Information Mediation in the Presence of Constraints and Uncertainties

Sandeep Pandey

May 2008
CMU-CS-08-125

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Christopher Olston, Chair
Christos Faloutsos
Geoffrey J. Gordon
Andrew Tomkins

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

© 2008 Sandeep Pandey

The views and conclusions contained in this document are those of the author, and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government, or any other entity.

Keywords: information mediator; search engine; constrained optimization; uncertainty; exploration/exploitation tradeoff; web crawling; web page ranking; advertisement ranking; web page discovery; set cover; greedy; synchronization; web page refreshing; randomized ranking; entrenchment effect; rank promotion; multi-armed bandit; click-through rate; BMMP

This thesis is dedicated to

my sister Sushma,
who has been an inspiration to me for her strength in face of overwhelming adversity,

and my parents,
whose selfless love and support, words can't describe.

Abstract

People often require a unified view of multiple disparate information sources. This role is played by *information mediators* such as Web search engines and database integration systems. Internally, information mediators perform three basic tasks: acquisition, analysis, and presentation of content. For example, Web search engines acquire Web pages via crawling, analyze the text and link structure of the crawled pages, and present lists of pages in response to user search queries.

In environments such as the Web that have very large amounts of content, the content acquisition and presentation tasks can be viewed as constrained optimization problems. On the acquisition side, the resources required to discover, and maintain synchronization with, all available online content vastly exceed the capacity of even the most well-provisioned information mediators. On the presentation side, the constraint is on user attention: users cannot be expected to sift through large amounts of content. Rather than being exhaustive, an information mediator must be selective in acquiring and presenting content, with selections driven by some meaningful objective function. This dissertation studies formulations of, and solutions to, these optimization problems.

A complication that arises in this setting is that some parameters needed to solve the optimization problem are unknown. For example, in the Web search context, due to autonomy of sources a search engine may not know the update rate of pages, making it difficult to conduct an optimal synchronization strategy. Similarly, due to sparsity of user feedback, the search engine may not have accurate page quality measurements, making it difficult to present search results in the optimal order. This dissertation studies means of simultaneously estimating unknown parameters and exploiting current estimates. We focus specifically on the Web search context, although many of our ideas apply to other contexts as well.

Thesis Committee

Christopher Olston (Chair)
Computer Science Department
Carnegie Mellon University

Christos Faloutsos
Computer Science Department
Carnegie Mellon University

Geoffrey J. Gordon
Machine Learning Department
Carnegie Mellon University

Andrew Tomkins
Search Department
Yahoo!

Acknowledgments

I am thankful to many people for helping me during my graduate studies. First of all, I would like to thank my advisor, Chris Olston. Chris is an excellent advisor and researcher who always went above and beyond the call of duty in supporting and guiding me. He spent countless hours helping me improve my research, writing and presentation skills. His ability of thinking clearly and paying attention to small details amazes me. Thinking back, I feel he epitomizes the skills that I lacked the most when I started working under him five years ago, and so this Ph.D., as a learning experience, couldn't have been any better.

I am thankful to my thesis committee members, Christos Faloutsos, Geoff J. Gordon and Andrew Tomkins for finding time to read this dissertation and for giving their invaluable feedback. A big thanks to Sharon Burks, Deborah A. Cavlovich and Frank Pfenning for being so kind and patient in dealing with my "unusual situation." I would like to thank all my collaborators from CMU, IIT, UCLA and Yahoo!: Deepak Agarwal, Deepayan Chakrabarti, Soumen Chakrabarti, Junghoo Cho, Anirban Dasgupta, Kedar Dhamdhere, Arpita Ghosh, Vanja Josifovski, Ravi Kumar, and Sourashis Roy.

Finally, I would like to thank my family, without whose love and support, none of this would have been possible. One life would simply not be enough to pay my gratitude to them. A big thanks to my dear friend Shruti for her unwavering faith in me. She was always there to encourage me when I doubted myself.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Online content is spread across a large number of disparate sources. People often want to have a unified view of content. As a result, there is much interest in *information mediators* which collect content from multiple sources and make it accessible to users in a uniform way. Examples of such applications include Web search engines and database integration systems. There are two major design choices involved in building an information mediator:

- *Content shipping vs. Request shipping*[1] : In the content shipping model the information mediator maintains a *cache* into which content is downloaded from content sources. User content requests are answered using the local cache afterward.[2] In the request shipping model, when a user submits a content request, it is dispatched to the content sources. Each source produces a partial result using its local content, and all these partial results are then merged by the mediator to output the final result.

- *Pull vs. Push* : The communication of data between the information mediator and the content sources can be either pull-based or push-based. In the pull-based model the information mediator requests the required content from the sources explicitly, while in the push-based model the sources automatically send the required content to the information mediator.

[1] This choice is also known as data warehouse vs. mediation in the database literature [114, 115].

[2] The content shipping model does not restrain from employing centralized or distributed model of computing. For example, the local cache can be stored at one location or distributed among several self-created peer locations. Similarly, the processing of content requests can also be centralized or distributed [11, 87, 103].

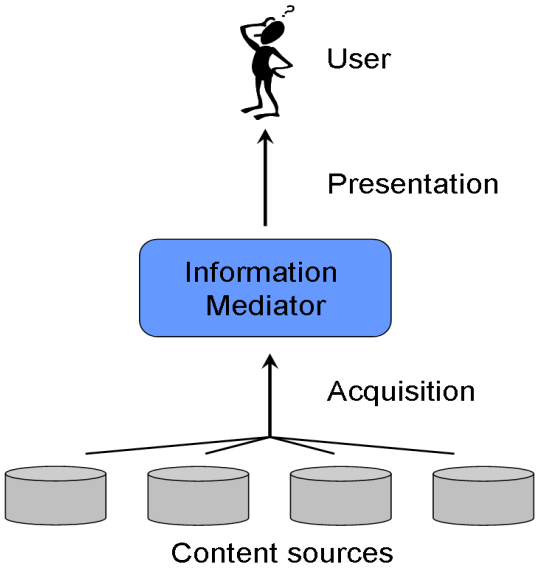

Figure 1.1: *Basic <content-shipping, pull>-based information mediator.*

This dissertation specifically focuses on Web-based content sources, *e.g.*, Web sites, Web databases. Since Web sources are largely pull-oriented (*e.g.*, follow the HTTP protocol) and have limited capability to process user content requests on behalf of information mediators, most of the information mediators in the Web context fall under *<content-shipping, pull>* architecture design. Figure 1.1 shows a *<content-shipping, pull>*-based information mediator (explained in detail later). One example of an environment that follows this design is search engines, which download Web pages from the Web and acquire advertisements from advertisers, and store these pages/ads into a local cache to answer user search queries, *e.g.*, Google [46], MSN [78], Yahoo! [120].[3] Another example is product review/recommendation providers which collect user reviews scattered on different Web sites on the Web and process them locally to present a unified view to the interested users, *e.g.*, Wize [118], Consumer Search [30]. As a third example, many Web-based continuous query (CQ) processing systems proposed in the literature are based on this architecture, *e.g.*, CONQUER [70], Niagara [79]. However, there are other scenarios in which the involved content sources are more cooperative and other architecture designs are employed, *i.e.*, file sharing [41, 45], Web databases [47, 48, 58].

[3]Strictly speaking, search engines are not entirely *<content-shipping, pull>*-based since they deal with sponsored advertisements using the *<content-shipping, push>*-based architecture model, *i.e.*, while search engines need to download pages from Web sites, advertisements are submitted to search engines by advertisers themselves.

Since <*content-shipping, pull*> is the predominant architecture used among Web-based information mediators, we focus on this architecture henceforth. However, many of our ideas also generalize to other architectures. In the <*content-shipping, pull*> architecture, the mediator maintains a local cache, called a *repository*, into which content is downloaded from Web sites. Then the mediator analyzes the text and link structure of the acquired content to facilitate answering user content requests. When a user submits a content request, the mediator returns the content items "useful" for the request. Hence, there are two ways in which Web-based information mediators interact with external agents: (a) acquisition of content and (b) presentation of content. We discuss them in detail next.

1.1 Acquisition of Content

The mediator polls Web sites to download content into its repository. However, Web sites update their content every now and then. Furthermore, some Web sites disappear from the Web over time while new Web sites appear. Answering user content requests correctly requires that the information mediator's repository is kept both *fresh* and *complete* with respect to Web sites. The freshness of repository depends on how up-to-date its content is, while the completeness depends on the fraction of Web content that has been downloaded.[4] Keeping the repository fresh and complete involves (a) *discovering* new Web pages and (b) continuously *synchronizing* with the known live Web pages. Since Web pages link to each other, discovery is achieved by polling known Web pages and following their outlinks. Synchronization also requires polling known Web pages in order to download their (possibly updated) content. The main question is how often to poll each page.

A simple scheme is to poll each Web page with the same period, *uniform polling*. In practice, the rate of polling is bounded, for various reasons, *e.g.*, politeness constraints enforced by Web sites, [5] limited network bandwidth and limited computation resources at the mediator. In the presence of a bounded polling rate and a large number of Web pages, uniform polling results in a very infrequent polling of Web pages. For example, given 10 billion Web pages [49] and 100 nodes with each node polling at the rate of 100 pages per second [24, 52] in parallel, uniform polling scheme polls each

[4] In practice, some undesirable pages (*e.g.*, so-called *spam* pages) may be intentionally omitted.

[5] A politeness constraint enforced by a Web site limits the number of times the Web site can be probed in a given time period by an external agent. Typically, Web sites impose these constraints to avoid getting large traffic from automated engines, *e.g.*, mediators.

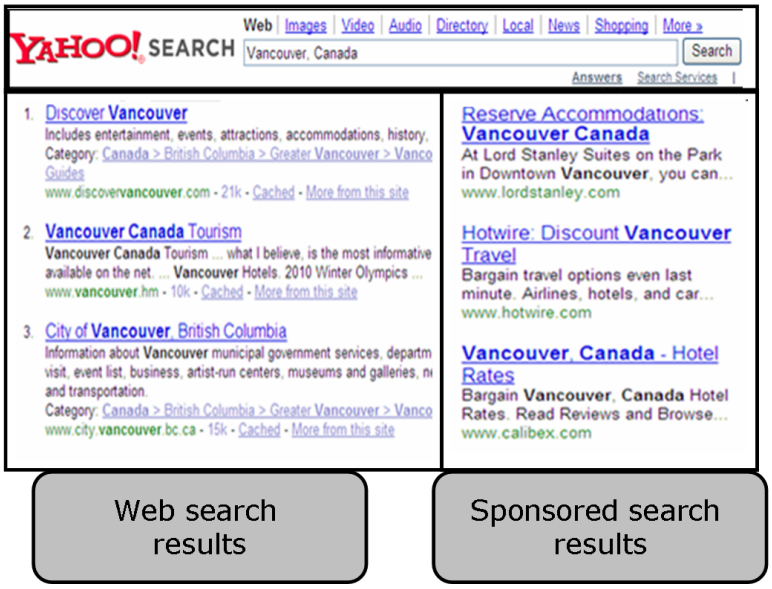

Figure 1.2: *Response for a search query on Yahoo! search.*

Web page once in every 12 days. On the other hand, Web pages are updated at a wide range of timescales [24], *e.g.*, some Web pages are updated on an hourly basis, while other Web pages are not updated for a long time (over 3 months). Hence, non-uniform polling is preferred.

Content acquisition is a constrained optimization problem. The constraint is limited resources available for polling pages. The objective is to maximize the freshness and completeness of the information mediator's repository (defined formally in Chapters 3 and 4).

1.2 Presentation of Content

In response to a user content request, the mediator searches its local repository and returns the content items deemed "useful" for the request. For example, search engines attempt to return relevant, high quality Web pages and advertisements for a user search query, as shown in Figure 1.2. In many cases the number of useful content items can be too large for users to read through. Moreover, Figure 1.3 shows that the likelihood of a user viewing a result item decreases with the item's position in the ranked list, as observed in [69]. Hence, it is important to present the most

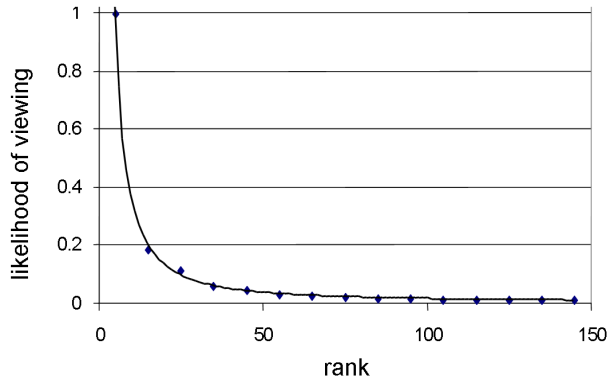

Figure 1.3: *Likelihood of viewing a result item as a function of its rank.*

useful results at the top of the list.

Similar to acquisition, content presentation is also a constrained optimization problem. In this case the constraint is limited user attention. The objective is to maximize the cumulative usefulness of results items as perceived by users (defined formally in Chapters 5 and 6).

1.3 Offline and Online Settings

We have described how the acquisition and presentation tasks can be formulated as constrained optimization problems. A complication that arises is that some parameters needed to solve the optimization problems are unknown. For example, the update rate of pages may not be known to the information mediator beforehand, making it difficult to conduct an optimal polling scheme. Similarly, the relevance of Web pages (or advertisements) may not be known in advance, making it difficult to present search results in the optimal order.

We study the acquisition and presentation tasks in two different scenarios:

- *Offline scenario*: all parameters relevant to the optimization problem (*e.g.*, rate of link creation or page relevance) are given.
- *Online scenario*: some characteristics are not fully known at the outset, but they can be estimated over time. For example, the rate at which a Web page is updated or is linked

to new Web pages can be estimated by polling the Web page a few times and observing its content. Similarly, the relevance of a Web page (or a product review) can be estimated by showing the page to users and observing their feedback, *e.g.*, how many users click on the page [61] or create links to it, how much time they spend reading it.

While the online scenario is more realistic in many cases, it is instructive to study the offline scenario as well because it is often simpler. Observe that in the online scenario, estimating an unknown parameter requires spending resources (*e.g.*, polling resources in the acquisition task) which are constrained. Hence, while allocating resources there is an *exploration/exploitation* tradeoff involved where exploration is required to estimate the unknown parameters, while exploitation is required to make use of the currently available parameter estimates to maximize the objective in hand.

1.4 Organization of Dissertation

This dissertation studies the content acquisition and presentation tasks in the offline and online settings. For convenience, we address these problems specifically in the context of search engines, although some of our ideas apply more broadly. The basic architecture of a search engine is shown in Figure 1.4. As depicted in the figure, the search engine's local repository consists of Web pages and advertisements. While Web pages are acquired by polling Web sites and dealt using the <content-shipping, pull> architecture, advertisements are submitted by advertisers themselves and thus handled using the <content-shipping, push> architecture. Since this dissertation focuses on the former architecture, we study the acquisition of Web pages only (and ignore the acquisition of ads). We study the presentation task of both pages and ads though, since the underlying approach is the same in both cases.

A complete organization of this dissertation is given in Figure 1.5. We begin by focusing on the acquisition task. In particular, we address the problem of discovering new Web pages in Chapter 3. In Chapter 4 we study how to synchronize a search engine's repository with known Web pages. Then we turn our attention to the presentation task. We focus on the problem of ranking Web pages in Chapter 5 and lastly in Chapter 6 we study how to rank ads, as shown in Figure 1.2, in response to user search queries.

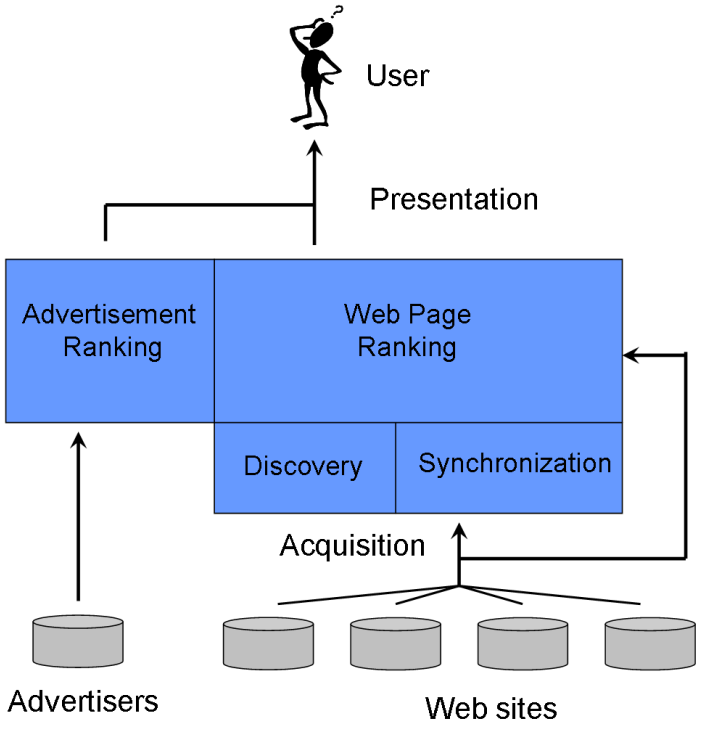

Figure 1.4: *Basic search engine architecture.*

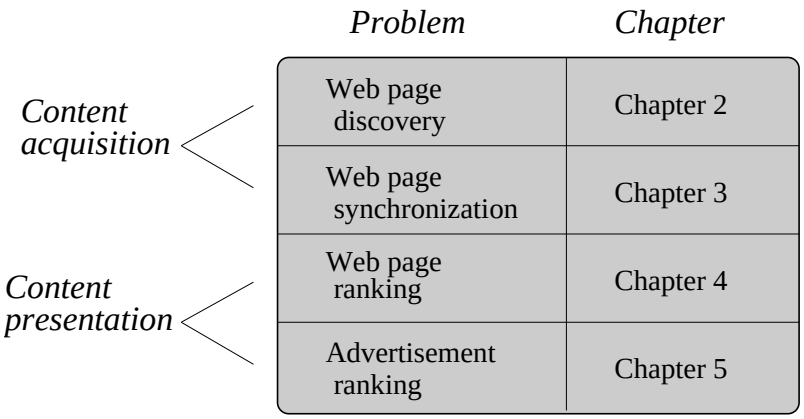

Figure 1.5: *Dissertation organization.*

Chapter 2

Related Work

In this chapter we discuss previous work that is broadly related to this dissertation; more specific related work is discussed in the relevant chapters. We divide the related work into the following categories: (a) architecture design of information mediators (Section 2.1), (b) content acquisition (Section 2.2), and (c) content presentation (Section 2.3).

2.1 Architecture Design

The problem of providing centralized access to distributed content has been well studied in the database community. The primary architecture designs that have been considered are *data warehousing* [114] and *mediation* [115] which we refer to as content-shipping and request-shipping, respectively. Their main focus has been on semantically integrating heterogeneous content, *i.e.*, translating the content provided by different sources which adhere to different content schemas, into a single schema [22, 86, 115]. In the Web search context, a tight integration of content is infeasible because Web content is largely unstructured and does not adhere to any specific schema. Hence, Web-based information mediators, *e.g.*, search engines, typically work by operating on content that has not been semantically integrated.

Many search engines have been developed in in the past, but for proprietary reasons there is not much detail that is available about them in the public domain. A basic search engine architecture has been described in [15, 20, 52]. In [15] a preliminary design of Google search engine is described.

While a large scale search engine is a complex system consisting of several modules, for illustration we can think of it consisting of three main modules: *crawler*, *indexer* and *searcher*. The crawler module is responsible for downloading pages from the Web and storing them into a local repository. The indexer module reads pages from the repository and creates an *inverted index*.[1] Given a word the inverted index returns the list of pages containing the word. The indexer module also extracts links from pages and indexes anchor text. Given a search query the searcher module uses the inverted index to retrieve and rank the pages relevant for the query.

Since we study the acquisition and presentation of content, and not the local computation that is performed at a search engine, our main focus is on the crawler and searcher modules. While deferring the discussion of searcher to Chapter 2.3, we discuss the related work on crawler next. A comprehensive description of a Web crawler, called Mercator, is given in [52]. The Internet Archive crawler is described in [18, 104]. Typically Web-scale crawlers download hundreds to thousands of pages per second. Achieving this downloading rate requires a high degree of parallelization whereby crawlers run on multiple machines and download multiple pages in parallel. The design of parallel crawlers and the challenges faced therein are discussed in [20, 24].

The basic principle of a crawler is as follows: it starts with an initial set of URLs. It downloads these pages and stores them into a local repository. By extracting the outlinks of the downloaded pages, the crawler discovers new URLs which are also downloaded and added to the repository. Then, the crawler goes out again to redownload the repository pages, and this cycle goes on endlessly. Depending on how the crawler redownloads the pages of the repository, two contrasting crawler designs have been proposed, called the *batch* and *incremental* crawler [26]. A batch crawler operates in cycles where it redownloads each page of the repository in each cycle. Between two successive cycles the crawler is "rested" for some time period, during which time the pages downloaded in the previous cycle are incorporated into the repository. On the other hand, an incremental crawler runs continuously. It prioritizes the redownloading of repository pages so that it redownloads more important pages more frequently, unlike the batch crawler which treats each repository page equally. The incremental crawler incorporates Web pages into the repository as soon as they are (re)downloaded, *i.e.*, *in-place updating* mechanism. As pointed out in [26], in principle, an incremental crawler can be more effective than a batch crawler since it has the freedom of prioritizing the redownloading of pages unlike the batch crawler. Of course, the performance improvement would

[1] An inverted index maps each word to the list of pages in which the word occurs. It is typically created by inverting the *forward index*, *i.e.*, index which maps each page to the list of words that occur in it.

critically depend on how well the incremental crawler performs the task of redownload prioritization which is the focus of Chapters 3 and 4.

2.2 Content Acquisition

As discussed in Chapter 1, acquiring content involves (a) discovering new content sources and (b) synchronizing with the known live sources. The difficulty of discovering content sources has been studied and documented. In particular, [80] estimated that roughly 320 million new pages are created every week and that half of the pages are replaced by new ones in every 9 months (i.e., half life of 9 months). Many other characteristics of Web evolution have also been studied, *e.g.*, the rates of page and link churn [31, 39, 80], the rates of duplicate evolution [38], and the change rates of individual pages [13, 26, 88]. The work presented in this dissertation on Web page discovery (Chapter 3) is the first work, to the best of our knowledge, that studies the discoverability of the Web from search engines' point of view. In particular, we characterize the arrival of new pages and provide algorithms for discovery that exploit this characterization.

The problem of synchronizing information mediator's repository is broadly discussed in the context of data warehousing in [114]. Four different types of information sources are discussed there: *cooperative*, *logged*, *queryable* and *snapshot* sources. The difference between the sources is in terms of the way they provide content updates. More specifically, the cooperative sources push content update notifications to the data warehouse, the logged sources maintain a log of content updates that can be pulled by the data warehouse, the queryable sources needs to be polled/queried for their content to detect updates, and the snapshot sources push periodic dumps/snapshots of content in an offline fashion. This dissertation assumes sources to fall under the queryable category, as is true for most sites on the Web.

As mentioned before, in the Web search context the content is largely unstructured. If the content is structured instead, as is assumed in the data warehousing setting, then the synchronization problem can be seen as a materialized view maintenance problem as pointed out in [114]. In particular, the data in the central warehouse acts as a *materialized view*, where the *base data* resides at the information sources. The challenge is in maintaining the materialized view while the base tables at remote locations are being autonomously updated. This problem has been extensively studied under several sub-problems, *e.g.*, view self-maintenance [50, 55], consistency

maintenance [116, 122] and online view maintenance [89]. A detailed survey of this work is given in [98].

In Chapter 4 we propose a synchronization approach for search engines. Our synchronization approach is significantly different than prior approaches because we consider the manner in which pages change and the way in which users query and view results. In particular, we note that lack of synchronization can lead to both *false-positives* and *false-negatives* in a search result where (a) a false-positive occurs when a user inspects a page in the search result that is not relevant to the search query, while (b) a false-negative occurs when the search engine omits a relevant, high-quality page from the search result. Then we propose a metric of search repository quality and derive our synchronization policy that prioritizes redownloading of Web pages based on the expected gain in repository quality. More details are provided in Chapter 4.

2.3 Content Presentation

Much work has been done on visualizing query results for Web-based information systems [23, 42, 62, 71, 73, 117]. For example, in [73] it is suggested that in the Web search context, browsing and searching should be smoothly integrated into a single interface. Also, the focus has been on displaying Web search results in the context of surrounding documents. For example, in *Cha-Cha* [23] and *AMIT* [117], search results are organized into a nested list to show the hierarchical structure of the owner Web sites in which these pages appear. In *Web-Cutter* [71] a map of the Web is built and search results are shown as highlighted nodes in the map. Also, there has been work on highlighting Web hyperlinks according to the result of a query and user needs [42, 62]. In this dissertation we do not focus on the visualization aspect of search results. Instead, we focus on the orthogonal problem of finding the usefulness of result pages so that they can be presented accordingly.

The usefulness of a result page depends on its relevance for the user search query for which it is being shown. The problem of estimating relevance of a query-page pair has been extensively studied under *Text Information Retrieval* [8, 40, 97]. This task is performed by the searcher module of the search engine as discussed in Chapter 2.1. Several retrieval models have been proposed in the literature that can be employed by the searcher, *e.g.*, *Boolean*, *Vector* and *Probabilistic* Model. The Boolean model classifies each page to be either relevant or irrelevant to the query, *i.e.*, there is

no notion of being partly relevant [108, 112]. The Vector Model generalizes the Boolean model by allowing the notion of a page being partially relevant for a query [63, 95, 96, 97]. In particular, in the Vector Model each query-page pair is assigned a similarity score, which is computed based on the weights of the terms present in the query and the page. A widely used weighting scheme is TFIDF as discussed in [8]. The result pages are then presented in decreasing order of similarity scores. The Probabilistic model proposes a probabilistic approach to estimate the relevance of query-page pairs [64, 91, 92, 105]. More specifically, in the probabilistic model the similarity score of a query-page pair is set to be equal to the odds of the page being relevant to the query (*i.e.*, probability of being relevant divided by the probability of being irrelevant). Some parameters involved in this computation can be estimated by taking user feedback into account, and this model is known as the *probabilistic relevance feedback* model.

In this dissertation we consider a simplified model of ranking result pages[2] whereby pages are first filtered by a Boolean model of relevance, *i.e.*, each page is classified to be either relevant or irrelevant. Then the relevant pages are ordered based on some notion of *quality*. We leave incorporating other retrieval models as future work. The challenge is that the quality of pages is not known beforehand, as mentioned in Chapter 1. We propose to estimate quality by taking user feedback (*e.g.*, link creation, click impression, browse time) into account. The prior work on relevance feedback [51, 61, 93] also takes user feedback into account, but they do not study the involved exploration-exploitation tradeoff (discussed in Chapters 5 and 6). We address this tradeoff and propose algorithms with a proven performance guarantee.

The problem of estimating quality through user feedback has also been studied in the context of recommendation systems [9, 67, 72, 90]. There are some fundamental differences between the recommendation setting and the Web search setting as pointed out in [72]. For example, in many recommendation systems, users have some control of what they see and give feedback for, while in the Web search setting it is mostly controlled by search engines. Also, in recommendation systems there is typically no explicit notion of reward, while it is present in the Web search setting (*e.g.*, search engines earn money through ads) and hence, the exploration/exploitation tradeoff and reward maximization take a markedly more relevant role.

[2]The model for ranking advertisements is presented in Chapter 6.

Chapter 3

Web Page Discovery

As discussed in Chapter 1, search engines download pages from the Web and maintain a local repository to answer user search queries. The objective is to keep the repository fresh and complete. Doing so requires (a) discovering new pages on the Web and (b) synchronizing the repository with known live pages. These tasks are performed by a module called a *crawler*, as discussed in Chapter 2.1. Given an initial set of URLs, a crawler operates by selecting a URL and then downloading the page into the repository. By extracting the outlinks of downloaded pages, the crawler discovers new pages on the Web which are then added to the URL list.

Both discovery and synchronization tasks require redownloading of pages in the repository. Depending on the manner in which the crawler redownloads these pages, two different types of crawler have been proposed in the literature, called the batch and the incremental crawler, as discussed in Chapter 2.1. Since incremental crawlers are believed to be more effective [24], this dissertation assumes the following setting: an initial batch crawl is performed to create a sufficiently large repository. Then an incremental crawler takes over and it maintains the repository by continuously performing the discovering and synchronization tasks. We study the challenges that arise in doing so, *i.e.*, redownload prioritization of pages in the incremental crawler setting for discovery and synchronization purposes. This chapter concentrates on discovery; the following chapter gives a treatment of synchronization.

3.1 Introduction

The primary mechanism for discovery is by downloading known pages and following their outlinks. Figure 3.1 illustrates the key challenges of our problem. First, page p_5 may be discovered by downloading 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 downloading new page p_4. We will study policies for redownloading known pages in order to minimize the overhead of discovering new pages. More specifically, we have three goals: (a) characterize the arrival of new pages, (b) provide algorithms for discovery that exploit this characterization and (c) measure the overhead of discovering these pages for various levels of coverage.

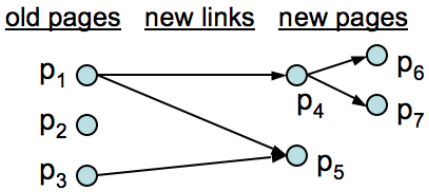

Figure 3.1: *Old pages linking to new pages.*

There are two primary mechanisms by which new pages arrive on the web. First, a Web site puts up a new page, and links to this new page from an existing page. Second, an entirely new Web site appears and is linked-to from an existing Web site. We focus on the first mechanism in this chapter, while the second mechanism is briefly studied in Appendix 9.1. A third possible mechanism is that one Web site puts up a new page without linking to it, and another Web site provides a link to the new page — this situation is very uncommon in general, and we do not study it.

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 minimal overhead. For example, in Figure 3.1, redownloading 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 our work. The fundamental questions are as follows:

1. **Basic feasibility** of the approach:
    - Is it the case for real Web sites that most new pages can be discovered via a small set of old pages?

2. **Key characteristics** that determine what crawling approaches are likely to work well:
   - To what extent are links to new pages redundant (as in $p_1 \to p_5$ and $p_3 \to p_5$ in Figure 3.1)?
   - Does the set of old pages that link to many new pages tend to remain consistent over time?
3. **Efficient crawl policies** for content discovery:
   - What is a good choice of old pages to seed the discovery process, given historical information and limited crawling resources?
   - What fraction of the crawling resources should be spent assessing the usefulness of various old pages, versus exploiting ones already known to be somewhat useful?

3.2 Problem 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 $X_t \stackrel{\Delta}{=} \cup_{j=1}^{t-1} V_j$ to be the set of *old pages* at time t, and define $Y_t \stackrel{\Delta}{=} V_t \setminus X_t$ to be the set of *new pages* at time t. The old pages X_t are pages that appeared before time t and the new pages Y_t are pages that appeared first at time t.

For convenience, we use the following representation for the old and new pages at any time t. Let $H_t = (X_t, Y_t, Z_t)$ be a bipartite graph consisting of the old pages X_t, the new pages Y_t, and an edge set Z_t. 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 3.1 shows the motivation for this definition: by downloading a node that reveals the start of a long chain of new pages, we may now proceed to download the entire chain of new pages recursively with no additional discovery overhead (as each node of the chain is new, and hence must be downloaded anyway).[1]

Next we introduce some notations and our evaluation metric.

