The Discoverability of the Web

Anirban Dasgupta Arpita Ghosh Ravi Kumar
Christopher Olston Sandeep Pandey Andrew Tomkins

Yahoo! Research, 701 First Ave, Sunnyvale, CA 94089.

{anirban, arpita, ravikumar, olston, spandey, atomkins}@yahoo-inc.com

ABSTRACT

Previous studies have highlighted the high arrival rate of new content on the web. We study the extent to which this new content can be efficiently discovered by a crawler. Our study has two parts. First, we study the inherent difficulty of the discovery problem using a maximum cover formulation, under an assumption of perfect estimates of likely sources of links to new content. Second, we relax this assumption and study a more realistic setting in which algorithms must use historical statistics to estimate which pages are most likely to yield links to new content. We recommend a simple algorithm that performs comparably to all approaches we consider.

We measure the *overhead* of discovering new content, defined as the average number of fetches required to discover one new page. We show first that with perfect foreknowledge of where to explore for links to new content, it is possible to discover 90% of all new content with under 3% overhead, and 100% of new content with 9% overhead. But actual algorithms, which do not have access to perfect foreknowledge, face a more difficult task: one quarter of new content is simply not amenable to efficient discovery. Of the remaining three quarters, 80% of new content during a given week may be discovered with 160% overhead if content is recrawled fully on a monthly basis.

Categories and Subject Descriptors

H.3.m [**Information Storage and Retrieval**]: Miscellaneous

General Terms

Algorithms, Experimentation, Measurements

Keywords

Crawling, discovery, set cover, max cover, greedy

1. INTRODUCTION

In this paper we are concerned with crawling the web in order to discover newly-arrived content. Figure 1 illustrates the key challenges of our problem. First, page p_5 may be discovered by crawling either page p_1 or page p_3, introducing a combinatorial cover problem that is NP-hard to solve exactly. Second, pages p_6 and p_7 may be discovered only by crawling new page p_4. We will study policies for recrawling known pages in order to minimize the overhead of discovering new content.

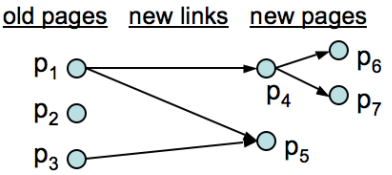

Figure 1: Old pages linking to new pages.

Our study has three goals: to characterize the arrival of new content; to provide algorithms for discovery that exploit this characterization; and to measure the overhead of discovering this content for various levels of freshness and coverage.

1.1 Motivation

Search engines today have strong freshness requirements at multiple timescales. Within minutes of breaking events, users expect to visit a search engine to gain access to news, blog posts, and other forms of content related to the event. Freshness requirements for such information ranges from minutes to hours. For less immediate content such as reviews of a new product, users are disappointed if a review exists on the web but not in the search engine index; freshness requirements here range from hours to days. And finally, obscure content that meets an information need should make its way to the index within days to weeks.

Despite these requirements, there are serious limitations on the ability of the crawler to procure new content in a timely manner. First, bandwidth remains limited, so downloading the entire web every day is not practical. But more importantly, requests per second to an individual website is also limited by politeness rules. Many sites are so large that they cannot be crawled from start to finish within a week under standard politeness assumptions. And many sites report crawler traffic as a significant fraction of total traffic, including multiple downloads of unchanged pages. Thus, careful management of the limited accesses available to a crawler is now mandatory.

Additionally, all crawlers must trade off recrawl of existing pages against first crawls of unseen pages; an understanding of the discoverability of new content allows an understanding of the diminishing returns of increasingly aggressive discovery policies.

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

Finally, an analysis of the discoverability of the web exposes an evolutionary property of the graph that is not well understood, namely, the mechanism by which new pages are "linked in" to the graph by modifications to old pages. Our lack of understanding of these matters raises concerns about the effectiveness of graph generators and even the effectiveness of the crawling model as an approach to timely discovery of new content going forward.

1.2 Problem discussion

There are two primary mechanisms by which new pages arrive on the web. First, a website puts up a new page, and links to this new page from an existing page. Second, an entirely new website appears and is linked-to from an existing website. A third possible mechanism is that one website puts up a new page without linking to it, and another website provides a link to the new page — this situation is very uncommon in our data, and we do not study it.

The relative fractions of pages appearing as a result of these two mechanisms depends on the elapsed time between observations. As this window shrinks, we will discover new sites at an earlier stage of their growth, and hence an increasing fraction of pages will appear as new pages on old sites. Even when the window is a full month, however, we show that 85–95% of new pages appear on existing sites, suggesting that the problem of analyzing known sites is of paramount importance. We therefore study this problem in greater detail.

Before proceeding, we must observe that no web crawler may actually crawl the entire reachable web. Due to infinite websites, spider traps, spam, and other exigencies of the real web, crawlers instead apply a crawl policy to determine when the crawling of a site should be deemed sufficient. Some sites are crawled exhaustively, while others are crawled only partially. In this paper, we focus only on sites that are to be crawled exhaustively, as the remaining sites have been deemed lower priority in terms of absolute coverage.

Suppose the crawler has performed an initial complete crawl of some site at time t. Now imagine that at time $t+\Delta$ the crawler must revisit the site and find all the new pages. If it is the case that a small set of old pages collectively links to all new pages, then the crawler can in principle discover new pages with minimum overhead. For example, in Figure 1, recrawling just page p_1 leads to discovery of all new pages.

How well this idea can work on the real web is the subject of this paper. The fundamental questions are as follows (this paper tackles several of these, as indicated by the forward pointers below; the remainder are left as future work):

Basic feasibility of the approach:

- Is it the case for real websites that most new pages can be discovered via a small set of old pages? *(Section 4)*

Key characteristics that determine what crawling approaches are likely to work well:

- To what extent are links to new content redundant (as in $p_1 \to p_5$ and $p_3 \to p_5$ in Figure 1)? *(Section 4)*
- Does the set of old pages that link to many new pages tend to remain consistent over time?

Efficient crawl policies for content discovery:

- What is a good choice of old pages to seed the discovery process, given historical information and a crawl budget? *(Section 5)*
- What fraction of the budget should be spent assessing the usefulness of various old pages, versus exploiting ones already known to be somewhat useful?

Our key findings are as follows. We show first that with perfect foreknowledge of where to explore for links to new content, it is possible to discover 90% of all new content with under 3% overhead, and 100% of new content with 9% overhead. But actual algorithms, which do not have access to perfect foreknowledge, face a more difficult task: one quarter of new content is simply not amenable to efficient discovery. Of the remaining three quarters, 80% of new content during a given week may be discovered with 160% overhead if content is recrawled fully on a monthly basis.

