Ranking the Web Frontier

Nadav Eiron, Kevin S. McCurley, John A. Tomlin
IBM Almaden Research Center

ABSTRACT

The celebrated PageRank algorithm has proved to be a very effective paradigm for ranking results of web search algorithms. In this paper we refine this basic paradigm to take into account several evolving prominent features of the web, and propose several algorithmic innovations. First, we analyze features of the rapidly growing "frontier" of the web, namely the part of the web that crawlers are unable to cover for one reason or another. We analyze the effect of these pages and find it to be significant. We suggest ways to improve the quality of ranking by modeling the growing presence of "link rot" on the web as more sites and pages fall out of maintenance. Finally we suggest new methods of ranking that are motivated by the hierarchical structure of the web, are more efficient than PageRank, and may be more resistant to direct manipulation.

Categories and Subject Descriptors

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

General Terms

Algorithms, Theory

Keywords

Ranking, PageRank, Hypertext

1. INTRODUCTION

The PageRank algorithm [28] of Page et. al. can be used to dramatically improve the quality of results from web search engines. The underlying idea of PageRank is to use the stationary distribution of a random walk on the web graph in order to assign relative ranks to the pages. While this basic paradigm has proven to be remarkably effective in practice, it leaves considerable room for enhancement to reflect emerging features of the web, and the global nature of the calculation presents a computational challenge as the web continues to grow. In this paper we examine several issues surrounding the basic paradigm of PageRank, and suggest several improvements over previously published work. Our main contributions are in two areas; an analysis and algorithms to handle the "dangling node" problem, and in devising ways to reduce the difficulty of computing pagerank while simultaneously addressing problems of rank manipulation.

The remainder of the paper is structured as follows. In Section 2 we define a basic problem addressed here, namely the problem of dangling nodes. In Section 3 we describe the data set that

we worked on for our experimental results. Section 4 describes previous treatments of the dangling node problem, and provides a framework that addressed some ambiguities in the previous work. In Section 5 we provide motivation for why dangling pages are important to the overall ranking process. In Section 6 we describe three new algorithms that incorporate information from different types of dangling links, with the goal of producing better ranking functions. In Sections 7 and 8 we consider two simpler forms of PageRank that exploit the hierarchical organization of information on the web and may have advantages for resisting manipulation. In Section 9 we describe some experimental results from some large-scale computations of PageRank and in particular why PageRank appears to be subject to direct manipulation today.

In principle the PageRank algorithm is simple. The Web is modeled by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and the rank or "importance" x_i of each for the n pages $i \in \mathcal{V}$ is defined recursively in terms of the pages which point to it:

$$x_i = \sum_{(j,i) \in \mathcal{E}} a_{ij} x_j, \qquad (1)$$

or in matrix terms, $x = Ax$. It is the form of the coefficients a_{ij} which gives rise to so many questions, including those considered here. For the system (1) to have a useful solution, A must be (column) stochastic, that is we must have $e^T A = e^T$, where e^T is the vector of all ones. Then x is the *principal eigenvector* corresponding to the principal eigenvalue unity[18].

The standard "ideal" assumption is that $\mathcal{G}$ is strongly connected, that is that every page can be reached by clicking through from every other page. In this case it is assumed that the a_{ij} can be given by $1/d_j$, where d_j is the out-degree of page j, in other words, it is assumed that a web surfer will follow the outlinks from the page with equal probability. However, in practice, the web is not strongly connected, and adjustments to the ideal case must be made.

Among the most common devices is the addition of links from pages with no outlinks to some or all of the other pages, and the use of "random jumps" not associated with actual links (sometimes referred to as teleportation). This device is usually represented by modifying (1) to be of the form:

$$x = \left[(1 - \alpha) f e^T + \alpha A \right] x \qquad (2)$$

where α is the probability of following an actual outlink from a page, $(1 - \alpha)$ is the probability of taking a "random jump" rather than following a link, and f is a stochastic vector (i.e. $e^T f = 1$). In other words the stochastic matrix in (2) is a convex combination of our A and a rank one matrix.