[1] In practice, the chain is not pursued indefinitely. For example, if a downloaded page is of poor quality, then its outlinks to new pages may not necessarily be followed. And even for a page of good quality, only a bounded number of its outlinks (of sufficient quality) are pursued to avoid getting trapped in unintentional or malicious traps, *e.g.*, "infinitely" deep sites [20].

Notation	*Definition*
X_t	Set of old pages at time t.
Y_t	Set of new pages at time t.
H_t	Bipartite graph (X_t, Y_t, Z_t) consisting of the old pages X_t, the new pages Y_t, and an edge set Z_t.
$N(S)$	Set of new pages that are efficiently discoverable given set S, *i.e.*, neighbors of pages in S in graph H_t.
J_{xy}	Jaccard coefficient between the neighborhood sets, $N(x)$ and $N(y)$, of pages x and y.
p_x	Fraction of new pages discoverable by page x, *i.e.*, $\|N(x)\|/N$ where N denotes the total number of new pages.

Table 3.1: *Summary of symbols and their meanings.*

Notation. For each $x \in X_t$, we denote by $N(x)$ the set of new pages 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 pages 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 pages, and a value close to 0 means that they are almost non-overlapping. For convenience, a summary of the notations used in this chapter is provided in Table 3.1. (Some of these notations are not introduced until later in the chapter, and should be ignored for now.)

3.2.1 Overhead Metric

In general, if a set O of old pages are redownloaded 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" downloads, in the sense that they do not return new pages, in order to generate enough information to discover the 100 new pages. The overhead is 5%, and is a direct measure of the fraction of additional downloads necessary to gather a given number of new pages, in other words, a measure of efficiency.

3.2.2 Discovery Optimization Problem

As mentioned before, we study the discovery task (and the synchronization task) in the context of incremental crawlers. An incremental crawler operates by prioritizing the redownloading of repository pages. We perform this prioritization at the granularity of k old pages. In other words, first we decide which k old pages are redownloaded next. After these pages have been redownloaded (as well as the new pages efficiently discoverable from them), the next batch of k old pages is selected, and so forth and so on.

The problem of redownloading prioritization for discovering new pages can be formulated as the following optimization problem:

- *Objective:* Maximize the number of discovered new pages, *i.e.*, covered pages in Y_t. (In other words, minimize the overhead.)
- *Constraint:* Download at most k old pages, *i.e.*, pages from X_t.
- *Uncertainty:* Graph H_t between the old and new pages is not completely known.

We address this optimization problem in the offline and online settings. The simpler case, *i.e.*, when H_t is known, is the offline problem (Chapter 1). We study it in Section 3.5 and show that it is NP-hard. The online problem is studied in Section 3.6 where we do not have access to the complete H_t and we aim to learn its key characteristics over time.

3.3 Chapter Outline

We start by studying the basic feasibility of our approach of discovering new content by crawling old pages, in Section 3.5. To do so we describe an algorithm called GREEDY in Section 3.5.2, which has complete information about H_t; this algorithm should be viewed as an upper bound on the performance of any realistic algorithm.[2] We describe some real world snapshots of graph H_t in Section 3.5.1 and then study the performance of GREEDY on this data in Section 3.5.3.

Next, in Section 3.6 we propose a family of algorithms that use information that is realistically available about H_t during crawling. In particular, they do not have access to H_t, but depending

[2]This algorithm is not strictly speaking an upper bound, as it makes approximations in order to solve an NP-hard problem (details in Section 3.5.2); however, the information available to the algorithm allows it to perform substantially better than any realistic algorithm we have seen.

on the model, they may have partial information about old pages (X_t) and statistical information about H_t based on partial information about $H_{t'}$ for $t' < t$.

3.4 Related Work

Numerous early web studies focused on properties of a snapshot of the web graph [10, 16, 37, 66, 75]. More recently, attention has turned to evolutionary properties of the corpus. In this evolutionary model, researchers have considered the growth of the Web [13], the rates of page and link churn [31, 39, 80], the rates of duplicate evolution [38], and the change rates of individual pages [13, 26, 39, 88].

Parallel to this line of work, there has been a significant body of work on synchronizing already-discovered content, which has been studied in [25, 32, 33, 84, 119]. Already-discovered pages are redownloaded 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 redownloaded, 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 redownloaded pages. To our knowledge, this specific problem has not been studied previously.

Finally, there has been work in ordering the frontier of a crawl [27, 36], in which various policies are studied from the perspective of estimating the quality of a candidate for first-time crawl. In our work the focus is on maximizing the size of the frontier while incurring a minimal overhead.

3.5 Feasibility of Discovering New Content

In this section we study the feasibility of discovering new content on the Web by crawling old pages. We start by describing the data used for this feasibility study.

3.5.1 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 the problem of discovering new pages on existing sites. 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 [80] 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 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.

3.5.2 An Algorithmic Upper Bound: GREEDY

Our goal is to study the extent to which a crawling policy can discover new content by crawling old pages on the Web. To obtain an upper bound on the performance of any such crawling policy, we assume complete information about H_t to be known in this section. Recall that $H_t = (X_t, Y_t, Z_t)$ is the bipartite graph between the old pages X_t and the new pages Y_t. As discussed before in Section 3.2.2, the problem of maximizing the number of new pages discovered is equivalent to the k-budgeted problem in that case. The dual formulation of our optimization problem is when we are given a constant $\rho \leq 1$ and the goal is to cover at least $\rho \cdot |Y_t|$ pages in Y_t using as few pages from X_t as possible (*i.e.*, incur minimum overhead). This problem is also NP-hard and is known as the *ρ-partial cover* problem.

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 [65, 101]. 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 page that covers the most uncovered new pages.

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 $|N(C_t)| < \rho |Y_t|$.

3.5.3 Measurements

In this section we present a series of feasibility measurements. In particular, we will measure 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. In Appendix 9.1 we will perform an analysis of the Chilean dataset,

in which we study the relative prominence of new pages on existing sites, versus new pages on new sites.

3.5.3.1 Cover Size

For each site at each time, we construct the bipartite graph H and employ GREEDY to cover new pages. In particular, we run GREEDY until 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 3.2(a-b), the x-axis represents the threshold k that is the maximum size cover we will employ for any site/time pair. Figure 3.2(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 3.2(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 3.2(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.

3.5.3.2 Page 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

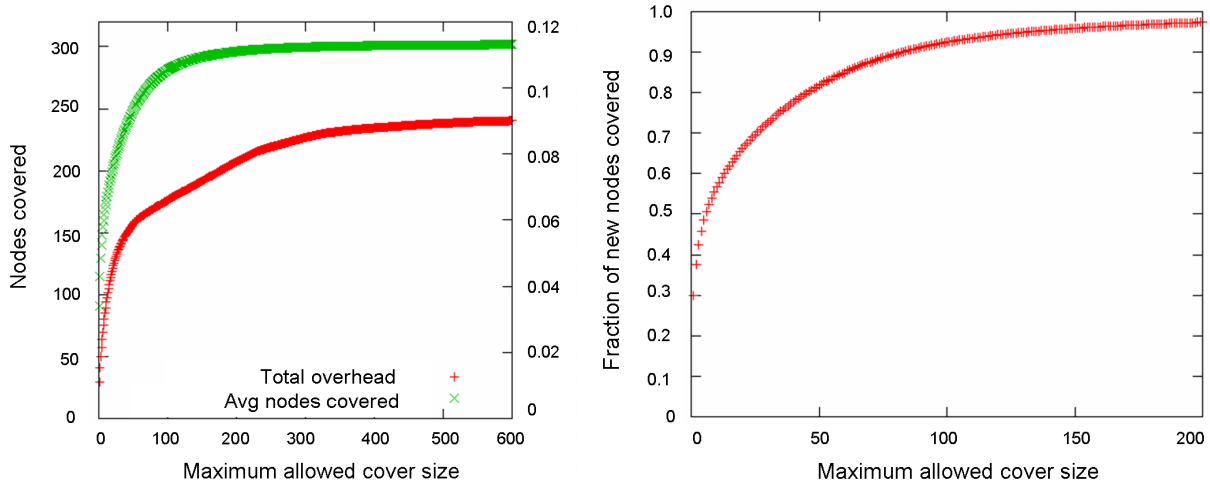

Figure 3.2: *(a) Overhead and number of covered pages, (b) fraction of new pages covered.*

in common. Figure 3.3 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 3.6.2.

3.5.3.3 Overhead of Discovering New Pages

Figure 3.4 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 3.6 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.

In Appendix 9.1 we study the overhead of discovering new pages on new sites (instead of new pages on old sites). For the remainder of this chapter we focus our exploration on existing sites.

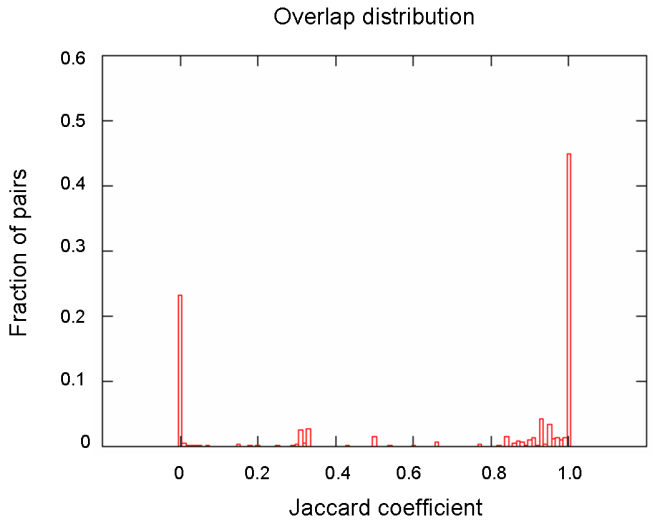

Figure 3.3: *Overlap distribution.*

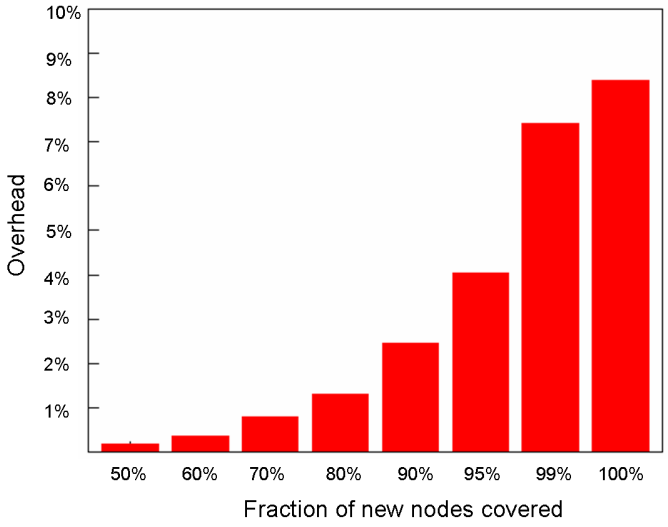

Figure 3.4: *Global discovery of new pages on old sites.*

3.6 History-based Algorithms

In the previous section we studied the feasibility of using a small set of existing pages to cover most of newly generated content — i.e., we measured whether there exists a small set of old pages with links to most of the new content. In this section we move to the algorithmic question of choosing such a set of pages when we do not have access to the entire bipartite graph H_t. We assume that we have access to the old pages X_t but not to Z_t, the set of edges, or to Y_t, the set of new pages. (In reality, we may only have access to a subset of X_t since some pages 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, downloads pages according to the number of new pages discovered historically when downloading 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 3.5.3.2, 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 5.4). Let S be the solution returned by an algorithm ALG. We define $\rho(\text{ALG})$ as the ratio of the number of new pages 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 pages.

3.6.1 Algorithm Based on Outdegree

We consider a very basic algorithm first. Suppose that for every old page i, we have an estimate of $p_i = |N(i)|/N$, the fraction of new pages covered by i. A natural algorithm is the following: pick k old pages with the largest p_i's and download these pages. 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 pages obtained from pages with degrees $p_{[2]}, ..., p_{[k]}$ that are distinct from the new pages obtained from $p_{[1]}$. The number of new pages found by the greedy algorithm is $(N \cdot p_{[1]}) + N_1$. The number of new pages found by the optimum cannot be greater than $(N \cdot p_{[1]}) + N_1 + N \cdot \sum_{i=k+1}^{2k} p_{[i]}$ (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 pages 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 3.5.3.2 lead to the next algorithm.

3.6.2 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 pages i, j, let p_{ij} be the fraction of new pages that i and j both cover: $p_{ij} = |N(i) \cap N(j)|/N$. Figure 3.3 empirically demonstrated that most pages 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 pages into groups that all link to almost the same set of new pages. Then,

Lemma 2 *Let $\epsilon_b, \epsilon_s \leq 1/3$. If for all pages i, j, we have either $J_{ij} \geq 1 - \epsilon_b$ or $J_{ij} \leq \epsilon_s$, then the set of old pages X_t can be partitioned into equivalence classes, where every pair of old pages 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, $|N(i) \setminus N(j)| \leq \epsilon_b \cdot |N(i) \cup N(j)|$, and similarly $|N(k) \setminus N(j)| \leq$

$\epsilon_b \cdot |N(k) \cup N(j)|$. So the most number of elements not in common between i and k is $\epsilon_b \cdot (|N(i) \cup N(j)| + |N(j) \cup N(k)|)$, i.e.,

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

that is strictly greater than ϵ_s for $\epsilon_b, \epsilon_s \leq 1/3$. The last line follows from $|N(i)| \geq |N(i) \cap N(j)| \geq (1-\epsilon_b) \cdot |N(i) \cup N(j)|$, 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 pages 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 page in i-th equivalence class and let n_i be the page 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 pages obtained by CLIQ. Denote by T_j the number of new pages obtained by adding j to S. From n_1 we get $T_1 = N \cdot q_1$ new pages. Define $q_{ij} = p_{n_i n_j} = |N(n_i) \cap N(n_j)|/N$. From the j-th page added by CLIQ, the number of new pages obtained is $T_j \geq (N \cdot q_j) - \sum_{i=1}^{j-1} (N \cdot q_{ij})$. Since n_i and n_j belong in different classes, $J_{n_i n_j} \leq \epsilon_s$, so

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

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

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

Now we upper bound the number of new pages covered by the optimum. The optimum cannot choose more than k pages from a class C_i, and so it cannot get more than $(1 + k \cdot \epsilon_b) \cdot q_i$ new pages from C_i: every new page added after n_i contributes no more than $\left(\epsilon \cdot N \cdot q_i\right)$ new pages 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_i's (chosen by CLIQ), and so optimum is upper bounded by $(1 + k \cdot \epsilon) \cdot N \cdot \sum_{i=1}^{k'} q_i$. So $\rho(\text{CLIQ}) \geq (1 - k' \cdot \epsilon_s)/(1 + k \cdot \epsilon_b)$ ■

In reality, not all pairs of old pages 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 pages into clusters recursively as follows. We choose a value for the parameter ϵ_b, and initialize with every page in its own cluster. We merge the clusters so that an old page 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 page in the cluster. (Note that this partitioning into clusters is well-defined.) We then run CLIQ using these clusters instead of equivalence classes.

3.6.3 Algorithm Based on Greedy Cover

Finally, we describe an algorithm COV that exploits previously observed cover information. Let S be the set of old pages 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 complete 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.

3.6.4 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:

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

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 page in X_t that ever linked to a new page 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.

3.6.5 Upper Bounds on Performance of Historical Algorithms

We begin by constructing an upper bound as follows. We implement the policy of downloading 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 (empirical) upper bounds our algorithms come, as a function of the amount of effort expended.

3.6.6 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). Occasionally, we will refer to the average cover size as *average depth*. 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 $\alpha = 1$ (*i.e.*, even weighting of all history) henceforth.

The results are shown in Table 3.2. Here are some conclusions that might be drawn from the data.

Budget	Depth	Over-head	Cove-rage	Budget	Depth	Over-head	Cove-rage	Budget	Depth	Over-head	Cove-rage
	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 3.2: *Analysis of covers produced by historical algorithms.*

(1) **Upper bound on historical algorithms.** Algorithm OD-ALL with infinite budget will eventually download 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.

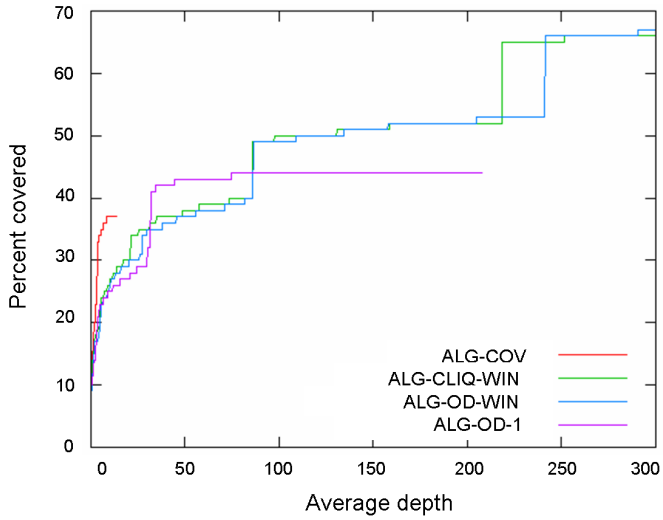

Figure 3.5: *Coverage as a function of average cover size, recrawl frequency* 1.

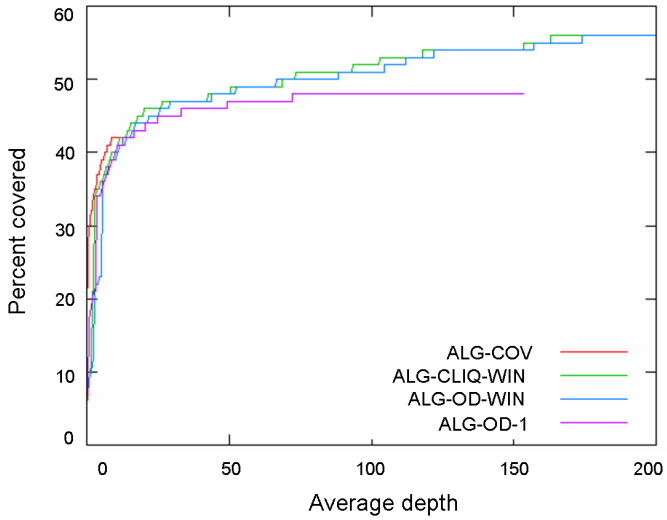

Figure 3.6: *Coverage as a function of average cover size, recrawl frequency* 4.

Based on these observations, we move to an evaluation of realistic candidates for discovery algorithms. Figure 3.5 plots coverage as a function of average depth (which is the same as 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 3.6 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.

3.7 Chapter Summary

In this chapter we studied the problem of discovering new pages by redownloading known pages. We formulated this as an optimization problem where the unknown parameter is the bipartite graph between the known and new pages. First, we studied the inherent difficulty of the offline discovery problem (*i.e.*, with perfect foreknowledge of which known pages link to which new pages) using a maximum cover formulation. Second, we studied the online problem which is a more realistic setting. Here the crawler must use historical statistics to estimate which known pages are most likely to yield links to new content. We recommended a simple algorithm that performs comparably to all approaches we consider.

We measured the overhead of discovering new content, defined as the average number of downloads required to discover one new page. We showed first that in the offline setting, it is possible to discover all new content with little overhead (9%). But online 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.

Chapter 4

Web Page Synchronization

In this chapter we turn from the issue of how to discover pages, to how to keep the repository synchronized as pages are updated on the Web.

4.1 Introduction

In response to a user search query, a search engine returns a list of relevant pages and ads, as shown in Figure 1.2. We refer to this response as the *slate* of the submitted query. Before we delve into the details of the synchronization task, we explain how a search engine processes the slates of queries. The slates are prepared using the local repository in which the search engine downloads and stores Web content. When a user submits a query, first the search engine finds out all the pages in the repository that are relevant for this query, as discussed in Chapter 2.3. Then it applies an internal scoring function to each relevant Web page. Applying this function to a page produces a numerical score, representing the best available estimate of the usefulness of the page to the user who submitted the query. Query results are then arranged on the slate in the form of a sequential list in descending order of score. When the user clicks on a link in the query result list, her Web browser fetches the current copy of the linked page from the live Web.[1]

If the repository is not closely synchronized with the Web, then the search engine may not include the most useful pages for a query at the top of the result list. Since users' attention is

[1] In this chapter we assume that the search engine does not serve copies of Web pages directly from its repository.

strongly biased toward the top of query result lists (Figure 1.3) and they have limited time to inspect results, users are likely to visit Web pages that are on the whole less useful than the ones they would visit if presented with an accurate slate, *i.e.*, a slate consisting of a hypothetical result list generated from a fully synchronized repository.

We present a simple example to illustrate the ways in which an out-of-date repository can adversely impact the user experience. Before proceeding we introduce some notation. Let $\mathcal{W}$ refer to the current collective content of the Web, and let $\mathcal{W}^L$ refer to the collective content of the search engine's local repository. For a Web page p, $\mathcal{W}[p]$ denotes the current live Web copy of p, and $\mathcal{W}^L[p]$ denotes whatever copy of p (if any) is currently stored in the repository.

Now, suppose the Web is tiny, such that $\mathcal{W} = \{\mathcal{W}[p_1], \mathcal{W}[p_2], \mathcal{W}[p_3]\}$. Suppose the repository $\mathcal{W}^L$ contains copies of two of the three pages currently available on the Web (namely, pages p_1 and p_2), plus a copy of one more page p_4, which has been removed from the Web since it was last downloaded into the repository. Hence, $\mathcal{W}^L = \{\mathcal{W}^L[p_1], \mathcal{W}^L[p_2], \mathcal{W}^L[p_4]\}$. Suppose furthermore that the repository copies of p_1 and p_2 are both out of date, such that $\mathcal{W}^L[p_1] \neq \mathcal{W}[p_1]$ and $\mathcal{W}^L[p_2] \neq \mathcal{W}[p_2]$. The content of each copy of each page is shown in Table 4.1.

Consider the query "cancer." For the sake of our example assume a simple Boolean scoring function that returns `true` if there is a keyword match, and `false` otherwise. Observe four types of discrepancies between the repository and the live Web, each of which leads to distorted results for this query: (1) Web pages with increased score not yet reflected in the repository, e.g., p_1, (2) pages with decreased score, e.g., p_2, (3) pages not yet discovered by the search engine, e.g., p_3, and (4) pages that have been removed from the Web but remain present in the repository, e.g., p_4.

The third discrepancy described above was studied under the discovery problem in Chapter 3. Hence, in this chapter we focus on the problem of synchronizing Web pages already present in the repository. To handle the synchronization of newly discovered pages, we can incorporate them into the repository and apply the same technique.

4.2 Problem Formulation

First we introduce a metric for the quality of a search engine's repository in Section 4.2.1. The metric takes into account the discrepancies that can arise due to an out-of-date repository as described above. Then in Section 4.2.2 we use this metric to formulate the synchronization task as a

Page p	**Web copy** $\mathcal{W}[p]$	**Search engine copy** $\mathcal{W}^L[p]$
p_1	New Technology: A new thyroid cancer therapy	New Technology: A new chipset designed for cell phones
p_2	Seminar: Important traffic laws and rules	Seminar: Cancer symptoms
p_3	Cancer Management: Early tests to detect breast cancer	(*Not present in the repository*)
p_4	(*Removed from the Web*)	Cancer association seeking volunteers to help raise awareness

Table 4.1: *Example scenario.*

constrained optimization problem.

4.2.1 Repository Quality Metric

We begin by introducing some additional notation. (For convenience, a summary of the notation we use is provided in Table 4.2.) Let $A(q, \mathcal{W}^L)$ denote the answer provided by a search engine in response to query q, which we assume is in the form of a ranked list, compiled according to scores computed over copies of Web pages stored in the local repository $\mathcal{W}^L$. Let $S(\mathcal{W}^L[p], q)$ denote the result of applying the search engine's scoring function S to the locally-available repository copy of p for query q. Similarly, let $S(\mathcal{W}[p], q)$ denote the result of applying the same scoring function to the live Web copy $\mathcal{W}[p]$ of page p for query q. We assume the scoring function provides an estimate of the usefulness of a page to a user who submits a particular query. We restrict the scoring function $S()$ to be one in which the score of a page depends on the content of that page only. (The score may also incorporate a global notion of "importance," e.g., PageRank [81], that is recomputed on occasion, at a time-scale that is large relative to the rate of redownloading pages.) We also assume the score of a page is zero if it does not contain at least one instance of every term in a query.

If $V(p, a)$ denotes the likelihood with which a typical user would view page p if presented with result list a (most likely influenced strongly by the rank position of p within a, as discussed below), then we can express the expected cumulative usefulness of the search engine's answer $a = A(q, \mathcal{W}^L)$

to query q as:

$$k \cdot \sum_{p \in a} V(p, a) \cdot S(\mathcal{W}[p], q)$$

where k is an arbitrary constant of proportionality. If we expect a certain workload $\mathcal{Q}$ of queries, with each query $q \in \mathcal{Q}$ issued with frequency f_q, we can write the expected average usefulness of querying the search engine as:

$$\sum_{q \in \mathcal{Q}} f_q \cdot k \cdot \sum_{p \in A(q, \mathcal{W}^L)} V(p, A(q, \mathcal{W}^L)) \cdot S(\mathcal{W}[p], q)$$

We model the *quality* of a repository $\mathcal{W}^L$ with respect to a particular scoring method $S()$ and an expected usage pattern (query workload $\mathcal{Q}$ and viewing likelihood function $V()$) as a scalar value $Q(\mathcal{W}^L)$. In particular we define $Q(\mathcal{W}^L)$ to be directly proportional to expected average usefulness:

$$Q(\mathcal{W}^L) \propto \sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p \in \mathcal{W}} V(p, A(q, \mathcal{W}^L)) \cdot S(\mathcal{W}[p], q) \tag{4.1}$$

We discuss how to model the viewing likelihood function now. Empirical measurements taken during an extensive user study [61] indicate that the expected viewing likelihood $V(p, a)$ depends primarily on the rank of p in a, denoted $R(p, a)$, as mentioned above. In light of these observations we model viewing likelihood as a function of rank, so that $V(p, a) = I(R(p, a))$ for some function $I(r)$, serves as a reasonable first-order approximation of true user behavior (the same model was also adopted in [28]). The function $I(r)$ can be estimated by monitoring user behavior and fitting a curve. For example, AltaVista usage logs analyzed in [28, 69] reveal that the following relationship holds quite closely:

$$I(r) = c \cdot r^{-3/2} \tag{4.2}$$

where c is a normalization constant.[2] By substituting into Equation 4.1 we obtain:

$$Q(\mathcal{W}^L) \propto \sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p \in \mathcal{W}} I(R(p, A(q, \mathcal{W}^L))) \cdot S(\mathcal{W}[p], q) \tag{4.3}$$

[2] User views were measured at the granularity of groups of 10 results in [69], and later extrapolated to individual pages in [28].

(The rank of a page not present in a result list is taken to be ∞, with $I(\infty) = 0$.)

Ideal Repository Quality. It is instructive to formulate an expression for the upper bound on search repository quality. As long as the inspection likelihood function $I(r)$ is monotonically nonincreasing, the expected cumulative score of visited pages is maximized when pages are always presented to users in descending order of their true score $S(\mathcal{W}[p], q)$. This ideal situation occurs when a search engine's repository is exactly synchronized with the Web at all times, such that $\mathcal{W}^L = \mathcal{W}$. Hence, we denote the highest possible search repository quality as $Q(\mathcal{W})$, where:

$$Q(\mathcal{W}) \propto \sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p \in \mathcal{W}} I(R(p, A(q, \mathcal{W}))) \cdot S(\mathcal{W}[p], q) \tag{4.4}$$

It is not difficult to construct a formal proof that presenting search results in descending order of true score (based on the live Web copy) does indeed achieve a tight upper bound on quality.

Normalized Quality Metric. It is convenient to represent search repository quality on a known, bounded scale. Hence we define the quality of repository $\mathcal{W}^L$ relative to the upper bound on quality corresponding to the case in which $\mathcal{W}^L = \mathcal{W}$, such that $Q(\mathcal{W}^L) \in [0, 1]$. In this way we arrive at our final, normalized expression for $Q(\mathcal{W}^L)$:

$$Q(\mathcal{W}^L) = \frac{\sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p \in \mathcal{W}} I(R(p, A(q, \mathcal{W}^L))) \cdot S(\mathcal{W}[p], q)}{\sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p \in \mathcal{W}} I(R(p, A(q, \mathcal{W}))) \cdot S(\mathcal{W}[p], q)} \tag{4.5}$$

4.2.2 Web Synchronization Optimization Problem

Similar to the discovery task in Chapter 3, we study the synchronization task under the incremental crawler setting. We perform redownload prioritization at the granularity of B pages, *i.e.*, first we decide which B pages are redownloaded next, and after these pages have been redownloaded, we select the next batch of B pages to redownload, and so forth and so on. (We assume uniform resource cost across all pages.) Also, we assume the crawler to employ the in-place updating mechanism, *i.e.*, the local repository is updated on the fly as Web pages are downloaded.

Based on the above metric of repository quality we formulate the synchronization problem as follows:

- *Objective:* Maximize the repository quality as defined by the quality metric $Q(\mathcal{W}^L)$ (Equa-

Notation	*Definition*
$\mathcal{W}$	Current content of the Web.
$\mathcal{W}^L$	Current content of the search engine's local repository.
$\mathcal{Q}$	Workload of queries.
f_q	Frequency of query q.
$\mathcal{W}[p]$	Current live Web copy of page p. Similarly, $\mathcal{W}^L[p]$ denotes the current search engine copy of p.
$S(\mathcal{W}[p], q)$	Score of the Web copy of page p for query q. Similarly, $S(\mathcal{W}^L[p], q)$ denotes the score of the local copy of p.
$A(q, \mathcal{W})$	Ranked list provided by the search engine in response to query q given repository $\mathcal{W}$.
$R(p, a)$	Rank of page p in result list a.
$V(p, a)$	Viewing likelihood of page p in result list a.
$I(r)$	Inspection probability of rank r in a result list.
$Q(\mathcal{W})$	Quality of repository $\mathcal{W}$.
$\Delta Q(p, t)$	Change in repository quality on redownloading of page p at time t.

Table 4.2: *Summary of symbols and their meanings.*

tion 4.5).[3]

- *Constraint:* Redownload at most B pages.
- *Uncertainty:* The update behavior of repository pages is unknown, *i.e.*, how $\mathcal{W}[p]$'s involved in the quality metric (Equation 4.5) change over time.

The optimization problem can be studied in the offline and online settings. In the offline setting the update behavior of pages is known while in the online setting it is unknown.

We propose a synchronization policy that prioritizes redownloading of Web pages based on the expected gain in repository quality. In particular, let $\Delta Q(p, t)$ denote the expected improvement in repository quality if p is redownloaded into the repository at time t (details in Section 4.5.1). Then in the offline setting we can compute $\Delta Q(p, t)$ exactly (using the current Web copy and search copy of p) and prioritize redownloading of pages accordingly. In the online setting we do not have access to the Web copy of p (before redownloading it), and so the principal challenge is how to estimate $\Delta Q(p, t)$ if a particular page were to be redownloaded, without redownloading it.