1.3 Related work

Numerous early web studies focused on properties of a snapshot of the web graph [2, 4, 12, 17, 18]. More recently, attention has turned to evolutionary properties of the corpus. In this evolutionary model, researchers have considered the growth of the web [3], the rates of page and link churn [8, 14, 19], the rates of duplicate evolution [13], and the change rates of individual pages [3, 5, 14, 22].

Parallel to this line of work, there has been a significant body of work on refreshing already-discovered content, which has been studied in [6, 9, 10, 21, 25]. Already-discovered pages are recrawled to keep the search engine local repository fresh so that the search queries are not answered incorrectly due to stale information, while the discovery of new pages is important for ensuring that as many relevant query results are shown as possible. It is tempting to view our problem as equivalent, with new outlinks taking the role of new content on existing pages, but there is a critical distinction: in our problem, many pages can be recrawled, each of which points to a new page, but the value depends on the union rather than the sum. If the pages all point to the same new content, there is very little value from a discoverability standpoint, but great value from the standpoint of the freshness of the recrawled pages. To our knowledge, this specific problem has not been studied previously.

Finally, there has been work in ordering the frontier of a crawl [7, 11], in which various policies are studied from the perspective of estimating the quality of a candidate for first-time crawl. This work is orthogonal to ours; once new pages have been discovered, it remains to prioritize them for crawling.

2. PRELIMINARIES

2.1 Formulation

A snapshot G_t of a given site at time t is a directed graph (V_t, E_t), where V_t is the set of nodes (pages) and E_t is the set of directed edges (hyperlinks). Define $X_t \triangleq \cup_{j=1}^{t-1} V_j$ to be the set of *old nodes* at time t, and define $Y_t \triangleq V_t \setminus X_t$ to be the set of *new nodes* at time t. The old nodes X_t are nodes that appeared before time t and the new nodes Y_t are nodes that appeared first at time t.

For convenience, we use the following representation for the old and new nodes at any time t. Let $H_t = (X_t, Y_t, Z_t)$ be a bipartite graph consisting of the old nodes X_t, the new nodes Y_t, and an edge set Z_t. This graph will reflect information relevant to our discovery problem, but will not reflect all information in the original graph. An edge $z = (x, y)$ exists whenever $y \in Y_t$ is *efficiently discoverable* from $x \in X_t$, i.e., there is a path from x to y of the form $x \to y_1 \to y_2 \to \cdots \to y_k = y$ where each $y_i \in Y_t$ is a new node. In this case we say that each y_i is *covered* by x_0. Figure 1 shows the motivation for this definition: by crawling a node that reveals the start of a long chain of new nodes, we may now proceed to download the entire chain of new content recursively with no additional discovery overhead (as each node of the chain is new, and hence must be crawled anyway).

The problem of discovering new content is then the following: cover the new nodes in Y_t using as few nodes from X_t as possible. Combinatorially speaking, there are two natural (unweighted) versions of this problem. The first is called the *k-budgeted cover* problem, where we are given a budget k, and want to cover as many nodes in Y_t as possible using k nodes from X_t. The second is called the *ρ-partial cover* problem, where we are given $\rho \leq 1$ and the goal is to cover at least $\rho|Y_t|$ nodes in Y_t using as few nodes from X_t as possible. Both problems are NP-hard [24].

We study different algorithms for these problems based on the amount of information that is available at the time when a node must be crawled. First, in Section 2.2 we describe an algorithm called GREEDY, which has complete information about H_t; this algorithm should be viewed as an upper bound on the performance of any realistic algorithm.[1] Next, in Section 5 we describe a family of algorithms that use information that is realistically available at the time when a node must be crawled. In particular, they do not have access to H_t, but depending on the model, they may have partial information about X_t and statistical information about H_t based on partial information about $H_{t'}$ for $t' < t$.

Note that we have not addressed the issue of nodes disappearing/dying between crawls. Our algorithms may be adapted in a straightforward manner to this case, but we focus in this paper on the basic case in which nodes do not disappear.

Notation. For each $x \in X_t$, we denote by $N(x)$ the set of new nodes efficiently discoverable from x, i.e., $N(x) = \{y \mid (x, y) \in Z_t\}$. For a subset S of X_t, we define $N(S) = \cup_{x \in S} N(x)$.

The *Jaccard coefficient* between two nodes x and y is

$$J_{xy} = \frac{|N(x) \cap N(y)|}{|N(x) \cup N(y)|}.$$

A Jaccard coefficient close to 1 means that x and y point to a very similar set of nodes, and a value close to 0 means that they are almost non-overlapping.

Key metric: overhead. In general, if a set O of old pages are crawled to discover $|N(O)|$ new pages, then we define the *overhead* of O as $|O|/|N(O)|$. Overhead numbers should be read as follows: if 100 new pages may be captured by a cover of size five, then an algorithm must perform five "wasted" fetches, in the sense that they do not return new pages, in order to generate enough information to fetch the 100 new pages. The overhead is 5%, and is a direct measure of the fraction of additional fetches necessary to gather a given number of new pages, in other words, a measure of efficiency.

[1] This algorithm is not strictly speaking an upper bound, as it makes approximations in order to solve an NP-hard problem; however, the information available to the algorithm allows it to perform substantially better than any realistic algorithm we have seen.

2.2 An algorithmic upper bound: GREEDY

While the maximization problem of k-budgeted cover admits a $(1 - 1/e)$-approximation algorithm, the minimization problem of ρ-partial cover can only be approximated to within a $\log |X_t|$ factor [16, 23]. Coincidentally, the same greedy algorithm can be used for both problems. For completeness, we present the greedy algorithm below. In words, the algorithm proceeds by repeatedly returning the old node that covers the most uncovered new nodes.

Algorithm GREEDY (X_t, Y_t, Z_t)

Set $C_t = \emptyset$.
While "not done" do,
  Find $x \in X_t \setminus C_t$ that maximizes $|N(x) \setminus N(C_t)|$;
    break ties arbitrarily.
  Set $C_t = C_t \cup \{x\}$.
Return C_t.

For the k-budgeted cover problem, the predicate "not done" is true as long as $|C_t| \leq k$. For the ρ-partial cover problem, this predicate is true as long as $|N(C_t)| < \rho|Y_t|$.

3. DATA

We consider two datasets, to address two distinct problems within our scope. First, we consider a sequence of complete crawls of a number of websites. This dataset allows us to study in detail the process by which new pages on a site are incorporated into the existing graph of the site. Second, we consider a sequence of complete crawls of the Chilean web. This dataset by contrast allows us to study inter-site linking, and particularly, the problem of discovering entirely new websites. We describe these two datasets below.