At this point it is convenient to observe [32] that solving the system (2) is equivalent to defining an additional, "virtual" node

$n + 1$, and defining an augmented system:

$$\begin{pmatrix} x \\ x_{n+1} \end{pmatrix} = \begin{pmatrix} \alpha A & f \\ (1-\alpha)e^T & 0 \end{pmatrix} \begin{pmatrix} x \\ x_{n+1} \end{pmatrix} \qquad (3)$$

It is easy to verify that the solutions of this system are in one-to-one correspondence with those of the modified problem (2). This augmentation technique will be pursued further below.

Rather than being a single well-specified algorithm, the Page-Rank algorithm has multiple variations that are collectively referred to as "PageRank". Included among these variations are:

- Different choices for the teleportation parameter α can produce different convergence rates and different rankings, but it has become standard to use the value $\alpha = 0.85$, which appears to strike a balance between achieving rapid convergence with minimal perturbation to the rankings.

- When a random jump is taken, it can be taken to one of an arbitrary set of pages, as defined by f. Previous suggestions include the choice of a uniform distribution among all pages (i.e. $f = e/n$), among a set of trusted "seed sites", uniformly among the set of all "top-level" pages of sites [28], or a personalized set of preferred pages.

- Various options for the handling of so-called "dangling nodes", namely nodes that either have no outlinks or for which no outlinks are known.

The last item has received relatively little attention in the past, but is the focus of the first part of this paper. Previous treatments have tended to treat the dangling nodes only in passing, with rather little detail on the effect of these dangling nodes, or simply omitting this issue entirely. In passing we note that some alternatives to Page-Rank are not sensitive to dangling nodes, but have received much less attention than PageRank (e.g., [2]). This attitude was perhaps justified in the days of a largely static web, but we believe that the relative effect of these nodes has increased over time, for a number of reasons. In particular, we believe that it is almost impossible to avoid the situation of having a crawl with more dangling pages than non-dangling pages. Moreover, we believe that different classes of dangling nodes provide useful information for ranking of non-dangling pages, and we describe improved methods that take these factors into account.

At last count our crawler has discovered links to over six billion URLs, and the introduction of sites with dynamic content leads us to believe that there is no reasonable upper bound on the number of URLs that one can find. This huge number of URLs means that the computational task of computing PageRank is formidable. There have been multiple papers [1, 20, 24, 22, 23, 9] that address the problem of efficient implementation. We address this point also, in section 7, by modifying the problem.

2. THE WEB FRONTIER

We use the term "frontier" to refer to the set of dangling pages. Pages may be dangling for a variety of reasons, the most obvious of which is that the crawler might not yet have crawled them. At the time that the original PageRank algorithm was conceived, most of the web was served as a static set of human-edited documents residing in filesystems. Under such an assumption, it made sense to reason about the percentage of the web covered by a crawl, and to attempt to index "all of the web". In the last few years the web has evolved to have a large number of sites that produce dynamic content, and are driven by databases. It has been estimated that dynamic pages are 100 times more numerous than static pages [19], but in fact the situation is even worse than this. There are an essentially infinite number of URLs (limited only by the size of the namespace, which is at least 64^{2000}). For example,

www.amazon.com currently uses dynamically generated URLs with a session ID embedded in them. Unless prior knowledge of this fact is given to a crawler, it can find an essentially unbounded number of URLs to crawl at this one site alone. [1]

The problem of indexing and ranking "the indexable web" therefore starts with the problem of identifying what constitutes "the usable web". One possible approach to avoiding all database-driven dynamic pages is to avoid crawling URLs that contain a '?' character in them. Unfortunately this heuristic is unreliable, and in particular it has become commonplace to use such query fragments in URLs that produce static content.

Moreover, the database-driven pages can be (and often are) presented with URLs that do not contain a '?' character in them, with arguments for a database query encoded differently in the URL. Because they appear to a crawler exactly as a static file-based web page, it is difficult to prune the number of pages to be crawled. As a result, search engines are now faced with a vast excess of potentially crawlable and indexable content. The fact that an essentially unbounded number of URLs can be found suggests that the steady state for any crawler and search engine is for there to always be a frontier of uncrawled pages, and that this frontier will always be relatively large.