[3]Note that the repository quality metric corresponds to the primal optimization problem (*i.e.*, maximize the objective given a fixed amount of resources), while the overhead metric of Section 3 applies to the dual optimization problem, *i.e.*, minimize the resources required to achieve the given objective. This difference in metrics is not intrinsic to the problems; both discovery and synchronization tasks can be studied as primal or dual problems.

4.3 Chapter Outline

In Section 4.5 we formally define the benefit of redownloading a Web page ($\Delta Q(p,t)$) which forms the basis for our offline synchronization policy. Then we move on to describe the online synchronization policy. In Section 4.6 we provide an efficient method for measuring $\Delta Q(p,t)$ approximately. Our method is tightly integrated with the process of updating an inverted index that is maintained over the repository, and incurs little additional overhead. We evaluate the effectiveness of our synchronization policy empirically using real Web data in Section 4.7. In particular we show that the improvement in quality yielded by redownloading a particular page is fairly consistent across time, making our approach feasible. We also compare our policy against prior Web page synchronization schemes, and show that our policy makes much more effective use of resources when measured according to our notion of repository quality.

4.4 Related Work

Web crawling is a well-studied research problem. The subproblem of synchronizing pages under resource constraints has been studied in [25, 119]. In [25], the optimization objective is to minimize the average *freshness* or *age* of pages in the repository, treating all pages and changes to pages with uniform importance. Unlike in our work, neither the manner in which pages change nor the way in which users query and view results are considered.

In [119] a metric that assesses the level of "embarrassment" to the search engine was proposed, along with a corresponding page synchronization policy. In the model of [119], embarrassment accrues whenever a user clicks on a search result link, only to discover that the destination page is not, in fact, relevant to the query she had issued. While a search engine with a high embarrassment level clearly does not provide quality service to its users, minimizing (or even eliminating) embarrassment is not all that is needed to ensure a good user experience. Consider that the omission of high-quality, relevant documents from search results generates no embarrassment, although it can degrade the quality of the user experience substantially (of course the user may not "know what she is missing"). This example illustrates the difference in philosophy between embarrassment-based crawling and our crawling paradigm. We provide a thorough experimental comparison of our page synchronization scheme with those of [25] and [119] in Section 4.7.

As mentioned before, we take search queries and their frequencies into account in our synchronization policy. For a large scale search engine the storage/computation overhead of considering each query in the crawler can be considerable. In [85] it is shown how to reduce this overhead in the context of prioritizing the crawling of uncrawled pages (*i.e.*, pages in the *frontier*). In particular, they give a method for identifying *needy queries* which are those queries whose search results can be improved the most by crawling. It is shown that instead of considering the complete query log, a crawler can focus on just the needy queries and still achieve most of the advantage of using the query log. While [85] provides a definition of needy queries for the purpose of prioritizing the frontier, a similar definition should be derivable for the synchronization task in hand. Doing this is left as future work; for the purpose of this chapter we will simply use the complete query log in our synchronization policy.

Work on *focused crawling* [21] concentrates on how to obtain an initial crawl of the portion of the Web likely to be of interest to a particular community of users. Our work is complementary. Our approach to incremental crawling can be used to keep the repository of a focused search engine up-to-date as the Web evolves.

4.5 New Web Synchronization Policy

Our Web synchronization policy is driven directly by our metric of search repository quality introduced in Section 4.2.1. More specifically, if $\Delta Q(p,t)$ denotes the expected improvement in repository quality if p is redownloaded into the repository at time t, then at each time step t our policy attempts to redownload pages that have highest $\Delta Q(p,t)$ values. Next we define $\Delta Q(p,t)$ formally.

4.5.1 Change in Quality

Before we describe $\Delta Q(p,t)$, we extend our notation to incorporate time as follows. Let $\mathcal{W}_t$ and $\mathcal{W}_t^L$ refer to the state of the live Web and of the local repository, respectively, at time t. Now consider a page p and let $\mathcal{W}_t^{L+p}$ refer to the state of the repository if it is altered by incorporating the latest version of p, such that $\mathcal{W}_t^{L+p}[p] = \mathcal{W}_t[p]$. (As mentioned before, we assume the process of redownloading a page and incorporating it into the repository occurs instantaneously.) We define

the *change in repository quality* $\Delta Q(p,t)$ due to redownloading page p at time t as:

$$\begin{aligned}\Delta Q(p,t) &= Q(W_t^{L+p}) - Q(W_t^L) \\ &= \frac{\sum_{q\in\mathcal{Q}} f_q \cdot \sum_{p'\in\mathcal{W}} \Delta I(p',q,\mathcal{W}_t^L,\mathcal{W}_t^{L+p}) \cdot S(\mathcal{W}_t[p'],q)}{\sum_{q\in\mathcal{Q}} f_q \cdot \sum_{p'\in\mathcal{W}} I(R(p',A(q,\mathcal{W}_t))) \cdot S(\mathcal{W}_t[p'],q)}\end{aligned} \tag{4.6}$$

where $\Delta I(p,q,\mathcal{W}_1,\mathcal{W}_2)$ denotes the change in the expected frequency with which users inspect page p as a consequence of issuing query q, if repository $\mathcal{W}_2$ is used instead of $\mathcal{W}_1$ to construct query answers. Formally:

$$\Delta I(p,q,\mathcal{W}_1,\mathcal{W}_2) = I(R(p,A(q,\mathcal{W}_2))) - I(R(p,A(q,\mathcal{W}_1))) \tag{4.7}$$

As an aside, we highlight two important yet subtle characteristics of $\Delta Q(p,t)$. First, the value of $\Delta Q(p,t)$ for a given page p depends on the current state of the Web at large $\mathcal{W}_t$, because our quality metric is normalized relative to the quality of a hypothetical ideal search engine that has perfect and instantaneous access to the live Web. Second, $\Delta Q(p,t)$ also depends on the current state of pages other than p in the search engine repository $\mathcal{W}_t^L$. Consequently, if we consider two pages p_1 and p_2 that are redownloaded nearly simultaneously although in some serial order, the improvement in quality attributed to the action of redownloading each page may depend on the order in which they are redownloaded. Both of these characteristics imply the following property: Given a page p and two moments of time t_1 and t_2 such that page p is never updated or redownloaded during the interval $[t_1,t_2]$ (i.e., both $\mathcal{W}_t[p]$ and $\mathcal{W}_t^L[p]$ remain unchanged for all $t \in [t_1,t_2]$), it is not necessarily the case that $\Delta Q(p,t_1) = \Delta Q(p,t_2)$.

4.5.2 Synchronization Policy

We are ready to describe our synchronization policy now. Suppose that, due to resource limitations, it is only possible to redownload and reindex up to B pages per time unit. In our crawling policy, as shown in Figure 4.1, page redownloading is scheduled on the basis of priorities. Each page p is assigned a numeric priority $P(p,t)$ and ideally, we would set $P(p,t) = \Delta Q(p,t)$. Since it is generally far from feasible to determine the precise value of $\Delta Q(p,t)$ (due to the absence of information about the update behavior of p), we substitute the best available estimate of the expected change in repository quality due to redownloading page p. Stated formally, we set $P(p,t) = E(\Delta Q(p,t))$,

where $E()$ denotes our estimation procedure. Since pages in the repository have been downloaded at least once in the past, the expected benefit in terms of repository quality of redownloading the page again in the future can be estimated using information observed during previous redownloads of the same page. In particular, we propose to estimate $\Delta Q(p,t)$ for present or future values of t based on the value of $\Delta Q(p,t')$ measured at one or more times t' at which page p was redownloaded in the past ($t' < t$). (We give a method for doing this extrapolation in Section 4.7.1.)

At time t:

- For each page p compute its priority $P(p,t) = E(\Delta Q(p,t))$.
- Redownload B pages of highest priority.

Figure 4.1: *Our synchronization policy.*

The main challenge is how to estimate the change in repository quality each time a page is redownloaded, without incurring substantial additional overhead. We address this issue next.

4.6 Estimating Changes in Quality During Crawler Operation

Our approach to page redownload scheduling hinges on the ability to measure the change in repository quality, $\Delta Q(p,t)$, each time a page is redownloaded. Clearly, a highly efficient method of measuring $\Delta Q(p,t)$ is needed. We focus on measuring the numerator of our expression for $\Delta Q(p,t)$ (Equation 4.6), since the denominator is the same across all pages and does not affect relative differences in priorities. Hence our goal is to measure the *absolute change in quality*, $\Delta Q_A(p,t)$, defined as:

$$\Delta Q_A(p,t) = \sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p' \in \mathcal{W}} \Delta I(p', q, \mathcal{W}_t^L, \mathcal{W}_t^{L+p}) \cdot S(\mathcal{W}_t[p'], q)$$

where $\mathcal{W}_t^L$ denotes the contents of the search engine repository before page p is redownloaded, and $\mathcal{W}_t^{L+p}$ denotes its contents afterward. From Equation 4.7, $\Delta I(p,q,\mathcal{W}_1,\mathcal{W}_2) = I(R(p,A(q,\mathcal{W}_2))) - I(R(p,A(q,\mathcal{W}_1)))$.

The anticipated workload $\mathcal{Q}$ can be estimated using recent query logs, and the function $I(r)$ can be determined from usage logs. For the remainder of this chapter we assume $I(r) = c \cdot r^{-3/2}$, following [28]. As mentioned before we restrict the scoring function $S()$ to be one in which the score of a page depends on the content of that page only.

Even if we sidestep the difficulty that true scores $S(\mathcal{W}_t[p'], q)$ of pages $p' \neq p$ are unavailable, say by substituting estimates, it is very expensive to compute $\Delta Q_A(p,t)$ directly. Doing so requires materializing the result list of every query affected by the change in content of page p, and for each list examining the scores and ranks of every page whose rank has changed. Therefore we seek an efficient approximation scheme.

4.6.1 Approximation Scheme

Our approximation scheme has the following two parts:

Approximating the Workload: Since most search engine queries consist of only one or two terms, we approximate the query workload by breaking each multiple-term query into a set of single-term queries. The resulting simplified workload, $\mathcal{Q}'$, consists of only single-term queries and their frequencies, where the frequency $f_{q'}$ of a single-term query $q' \in \mathcal{Q}'$ is set equal to the sum of the frequencies of the queries in $\mathcal{Q}$ in which q' occurs. Now, observe that for any query q in $\mathcal{Q}'$ consisting of a term that occurs in neither $\mathcal{W}_t^L[p]$ nor $\mathcal{W}_t^{L+p}[p]$, $S(\mathcal{W}_t^{L+p}[p], q) = S(\mathcal{W}_t^L[p], q) = 0$ so the result of q remains unchanged by the update to page p. Hence we arrive at the following approximate expression for $\Delta Q_A(p,t)$:

$$\Delta Q_A(p,t) \approx \sum_{q \in \mathcal{S}} f_q \cdot \sum_{p' \in \mathcal{W}} \Delta I(p', q, \mathcal{W}_t^L, \mathcal{W}_t^{L+p}) \cdot S(\mathcal{W}_t[p'], q)$$

where $\mathcal{S} = \mathcal{Q}' \cap (\mathcal{W}_t^L[p] \cup \mathcal{W}_t^{L+p}[p])$.

Approximating the Score-Rank Correspondence: To avoid computing result lists directly, we use precomputed histograms that provide an approximate mapping between score and rank among the results of a particular query. In particular, for each query $q \in \mathcal{Q}'$ we maintain a histogram H_q of result scores, consisting of the average score for each of a sequence of ranges of rank values, with ranges distributed in an exponential fashion so that scores corresponding to small rank values are

approximated most accurately. Since they are only intended to provide an approximate mapping between score and rank, these histograms need only be updated periodically, and can be made very small so as to fit in main memory (in our experiments described in Section 4.7, we used three buckets per histogram; the space requirement is just 20 bytes per query). Let $\hat{R}(s, H_q)$ denote the result of using histogram H_q to estimate the rank in the result of query q of a page whose score is s. Conversely let $\hat{S}(r, H_q)$ denote the result of using H_q to estimate the score of a page appearing at rank position r in the result of query q. Using our technique of approximating the relationship between score and rank for a particular query using a histogram, we estimate $\Delta Q_A(p,t)$ as follows.

At the time page p is redownloaded, suppose we are able to determine the set $\mathcal{S}$ of queries affected by the changes in p, as well as for each $q \in \mathcal{S}$ the scores for p both before and after the redownload is applied, i.e., $S(\mathcal{W}_t^L[p], q)$ and $S(\mathcal{W}_t^{L+p}, q)$, efficiently. (We describe how to obtain these quantities efficiently later in Section 4.6.2.) For notational ease let $s_1 = S(\mathcal{W}_t^L[p], q)$ and $s_2 = S(\mathcal{W}_t^{L+p}[p], q)$. For each query $q \in \mathcal{S}$ we estimate $R(p, A(q, \mathcal{W}_t^L))$ and $R(p, A(q, \mathcal{W}_t^{L+p}))$ as $r_1 = \hat{R}(s_1, H_q)$ and $r_2 = \hat{R}(s_2, H_q)$, respectively. Our expression for the component of $\Delta Q_A(p,t)$ corresponding to query q becomes:

$$\begin{aligned}\sum_{p' \in \mathcal{W}} \Delta I(p', q, \mathcal{W}_t^L, \mathcal{W}_t^{L+p}) \cdot S(\mathcal{W}_t[p'], q) \quad &\approx \quad \big((I(r_2) - I(r_1)) \cdot s_2\big) \\ &\quad + \sum_{p' \in \mathcal{W}, p' \neq p} \Delta I(p', q, \mathcal{W}_t^L, \mathcal{W}_t^{L+p}) \cdot S(\mathcal{W}_t[p'], q)\end{aligned}$$

Now we focus on transforming the second term into a form that is amenable to efficient evaluation. Assume $r_1 < r_2$ (the case in which $r_1 > r_2$ is symmetric). We transform the summation over pages into a summation over rank positions affected by the shift in rank of page p, and invoke our histogram function to obtain a ballpark estimate of true scores:

$$\sum_{p' \in \mathcal{W}, p' \neq p} \Delta I(p', q, \mathcal{W}_t^L, \mathcal{W}_t^{L+p}) \cdot S(\mathcal{W}_t[p'], q) \approx \sum_{r=r_1+1}^{r_2} \big(I(r-1) - I(r)\big) \cdot \hat{S}(r, H_q)$$

Assume now that r_1 and r_2 fall into the same histogram bucket B_i (it is straightforward to extend our expressions to handle the case in which r_1 and r_2 span multiple buckets). We model the scores for rank values within a single histogram bucket as being evenly distributed. Let δ_i denote the difference between the scores for two consecutive rank positions in bucket B_i. For rank position r ($r_1 < r < r_2$), $\hat{S}(r, H_q) = s_1 - (r - r_1) \cdot \delta_i$. Substituting into our expression above and simplifying,

we obtain:

$$\big(I(r_1)\cdot\hat{S}(r_1+1,H_q)\big)-\sum_{r=r_1+1}^{r_2-1}\big(I(r)\cdot\delta_i\big)-\big(I(r_2)\cdot\hat{S}(r_2,H_q)\big)$$

For cases in which (r_2-r_1) is small we evaluate the above expression exactly, using $I(r)=c\cdot r^{-3/2}$. When (r_2-r_1) is large we use an approximate form of the middle term derived by substituting a definite integral in place of the summation. A closed-form solution for the integral is easy to obtain. The net result of applying the integral approximation is:

$$\sum_{k=i}^{j}k^{-3/2}\approx 2\cdot\left(\frac{1}{\sqrt{i}}-\frac{1}{\sqrt{j-1}}\right)$$

We found our experimental results (Section 4.7) not to be very sensitive to the settings of our approximation parameters: the cutoff for treating rank value differences as "large" and the number of histogram buckets to use.

4.6.2 Taking Measurements During Index Maintenance

Our scheme for estimating the change in repository quality upon redownloading page p described in Section 4.6.1 takes as input the set $\mathcal{S}\subseteq\mathcal{Q}'$ of single-term queries affected by the changes in p, and for each $q\in\mathcal{S}$ the scores for p both before and after the redownload is applied, i.e., $S(\mathcal{W}_t^L[p],q)$ and $S(\mathcal{W}_t^{L+p},q)$. Conveniently, it is possible to compute these scores efficiently by closely coupling the measurement process with the process of updating the inverted index, which is a necessary operation that makes newly-downloaded content "searchable."

An inverted index contains lists of *postings* extracted from the repository. A posting corresponds to a unique term/page pair, and typically contains the number of times term appears in the page, font sizes, and any other information required to evaluate the scoring function. Postings are typically updated in batches, after a set of pages have been redownloaded into the repository. During the index updating process, postings corresponding to terms no longer present in pages are removed, and new postings are added corresponding to new terms. With our measurement technique, whenever a posting corresponding to term $\mathcal{T}$ in page p is added or removed, the resulting shift (if any) in the score of p for query $q=\{\mathcal{T}\}$ is recorded in a special in-memory buffer. After processing of the batch of updates has completed, ΔQ_A estimates are computed using the procedure of Section 4.6.1.

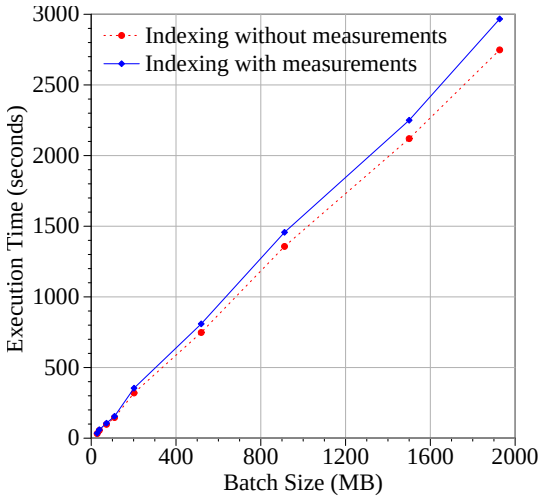

Figure 4.2: *Overhead of our measurement scheme.*

4.6.3 Overhead of Measurement Scheme

We integrated our quality change measurement scheme with the indexing component of Lucene [59], a publicly-available document indexing and retrieval system. Figure 4.2 shows the time it takes to index a batch of HTML pages, both with and without our special measurement code. Batch size (in megabytes) is plotted on the x-axis. Total running time is plotted on the y-axis. Our measurement scheme incurs very modest overhead of $7 - 8\%$.

4.7 Experiments

We compared our synchronization scheme with other schemes proposed in the literature, using simulations over real Web evolution data. We used two different data sets (both from the UCLA WebArchive project data[113, 80]):

1. **B**oston **D**ata **Set** (BDS): A 48-week archive of a single Web site, www.boston.com. The complete Web site was crawled once every week. Since our focus is on redownloading the pages that persist over an extended period of time, pages not present in all 48 weekly snapshots were removed. The remaining Web pages number around $16,000$.

2. **M**ultiple site **D**ata **S**et (MDS): A 48-week archive of 15 different Web sites, each sampled from a different OpenDirectory topic area. As with BDS, pages not present in every weekly snapshot were removed. Furthermore, in order to emphasize the role played by Web page redownloading in the relatively short duration of the Web evolution data we had access to, and also to reduce the time required to perform each run of our experiments, we only retained pages that changed in some way (as determined by a checksum) at least once during the 48-week period. The final data set consists of around 19,000 pages.

To obtain query workloads for our experiments we used the publicly-available AltaVista query log [5]. It consists of around seven million single-term and multi-term queries. Since our data sets are concentrated around fairly specific topics, whereas the topics represented in the query log are quite broad, we created workloads specific to each data set by filtering queries based on relevance to the pages in each data set. In particular, we eliminated queries for which the sum of TF-IDF scores across all pages in a data set was below a certain threshold. The threshold was chosen based on observing a knee in the distribution that we felt would serve as a natural cutoff point for query relevance.

Next we describe each of the three page synchronization strategies we evaluated in turn.

4.7.1 Web Page Synchronization Schemes Evaluated

Staleness-Based Synchronization Policy:
With staleness-based synchronization (*SBR*) [25], the objective is to minimize the number of *stale* pages in the search engine repository.[4] It is shown in [25] that under the staleness objective, when resources are limited it is best to abandon redownloading of frequently updated pages in favor of redownloading of other, less frequently updated pages.

In the simplest implementation of *SBR*, the repository copy of a page is considered stale if it is not identical to the current Web copy. Since Web pages are often updated in fairly minor ways (e.g., advertisements, timestamps) we used the standard method of *shingling* [14, 17] as a heuristic for discriminating between significant and insignificant updates. A page is considered stale if the fraction of shingles that differ between the repository copy and Web copy of the page exceeds a

[4] In [25] an alternative optimization objective, minimizing average *age*, is also proposed. Our preliminary experiments showed that age-based synchronization did not perform as well as staleness-based synchronization under our metric, so we did not consider it further.

particular threshold $\tau_{SBR} \in [0, 1]$. In our experiments we tested values of τ_{SBR} throughout the range $[0, 1]$.

The work in [25] focuses uniquely on determining with which frequency to redownload each page. No algorithm is provided for scheduling redownloads in the presence of hard resource constraint. We used the transportation algorithm suggested in [119] for this purpose.

Embarrassment-Based Synchronization Policy:

With embarrassment-based synchronization (*EBR*) [119], the objective is to minimize the level of "embarrassment" to a search engine provider. Embarrassment accrues whenever a user clicks on a search result link, only to discover that the destination page is not, in fact, relevant to the query she had issued. (A boolean notion of relevance is assumed.)

The work in [119] applies to a wide variety of page update models, including the fairly general *quasi-deterministic* model. We did not feel our 48-week data set contained a sufficient duration of data to fit a reliable quasi-deterministic model, so we used the simpler *Poisson* update model, as done in [25].

An important parameter in *EBR* is $d(p)$, which denotes the probability that if the repository copy of page p is out-of-date with respect to the current Web copy (i.e., $\mathcal{W}^L[p] \neq \mathcal{W}[p]$), whenever the search engine presents page p to a user, p turns out to be an irrelevant response for the query that was issued (note that $d(p)$ is a query-independent parameter). No method of estimating this parameter is provided in [119]. Since the shingling technique is a widely-accepted way of measuring the difference between two Web pages, or two copies of the same page, we apply it here. In particular, we assume that if a page undergoes an update, it becomes irrelevant to an average query if the fraction of shingles that change exceeds a configurable threshold τ_{EBR}. We compute $d(p)$ as the fraction of updates to page p that induce at least τ_{EBR} fraction of the shingles to change. In our experiments we tested values of τ_{EBR} throughout the range $[0, 1]$.

Our Synchronization Policy:

Our page synchronization scheme is parameterized by a scoring function $S()$. While our approach is compatible with a wide variety of possible scoring functions, for our experiments we needed to use a specific scoring method. Since no standard exists, we used two well-accepted methods that we feel constitute two extremes among the spectrum of options: (1) the well-known **TF-IDF** metric [97], using the variant employed in the popular Lucene software [59], and (2) **inlink count** obtained by

querying Google, which we used as a surrogate for PageRank [81] due to lack of adequate data (it has been suggested that inlink count and PageRank yield similar results [106]). In both cases the result of a query consists of a list of all pages that contain every term in the query, arranged in descending order of score.

In our page synchronization scheme, each page p is assigned an associated priority value $P(p,t)$, which may vary over time. Page redownloading is scheduled according to priority. The priority of a page is set equal to the expected change in repository quality of that page is redownloaded, as estimated by extrapolating from past measurements of this quantity taken during previous redownloads of the same page. These measurements are obtained using the estimation procedure of Section 4.6.

A variety of extrapolation methods can be used. The option we selected for our experiments is as follows. Given a set $\mathcal{R}(p)$[5] of time instants of past redownloads of page p, let:

$$\delta Q_A(p) = \frac{1}{|\mathcal{R}(p)|} \sum_{t \in \mathcal{R}(p)} \frac{\Delta Q_A(p,t)}{t - LR(p,t)}$$

where $LR(p,t)$ denotes the time of the most recent redownload of page p prior to t. Set $P(p,t) = \delta Q_A(p) \cdot (t - LR(p,t))$.

4.7.2 Estimation of Page Change Characteristics

Each of the page synchronization schemes we consider relies on forecasting of Web page change behavior based on behavior observed in the past. In particular, for each page p *SBR* requires a Poisson change rate parameter $\lambda(p)$, *EBR* requires a query irrelevance probability parameter $d(p)$, and our synchronization policy requires a time-normalized quality change value $\delta Q_A(p)$. We opted against splitting our data sets to perform parameter fitting and evaluation over different portions (say, 24 weeks each), because shortening our somewhat short 48-week data any further would make it difficult to obtain reliable performance measurements. Plus, in this chapter we do not focus on the forecasting problem, and we seek to compare all three methods on equal footing, independent of the forecasting method used. Therefore, for all three policies we used the entire 48-week data set

[5] We envision that in a real deployment the set $\mathcal{R}(p)$ would be determined based on a sliding window of recent redownloads of page p. (Other heuristics for favoring recent observations, such as exponentially-decayed averaging, warrant investigation as well; we leave this topic as future work.)

to estimate the necessary parameter for each page p.[6]

Still, we wanted to check that quality change values $\delta Q_A(p)$ are amenable to forecasting based on past measurements. For this purpose we estimated $\delta Q_A(p)$ values (using our approximation method of Section 4.6.1) for each page, once over the first 24 weeks of our data set, and again over the second 24 weeks, under the scenario in which every update to a page triggers an immediate redownload. We then compared the $\delta Q_A(p)$ estimates across the two 24-week periods. Figure 4.3 shows the outcome, for each of our two data sets under each of the two scoring functions we tested. In each graph, $\delta Q_A(p)$ over weeks $1-24$ is plotted on the x-axis, and $\delta Q_A(p)$ over weeks $25-48$ is plotted on the y-axis. Each dot in each graph corresponds to one page. When $\delta Q_A(p) = 0$, that indicates no change in repository quality due to updates to page p. Beyond that the scale of the axes is immaterial (since we are not measuring normalized quality). Each graph is plotted on a log-log scale, with pages with a value of 0 for one of the two $\delta Q_A(p)$ measurements inserted artificially along the edges. Pages with $\delta Q_A(p) = 0$ for weeks $1-24$ as well as weeks $25-48$ are not plotted (hence these graphs present a conservative view of amenability to forecasting). Dots are colored according to quantiles of proximity to the diagonal; see the key below the graphs. Points that are close to the diagonal ($y = x$ line) correspond to pages whose $\delta Q_A(p)$ values remain fairly consistent in both halves of the data set, implying that they can be forecasted accurately at this time-scale based on past measurements. These findings are in accord with those presented in [80], which assessed amenability to forecasting of the Web page change characteristics as measured by TF-IDF cosine similarity directly.

4.7.3 Comparison of Page Synchronization Schemes

We compared the three page synchronization schemes (*SBR*, *EBR*, and our crawling policy) using our repository quality metric which, as we have argued, we believe serves as a suitable metric for evaluating a crawler serving a search engine. Of course, crawling can also be used for other purposes (archival, mining, etc.), in which case our metric is not appropriate. For the purpose of evaluating the performance of a synchronization scheme we applied the precise formula for repository quality (Equation 4.5), and did not rely on any approximation techniques.

For this experiment we provided each synchronization scheme with a fully synchronized repository at week 1, and then allowed a fixed number of pages, B, to be redownloaded every week

[6] Note that for *SBR* and *EBR*, different settings for the shingles threshold τ_{SBR} (τ_{EBR}, respectively) result in potentially different $\lambda(p)$ ($d(p)$) values, which is precisely the purpose of varying the threshold.

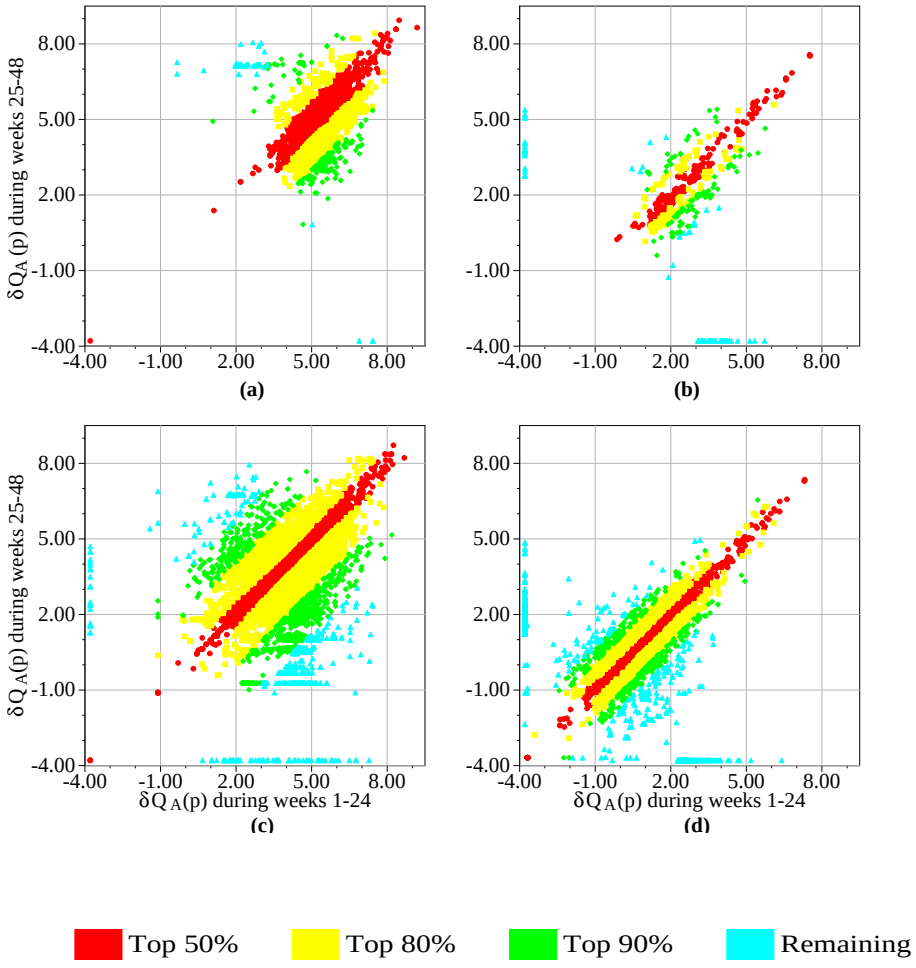

Figure 4.3: *Amenability to forecasting of time-normalized change in quality ($\delta Q_A(p)$). The four graphs shown correspond to* **(a)** *BDS data set with TF-IDF scoring function,* **(b)** *BDS with inlink count scoring function,* **(c)** *MDS data set with TF-IDF, and* **(d)** *MDS with inlink count. All graphs are on a log-log scale.*

for the remaining 47 weeks. We compared page synchronization schemes in terms of the resource requirement (B value) necessary to achieve a certain level of repository quality according to our metric, for two different scoring functions, TF-IDF and inlink count, over each of our two data sets, BDS and MDS. The results are plotted in Figure 4.4. In each graph, repository quality is plotted on the x-axis, and the resource requirement B is plotted on the y-axis. For each of *SBR* and *EBR*, for each B value the best repository quality level obtained using shingle threshold values

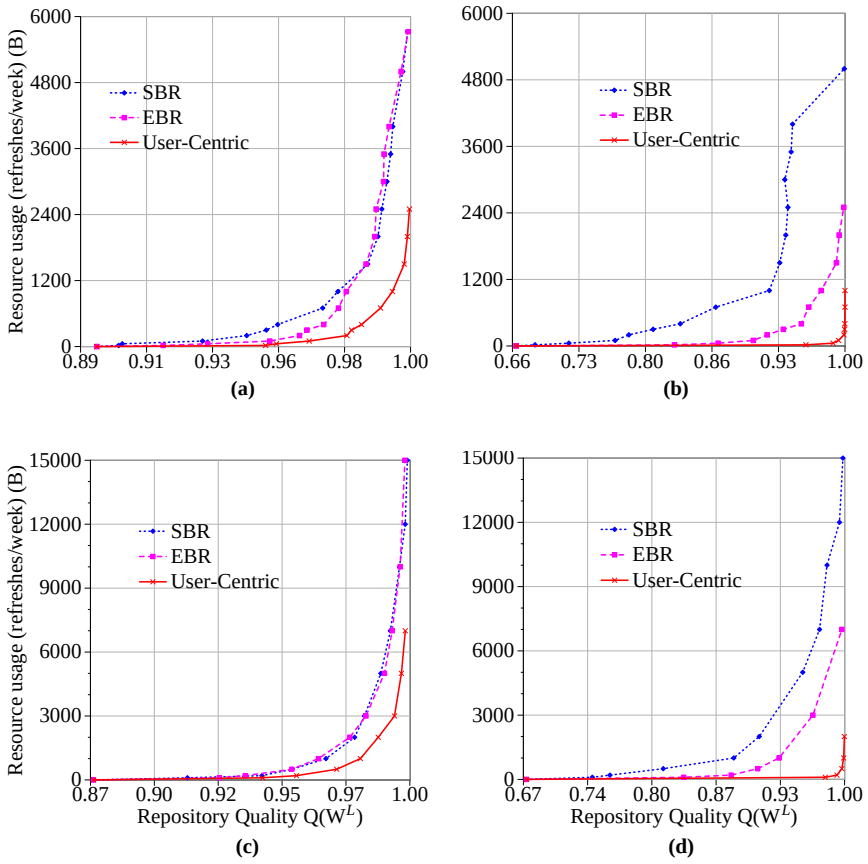

Figure 4.4: *Repository quality versus resource usage. The different graphs are for* **(a)** *BDS data set with TF-IDF scoring function,* **(b)** *BDS with inlink count scoring function,* **(c)** *MDS data set with TF-IDF, and* **(d)** *MDS with inlink count.*

$\tau \in \{0.1, 0.2, \ldots, 0.9, 1.0\}$ is plotted. For both data sets and both scoring functions, our page synchronization scheme requires substantially fewer resources to achieve the same level of repository quality than either of *SBR* and *EBR*.

