Ph.D. Thesis Proposal: Optimal observation with WWW applications[*]

Brian E. Brewington[†]
Thayer School of Engineering
Dartmouth College
Hanover, New Hampshire 03755-8000

brian.e.brewington@Dartmouth.edu

January 7, 1999

Abstract

This thesis proposal deals with optimal observation of large collections of changing objects. These objects can change at random times, so we cannot know the state of objects for times at which they are unobserved. The goal of an observer is to maintain acceptably accurate state estimates while minimizing observational cost. In the thesis work, our goals are to (1) create models for this type of system, (2) show how these models can be empirically constructed by actually observing real systems, and (3) develop efficient algorithms for the optimal allocation of observation resources within this framework. An example of this type of optimization arises in the observation of World Wide Web (WWW) documents by web search engines and related web software applications. Our initial results include (1) developing statistical models for such systems; (2) collection of empirical data about how web documents change; and (3) development of finite-horizon algorithms for maximizing an index's accuracy. Although the algorithms presented are intended for optimizing the recency of document indices, they are general enough to be applied to any dynamic collection of objects. The work is important and novel in that it takes proper account of the cost of observing, a concept that is crucial to monitoring problems in many communications systems.

1 Introduction

In any system in which observation resources are limited, a choice must be made as to how they are best allocated. The guiding principles in this allocation are essentially independent of the system under observation: an observer desires comprehensiveness and accuracy, attained at the price of making observations and interpreting the results. While a system is not under observation, its state is accumulating uncertainty according to the dynamics of its evolution. When this uncertainty becomes intolerable, observations are

[*]available from http://comp-engg-www.dartmouth.edu/~brew/research/proposal.ps
[†]Partially supported by AFOSR grant F49620-97-1-0382.

made and combined with past state estimates to form a sufficiently accurate estimate of the current system state.

This is not a new idea–biological sensory systems are always forming state estimates using limited observation capability. Moreover, they are extremely efficient, in that they rarely waste effort paying attention to slow-changing or unimportant systems. Human vision discerns movement within static fields, and smells and sounds cease to be novel after they have been sensed for some short time. Consider where visual focus is directed when driving: the road is the source of the most important information, as well as the source which changes most quickly. Changes in the state of other items, such as the rear-view mirror image, the vehicle speed, the radio reception, and the fuel remaining are deemed either less important or more slowly varying, and are therefore given less frequent visual attention.

To illustrate and formalize some of these characteristics of observation systems, we propose an in-depth analysis of an example. A World Wide Web (WWW) search engine is a dedicated observation system with the goal of having the most accurate picture of the Internet possible at any given time. To achieve this goal, limited computational, network, and storage resources are devoted to scouring the Web for new documents, and also to re-examining old documents to inspect them for changes. Whether done in sequence or in parallel, a search engine must always decide what page or pages to examine next. This reduces to some simple questions: when is the best time to re-examine a document, given knowledge of that document's history and the priority placed on having correct knowledge of its state? Indeed, how should we describe a document's state?

If resources were unlimited, optimality would be a non-issue: each and every document could be monitored as frequently as desired, watching for changes to appear. Of course, an observation does have obvious costs associated with it, since a machine uses time (some network latency and some CPU cycle time) to retrieve and inspect a document, and disk space to store the results. In exchange for this cost, the search engine benefits from a more current index of previously explored documents, a more comprehensive collection (if new documents are discovered), and an accurate picture of the "dynamics" of the documents in question.

An understanding of how documents change is necessary to maximize the recency of the index, since knowing a document's change dynamics allows us to waste fewer observations. When the engine must decide which document to examine next, some documents will be more likely to have changed since last inspection than others. It makes sense to re-examine these more often than documents which exhibit greater stability.

Algorithms for selecting the next observation can also account for how the collection may look as a result of page checks yet to be run, so that planning can take into account the likely outcome of making observations. Looking ahead requires that we consider not only the immediate result of an observation, but also the long-term value of the information so obtained. If the index is being used for user searches, then it has the most value for frequently-requested documents. It seems reasonable that resources should be preferentially allocated to the documents that are popular, fast-changing, or both.

For all documents, this requirement can be interpreted within a common framework: we will choose to observe a document when the uncertainty in its state becomes intolerable. A documents's dynamics will determine how quickly we lose confidence in previous observations, and the popularity of that document will determine how much uncertainty is tolerable. This fundamental idea was developed in [CBB$^+$97] in the context of alleviating uncertainty through communications actions. The work described systems in which knowledge of dynamics was put to use in anticipating the accumulation of uncertainty.

We do not begin with knowledge of the dynamics of our documents prior to their observation, so planning must involve building models from previous observations. This is a tricky problem, since we will be basing observation schedules on these dynamic models. Similar problems in the experimental design literature are referred to as "allocation problems" [HS98], classic examples of which are "bandit problems" [Ber87]. The terms stems from the plight of a casino gambler, having the option of playing some number of slot machines (colloquially known as "one-armed bandits"). The payoff from each machine is governed by some unknown probability distribution. The gambler wishes to maximize his winnings, but without knowing which slot machine is likely to pay off, his strategy is initially uncertain. Every play he makes is a mixture of experiment and attempted profitability—the gambler bets the cost of one observation that he will eventually increase his expected winnings. This is done by modifying strategy and playing the most profitable machines, while still occasionally exploring less well-understood possibilities.

For the search engine, the parallel is played out with the search engine's web robot as the gambler, and individual documents as slot machines. The payoff is expressed in terms of the expected benefit gained by re-indexing a page. Though bandit problems are fairly tractable for problems involving only a few machines, they quickly become complicated as problem size increases. Inasmuch as a lower bound on the size of the indexable WWW has been estimated to be near 320 million pages [LG98], the web gambler faces a challenging task. Further complicating the game, a document's change probability is generally nonstationary: page changes tend to become more likely after some time has passed since the last change. The web robot also assigns additional value to having looked at all pages in the collection; the gambler would be happy to play only one machine forever, if it always paid off. Still, the underlying problem remains the same: what is the best way to bet the cost of observation to maximize winnings?

2 Analysis and modeling

Before considering the value of experimentation, it is necessary to understand how the problem can be approached if we have precise knowledge of document dynamics. To do so, three things are essential: a representation of the document's state, knowledge of the dynamics of state evolution, and a formalization of the value of perfect state knowledge.

2.1 State definition

To choose an appropriate state definition, we first consider the reasons pages change. The simplest possible model assumes that page changes occur like lightbulbs failing, such that page "lifetimes" are governed by a single-parameter memoryless process. Some promising work has been conducted for describing the maintenance of document collections in which pages change according to exponential distributions with known parameters [CLW97], in which it is shown that all pages in a collection should be revisited at intervals that are as regular as possible. However, some page change dynamics may not be well-described by memoryless processes, so these results may not apply. Additionally, it may not be easy to determine in advance which pages' dynamics are memoryless and which are not. In this section we demonstrate that the time since a page last changed can be an important factor in predicting the likelihood of future changes.

Viewed very simplistically, page changes occur when a page's maintainer notices and reacts to newly-discovered information related to that page's content. The time between page changes is partly from waiting for these motivations to arrive, and partly from waiting for the maintainer to respond. By modeling these two times as independent random variables, we can draw some simple conclusions about their sum. Although the accumulation of motivation may progress according to rather complicated dynamics, assume for this discussion that motivation arrives in a memoryless fashion, such that the time between these arrivals is governed by

$$D_1(t) = \lambda_1 e^{-\lambda_1 t}, t \geq 0. \tag{1}$$

Further, assume that the time it takes the maintainer to respond to the arrival of new motivation is a similar random variable:

$$D_2\left(t\right) = \lambda_2 e^{-\lambda_2 t}, t \geq 0 \tag{2}$$

In (1) and (2), the parameters λ_i describe the speed of the two processes, with large values corresponding to shorter times. The total time taken from motivation to response is the sum of these two random variables. Since the two times are independent, the probability of a particular sum is the product of the two probabilities of the summands. To find the probability of a particular sum t, we integrate the product over all possible two-term combinations that add to t. This integral is a convolution of (1) and (2):

$$D_3\left(t\right) = D_1\left(t\right) * D_2\left(t\right) = \lambda_1\lambda_2 \int_{x=0}^{x=t} e^{-\lambda_1 x} e^{-\lambda_2 (t-x)} dx = \frac{\lambda_1\lambda_2}{\lambda_2 - \lambda_1}\left[e^{-\lambda_1 t} - e^{-\lambda_2 t}\right] \tag{3}$$

If $\lambda_1 = \lambda_2$, the final result is indeterminate (0/0), but this problem can be avoided by simplifying the integrand of (3). Notice also that (3) is maximized for the value of t where $D_1(t)$ and $D_2(t)$ intersect:

$$t_{max} = \left(\lambda_1 - \lambda_2\right) \ln\left(\frac{\lambda_1}{\lambda_2}\right) \tag{4}$$

An example of how the distributions are related is shown in Figure 1. Next, we show that the sum of these two memoryless variables is no longer memoryless. For a memoryless distribution $f(t)$, the passage of a time t_0 should not change the distribution. If $f(t)$ is causal (i.e., $f(t) = 0$ for $t < 0$), then this implies that:

$$f\left(t\right) = \frac{f\left(t + t_0\right)}{1 - \int_0^{t_0} f\left(t\right) dt} \tag{5}$$

This is just a requirement that the distribution $f\left(t\right)$ be self-similar, such that if time advances by t_0, a simple rescaling of $f\left(t + t_0\right)$ will recover the original function $f\left(t\right)$, as is easily confirmed for exponential distributions. To show that (3) is not memoryless, we can use (5), or we can simply observe that it cannot be self-similar. Clearly, as the time t_0 advances past t_{max}, the function goes from being unimodal (having maximum at $t = t_{max} - t_0$) to being monotonically decreasing (having maximum at $t = 0$), and therefore does not preserve the original shape.