Site recrawl dataset. We consider a repeated crawl of 200 web sites over a period of many weeks. This dataset was used in earlier work by Ntoulas, Cho, and Olston; see [20] for more details about the crawl and the principles used to select the web sites. The authors of that work have continued to collect data, and have generously allowed us to employ more recent snapshots than those in their reported results.

Of the 200 web sites they crawl, we removed those sites that contained fewer than 100 pages in any snapshot (i.e., the site did not have significant size) or more than 200,000 pages (which was a crawler-imposed upper bound on the number of pages per site, introducing skew into the analysis of new pages). This resulted in 77 sites. Of these sites, we selected 42 that were well-represented at each snapshot, and that did not show any gross anomalies.

The 42 websites in the results dataset were crawled repeatedly over a period of 23 weeks from 11/14/2004 to 6/12/2005 (the crawler did not execute during every week). The total number of pages at the first timestep was 640,489 and 223,435 new pages appeared over this period, of which about 40% are directly linked to some old page.

For each of the web sites and for each snapshot, we first parsed the crawl output, and extracted the outlinks and redirect information. We omitted all off-site links and focused only on on-site links. We also discarded orphans — pages in Y_t that are not covered by any page in X_t. Orphans accounted for less than 5% of the new pages in our dataset. We then constructed the bipartite graph H_t defined above for the purposes of analysis; recall that this step involves examining paths from old pages to new pages.

Chilean web dataset. We employ a new data set to study this problem, based on the Chilean web. We have three snapshots of the Chilean web, based on complete crawls performed monthly for three months; the first snapshot had 7.40M pages and 67.50M edges and the third snapshot had 7.43M pages and 70.66M edges.

4. MEASUREMENTS

In this section we present a series of measurements on both of our datasets. In addition to basic properties of the data, we will study in detail the extent to which algorithm GREEDY is able to efficiently cover the new content.

We will begin with a series of experiments on the site recrawl dataset, studying the discovery of new pages on existing sites. Based on the results of this analysis, we will then turn in Section 4.4 to an analysis of the Chilean dataset, in which we will study the relative prominence of new pages on existing sites, versus new pages on new sites.

4.1 Cover size

For each site at each time, we construct the bipartite graph H and employ GREEDY to cover all new pages. Figure 2 plots a point for each site, each timestep, and each partial cover (i.e., for a cover of size 10, we show a point for the first node of the cover, the first two nodes, and so forth—each successive partial cover captures more nodes with higher overhead than the previous partial cover). A point at location (x, y) represents a cover of size x that covers y new pages. The figure represents approximately 400 trajectories, one for each site at each timestep, but many of these are overlapping; the lower graph of the figure shows a breakout of the smaller trajectories at a larger scale.[2]

The graph clearly shows the diminishing returns as each cover grows. Further, an examination of the knee of the curves shows that most covers efficiently capture 90% of the total new pages, but must work much harder to cover the remaining 10%.

We present a detailed aggregation of these numbers in Figure 3(a-b). We perform an experiment in which we employ GREEDY for a particular site at a particular time, but terminate processing when either all new pages have been covered, or the current cover has reached a certain size k; this corresponds to the k-budgeted cover problem. In Figure 3(a-b), the x-axis represents the threshold k that is the maximum size cover we will employ for any site/time pair.

[2]The outlier trajectory in the top graph of Figure 2 is from the site `oreilly.com`. It came about when a content management change caused over 2,000 catalog entries to introduce a link to a new variant of the same page; the new destination was discoverable from no other page on the site. Thus, the limit of the anomalous trajectory is a line of slope 1, in which each recrawl event yields a single page of new content.

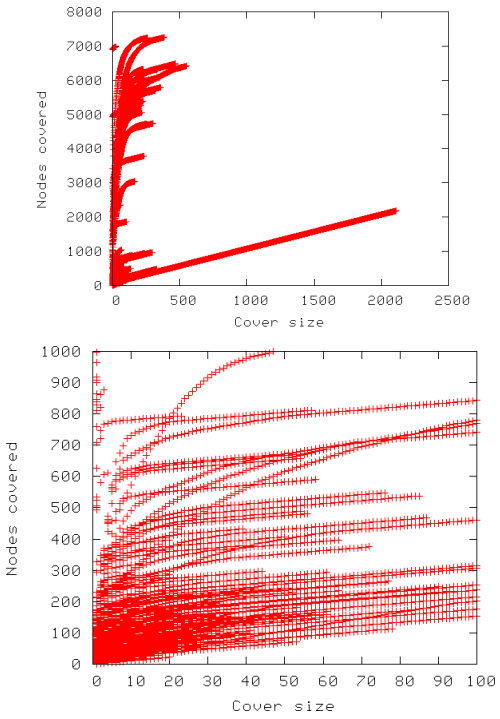

Figure 2: Cover size versus the number of pages covered.

Figure 3(a) shows two curves. The higher curve is measured on the left axis; it shows for each value of k the average number of new pages captured by the cover. However, notice that for a fixed value of k, each site/time pair might have a cover of k or smaller, depending on whether a smaller cover was adequate to capture all the new pages. We therefore also include the lower curve, which is measured on the right axis. It shows for each value of k the overhead of the cover. As k grows large, the number of pages covered tops out at about 300 on average, which is a reflection of our dataset. However, the overhead never exceeds 9%, indicating that although the rightmost region of the curve returns 300 new pages per cover, with $k = 600$, nonetheless the "average" cover size is in fact only 9% of 300, or about 27.

We mention in passing that, while the x-axis of the figure has been truncated at 600 to focus on the region of interest, the remainder of both curves are stable at 300 and 9% respectively.

Figure 3(a) is therefore a measure of how efficiently covers truncated at a certain size can return new content, but so far we have said nothing about what fraction of the total new content has been returned. Figure 3(b) covers this question. Once again, the x-axis represents the threshold k on the cover size, and the y-axis now shows the overall fraction of new pages that would be covered, if all covers were truncated at size k. Setting $k = 200$, we cover 97.3% of all new content. We cover 90% of new content once k reaches 83.