We highlight the primary underlying reasons for this result using the following two examples taken from our boston.com data set:

Example 1: Figure 4.5(a) shows an advertisement added to a Web page in the boston.com data set. As it turned out, although the new advertisement consists of a large textual segment, none of the terms in the advertisement match frequently-issued queries in the AltaVista query workload. Hence, from the perspective of our repository quality metric it is not important to capture the content of

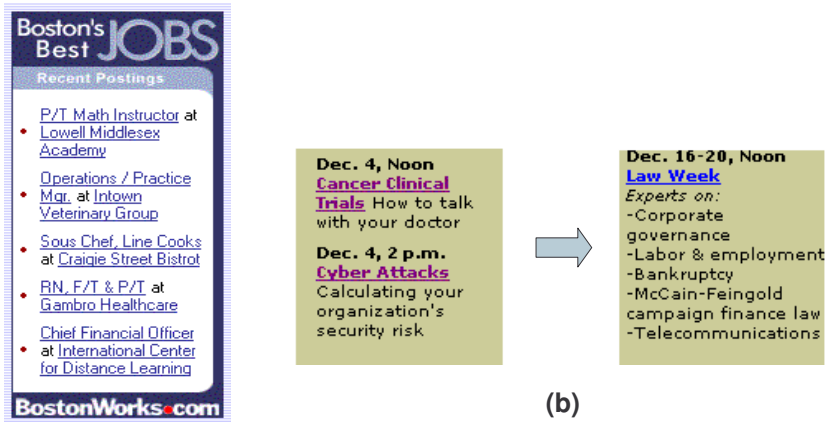

Figure 4.5: *Examples drawn from our real-world boston.com data set.*

the advertisement. Consequently our page synchronization scheme did not devote resources to redownloading this page (which turned out not to be updated in any way other than changing of advertising material), leaving more resources available for other tasks. This example illustrates that heuristics for estimating the importance of an update based on the number of words that change do not always work well.

Example 2: Figure 4.5(b) shows a portion of a Web page containing seminar announcements, that was updated to remove outdated announcements and replace them with a new announcement of an upcoming law seminar series. If this page is not redownloaded in a timely fashion, users querying for, say "Boston campaign finance" would not see this page among the query results even though it should appear (and be placed at a good rank position under at least some scoring functions). Our repository quality metric is particularly good at characterizing the importance of keeping this page up to date in the repository, by noting the high degree of match between frequent queries and evolving content (for example, the query "cancer" occurs frequently in the AltaVista query workload). This example illustrates (1) the importance of accounting for false negatives as well as false positives, and (2) that certain frequently-updated pages merit the devotion of precious redownloading resources, if it is the case that the updates tend to have a large impact on the user experience.

One may be inclined to suppose that, say, 95% repository quality is sufficient, and that there is no need to shoot for quality values very close to 100%. However, the difference between 95% and 99% repository quality can have a significant impact on the user experience. In fact, we came across Example 2 by examining a scenario in which *SBR* and *EBR* each achieved $\sim 95\%$ quality, whereas our scheme attained over 99% quality under the same resource constraint. Both *SBR* and *EBR* neglected to redownload this important seminar announcement page, leading to a substantial degradation in the quality of search results for a large number of (simulated) users.

4.8 Chapter Summary

In this chapter we studied the problem of synchronizing search engines' local repository. We introduced our repository quality metric which takes user experience into account, in terms of which queries are issued, with what frequency, and which results are inspected by users. Using this metric we formulated the synchronization task as an optimization problem, where the unknown parameter is the update behavior of pages. For the offline problem (*i.e.*, when the update behavior of pages is known) we proposed a synchronization policy which computes the benefit of redownloading each page, measured in terms of repository quality, and then schedules redownloading tasks accordingly.

For the online problem we showed that the benefit of redownloading a particular page is amenable to prediction based on measurements of the benefit of downloading the page in the past. We devised an efficient method for taking these measurements that is tightly integrated with the process of updating an inverted index maintained over the repository and incurs little additional overhead. Lastly we compared our page synchronization scheme against prior schemes empirically using real Web data. Our results demonstrate that our scheme requires substantially fewer resources to achieve the same user experience quality, leaving more resources for other important tasks such as discovering and downloading new pages.

Note that we studied the discovery task under the objective of maximizing the number of newly discovery pages, while for the synchronization task our goal was to redownload known pages so as to maximize a user-centric metric of repository quality. A part of our future work is to optimize the discovery and synchronization tasks together, possibly under one objective, since both tasks share the same crawling resources. We discuss future work in detail in Chapter 7.2.

Chapter 5

Web Page Ranking

Recall from Figure 1.4 that search engines perform two main tasks: content acquisition and presentation. In the previous two chapters we studied content acquisition. In this chapter and the next, we focus on content presentation. Content presentation involves arranging the relevant pages and ads on the slates of search queries submitted by users. In this chapter we focus on the presentation of Web pages while postponing the presentation of ads to the next chapter.

5.1 Introduction

As discussed in Chapter 4, search engines apply a scoring function to arrange pages on the slate of a query. Ideally, the scoring function should compute some intrinsic measure of usefulness or *quality* of pages that are relevant to a given query. Quality cannot be measured directly. However, various notions of *popularity*, such as number of in-links, PageRank [81], number of visits, etc., can be measured. Most Web search engines assume that popularity is closely correlated with quality, and rank results according to popularity.

Unfortunately, the correlation between popularity and quality is very weak for newly-created pages that have few visits and/or in-links. Worse, the process by which new, high-quality pages accumulate popularity is actually inhibited by search engines. Since search engines always lists highly popular pages at the top, and because users usually focus their attention on the top few results [61, 69], newly-created but high-quality pages are "shut out." This increasing "entrenchment

effect" has witnessed broad commentary across political scientists, the popular press, and Web researchers [43, 53, 57, 77, 100, 110] and even led to the term *Googlearchy*. In a recent study, Cho and Roy [28] show that heavy reliance on a search engine that ranks results according to popularity can delay widespread awareness of a high-quality page by a factor of over 60, compared with a simulated world without a search engine in which pages are accessed through browsing alone.

Even if we ignore the (contentious) issue of fairness, treating popularity as a surrogate for quality is unacceptable since it negatively affects the goal for which search engines are designed, *i.e.*, help users in finding high-quality content. Assuming a notion of intrinsic page quality as perceived by users, a hypothetical ideal search engine would bias users toward visiting those pages of the highest quality at a given time, regardless of popularity. On the other hand, a search engine which treats popularity as a surrogate for quality sets up a vicious cycle of neglect for new pages, and make entrenched pages collect an increasing fraction of user attention. Given that some of these new pages will generally have higher quality than some entrenched pages, the popularity-based search engine clearly fails to maximize an objective based on average quality of search results seen by users. We formalize this in Section 5.4.

5.1.1 Entrenchment Effect in Other Contexts

The entrenchment effect may not be unique to the Web search engine context. For example, consider recommendation systems [67], which are widely used in e-commerce [107]. Many users decide which items to view based on recommendations, but these systems make recommendations based on user evaluations of items they view. This circularity leads to the well-known *cold-start* problem, and is also likely to lead to entrenchment.

Indeed, Web search engines can be thought of as recommendation systems that recommend Web pages. The entrenchment effect is particularly acute in the case of Web search, because the sheer size of the Web forces large numbers of users to locate new content using search engines alone.

5.1.2 Overview of Our Approach

We propose a very simple modification to the method of ranking search results according to popularity: promote a small fraction of unexplored pages up at random rank positions in the result list. A new page now has some chance of attracting clicks and attention even if the initial popularity of

the page is very small. If a page has high quality, the rank boost gives the page a chance to prove itself. (Detailed definitions and algorithms are given later in the chapter.)

Still, the question remains as to how aggressively one should promote new pages. Many new pages on the Web are not of high quality. Therefore, the extent of rank promotion has to be limited very carefully, lest we negate the benefits of popularity-based ranking by displacing pages known to be of high quality too often. With rank promotion there is an inherent tradeoff between *exploration* of new pages and *exploitation* of pages already known to be of high quality. We study how to balance these two aspects, in the context of an overarching objective of maximizing the average quality of search results viewed by users, amortized over time. In particular we seek to answer the following questions:

- Which pages should be treated as candidates for exploration, i.e., included in the rank promotion process so as to receive transient rank boosts?
- Which pages, if any, should be exploited unconditionally, i.e., protected from any rank demotion caused by promotion of other pages?
- What should be the overall ratio of exploration to exploitation?

Before we can begin to address these questions, we must model the relationship between user queries and search engine results. We categorize the pages on the Web into disjoint groups by *topic*, such that each page pertains to exactly one topic. Let $\mathcal{P}$ be the set of pages devoted to a particular topic T (e.g., "swimming" or "Linux"), and let $\mathcal{U}$ denote the set of users interested in topic T. We say that the users $\mathcal{U}$ and pages $\mathcal{P}$ corresponding to topic T, taken together make up a *Web community*. (Users may participate in multiple communities.) For now we assume all users access the Web uniquely through a (single) search engine. (We relax this assumption later in Section 5.9.) We further assume a one-to-one correspondence between queries and topics, so that each query returns exactly the set of pages for the corresponding community. Although far from perfect, we believe this model preserves the essence of the dynamic process we seek to understand.

Communities are likely to differ a great deal in terms of factors like the number of users, the number of pages, the rate at which users visit pages, page lifetimes, etc. These factors play a significant role in determining how a given rank promotion scheme influences page popularity evolution. For example, communities with very active users are likely to be less susceptible to the entrenchment effect than those whose users do not visit very many pages. Consequently, a given rank promotion

scheme is bound to create quite different outcomes in the two types of communities. Hence, we provide an analytical method (in Section 5.6) for predicting the effect of deploying a particular randomized rank promotion scheme in a given community, as a function of the most important high-level community characteristics.

5.1.3 Experimental Study

We seek to model a very complex dynamical system involving search engines, evolving pages, and user actions, and trace its trajectory in time. It is worth emphasizing that even if we owned the most popular search engine in the world, "clean-room" experiments would be impossible. We could not even study the effect of different choices of a parameter, because an earlier choice would leave large-scale and indelible artifacts on the Web graph, visit rates, and popularity of certain pages. Therefore, analysis and simulations are inescapable, and practical experiments (as in Section 5.10) must be conducted in a sandbox.

Through a combination of analysis and simulation, we arrive at a particular recipe for randomized rank promotion that balances exploration and exploitation effectively, and yields good results across a broad range of community types. Robustness is desirable because, in practice, communities are not disjoint and therefore their characteristics cannot be measured reliably.

5.2 Chapter Outline

In Section 5.4 we formalize the problem of ranking Web pages. In particular, we present our model of Web page popularity, describe the exploration/exploitation tradeoff as it exists in our context, and introduce two metrics for evaluating rank promotion schemes. We then propose our randomized rank promotion method in Section 5.5, and supply an analytical model of page popularity evolution under randomized rank promotion in Section 5.6. In Sections 5.7–5.9 we present extensive analytical and simulation results, and recommend and evaluate a robust recipe for randomized rank promotion. In Section 5.10 we describe a real-world study that we conducted to test the effectiveness of rank promotion.

5.3 Related Work

The entrenchment effect has been attracting attention for several years [43, 53, 57, 77, 100, 110], but formal models for and analysis of the impact of search engines on the evolution of the Web graph [10, 76] or on the time taken by new pages to become popular [28] are recent.

A few solutions to the entrenchment problem have been proposed [7, 29, 121]. They rely on variations of PageRank: the solutions of [7, 121] assign an additional weighting factor based on page age; that of [29] uses the derivative of PageRank to forecast future PageRank values for young pages.

Our approach, randomized rank promotion, is quite different in spirit. The main strength of our approach is its simplicity—it does not rely on measurements of the age or PageRank evolution of individual Web pages, which are difficult to obtain and error-prone at low sample rates. (Ultimately, it may make sense to use our approach in conjunction with other techniques, in a complementary fashion.)

The exploration/exploitation tradeoff that arises in our context is akin to problems studied in the bandit theory [12]. However, direct application of bandit models to Web page ranking poses some challenges, as described next. Search engines commonly employ one or more of following three kinds of metrics to estimate quality of pages: (a) *link-based* metric is based on the number of inlinks/outlinks of a page, (b) *click-based* metric is based on the number of clicks that users make to a page when it is displayed in search results, while (c) *visit-based* metric uses the number of visits that a page attracts either through a search engine or from somewhere else.

Link-based is the most established metric for Web page ranking currently. The difficulty in applying bandit models under this metric is that when the search engine shows a page to a user (which corresponds to taking an action in the bandit model), the user feedback or assessment of the displayed page is not received immediately. In particular, if the user likes the page displayed by the search engine, then he/she creates a link to the page and this gives the search engine a positive feedback for the page. But this feedback comes after a significant delay, for two reasons: (a) the user takes time to create a link to the page after it was shown to her and (b) the search engine takes time to crawl pages and discover this newly created link. The lack of immediate feedback is a departure from the conventional bandit problem which assumes an immediate feedback/reward.

Click/visit-based metrics also suffer from this delayed feedback problem, though the problem is not as severe under them as it is under the link-based metric. More specifically, for visit/click-based quality metrics the user feedback is received in the form of visits/clicks which may take a while

to be captured. However, this delay is generally fairly small (in comparison to the delay involved in the link-based metric) and so, the search engines which rely heavily on these metrics are better suited for using bandit models. So that our work is applicable to the link-based metric, we propose a different approach than bandit model in this chapter. There is some work on bandit model with delayed responses [34, 35, 111]; we leave studying them as future work (more details in Chapter 7.3).

5.4 Problem Formulation

In this section we formalize the problem of ranking Web pages. First we introduce the model of Web page popularity, adopted from [28], that we use in the rest of this chapter. (For convenience, a summary of the notation we use is provided in Table 5.1.) Recall from Section 5.1.2 that in our model the Web is categorized into disjoint groups by topic, such that each page pertains to exactly one topic. Let $\mathcal{P}$ be the set of pages devoted to a particular topic T, and let $\mathcal{U}$ denote the set of users interested in topic T. Let $n = |\mathcal{P}|$ and $u = |\mathcal{U}|$ denote the number of pages and users, respectively, in the community.

5.4.1 Page Popularity

In our model, time is divided into discrete intervals, and at the end of each interval the search engine measures the popularity of each Web page according to in-link count, PageRank, user traffic, or some other indicator of popularity among users. Usually it is only possible to measure popularity among a minority of users. Indeed, for in-link count or PageRank, only those users who have the ability to create links are counted. For metrics based on user traffic, typically only users who agree to install a special toolbar that monitors Web usage, as in [4], are counted. Let $\mathcal{U}_m \subseteq \mathcal{U}$ denote the set of *monitored users*, over which page popularity is measured, and let $m = |\mathcal{U}_m|$. We assume $\mathcal{U}_m$ constitutes a representative sample of the overall user population $\mathcal{U}$.

Let the total number of user visits to pages per unit time be fixed at v_u. Further, let v denote the number of visits per unit time by monitored users, with $v = v_u \cdot \frac{m}{u}$. The way these visits are distributed among pages in $\mathcal{P}$ is determined largely by the search engine ranking method in use; we will come back to this aspect later. For now we simply provide a definition of the visit rate of a page $p \in \mathcal{P}$.

Symbol	Meaning
$\mathcal{P}$	Set of Web pages in community
n	$= \|\mathcal{P}\|$
$\mathcal{U}$	Set of users in community
u	$= \|\mathcal{U}\|$
$\mathcal{U}_m$	Set of monitored users in community
m	$= \|\mathcal{U}_m\|$
$P(p,t)$	Popularity among monitored users of page p at time t
$V_u(p,t)$	Number of user visits to page p during unit time interval at t
$V(p,t)$	Number of visits to p by monitored users at t
v_u	Total number of user visits per unit time
v	Number of visits by monitored users per unit time
$A(p,t)$	Awareness among monitored users of page p at time t
$Q(p)$	Intrinsic quality of page p
l	Expected page lifetime

Table 5.1: *Notation used in this chapter.*

Definition 5.4.1 *(Visit Rate) The visit rate of page p at time t, $V(p,t)$, is defined as the number of times p is visited by any monitored user within a unit time interval at time t.*

Similarly, let $V_u(p,t)$ denote the number of visits by any user in $\mathcal{U}$ (monitored and unmonitored users alike) within a unit time interval at time t. We require that $\forall t, \sum_{p\in\mathcal{P}} V_u(p,t) = v_u$ and $\forall t, \sum_{p\in\mathcal{P}} V(p,t) = v$. Once a user visits a page for the first time, she becomes "aware" of that page.

Definition 5.4.2 *(Awareness) The awareness level of page p at time t, $A(p,t)$, is defined as the fraction of monitored users who have visited p at least once by time t.*

We define the popularity of page p at time t, $P(p,t) \in [0,1]$, as follows:

$$P(p,t) = A(p,t) \cdot Q(p) \tag{5.1}$$

where $Q(p) \in [0,1]$ (*page quality*) denotes the extent to which an average user would "like" page p

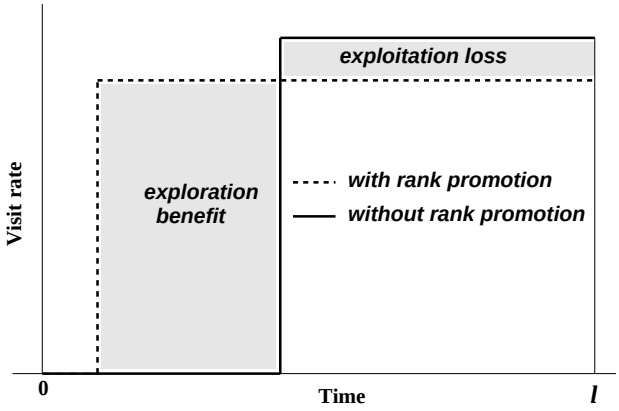

Figure 5.1: *Exploration/exploitation tradeoff.*

if she were aware of p.

In our model page popularity is a monotonically nondecreasing function of time. Therefore if we assume nonzero page viewing probabilities, for a page of infinite lifetime $\lim_{t\to\infty} P(p,t) = Q(p)$.

5.4.2 Metrics and Exploration/Exploitation Tradeoff

Time-To-Become-Popular Metric: If pages are ranked strictly according to current popularity, it can take a long time for the popularity of a new page to approach its quality. Artificially promoting the rank of new pages can potentially accelerate this process. One important objective for rank promotion is to minimize the time it takes for a new high-quality page to attain its eventual popularity, denoted *TBP* for "time to become popular." In this chapter we measure TBP as the time it takes for a high-quality page to attain popularity that exceeds 99% of its quality level.

Figure 5.1 shows popularity evolution curves for a particular page having very high quality created at time 0 with lifetime l, both with and without rank promotion. (It has been shown [28] that popularity evolution curves are close to step-functions.) Time is plotted on the x-axis. The y-axis plots the number of user visits per time unit. Note that while the page becomes popular earlier when rank promotion is applied, the number of visits it receives once popular is somewhat lower than in the case without rank promotion. That is because systematic application of rank promotion inevitably comes at the cost of fewer visits to already-popular pages.

Quality-Per-Click Metric and Exploration/Exploitation Tradeoff: The two shaded regions

of Figure 5.1 indicate the positive and negative aspects of rank promotion. The *exploration benefit* area corresponds to the increase in the number of additional visits to this particular high-quality page during its lifetime made possible by promoting it early on. The *exploitation loss* area corresponds to the decrease in visits due to promotion of other pages, which may mostly be of low quality compared to this one. Clearly there is a need to balance these two factors. The TBP metric is one-sided in this respect, so we introduce a second metric that takes into account both exploitation and exploitation: *quality-per-click*, or QPC for short. QPC measures the average quality of pages viewed by users, amortized over a long period of time. We believe that maximizing QPC is a suitable objective for designing a rank promotion strategy.

We now derive a mathematical expression for QPC in our model. First, recall that the number of visits by any user to page p during time interval t is denoted $V_u(p,t)$. We can express the cumulative quality of all pages in $\mathcal{P}$ viewed at time t as $\sum_{p\in\mathcal{P}} V_u(p,t)\cdot Q(p)$. Taking the average across time in the limit as the time duration tends to infinity, we obtain:

$$\lim_{t\to\infty} \sum_{t_l=0}^{t} \sum_{p\in\mathcal{P}} \big(V_u(p,t_l)\cdot Q(p)\big)$$

By normalizing, we arrive at our expression for QPC:

$$QPC \quad = \quad \lim_{t\to\infty} \frac{\sum_{t_l=0}^{t} \sum_{p\in\mathcal{P}} \big(V_u(p,t_l)\cdot Q(p)\big)}{\sum_{t_l=0}^{t} \big(\sum_{p\in\mathcal{P}} V_u(p,t_l)\big)} \tag{5.2}$$

5.4.3 Web Page Ranking Optimization Problem

Formally, the problem of ranking pages poses the following optimization problem:

- *Objective:* Maximize the average quality of results perceived by users (QPC), given in Equation 5.2.
- *Constraint:* Users pay limited attention to search results (*i.e.*, bounded visit rate $V(p,t)$).
- *Uncertainty:* Page quality values (*i.e.*, $Q(p)$'s) are unknown.

The offline problem, *i.e.*, when page quality values are known, has a simple solution in this case: rank the pages in decreasing order of quality. Doing so maximizes the QPC expression in Equation 5.2 because visit rate $V_u(p,t)$ decreases as the rank of page p in the ranked list increases

(from Figure 1.3; details in Section 5.6.3). So, when pages are ranked in decreasing order of quality, the maximum quality page gets the top rank and attracts the highest visit rate, and so on.

Next we focus on the online problem where the page quality values are unknown.

5.5 Randomized Rank Promotion

We now describe our simple randomized rank promotion scheme (this description is purely conceptual; more efficient implementation techniques exist).

Let $\mathcal{P}$ denote the set of n responses to a user query. A subset of those pages, $\mathcal{P}_p \subseteq \mathcal{P}$ is set aside as the *promotion pool*, which contains the set of pages selected for rank promotion according to a predetermined rule. (The particular rule for selecting $\mathcal{P}_p$, as well as two additional parameters, $k \geq 1$ and $r \in [0, 1]$, are configuration options that we discuss shortly.) Pages in $\mathcal{P}_p$ are sorted randomly and the result is stored in the ordered list $\mathcal{L}_p$. The remaining pages $(\mathcal{P} - \mathcal{P}_p)$ are ranked in the usual deterministic way, in descending order of popularity; the result is an ordered list $\mathcal{L}_d$. The two lists are merged to create the final result list $\mathcal{L}$ according to the following procedure:

1. The top $k-1$ elements of $\mathcal{L}_d$ are removed from $\mathcal{L}_d$ and inserted into the beginning of $\mathcal{L}$ while preserving their order.

2. The element to insert into $\mathcal{L}$ at each remaining position $i = k, k+1, \ldots, n$ is determined one at a time, in that order, by flipping a biased coin: with probability r the next element is taken from the top of list $\mathcal{L}_p$; otherwise it is taken from the top of $\mathcal{L}_d$. If one of $\mathcal{L}_p$ or $\mathcal{L}_d$ becomes empty, all remaining entries are taken from the nonempty list. At the end both of $\mathcal{L}_d$ and $\mathcal{L}_p$ will be empty, and $\mathcal{L}$ will contain one entry for each of the n pages in $\mathcal{P}$.

The configuration parameters are:

- **Promotion pool ($\mathcal{P}_p$):** We consider two rules for determining which pages are promoted: (a) the *uniform* promotion rule, in which every page is included in $\mathcal{P}_p$ with equal probability r, and (b) the *selective* promotion rule, in which all pages whose current awareness level among monitored users is zero (i.e., $\mathcal{A}(p, t) = 0$) are included in $\mathcal{P}_p$, and no others. (Other rules are of course possible; we chose to focus on these two in particular because they roughly correspond to the extrema of the spectrum of interesting rules.)

- **Starting point (k):** All pages whose natural rank is better than k are protected from the effects of promoting other pages. A particularly interesting value is $k = 2$, which safeguards the top result of any search query, thereby preserving the "feeling lucky" property that is of significant value in some situations.
- **Degree of randomization (r):** When k is small, this parameter governs the tradeoff between emphasizing exploration (large r) and emphasizing exploitation (small r).

Our goal is to determine settings of the above parameters that lead to good TBP and QPC values. The remainder of this chapter is dedicated to this task. Next we present our analytical model of Web page popularity evolution, which we use to estimate TBP and QPC under various ranking methods.

5.6 Analytical Model

Our analytical model has these features:

- Pages have finite lifetime following an exponential distribution (Section 5.6.1). The number of pages and the number of users are fixed in steady state. The quality distribution of pages is stationary.
- The expected awareness, popularity, rank, and visit rate of a page are coupled to each other through a combination of the search engine ranking function and the bias in user attention to search results (Sections 5.6.2 and 5.6.3).

Given that (a) modern search engines appear to be strongly influenced by popularity-based measures while ranking results, and (b) users tend to focus their attention primarily on the top-ranked results [61, 69], it is reasonable to assume that the expected visit rate of a page is a function of its current popularity (as done in [28]):

$$V(p,t) \quad = \quad F(P(p,t)) \tag{5.3}$$

where the form of function $F(x)$ depends on the ranking method in use and the bias in user attention. For example, if ranking is completely random, then $V(p,t)$ is independent of $P(p,t)$ and the same for all pages, so $F(x) = v \cdot \frac{1}{n}$. (Recall that v is the total number of monitored user visits per unit time.)

If ranking is done in such a way that user traffic to a page is proportional to the popularity of that page, $F(x) = v \cdot \frac{x}{\phi}$, where ϕ is a normalization factor; at steady-state, $\phi = \sum_{p \in \mathcal{P}} P(p,t)$. If ranking is performed the aforementioned way 50% of the time, and performed randomly 50% of the time, then $F(x) = v \cdot \left(0.5 \cdot \frac{x}{\phi} + 0.5 \cdot \frac{1}{n}\right)$. For the randomized rank promotion we introduced in Section 5.5 the situation is more complex. We defer discussion of how to obtain $F(x)$ to Section 5.6.3.

5.6.1 Page Birth and Death

The set of pages on the Web is not fixed. Likewise, we assume that for a given community based around topic T, the set $\mathcal{P}$ of pages in the community evolves over time due to pages being created and retired. To keep our analysis manageable we assume that the rate of retirement matches the rate of creation, so that the total number of pages remains fixed at $n = |\mathcal{P}|$. We model retirement of pages as a Poisson process with rate parameter λ, so the expected lifetime of a page is $l = \frac{1}{\lambda}$ (all pages have the same expected lifetime[1]). When a page is retired, a new page of equal quality is created immediately, so the distribution of page quality values is stationary. When a new page is created it has initial awareness and popularity values of zero.

5.6.2 Awareness Distribution

We derive an expression for the distribution of page awareness values, which we then use to obtain an expression for quality-per-click (QPC). We analyze the steady-state scenario, in which the awareness and popularity distributions have stabilized and remain steady over time. Our model may not seem to indicate steady-state behavior, because the set of pages is constantly in flux and the awareness and popularity of an individual page changes over time. To understand the basis for assuming steady-state behavior, consider the set $\mathcal{C}_t$ of pages created at time t, and the set $\mathcal{C}_{t+1}$ of pages created at time $t+1$. Since page creation is governed by a Poisson process the expected sizes of the two sets are equal. Recall that we assume the distribution of page quality values remains the same at all times. Therefore, the popularity of all pages in both $\mathcal{C}_t$ and $\mathcal{C}_{t+1}$ will increase from the starting value of 0 according to the same popularity evolution law. At time $t+1$, when the pages in $\mathcal{C}_t$ have evolved in popularity according to the law for the first time unit, the new pages in $\mathcal{C}_{t+1}$ introduced at time $t+1$ will replace the old popularity values of the $\mathcal{C}_t$ pages. A symmetric effect

[1] In reality, lifetime might be a positively correlated with popularity. If so, popular pages would remain entrenched for a longer time than under our model, leading to even worse TBP than our model predicts.

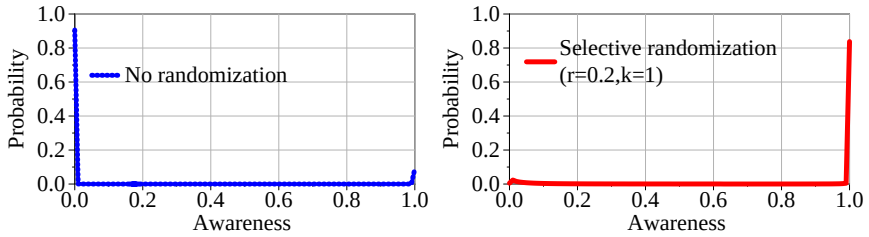

Figure 5.2: *Awareness distribution of pages of high quality under randomized and nonrandomized ranking.*

occurs with pages that are retired, resulting in steady-state behavior overall. In the steady-state, both popularity and awareness distributions are stationary.

The steady-state awareness distribution is given as follows.