If the decay constants in D_1 or D_2 are very different, then the resulting page change interval distribution will be nearly memoryless. When either λ_i is large, then D_i resembles an impulse $\delta(0)$, so the convolution (3) will return a distribution that is very nearly exponential with the smaller decay parameter. This corresponds

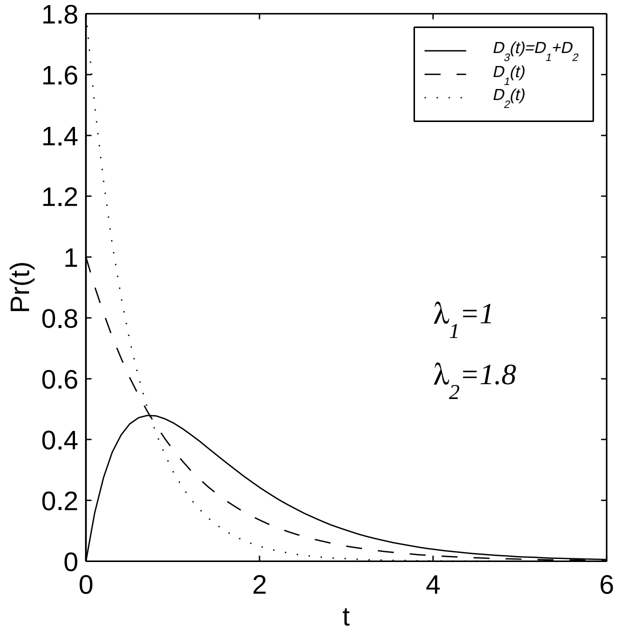

Figure 1: The sum of two memoryless random variables is no longer memoryless

to the case in which either D_1 or D_2 dominates the sum: reasons to change a page appear much more quickly (λ_1 large) than the responses, or the page maintainer is very quick to respond when motivators appear (λ_2 large). Regardless of how close λ_1 and λ_2 are, (3) will always be dominated by the smaller decay parameter for large t.

This simple model of how pages change suggests that it may not always be appropriate to assume that page changes are independent of the age (time since the page was last updated) of the page. We temper this by saying that the distribution may be nearly memoryless under some circumstances. Nonetheless, we model the probability of a page's modification as a function of the time since the last modification, to account for the possibility of time-dependence.

2.2 Time-since-modification as a Markov chain

Since the probability of a page changing may be a function of the time elapsed since the previous modification, at least some portion of the page's state must be its age. While other factors undoubtedly influence a page's likelihood of change, this proposal assumes that the likelihood of a change is completely determined by the age. While this age could be treated as a continuous state, quantizing will allow us to keep track of a large number of models by decreasing storage requirements per monitored document. Any reasonable discretization of this time will serve our purpose, such as the number of days since the last page change, for example. Using this definition of state, a probability distribution of the time intervals between changes can be represented as a Markov chain. This is an oft-used means for describing time-dependent recurrent events; see [Fel68] for example. Given a state $s = n$ days since the last change, there are only two things that can occur: either the state will advance to $s = n + 1$, or it will reset to $s = 0$. If we model $N + 1$ states, then the state $s = N + 1$ can be treated as "N or more days since last change." In this way, we can define a matrix of transition probabilities as

$$\mathbf{M} = \begin{bmatrix} p_{reset}(0) & 1 - p_{reset}(0) & 0 & \cdots \\ p_{reset}(1) & 0 & 1 - p_{reset}(1) & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ p_{reset}(N) & 0 & \cdots & 1 - p_{reset}(N) \end{bmatrix} \tag{6}$$

If $p_{reset}(t) \neq 1 \forall t \in [0, N]$, then all states of $\mathbf{M}$ are recurrent. The function $p_{reset}(t)$ can be determined either from the distribution of time intervals between changes, or from the distribution of observed ages for a particular page. If we find either by estimation from samples, we are implicitly assuming stationarity of the underlying change dynamics[1]. Both the age and lifetime distributions (whether or not they can be modeled) can be used to calculate the conditional probability that a reset will occur in the following time interval, given that no change has occurred for t time steps. If the time between page changes (lifetime) is distributed according to a discrete probability mass function $f(t)$, then

$$p_{reset}(t) = \frac{f(t)}{1 - \sum_{k=0}^{t-1} f(k)} \tag{7}$$

In modeling $\mathbf{M}$ for a page, we will only be able to define $f(t)$ for values up to some $t = N$. At the upper age limit, the definition for p_{reset} breaks down, since $p_{reset}(N)$ would be unity. This would imply that if the page ever reaches age N, then it will change with probability 1 within the next time step. In reality, though, there must be some chance that the state will remain "N or more time intervals since last change," or $s_i = N$. One way to represent this possibility is to assume that $p_{reset}(N) = p_{reset}(N-1)$, or that the reset times are memoryless after a certain time has elapsed.

[1]Additional model types are probably needed and will be addressed further in the thesis. For example, empirical modelling will be inaccurate for a page that changes every weekday, but never on the weekends. The inaccuracy in the model corresponds to the fact that we have not captured the entire state.

Generally, it will be easier to observe ages of objects rather than change intervals[2]. Webpages typically have a `LAST_MODIFIED` date in the `HTTP` header, allowing us to sample the age of the document at any time. Over time, one can build up a distribution $g(t)$ of observed ages for the document. Moving from an age distribution $g(t)$ to p_{reset} is straightforward. As with the function $f(t)$ defined for (7), we express the age t of an object in some arbitrary units of time. The two probabilities that we wish to determine for age t are the probability of aging to $t+1$ and the probability of being reset to age $t=0$. That is, some fraction of the objects "dies" (changes) before reaching the next age. This fraction, which is $p_{reset}\left(t\right)$, is just the percent difference between successive elements of the age distribution:

$$p_{reset}\left(t\right) = \frac{g\left(t\right) - g\left(t+1\right)}{g\left(t\right)} \tag{8}$$

To get a feeling for how the matrix $\mathbf{M}$ corresponds to different change distributions, we present some simple examples. If a page changes on a purely periodic basis, say every 3rd time unit, then the matrix $\mathbf{M}$ will be

$$\mathbf{M} = \begin{bmatrix} 0 & 1 & 0 & \cdots \\ 0 & 0 & 1 & \cdots \\ 1 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{9}$$

If the system changes according to a Poisson process, then it is memoryless, so the reset probability is independent of state:

$$\mathbf{M} = \begin{bmatrix} \rho & 1-\rho & 0 & \cdots \\ \rho & 0 & 1-\rho & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \cdots & 0 & 1-\rho \end{bmatrix} \tag{10}$$

We emphasize that this particular definition of state is not the only useful one; other models can emphasize different aspects of the problem. If we were interested only in capturing the fact that one or more changes had occurred, a model could be used that is similar to the one used in the classic machine replacement problem [How60]. This model would be similar to $\mathbf{M}$, except that it would not monitor the age after a page change had occurred. The system would begin in a transient state, "unchanged, with age i", and when a change occurs, the system would enter an absorbing state, "changed, with unknown age". Observation

[2]Observing change intervals directly would involve recording the state and then monitoring the page at a high sampling rate. There can be serious difficulties with aliasing, as any change which is observed might be only the last change in a long series.

would force a reset to the transient state having the appropriate age. This and other state definitions will be explored more fully in the thesis.

Formally, we define the ij^{th} element of $\mathbf{M}$ to be the probability that a page of age i will become age j on the following time step. More generally, by raising $\mathbf{M}$ to an integer power k, we can find the probability (for initial age i) that the system became age j after k time steps have elapsed since the last observation:

$$\text{Prob}\left(s_{t+k} = j | s_t = i\right) = \left[\mathbf{M}^k\right]_{ij} \tag{11}$$

As k grows without bound, the ij^{th} element of $\mathbf{M}^k$ approaches the stationary transition probability π_j, which is the probability that the page will be observed to be age j if we are given no knowledge of the history of the page. This is precisely the age distribution mentioned above, $g(j)$:

$$\lim_{k\to\infty} \left[\mathbf{M}^k\right]_{ij} = \pi_j = g\left(j\right) \tag{12}$$

2.3 Defining a cost function for the index

Using the model just developed, we can define an objective function that can be optimized for the index of the collection. The objective in a real system could be fairly complex; we will not seek to incorporate all these features into our initial work on the cost function. Instead, here are two possible choices for objective functions:

1. Minimize the expected number of pages that are incorrectly indexed in the collection. Objects in the collection are indexed correctly if no changes have occurred in the object that have not been recorded in the index.

