $P$ there is an initial *learning phase* in which the maximum refresh period is bounded by $l \cdot n$ where $n$ denotes the total number of past refreshes of $P$ and $l > 0$ is a *learning parameter*. Hence initially the policy is required to perform refreshes in rapid succession, but the maximum refresh period grows linearly over time. (Other functions to control the learning phase are also possible, and may be worth studying in future work.)

## 4.3 Bound-Based Policy

Our second policy does not attempt to fit a divergence curve based on a generative model. Instead, it places conservative bounds on divergence, and uses the bounds to adjust the refresh period $\phi_P$ adaptively.

In addition to the $h$ change profiles, for each page the bound-based policy maintains a *reference time* $t_R$. The policy attempts to find the optimal time for the next refresh, $t_R + \hat{\phi}_P$, where $\hat{\phi}_P$ denotes the optimal refresh period. Once the policy believes it has reached or passed time $t_R + \hat{\phi}_P$ it resets the reference time $t_R$ to the current time, and repeats the process. Each iteration is seeded with the refresh period used in the prior iteration, so that the process converges (assuming the page behavior is stationary).

The exact policy is as follows:

1. **Determine divergence bounds.** For each change profile, determine upper and lower bounds on divergence as shown in Figure 7. The upper bound curve represents the extreme situation where the jump in divergence between two successive observations occurs immediately following the first observation. The lower bound curve represents the opposite extreme, in which the jump in divergence occurs immediately prior to the second observation. This manner of fitting divergence bounds is conservative: it assumes no model other than monotonic divergence.

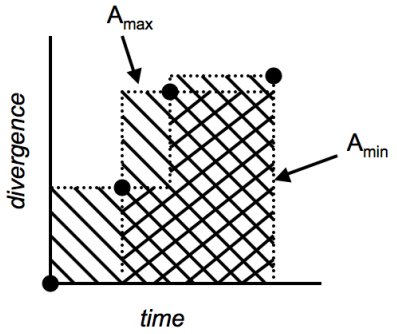


**Figure 7: A change profile, with lower and upper bounds on the area under the divergence curve.**

2. **Combine divergence bounds.** Combine the upper bounds from the $h$ change profiles into a single one by averaging. Combine the lower bounds in the same fashion.
3. **Determine utility bounds.** Compute the area under the divergence upper bound curve, in the interval $[0, t - t_R]$, where $t$ is the current time and $t_R$ is the reference time. Do the same for the divergence lower bound curve. Label these two areas $A_{max}$ and $A_{min}$, respectively, as shown in Figure 7. Then compute corresponding bounds on utility $U$ as per Equation 4, yielding $U_{min}$ and $U_{max}$ respectively.
4. **Adjust refresh period and reference time.**
   - If $U_{max} < T$, set $\phi_P := (t - t_R) \cdot 2$.
   - If $U_{min} \geq T$, reset the reference time to the present time ($t_R := t$), and set $\phi_P := \phi_P / 2$.

The rationale behind the above policy is that if the upper bound on utility is below the utility threshold $T$, the period $(t - t_R)$ was shorter than the optimal refresh period $\hat{\phi}_P$, so we should continue to explore larger refresh periods. We do so by doubling the quantity $(t - t_R)$ and scheduling the next refresh that many time units in the future. On the other hand, if the lower bound on utility is above the utility threshold, the period $(t - t_R)$ was longer than the optimal period $\hat{\phi}_P$, so we need to start over with a new reference time and explore shorter refresh periods. To ensure that we use a small enough period to start with, we use half of the current period $\phi_P$.

There is a third case in which the utility bounds straddle the utility threshold, i.e., $U_{min} < T \leq U_{max}$. In this case we take the neutral action of leaving the refresh period as it is.

## 4.4 Setting the Utility Threshold

Overall, crawling resources must be shared between discovery and retrieval of new content, and refreshing of old content [7]. Hence there is an intrinsic tradeoff between freshness and coverage. In view of this tradeoff, the following overall crawling strategy seems appropriate: when there is an opportunity to boost freshness significantly by

refreshing old content, do so; dedicate all other resources to acquiring new content.