Theorem 2 *Among all pages in $\mathcal{P}$ whose quality is q, the fraction that have awareness $a_i = \frac{i}{m}$ (for $i = 0, 1, \ldots, m$) is:*

$$f(a_i|q) = \frac{\lambda}{(\lambda + F(0)) \cdot (1 - a_i)} \prod_{j=1}^{i} \frac{F(a_{j-1} \cdot q)}{\lambda + F(a_j \cdot q)} \tag{5.4}$$

where $F(x)$ is the function in Equation 5.3.

Proof: See Appendix 9.2. ■

Figure 5.2 plots the steady-state awareness distribution for pages of highest quality, under both nonrandomized ranking and selective randomized rank promotion with $k = 1$ and $r = 0.2$, for our default Web community characteristics (see Section 5.7.1). For this graph we used the procedure described in Section 5.6.3 to obtain the function $F(x)$.

Observe that if randomized rank promotion is used, in steady-state most high-quality pages have large awareness, whereas if standard nonrandomized ranking is used most pages have very small awareness. Hence, under randomized rank promotion most pages having high quality spend most of their lifetimes with near-100% awareness, yet with nonrandomized ranking they spend most of their lifetimes with near-zero awareness. Under either ranking scheme pages spend very little time in the middle of the awareness scale, since the rise to high awareness is nearly a step function.

Given an awareness distribution $f(a|q)$, it is straightforward to determine expected time-to-become-popular (TBP) corresponding to a given quality value (formula omitted for brevity). Expected quality-per-click (QPC) is expressed as follows:

$$QPC = \frac{\sum_{p \in \mathcal{P}} \sum_{i=0}^{m} f(a_i|Q(p)) \cdot F(a_i \cdot Q(p)) \cdot Q(p)}{\sum_{p \in \mathcal{P}} \sum_{i=0}^{m} f(a_i|Q(p)) \cdot F(a_i \cdot Q(p))}$$

where $a_i = \frac{i}{m}$. (Recall our assumption that monitored users are a representative sample of all users.)

5.6.3 Popularity to Visit Rate Relationship

In this section we derive the function $F(x)$ used in Equation 5.3, which governs the relationship between $P(p,t)$ and the expectation of $V(p,t)$. As done in [28] we split the relationship between the popularity of a page and the expected number of visits into two components: (1) the relationship between popularity and rank position, and (2) the relationship between rank position and the number of visits. We denote these two relationships as the functions F_1 and F_2 respectively, and write:

$$F(x) \quad = \quad F_2(F_1(x))$$

where the output of F_1 is the rank position of a page of popularity x, and F_2 is a function from that rank to a visit rate. Our rationale for splitting F in this way is that, according to Figure 1.3, the likelihood of a user visiting a page presented in a search result list depends primarily on the rank position at which the page appears.

We begin with F_2, the dependence of the expected number of user visits on the rank of a page in a result list. As discussed in Chapter 4 the following relationship holds quite closely based on analysis of AltaVista usage logs [28, 69]:

$$F_2(x) \quad = \quad \theta \cdot x^{-3/2} \tag{5.5}$$

where θ is a normalization constant, which we set as:

$$\theta \quad = \quad \frac{v}{\sum_{i=1}^{n} i^{-3/2}}$$

where v is the total number of monitored user visits per unit time.

Next we turn to F_1, the dependence of rank on the popularity of a page. Note that since the awareness level of a particular page cannot be pinpointed precisely (it is expressed as a probability

distribution), we express $F_1(x)$ as the *expected* rank position of a page of popularity x. In doing so we compromise accuracy to some extent, since we will determine the expected number of visits by applying F_2 to the expected rank, as opposed to summing over the full distribution of rank values. (We examine the accuracy of our analysis in Sections 5.7.2 and 5.7.3.)

Under nonrandomized ranking, the expected rank of a page of popularity x is one plus the expected number of pages whose popularities surpass x. By Equation 5.1, page p has $P(p,t) > x$ if it has $A(p,t) > x/Q(p)$. From Theorem 2 the probability that a randomly-chosen page p satisfies this condition is:

$$\sum_{i=1+\lfloor m \cdot x/Q(p) \rfloor}^{m} f\left(\frac{i}{m}\middle| Q(p)\right)$$

By linearity of expectation, summing over all $p \in \mathcal{P}$ we arrive at:

$$F_1(x) \approx 1 + \sum_{p \in \mathcal{P}} \left(\sum_{i=1+\lfloor m \cdot x/Q(p) \rfloor}^{m} f\left(\frac{i}{m}\middle| Q(p)\right) \right) \tag{5.6}$$

(This is an approximate expression because we ignore the effect of ties in popularity values, and because we neglect to discount one page of popularity x from the outer summation.)

The formula for F_1 under uniform randomized ranking is rather complex, so we omit it. We focus instead on selective randomized ranking, which is a more effective strategy, as we will demonstrate shortly. Under selective randomized ranking the expected rank of a page of popularity x, when $x > 0$, is given by:

$$F_1'(x) \approx \begin{cases} F_1(x) & \text{if } F_1(x) < k \\ F_1(x) + \min\{\frac{r \cdot (F_1(x)-k+1)}{(1-r)}, z\} & \text{otherwise} \end{cases}$$

where F_1 is as in Equation 5.6, and z denotes the expected number of pages with zero awareness, an estimate for which can be computed without difficulty under our steady-state assumption.

The above expressions for $F_1(x)$ or $F_1'(x)$ each contain a circularity, because our formula for $f(a|q)$ (Equation 9.1) contains $F(x)$. It appears that a closed-form solution for $F(x)$ is difficult to obtain. In the absence of a closed-form expression one option is to determine $F(x)$ via simulation. The method we use is to solve for $F(x)$ using an iterative procedure, as follows.

We start with a simple function for $F(x)$, say $F(x) = x$, as an initial guess at the solution. We then substitute this function into the right-hand side of the appropriate equation above to produce

a new $F(x)$ function in numerical form. We then convert the numerical $F(x)$ function into symbolic form by fitting a curve, and repeat until convergence occurs.[2] (In each iteration we adjust the curve slightly so as to fit the extreme points corresponding to $x = 0$ and $x = 1$ especially carefully.) Interestingly, we found that using a quadratic curve in log-log space led to good convergence for all parameter settings we tested, so that:

$$\log F = \alpha \cdot (\log x)^2 + \beta \cdot \log x + \gamma$$

where α, β, and γ are determined using a curve fitting procedure. We later verified via simulation that across a variety of scenarios $F(x)$ can be fit quite accurately to a quadratic curve in log-log space.

5.7 Effect of Randomized Rank Promotion and Recommended Parameter Settings

In this section we report our measurements of the impact of randomized rank promotion on search engine quality. We begin by describing the default Web community scenario we use in Section 5.7.1. Then we report the effect of randomized rank promotion on TBP and QPC in Sections 5.7.2 and 5.7.3, respectively. Lastly, in Section 5.7.4 we investigate how to balance exploration and exploitation, and give our recommended recipe for randomized rank promotion.

5.7.1 Default Scenario

For the results we report in this chapter, the default[3] Web community we use is one having $n = 10,000$ pages. The remaining characteristics of our default Web community are set so as to be in proportion to observed characteristics of the entire Web, as follows. First, we set the expected page lifetime to $l = 1.5$ years (based on data from [80]). Our default Web community has $u = 1000$ users making a total of $v_u = 1000$ visits per day (based on data reported in [54], the number of Web users is roughly one-tenth the number of pages, and an average user queries a search engine about

[2] Though the convergence is not guaranteed to happen, in our experiments it always did.
[3] We supply results for other community types in Section 5.8.

once per day). We assume that a search engine is able to monitor 10% of its users, so $m = 100$ and $v = 100$.

As for page quality values, we had little basis for measuring the intrinsic quality distribution of pages on the Web. As the best available approximation, we used the power-law distribution reported for PageRank in [28], with the quality value of the highest-quality page set to 0.4. (We chose 0.4 based on the fraction of Internet users who frequent the most popular Web portal site, according to [99].)

5.7.2 Effect of Randomized Rank Promotion on TBP

Figure 5.3 shows popularity evolution curves derived from the awareness distribution determined analytically for a page of quality 0.4 under three different ranking methods: (1) nonrandomized ranking, (2) randomized ranking using uniform promotion with the starting point $k = 1$ and the degree of randomization $r = 0.2$, and (3) randomized ranking using selective promotion with $k = 1$ and $r = 0.2$. This graph shows that, not surprisingly, randomized rank promotion can improve TBP by a large margin. More interestingly it also indicates that selective rank promotion achieves substantially better TBP than uniform promotion. Because, for small r, there is limited opportunity to promote pages, focusing on pages with zero awareness turns out to be more effective than uniform promotion. (It is worth exploring a relaxed version of selective promotion method that focuses on zero awareness pages as well as pages of small non-zero awareness. We leave it as future work.)

Figure 5.4 shows TBP measurements for a page of quality 0.4 in our default Web community, for different values of r (fixing $k = 1$). As expected, increased randomization leads to lower TBP, especially if selective promotion is employed.

To validate our analytical model, we created a simulator that maintains an evolving ranked list of pages (the ranking method used is configurable), and distributes user visits to pages according to Equation 5.5. Our simulator keeps track of awareness and popularity values of individual pages as they evolve over time, and creates and retires pages as dictated by our model. After a sufficient period of time has passed to reach steady-state behavior, we take measurements. These results are plotted in Figure 5.4, side-by-side with our analytical results. We observe a close correspondence between our analytical model and our simulation.[4]

[4]Our analysis is only intended to be accurate for small values of r, which is why we only plot results for $r < 0.2$. From a practical standpoint only small values of r are of interest.

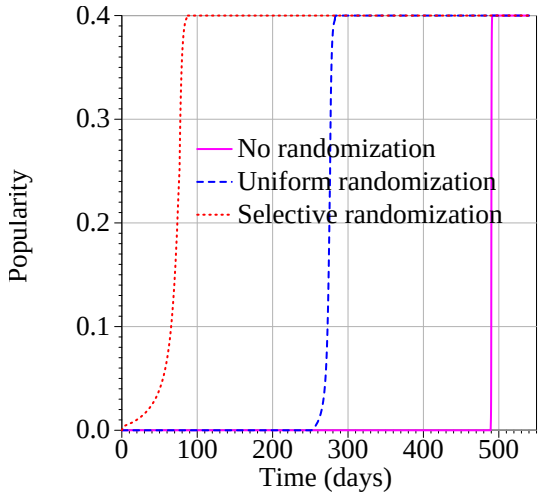

Figure 5.3: *Popularity evolution of a page of quality $Q = 0.4$ under nonrandomized, uniform randomized, and selective randomized ranking.*

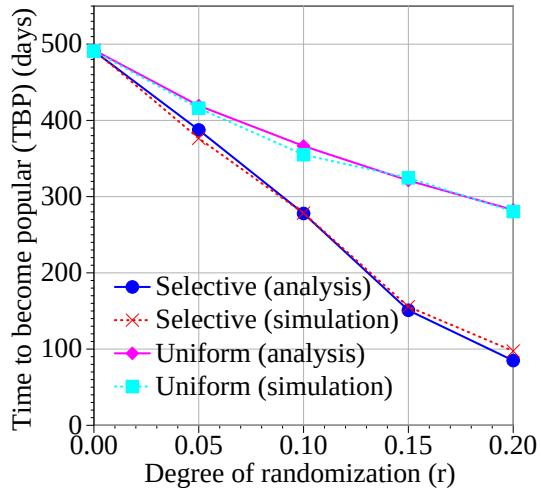

Figure 5.4: *Time to become popular (TBP) for a page of quality* 0.4 *in default Web community as degree of randomization (r) is varied.*

5.7.3 Effect of Randomized Rank Promotion on QPC

We now turn to quality-per-click (QPC). Throughout this chapter (except in Section 5.9) we normalize all QPC measurements such that $QPC = 1.0$ corresponds to the theoretical upper bound

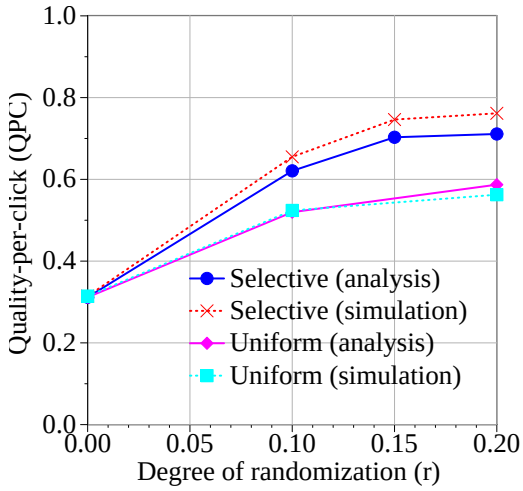

Figure 5.5: *Quality-per-click (QPC) for default Web community as degree of randomization (r) is varied.*

achieved by ranking pages in descending order of quality. The graph in Figure 5.5 plots normalized QPC as we vary the promotion rule and the degree of randomization r (holding k fixed at $k = 1$), under our default Web community characteristics of Section 5.7.1. For a community with these characteristics, a moderate dose of randomized rank promotion increases QPC substantially, especially under selective promotion.

5.7.4 Balancing Exploration, Exploitation, and Reality

We have established a strong case that selective rank promotion is superior to uniform promotion. In this section we investigate how to set the other two randomized rank promotion parameters, k and r, so as to balance exploration and exploitation and achieve high QPC. For this purpose we prefer to rely on simulation, as opposed to analysis, for maximum accuracy.

The graph in Figure 5.6 plots normalized QPC as we vary both k and r, under our default scenario (Section 5.7.1). As k grows larger, a higher r value is needed to achieve high QPC. Intuitively, as the starting point for rank promotion becomes lower in the ranked list (larger k), a denser concentration of promoted pages (larger r) is required to ensure that new high-quality pages are discovered by users.

For search engines, we take the view that it is undesirable to include a noticeable amount of randomization in ranking, regardless of the starting point k. Based on Figure 5.6, using only 10%

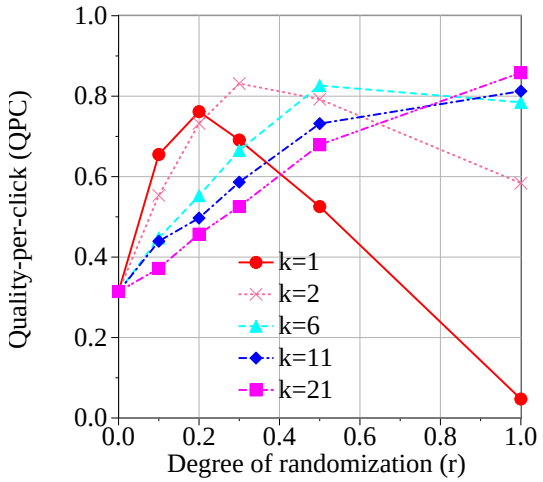

Figure 5.6: *Qualitiy-per-click (QPC) for default Web community under selective randomized rank promotion, as degree of randomization (r) and starting point (k) are varied.*

randomization ($r = 0.1$) appears sufficient to achieve most of the benefit of rank promotion, as long as k is kept small (e.g., $k = 1$ or 2).[5] Under 10% randomization, roughly one page in every group of ten query results is a new, untested page, as opposed to an established page. We do not believe most users are likely to notice this effect, given the amount of noise normally present in search engine results.

A possible exception is for the topmost query result, which users often expect to be consistent if they issue the same query multiple times. Plus, for certain queries users expect to see a single, "correct," answer in the top rank position (e.g., most users would expect the query "Carnegie Mellon" to return a link to the Carnegie Mellon University home page at position 1), and quite a bit of effort goes into ensuring that search engines return that result at the topmost rank position. That is why we include the $k = 2$ parameter setting, which ensures that the top-ranked search result is never perturbed.

Recommendation: *Introduce* 10% *randomization starting at rank position* 1 *or* 2*, and exclusively target zero-awareness pages for random rank promotion.*

[5] Note that $r = 0.2$ performs very well in Figure 5.6 but we prefer $r = 0.1$ over it so that the effect of randomization is small and not noticed by users.

5.8 Robustness Across Different Community Types

In this section we investigate the robustness of our recommended ranking method (selective promotion rule, $r = 0.1$, $k \in \{1, 2\}$) as we vary the characteristics of our testbed Web community. Our objectives are to demonstrate: (1) that if we consider a wide range of community types, amortized search result quality is never harmed by our randomized rank promotion scheme, and (2) that our method improves result quality substantially in most cases, compared with traditional deterministic ranking. In this section we rely on simulation rather than analysis to ensure maximum accuracy.

5.8.1 Influence of Community Size

Here we vary the number of pages in the community, n, while holding the ratio of users to pages fixed at $u/n = 10\%$, fixing the fraction of monitored users as $m/u = 10\%$, and fixing the number of daily page visits per user at $v_u/u = v/m = 1$. Figure 5.7 shows the result, with community size n plotted on the x-axis on a logarithmic scale. The y-axis plots normalized QPC for three different ranking methods: nonrandomized, selective randomized with $r = 0.1$ and $k = 1$, and selective randomized with $r = 0.1$ and $k = 2$. With nonrandomized ranking, QPC declines as community size increases, because it becomes more difficult for new high-quality pages to overcome the entrenchment effect. Under randomized rank promotion, on the other hand, due to rank promotion QPC remains high and fairly steady across a range of community sizes.

5.8.2 Influence of Page Lifetime

Figure 5.8 shows QPC as we vary the expected page lifetime l while keeping all other community characteristics fixed. (Recall that in our model the number of pages in the community remains constant across time, and when a page is retired a new one of equal quality but zero awareness takes its place.) The QPC curve for nonrandomized ranking confirms our intuition: when there is less churn in the set of pages in the community (large l), QPC is penalized less by the entrenchment effect. More interestingly, the margin of improvement in QPC over nonrandomized ranking due to introducing randomness is greater when pages tend to live longer. The reason is that with a low page creation rate the promotion pool can be kept small. Consequently new pages benefit from larger and more frequent rank boosts, on the whole, helping the high-quality ones get discovered quickly.

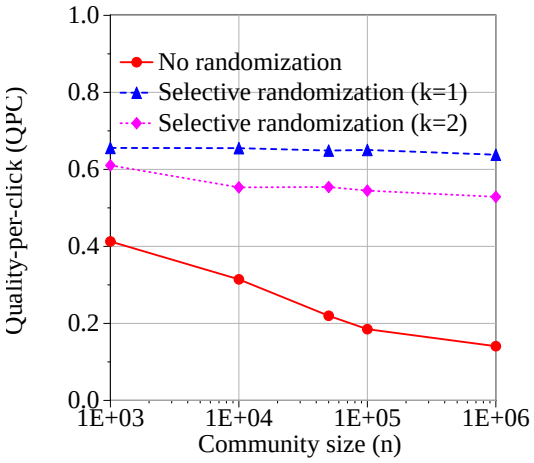

Figure 5.7: *Influence of community size.*

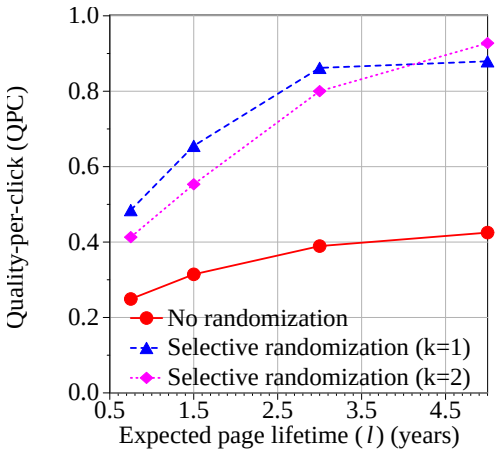

Figure 5.8: *Influence of page lifetime.*

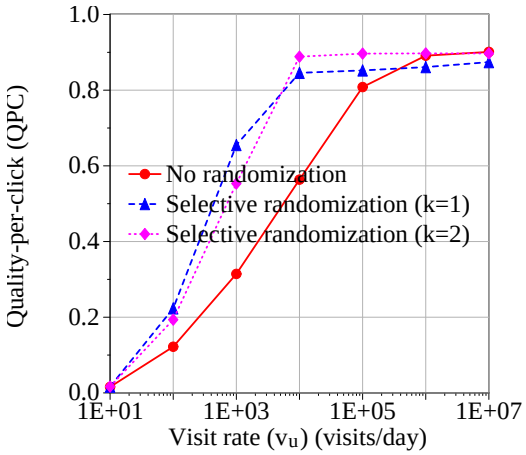

Figure 5.9: *Influence of visit rate.*

5.8.3 Influence of Visit Rate

The influence of the aggregate user visit rate on QPC is plotted in Figure 5.9. Visit rate is plotted on the x-axis on a logarithmic scale, and QPC is plotted on the y-axis. Here, we hold the number of pages fixed at our default value of $n = 10,000$ and use our default expected lifetime value of $l = 1.5$ years. We vary the total number of user visits per day v_u while holding the ratio of daily page visits to users fixed at $v_u/u = 1$ and, as always, fixing the fraction of monitored users as $m/u = 10\%$.

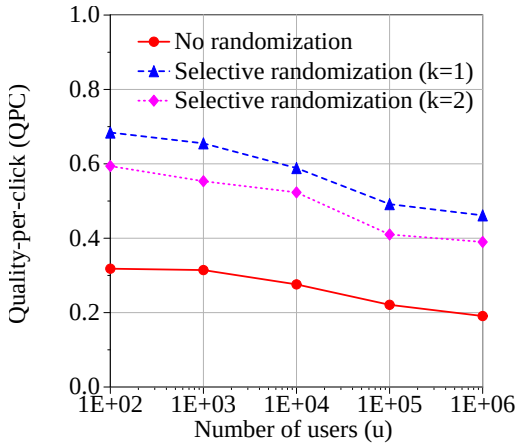

Figure 5.10: *Influence of size of user population.*

From Figure 5.9 we see first of all that, not surprisingly, popularity-based ranking fundamentally fails if very few pages are visited by users. Second, if the number of visits is very large (1000 visits per day to an average page), then there is no need for randomization in ranking (although it does not hurt much). For visit rates within an order of magnitude on either side of $0.1 \cdot n = 1000$, which matches the average visit rate of search engines in general when n is scaled to the size of the entire Web,[6] there is significant benefit to using randomized rank promotion.

5.8.4 Influence of Size of User Population

Lastly we study the affect of varying the number of users in the community u, while holding all other parameters fixed: $n = 10,000$, $l = 1.5$ years, $v_u = 1000$ visits per day, and $m/u = 10\%$. Note that we keep the total number of visits per day fixed, but vary the number of users making those visits. The idea is to compare communities in which most page visits come from a core group of fairly active users to ones receiving a large number of occasional visitors. Figure 5.10 shows the result, with the number of users u plotted on the x-axis on a logarithmic scale, and QPC plotted on the y-axis. All three ranking methods perform somewhat worse when the pool of users is large, although the performance ratios remain about the same. The reason for this trend is that with a larger user pool, a stray visit to a new high-quality page provides less traction in terms of overall awareness.

[6] According to our rough estimate based on data from [54].

5.9 Mixed Surfing and Searching

The model we have explored thus far assumes that users make visit to pages only by querying a search engine. While a very large number of surf trails start from search engines and are very short, nonnegligible surfing may still be occurring without support from search engines. We use the following model for mixed surfing and searching:

- While performing *random surfing* [81], users traverse a link to some neighbor with probability $(1-c)$, and jump to a random page with probability c. The constant c is known as the *teleportation probability*, typically set to 0.15 [60].
- While browsing the Web, users perform random surfing with probability x. With probability $(1-x)$ users query a search engine and browse among results presented in the form of a ranked list.

We still assume that there is only one search engine that every user uses for querying. However, this assumption does not significantly restrict the applicability of our model. For our purposes the effect of multiple search engines that present the same ranked list for a query is equivalent to a single search engine that presents the same ranked list and gets a user traffic equal to the sum of the user traffic of the multiple search engines.

Assuming that page popularity is measured using PageRank, under our mixed browsing model the expected visit rate of a page p at time t is given by:

$$\begin{aligned} V(p,t) &= (1-x)\cdot F(P(p,t)) \\ &+ x\cdot\left((1-c)\cdot\frac{P(p,t)}{\sum_{p'\in\mathcal{P}}P(p',t)}+c\cdot\frac{1}{n})\right) \end{aligned}$$

Figure 5.11 shows absolute QPC values for different values of x (based on simulation). Unlike with other graphs in this chapter, in this graph we plot the absolute value of QPC, because the ideal QPC value varies with the extent of random surfing (x). Recall that $x = 0$ denotes pure search engine based surfing, while $x = 1$ denotes pure random surfing. Observe that for all values of x, randomized rank promotion performs better than (or as well as) nonrandomized ranking. It is interesting to observe that when x is small, random surfing helps nonrandomized ranking, since random surfing increases the chances of exploring unpopular pages (due to the teleportation probability). However, beyond a certain extent, it does not help as much as it hurts (due to the

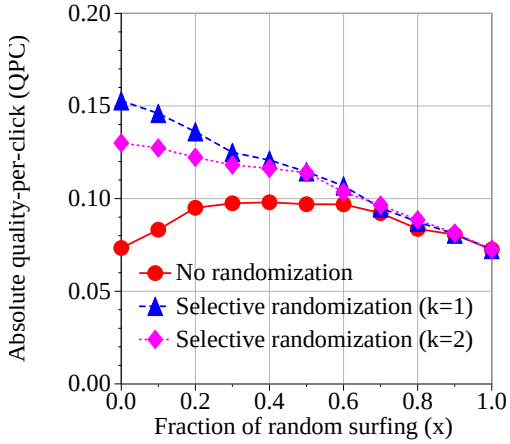

Figure 5.11: *Influence of the extent of random surfing.*

exploration/exploitation tradeoff as was the case for randomized rank promotion).

5.10 Real-World Effectiveness of Rank Promotion

In this section we describe a real-world study we conducted to test the effectiveness of rank promotion.

5.10.1 Experimental Procedure

For this experiment we created our own small Web community consisting of several thousand Web pages containing entertainment-oriented content, and nearly one thousand volunteer users who had no prior knowledge of this project.

Pages: We focused on entertainment because we felt it would be relatively easy to attract a large number of users. The material we started with consisted of a large number of jokes gathered from online databases. We decided to use "funniness" as a surrogate for quality, since users are generally willing to provide their opinion about how funny something is. We wanted the funniness distribution of our jokes to mimic the quality distribution of pages on the Web. As far as we know PageRank is the best available estimate of the quality distribution of Web pages, so we downsampled our

initial collection of jokes and quotations to match the PageRank distribution reported in [28]. To determine the funniness of our jokes for this purpose we used numerical user ratings provided by the source databases. Since most Web pages have very low PageRank, we needed a large number of nonfunny items to match the distribution, so we chose to supplement jokes with quotations. We obtained our quotations from sites offering insightful quotations not intended to be humorous. Each joke and quotation was converted into a single Web page on our site.

Overall site: The main page of the Web site we set up consisted of an ordered list of links to individual joke/quotation pages, in groups of ten at a time, as is typical in search engine responses. Text at the top stated that the jokes and quotations were presented in descending order of funniness, as rated by users of the site. Users had the option to rate the items: we equipped each joke/quotation page with three buttons, labeled "funny," "neutral," and "not funny." To minimize the possibility of voter fraud, once a user had rated an item the buttons were removed from that item, and remained absent upon all subsequent visits by the same user to the same page.

Users: We advertised our site daily over a period of 45 days, and encouraged visitors to rate whichever jokes and quotations they decided to view. Overall we had 962 participants. Each person who visited the site for the first time was assigned at random into one of two user groups (we used cookies to ensure consistent group membership across multiple visits, assuming few people would visit our site from multiple computers): one group for which rank promotion was used, and one for which rank promotion was not used. For the latter group, items were presented in descending order of current popularity, measured as the number of funny votes submitted by members of the group.[7] For the other group of users, items were also presented in descending order of popularity among members of the group, except that all items that had not yet been viewed by any user were inserted in a random order starting at rank position 21 (This variant corresponds to selective promotion with $k = 21$ and $r = 1$.).[8] A new random order for these zero-awareness items was chosen for each unique user. Users were not informed that rank promotion was being employed.

Content rotation: For each user group we kept the number of accessible joke/quotation items fixed at 1000 throughout the duration of our 45-day experiment. However, each item had a finite lifetime of less than 45 days. Lifetimes for the initial 1000 items were assigned uniformly at random

[7] Due to the relatively small scale of our experiment there were frequent ties in popularity values. We chose to break ties based on age, with older pages receiving better rank positions, to simulate a less discretized situation.

[8] We chose this setting since it seemed to work the best in our simulations for the characteristics of the joke scenario that we studied (*e.g.*, number of users, number of jokes, lifetime of jokes).

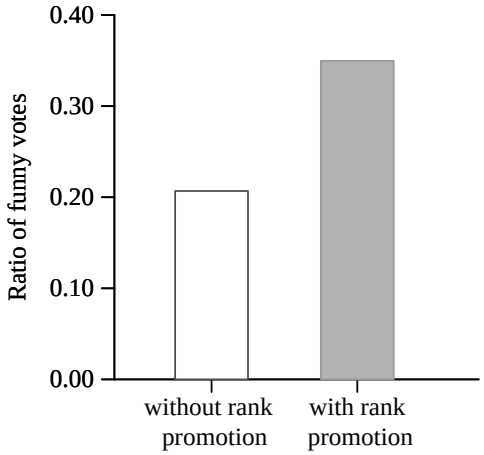

Figure 5.12: *Improvement in overall quality due to rank promotion in live study.*

from $[1, 30]$, to simulation a steady-state situation in which each item had a real lifetime of 30 days. When a particular item expired we replaced it with another item of the same quality, and set its lifetime to 30 days and its initial popularity to zero. At all times we used the same joke/quotation items for both user groups.

5.10.2 Results

First, to verify that the subjects of our experiment behaved similarly to users of a search engine, we measured the relationship between the rank of an item and the number of user visits it received. We discovered a power-law with an exponent remarkably close to $-3/2$, which is precisely the relationship between rank and number of visits that has been measured from usage logs of the AltaVista search engine (see Section 5.6.3 for details).

We then proceeded to assess the impact of rank promotion. For this purpose we wanted to analyze a steady-state scenario, so we only measured the outcome of the final 15 days of our experiment (by then all the original items had expired and been replaced). For each user group we measured the ratio of funny votes to total votes during this period. Figure 5.12 shows the result. The ratio achieved using rank promotion was approximately 60% larger than that obtained using strict ranking by popularity.

5.11 Chapter Summary

In this chapter we studied the problem of ranking result pages. The objective is to rank pages in decreasing order of quality while the page quality values are not known beforehand. The standard method of ranking search results deterministically according to popularity has a significant flaw: high-quality Web pages that happen to be new are drastically undervalued. To address this problem we proposed our randomized ranking scheme of promoting unexplored pages at random rank positions in the result list. We showed through extensive simulation of a wide variety of Web community types that our randomized ranking scheme (using just 10% randomization) consistently leads to much higher-quality search results compared with strict deterministic ranking. We also presented results of a real-world study which demonstrated the effectiveness of rank promotion. We developed new analytical models of Web page popularity evolution under deterministic and randomized search result ranking, and introduced formal metrics by which to evaluate ranking methods.

Chapter 6

Advertisement Ranking

Search engines display relevant advertisements along with relevant pages, as shown in Figure 1.2. It serves two purposes: (a) for search engines it acts as a major source of revenue, and (b) for users it brings relevant advertisements for commercial products/services, especially useful for queries with "commercial intent." In the previous chapter we studied how to rank pages. In this chapter we focus on the presentation of advertisements.