There are many other reasons why a page might be considered a dangling page. It might be protected by a robots.txt, and therefore off-limits under the standard practice of crawling. Such pages may however contain very high-quality information, and therefore be of great interest to readers and worthy of indexing. It is important to note that even if a page cannot be crawled, it may still be indexable using its anchor text. While anchor text is not a substitute for full text indexing, it has proved to be remarkably effective in satisfying most web search queries [11, 14, 17]. Thus it is important to be able to evaluate the relative ranking of dangling nodes, and in fact when we calculated PageRank we found several pages in the top 100 that were protected by robots.txt. Paradoxically, there may even be good reasons to calculate a rank of a page that no longer exists (e.g., a significant document that was removed for political or legal reasons).

Another reason for dangling nodes is pages that genuinely have no outlink. For example, most PostScript and PDF files on the web contain no embedded outlinks, and yet the content tends to be of relatively high quality. A URL might also be a dangling page if it has a meta tag indicating that links should not be followed from the page, or if it requires authentication (e.g., most of the Wall Street Journal site). Other reasons for dangling nodes include those that return a 500 class response at crawl time due to a configuration problem, or servers that are not resolvable in DNS, or routing problems, etc.

Our recent experience with crawling has shown that even after crawling well over a billion pages, the number of uncrawled pages still far exceeds the number of crawled pages. At the time that we took a snapshot for this paper, we had crawled approximately 1.1 billion pages but the crawler knew about 4.75 billion pages. This is partly a function of crawl ordering, but our experience shows that the fraction of dangling pages is likely to always remain large for any reasonable expenditure of resources, and that there will be many pages with dangling links from them. In Figure 1 we show the snapshot of the distribution of the number of dangling outlinks from a given page after crawling approximately 100 million pages. Nearly 20% of all pages turn out to have none of their outlinks

---

[1] Andrei Broder has mused that the size of the web depends strongly on whether his laptop is on the web, since it can be configured to produce links to an essentially infinite number of URLs that are all crawlable.

crawled yet, and only 25% have had their outlinks fully explored, with the rest in the middle. Thus if a crawl was halted at this point, 75% of the pages would still have dangling outlinks. By adjusting the crawl strategy we can change the shape of the graph, but the size of the uncrawled queue will always be a large fraction of the total URLs known.
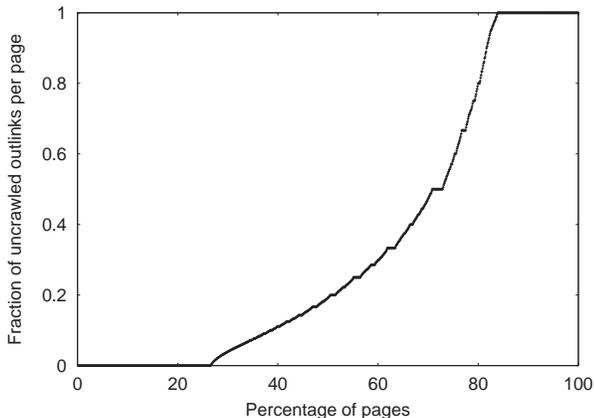

Figure 1: After truncating our billion page crawl at 100 million pages, we have crawled some fraction of the outlinks for each of the pages discovered so far. Here we show the cumulative distribution of the number of pages vs. the fraction of their outlinks that have been crawled. We have only crawled all the outlinks of 25% of the pages, which means that most pages have links into the frontier.

All of this suggests that the problem of dealing with large numbers of dangling nodes is pervasive and important. If we omit the ranking of dangling nodes, we are ignoring a majority of the potential pages, and we are neglecting their effect on the rankings of pages we have already crawled. Moreover, given that a crawler is faced with a limitless supply of URLs to crawl, it becomes important to assign ranks to dangling pages in order to efficiently manage crawling resources [1, 10].