From Section 2.2 we know that basing refresh decisions on a fixed threshold of utility, measured according to Equation 4, is optimal in terms of freshness achieved per unit cost. We leave the utility threshold $T$ as a parameter to be set by a human administrator who can judge the relative importance of freshness and coverage in the appropriate business context. $T$ is set properly iff (1) it would be preferable to receive a freshness boost of magnitude $T$ (in units of divergence $\times$ time) rather than download a new page, and (2) it would be preferable to download a new page than to receive a freshness boost of $T - \epsilon$.[6]

In a parallel crawler, the value of $T$ may be broadcast to all nodes at the outset of crawling (and during occasional global tuning). Subsequently, all refresh scheduling decisions are local, because they depend only on $T$ and a given page's change profiles.

## 4.5 Bounding Risk

Given that web servers are autonomous and pages can change arbitrarily at any time, it is important to mitigate the risk associated with waiting a long time between refreshes. Our online revisitation policies (Sections 4.2 and 4.3) aim to refresh a page whenever the estimated utility penalty of not doing so exceeds $T$, in a best-effort manner. We wish also to guarantee that in the worst case, the utility penalty incurred without performing a refresh is at most $\rho \cdot T$, where $\rho \geq 1$ is a *risk control parameter*.

Let $D_{max}$ denote the maximum divergence value allowed under the chosen page divergence metric. ($D_{max} = 1$ under our fragment staleness metric.) The maximum loss in utility incurred during $t$ time units is $t \cdot D_{max}$. To cap the utility loss between refreshes at $\rho \cdot T$, we restrict the refresh period $\phi_P$ to remain less than or equal to $\rho \cdot T / D_{max}$ at all times.

# 5. ONLINE EXPERIMENTS

We evaluate our online page revisitation policies empirically, using simulation over our web data sets described in Section 3.1. There are three sets of experiments:

- Comparison between our two policies. (Section 5.1)
- Measurement of the space footprint. (Section 5.2)
- Comparison with previous work. (Section 5.3)

## 5.1 Curve-Fitting vs. Bound-Based Policy

In our first experiment, we compare the performance of our two online revisitation policies: the one based on curve-fitting (Section 4.2) and the one based on divergence bounds (Section 4.3).

We perform separate measurements of *warm-up behavior* and *steady-state behavior*. Warm-up behavior occurs during the time following discovery of a new page, while an online policy focuses on learning the page's change patterns. Steady-state behavior follows, wherein the policy focuses primarily on exploiting its accumulated understanding of the page.

Figure 8 shows the policies' warm-up behavior—their performance in the first 30 days of our high-quality data set.

[6]Designing a meta-policy that quantifies the utility of downloading various new pages, and weighs that against the utility of refreshing, is beyond the scope of this paper.

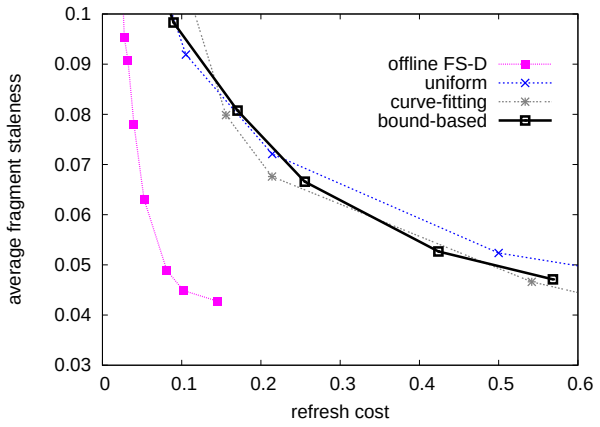


**Figure 8: Warm-up performance of online revisitation policies, on high-quality data set.**

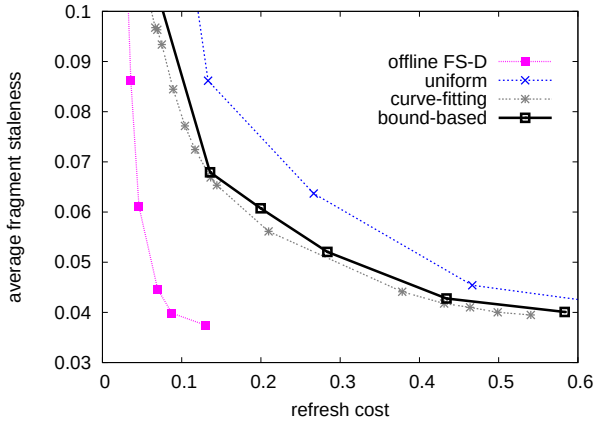


**Figure 9: Steady-state performance of online policies, on high-quality data set.**

Figure 9 shows their steady-state behavior—their performance in the final 30 days (there are 60 days total). In both graphs refresh cost is plotted on the x-axis, and average fragment staleness is plotted on the y-axis.