6.1 Problem Formulation

Successful advertisement placement relies on knowing the appeal or "clickability" of advertisements. The main difficulty is that the appeal of new advertisements that have not yet been "vetted" by users can be difficult to estimate. In this chapter we study the problem of placing advertisements to maximize a search engine's revenue, in the presence of uncertainty about appeal.

Following [74], we formulate the problem of selecting advertisements to present as illustrated in Figure 6.1. There are m advertisers $A_1, A_2 \ldots A_m$ who wish to advertise on a search engine. The search engine runs a large auction where each advertiser submits its bids to the search engine for the query phrases in which it is interested. Advertiser A_i submits advertisement $a_{i,j}$ to *target* query phrase Q_j, and promises to pay $b_{i,j}$ amount of money for each click on this advertisement, where $b_{i,j}$ is A_i's bid for advertisement $a_{i,j}$. Each ad $a_{i,j}$ has an associated *click-through rate* (CTR) $c_{i,j}$ which denotes the probability of a user to click on advertisement $a_{i,j}$ given that the advertisement

was displayed to the user for query phrase Q_j. The CTRs of ads are not known to the search engine beforehand.[1]

Advertiser A_i can also specify a daily budget (d_i) that is the total amount of money it is willing to pay for the clicks on its advertisements in a day. Given a user search query on phrase Q_j, the search engine selects a constant number $C \geq 1$ of advertisements from the candidate set of advertisements $\{a_{*,j}\}$, targeted to Q_j. The objective in selecting advertisements is to maximize the search engine's total revenue. The arrival sequence of user queries is not known in advance. Hence, we have the following optimization problem:

- *Objective:* Maximize the search engine's advertising revenue (*i.e.*, sum of $c_{i,j} \cdot b_{i,j}$ for displayed ads while respecting advertisers' budget constraints).
- *Constraint:* Select C or fewer ads to display for each query.
- *Uncertainty:* Click-through rates of ads (*i.e.*, $c_{i,j}$ values) are unknown.

For now we assume that each day a new set of advertisements is given to the search engine and the set remains fixed through out the day; we drop both of these assumptions later.

We show the space of problem variants (along with the best known advertisement policies) in Figure 6.2. The offline problem, *i.e.*, when CTRs are known, and the problem when CTR is equal to 1 for all ads have been studied before. For the sake of completeness we summarize the results here. GREEDY refers to selection of advertisements according to expected revenue (*i.e.*, $c_{i,j} \cdot b_{i,j}$). In Cells I and III GREEDY performs as well as the optimal policy, where the optimal policy also knows the arrival sequence of queries in advance. We write "ratio=1" in Figure 6.2 to indicate that GREEDY has the competitive ratio of 1. For Cells II and IV the greedy policy is not optimal, but is nevertheless 1/2 competitive. An alternative policy for Cell II was given in [74], which we refer to as MSVV; it achieves a competitive ratio of $1 - 1/e$.

6.2 Overview of Our Approach

In our work we give the first policies for the online problem (*i.e.*, Cells V and VI) where we must choose which advertisements to display while simultaneously estimating click-through rates of

[1] We assume the CTR of an ad to be independent of its placement in the displayed page.

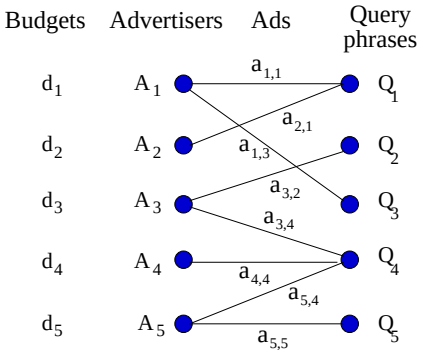

Figure 6.1: *Advertiser and query model.*

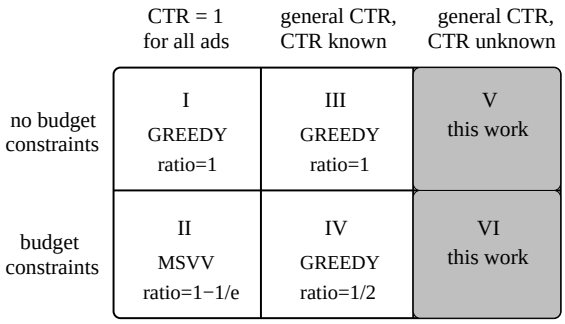

Figure 6.2: *Problem variants.*

advertisements. The main issue we face while addressing Cells V and VI is to balance the exploration/exploitation tradeoff. To maximize short-term revenue, the search engine should *exploit* its current, imperfect CTR estimates by displaying advertisements whose estimated CTRs are large. On the other hand, to maximize long-term revenue, the search engine needs to *explore*, *i.e.*, identify which advertisements have the largest CTRs. This kind of exploration entails displaying advertisements whose current CTR estimates are of low confidence, which inevitably leads to displaying some low-CTR ads in the short-term. This kind of tradeoff between *exploration* and *exploitation* shows up often in practice, e.g., in clinical trials, and has been extensively studied in the context of the *multi-armed bandit* problem [12]. We draw upon and extend the existing bandit literature to solve the advertisement problem in the case of unknown CTR.

6.3 Chapter Outline

In Section 6.5 we show that the unbudgeted variant of the problem (Cell V in Figure 6.2) is an instance of the multi-armed bandit problem. Then, in Section 6.6 we introduce a new kind of bandit problem that we termed the *budgeted multi-armed multi-bandit* problem (BMMP), and show that the budgeted unknown-CTR advertisement problem (Cell VI) is an instance of BMMP. We propose policies for BMMP and give performance bounds. We evaluate our policies empirically over real-world data in Section 6.7. Also, in Section 6.8 we show how to extend our policies to address various practical considerations, *e.g.*, exploiting any similarity or prior information available about the CTRs of ads, and permitting advertisers to submit and revoke advertisements at any time, not

just at day boundaries.

6.4 Related Work

We have already discussed the work of [74], which addresses the advertisement problem under the assumption that CTRs are known. There has not been much published work on estimating CTRs. Reference [72] discusses how contextual information such as user demographic or ad topic can be used to estimate CTRs, and makes connections to the recommender and bandit problems, but stops short of presenting technical solutions. Some methods for estimating CTRs are proposed in [56] with the focus of thwarting click fraud.

Reference [2] studies how to maximize user clicks on banner ads. The key problem addressed in [2] is to satisfy the contracts made with the advertisers in terms of the minimum guaranteed number of impressions (as opposed to the budget constraints in our problem). Reference [94] looks at the advertisement problem from an advertiser's point of view, and gives an algorithm for identifying the most profitable set of keywords for the advertiser.

6.5 Unbudgeted Unknown-CTR Advertisement Problem

In this section we address Cell V of Figure 6.2, where click-through rates are initially unknown and budget constraints are absent (*i.e.*, $d_i = \infty$ for all advertisers A_i). Our unbudgeted problem is an instance of the multi-armed bandit problem [12], which is the following: we have K *arms* where each arm has an associated reward and payoff probability. The payoff probability is not known to us while the reward may or may not be known (both versions of the bandit problem exist). With each *invocation* we *activate* exactly $C \leq K$ arms.[2] Each activated arm then yields the associated reward with its payoff probability and nothing with the remaining probability. The objective is to determine a *policy* for activating the arms so as to maximize the total reward over some number of invocations.

To solve the unbudgeted unknown-CTR advertisement problem, we create a multi-armed bandit problem instance for each query phrase Q, where ads targeted for the query phrase are the arms,

[2] The conventional multi-armed bandit problem is defined for $C = 1$. We generalize it to any $C \geq 1$ in this work.

bid values are the rewards and CTRs are the payoff probabilities of the bandit instance. Since there are no budget constraints, we can treat each query phrase independently and solve each bandit instance in isolation. [3] The number of invocations for a bandit instance is not known in advance because the number of queries of phrase Q in a given day is not known in advance.

A variety of policies have been proposed for the bandit problem, *e.g.*, [3, 6, 68], any of which can be applied to our unbudgeted advertisement problem. The policies proposed in [6] are particularly attractive because they have a known performance bound for any number of invocations not known in advance (in our context the number of queries is not known a priori). In the case of $C = 1$, the policies of [6] make $O(ln\ n)$ number of *mistakes*, on expectation, in n invocations (which is also the asymptotic lower bound on the number of mistakes [68]).[4] A mistake occurs when a suboptimal arm is chosen by a policy (the optimal arm is the one with the highest expected reward).

We consider a specific policy from [6] called UCB and apply it to our problem (other policies from [6] can also be used). UCB is proposed under a slightly different reward model; we adapt it to our context to produce the following policy that we call *MIX* (for mixing exploration with exploitation). We prove a performance bound of $O(ln\ n)$ mistakes for MIX for any $C \geq 1$ in Appendix 9.4.

Policy MIX :

Each time a query for phrase Q_j arrives:

1. *Display the C ads targeted for Q_j that have the highest priority. The priority $P_{i,j}$ of ad $a_{i,j}$ is a function of its current CTR estimate ($\hat{c}_{i,j}$), its bid value ($b_{i,j}$), the number of times it has been displayed so far ($n_{i,j}$), and the number of times phrase Q_j has been queried so far in the day (n_j). Formally, priority $P_{i,j}$ is defined as:*

$$P_{i,j} = \begin{cases} \left(\hat{c}_{i,j} + \sqrt{\frac{2\ ln\ n_j}{n_{i,j}}}\right) \cdot b_{i,j} & \text{if } n_{i,j} > 0 \\ \infty & \text{otherwise} \end{cases}$$

2. *Monitor the clicks made by users and update the CTR estimates $\hat{c}_{i,j}$ accordingly. $\hat{c}_{i,j}$ is the average click-through rate observed so far,* i.e., *the number of times ad $a_{i,j}$ has been clicked on divided by the total number of times it has been displayed.*

Policy MIX manages the exploration/exploitation tradeoff in the following way. The priority function has two factors: an exploration factor $\left(\sqrt{\frac{2\ ln\ n_j}{n_{i,j}}}\right)$ that diminishes with time, and an ex-

[3] We assume CTRs to be independent of one another. Also, we ignore the latency involved in obtaining user clicks.
[4] The constant in $O(ln\ n)$ depends on the payoff probabilities and rewards values of the bandit arms.

ploitation factor ($\hat{c}_{i,j}$). Since $\hat{c}_{i,j}$ can be estimated only when $n_{i,j} \geq 1$, the priority value is set to ∞ for an ad which has never been displayed before.

Importantly, the MIX policy is practical to implement because it can be evaluated efficiently using a single pass over the ads targeted for a query phrase. Furthermore, it incurs minimal storage overhead because it keeps only three numbers ($\hat{c}_{i,j}$, $n_{i,j}$ and $b_{i,j}$) with each ad and one number (n_j) with each query phrase.

6.6 Budgeted Unknown-CTR Advertisement Problem

We now turn to the more challenging case in which advertisers can specify daily budgets (Cell VI of Figure 6.2). Recall from Section 6.5 that in the absence of budget constraints, we were able to treat the bandit instance created for a query phrase independent of the other bandit instances. However, budget constraints create dependencies between query phrases targeted by an advertiser. To model this situation, we introduce a new kind of bandit problem that we call **B**udgeted **M**ulti-armed **M**ulti-bandit **P**roblem (BMMP), in which multiple bandit instances are run in parallel under overarching budget constraints. We derive generic policies for BMMP and give performance bounds.

6.6.1 Budgeted Multi-armed Multi-bandit Problem

BMMP consists of a finite set of multi-armed bandit instances, $\mathcal{B} = \{B_1, B_2 \ldots B_{|\mathcal{B}|}\}$. Each bandit instance B_i has a finite number of arms and associated rewards and payoff probabilities as described in Section 6.5. In BMMP each arm also has an associated *type*. Each type $T_i \in \mathcal{T}$ has budget $d_i \in [0, \infty]$ which specifies the maximum amount of reward that can be generated by activating all the arms of that type. Once the specified budget is reached for a type, the corresponding arms can still be activated but no further reward is earned.

With each invocation of the bandit system, one bandit instance from $\mathcal{B}$ is invoked; the policy has no control over which bandit instance is invoked. Then the policy activates C arms of the invoked bandit instance, and the activated arms generate some (possibly zero) total reward.

It is easy to see that the budgeted unknown-CTR advertisement problem is an instance of BMMP. Each query phrase acts as a bandit instance and the ads targeted for it act as bandit arms, as described in Section 6.5. Each advertiser defines a unique type of arms and gives a budget

constraint for that type; all ads submitted by an advertiser belong to the type defined by it. When a query is submitted by a user, the corresponding bandit instance is invoked.

We now show how to derive a policy for BMMP given as input a policy POL for the regular multi-armed bandit problem such as one of the policies from [6]. The derived policy, denoted by *BPOL* (**B**udget-aware POL), is as follows:

- Run $|\mathcal{B}|$ instances of POL in parallel, denoted $\text{POL}_1, \text{POL}_2, \ldots \text{POL}_{|\mathcal{B}|}$.
- Whenever bandit instance B_i is invoked:
  1. Discard any arm(s) of B_i whose type's budget is newly depleted, *i.e.*, has become depleted since the last invocation of B_i.
  2. If one or more arms of B_i was discarded during step 1, restart POL_i.
  3. Let POL_i decide which of the remaining arms of B_i to activate.

Observe that in the second step of BPOL, when POL is restarted, POL loses any state it has built up, including any knowledge gained about the payoff probabilities of bandit arms. Surprisingly, despite this seemingly imprudent behavior, we can still derive a good performance bound for BPOL, provided that POL has certain properties, as we discuss in the next section. In practice, since most bandit policies can take prior information about the payoff probabilities as input, when restarting POL we can supply the previous payoff probability estimates as the prior (as done in our experiments).

6.6.2 Performance Bound for BMMP Policies

Let S denote the sequence of bandit instances that are invoked, *i.e.*, $S = \{S(1), S(2) \ldots S(N)\}$ where $S(n)$ denotes the index of the bandit instance invoked at the n^{th} invocation. We compare the performance of BPOL with that of the optimal policy, denoted by OPT, where OPT has advance knowledge of S and the exact payoff probabilities of all bandit instances.

We claim that $bpol(N) \geq opt(N)/2 - O(f(N))$ for any N, where $bpol(N)$ and $opt(N)$ denote the total expected reward obtained after N invocations by BPOL and OPT, respectively, and $f(n)$ denotes the expected number of mistakes made by POL after n invocations of the the regular multi-armed bandit problem (for UCB, $f(n)$ is $O(ln\ n)$ [6]). Our complete proof is rather involved. Here we give a high-level outline of the proof (the complete proof is given in Appendix 9.3). For simplicity we focus on the $C = 1$ case; $C \geq 1$ is a simple extension thereof.

Since bandit arms generate rewards stochastically, it is not clear how we should compare BPOL and OPT. For example, even if BPOL and OPT behave in exactly the same way (activate the same arm on each bandit invocation), we cannot guarantee that both will have the same total reward in the end. To enable meaningful comparison, we define a *payoff instance*, denoted by I, such that $I(i,n)$ denotes the reward generated by arm i of bandit instance $S(n)$ for invocation n in payoff instance I. The outcome of running BPOL or OPT on a given payoff instance is deterministic because the rewards are fixed in the payoff instance. Hence, we can compare BPOL and OPT on per payoff instance basis. Since each payoff instance arises with a certain probability, denoted as $\mathbb{P}(I)$, by taking expectation over all possible payoff instances of execution we can compare the expected performance of BPOL and OPT.

Let us consider invocation n in payoff instance I. Let $B(I,n)$ and $O(I,n)$ denote the arms of bandit instance $S(n)$ activated under BPOL and OPT respectively. Based on the different possibilities that can arise, we classify invocation n into one of three categories:

- *Category 1:* The arm activated by OPT, $O(I,n)$, is of smaller or equal expected reward in comparison to the arm activated by BPOL, $B(I,n)$. The expected reward of an arm is the product of its payoff probability and reward.
- *Category 2:* Arm $O(I,n)$ is of greater expected reward than $B(I,n)$, but $O(I,n)$ is not available for BPOL to activate at invocation n due to budget restrictions.
- *Category 3:* Arm $O(I,n)$ is of greater expected reward than $B(I,n)$ and both arms $O(I,n)$ and $B(I,n)$ are available for BPOL to activate, but BPOL prefers to activate $B(I,n)$ over $O(I,n)$.

Let us denote the invocations of category k (1, 2 or 3) by $\mathcal{N}^k(I)$ for payoff instance I. Let $bpol_k(N)$ and $opt_k(N)$ denote the expected reward obtained during the invocations of category k (1, 2 or 3) by BPOL and OPT respectively. In Appendix 9.3 we show that

$$bpol_k(N) = \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^k(I)} I(B(I,n),n) \Big)$$

Similarly,

$$opt_k(N) = \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^k(I)} I(O(I,n),n) \Big)$$

Then for each k we bound $opt_k(N)$ in terms of $bpol(N)$. In Appendix 9.3 we provide proof of each of the following bounds:

Lemma 3 *$opt_1(N) \leq bpol_1(N)$.*

Lemma 4 *$opt_2(N) \leq bpol(N) + (|\mathcal{T}| \cdot r_{max})$, where $|\mathcal{T}|$ denotes the number of arm types and r_{max} denotes the maximum reward.*

Lemma 5 *$opt_3(N) = O(f(N))$.*

From the above bounds we obtain our overall claim:

Theorem 3 *$bpol(N) \geq opt(N)/2 - O(f(N))$, where $bpol(N)$ and $opt(N)$ denote the total expected reward obtained under BPOL and OPT respectively.*

Proof:

$$\begin{aligned} & opt(N) \\ &= opt_1(N) + opt_2(N) + opt_3(N) \\ &\leq bpol_1(N) + bpol(N) + \big(|\mathcal{T}| \cdot r_{max}\big) + O(f(N)) \\ &\leq 2 \cdot bpol(N) + O(f(N)) \end{aligned}$$

Hence, $bpol(N) \geq opt(N)/2 - O(f(N))$. ■

Next we use our generic BPOL framework to derive a policy for the budgeted unknown-CTR advertisement problem.

6.6.3 Policy BMIX and its Variants

We supply MIX (Section 6.5) as input to our BPOL framework, and obtain *BMIX* as the output. Policy BMIX operates as follows:

- *Each time a query for phrase Q_j arrives:*
  1. *For ads whose advertisers have not depleted their budgets yet, compute the priorities as defined in Policy MIX, and display the C ads of highest priority.*

2. *Update the CTR estimates ($\hat{c}_{i,j}$) of the displayed ads by monitoring user clicks.*

Note that it is not necessary to restart the MIX instance for Q_j when an advertiser's budget is depleted as done in the generic BPOL (Section 6.6.1). The reason is that MIX maintains state (*i.e.*, $n_{i,j}$, $\hat{c}_{i,j}$'s) on a per-ad basis, so it can continue from where it left off if some ads are removed from consideration "in-flight".

In Appendix 9.4 we show that for MIX, $f(n)$ is $O(ln\ n)$ for any $C \geq 1$. Hence, using our general result of Section 6.6.2, we know that the average revenue generated by BMIX is at least $opt(N)/2 - O(ln\ N)$ for any $C \geq 1$ where $opt(N)$ denotes the optimal revenue generated from answering N user queries.

So far, for modeling purposes, we have assumed the search engine receives an entirely new batch of advertisements each day. In reality, ads may persist over multiple days. With BMIX, we can carry forward an ad's CTR estimate ($\hat{c}_{i,j}$) and display count ($n_{i,j}$) from day to day until an ad is revoked, to avoid having to re-learn CTR's from scratch each day. Of course the daily budgets reset daily, regardless of how long each ad persists. In fact, with a little care we can permit ads to be submitted and revoked at arbitrary times (not just at day boundaries). We describe this extension, as well as how we can incorporate and leverage prior beliefs about CTR's, in Section 6.8.

Next we propose some variants of BMIX. We do not derive a theoretical performance bound for these variants, however we expect them to perform well in practice as demonstrated in Section 4.7

1. Varying the Exploration Factor. Internally, BMIX runs instances of MIX to select which ads to display. As mentioned in Section 6.5, the priority function of MIX consists of an exploration factor ($\sqrt{\frac{2\ ln\ n_j}{n_{i,j}}}$) and an exploitation factor ($c_{i,j}$). In [6] it was shown empirically that the following heuristical exploitation factor performs well, despite the absence of a known performance guarantee:

$$\sqrt{\frac{ln\ n_j}{n_{i,j}} \cdot \min\left\{\frac{1}{4}, V_{i,j}(n_{i,j}, n_j)\right\}} \quad \text{where} \quad V_{i,j}(n_{i,j}, n_j) = \left(\hat{c}_{i,j} \cdot (1 - \hat{c}_{i,j})\right) + \sqrt{\frac{2\ ln\ n_j}{n_{i,j}}}$$

Substituting this expression in place of $\sqrt{\frac{2\ ln\ n_j}{n_{i,j}}}$ in the priority function of BMIX gives us a new (heuristical) policy we call *BMIX-E*.

2. Budget Throttling. It is shown in [74] that in the presence of budget constraints, it is beneficial to display the ads of an advertiser less often as the advertiser's remaining budget decreases.

In particular, they propose to multiply bids from advertiser A_i by the following *discount factor*:

$$\phi(d_i') = 1 - e^{-d_i'/d_i}$$

where d_i' is the current remaining budget of advertiser A_i for the day and d_i is its total daily budget. Following this idea we can replace $b_{i,j}$ by $\left(\phi(d_i') \cdot b_{i,j}\right)$ in the priority function of BMIX, yielding a variant we call *BMIX-T*. Policy *BMIX-ET* refers to use of heuristics 1 and 2 together.

6.7 Experiments

From our general result of Section 6.6, we have a theoretical performance guarantee for BMIX. In this section we study BMIX and its variants empirically. In particular, we compare them with the greedy policy proposed for the known-CTR advertisement problem (Cells 1-IV in Figure 6.2). GREEDY displays the C ads targeted for a query phrase that have the highest $\left(\hat{c}_{i,j} \cdot b_{i,j}\right)$ values among the ads whose advertisers have enough remaining budgets; to induce a minimal amount of exploration, for an ad which has never been displayed before, GREEDY treats $\hat{c}_{i,j}$ as ∞ (our policies do this as well). GREEDY is geared exclusively toward *exploitation*. Hence, by comparing GREEDY with our policies, we can gauge the importance of *exploration*.

6.7.1 Experiment Setup

We evaluate advertisement policies by conducting simulations over real-world data. Our data set consists of a sample of 85,000 query phrases selected at random from the Yahoo! query log for the date of February 12, 2006. Since we have the frequency counts of these query phrases but not the actual order, we ran the simulations multiple times with random orderings of the query instances and report the average revenue in all our experiment results.[5] The total number of query instances is 2 million. For each query phrase we have the list of advertisers interested in it and the ads submitted by them to Yahoo!. We also have the budget constraints of the advertisers. Roughly 60% of the advertisers in our data set impose daily budget constraints.

In our simulation, when an ad is displayed, we decide whether a click occurs by flipping a coin

[5] Since we used random orderings of the query instances instead of the actual query sequence, we may have lost some interesting structure there.

weighted by the true CTR of the ad. Since true CTRs are not known to us (this is the problem we are trying to solve!), we took the following approach to assign CTRs to ads: from a larger set of Yahoo! ads we selected those ads that have been displayed more than thousand times, and therefore we have highly accurate CTR estimates. We regarded the distribution of these CTR estimates as the true CTR distribution. Then for each ad $a_{i,j}$ in the dataset we sampled a random value from this distribution and assigned it as CTR $c_{i,j}$ of the ad. (Although this method may introduce some skew compared with the (unknown) true distribution, it is the best we could do short of serving live ads just for the purpose of measuring CTRs [6]).

We are now ready to present our results. To start with we consider a simple setting where the set of ads is fixed and no prior information about CTR is available. We study the more general setting in Section 6.8.

6.7.2 Exploration/Exploitation Tradeoff

We ran each of the policies for a time horizon of ten days; each policy carries over its CTR estimates from one day to the next. Budget constraints are renewed each day. For now we fix the number of displayed ads (C) to 1. Figure 6.3 plots the revenue generated by each policy after a given number of days (for confidentiality reasons we have changed the unit of revenue). All policies (including GREEDY) estimate CTRs based on past observations, so as time passes by their estimates become more reliable and their performance improves. Note that the exploration factor of BMIX-E causes it to perform substantially better than that of BMIX. The budget throttling heuristic (BMIX-T and BMIX-ET) did not make much difference in our experiments.

All of our proposed policies perform significantly better than GREEDY, which underscores the importance of balancing exploration and exploitation. GREEDY is geared exclusively toward exploitation, so one might expect that early on it would outperform the other policies. However, that does not happen because GREEDY immediately fixates on ads that are not very profitable (*i.e.*, low $c_{i,j} \cdot b_{i,j}$).

Next we vary the number of ads displayed for each query (C). Figure 6.4 plots total revenue over ten days on the y-axis, and C on the x-axis. Each policy earns more revenue when more ads are displayed (larger C). Our policies outperform GREEDY consistently across different values of C. In

[6] We also attempted to compute the CTRs of ads in our dataset based on their past click and display information. However, for a non-negligible number of ads we did not have enough number of displays to compute their CTR estimates reliably.

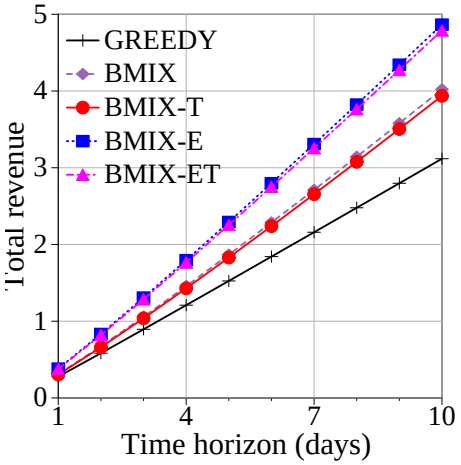

Figure 6.3: *Revenue generated by different advertisement policies (C=1).*

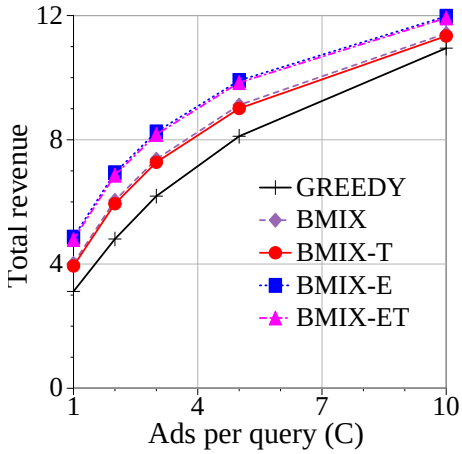

Figure 6.4: *Effect of C (number of ads displayed per query).*

fact, GREEDY must display almost twice as many ads as BMIX-E to generate the same amount of revenue.

6.8 Practical Extensions of BMIX

In Section 6.6 we studied BMIX in a simple setting where the set of ads is fixed and no prior information about CTR is available. We consider the more general setting now.

6.8.1 Exploiting Prior Information About CTRs

In practice, search engines may have some prior information available about the CTRs of ads even before displaying them and gauging user response. The prior information may come from various sources such as textual relevance of the ad to the query phrase or trustworthiness of the advertiser who submitted the ad. We do not propose any method of deriving the prior information in this paper; instead we focus on studying how the prior information, if it is available, can be used in the advertisement policies and what difference it makes on their performance. For instance, it would be interesting to find out whether our policies perform any better than GREEDY if the prior estimates of CTRs are reasonably correct.

Modeling Prior Information: We use the following model of prior information. Suppose the true CTR of ad $a_{i,j}$ is $c_{i,j}$. We assume that the search engine does not know the CTR value a priori, but has a prior distribution on the CTR. We set the form of prior distribution to a beta distribution [7] $beta_{i,j}(\alpha_{i,j}, \beta_{i,j})$ where $\alpha_{i,j}$ and $\beta_{i,j}$ are its parameters. We denote the mean and the variance of $beta_{i,j}$ by $\hat{\mu}_{i,j}$ and $\hat{\sigma}_{i,j}$.

In our experiments we synthetically generate the prior distributions of ads. While generating these distributions, we vary two parameters: (a) the fraction of ads for which the prior distribution is available, denoted by p, and (b) the accuracy of prior information, denoted by v. To synthesize a prior distribution, we take the following two steps: (a) given the true CTR value $c_{i,j}$ we create a beta distribution with mean $c_{i,j}$ and variance $c_{i,j} \cdot (1 - v)$ and (b) we then sample the mean of prior distribution, $\hat{\mu}_{i,j}$, from the created beta distribution and set the variance, $\hat{\sigma}_{i,j}$, to $\hat{\mu}_{i,j} \cdot (1 - v)$.

To give an intuition of how far the initial CTR estimate $\hat{\mu}_{i,j}$ can be from the actual CTR $c_{i,j}$ for different values of v, we consider an ad of CTR equal to 0.2. When $v = 0.9$, $\hat{\mu}_{i,j}$ is set between 0.1 and 0.3 with 0.58 probability. When $v = 0.95$, this probability increases to 0.68 and when $v = 0.98$, it is almost 0.90.

Exploiting Prior Information: Next we show how we use the prior distributions of CTRs in our advertisement policies. All our policies including GREEDY use CTR estimates ($\hat{c}_{i,j}$'s) in deciding which ads to display. We use the prior distributions to find these CTR estimates. Initially, for each ad $a_{i,j}$ the estimate of its CTR is the mean of its prior distribution $beta_{i,j}(\alpha_{i,j}, \beta_{i,j})$, hence, $\hat{c}_{i,j} = \hat{\mu}_{i,j} = \frac{\alpha_{i,j}}{\alpha_{i,j}+\beta_{i,j}}$.

Once ad $a_{i,j}$ has been displayed for query phrase Q_j, we condition its prior distribution using the click observation of the ad and obtain the posterior distribution of its CTR. In particular, if the prior distribution for ad $a_{i,j}$ is $beta_{i,j}(\alpha_{i,j}, \beta_{i,j})$ and suppose that $s_{i,j}$ denotes the number of times the ad was clicked on when it was displayed for Q_j while $f_{i,j}$ denotes number of times it was not, then the posterior distribution of CTR is simply $beta_{i,j}(\alpha_{i,j} + s_{i,j}, \beta_{i,j} + f_{i,j})$. Given the posterior distribution, the CTR estimate (or the mean) is $\frac{\alpha_{i,j}+s_{i,j}}{\alpha_{i,j}+\beta_{i,j}+s_{i,j}+f_{i,j}}$. We use this CTR estimate in all the advertisement policies (GREEDY, BMIX and its variants).

[7] The event of clicking on an ad is a Bernoulli random variable. It is standard to use a beta distribution for modeling the prior of Bernoulli event [19].