2. Minimize the total time out-of-date for the entire collection (as was developed in [CLW97]). Unrecorded object changes add to the cost in proportion to how long the object is expected to have been out-of-date.

We restrict the problem space using some simplifying assumptions. First, assume the document collection contains a constant d documents. Second, assume that pages can be retrieved at a rate of α pages per day. Define the states corresponding to the rows of the matrix $\mathbf{M}_r$ to denote the age in days of the r^{th} page in the collection. Any unit of time could be used, so long as it is consistent across the collection and the rate

α is expressed in the same unit. For each page r, we know that it was last observed k_r days ago to have age i_r. The probability that page r has changed during this k_r-day interval is

$$\begin{aligned} \mathrm{Prob}\left(change|\left\{i_r, \mathbf{M}_r, k_r\right\}\right) &= \sum_{j\in[0,i_r+k_r-1]} \left[\mathbf{M}_r^{k_r}\right]_{i_r j} \\ &= 1-\left[\mathbf{M}_r^{k_r}\right]_{i_r(i_r+k_r)} \end{aligned} \tag{13}$$

We note that this cost is between 0 and 1, corresponding to the cases in which we are absolutely certain that the page is not out-of-date (0) or is out-of-date (1) at the end of a day.

Using this result, we can formulate costs such as the two listed above. For example, at any given time, the expected total number of pages that are incorrectly indexed (the indexed version is out-of-date) is just the sum of the probabilities listed in (13) over the entire collection:

$$C_{(1)} = \sum_{r=1}^{d} \left(1-\left[\mathbf{M}_r^{k_r}\right]_{i_r(i_r+k_r)}\right) \tag{14}$$

Since all terms are on [0,1], the cost $C_{(1)}$ is nonnegative and can be no greater than the number of documents d.

We can calculate the second metric, the expected time out-of-date, by performing a similar sum in which the out-of-date probabilities weight the number of days out of date to which they correspond:

$$Z\left(i_r, k_r\right) = \sum_{j=0}^{k_r-1} \left(k_r - j\right) P\left(j+i_r, i_r\right) \tag{15}$$

Here, $P(j, i_r)$ is the probability that a page goes out-of-date at exactly age j, given that the page was last observed to have age i_r. This can be found from the conditional probabilities $p_{reset}(t)$. For example, $P(1,0)$ is, trivially, just $p_{reset}(0)$. The probability of having gone out of date on the second day, $P(2,0)$, requires that the page has both aged past the first day, with probability $(1-p_{reset}(0))$, and reset on the second day, with probability $p_{reset}(1)$. These are independent events, so

$$P(2,0) = (1-p_{reset}(0))p_{reset}(1). \tag{16}$$

This same logic describes $P(N, i_r)$, since the event of resetting on the N^{th} day is conditioned upon the independent events of having reached that age, one day at a time. Generally,

$$P(j, i_r) = p_{reset}(j)\,[1 - p_{reset}(j-1)] \ldots [1 - p_{reset}(i_r)]\,, \tag{17}$$

which can be written as the repeated product

$$P(j, i_r) = p_{reset}(j) \prod_{m=i_r}^{j-1} [1 - p_{reset}(m)]. \tag{18}$$

The terms in the product describe the independent probabilities of progressing through each of the ages from i_r to $j-1$, which we multiply by the probability of a change on day j. Note that these probabilities can be calculated recursively:

$$P(j+1, i_r) = P(j, i_r)\,\frac{[1 - p_{reset}(j)]}{p_{reset}(j)}\,p_{reset}(j+1) \tag{19}$$

Recursion can also be used to simplify the calculation of (15). As an example, consider the difference between finding $Z(i_r, 3)$ and $Z(i_r, 4)$, which demonstrates how to advance the cost contributed by a single page if one day elapses:

$$\begin{aligned} Z(i_r, 3) &= 3P(i_r, i_r) + 2P(i_r+1, i_r) + P(i_r+2, i_r) \\ Z(i_r, 4) &= 4P(i_r, i_r) + 3P(i_r+1, i_r) + 2P(i_r+2, i_r) + P(i_r+3, i_r) \end{aligned} \tag{20}$$

If the argument j is advanced by 1, we can simply add to our previous cost value a partial sum of the sequence $P(j, i_r)$. Using (20), the difference in consecutive values would be calculated as:

$$\begin{aligned} Z(i_r, 4) - Z(i_r, 3) &= P(i_r, i_r) + P(i_r+1, i_r) + P(i_r+2, i_r) + P(i_r+3, i_r) \\ &= S_3(i_r) + P(i_r+3, i_r) \end{aligned} \tag{21}$$

where we have defined

$$S_N(i_r) = \sum_{j=0}^{N-1} P(j + i_r, i_r) \tag{22}$$

to be the N^{th} partial sum of the sequence of probabilities $P(i_r + j, i_r)$. Obviously, this partial sum can be calculated recursively as well, since $S_{N+1}(i_r) = S_N(i_r) + P(i_r + N, i_r)$. Being a partial sum of mutually exclusive and exhaustive probabilities, $S_N(i_r)$ approaches 1 as $N \to \infty$. The probabilities are exclusive, since a page cannot become out-of-date on more than one day, and they are exhaustive in that the only possible times for the page to go out-of-date are $t \in [i_r, \infty]$.

Summarizing, we can recursively calculate Z as

$$Z\left(i_r, N+1\right) = Z\left(i_r, N\right) + S_{N+1}\left(i_r\right) \tag{23}$$

In the limit, each step simply adds one to the expected number of days out-of-date, corresponding to the notion that an infinitely old page will have been modified at least once, and therefore grows one more day out-of-date for each day that goes by without an observation.

Having defined $Z(i_r, N)$, the cost function for the entire collection is just the sum of this metric over all documents:

$$C_{(2)} = \sum_{r=1}^{d} Z\left(i_r, k_r\right) \tag{24}$$

2.3.1 Greedy cost minimization

Using (14) and (24), we can find the best way to reduce the costs that we will incur in the coming day. By our assumption, we can check α pages per day. The smallest possible cost for the following day can be obtained if we fetch and re-index the α pages corresponding to the largest terms in the cost summation. These terms correspond to those pages with the largest probability of being out of date (14), or those that we expect to have been out-of-date the longest (24). For all of the α pages we fetch, the probability of being incorrectly indexed is zero. If there is no single best choice for the α pages, then we can select α at random from the pool of best choices. This situation occurs when applying (14) if more than α pages have probability 1 (to working precision).

These are the greedy versions of cost minimization, inasmuch as the greatest immediate cost reduction is chosen. Refinements of these methods operate along the same lines, observing those α pages which afford the greatest reduction in cost. More complex methods can find the expected cost for more than a single day, and the actions corresponding to the lowest long-term expected cost can be used. These will be discussed in a later section, giving more assurance of long-term optimal performance.

2.3.2 "Liveness" conditions

While having the advantage of being relatively simple, greedy algorithms may force suboptimal long-term performance. For example, in the re-indexing system, there is the possibility of never checking some subset of the pages. This is a situation that must be avoided, especially if all items to be indexed are equally important. In queuing models for computer operating systems, the analogous constraint that all processes be served is termed a "liveness" condition, which would not be met if there were a subset of pages that changed so quickly that its members always contributed the largest terms in the cost function. Fortunately, the two algorithms presented both can be shown to eventually check all pages.

We first consider liveness with respect to the expected pages out-of-date described by (14). Since each term in the summation is a probability, no term can be larger than 1. Moreover, once a term becomes unity, it remains so until the page to which it corresponds is checked. If all terms eventually become 1, then all pages will be checked. Examining the form of each probability term (as given in (13)), it is clear that this probability becomes unity for page r for any time that forces the sum to include an entire row of the matrix $\mathbf{M}_r$. As mentioned above, if there are more than α pages corresponding to terms having probability 1, then some subsidiary selection process must be used. If no preference is given to any page, then even a random selection will guarantee that all pages having the maximum probability (in this case, 1) will be checked eventually. Therefore, the liveness condition is satisfied by the one-step greedy minimization of the expected number of non-current pages.

Liveness follows in similar fashion for the expected number of time units out-of-date in (23). If the pages with the α largest values of $Z(i_r, k_r)$ are selected for observation, as in the greedy algorithm, then any page will eventually be included in the observed set. This stems from the fact that $Z(i_r, k_r)$ will increase without bound if page r is unchecked, thereby guaranteeing that page r will have a large enough cost to guarantee its inclusion in the observed set (if we wait long enough).