3. EXPERIMENTAL METHODOLOGY

The current work is based upon observations from a large crawl performed at IBM Almaden. We began by extracting the links from more than a billion pages, with more than 37 billion links, extending to more than 4.75 billion URLs. The process of extracting the links and constructing the graph took a considerable amount of computation on a cluster of machines. Since we had only 32-bit machines available for most of this work, we spent a considerable amount of effort to reduce the data set down to something manageable. We began by replacing URLs by 9-byte hash values, which produced a data set of approximately 650 gigabytes for the links alone (saving 4.75 billion URLs alone consumes over 300 gigabytes before compression!). We then separated the dangling links from the non-dangling links, leaving us with approximately 19.2 billion non-dangling links. Following this step we assigned 4-byte IDs to the URLs, and rewrote the graph in a much more compact format. Aside from the treatment of dangling pages, this is similar to the procedure used by previous authors [20, 8], though on a much larger scale.

A large amount of effort went into this data gathering step, and the amount of time dedicated to the actual PageRank calculations was much shorter. It is therefore imperative that any evaluation of algorithms for ranking of pages should take the data extraction and preprocessing into account. On the other hand, the link gathering and preprocessing of links can theoretically be done in parallel with the crawl, whereas the actual PageRank calculation is generally done offline (but see [9]). For this reason, methods that accelerate the convergence of the iterative PageRank algorithm [24, 3, 22, 23] can still be very effective. Moreover, algorithms that can work with only local information in an online fashion [1] are all the more appealing given that PageRank requires so much data. For smaller data sets (e.g., intranets) this would not be much of a problem.

4. HANDLING DANGLING PAGES

We begin by reviewing some of the suggestions that have been made previously for handling the dangling pages. In the original Pagerank paper [28] the authors suggested simply removing the links to dangling pages from the graph, and calculating the Page-Rank on the remaining pages. After doing so it was suggested that they can be "added back in" without significantly affecting the results, but the details of how to do this are lacking (as is the analysis of the effect from dangling nodes). In [23] the authors suggested removing the dangling nodes and then re-inserting them "for the last few iterations". Note that removing the dangling links entirely will skew the results on the non-dangling nodes somewhat, since the outdegrees from the pages are adjusted to reflect the lack of links to dangling nodes. This approach was also suggested in [6], where they also note that this approach seems preferable to leaving them in the calculation. This is also the approach used in [20].

Note that the process of removing dangling nodes may itself produce new dangling nodes, and the process could therefore be repeated iteratively until no dangling nodes remain. In theory this may remove everything, but in practice the process terminates quickly on the web. We chose to omit this iterative step in our process, as we preferred to save as much information as possible about the graph and compute ranks for dangling pages anyway.

One alternative method for handling dangling pages is to jump to a randomly selected page with probability 1 from every dangling node. This approach is mentioned in [23] and treated more formally as follows. Suppose the nodes $\mathcal{V}$ of the graph ($n = |\mathcal{V}|$) can be partitioned into two subsets:

1. $\mathcal{C}$ corresponds to a completely (strongly) connected subgraph ($|\mathcal{C}| = m$).
2. The remaining nodes in subset $\mathcal{D}$ have links from $\mathcal{C}$ but no outlinks.

In addition, following the approach in (3) assume a virtual $(n+1)^{th}$ node to and from which random jumps may be made. The new node set is denoted $\mathcal{V}' = \mathcal{V} \cup \{n+1\}$. In addition we add new edges $(i, n+1)$ for $i \in \mathcal{D}$ and $(n+1, j)$ for $j \in \mathcal{C}$ to define an expanded edge set $\mathcal{E}'$.

Suitably partitioning the matrix and vector in (3) the PageRank of the nodes in $\mathcal{V}'$ may be computed via the principal eigenvector computation:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \alpha C & O & e/m \\ \alpha D & O & 0 \\ (1-\alpha)e^T & e^T & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad (4)$$

where, if d_j is the out-degree of node j

$$c_{ij} = \begin{cases} d_j^{-1} & \text{if } (i,j) \in \mathcal{E} \text{ and } i,j \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases}$$

$$d_{ij} = \begin{cases} d_j^{-1} & \text{if } (i,j) \in \mathcal{E} \text{ and } i \in \mathcal{C}, j \in \mathcal{D} \\ 0 & \text{otherwise} \end{cases}$$

and x, y, z are of the row dimension of C, D and 1, and e is the vector of 1's of conforming dimension. Note that the individual equations are:

$$x = \alpha C x + (z/m)e \qquad (5)$$
$$y = \alpha D x \qquad (6)$$
$$z = (1-\alpha)x + e^T y \qquad (7)$$
$$= \{(1-\alpha)e^T + \alpha e^T D\}x \qquad (8)$$