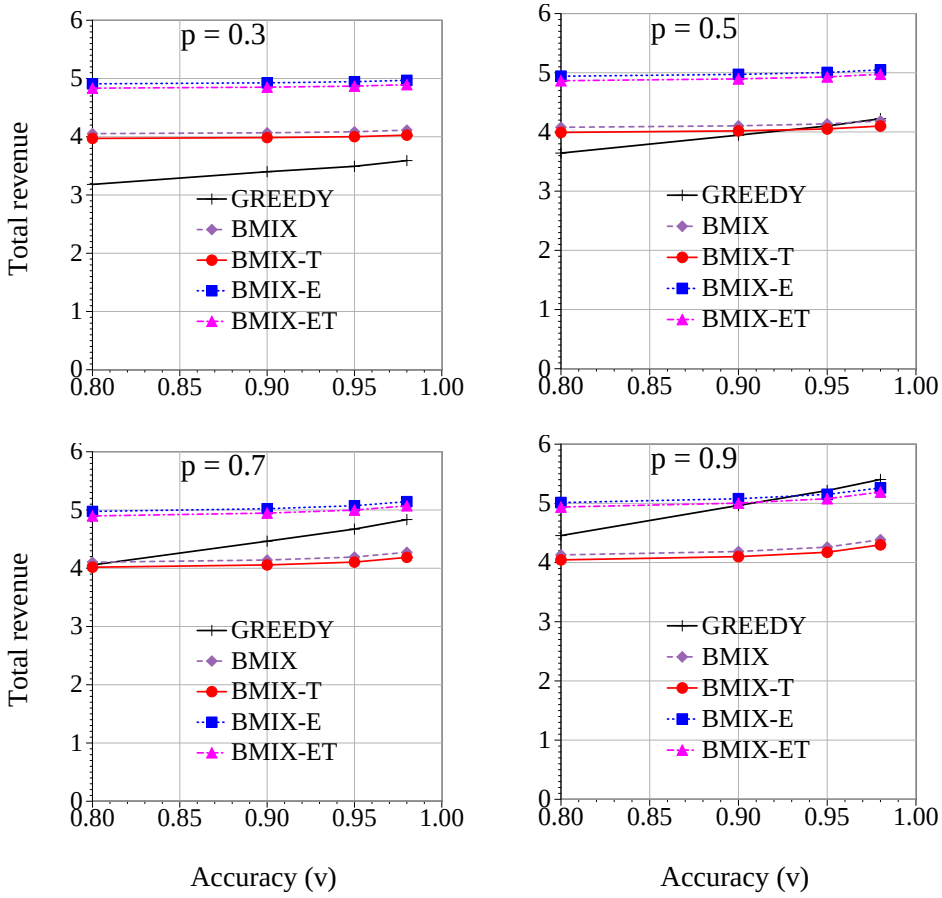

Figure 6.5: *Effect of the prior information.*

6.8.2 Performance Comparison

For a given p and v we simulate the advertisement policies for a time horizon of ten days and measure the total revenue generated. The results are shown in Figure 6.5, with v plotted on the x-axis and the total revenue plotted on the y-axis. The four graphs are for different values of p.

For a given value of p, if we increase v the prior estimates of CTRs ($\hat{\mu}_{i,j}$) get closer to the actual CTRs ($c_{i,j}$), hence, all of the policies perform better. Similarly, if we increase p for a fixed v, the policies get the prior distributions for more ads and they perform better. Note that unlike GREEDY our policies are not affected significantly by the prior distribution of CTRs. GREEDY does not have any provision for exploration, so it relies heavily on the prior distributions. On the other hand, our policies only use the prior distributions to start with (they keep low confidence in the prior

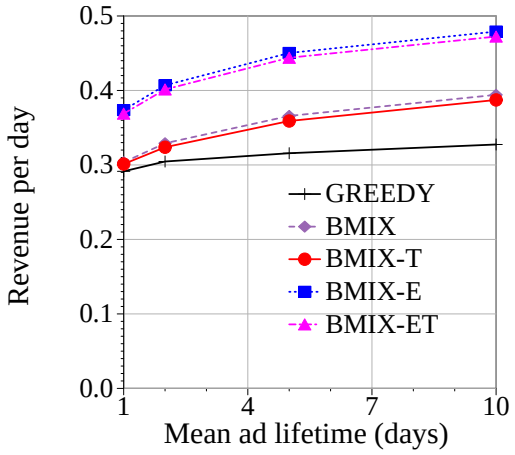

Figure 6.6: *Effect of ad lifetime.*

distributions due to small $\alpha_{i,j} + \beta_{i,j}$) and once in steady state they largely rely on their own CTR estimates.

Except when the amount (p) and accuracy (v) of prior information is exceptionally high, our policies significantly outperform GREEDY. Furthermore, our policies are never substantially worse than GREEDY.

6.8.3 Allowing Submission/Revocation of Ads at Any Time

We now consider the scenario where advertisers can submit or revoke ads at any time. Observe that BMIX (and its variants) seamlessly extends to this scenario. We make BMIX look at all the ads that are available at the time a query phrase is being answered, hence any deleted ad is not considered while every newly submitted one is.

Next we evaluate our policies empirically in this scenario. We use the following model of submission and revocation of ads: an ad stays with the search engine for a lifetime that is distributed according to a Poisson random variable with the mean set to λ. The ad is revoked once its life is over. When the ad is revoked, we submit a new ad with identical characteristics to the just revoked one. Hence, the rates of submission and revocation of ads are kept the same.

Since the ads are in flux in this experiment, we ran our experiment for a long enough time

horizon (100 days) to reach steady state. Figure 6.6 shows the result, with mean lifetime plotted on the x-axis and the revenue generated per day in the steady state on the y-axis. As expected, as the average lifetime of ads (λ) increases, the performance gap between our policies and GREEDY increases. When ads tend to remain in the system for a long time, the exploration done by our policies pays off the most. Even for a reasonably short lifetime of ads, *e.g.*, one day, our policies still outperform GREEDY.

6.8.4 Exploiting Dependencies in CTRs of Ads

So far we assumed the CTRs of ads to be independent of one another. In practice, we often find ads that are similar (*e.g.*, similar textual content, same advertiser) exhibit similar CTR values. In the presence of a large number of ads, it is important to exploit such dependencies in CTRs to expedite the learning process. Fundamentally, we can think of this problem as the bandit problem with dependent arms. Early work in this direction is described in [82, 83], which has two parts: (a) studying and characterizing the dependencies between CTRs, using real-world advertisement data and (b) proposing new bandit algorithms that make use of these dependencies.

In [82] the dependencies between CTRs of ads is modeled by partitioning them into clusters whereby the assumption is that all the ads that belong to the same cluster are likely to have similar CTRs. (These clusters were obtained by classifying ads into a given taxonomy using supervised learning.) Classifying ads into clusters introduces a structure in the bandit problem which can be exploited to reduce the learning cost. A *two-level* bandit policy is proposed for doing this. In the two-level policy, first a bandit policy is run over clusters to pick a cluster, and then inside that cluster another bandit policy is run to select an ad to display. Regret bounds are provided in [83].

Reference [1] studies bandit problems with dependent arms in the context of banner advertisements. Unlike the clustering approach of [82, 83], they model dependencies between ads by assuming that CTRs are a linear function of logical combinations of various ad features. They propose policies for this bandit problem and prove regret bounds. Their policies work by maintaining a hypothesis for the linear function relating features with CTRs and updating this hypothesis over time.

6.9 Chapter Summary

In this chapter we studied the problem of selecting which ads to display in order to maximize revenue, when click-through rates are not initially known. We dealt with the underlying exploration/exploitation tradeoff using multi-armed bandit theory. While we rejected this approach for Web page ranking, in case of advertisement ranking we get immediate feedback in the form of user clicks and thus, we could employ the conventional bandit theory.

In the process we contributed to bandit theory by proposing a new variant of the bandit problem that we call budgeted multi-armed multi-bandit problem (BMMP). We proposed a policy for solving BMMP and derived a performance guarantee. We gave extensions of our policy to address various practical considerations. Extensive experiments over real ad data demonstrate substantial revenue gains compared to a greedy strategy that has no provision for exploration.

Chapter 7

Future Work

We now discuss potential directions for future work.

7.1 Web Page Discovery

In Chapter 3 all our 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 (described in Section 6.5). 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 [6, 44] 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 (Section 3.6.4).

7.2 Web Page Synchronization

In Chapter 4 we proposed our page synchronization scheme under the assumption that the scoring function depends on the page content only. An important research direction is to extend the scheme for scoring functions in which the score of a page depends partially on the content of other pages (as with anchortext inclusion methods [15]). In principle such an extension can be made without compromising the crucial ability to estimate changes in repository quality during index maintenance operations. Another problem left as future work is to determine the most effective method of forecasting the change in quality due to redownloading a page based on historical observations.

Lastly, observe that the synchronization task also faces the exploration/exploitation tradeoff similar to the discovery task. For example, our synchronization policy redownloads pages based on their $\Delta Q(p,t)$ estimates, which are inferred from the past redownloads of the same pages. Hence, the policy can fall into a vicious cycle where some page are not redownloaded because their current $\Delta Q(p,t)$ estimates are poor, and since they are not redownloaded their $\Delta Q(p,t)$ estimates do not change irrespective of their future behavior. This cycle can be avoided by designing synchronization policies which have an explicit provision for exploring $\Delta Q(p,t)$ values, *i.e.*, exploration-based algorithms.

Another interesting research direction is to consider the discovery and synchronization tasks together. We studied the discovery task under the objective of maximizing the number of newly discovery pages, while for the synchronization task our goal was to redownload known pages so as to maximize a user-centric metric of repository quality. Since both tasks require redownloading of known pages and share the same crawling resources, it is natural to optimize them together, possibly under one objective, for better usage of available resources.

7.3 Web Page Ranking

In Chapter 5 we proposed our randomized ranking scheme and analyzed it under a popularity evolution model. Some other models of Web evolution have been previously proposed in [10, 14]. To strengthen the case of our randomizing approach it needs to be evaluated under these evolution models as well (recall that performance evaluation through real-world experiments is impossible for the reasons mentioned in Section 5.1.2).

Another direction for future work is to apply bandit models to this problem. In doing so we face a couple of challenges. One challenge is that for link/visit-based quality metrics, search engines do not receive immediate feedback, as discussed in Chapter 5.2, while the conventional bandit problem assumes an immediate feedback for the action taken. For example, when a user is shown a page and if she likes it, she does not create a link to the page right then. Instead, the link is created after a non-trivial amount of time. The work on bandit models with delayed responses may prove helpful here [35, 111]. In case of visit-based quality metrics (*e.g.*, a page is considered of high quality if a user likes it and visits it frequently afterwards) the feedback is not only delayed but is spread over time. Moreover, such feedback is difficult to be closely monitored by search engines. Hence, the ranking policy also needs to be robust enough to handle missing or erroneous feedback observations. (An example of a robust policy is our randomized policy of Section 5.5 which does not strongly rely on such measurements.)

7.4 Advertisement Ranking

The advertisement ranking problem of Chapter 6 can also be extended in several ways. As mentioned in Section 6.8.4, one such extension is to exploit similarity in ad attributes while inferring CTRs, instead of estimating the CTR of each ad independently. Early work in this direction is described in [82, 83].

Another research direction is to deal with *click fraud* while displaying advertisements. Click fraud happens when fraudulent clicks are made, usually with the intent of increasing the payment of the advertiser [56, 102, 109]. This merits studying an adversarial formulation of the advertisement ranking problem. Also, this leads to general consideration of how to manage exploration versus exploitation in game-theoretic scenarios.

Chapter 8

Summary

This dissertation studied information mediators which collect content and present it to users. After discussing the design choices involved in the architecture of information mediators we focused on the <*content-shipping, pull*>-based mediators, which include search engines. We discussed how search engines work by performing the content acquisition and presentation tasks. We noted that both these tasks pose resource-constrained optimization problems and some parameters relevant to this problems may or may not be known leading to the offline and online scenarios, as shown in Table 8.1. We began by studying content acquisition in Chapters 3-4. Content acquisition has two aspects: discovering new content and synchronizing known content.

In Chapter 3 we focused on discovery, and studied the extent to which new pages can be efficiently discovered by a crawler by redownloading known pages. We formalized this problem as an optimization problem where the unknown parameter is the bipartite graph between the old and new pages. First we showed that the offline problem is NP-hard and can be approximated using a greedy algorithm. Then we studied the online setting in which algorithms must use historical statistics to estimate which pages are most likely to yield links to new content.

In Chapter 4 we studied synchronization, with the focus being on maximizing the quality of the user experience. We proposed our synchronization policy for the offline problem in which page redownloading tasks are scheduled in terms of their benefit on repository quality. Then for the online problem we showed that the benefit of redownloading a particular page is amenable to prediction based on measurements of the benefit of downloading the page in the past. We devised an efficient method for taking these measurements and compared our page synchronization scheme against prior schemes empirically using real Web data.

Task	Objective	Constraint	Uncertainty
Web page discovery	Maximize the number of new pages discovered	Limited crawling resources	Unknown bipartite link graph between repository pages and new pages
Web page synchronization	Maximize the repository quality (Equation 4.1)	Limited crawling resources	Unknown update behavior of repository pages
Web page ranking	Maximize the quality of results perceived by users (Equation 5.2)	Limited user attention paid to search results	Unknown quality values of pages
Advertisement ranking	Maximize the search engine's advertising revenue	Limited user attention paid to ads	Unknown click-through rates of ads

Table 8.1: Content acquisition and presentation tasks formulated as optimization problems with constraints and uncertainty.

The second part of this dissertation focuses on the presentation task which involves ranking pages and ads in response to user search queries. In Chapter 5 we studied the problem of ranking Web pages where the unknown parameters are the page quality values. The offline problem, *i.e.*, when page quality values are known, has a simple solution in this case: rank the pages in decreasing order of quality. For the online problem we argued how the deterministic popularity-based ranking scheme suffers from the entrenchment effect. We proposed a randomized ranking policy to address it, and showed through extensive simulation that our randomized ranking scheme consistently leads to much higher-quality search results compared with strict deterministic ranking.

Finally, in Chapter 6 we studied the problem of ranking ads where the unknown parameters are the click-through rates of ads. The offline problem had been studied before [74]. We studied the online problem and dealt with the underlying exploration/exploitation tradeoff using multi-armed bandit theory. We were able to use bandit theory because search engines get instant feedback during advertisement ranking in the form of user clicks (while the feedback is delayed in case of Web page ranking where link-based quality metric is primarily used). We proposed a bandit-based advertisement policy and gave extensions of our policy to address various practical considerations. Extensive experiments over real ad data demonstrate substantial revenue gains compared to a greedy strategy that has no provision for exploration.

Bibliography

[1] N. Abe, A. Biermann, and P. Long. Reinforcement Learning with Immediate Rewards and Linear Hypotheses. *Algorithmica*, 37(4):263–293, 2003.

[2] N. Abe and A. Nakamura. Learning to Optimally Schedule Internet Banner Advertisements. In *Proceedings of the 16th International Conference on Machine Learning*, 1999.

[3] R. Agrawal. Sample Mean Based Index Policies with O(log n) Regret for the Multi-Armed Bandit Problem. *Advances in Applied Probability*, 27:1054–1078, 1995.

[4] Alexa Web Search. http://www.alexa.com/.

[5] AltaVista Query Log. http://ftp.archive.org/AVLogs/.

[6] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time Analysis of the Multi-Armed Bandit Problem. *Machine Learning*, 47:235–256, 2002.

[7] R. Baeza-Yates, F. Saint-Jean, and C. Castillo. Web Structure, Dynamics and Page Quality. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval*, 2002.

[8] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, NY, 1999.

[9] M. Balabanovic. Exploring Versus Exploiting when Learning User Models for Text Recommendation. *User Modeling and User-Adapted Interaction*, 8(1-2):71–102, 1998.

[10] A.-L. Barabasi and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286:509–512, 1999.

[11] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. MINERVA: Collaborative P2P Search. In *Proceedings of the 31st International Conference on Very Large Databases*, 2005.

[12] D. A. Berry and B. Fristedt. Bandit Problems: Sequential Allocation of Experiments. Chapman and Hall, London, 1985.

[13] B. Brewington and G. Cybenko. How Dynamic is the Web? In *Proceedings of the 9th International World Wide Web Conference*, 2000.

[14] S. Brin, J. Davis, and H. Garcia-Molina. Copy Detection Mechanisms for Digital Documents. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, 1995.

[15] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Proceedings of the 7th International World Wide Web Conference*, 1998.

[16] 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.

[17] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic Clustering of the Web. In *Proceedings of the 6th International World Wide Web Conference*, 1997.

[18] M. Burner. Crawling Towards Eternity: Building An Archive of The World Wide Web. *Web Techniques Magazine*, 2(5):37–40, 1997.

[19] G. Casella and R. L. Berger. Statistical Inference. Thomson Learning, 2001.

[20] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data.* Morgan Kaufmann, 2002.

[21] S. Chakrabarti, M. van den Berg, and B. Dom. Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. In *Proceedings of the 8th International World Wide Web Conference*, 1999.

[22] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In

Proceedings of the l00th Anniversary Meeting of the Information Processing Society of Japan, 1994.

[23] M. Chen, A. Hearst, A. Marti, J. Hong, and J. Lin. Cha-Cha: A System for Organizing Intranet Results. In *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems*, 1999.

[24] J. Cho. Crawling the Web: Discovery and Maintenance of Large-Scale Web Data. *Ph.D. thesis, Stanford Computer Science Department*, 2001.

[25] J. Cho and H. Garcia-Molina. Synchronizing a Database to Improve Freshness. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 2000.

[26] J. Cho and H. Garcia-Molina. The Evolution of the Web and Implications for an Incremental Crawler. In *Proceedings of the 26th International Conference on Very Large Databases*, 2000.

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

[28] J. Cho and S. Roy. Impact of Search Engines on Page Popularity. In *Proceedings of the 13th International World Wide Web Conference*, 2004.

[29] J. Cho, S. Roy, and R. Adams. Page Quality: In Search of an Unbiased Web Ranking. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, 2005.

[30] Consumer Search. http://www.consumersearch.com/.

[31] F. Douglis, A. Feldmann, and B. Krishnamurthy. Rate of Change and Other Metrics: A Live Study of the World Wide Web. In *Proceedings of the 1st USENIX Symposium on Internet Technologies and Systems*, 1997.

[32] J. E. Coffman, Z. Liu, and R. R. Weber. Optimal Robot Scheduling for Web Search Engines. *Journal of Scheduling*, 1(1):15–29, 1998.

[33] J. Edwards, K. S. McCurley, and J. A. Tomlin. An Adaptive Model for Optimizing Performance of an Incremental Web Crawler. In *Proceedings of the 10th International World Wide Web Conference*, 2001.

[34] S. G. Eick. Gittins Procedures for Bandits with Delayed Responses. *Journal of the Royal Statistical Society, Series B (Methodological)*, 50(1):125–132, 1988.

[35] S. G. Eick. The Two-Armed Bandit with Delayed Responses. *The Annals of Statistics*, 16(1):254–264, 1988.

[36] N. Eiron, K. S. McCurley, and J. A. Tomlin. Ranking the Web Frontier. In *Proceedings of the 13th International World Wide Web Conference*, 2004.

[37] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On Power-law Relationships of the Internet Topology. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1999.

[38] D. Fetterly, M. Manasse, and M. Najork. On the Evolution of Clusters of Near-Duplicate Web Pages. In *Proceedings of the 1st Conference on Latin American Web Congress*, 2003.

[39] 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.

[40] W. B. Frakes and R. A. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1992.

[41] Freenet Home Page. http://freenet.sourceforge.com/.

[42] G. W. Furnas and S. J. Rauch. Considerations for Information Environments and the NaviQue Workspace. In *Proceedings of the 3rd ACM Conference on Digital libraries*, 1998.

[43] S. L. Gerhart. Do Web Search Engines Suppress Controversy? http://firstmonday.dk/issues/issue9_1/gerhart/index.html#note5.

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

[45] Gnutella Development Home Page. http://gnutella.wego.com/.

[46] Google. http://www.google.com/.

[47] L. Gravano, C.-C. K. Chang, H. Garcia-Molina, and A. Paepcke. STARTS: Stanford Proposal for Internet Meta-searching. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of data*, 2004.

[48] L. Gravano, H. Garcia-Molina, and A. Tomasic. GlOSS: Text Source Discovery over the Internet. *ACM Transactions on Database Systems*, 24(2):229–264, 1999.

[49] A. Gulli and A. Signorini. The Indexable Web is More than 11.5 Billion Pages. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, 2005.

[50] A. Gupta and I. S. Mumick. Maintenance of Materialized Views: Problems, Techniques, and Applications. *IEEE Quarterly Bulletin on Data Engineering; Special Issue on Materialized Views and Data Warehousing*, 18(2):3–18, 1995.

[51] D. Harman. Relevance Feedback Revisited. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1992.

[52] A. Heydon and M. Najork. Mercator: A Scalable, Extensible Web Crawler. *World Wide Web*, 2(4):219–229, 1999.

[53] M. Hindman, K. Tsioutsiouliklis, and J. A. Johnson. Googlearchy: How a Few Heavily-Linked Sites Dominate Politics on the Web. http://www.princeton.edu/~mhindman/googlearchy--hindman.pdf.

[54] How Much Information? http://www.sims.berkeley.edu/research/projects/how-much-info-2003/, 2003.

[55] N. Huyn. Multiple-View Self-Maintenance in Data Warehousing Environments. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, 1997.

[56] N. Immorlica, K. Jain, M. Mahdian, and K. Talwar. Click Fraud Resistant Methods for Learning Click-Through Rates. In *Proceedings of the First International Workshop on Internet and Network. Economics*, 2005.

[57] L. Introna and H. Nissenbaum. Defining the Web: The Politics of search Engines. *IEEE Computer Magazine*, 33(1):54–62, 2000.

[58] P. Ipeirotis and L. Gravano. Distributed Search over the Hidden-Web: Hierarchical Database Sampling and Selection. In *Proceedings of the 28th International Conference on Very Large Databases*, 2002.

[59] Jakarta Lucene. http://jakarta.apache.org/lucene/docs/index.html.

[60] G. Jeh and J. Widom. Scaling Personalized Web Search. In *Proceedings of the 12th International Conference on World Wide Web*, 2003.

[61] T. Joachims. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

[62] T. Joachims, D. Freitag, and T. Mitchell. WebWatcher: A Tour Guide for the World Wide Web. In *Proceedings of International Joint Conference on Artificial Intelligence*, 1997.

[63] K. S. Jones. Statistical Interpretation of Term Specificity and its Application in Retrieval. *Journal of Documentation*, 28(1):11–20, 1972.

[64] K. S. Jones, S. Walker, and S. E. Robertson. A Probabilistic Model of Information Retrieval: Development and Comparative Experiments. *Information Processing and Management*, 36(6):809–840, 2000.

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

[66] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for Emerging Cyber-Communities. *WWW8 / Computer Networks*, 31:1481–1493, 1999.

[67] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Recommendation Systems: A Probabilistic Analysis. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, 1998.

[68] T. Lai and H. Robbins. Asymptotically Efficient Adaptive Allocation Rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

[69] R. Lempel and S. Moran. Predictive Caching and Prefetching of Query Results in Search Engines. In *Proceedings of the 12th International World Wide Web Conference*, 2003.

[70] L. Liu, C. Pu, W. Tang, and W. Han. CONQUER: A Continual Query System for Update Monitoring in the WWW. *International Journal of Computer Systems, Science and Engineering*, 14(2):99–112, 1999.

[71] Y. S. Maarek, M. Jacovi, M. Shtalhaim, S. Ur, D. Zernik, and I. Z. Ben-Shaul. WebCutter: A System for Dynamic and Tailorable Site Mapping. In *Proceedings of the 6th International World Wide Web Conference*, 1997.

[72] O. Madani and D. Decoste. Contextual Recommender Problems. In *Proceedings of the 1st International Workshop on Utility-based Data Mining*, 2005.

[73] U. Manber, M. Smith, and B. Gopal. Webglimpse: Combining Browsing and Searching. In *Proceedings of the 1997 Usenix Technical Conference*, 1997.

[74] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. AdWords and Generalized On-line Matching. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, 2005.

[75] M. Mitzenmacher. A Brief History of Lognormal and Power Law Distributions. *Internet Mathematics*, 1(2):226–251, 2004.

[76] R. Motwani and Y. Xu. Evolution of page popularity under random web graph models. In *Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2006.

[77] A. Mowshowitz and A. Kawaguchi. Bias on the Web. *Communcations of the ACM*, 45(9):56–60, 2002.

[78] MSN. http://www.msn.com/.

[79] Naughton *et al.* The Niagara Internet Query System. *IEEE Data Engineering Bulletin*, 24(2):27–33, 2001.

[80] A. Ntoulas, J. Cho, and C. Olston. What's New on the Web? The Evolution of the Web from a Search Engine Perspective. In *Proceedings of the 13th International World Wide Web Conference*, 2004.

[81] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. *Stanford Digital Library Technologies Project*, 1998.

[82] S. Pandey, D. Agarwal, D. Chakrabarti, and V. Josifovski. Bandits for Taxonomies: A Model-based Approach. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, 2007.

[83] S. Pandey, D. Chakrabarti, and D. Agarwal. Multi-armed Bandit Problems with Dependent Arms. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[84] S. Pandey and C. Olston. User-centric Web Crawling. In *Proceedings of the 14th International World Wide Web Conference*, 2005.

[85] S. Pandey and C. Olston. Crawl Ordering by Search Impact. In *Proceedings of the 1st International Conference on Web Search and Data Mining*, 2008.

[86] Y. Papakonstantinou, A. Gupta, H. Garcia-Molina, and J. D. Ullman. A Query Translation Scheme for Rapid Implementation of Wrappers. In *Proceedings of the 4th International Conference on Deductive and Object-Oriented Databases*, 1995.

[87] C. Peery, F. M. Cuenca-Acuna, R. P. Martin, and T. D. Nguyen. Wayfinder: Navigating and Sharing Information in a Decentralized World. In *Proceedings of the 30th International Conference on Very Large Databases*, 2004.

[88] J. Pitkow and P. Pirolli. Life, Death, and Lawfulness on the Electronic Frontier. In *Proceedings of the SIGCHI conference on Human factors in Computing Systems*, 1997.

[89] D. Quass and J. Widom. On-line Warehouse View Maintenance. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, 1997.

[90] P. Resnick and H. R. Varian. Recommender Systems. *Communications of the ACM*, 40(3):56–58, 1997.

[91] S. Robertson and K. S. Jones. Relevance Weighting of Search Terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[92] S. E. Robertson and S. Walker. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994.

[93] J. J. Rocchio. *Relevance Feedback in Information Retrieval.* In Gerard Salton, editor, The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[94] P. Rusmevichientong and D. Williamson. An Adaptive Algorithm for Selecting Profitable Keywords for Search-Based Advertising Services. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, 2006.

[95] G. Salton. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[96] G. Salton and M. E. Lesk. Computer Evaluation of Indexing and Text Processing. *Journal of the ACM*, 15(1):8–36, 1968.

[97] G. Salton and C. S. Yang. On the Specification of Term Values in Automatic Indexing. *Journal of Documentation*, 29(4):351–372, 1973.

[98] S. Samtani, M. K. Mohania, V. Kumar, and Y. Kambayashi. Recent Advances and Research Problems in Data Warehousing. In *Proceedings of the Workshops on Data Warehousing and Data Mining: Advances in Database Technologies*, 1998.

[99] Search Engine Watch. http://searchenginewatch.com/.

[100] C. Sherman. Are Search Engines Biased? http://searchenginewatch.com/searchday/article.php/2159431.

[101] P. Slavik. Approximation Algorithms for Set Cover and Related Problems. *Ph.D. thesis, University at Buffalo, SUNY*, 1998.

[102] B. Stone. When Mice Attack: Internet Scammers Steal Money with 'Click Fraud'. *Newsweek*, 2005.

[103] T. Suel, C. Mathur, J.-W. Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasundaram. ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval. In *Proceedings of the 6th International Workshop on the Web and Databases*, 2003.

[104] The Internet Archive. http://www.archive.org/.

[105] H. Turtle and W. B. Croft. Evaluation of an Inference Network-Based Retrieval Model. *ACM Transactions on Information Systems*, 9(3):187–222, 1991.

[106] T. Upstill, N. Craswell, and D. Hawking. Predicting Fame and Fortune: PageRank or Indegree? In *Proceedings of the 8th Australasian Document Computing Symposium*, 2003.

[107] H. R. Varian. Resources on Collaborative Filtering. http://www.sims.berkeley.edu/resources/collab/.

[108] J. Verhoeff, W. Goffman, and J. Belzer. Inefficiency of the Use of Boolean Functions for Information Retrieval Systems. *Communications of the ACM*, 4(12):557–558, 1961.

[109] N. Vidyasagar. India's Secret Army of Online Ad 'Clickers'. *The Times of India*, 2004.

[110] J. Walker. Links and Power: The Political Economy of Linking on the Web. In *Proceedings of the 13th ACM Conference on Hypertext and Hypermedia*, 2002.

[111] X. Wang. A Bandit Process with Delayed Responses. *Statistics and Probability Letters*, 48(3):303–307, 2000.

[112] S. Wartik. *Boolean Operations*. Information Retrieval: Data Structures and Algorithms, Prentice-Hall, Englewood Cliffs, NJ, 1992.

[113] WebArchive. http://webarchive.cs.ucla.edu/.

[114] J. Widom. Research Problems in Data Warehousing. In *Proceedings of the 4th International Conference on Information and Knowledge Management*, 1995.

[115] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3):38–49, 1992.

[116] J. L. Wiener, H. Gupta, W. Labio, Y. Zhuge, H. Garcia-Molina, and J. Widom. A System Prototype for Warehouse View Maintenance. In *The Workshop on Materialized Views*, 1996.

[117] K. Wittenburg and E. Sigman. Integration of Browsing, Searching, and Filtering in an Applet for Web Information Access. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1997.

[118] Wize. http://wize.com/.

[119] J. L. Wolf, M. S. Squillante, P. S. Yu, J. Sethuraman, and L. Ozsen. Optimal Crawling Strategies for Web Search Engines. In *Proceedings of the 11th International World Wide Web Conference*, 2002.

[120] Yahoo! http://www.yahoo.com/.

[121] P. S. Yu, X. Li, and B. Liu. On the Temporal Dimension of Search. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers and Posters*, 2004.

[122] Y. Zhuge, H. Garcia-Molina, and J. L. Wiener. Consistency Algorithms for Multi-Source Warehouse View Maintenance. *Distributed and Parallel Databases*, 6(1):7–40, 1998.

Chapter 9

Appendix

9.1 Overhead of Discovering New Web Sites

In this section we study the relative importance of discovering new pages on old sites, versus new pages on new sites. 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 9.1 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 9.1 shows the number of existing pages that must be downloaded in order to discover new web sites. Comparing Figures 3.4 and 9.1, we see that many more pages must be downloaded to

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

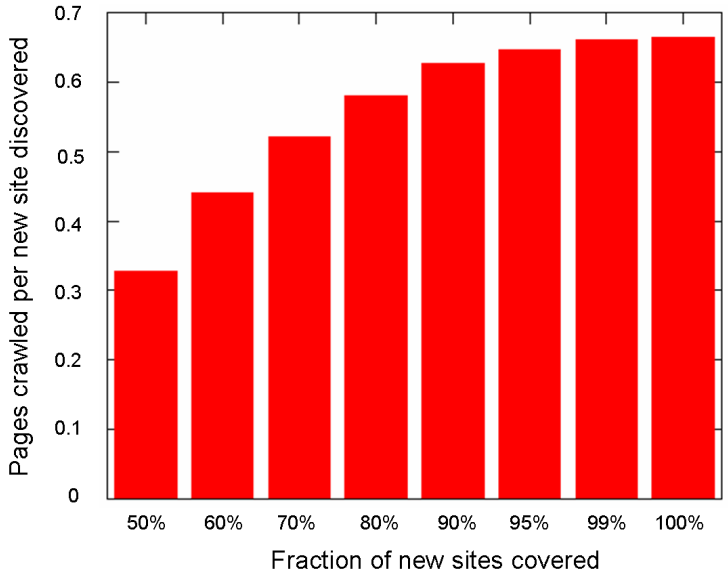

Figure 9.1: *Chile site-level discovery.*

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 download 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.