To show this, we define the smallest member of the inspected set of α inspected documents at time step t to have expected time out-of-date $C_{min}(t)$. For the collection, there will be some time C_{max} that is greater than or equal to $C_{min}(t)$ for all t. That is, there is a largest member of the set of smallest costs. In order for a document to be included in the inspected set of α documents, it is sufficient to guarantee that its expected total time out-of-date be greater than C_{max}. From (23), as k_r grows without bound, we are eventually just adding unity to the cost with each passing day. Therefore, the expected days out-of-date $Z\left(i_r, k_r\right)$ will eventually become greater than any finite value C_{max}. By analogy, imagine we have a lawn in which each blade of grass will grow forever (becoming more out-of-date) if we never cut (observe) it. Although grass in the lawn grows at different rates, no matter how high we set the wheels of the mower, we can always be assured that any blade will eventually grow high enough be cut.

2.3.3 Extending the horizon: two-day cost functions

Having an assurance of liveness is not enough to be satisfied with the long-term performance of the system. The one-step algorithm does not take into account anything other than the current probability of change for various pages. Indeed, no one-step method will make use of the difference in change rates among pages. By using our suspicions of likely behavior beyond the first time step, we can improve expected performance in future phases of observation. We explore this by considering cost functions evaluated over two days.

A simple two-day example Consider the following simple system that demonstrates how to take advantage of page change rates. There are two pages, A and B. Page A changes quickly: it has a probability of 85% of having been changed today, and if unobserved, it will have a 95% chance tomorrow. If, however, we observe it today, there will be a 25% chance of it having been changed by the end of tomorrow. Page B changes more slowly. It has an 80% chance today, which becomes 81% tomorrow if B is unchecked today. If it is checked today, then tomorrow's probability will be 5%. Assume we can only choose one of these to observe per day, and that we wish to minimize the total expected number of pages out-of-date over the two-day period:

$$\sum_{t=0}^{1} \kappa^t \left[\text{Prob}(A \text{ changed}, t) + \text{Prob}(B \text{ changed}, t)\right] \tag{25}$$

Here, $\kappa \in [0,1]$ is a "discount factor" with which we account for the relative value of cost avoided today versus that on subsequent days. This is a reflection of the possibility that immediate benefit may be more valuable than deferred benefit. For this illustration, we assume $\kappa = 1$.

There are four possible strategies for two-day observation. These can be written as sequences of observations, namely AA, AB, BA, and BB. If we observe a page, then it contributes zero cost on that day, since we consider it "up-to-date" if indexed within the last day. Therefore, the cost for observing A on both days is exactly the cost of not observing B on those days, namely, $0.8 + 0.81 = 1.61$. Likewise, we can find the two-day cost for each possible sequence of observations, as shown in Table 1. As can be seen, the greedy

Sequence	Cost	Comment
AB	$0.8 + 0.25 = 1.05$	lower first day cost; one-day algorithm would pick this one
BA	$0.85 + 0.05 = 0.90$	lower total cost; two-day algorithm would pick this one
AA	$0.8 + 0.81 = 1.61$	page B ignored
BB	$0.85 + 0.95 = 1.80$	page A ignored

Table 1: Possible costs in example two-page, one-check system

algorithm would select a suboptimal two-day strategy.

General two-day costs In the general case, we have d pages and α observations. Each of the four possible strategies chosen for page r will have a single cost associated with it: (i) the cost of not observing on the first day but then observing on the second; (ii) the cost of not observing on either day; (iii) the cost of observing on the first day but not on the second; and (iv) the cost of observing on both days. The current cost calculations have the appealing feature that choosing to observe one page does not affect the cost that might be contributed by other pages. That is, the cost of not observing a page on any given day depends only on its age, dynamics, and how recently it was observed.

Moving through these cost possibilities in order, we know that the cost of not observing on the first day is the same as was given in (13):

$$C_{XO_r} = \sum_{j \in [0, i_r + k_r - 1]} \left[\mathbf{M}_r^{k_r}\right]_{i_r j} = 1 - \left[\mathbf{M}_r^{k_r}\right]_{i_r (i_r + k_r)} \leq 1 \tag{26}$$

Here, we introduce some new notation; we write two-day strategies for page r as sequences of X's (do not observe) and O's (observe). The first letter is the action on the first day, and the second letter indicates the second day's action. Thus the two-day cost of not observing page r on the first day and then observing it on the second day is written C_{XO_r}.

Alternatively, if we choose not to observe on the second day, we will add additional cost to C_{XO_r} to obtain C_{XX_r}. The probability that the page has gone out-of-date by the second day is exactly like (26), except that the page has aged by one day. This is accounted for by incrementing the value of k_r; if we choose not to observe on the second day, the cost for that day only is:

$$\sum_{j \in [0, i_r + k_r]} \left[\mathbf{M}_r^{k_r + 1}\right]_{i_r j} = 1 - \left[\mathbf{M}_r^{k_r + 1}\right]_{i_r (i_r + k_r + 1)} \leq 1 \tag{27}$$

We note that this second-day cost is greater than or equal to the first-day cost: by waiting an additional day, we can only add to the probability of having gone out of date. Specifically, we add the probability that the page went out of date on the additional day, which we calculated in (18). Adding (27) to the first-day cost (26), we obtain C_{XX_r}: the two-day cost of not observing on either day (denoted C_{XX_r}):

$$\begin{aligned} C_{XX_r} &= 2 - \left[\mathbf{M}_r^{k_r+1}\right]_{i_r(i_r+k_r+1)} - \left[\mathbf{M}_r^{k_r}\right]_{i_r(i_r+k_r)} \\ &= 2C_{XO_r} + P(k_r + i_r + 1, i_r) \end{aligned} \tag{28}$$

This cost is the entire cost for both the AA and BB options in Table 1, since both required that one of the pages not be observed for both days. In general, this will be the largest possible cost for a single page over the two-day period, but it cannot be greater than 2. Moreover, from the second line of (28) we know that

$$C_{XX_r} \geq 2C_{XO_r} \geq 0 \tag{29}$$

Next, we consider the cost C_{OX_r}, incurred if we observe page r on the first day but not on the second day. Zero cost is contributed on the first day, and the second day contribution depends on the new value of $k_r^+ = 1$ (days since the first-day observation), and the newly-observed age i_r^+. Since only a distribution of possible values of i_r^+ is known, the second-day cost will be a weighted sum of probabilities from the matrix $\mathbf{M}_r$ of one-day state transition probabilities (6). If the page was observed to be in state j_r on the first day, then the probability of it being out-of-date at the end of the second day is just $p_{reset}(j_r)$, as defined in (7). These probabilities are also the first column of the matrix $\mathbf{M}_r$. In our cost function, the values in this column vector will contribute in proportion to their probability of occurrence. These probabilities are obtained from the row over which we sum in (26), or the distribution of possible ages on the previous day. Therefore, the probability that the page is out-of-date at the end of the second day, given that the page was observed on the first day, is

$$C_{OX_r} = \sum_{j_r=0}^{N} \left[\mathbf{M}_r^{k_r}\right]_{i_r j_r} \mathbf{M}_{j_r 1} \leq 1 \tag{30}$$

Note that this cost, like C_{XO_r}, can be no greater than 1. If we expect that page r has changed with some large probability, then we would rather observe it today than tomorrow: $C_{XO_r} > C_{OX_r}$. Alternatively, if we expect that no change has occurred yet, then we would rather wait to make an observation. For example, if a page will change tomorrow with probability 1, but definitely will not have changed by the end of today, then it benefits us to wait until tomorrow to observe it: $0 = C_{XO_r} < C_{OX_r} = 1$. Most cases are less extreme than this, but we emphasize that the sign of the difference $C_{XO_r} - C_{OX_r}$ can be positive or negative.

Finally, for completeness, we note that two observations of page r will force it to contribute zero cost:

$$C_{OO_r} = 0 \tag{31}$$

This is generally the best option when page r is changing very rapidly, such that $C_{XO_r} \approx 1$ and $C_{XX_r} \approx 2$. Clearly, this implies that the page changes essentially everyday, making it an almost guaranteed success.

Now that we can determine a cost for all possible strategies, we are able to add these costs for an entire collection. Any observation plan for n days will partition the d pages into 2^n groups corresponding to the possible n-day strategies. In the case of $n = 2$, the groups are:

$$\begin{aligned} \mathcal{OO} &= \left[OO_1, OO_2, \ldots OO_{|\mathcal{OO}|}\right], \\ \mathcal{XO} &= \left[XO_1, XO_2, \ldots XO_{|\mathcal{XO}|}\right], \\ \mathcal{OX} &= \left[OX_1, OX_2, \ldots OX_{|\mathcal{OX}|}\right], \text{ and} \\ \mathcal{XX} &= \left[XX_1, XX_2, \ldots XX_{|\mathcal{XX}|}\right]. \end{aligned} \tag{32}$$

The sizes of the groups must be nonnegative, and are constrained so that no more than α observations can be made in a single day, and all documents are accounted for:

$$\begin{aligned} |\mathcal{OO}| + |\mathcal{OX}| + |\mathcal{XO}| + |\mathcal{XX}| &= d \\ |\mathcal{OO}| + |\mathcal{OX}| &= \alpha \\ |\mathcal{OO}| + |\mathcal{XO}| &= \alpha \end{aligned} \tag{33}$$

The first relation is a constraint that each document is assigned exactly one strategy, and the next two constrain the number of observations per day. Notice that these imply $|\mathcal{OX}| = |\mathcal{XO}|$. Since there are three equations and four unknowns, the system reduces to a single parameter choice. Once any one of the sizes is fixed, then the others are fixed as well—we could, for example, freely choose the number of documents that would be observed on both days at some $0 \le k \le \alpha$.