4.1.1 90% covers

Based on the above observations, it appears possible to cover 90% of new content with relatively low overhead. We therefore adopt this somewhat arbitrary threshold, and study

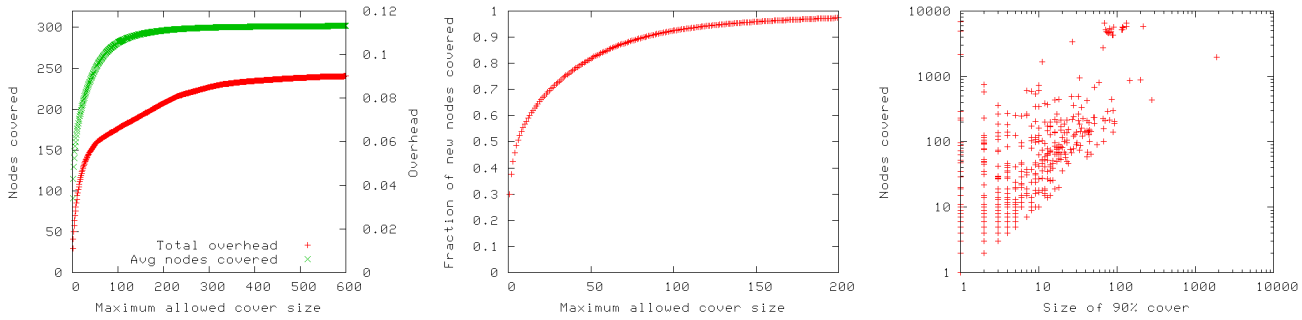

Figure 3: (a) Overhead and number of covered pages, (b) fraction of new pages covered, (c) 90% cover statistics.

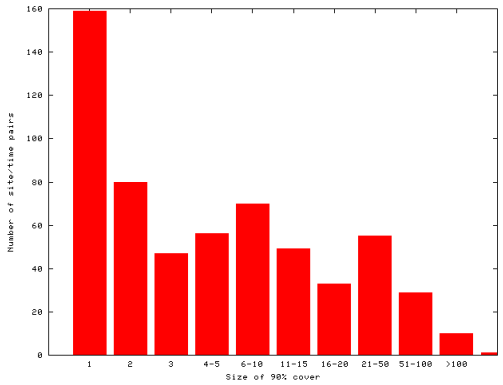

Figure 4: Histogram of 90% cover sizes.

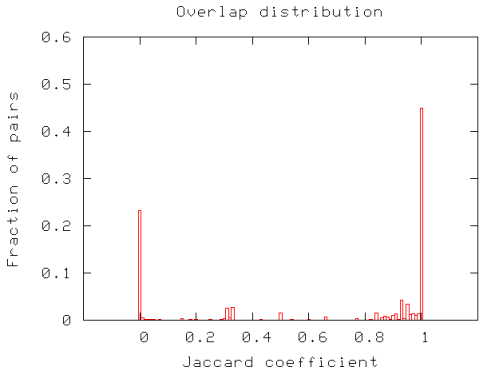

Figure 5: Overlap distribution.

the nature of covers that capture at least 90% of the new pages for a give site/time pair. Figure 3(c) is a scatter plot showing a detailed breakout of this information. A point at (x, y) means that a particular site at a particular time had a cover of size x that covered y new pages.

As algorithm Greedy adds a page only when it results in the discovery of at least one page of new content, there are no points below the line $y = x$. The figure is promising to the extent that there is a significant mass of points far from the line. Note that the figure is shown in log-log scale, and there are clearly numerous points in which a small cover produces a large number of new pages.

We may ask about the distribution of sizes of these 90% covers. Figure 4 shows this distribution as a histogram, showing the number of site/time pairs for which the 90% cover has a certain absolute size. Small covers of five or fewer pages suffice to capture 90% of the new content of most sites, but for a nontrivial number of sites, covers of more than a hundred pages are required. No site in our sample ever required a 90% cover of more than one thousand pages.

4.2 Node redundancy

If no two old pages link to the same new page, then the cover problems addressed by Algorithm Greedy become trivial; the problem is interesting only when there is overlap in the set of new pages covered by old pages. In our data, most pairs of pages (within a site) fall into one of two categories: either they link to almost the same set of new pages, or they have almost no new pages in common. Figure 5 shows that a significant fraction of pairs have Jaccard coefficient very close to 0 or very close to 1. This has important algorithmic implications, as we will see later in Section 5.2.

4.3 Overhead of discovering new pages

Figure 6 shows the overhead for various cover sizes. As the figure shows, and as stated above, we attain 90% covers with 3% overhead, and 100% covers with 9% overhead.

Recall, however, that these numbers are the results of a thought experiment in which a crawler happens to pick a near-perfect set of pages to crawl in order to find new content; they represent a goal we would like to attain. The reader should be heartened that the numbers look so promising, but should await Section 5 to determine whether these numbers can be matched by a real algorithm that must search for new content in a more hit-or-miss fashion.

4.4 Overhead of discovering new sites

In this section we study the relative importance of discovering new pages on old sites, versus new pages on new sites. We have presented statistics showing the performance of Algorithm Greedy on each individual site, aggregated in various ways. We now ask how Greedy might perform across all sites at once, by operating on the union of the bipartite graphs H_t corresponding to each individual site.

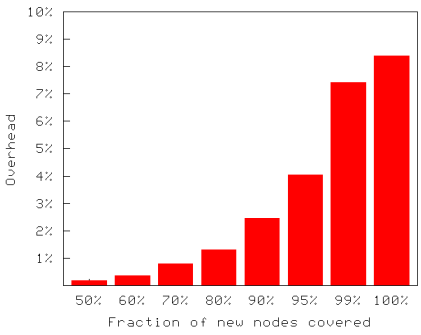

Figure 6: Global discovery of new pages on old sites.

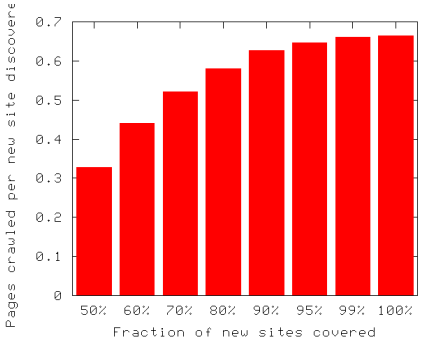

Figure 7: Chile site-level discovery.

Snapshot	new pages on new sites	new pages on old sites	Pr[new site | new page]
$1 \rightarrow 2$	452,461	2,404,045	16%
$2 \rightarrow 3$	173,537	2,272,799	7%

Table 1: Fraction of new pages appears on new sites versus old sites in the Chilean web data set.