9.2 Proof of Theorem 2

We prove the following theorem from Section 5.6.2 next.

Theorem 2 *Among all pages in $\mathcal{P}$ whose quality is q, the fraction that have awareness $a_i = \frac{i}{m}$ (for $i = 0, 1, \ldots, m$) is:*

$$f(a_i|q) = \frac{\lambda}{(\lambda + F(0)) \cdot (1 - a_i)} \prod_{j=1}^{i} \frac{F(a_{j-1} \cdot q)}{\lambda + F(a_j \cdot q)} \tag{9.1}$$

where $F(x)$ is the function in Equation 5.3.

Proof: Because we consider only the pages of quality q and we focus on steady-state behavior, we will drop q and t from our notation unless it causes confusion. For example, we use $f(a)$ and $V(p)$ instead of $f(a|q)$ and $V(p,t)$ in our proof.

We consider a very short time interval dt during which every page is visited by at most one monitored user. That is, $V(p)dt < 1$ for every page p. Under this assumption we can interpret $V(p)dt$ as the probability that the page p is visited by one monitored user during the time interval dt.

Now consider the pages of awareness $a_i = \frac{i}{m}$. Since these pages are visited by at most one monitored user during dt, their awareness will either stay at a_i or increase to a_{i+1}. We use $\mathcal{P}_S(a_i)$ and $\mathcal{P}_I(a_i)$ to denote the probability that that their awareness remains at a_i or increases from a_i to a_{i+1}, respectively. The awareness of a page increases if a monitored user who was previously unaware of the page visits it. The probability that a monitored user visits p is $V(p)dt$. The probability that a random monitored user is aware of p is $(1 - a_i)$. Therefore,

$$\begin{aligned}\mathcal{P}_I(a_i) &= V(p)dt(1-a_i) = F(P(p))dt(1-a_i) \\ &= F(qa_i)dt(1-a_i) \end{aligned} \tag{9.2}$$

Similarly,

$$\mathcal{P}_S(a_i) = 1 - \mathcal{P}_I(a_i) = 1 - F(qa_i)dt(1-a_i) \tag{9.3}$$

We now compute the fraction of pages whose awareness is a_i after dt. We assume that before dt, $f(a_i)$ and $f(a_{i-1})$ fraction of pages have awareness a_i and a_{i-1}, respectively. A page will have awareness a_i after dt if (1) its awareness is a_i before dt and the awareness stays the same or (2) its awareness is a_{i-1} before dt, but it increases to a_i. Therefore, the fraction of pages at awareness a_i after dt is potentially

$$f(a_i)\mathcal{P}_S(a_i) + f(a_{i-1})\mathcal{P}_I(a_{i-1}).$$

However, under our Poisson model, a page disappears with probability λdt during the time interval dt. Therefore, only $(1 - \lambda dt)$ fraction will survive and have awareness a_i after dt:

$$[f(a_i)\mathcal{P}_S(a_i) + f(a_{i-1})\mathcal{P}_I(a_{i-1})](1 - \lambda dt)$$

Given our steady-state assumption, the fraction of pages at a_i after dt is the same as the fraction of pages at a_i before dt. Therefore,

$$f(a_i) = [f(a_i)\mathcal{P}_S(a_i) + f(a_{i-1})\mathcal{P}_I(a_{i-1})](1 - \lambda dt). \tag{9.4}$$

From Equations 9.2, 9.3 and 9.4, we get

$$\frac{f(a_i)}{f(a_{i-1})} = \frac{(1 - \lambda dt)F(qa_{i-1})dt(1 - a_{i-1})}{(\lambda + F(qa_i))dt(1 - a_i)}$$

Since we assume dt is very small, we can ignore the second order terms of dt in the above equation and simplify it to

$$\frac{f(a_i)}{f(a_{i-1})} = \frac{F(qa_{i-1})(1 - a_{i-1})}{(\lambda + F(qa_i))(1 - a_i)} \tag{9.5}$$

From the multiplication of $\frac{f(a_i)}{f(a_{i-1})} \times \frac{f(a_{i-1})}{f(a_{i-2})} \times \cdots \times \frac{f(a_1)}{f(a_0)}$, we get

$$\frac{f(a_i)}{f(a_0)} = \frac{1 - a_0}{1 - a_i} \prod_{j=1}^{i} \frac{F(qa_{j-1})}{\lambda + F(qa_j)} \tag{9.6}$$

We now compute $f(a_0)$. Among the pages with awareness a_0, $\mathcal{P}_S(a_0)$ fraction will stay at a_0 after dt. Also, λdt fraction new pages will appear, and their awareness is a_0 (recall our assumption that new pages start with zero awareness). Therefore,

$$f(a_0) = f(a_0)\mathcal{P}_S(a_0)(1 - \lambda dt) + \lambda dt \tag{9.7}$$

After rearrangement and ignoring the second order terms of dt, we get

$$f(a_0) = \frac{\lambda}{F(qa_0) + \lambda} = \frac{\lambda}{F(0) + \lambda} \tag{9.8}$$

By combining Equations 9.6 and 9.8, we get

$$\begin{aligned} f(a_i) &= f(a_0)\frac{1 - a_0}{1 - a_i} \prod_{j=1}^{i} \frac{F(qa_{j-1})}{\lambda + F(qa_j)} \\ &= \frac{\lambda}{(\lambda + F(0))(1 - a_i)} \prod_{j=1}^{i} \frac{F(qa_{j-1})}{\lambda + F(qa_j)} \end{aligned}$$

■

9.3 Performance Bound for BMMP Policies

We prove the lemmas of Section 6.6 here. First we give some background. Recall that we have defined payoff instance I such that $I(i,n)$ denotes the reward for arm i of bandit instance $S(n)$ for invocation n in payoff instance I. Since $I(i,n)$ takes a particular reward value with a certain probability, say $\mathbb{P}(I(i,n))$, we can get the probability with which payoff instance I arises by multiplying the probabilities of all $I(i,n)$'s, hence $\mathbb{P}(I) = \prod_{n=1}^{N} \prod_{i \in S(n)} \mathbb{P}(I(i,n))$. Let $\mathcal{I}$ denote the space consisting of all payoff instances, then $\sum_{I \in \mathcal{I}} \mathbb{P}(I) = 1$.

The total expected reward obtained under BPOL, $bpol(N)$, is:

$$bpol(N) \quad = \quad \sum_{I \in \mathcal{I}} \big(\mathbb{P}(I) \cdot bpol(I,N) \big)$$

where $bpol(I,N)$ denotes the total reward obtained in payoff instance I. Also,

$$bpol(I,N) \quad = \quad \sum_{n=1}^{N} Z(B(I,n),n,I)$$

where $Z(i,n,I)$ denotes the reward obtained by activating arm i of bandit instance $S(n)$ for invocation n in payoff instance I.

Since BPOL activates the arms of only those types whose budgets have not depleted yet, $Z(B(I,n),n,I) = I(B(I,n),n)$. Hence,

$$\begin{aligned} bpol(N) \quad &= \quad \sum_{I \in \mathcal{I}} \left(\mathbb{P}(I) \cdot \sum_{n=1}^{N} Z(B(I,n),n,I) \right) \\ &= \quad \sum_{I \in \mathcal{I}} \left(\mathbb{P}(I) \cdot \sum_{n=1}^{N} I(B(I,n),n) \right) \end{aligned}$$

Similarly,

$$opt(N) = \sum_{I \in \mathcal{I}} \left(\mathbb{P}(I) \cdot \sum_{n=1}^{N} I(O(I,n),n) \right)$$

Some further notation: let $\mu_{i,B}$ denote the expected reward of arm i of bandit instance B. Let $d_I(T,n)$ denote the remaining budgets of type T at invocation n under BPOL in payoff instance I. As mentioned in Section 6.6, we classify each invocation n into the following three categories.

- *Category 1:* If $\mu_{B(I,n),S(n)} \geq \mu_{O(I,n),S(n)}$.
- *Category 2:* If $\{\mu_{B(I,n),S(n)} < \mu_{O(I,n),S(n)}\} \wedge \{d_I(T,n) < I(O(I,n),n)\}$.
- *Category 3:* If $\{\mu_{B(I,n),S(n)} < \mu_{O(I,n),S(n)}\} \wedge \{d_I(T,n) \geq I(O(I,n),n)\}$.

For payoff instance I, let us denote the invocations of category k (1, 2 or 3) by $\mathcal{N}^k(I)$. Let

$$bpol_k(N) = \sum_{I \in \mathcal{I}} \left(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^k(I)} I(B(I,n),n) \right)$$

It is easy to see that $bpol(N) = \sum_{k=1}^{3} bpol_k(N)$. Similarly, $opt(N) = \sum_{k=1}^{3} opt_k(N)$ where

$$opt_k(N) = \sum_{I \in \mathcal{I}} \left(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^k(I)} I(O(I,n),n) \right)$$

Lemma 3 $opt_1(N) \leq bpol_1(N)$.

Proof: Recall that:

$$bpol_1(N) = \sum_{I \in \mathcal{I}} \left(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^1(I)} I(B(I,n),n) \right)$$

For any predicate Π we define $\{\Pi(\mathrm{x})\}$ to be the indicator function of the event $\Pi(\mathrm{x})$; i.e., $\{\Pi(\mathrm{x})\}$

$= 1$ if $\Pi(\mathrm{x})$ is true and $\Pi(\mathrm{x}) = 0$ otherwise. Using the definition of category 1,

$$\begin{aligned} & bpol_1(N) \\ &= \sum_{I\in\mathcal{I}} \left(\mathbb{P}(I) \cdot \sum_{n=1}^{N} \left(\{\mu_{B(I,n),S(n)} \geq \mu_{O(I,n),S(n)}\} \cdot I(B(I,n),n) \right) \right) \\ &= \sum_{n=1}^{N} \sum_{I\in\mathcal{I}} \left(\mathbb{P}(I) \cdot \{\mu_{B(I,n),S(n)} \geq \mu_{O(I,n),S(n)}\} \cdot I(B(I,n),n) \right) \end{aligned}$$

For a given n we divide payoff instance I into two parts I_1 and I_2 where I_1 consists of $I(i,n')$'s for $n' < n$ and I_2 consist of $I(i,n')$'s for $n' \geq n$. By definition, the arm selected by BPOL (and OPT) at the n^{th} invocation only depends on I_1. Hence, we denote $B(I,n)$ and $O(I,n)$ by $B(I_1,n)$ and $O(I_1,n)$ for the rest of this proof. Clearly, payoff instance space $\mathcal{I} = \mathcal{I}_1 \times \mathcal{I}_2$ where $\mathcal{I}_1$ and $\mathcal{I}_2$ denote the payoff instance spaces for I_1 and I_2 respectively and $\times$ denotes the cross product.

$$\begin{aligned} & bpol_1(N) \\ &= \sum_{n=1}^{N} \sum_{I_1\in\mathcal{I}_1} \sum_{I_2\in\mathcal{I}_2} \left(\mathbb{P}(I_1) \cdot \mathbb{P}(I_2) \cdot \{\mu_{B(I_1,n),S(n)} \geq \mu_{O(I_1,n),S(n)}\} \cdot I_2(B(I_1,n),n) \right) \\ &= \sum_{n=1}^{N} \sum_{I_1\in\mathcal{I}_1} \left(\mathbb{P}(I_1) \cdot \{\mu_{B(I_1,n),S(n)} \geq \mu_{O(I_1,n),S(n)}\} \cdot \sum_{I_2\in\mathcal{I}_2} \left(\mathbb{P}(I_2) \cdot I_2(B(I_1,n),n) \right) \right) \\ &= \sum_{n=1}^{N} \sum_{I_1\in\mathcal{I}_1} \left(\mathbb{P}(I_1) \cdot \{\mu_{B(I_1,n),S(n)} \geq \mu_{O(I_1,n),S(n)}\} \cdot \mu_{B(I_1,n),S(n)} \right) \end{aligned}$$

Similarly,

$$opt_1(N) = \sum_{n=1}^{N} \sum_{I_1\in\mathcal{I}_1} \left(\mathbb{P}(I_1) \cdot \{\mu_{B(I_1,n),S(n)} \geq \mu_{O(I_1,n),S(n)}\} \cdot \mu_{O(I_1,n),S(n)} \right)$$

Since $\mu_{B(I_1,n),S(n)} \geq \mu_{O(I_1,n),S(n)}$ in the terms contributing to the above summations, we get $bpol_1(N) \geq opt_1(N)$.

■

Lemma 4 *$opt_2(N) \leq bpol(N) + (|\mathcal{T}| \cdot r_{max})$ where $|\mathcal{T}|$ denotes the number of arm types and r_{max} denotes the maximum reward.*

Proof: Recall that $\mathcal{N}^2(I)$ denotes the sequence of invocations of category 2 for payoff instance I. Let us denote the set of $O(I,n)$'s for $n \in \mathcal{N}^2(I)$ by $\mathcal{O}^2(I)$, *i.e.*, $\mathcal{O}^2(I) = \{O(I,n) \mid n \in \mathcal{N}^2(I)\}$. Furthermore, let $\mathcal{T}^2(I)$ denote the set of types covering the arms of set $\mathcal{O}^2(I)$. Consider any type T from set $\mathcal{T}^2(I)$. By definition of category 2, we know that the remaining budget of type T drops below r_{max} at some point in BPOL. Therefore, $d_I(T, N+1) < r_{max}$ (here $d_I(T, N+1)$ denotes the remaining budget of type T in BPOL after all N bandit instances of sequence S have been invoked).

Since the total reward given by the arms of a type is the difference of its initial budget $d_I(T,1)$ and the final budget $d_I(T, N+1)$,

$$\begin{aligned}
bpol(I,N) &= \sum_{T \in \mathcal{T}} \Big(d_I(T,1) - d_I(T,N+1) \Big) \\
&\geq \sum_{T \in \mathcal{T}^2(I)} \Big(d_I(T,1) - d_I(T,N+1) \Big) \\
&\geq \sum_{T \in \mathcal{T}^2(I)} \Big(d_I(T,1) - r_{max} \Big) \\
&= \sum_{T \in \mathcal{T}^2(I)} \big(d_I(T,1) \big) - \big(|\mathcal{T}^2(I)| \cdot r_{max} \big) \\
&\geq \sum_{T \in \mathcal{T}^2(I)} \big(d_I(T,1) \big) - \big(|\mathcal{T}| \cdot r_{max} \big)
\end{aligned}$$

By rearranging the terms,

$$\sum_{T \in \mathcal{T}^2(I)} d_I(T,1) \leq bpol(I,N) + \big(|\mathcal{T}| \cdot r_{max} \big)$$

Now we derive a bound for $opt_2(N)$. Recall that:

$$opt_2(N) = \sum_{I \in \mathcal{I}} \Big(\mathbb{P}(I) \cdot \sum_{n \in \mathcal{N}^2(I)} \big(I(O(I,n), n) \big) \Big)$$

Since we know that the total reward given by the arms of a type can never exceed its initial

budget,

$$\begin{aligned} opt_2(N) &\leq \sum_{I\in\mathcal{I}}\Big(\mathbb{P}(I)\cdot\sum_{T\in\mathcal{T}^2(I)} d_I(T,1)\Big) \\ &\leq \sum_{I\in\mathcal{I}}\Big(\mathbb{P}(I)\cdot\Big(bpol(I,N)+\big(|\mathcal{T}|\cdot r_{max}\big)\Big)\Big) \\ &= \sum_{I\in\mathcal{I}}\Big(\mathbb{P}(I)\cdot bpol(I,N)\Big)+\Big(|\mathcal{T}|\cdot r_{max}\Big) \\ &= bpol(N)+\Big(|\mathcal{T}|\cdot r_{max}\Big) \end{aligned}$$

Hence, $opt_2(N) \leq bpol(N)+\big(|\mathcal{T}|\cdot r_{max}\big)$. ■

Lemma 5 *$opt_3(N) = O(f(N))$ where $f(n)$ denotes the expected number of mistakes made by POL for any finite n.*

Proof: Recall that:

$$opt_3(N) = \sum_{I\in\mathcal{I}}\Big(\mathbb{P}(I)\cdot\sum_{n\in\mathcal{N}^3(I)}\big(I(O(I,n),n)\big)\Big)$$

Since $I(i,n)\leq r_{max}$ for all i and n,

$$\begin{aligned} opt_3(N) &\leq \sum_{I\in\mathcal{I}}\Big(\mathbb{P}(I)\cdot\sum_{n\in\mathcal{N}^3(I)} r_{max}\Big) \\ &= \sum_{I\in\mathcal{I}}\Big(\mathbb{P}(I)\cdot\sum_{n\in\mathcal{N}^3(I)}\big(r_{max}\cdot\sum_{i=1}^{|\mathcal{B}|}\{S(n)=i\}\big)\Big) \\ &= r_{max}\cdot\sum_{i=1}^{|\mathcal{B}|} C_i(N) \end{aligned}$$

where $C_i(N)$ denotes the expected number of times bandit instance B_i happens to be invoked during the invocations of category 3:

$$C_i(N) = \sum_{I\in\mathcal{I}}\Big(\mathbb{P}(I)\cdot\sum_{n\in\mathcal{N}^3(I)}\{S(n)=i\}\Big)$$

In Lemma 6 we show that for every $B_i \in \mathcal{B}$, $C_i(N) = O(f(N))$. Hence, $opt_3(N) = O(f(N))$. ■

Lemma 6 *For every bandit instance B_i in $\mathcal{B}$, $C_i(N) = O(f(N))$.*

Proof: Let $\mathcal{S}^i$ denote the sequence of invocations at which bandit instance B_i is invoked in sequence S, *i.e.*, $\mathcal{S}^i = \{n \mid S(n) = i\}$. We analyze BPOL now. Recall that in BPOL as the arms of a bandit instance run out of budget, they are being successively discarded. Let $\mathcal{S}^i_d(I)$ denote the sequence of those invocations at which an arm(s) of B_i is discarded in payoff instance I. We call the sequence of invocations of B_i between two successive invocations of sequence $\mathcal{S}^i_d(I)$ a *batch*. The number of batches is upper bounded by the number of arms of B_i (which is finite). Clearly, within a batch the set of available arms of bandit instance B_i remains fixed and POL operates on them independently and uninterruptedly.

Consider a batch in payoff instance I. Now pick an invocation n of category 3 in the batch when bandit instance B_i is invoked. By definition of category 3, both arms $B(I,n)$ and $O(I,n)$ are available to choose for POL at n. By choosing arm $B(I,n)$ over $O(I,n)$, POL makes a *mistake* of choosing suboptimal arm since $\mu_{B(I,n),S(n)} < \mu_{B(I,n),S(n)}$. Hence, we have shown that in a given batch, each invocation of category 3 is caused by a mistake of POL. Given the performance bound of POL, the expected number of such mistakes in a batch is $f(\textit{batch length})$, hence $O(f(N))$. Since the number of batches is finite, $C_i(N)$ is $O(f(N))$. ■

9.4 Performance Bound for MIX

The optimal policy for the unbudgeted unknown-CTR advertisement problem is to display the C ads of the highest expected reward $(c_{i,j} \cdot b_{i,j})$ for each query phrase. We prove that MIX makes $O(ln\ N)$ mistakes, on expectation, for any $C \geq 1$ where N denotes the number of queries answered. A mistake occurs when an ad of less expected reward $(c_{i,j} \cdot b_{i,j})$ is displayed for a query phrase while keeping an ad of higher expected reward out. Since MIX is adapted from UCB, our proof is largely inherited from [6].

Consider query phrase $Q_j \in \mathcal{Q}$. Let $\mathcal{A}_j$ denote the set of ads for phrase Q_j and let $\mathcal{G}_j$ denote the set of C ads of the highest expected rewards. For simplicity, we assume that each ad has a unique expected reward. Clearly, a mistake occurs when an ad from set $\mathcal{A}_j - \mathcal{G}_j$ is displayed for Q_j. We

denote the number of times ad $a_{i,j}$ is displayed by MIX by $m_{i,j}(n_j)$ where n_j denotes the number of times query phrase Q_j has been answered so far.

Theorem 4 *For any ad $a_{i,j} \in \{\mathcal{A}_j - \mathcal{G}_j\}$, $\mathbb{E}(m_{i,j}(N_j)) = O(ln\ N_j)$ where $\mathbb{E}$ denotes the expectation.*

Proof: Recall the priority function of MIX:

$$P_{i,j} = \begin{cases} \left(\hat{c}_{i,j} + \sqrt{\frac{2\ ln\ n_j}{n_{i,j}}}\right) \cdot b_{i,j} & \text{if } n_{i,j} > 0 \\ \infty & \text{otherwise} \end{cases}$$

Here $\hat{c}_{i,j}$ denote the current CTR estimate of ad $a_{i,j}$ based on the past observations, $b_{i,j}$ is its bid value, $n_{i,j}$ denotes the number of times $a_{i,j}$ has been displayed so far for phrase Q_j and n_j denotes the number of times phrase Q_j has been queried so far in the day. We denote the CTR estimated after $n_{i,j}$ display of ads by $\hat{c}_{i,j}(n_{i,j})$. For notation convenience, we denote $\sqrt{\frac{2\ ln\ n_j}{n_{i,j}}}$ in the priority function by $g(n_j, n_{i,j})$.

Some further notation: For any predicate Π we define $\{\Pi(\mathrm{x})\}$ to be the indicator function of the event $\Pi(\mathrm{x})$; i.e., $\{\Pi(\mathrm{x})\} = 1$ if $\Pi(\mathrm{x})$ is true and $\Pi(\mathrm{x}) = 0$ otherwise. For the n_j^{th} occurrence of query phrase Q_j, let $L_j(n_j)$ denote the ad of lowest priority value in G_j and let $\mathcal{U}_j(n_j)$ denote the set of C ads displayed by MIX. Consider $a_{i,j} \in \{\mathcal{A}_j - \mathcal{G}_j\}$, then:

$$\begin{aligned} m_{i,j}(N_j) &= 1 + \sum_{n_j=|\mathcal{A}_j|+1}^{N_j} \{a_{i,j} \in \mathcal{U}_j(n_j)\} \\ &\qquad \{\text{since each ad from } \mathcal{A}_j \text{ is displayed once initially}\} \\ &\leq l + \sum_{n_j=|\mathcal{A}_j|+1}^{N_j} \{a_{i,j} \in \mathcal{U}_j(n_j),\ m_{i,j}(n_j - 1) \geq l\} \\ &\qquad \{\text{where } l \text{ is an arbitrary positive integer}\} \end{aligned}$$

In order for ad $a_{i,j}$ to be displayed on the n_j^{th} occurrence of query phrase Q_j, its priority must

be greater than or equal to the priority of $L_j(n_j)$, hence,

$$
\begin{aligned}
& m_{i,j}(N_j) \\
\leq\ & l + \sum_{n_j=|\mathcal{A}_j|+1}^{N_j} \sum_{a_{k,j}\in\mathcal{G}_j} \Bigg\{ \Big(\hat{c}_{i,j}(m_{i,j}(n_j-1)) + g(n_j-1, m_{i,j}(n_j-1))\Big)\cdot b_{i,j} \geq \\
& \Big(\hat{c}_{k,j}(m_{k,j}(n_j-1)) + g(n_j-1, m_{k,j}(n_j-1))\Big)\cdot b_{k,j},\quad a_{k,j} = L_j(n_j),\quad m_{i,j}(n_j-1) \geq l \Bigg\} \\
\leq\ & l + \sum_{n_j=|\mathcal{A}_j|+1}^{N_j} \sum_{a_{k,j}\in\mathcal{G}_j} \Bigg\{ \max_{l\leq s_i<n_j} \Big(\hat{c}_{i,j}(s_i) + g(n_j-1, s_i)\Big)\cdot b_{i,j} \geq \\
& \min_{0<s_k<n_j} \Big(\hat{c}_{k,j}(s_k) + g(n_j-1, s_k)\Big)\cdot b_{k,j},\quad a_{k,j} = L_j(n_j) \Bigg\} \\
& \qquad\qquad \{\text{since } \forall a_{k,j},\ 0 < m_{k,j}(n_j-1) < n_j\} \\
\leq\ & l + \sum_{n_j=1}^{N_j} \sum_{a_{k,j}\in\mathcal{G}_j} \sum_{s_k=1}^{n_j} \sum_{s_i=l}^{n_j} \Bigg\{ \Big(\hat{c}_{i,j}(s_i) + g(n_j, s_i)\Big)\cdot b_{i,j} \geq \Big(\hat{c}_{k,j}(s_k) + g(n_j, s_k)\Big)\cdot b_{k,j}, \\
& a_{k,j} = L_j(n_j+1) \Bigg\}
\end{aligned}
$$

Now we focus our attention on $\Big(\hat{c}_{i,j}(s_i) + g(n_j, s_i)\Big)\cdot b_{i,j} \geq \Big(\hat{c}_{k,j}(s_k) + g(n_j, s_k)\Big)\cdot b_{k,j}$ where $a_{k,j} \in \mathcal{G}_j$. Let us call it condition Y. Observe the following three terms:

$$\hat{c}_{k,j}(s_k) \leq c_{k,j} - g(n_j, s_k) \tag{9.9}$$

$$\hat{c}_{i,j}(s_i) \geq c_{i,j} + g(n_j, s_i) \tag{9.10}$$

$$c_{k,j}\cdot b_{k,j} < c_{i,j}\cdot b_{i,j} + 2\cdot g(n_j, s_i)\cdot b_{i,j} \tag{9.11}$$

It is easy to see that if none of these terms are true, then condition Y can not hold true. Hence, we can replace condition Y in the above equation by condition $\{1 \vee 2 \vee 3\}$ since the replacement

does not make the RHS any smaller.

$$
\begin{aligned}
& m_{i,j}(N_j) \\
& \leq \quad l + \sum_{n_j=1}^{N_j} \sum_{a_{k,j} \in \mathcal{G}_j} \sum_{s_k=1}^{n_j} \sum_{s_i=l}^{n_j} \Bigg(\Big\{ \hat{c}_{k,j}(s_k) \leq c_{k,j} - g(n_j, s_k) \Big\} + \Big\{ \hat{c}_{i,j}(s_i) \geq c_{i,j} + g(n_j, s_i) \Big\} \\
& \qquad\qquad + \Big\{ c_{k,j} \cdot b_{k,j} < c_{i,j} \cdot b_{i,j} + 2 \cdot g(n_j, s_i) \cdot b_{i,j} \Big\} \Bigg) \cdot \{ a_{k,j} = L_j(n_j + 1) \}
\end{aligned}
\tag{9.12}
$$

We bound the probability of Terms 9.9 and 9.10 using Chernoff-Hoeffding bound:

$$
\Pr\{ \hat{c}_{k,j}(s_k) \leq c_{k,j} - g(n_j, s_k) \} \leq e^{-4 \cdot (ln\ n_j)} = n_j^{-4}
$$

$$
\Pr\{ \hat{c}_{i,j}(s_i) \geq c_{i,j} + g(n_j, s_i) \} \leq e^{-4 \cdot (ln\ n_j)} = n_j^{-4}
$$

Recall that we can set l to any positive integer. We set l to $l_o = \lceil (8 \cdot (ln\ N_j) \cdot b_{i,j}^2) / \Delta_{i,j}^2 \rceil$ where $\Delta_{i,j} = \min_{a_{k,j} \in \mathcal{G}_j} \left(c_{k,j} \cdot b_{k,j} - c_{i,j} \cdot b_{i,j} \right)$. For $a_{k,j} \in \mathcal{G}_j$ and $s_i \geq l_o$, Term 9.11 is false because:

$$
\begin{aligned}
c_{k,j} \cdot b_{k,j} - c_{i,j} \cdot b_{i,j} - 2 \cdot g(n_j, s_i) \cdot b_{i,j} &= c_{k,j} \cdot b_{i,j} - c_{i,j} \cdot b_{i,j} - 2 \cdot \sqrt{2(ln\ n_j)/s_i} \cdot b_{i,j} \\
&\geq c_{k,j} \cdot b_{k,j} - c_{i,j} \cdot b_{i,j} - 2 \cdot \sqrt{2(ln\ N_j)/l_o} \cdot b_{i,j} \\
&= c_{k,j} \cdot b_{k,j} - c_{i,j} \cdot b_{i,j} - \Delta_{i,j} \\
&\geq 0
\end{aligned}
$$

Hence, by taking expectation of Equation 9.12,

$$
\begin{aligned}
&\mathbb{E}(m_{i,j}(N_j)) \\
&\quad \leq \; l + \sum_{n_j=1}^{N_j} \sum_{a_{k,j} \in \mathcal{G}_j} \sum_{s_k=1}^{n_j} \sum_{s_i=l}^{n_j} \Big(\Pr\{\hat{c}_{k,j}(s_k) \leq c_{k,j} - g(n_j, s_k)\} + \Pr\{\hat{c}_{i,j}(s_i) \geq c_{i,j} + g(n_j, s_i)\} \\
&\qquad\qquad + \Pr\{c_{k,j} \cdot b_{k,j} < c_{i,j} \cdot b_{i,j} + 2 \cdot g(n_j, s_i) \cdot b_{i,j}\} \Big) \cdot \Pr\{a_{k,j} = L_j(n_j + 1)\} \\
&\quad \leq \; l_o + \sum_{n_j=1}^{N_j} \sum_{a_{k,j} \in \mathcal{G}_j} \sum_{s_k=1}^{n_j} \sum_{s_i=l_o}^{n_j} \Big(\Pr\{\hat{c}_{k,j}(s_k) \leq c_{k,j} - g(n_j, s_k)\} \\
&\qquad\qquad + \Pr\{\hat{c}_{i,j}(s_i) \geq c_{i,j} + g(n_j, s_i)\} \Big) \cdot \Pr\{a_{k,j} = L_j(n_j + 1)\} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \{ \text{ by setting } l = l_o \; \} \\
&\quad \leq \; l_o + \sum_{n_j=1}^{\infty} \sum_{s_k=1}^{n_j} \sum_{s_i=1}^{n_j} \sum_{a_{k,j} \in \mathcal{G}_j} 2 \cdot n_j^{-4} \cdot \Pr\{a_{k,j} = L_j(n_j + 1)\} \\
&\quad = \; l_o + \sum_{n_j=1}^{\infty} \sum_{s_k=1}^{n_j} \sum_{s_i=1}^{n_j} 2 \cdot n_j^{-4} \\
&\quad \leq \; \left\lceil \frac{8 \cdot (ln\ N_j) \cdot b_{i,j}^2}{\Delta_{i,j}^2} \right\rceil + (1 + \frac{\pi^2}{3})
\end{aligned}
$$

Hence $\mathbb{E}(m_{i,j}(N_j)) = O(ln\ N_j)$. ■

Given the above result it is clear that the total expected number of mistakes made by MIX for N queries, $\sum_{Q_j \in \mathcal{Q}} \sum_{a_{i,j} \in \mathcal{A}_j - \mathcal{G}_j} \mathbb{E}(m_{i,j}(N_j))$, is $O(ln\ N)$.