The "observe-n-times" group (e.g., group $\mathcal{OO}$) will always contribute zero cost; other groups may have nonzero cost. The cost for the collection is the sum of the cost over all members of all nonzero cost groups:

$$C^2_{(1)} = \sum_{k \in XX} C_{XX_k} + \sum_{l \in OX} C_{OX_l} + \sum_{m \in XO} C_{XO_m} \tag{34}$$

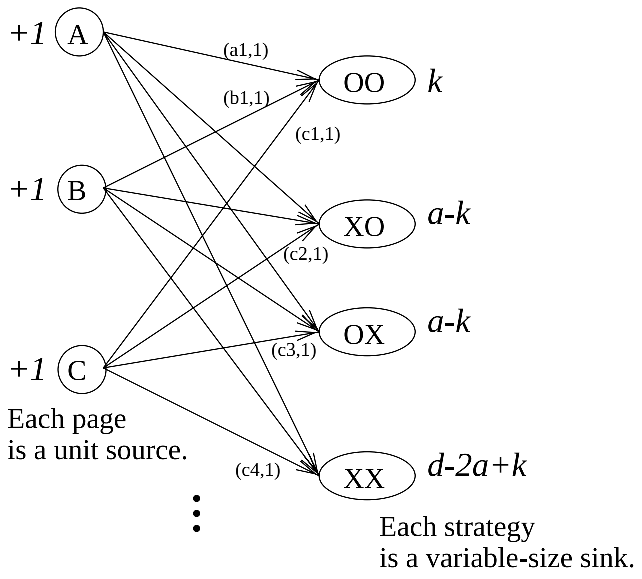

Figure 2: Two-day strategy assignment $(capacity, cost)$ network with k observations in class $\mathcal{OO}$

In a collection of d documents, in which we can check α per day, there are ${}_dC_\alpha^2$ possible combinations of two-day strategies. A brute-force approach, in which we evaluate a cost for each option, is entirely infeasible for the collection sizes under consideration.

Foruntately, the two-day problem can be stated as a simple minimum-cost network flow problem, as diagrammed in Figure 2. In this context, we think of documents "flowing" to particular two-day strategies, building the sets (32). Since no document can be assigned to more than one strategy, this is an integer flow problem. If the costs, sources, and sinks were real numbers, the problem restatement would not have helped—the multicommodity minimum cost integer flow problem has been shown to be NP-complete (see [CCPS98] for example). However, since the sources and sinks are all of integer size, integer flows can be found using standard algorithms without additional constraints [Ber98].

Each graph of this type has d unit sources corresponding to the documents in the collection, and 4 variable-size sinks corresponding to the allowable strategies. The sink sizes enforce the constraints on the number of observations and the number of strategies to be assigned. All sources are connected to all sinks by unit capacity links. Costs on these links correspond to those presented above as C_{XO_r} (26), C_{XX_r} (28), C_{OX_r} (30), and C_{OO_r} (31). Only a few costs are (generically) labeled in Figure 2 to avoid clutter.

Turning attention again to the sinks, we recall from (33) that the set sizes can be fixed by making

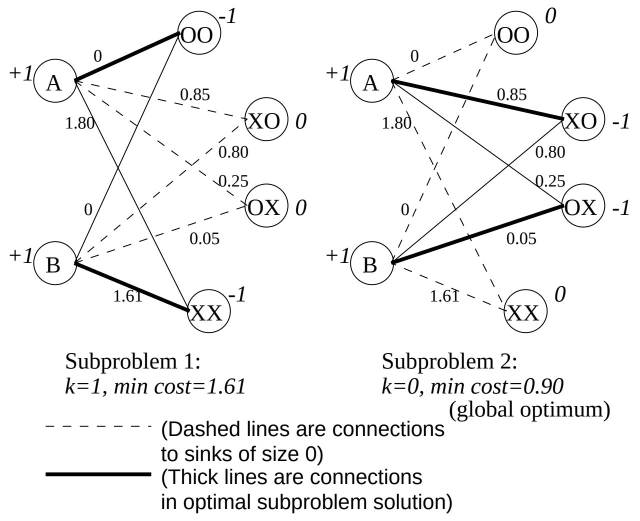

Figure 3: Two-day strategy assignment network corresponding to Table 1

one free parameter choice, say $|\mathcal{OO}| = k, 0 \leq k \leq \alpha$. This determines the sizes of the other strategy groups, which is also the size of the corresponding sinks. For each allowable value of k, there is a single lowest cost "strategy flow"; Once the sink sizes are set, the solution of the subproblem proceeds according to the standard methodology (see [Ber98] for more) for finding minimum-cost flows. Several algorithms are available; all choices (for non-integer flow formulations) are either weakly or strongly polynomial in the number of edges. Each subproblem can be solved in polynomial time (no worse than $O(16d^2 \log 4d)$ [SKPT94]), so solving α such problems (one for each allowed value of k) is no worse than $O(16\alpha d^2 \log 4d)$. The optimal strategy is to choose the k for which the flow cost is smallest. As an example, in Figure 3 we show the networks corresponding to the two subproblems which we solved in forming Table 1.

The problem is (at worst) a series of α minimum-cost flow problems, although in most cases, it is likely that the complexity is much better than stated above. Each subproblem is just that of a transportation problem form d unit sources to 2^n sinks (for n days). The reduced complexity is expected because the solutions to the flow problems should be related. For example, we might start with $k = \alpha$, solve the flow problem, and then back out one element from the $\mathcal{OO}$ group and choose where to reassign it. For instance, a simplex-based solution [Lue84] to any one of the subproblems (the network corresponding to a particular choice of $|\mathcal{OO}| = k$) need not waste time using artificial variables to find an initial basic feasible solution, since these solutions are easily found through other means. Moreover, the better the initial guess, the faster the simplex method will find a solution. Therefore, if we create our initial basic feasible solution for the

$k+1^{\text{th}}$ subproblem from the optimal solution to the k^{th} subproblem, the simplex algorithm could converge more quickly than it might if starting from other initial solutions.

Speculation aside, much work remains to be done on making the method scale in the number of pages, observations, and days. Since the graphs involved will often be very large, our work will include approximation methods for obtaining near-optimal flows. Promising steps have been taken through application of properties like (29)[3], and in grouping documents with similar dynamics into “meta-documents” to reduce the number of nodes in the graph. Approximation will be indispensible in a system where optimality is less of a concern than execution speed.

2.4 Accounting for unequal cost of observation

Our problem statement above includes a number of assumptions; we consider a more general case in which one of these is removed. Specifically, throughout the preceding discussions, we have assumed that α pages could be checked per day, and all at the same rate. While this may be true on average, there is definitely a variation in service times required for the processing of pages. Further, these times can vary dramatically even for a single document. Download and processing times are both proportional to document size, and available bandwidth depends strongly upon the time of day (e.g., one expects long service time around 4:00 PM EST). Real systems might benefit by accounting for this variation both among different documents and for a single document.

2.4.1 Deterministic document retrieval times

We assume that all documents require some constant time to process, but that this time may not be the same for different documents. This requires us to restate our objective, since we can no longer count on a constant number of pages processed per day. Specifically, we wish to discover incorrectly indexed pages as quickly as possible.

To quantify this, we introduce some notation. We would like to determine an optimal ordering of pages to check, $S^* = \{s_1, s_2, \ldots, s_d\}$, such that the expected time t taken to find an incorrectly indexed page is minimized. Each page i has a fixed probability P_i of having been changed. In order to consider P_i constant and calculable from (13), we must create a new list S whenever the probabilities P_i have changed. A time

[3]For example, it is clear that pages with the largest values of C_{XX_r} should be on a “hit-list” for observation sometime in the near future, due to the high cost of leaving them untouched.