The different points associated with a policy correspond to different utility threshold ($T$) values. For the bound-based policy we used $h = 5$ change profiles, and risk parameter $\rho = 10$. For the curve-fitting policy we again used $h = 5$, and we tuned the learning factor $l$ such that the investment in learning is proportional to the refresh cost incurred.

For reference, in both graphs we also show the performance of uniform refreshing and of the offline FS-D policy (Section 3.4). Ideally no policy should perform worse than uniform refreshing. FS-D is a hypothetical policy that has full knowledge of the page's behavior even when it is not being refreshed. It is effectively impossible for an online policy to approach the performance of an offline one.

From Figure 8 we see that during warm-up the curve-fitting and bound-based policies perform about the same amount of exploration (reflected as refresh cost). For a given exploration cost, the staleness level during warm-up roughly coincides with uniform refreshing. Turning to Figure 9 we see that in steady-state both of our policies perform significantly better than uniform refreshing, with curve-fitting slightly ahead of bound-based.

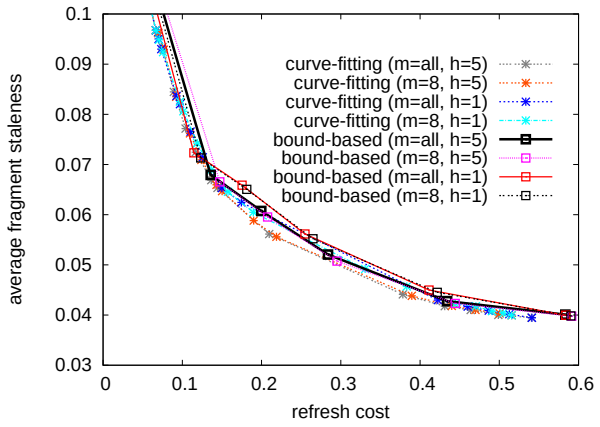


**Figure 10: Impact of space-saving techniques on performance.**

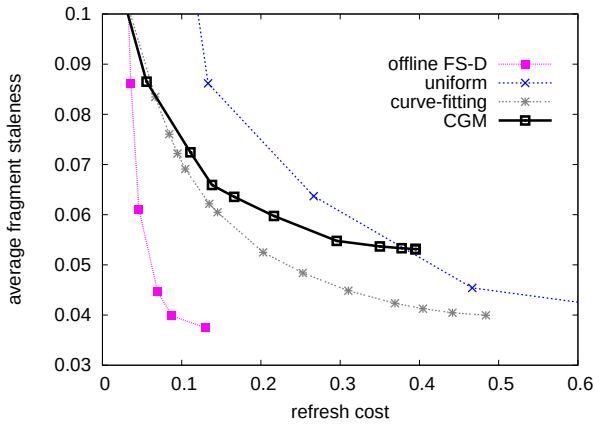


**Figure 11: Comparison with previous work, on high-quality data set.**

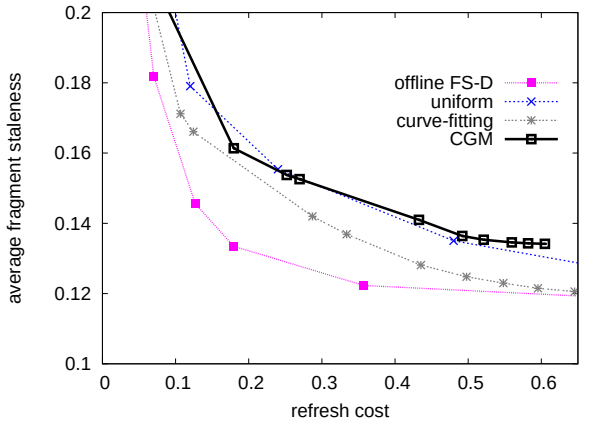


**Figure 12: Comparison with previous work, on random data set.**

## 5.2 Space Footprint

Our online revisitation policies maintain per-page state. In the presence of billions of pages, it is important to consider the total space footprint of this state.