When asked for a 90% cover, such an algorithm may cover 90% of each site, or may cover many sites completely while covering others only superficially, based on the relative gains of each crawl event. We observe in passing that such an algorithm is simple to implement once Algorithm Greedy has already run on each site: a greedy cover of disjoint sets may be constructed simply by interleaving the greedy covers of each set, in a greedy manner. That is, each site may be viewed as a set of candidate pages, ordered according to the greedy cover for that site. The algorithm must choose the top remaining page from some site, and it does so by selecting the one that covers the largest number of uncovered resources.

We perform the following experiment on the three snapshots of the Chilean web. We begin by observing that a site that appears for the first time during the second or third month contains on average 18.1 pages. Thus, the effort of discovering such a site may be amortized across the 18+ pages that will be returned by crawling the site. Table 1 considers each page that occurred for the first time in the second or third month of the crawl, and checks to see whether the domain of the page occurred earlier. As the results show, 16% of new pages in the second snapshot, and 7% of pages in the third snapshot, occur on sites that did not appear during the previous snapshot. This suggests that the vast majority of new content appears on existing sites, rather than new sites.

Figure 7 shows the number of existing pages that must be crawled in order to discover new web sites. Comparing Figures 6 and 7, we see that many more pages must be crawled to discover new sites than to discover new pages on existing sites. (The main reason the latter problem is easier is the propensity of webmasters to include useful pages guiding us to new information, e.g., the "What's new" page.) In the new site discovery problem, depending on the fraction of new sites that must be covered, each page fetch will yield between 1.5 and 3 new sites. However, as we observed above, each of these sites will return on average 18.1 pages, resulting in an overall overhead of just 3.7%, even for 100% coverage. These results, combined with the observation from Table 1 that most new pages occur on old sites, convince us to focus the remainder of our exploration on existing sites.

5. HISTORY-BASED ALGORITHMS

In the previous section we studied the feasibility of using a small set of existing nodes to cover most of newly generated content — i.e., we measured whether there exists a small set of old nodes with links to most of the new content. In this section we move to the algorithmic question of choosing such a set of nodes when we do not have access to the entire bipartite graph H_t. We assume that we have access to the old nodes X_t but not to Z_t, the set of edges, or to Y_t, the set of new nodes. (In reality, we may only have access to a subset of X_t since some nodes in X_t may not have been discovered at t due to incomplete crawling before t. We ignore this for now.)

In this section we explore algorithms that use historical information, i.e., statistics from H_{t-i}, in order to discover new content in H_t. There are two separate questions: how to aggregate information from the various H_{t-i} to estimate relevant statistics, and second and more open-ended, which statistics lead to good covers?

To address this, we describe and evaluate three algorithms that employ different statistics gathered from past observations to solve the k-budgeted cover problem. The first algorithm, Od, crawls pages according to the number of new pages discovered historically when crawling the page. The second algorithm Cliq employs past degree information as well, and in addition uses information about overlaps in the set of pages discovered by each pair of pages. Rather than computing and storing all pairwise information between existing pages, Cliq groups existing pages into clusters that have produced the same set of pages in the past, according to the gap observation of Section 4.2, and employs this information in order to choose a cover. The third algorithm Cov uses historical results of the algorithm Greedy, i.e. it chooses to track pages that were previously in the cover constructed from full recrawls of the data.

In what follows, we define $S^\star$ be the optimal solution to the k-budgeted cover problem on H_t (Section 2). Let S be the solution returned by an algorithm Alg. We define $\rho(\text{Alg})$ as the ratio of the number of new nodes covered by S to that covered by $S^\star$, i.e., $\rho(\text{Alg}) = N(S)/N(S^\star)$. We use N to denote the total number of new nodes.

5.1 Algorithm based on outdegree

We consider a very basic algorithm first. Suppose that for every old node i, we have an estimate of $p_i = |N(i)|/N$, the fraction of new nodes covered by i. A natural algorithm is the following: pick k old nodes with the largest p_i's and crawl these nodes. We refer to this algorithm as OD. Below, we state a bound on its performance, if the p_i's are correct estimates. Subsequently, we will define variants of this algorithm that are amenable to experimentation, based on different approaches to estimating the p_i values.

LEMMA 1. *Let $p_{[j]}$ denote the j-th largest of the p_i's. Then, $\rho(\text{OD}) \geq \frac{p_{[1]}}{p_{[1]}+\sum_{i=k+1}^{2k-1} p_{[i]}}$.*

PROOF. Suppose there are N_1 new nodes obtained from nodes with degrees $p_{[2]}, ..., p_{[k]}$ that are distinct from the new nodes obtained from $p_{[1]}$. The number of new nodes found by the greedy algorithm is $Np_{[1]} + N_1$. The number of new nodes found by the optimum cannot be greater than $Np_{[1]} + N_1 + N\sum_{i=k+1}^{2k} p_{[i]}$ (recall that $p_{[i]}$ are decreasing). So

$$\rho(\text{OD}) \geq \frac{Np_{[1]} + N_1}{Np_{[1]} + N_1 + N\sum_{i=k+1}^{2k-1} p_{[i]}} \geq \frac{p_{[1]}}{p_{[1]} + \sum_{i=k+1}^{2k-1} p_{[i]}}.$$

□

The above bound is tight. If the degree distribution of nodes in X_t is a power law, the bound shows that this naive algorithm will perform very well. However the presence of mirrors can cause this fraction to be as small as $1/k$. This, together with the observations in Section 4.2 lead to the next algorithm.

5.2 Algorithms based on overlap

Here we describe an algorithm for choosing a small cover that exploits estimated overlap information. Let p_i be as above, and for a pair of old nodes i, j, let p_{ij} be the fraction of new nodes that i and j both cover: $p_{ij} = |N(i) \cap N(j)|/N$. Figure 5 empirically demonstrated that most nodes overlap in either a very large or a very small set of links. We state a lemma showing that under an idealized form of the observation, it is possible to uniquely partition nodes into groups that all link to almost the same set of new nodes. Then,

LEMMA 2. *Let $\epsilon_b, \epsilon_s \leq 1/3$. If for all nodes i, j, we have either $J_{ij} \geq 1 - \epsilon_b$ or $J_{ij} \leq \epsilon_s$, then the set of old nodes X_t can be partitioned into equivalence classes, where every pair of old nodes i, j in an equivalence class has Jaccard coefficient $J_{ij} \geq (1 - \epsilon_b)$.*