T_i is required for processing the i^{th} page. The time t expected to find an incorrectly indexed page can be expressed by a probability-weighted sum of these times. If we assume some ordering S as listed above, then this time can be written:

$$t = P_{s_1}T_{s_1} + (1 - P_{s_1})\left[P_{s_2}T_{s_2} + (1 - P_{s_2})\left[P_{s_3}T_{s_3} + (1 - P_{s_3})\left[\ldots\right]\right]\right] \tag{35}$$

The solution to scheduling problems having this form is worked out by Moizumi in his work on the "traveling agent problem" [Moi98]. In Moizumi's analagous version of the problem, a single mobile agent [Gra97] is tasked with visiting a sequence of machines in search of information. As in our problem, there is a processing time T_i and a probability of success P_i associated with the search for information on each machine. The planning problem is to determine the order in which to visit machines to minimize the time it takes the agent to find the desired information. It is shown that the machines should be visited in decreasing order of P_i/T_i. The proof is based on an exchange argument, in which a criterion is derived to justify the exchange of two machines in the list. If all possible exchanges are performed, then a sorted list of P_i/T_i will result:

$$S^* = \{s_1, s_2, \ldots, s_d\}\ , \text{ where } \frac{P_{s_1}}{T_{s_1}} \geq \frac{P_{s_2}}{T_{s_2}} \geq \cdots \geq \frac{P_{s_d}}{T_{s_d}} \tag{36}$$

This result is intuitively pleasant—we have moved from obtaining some fixed amount of benefit per page, to an expected benefit per unit time. In an economic context, we think of comparing salaries being offered by different employers. Note that this does not necessarily correspond to the maximum income over time, since the probabilities can change. Still, to minimize the time taken to acquire our *next unit* of income, we will always wish to work for the highest salary for as long as we are allowed to do so. This corresponds to the intuitive notion that a page having $T_i = 1$ second and $P_i = 0.9$ would have a payoff rate of 0.9 changed pages observed per second, and would provide the same utility as checking a sequence of two pages both having $T_i = 0.5$ seconds and $P_i = 0.45$. These two could be checked within one second, and if their changes were independent, then the expected changed pages observed per second would also be 0.9. In this formulation, we also assume that there is no reason to prefer a correct index entry for one document over that for another; the value of a correctly indexed page is independent of the page.

Metrics such as that in (24) require a new problem statement, since the discovery of a changed page is valued in proportion to its expected number of days out-of-date. In this case, (35) is rewritten to account for the relative merit of indexing one page versus another. We define B_i to be the value obtained with probability P_i from page i in time T_i. If we assume the benefit B_i for a page is acquired linearly over the time T_i, then the problem can still be stated as a minimization of the expected time it takes to obtain one unit of benefit. The changes to the expected time (35) are cosmetic if we assume that a unit of benefit can be obtained in time T_i/B_i:

$$t = P_{s_1}\frac{T_{s_1}}{B_{s_1}} + (1 - P_{s_1})\left[P_{s_2}\frac{T_{s_2}}{B_{s_2}} + (1 - P_{s_2})\left[P_{s_3}\frac{T_{s_3}}{B_{s_3}} + (1 - P_{s_3})\left[\ldots\right]\right]\right] \tag{37}$$

The solution to this problem is equivalent to that of an auxiliary problem with unit benefits and new times $T_i' = T_i/B_i$. We can substitute the times T_i' into (36) to obtain the optimal ordering

$$S^* = \{s_1, s_2, \ldots, s_d\}, \text{ where } \frac{B_{s_1}P_{s_1}}{T_{s_1}} \geq \frac{B_{s_2}P_{s_2}}{T_{s_2}} \geq \cdots \geq \frac{B_{s_d}P_{s_d}}{T_{s_d}} \tag{38}$$

We can calculate the values to be sorted into lists of the form in (36) and (38). Namely, for page r the lost benefit rate (or lost salary) for that page in terms of expected unobserved changed pages per second is

$$\frac{P_r}{T_r} = \frac{1}{T_r}\left(1 - \left[M_r^{k_r}\right]_{i_r(i_r+k_r)}\right) \tag{39}$$

In terms of incremental improvement in the expected number of days out-of-date, using (15), we have

$$\frac{P_r B_r}{T_r} = \frac{1}{T_r} Z\left(i_r, k_r\right) \tag{40}$$

Moizumi's problem formulation differs from ours in some important ways. First, both our probability of "success" and our processing latency are both strong functions of time. Therefore, planning by the P_i/T_i method will only be valid for time scales on which both the probability and time spent are essentially constant. While this may be an appropriate assumption for timescales on the order of a few minutes, it is certainly not correct when used for longer scales. The tradeoff for ignoring variation in the probability and the retrieval time is that we accept that some error will develop in the ordering as time elapses. After either the probability or the retrieval time has changed significantly, we must re-examine the order in which we had planned to fetch pages. The fact that we might need to re-plan also implies that a one-day greedy planning method is almost certainly not optimal, by the same arguments presented in the construction of the Table 1 example.

The second difference in the two problems is a smaller one, but is still worth noting. In the agent planning problem, the agent is allowed to stop after the information is found. In other words, only one unit of benefit needs to be acquired, and then the task is over. Our problem is different in that we are required to continue even after a success—when a changed page is found, there is certainly no reason to stop looking for more. Instead, our problem is like a sequence of traveling agent problems, in which we must find the optimal sequence of machines to visit so as to maximize incremental benefit. After each success, we re-plan so as to obtain further benefits as soon as possible. If we are already using a sorted list of benefit rates, and those benefit rates are still accurate, then there is nothing new to do after a page is successfully re-indexed. The first page retrieved in the "new" problem would have been the next page fetched in the old problem, since the ones just checked now have zero probability of being out-of-date.

2.4.2 Stochastic retrieval times

Up to this point in our discussion of accounting for variable observation costs, we have been assuming that there is a single, average service time that characterizes each page in the collection. Of course, there is much more information available regarding service time. This time depends upon the available network bandwidth; large and rapid variance in download time is quite common [HL97].

A speedup or slowdown in download times during the execution of an observation schedule would not have a dramatic effect on the optimality of the ordering if the effect was the same for all pages at the time of a single retrieval. That is, if all pages experience a uniform rate change by some fixed multiplicative factor, then the relative order is still the same. To see this, consider that terms of the form of (39) and (40) will have the same ordering if all times are scaled by the same value[4]. Therefore, even though it may take a great deal longer to move through the list, the order in which we would like to fetch pages would be unaffected. It is unclear whether this is a reasonable simplification in practice, but the problem is made much more complex if we cannot assign a single time to each page.

The worst case, as far as complexity is concerned, is that in which we can arbitrarily select the transfer time for a particular document by scheduling it for a particular time of day. Rather than being able to simply order P_i/T_i terms, each document would have an entire distribution of costs associated with it, since the order in which the documents are fetched affects the processing time distributions associated with them. Cases discussed up to this point did not have this property; the benefit obtained by retrieving one document

[4]A uniform processing rate change is equivalent to a change in time units.

did not affect that which could be obtained by observing another. While this problem may be discussed in the thesis, it is unlikely that any methods based on a distribution of values of T_i would be acceptable in practice for anything but the smallest domains, such as non-WWW observation problems.

2.4.3 Difficulties due to variation in retrieval times

Using the methods suggested above for pages with non-constant retrieval times will result in indexing preference being given to documents that are either closer to the database or smaller in size (both make T_r small). That is, we select between two documents with identical change probability based upon either how far away they are (if of identical size) or how large they are (if at the same location). If we truly value only the expected number of current pages or some other metric that does not account for how the entire collection is treated, then this is not a problem. Intuitively, though, an index should not assign preference based strictly upon convenience of indexing. Methods are needed for removing unwanted bias against documents that are either larger or farther away.

If this bias is not removed, then liveness conditions might not be met in our formulations to this point. In the case in which all the retrieval times are the same (namely, $1/\alpha$), we had assurance that every document would be examined, because all pages would eventually reach a stage in which they would have been changed with probability 1. Sorting by this value guaranteed that all pages would eventually be revisited. However, when we divide through by retrieval time, even a page that was absolutely guaranteed to have changed since the last observation might not ever be fetched if the time required to examine it was too large. This could be due to the document being large, or being at a great distance. If we are only able to observe from one location, then we require weighting parameters to remove the bias against large or distant documents.

2.5 Reducing observation costs using mobile agents

These two biases naturally lead into two targeted solutions whereby we might make limited use of remote sites. Up until this point, we have assumed that there were few things we could do to reduce the price of making a particular observation. The best that could be done was to make observations at different times of day, by which we can do a fair job of selecting the observation cost. We are still inclined to remove the biases mentioned above, and two approaches seem natural. It is critical that observation time be reduced. The two sources of this time are network latency and document size. In this section, we discuss how these difficulties can be ameliorated by moving the observer closer to the data and using encoding schemes so as to enable more frequent observation of large pages.

To even consider use of a remote observation post, there must be remote machines available for our use. We may not have significant privileges on these machines, but even a very limited use, such as making a single observation from a remote machine, could be helpful. Mobile agents [Gra97], or autonomous programs that can migrate under their own control, are a means by which such limited access might be granted. If we have this access, then we can choose to make observations from machines based upon whether or not it benefits us to do so. By adding mobility to the observer, we give it the freedom to make more observations in the same time (being in closer proximity to the resource) and the chance to perform pre-filtering on the result. We emphasize that there is no need to use an agent solution if full access to a machine exists; the idea of distributed web robots is already in use (e.g., [AL98]). If we have full access to a machine, the only reason not to have a search robot permanently resident on the machine would be the overhead involved in merging results from distributed observers.