For each page our policies maintain $h$ change profiles, each of which consists of a short sequence of numeric (time, divergence) pairs. Furthermore, for each change profile a summary of the base version must be kept, to enable computation of divergence values relative to the base version.

A page summary consists of a list of page fragments. To save space, each fragment is encoded as a hash value of the shingle string, using a collision-resistant hash function. Furthermore, following [2], rather than storing the complete list of fragments, we retain only the $m$ fragments of minimal hash value, for some constant $m > 0$. A simple unbiased estimator of the Jaccard distance based on minimal hash value sets is given in [2].

The experiments reported in Section 5.1 used $m = \infty$ (all fragments) and $h = 5$, where $h$ is the number of historical change profiles and base version summaries maintained. Figure 10 shows the performance of our curve-fitting and bound-based policies under different values of $m$ and $h$. As we can see, there is little degradation in performance due to truncating history and using compact fragment summaries to estimate divergence.

With $m = 8$ and $h = 1$, we store just one base version summary containing just 8 shingle hashes, which requires 32 bytes. In our experiments, change profile sizes ranged from 4 to 12 observations, for a space footprint of between 32 and 96 bytes. The overall space footprint is between 64 and 128 bytes per page. If there are 10 billion pages, in the worst case the total space footprint amounts to a little over a terabyte. Modern crawlers spread the per-page state across more than a thousand crawler nodes, for about a gigabyte of state per node, which easily fits in memory on each node.

## 5.3 Comparison with Previous Work

We compare our online curve-fitting policy (Section 4.2) against the policy proposed by Cho and Garcia-Molina for optimizing holistic staleness [4] coupled with a method of estimating change frequency proposed by the same authors [5], which we refer to as the CGM policy. Since that work does not propose a way of scheduling exploratory refreshes for learning purposes, we use a simple linearly-growing warm-up schedule for both CGM and our own policy. After the warm-up period we allow each policy to set its own refresh schedule. We measure post-warm-up performance.

Figure 11 shows the result of this comparison, on the high-quality data set. Almost 90% of the content fragments are static, so it is trivial to achieve staleness below 0.1. On the other extreme, 3% of the content is so dynamic that even under bi-daily refreshing it does not remain synchronized for long. Hence the "playing field" on which the policies compete spans the 0.03—0.1 staleness range.

Figure 12 shows the same comparison, but on the random data set. Observe that the minimum staleness level achievable on the random data set is much higher than on the high-quality data set. This difference is due to the fact that random pages tend to have more dynamic content than high-quality ones, perhaps aimed at attracting the attention of search engines and users.

As expected our policy, which is optimized for the fragment staleness metric, outperforms CGM in terms of fragment staleness per refresh cost, on both data sets. Although the performance difference is small in the low-cost/high-staleness part of the graphs (upper-left portion), the gap becomes substantial as we move to the moderate-cost/moderate-staleness part (middle portion) and beyond.

For example, on the high-quality data set our policy incurs refresh cost 0.2 instead of 0.4 to achieve the same moderate staleness level, which translates to refreshing pages once every 10 days instead of every 5 days, on average. If there are 1 billion high-quality pages, this difference amounts to 100 million saved page downloads per day.

CGM assumes that every time a page changes, its old content is wiped out (i.e., churn behavior), and therefore ignores all frequently changing pages. In contrast, our policy differentiates between churn and scroll behavior, and revisits scroll content often to acquire its longevous content. The only way to get CGM to visit that content is to have it visit all fast-changing content, thus incurring unnecessarily high refresh cost.

# 6. SUMMARY

We identified information longevity as a distinct web evolution characteristic, and a key factor in crawler effectiveness. Previous web evolution models do not account for the information longevity characteristics we found on the web, so we proposed a new evolution model that fits closely with actual observations.

We brought our findings to bear on the recrawl scheduling problem. We began by formulating a general theory of optimal recrawling in which the optimization objective is to maximize **correct information** $\times$ **time**. The theory led us to two simple online recrawl algorithms that target longevous content, and outperform previous approaches on real web data.

## Acknowledgements

We thank Vladimir Ofitserov for his invaluable assistance with data gathering and analysis. Also, we thank Ravi Kumar for suggesting a way to interpolate divergence values.