PROOF. We will show that for such ϵ, if $J_{ij} \geq 1 - \epsilon_b$, $J_{jk} \geq 1 - \epsilon_b$, then J_{ik} cannot be less than ϵ_s. From the assumptions, $|N(i) \setminus N(j)| \leq \epsilon_b |N(i) \cup N(j)|$, and similarly $|N(k) \setminus N(j)| \leq \epsilon_b |N(k) \cup N(j)|$. So the most number of elements not in common between i and k is $\epsilon_b(|N(i) \cup N(j)| + |N(j) \cup N(k)|)$, i.e.,

$$\begin{aligned}
|N(i) \cap N(k)| &\geq |N(i) \cup N(k)| - \\
&\quad \epsilon_b(|N(i) \cup N(j)| + |N(j) \cup N(k)|) \\
\Rightarrow \quad J_{ik} &\geq 1 - \epsilon_b \frac{(|N(i) \cup N(j)| + |N(j) \cup N(k)|}{|N(i) \cup N(k)|} \\
&\geq 1 - \epsilon_b \left(\frac{|N(i) \cup N(j)|}{|N(i)|} + \frac{|N(k) \cup N(j)|}{|N(k)|} \right) \\
&\geq 1 - \epsilon_b \left(\frac{1}{1-\epsilon_b} + \frac{1}{1-\epsilon_b} \right),
\end{aligned}$$

that is strictly greater than ϵ_s for $\epsilon_b, \epsilon_s \leq 1/3$. The last line follows from $|N(i)| \geq |N(i) \cap N(j)| \geq (1 - \epsilon_b)|N(i) \cup N(j)|$, and similarly for k. In summary, we showed that $J_{ij} \geq (1 - \epsilon_b)$, $J_{jk} \geq (1 - \epsilon_b) \Rightarrow J_{ik} > \epsilon_s$ for $\epsilon_b, \epsilon_s \leq 1/3$. Recall that $J_{\cdot,\cdot}$ is a metric. By our assumption, J_{ik} is either greater equal $(1 - \epsilon_b)$ or less equal ϵ_s, so we have shown that $J_{ik} \geq (1 - \epsilon_b)$, i.e., old nodes can be partitioned into equivalence classes. □

We analyze the performance of the following algorithm, CLIQ. Let $C_1, \ldots, C_\ell$ be the equivalence classes as above and let $k' = \min(k, \ell)$. Let $q_i = \max_{j \in C_i} p_j$ be the degree of the highest-degree node in i-th equivalence class and let n_i be the node with this degree. We first sort $C_1, \ldots, C_\ell$ in order of descending p_i's. The output S of the algorithm is the set of n_i's corresponding to the k' largest q_i's.

THEOREM 1. $\rho(\text{CLIQ}) \geq \frac{1-k'\epsilon_s}{1+k\epsilon_b}$.

PROOF. First we lower bound the number of new nodes obtained by CLIQ. Denote by T_j the number of new nodes obtained by adding j to S. From n_1 we get $T_1 = Nq_1$ new nodes. Define $q_{ij} = p_{n_i n_j} = |N(n_i) \cap N(n_j)|/N$. From the j-th node added by CLIQ, the number of new nodes obtained is $T_j \geq Nq_j - \sum_{i=1}^{j-1} Nq_{ij}$. Since n_i and n_j belong in different classes, $J_{n_i n_j} \leq \epsilon_s$, so

$$\begin{aligned}
q_{ij} &\leq \frac{J_{n_i n_j} |N(n_i) \cup N(n_j)|}{N} \\
&\leq \frac{\epsilon_s(|N(n_i)| + |N(n_j)|)}{N} \\
&= \epsilon_s(q_i + q_j).
\end{aligned}$$

Substituting above, $T_j \geq Nq_j - N\epsilon_s \sum_{i=1}^{j-1}(q_i + q_j)$. Summing over all j,

$$\begin{aligned}
\sum_{i=1}^{k'} T_i &\geq \sum_{i=1}^{k'} \left(Nq_i - \sum_{j<i} N\epsilon_s(q_i + q_j) \right) \\
&\geq \sum_{i=1}^{k'} Nq_i(1 - k'\epsilon_s).
\end{aligned}$$

Now we upper bound the number of new nodes covered by the optimum. The optimum cannot choose more than k nodes from a class C_i, and so it cannot get more than $(1 + k\epsilon_b)q_i$ new nodes from C_i: every new node added after n_i contributes no more than $\epsilon N q_i$ new nodes to $N(C_i)$. Since the cliques are ranked in order of decreasing degree, the q_i's of the k' cliques chosen by the optimum are upper bounded by the k' highest q_is (chosen by CLIQ), and so optimum is upper bounded by $(1 + k\epsilon)N \sum_{i=1}^{k'} q_i$. So $\rho(\text{CLIQ}) \geq (1 - k'\epsilon_s)/(1 + k\epsilon_b)$. □

In reality, not all pairs of old nodes may satisfy the condition in Lemma 2 with sufficiently small values of ϵ_b, ϵ_s, in which case we do not obtain the equivalence classes in Lemma 2. We use a modified version of the algorithm, in which we first group the old nodes into clusters recursively as follows. We choose a value for the parameter ϵ_b, and initialize with every node in its own cluster. We merge the clusters so that an old node i belongs to a cluster C if $\max_{j \in C} J_{ij} \geq 1 - \epsilon_b$, i.e., it has high overlap with any other node in the cluster. (Note that this partitioning into clusters is well-defined.) We then run CLIQ using these clusters instead of equivalence classes.

5.3 Algorithm based on greedy cover

Finally, we describe an algorithm COV that exploits previously observed cover information. Let S be the set of old nodes returned by the GREEDY algorithm for the k-budgeted cover on $H_{t'}$ where t' is the index of the most recent complete recrawl. The algorithm COV uses this set S of size k as the cover till the next recrawl. Note that this algorithm has the following disadvantages over CLIQ : a cover cannot be defined unless the site is completely crawled, whereas pairwise overlap information can still be gathered from partial recrawls. Also, it is not easy to 'average' cover information from multiple recrawls but overlap information can be averaged across recrawls.

5.4 Aggregating past observations

We now define several variants of OD and CLIQ in which information from multiple historical recrawls is aggregated to determine future behavior of discovery crawls. For concreteness, we assume the site is fully crawled every Δ weeks, and our goal is to discover new content in between these periodic full recrawls.

For fixed Δ, we may estimate the degree statistics p_i using exponential weighting with parameter α:

$$p_i^t = \left(\sum_{t'} \alpha^{t-t'} p_i^{t'} \right) / \left(\sum_{t'} \alpha^{t-t'} \right),$$

where t' ranges over the time indices when a full recrawl was performed. We refer to OD with this method of estimating p_i as OD-WIN. We define OD-ALL as the particular instance of OD-WIN with recrawl frequency $\Delta = 1$; this algorithm must discover new content using complete information about all prior weeks. Similarly, for any Δ we define OD-1 as the algorithm that estimates p_i^t based on the most recent recrawl, consulting no further historical information.