In particular, we seek solutions that remove the bias against large or distant documents. To accomplish this, the simplest type of remote observer might migrate to the vicinity of a document (or collection) of interest, compress the documents and send them back to the home machine, where they would be decompressed and analyzed. If the remote machines are at a great distance, this could result in a significant time savings. More complicated agents might transmit only the changes in page state in some compressed form. This would be especially appropriate for large pages that only experienced minor changes. Being able to transmit changes in state this way is a large step towards the use of "delta encoding" (analogous to `MPEG`) schemes for `HTTP` transmissions [MDFK97]. The key to making this type of encoding work is to package the agent with knowledge of the previous state of the page. Then, when a change is observed, the agent need only transmit the change in the page's index entry, not the entire page or even the entire index entry. This scheme has the desired features of reducing network traffic as well as removing bias against large or distant pages.

As packaging agents with an index entry and a lookup mechanism might produce a rather large piece of code, a simpler filter would probably be a better candidate for a remote observer. Large routines such as compression algorithms might be made available as part of standard libraries on remote machines, obviating the need to carry compression code from machine to machine. The proxy server agent could be modified so that it would only return requested pages if they did not match a previously hashed version of the page,

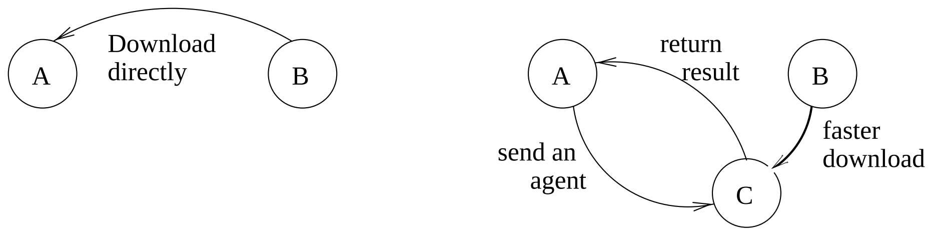

Figure 4: Remote observation

carried with the agent. This agent could observe these documents more frequently (being closer to the resource of interest) and then send pages back to the server (compressed, if utilities are available to the agent) only if they had changed. A typical retrieval task would entail sending an agent with instructions to look at some set of documents and return compressed versions of those that do not match a hash carried by the agent. Mobility proves useful in that an agent can move to a better vantage point prior to making each subset of its observations.

The idea of distributing observers which then report back to a central dispatcher is one step towards making the indexing process less of a "pull" technology and more of a "push" one. Formally, this difference can be seen in the decreased symmetry of messages between the agent and the indexing system, as discussed in [CB98]. A single monitoring agent could return an arbitrary number of observation results to its point of origin. Clearly, dispatching such an agent to perform observations results in a large observation cost savings from the perspective of the central indexing system, but it is not yet clear what form this savings will take.

While mobility may be a valuable option, we need to know the relative merit of observing locally versus using a mobile agent on a remote machine. Consider three machines, A, B, and C, as shown in Figure 4. Machine A contains the main document index database, machine B contains a document of interest, and machine C is available for use to a mobile agent. Note that this is the only mode in which machine C can be used; for whatever reason, we are not allowed to compile and install code there on a permanent basis. We wish to determine what observation scenarios favor the use of a mobile observer in this situation. To do this, consider a comparison of the two scenarios shown in Figure 4; specifically, compare the time it takes to transfer a document directly from B to A versus the time it takes to send a remote observer to C, observe the document at B and return relevant information. More precisely, the document has size S_D bytes. The link from machine i to j has latency L_{ij} seconds, and an effective transfer rate of N_{ij} seconds per byte. k

observations made directly from machine A would take time

$$t_1 = k\left(2L_{AB} + N_{AB}S_D\right) \tag{41}$$

The request for the document is assumed to be small enough that the time taken to transfer it is essentially the same as the link latency. Half of the latency term is due to this request, and half is due to the response. We compare t_1 with the time t_2 taken to perform the same k observations using a mobile agent of size $S_A = \beta S_D$ transferred to machine C. Further, we assume that the agent is clever enough to compress the document to a fraction η of its original size before transmitting ηS_D bytes of index information back to A. The total time is then

$$t_2 = \left(L_{AC} + N_{AC}\beta S_D + C_{startup}\right) + k\left[\left(2L_{CB} + N_{CB}S_D\right) + \gamma\left(L_{AC} + N_{AC}\eta S_D\right)\right] \tag{42}$$

To initiate the agent on the remote machine takes a time $C_{startup}$. For simple agents, this should be the only computation time on the same order as the transfer times. Other computation times, such as compression/decompression times, are assumed negligible by comparison. This might be accomplished through some form of run-length encoding, so that the compression and decompression times partly overlap the transmission time. Also, notice that the agent's messages to the home machine are only required for the fraction γ of the observations on which a changed page is observed, since no communication is necessary if a page has not changed. It is immediately clear that the download portion of t_2, namely the term $k\left(2L_{CB} + N_{CB}S_D\right)$, must be strictly less than t_1 to even consider the use of a remote observer. If this is the case, then the sum of the outer terms in (42) must be less than the savings in download time for it to be worthwhile to observe remotely. This is a restatement of the obvious fact that remote observation is a good option when we save more by downloading at the alternate site than we spend in sending an agent and returning results. The times t_1 and t_2 can be estimated in advance to determine the relative benefit of the two modes of observation.

To clarify the relation between the times, assume the latency terms L_{ij} and the startup cost are zero. Ignoring the latency terms is reasonable: when the latency term is large, it indicates that the first packet will take a long time to arrive and that all subsequent packets will probably be of similar speed. Since transmission is a two-way procedure (requiring acknowledgement packets as well as trasmitted ones), the N_{ij} term has tends to be large when L_{ij} is large. Particularly for large messages, it is likely that the $N_{ij}S_D$

terms will dominate transfer times. The startup time on the remote machine is harder to ignore, although as the number of observations becomes larger, this time becomes less of an issue. After removing these terms, normalize by document size S_D, and divide through by k to obtain a requirement on the normalized time per observation, as a function of k.

$$1 > \frac{N_{AC}}{N_{AB}} \left(\frac{\beta}{k} + \gamma\eta \right) + \frac{N_{CB}}{N_{AB}} \tag{43}$$

Note the behavior of this average time as k becomes large. As would be expected, the portion of the first term due to the initial transmission of the agent disappears, and we are left with two terms. One term gives the relative speed of remote observation as compared to standard observation. The other term describes the compression achieved by sending fewer messages and reducing their size. The sum of these two normalized times must be less than unity in order to make a remote observation worthwhile.

2.6 Multiple tasks, filtering, and change assessment

This was a relatively simple comparison, but we state it to emphasize that agents can present advantages even in the most basic usage. However, an observation agent can be given tasks that are arbitrarily complex. For example, it need not perform observations of only a single page, and it can be free to move to a better vantage point if this saves time. In fact, the agent will multiply its efficiency if it observes many documents that might be resident on the target machine, B. An agent that is to observe multiple documents will be slightly larger than one tasked with observing a single page, since it will be required to carry some compressed version of the state of target documents as well as extra observation instructions. Still, the compressed version might be quite small, perhaps only a few integers worth of hash values per page. Although this added weight will increase the time to dispatch the remote observer, the extra time is a relatively small, one-time cost. Multiple observations of multiple documents will serve to amortize this cost over time. Additionally, as an agent completes observations, it can "diet" by dumping data corresponding to completed tasks, whereupon it can migrate a bit more quickly to the next observation site.

Whether observed by an remote agent, or by a robot running on the home machine, there must be a well-defined means by which to determine whether or not a page has "changed." We have been rather slippery about avoiding explanation of what might be meant by this, so as allow for more general types of observation. If one defines a change in the strict sense of whether or not any bytes were altered, this may be problematic. There will be situations in which the object in question has certainly changed, but the change that occurred was insignificant: unimportant changes would include the "counter" images on some webpages, randomized advertisements chosen for display, extra whitespace in the `HTML` source, and anything else essentially unrelated to the content of a document. We emphasize that a document's content is user-specific: it is always necessary to assess what portion of a page's content is of interest. For example, certain robots (users, in their own right) may be tasked with looking for new links. Changed pages that do not have new links are then no longer of interest, and are therefore not considered to have been altered. Simple filtering tasks such as this could be carried out to determine if a document's change is of interest.

In this light, a change is best defined not just by changing page content, but rather by an alteration in the projection of page content into the subspace of user needs and interests. This forces us to attempt to divide page content into "content-related" and "non-content-related" subspaces. Such subspaces could be very difficult to characterize, requiring some analysis of page content and of what it was about the page that changed. Encapsulated in this is the need to recognize things such as advertisements and any content that varies somewhat randomly. Another approach to this would be setting a threshold on how large a change must be before it is considered noteworthy. This could be as simple as choosing a percentage change, and monitoring the number of bytes in the page. If it ever moved outside of this range, then a "change" would have occurred. The threshold would then be reset, and the process would continue.