# 7. REFERENCES

[1] Z. Bar-Yossef, A. Z. Broder, R. Kumar, and A. Tomkins. Sic Transit Gloria Telae: Towards an Understanding of the Web's Decay. In *Proc. WWW*, 2004.

[2] A. Z. Broder, S. C. Glassman, and M. S. Manasse. Syntactic clustering of the web. In *Proc. WWW*, 1997.

[3] J. Cho and H. Garcia-Molina. The Evolution of the Web and Implications for an Incremental Crawler. In *Proc. VLDB*, 2000.

[4] J. Cho and H. Garcia-Molina. Effective Page Refresh Policies for Web Crawlers. *ACM Transactions on Database Systems*, 28(4), 2003.

[5] J. Cho and H. Garcia-Molina. Estimating frequency of change. *ACM Transcations on Internet Technology*, 3(3), 2003.

[6] E. Coffman, Z. Liu, and R. R. Weber. Optimal robot scheduling for web search engines. *Journal of Scheduling*, 1, 1998.

[7] J. Edwards, K. S. McCurley, and J. A. Tomlin. An Adaptive Model for Optimizing Performance of an Incremental Web Crawler. In *Proc. WWW*, 2001.

[8] D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener. A large-scale study of the evolution of web pages. In *Proc. WWW*, 2003.

[9] A. Ntoulas, J. Cho, and C. Olston. What's New on the Web? The Evolution of the Web from a Search Engine Perspective. In *Proc. WWW*, 2004.

[10] C. Olston and J. Widom. Best-effort cache synchronization with source cooperation. In *Proc. ACM SIGMOD*, 2002.

[11] The Open Directory Project. `http://dmoz.org`.

[12] S. Pandey and C. Olston. User-centric web crawling. In *Proc. WWW*, 2005.

[13] J. Wolf, M. Squillante, P.S.Yu, J.Sethuraman, and L. Ozsen. Optimal Crawling Strategies for Web Search Engines. In *Proc. WWW*, 2002.

# APPENDIX

# A. OPTIMAL RECRAWLING FOR HOLISTIC STALENESS

We specialize our model of optimal recrawling from Section 2.2 to the case where the divergence metric of interest is holistic staleness.

Following the model of [4], suppose each page $P$ experiences updates according to a Poisson process with rate parameter $\lambda_P$, where each update is substantial enough to cause the crawled version to become stale. Consider a particular page $P$ that was most recently refreshed at time 0. The probability that $P$ has undergone at least one update during the interval $[0, t]$ is $1 - e^{-\lambda_P \cdot t}$. Hence the expected divergence $D(P, P')$ is given by $D^*_P(t) = 1 - e^{-\lambda_P \cdot t}$. The expected utility of refreshing $P$ at time $t$, according to Equation 4, is found to be:

$$U_P(t) = \frac{1}{\lambda_P} - \left(t + \frac{1}{\lambda_P}\right) e^{-\lambda_P \cdot t}$$

The policy of refreshing a page whenever $U_P(t) \geq T$ (for some constant $T$) results in identical schedules to those resulting from the optimal policy derived in [4] for the same holistic staleness objective.

# B. INTERPOLATION OF DIVERGENCE

Our interpolation method starts by assigning to each page an *eagerness* score, based on whether the divergence curve tends to be concave (high eagerness) or convex (low eagerness). A page with a high eagerness score contains dynamic content that has rapid turnover, such as advertisements.

Given three consecutive snapshots $P(t_0)$, $P(t_1)$ and $P(t_2)$ of a page $P$, eagerness is measured as:

$$E(P, t_0) = \frac{D(P(t_0), P(t_1))}{D(P(t_0), P(t_2))}$$

The overall eagerness score $E(P) \in [0, 1]$ is found by averaging over all $(t_0, t_1, t_2)$ triples for which all three snapshots were downloaded successfully.

Now, suppose we have two consecutive snapshots $P(t_x)$ and $P(t_{x+1})$ and wish to estimate divergence inside the interval $(P(t_x), P(t_{x+1}))$, relative to some base version $P(t_B)$. We do so by taking the average of the two endpoint divergence values, weighted by eagerness. Mathematically, for any $t \in (t_x, t_{x+1})$ we assume $D(P(t_B), P(t)) = (1 - E(P)) \cdot D(P(t_B), P(t_x)) + E(P) \cdot D(P(t_B), P(t_{x+1}))$.