To approximate the statistics for CLIQ, we do the following. To the set of all clusters from the most recent recrawl, we add one cluster for every old node in X_t that ever linked to a new node in any past recrawl. The q_i for these singleton clusters is the estimate p_i^t as computed above. We apply CLIQ to this set of clusters with the corresponding parameters. We will refer to this algorithm as CLIQ-WIN. As above, we refer to the version of the algorithm with p_i^t measured from the most recent recrawl as CLIQ-1.

5.5 Upper bounds on performance of historical algorithms

We begin by constructing an upper bound as follows. We implement the policy of crawling at time t every page that historically yielded any link to a new page at time $t-1$ or before. Any new page that cannot be discovered by this technique will be very difficult to find; in fact, it is hard to imagine finding such pages without simply exploring the entire site. The result of this experiment is that we discover only 74% of new content, suggesting that roughly a quarter of new content is simply not amenable to efficient discovery.

We then perform an experiment to explore the decay in discovery as we use increasingly remote information, as follows. We imagine a periodic full recrawl of a site every w timesteps, and at each week we make use only of pages that linked to a new page during some past periodic recrawl; thus, if $w = 4$ we make use of information that is one, two or three timesteps old. The following table shows the results.

Recrawl policy	Full	Periodic, $w = 2$	Periodic, $w = 4$
New pages	74%	64%	59%

Thus, it is theoretically possible to discover 74% of new pages with an amount of overhead lower than crawling the entire web, but as the freshness of our information decays, the fraction of new content we can realistically expect to discover also drops. In the following section we will study how close to these upper bounds our algorithms come, as a function of the amount of effort expended.

5.6 Analysis of historical algorithms

Some care is required in our evaluation methodology for this section. We compare a number of algorithms that may have access to differing amounts of historical information, and hence differing numbers of candidate pages to recrawl. Thus, we may see an algorithm that performs very well when asked to produce a cover of 80%, but that is unable to produce a cover of 90%. We adopt the following methodology to allow a meaningful comparison of such policies.

We fix a budget k, which is the maximum number of recrawls that may be performed at a particular site. We evaluate each algorithm at each time, and ask it to cover as large a set of new pages as possible, using no more than k old pages. We then measure for each algorithm the average cover size produced (which may be less than k), the average overhead, and the average coverage (measured as total number of covered pages on all sites at all timesteps divided by total number of new pages on all sites and all time steps). We repeat these measurements for all values of k, so that we can for instance compare covers of a particular average depth, or a particular level of coverage.

We performed an experiment to compare all our historical algorithms against an approximation to optimal, in the form of Algorithm GREEDY. For all versions of CLIQ, we used $\epsilon_b = 0.8$. We evaluated various values for the exponential decay parameter α, and found that $\alpha = 0.8$ and $\alpha = 1$ perform well. We adopt $\alpha = 1$ henceforth.

The results are shown in Table 2. Here are some conclusions that might be drawn from the data.

(1) **Upper bound on historical algorithms.** Algorithm OD-ALL with infinite budget will eventually crawl every page that has historically produced an outlink to new content. Disturbingly, even this aggressive approach is sufficient to cover only 74% of the new content. This suggests that much new content during any given week is extremely difficult to discover.

(2) **Extent of historical information.** Algorithms OD-WIN and CLIQ-WIN, averaged over recrawl frequencies ranging from 2 to 6, capture 69% of the new content. Algorithm

OD-1, which has access only to the information from the most recent recrawl, is able to capture only 44% of the new content — the set of old pages considered for any time step is the smallest for OD-1. Thus, the entire collection of pages that referenced new content during the previous week is not adequate to discover new content during the current week, and in fact captures only 55% of the content that can be discovered using pages that have historically linked to new content. Purely recent statistics are not sufficient to discover new content effectively.

(3) **Comparison between different statistics.** The algorithms CLIQ-WIN and OD-WIN perform similarly to each other in both overhead and coverage, while the COV algorithm has lesser overhead, but with less coverage. We observe that incorporating aggregated past information significantly reduces the overhead of OD, but has smaller impact on CLIQ-1. Recall that the primary advantage of the CLIQ-1/CLIQ-WIN family is that they make more efficient use of collections of pages, all of which reference the same new content. The impact of aggregated historical statistics is sufficient to make this overlap almost irrelevant in terms of both overhead and coverage, and therefore it is enough to track degree statistics over time.

Based on these observations, we move to an evaluation of realistic candidates for discovery algorithms. Figure 8 plots coverage as a function of average depth (which is equivalent to average cover size) based on statistical information created during the previous timestep (and earlier for algorithms that aggregate historical information). There are two conclusions. First, COV performs very well up to 32% coverage, then is unable to cover any more new content. Second, Algorithm CLIQ and algorithm OD perform very similarly, and have the best coverage in the limit.

Figure 9 shows the same information when historical data is available based only on monthly recrawls. The scaling of the x-axis allows the overhead of the algorithms to be compared, but does not show that total coverage asymptotes at 59% rather than 69% when more recent information is available.

Our conclusion is the following. For highly efficient discovery of a smaller fraction of new content, COV performs exceptionally well. But for discovery of as much new content as is realistically possible, algorithm OD-WIN performs nearly as well as alternatives and is particularly simple to implement.

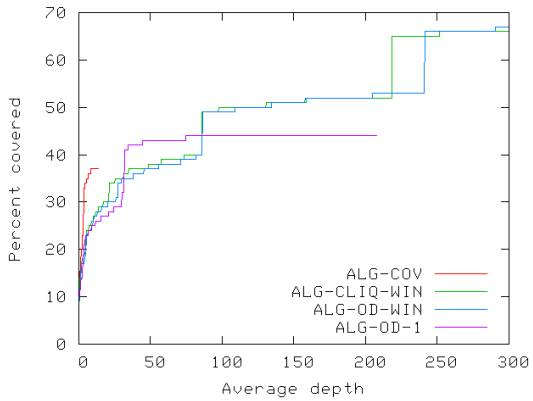

Figure 8: Coverage as a function of average cover size, recrawl frequency 1.

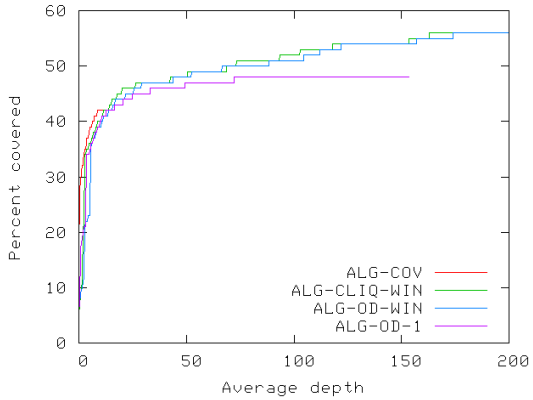

Figure 9: Coverage as a function of average cover size, recrawl frequency 4.