Accurate projections onto user needs will probably only become more important as the content of web pages becomes more dynamic. It is quite likely that in the future no two documents obtained from the same URL will be entirely "alike." That is, content will be customized to such a degree that documents will never remain the same for more than one viewing. The basic question is, what portion of a page is drawing the users to examine it? In other words, a change is important if it occurs in that portion of the content which draws users to examine the page. Changes that are not part of user's search criteria are irrelevant and should be ignored. Obviously, separating the important changes from the unimportant ones is quite tricky, requiring us to classify changes topically, stylistically, syntactically, and so on. The real benefit to the user is not to have an index that is 100% correct, but rather a current index of relevant topics only. In this light, if a change is important to anyone, then it is worth noting, but the more users that benefit, the greater the priority.

Dates	Task
10-12/1998	Initial theoretical work and proposal
1-2/1999	Expand analytical models, to include multi-step formulation of non-constant retrieval time.
1-3/1999	Simulation and performance analysis of algorithms in `MATLAB`
3-4/1999	Program web robot database interface (parser already exists)
3-4/1999	Program web page retrieval module (to be built upon the `wget` engine).
4/1999	Debug retrieval engine
4/1999-1/2000	Robot startup and continuous operation
4/1999	Run robot with uniform indexing strategy for initial month
6-7/1999	Continuous-time problem formulations, and alternative (non-WWW) applications
5/1999	(Optional, but desirable) Program CGI interface to the search database.
7-9/1999	Further expansion to include alternative state definitions and dynamic collection metrics.
6/1999	Experiment with remote observation via transportable agents—coding and testing stage.
7/1999	Test remote observation methods
8/1999	Consider/develop remote observation via meta-search (i.e., use of existing search tools as remote tools
9-1999	Robot performance analysis, benchmarked against Dartmouth's own search indexing system and external engines.
9-10/1999	Desired new features can be integrated into web robot; different coverage area.
10/1999-3/2000	Ongoing robot performance analysis
1-3/2000	Pursue something novel and inspirational, depending upon what looks interesting and worthwhile.
3-5/2000	Complete written thesis
5/2000	Final editing
5/2000	Present thesis

Table 2: Proposed schedule of future work

3 Schedule of future work

Many possibilities have been presented, making it important to know what results will indicate that the thesis is complete. In this section, we discuss the work that will be conducted over the next year and a half to set forth a contract of sorts. Our goals in the thesis work are twofold: we would like to develop the theory of optimal observation, and show how it can be implemented in practical applications. A preliminary schedule for achieving these goals is shown in Table 2.

Following further development of the theory, simulations will be written in `MATLAB` to test and compare the algorithms to be used in the operation of an optimal web robot. We then plan to use these algorithms to optimize actual WWW observation. Code has already been written or obtained for the main features of the robot, namely `HTTP` page retrieval and parsing. The code yet to be written includes a simple indexing scheme, an interface to our Informix database server, and perhaps even an external search interface, although these are not likely to be critical in the earliest implementations. A number of tests are planned, the largest of which involves running a web robot on some limited domain (most likely pages within `*.Dartmouth.edu`).

Robots will be benchmarked against existing engines, such as Dartmouth's own engine[5] and also external commercial engines. We will monitor the "efficiency" of the engine, by keeping tabs on the results of our observations. In general, an observation scheme should show an increase in the number of changed pages discovered per observation, although we note that this could be increased simply by not observing for a very long time. Nonetheless, if this is not observed, then existing models of page behavior will probably need to be revised. Some limited comparisons showing the merits of remote observation are planned. These will involve the construction of observation agents to collect data remotely. If possible, this will proceed via a comparison of the different methods of monitoring a remote domain (where we are allowed to run agents) both locally and from remote observation outposts.

Beyond practical implementations, there is a great deal of mathematical study yet to be done, so as to make the theory more general. One of the most promising directions is the movement of the theory into continuous-time domains. For example, equations like (6) can be generalized to arbitrarily fine quantizations of time, whereupon they approach some continuous analog. The system described to motivate time-dependence was a very simple continuous system that we would like to expand upon. Systems like this have been explored in advanced statistics, in the context of "renewal processes" [Fel71]. In exploring this theory, we seek to generalize the discrete versions into continuous ones. When a continuous theory is adequately developed, it will be possible to investigate other real-world applications.

It is likely that the theory may already have been considered in other domains, such as in the decision of how fine a mesh is needed in a finite-element application (spatial observation), or in the choice of sampling rates in A/D systems that must monitor multiple channels. For this reason, an extensive exploration of related work will be take place throughout the development of the thesis.

[5]http://ultraseek.dartmouth.edu:8765/

4 Conclusion

The "information age" is defined by our having access to unprecedented quantities of information, so vast in fact that we are forced to intelligently allocate observation resources. The first steps have been taken in the development of a theory of optimal observation, as we begin investigation of a means by which we can optimize monitoring of large collections of changing objects. In particular, a means has been presented by which to streamline the observation and maintenance of a collection of WWW documents. Specifically, we have suggested a Markovian model, describing the dependence of change probability on age. This model was used to construct two types of cost metrics, both of which are based upon being able to calculate the probability of a page having been changed after some time has elapsed since the last observation. Thus far, we have assumed a static collection with known document-change statistics. In the coming year and a half, this theory will be implemented, and expanded into more realistic scenarios involving non-constant retrieval times and time-varying document collections. In the thesis work to come, we hope not only to extend the theory into this domain, but also to generalize it so that it can be applied to observation of any object collection.

References

[AL98] Anders Ards and Sigfrid Lundberg. Regional distributed WWW search and indexing service — the DESIRE way. In *Proceedings of the Seventh World Wide Web conference*, April 1998. Available from http://decweb.ethz.ch/WWW7/1900/com1900.htm, also in Computer Networks and ISDN systems, Vol. 30, issues 1-7 article FP17.

[Ber87] D. M. Bertsekas. *Dynamic Programming.* Prentice Hall, 1987.

[Ber98] D. M. Bertsekas. *Network Optimization: Continuous and Discrete Models.* Athena Scientific, 1998.

[CB98] George Cybenko and Brian Brewington. The foundations of information push and pull. In D. O'Leary, editor, *Mathematics of Information.* Springer–Verlag, 1998. To appear; available from http://www.dartmouth.edu/~gvc/pp.ps.

[CBB$^+$97] George V. Cybenko, Aditya Bhasin, Brian Brewington, Robert Gray, Katsuhiro Moizumi, and Kartik Raghavan. The Shannon Machine: A system for networked communication and computation. available via anonymous ftp://witness.dartmouth.edu/pub/shannon.ps, 1997.

[CCPS98] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization.* Wiley-Interscience, 1998.

[CLW97] E. G. Coffman, Z. Liu, and R. R. Weber. Optimal robot scheduling for web search engines. *Journal of Scheduling*, 1997. available at http://www.inria.fr/mistral/personnel/Zhen.Liu/Papers/.

[Fel68] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 3rd edition, 1968.

[Fel71] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 2. Wiley, 2nd edition, 1971.

[Gra97] Robert Gray. *Agent Tcl: A flexible and secure mobile-agent system.* PhD thesis, Dept. of Computer Science, Dartmouth College, June 1997. Available as Dartmouth Computer Science Technical Report TR98-327.

[HL97] Bernardo A. Huberman and Rajan M. Lukose. Social dilemmas and internet congestion. *Science*, 277:535–537, July 25 1997.

[How60] R. Howard. *Dynamic Programming and Markov Processes.* Jointly published by MIT Technology Press and Wiley, 1960.

[HS98] J. P. Hardwick and Q. F. Stout. Flexible algorithms for creating and analyzing adaptive sampling procedures, 1998. To appear. Also see http://www.eecs.umich.edu/~qstout/abs/Seattle97.html.

[LG98] S. Lawrence and C. L. Giles. Searching the World Wide Web. *Science*, 28:98–100, April 1998. Available by request at http://www.neci.nj.nec.com/homepages/lawrence/.

[Lue84] D. G. Luenberger. *Linear and Nonlinear Programming.* Addison Wesley, 2nd edition, 1984.

[MDFK97] J. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy. Potential benefits of delta-encoding and data compression for HTTP. In *Proceedings of ACM*

SIGCOMM'97 Conference, pages 181–194, September 1997. Available from http://www.research.digital.com/wrl/techreports/abstracts/97.4.html.

[Moi98] Katsuhiro Moizumi. *The mobile agent planning problem.* PhD thesis, Thayer School of Engineering, Dartmouth College, November 1998.

[SKPT94] Clifford Stein, Philip Klein, Serge Plotkin, and Eva Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM J. Computing*, 23:466–487, 1994. Available from http://www.cs.dartmouth.edu/˜cliff/papers/MultiFlowUnit.ps.Z.