6. FUTURE WORK

All the previous algorithms assume periodic complete recrawls to aid discovery of new content but do not account for the associated cost. Ideally we would like to allocate the crawler budget more efficiently to simultaneously exploit already known high yield pages as well as explore other possible pages with unknown yield.

Given a limited crawler budget, we model the tradeoff between crawling pages with known high yield (exploitation), and pages with unknown yield to discover other high yield pages (exploration) as an instance of the *multi-armed bandit* problem [15, 1], which is the following: the bandit has n arms, and each arm is associated with a fixed payoff probability that is unknown to the policy. At every timestep each of the n arms generates unit reward with the corresponding payoff probability. The bandit policy can activate k arms at each timestep. On activating an arm, the policy collects the reward generated by that arm in that timestep (which is either 0 or 1), and can simultaneously update its estimate of the payoff probability of that arm. The aim is to maximize the total expected payoff of the policy over time.

Note that designing a reward function for a bandit policy is nontrivial in this setting, since the total reward of a set of k arms can be less than the sum of the rewards from each arm, unlike the usual setting. However, based on the performance of OD-WIN, we design the bandit policy to converge to the set of k arms with the highest (aggregated) outdegrees. In our case the arrival of each new page defines a timestep. Each existing page is an arm of the bandit with payoff probability p_i, the mean fraction of new pages it covers. Unlike the conventional bandit formulation, k arms are not activated for each new page arrival, but rather in batches corresponding to snapshots.

Various bandit policies [1, 15] can be used with the above formulation. Early experiments indicate that the bandit policies can lead up to discovery of 64% coverage of new content, with overhead comparable to OD-WIN. We hope to include detailed experiments in a full version.

Acknowledgments. We are very grateful to our colleagues who have provided us with valuable data to perform this

Budget	Depth	Overhead	Coverage	Budget	Depth	Overhead	Coverage	Budget	Depth	Overhead	Coverage
	Cliq-Win				Cov				Od-Win		
1	0.00	14.77	8%	1	0.00	13.40	9%	1	0.00	14.73	8%
10	4.34	49.75	19%	10	2.91	37.04	23%	10	4.35	51.81	18%
100	37.09	100.2	37%	100	9.36	13.89	37%	100	37.30	120.5	35%
1000	218.34	153.8	52%	1000	11.80	13.89	37%	1000	218.07	151.5	53%
10000	647.63	156.3	69%	10000	13.40	13.89	37%	10000	649.17	153.8	69%
	Od − 1				Optimal				Od-All-1		
1	0.00	13.48	9%	1	0.00	2.22	56%	1	0.00	7.79	16%
10	3.65	45.25	21%	10	3.03	12.79	81%	10	4.49	42.37	24%
100	21.82	106.4	28%	100	9.65	26.39	95%	100	40.09	121.9	36%
1000	67.49	109.9	43%	1000	11.96	26.74	98%	1000	249.05	161.3	55%
10000	181.77	109.9	44%	10000	13.56	26.74	100%	10000	870.83	163.9	74%

Table 2: Analysis of covers produced by historical algorithms.

study. Thanks in particular to Junghoo Cho of UCLA for providing the site recrawl data and to Ricardo Baeza-Yates, Carlos Castillo, and Rodrigo Scheihing of Yahoo! Research Barcelona for providing the Chilean web data.

7. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2/3):235–256, 2002.

[2] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[3] B. E. Brewington and G. Cybenko. How dynamic is the web? *WWW9 / Computer Networks*, 33(1-6):257–276, 2000.

[4] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *WWW9 / Computer Networks*, 33(1-6):309–320, 2000.

[5] J. Cho and H. Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *Proc. 26th VLDB*, pages 200–209, 2000.

[6] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In *Proc. SIGMOD*, pages 117–128, 2000.

[7] J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through URL ordering. *WWW8 / Computer Networks*, 30(1-7):161–172, 1998.

[8] F. Douglis, A. Feldmann, and B. Krishnamurthy. Rate of change and other metrics: A live study of the world wide web. In *Proc. 1st USENIX Symposium on Internet Technologies and Systems*, 1997.

[9] J. E. Coffman, Z. Liu, and R. R. Weber. Optimal robot scheduling for web search engines. *Journal of Scheduling*, 1(1):15–29, 1998.

[10] J. Edwards, K. McCurley, and J. Tomlin. An adaptive model for optimizing performance of an incremental web crawler. In *Proc. 10th WWW*, pages 106–113, 2001.

[11] N. Eiron, K. S. McCurley, and J. A. Tomlin. Ranking the web frontier. In *Proc. 13th WWW*, pages 309–318, 2004.

[12] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proc. SIGCOMM*, pages 251–262, 1999.

[13] D. Fetterly, M. Manasse, and M. Najork. the evolution of clusters of near-duplicate web pages. In *Proc. 1st LA-WEB*, pages 37–45, 2003.

[14] D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener. A large-scale study of the evolution of web pages. *Software Practice and Experience*, 34(2):213–237, 2004.

[15] J. Gittins. *Bandit Processes and Dynamic Allocation Indices*. John Wiley, 1989.

[16] M. Kearns. *The Computational Complexity of Machine Learning*. MIT Press, Cambridge, 1990.

[17] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the web for emerging cyber-communities. *WWW8 / Computer Networks*, 31:1481–1493, 1999.

[18] M. Mitzenmacher. A brief history of lognormal and power law distributions. *Internet Mathematics*, 1(2):226–251, 2004.

[19] 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. 13th WWW*, pages 1–12, 2004.

[20] 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. 13th WWW*, pages 1–12, 2004.

[21] S. Pandey and C. Olston. User-centric web crawling. In *Proc. 14th WWW*, pages 401–411, 2005.

[22] J. Pitkow and P. Pirolli. Life, death, and lawfulness on the electronic frontier. In *Proc. CHI*, pages 383–390, 1997.

[23] P. Slavik. *Approximation Algorithms for Set Cover and Related Problems*. PhD thesis, SUNY at Buffalo, 1998.

[24] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[25] J. L. Wolf, M. S. Squillante, P. S. Yu, J. Sethuraman, and L. Ozsen. Optimal crawling strategies for web search engines. In *Proc. 11th WWW*, pages 136–147, 2002.