We may exploit this structure to compute x (and z) from a reduced eigen-system:

$$\begin{pmatrix} \hat{x} \\ \hat{z} \end{pmatrix} = \begin{pmatrix} \alpha C & e/m \\ (1-\alpha)e^T + \alpha e^T D & 0 \end{pmatrix} \begin{pmatrix} \hat{x} \\ \hat{z} \end{pmatrix} \qquad (9)$$

since the reduced matrix is column stochastic and

$$\hat{x} = \alpha C \hat{x} + (\hat{z}/m)e \qquad (10)$$
$$\hat{z} = \{(1-\alpha)e^T + \alpha e^T D\}\hat{x} \qquad (11)$$

After solving this system by the normal *Power Iteration*: $x^{(k+1)} = A x^{(k)}$ or similar iterative methods, we may compute

$$\hat{y} = \alpha D \hat{x}. \qquad (12)$$

Comparing the two systems we see that $\hat{x}, \hat{y}, \hat{z}$ are a solution of (4) and may be identified with x, y, z. We may thus solve the reduced eigensystem problem (9) to obtain $x = \hat{x}$, and then compute the ranks of the nodes in $\mathcal{D}$ from (12), in a *single step*. This will lead to significant savings unless $|\mathcal{D}|$ is small.

4.1 Simple examples

In extreme cases the existence of dangling pages could have significant effects on the ranking of non-dangling pages. Consider the simplest example possible in figure 2(a). In this example it makes a crucial difference whether we allow teleportation to dangling nodes. If we were to allow uniform jumps from the dangling node to all nodes (including the dangling node), then the transition matrix would be

$$\begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & 0 & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{3} \end{bmatrix}$$

and there is no need for teleportation because the graph is now strongly connected. In any event we get a higher rank for page 3 than for pages 1 and 2.

Now consider the case where teleportation to dangling nodes is avoided, using the notation of the previous section. Here $\mathcal{C} = \{1, 2\}$ and $\mathcal{D} = \{3\}$. In this case the reduced transition matrix (9) matrix (with $\alpha = .85$) would be:

$$\begin{bmatrix} 0 & 0.425 & \frac{1}{2} \\ 0.425 & 0 & \frac{1}{2} \\ 0.575 & 0.575 & 0 \end{bmatrix}$$

and the corresponding normalized PageRank scores are (x_1, x_2, z) = (0.31746, 0.31746, 0.365079). Hence the vitual page gets a higher score than the other pages, which is intuitively clear since it has two inlinks while the others have only one each. When we back out the rank of the dangling page 3, via (12) it has value 0.269841, lower than 1 or 2, and also less than the rank of the virtual node, since the latter also receives random (teleportation) jumps from 1 and 2.

A slightly more complicated example in Figure 2(b) illustrates that dangling pages can have a higher rank than non-dangling pages, even when teleportation to dangling nodes is forbidden. Solving
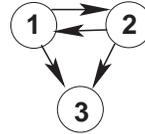
(a) Three page example   (b) Six page example
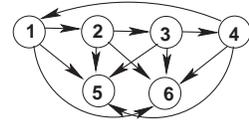
Figure 2: Simple examples with dangling links. The six page example shows that dangling pages can have higher rank than non-dangling pages

the reduced system, and backing out the y values, we obtain normalized ranks (including that of the virtual node) of $(x, y, z) =$ (0.122883, 0.111862, 0.108739, 0.107855, 0.143159, 0.0973207, 0.308181) Note that the virtual page has the highest rank, as we might expect, but that page 5 has a higher rank than any of the connected pages $1, ..., 4$.

5. LINK ROT AND RANKING

Some dangling pages have already been noted to be of extremely high quality, and a page that points to them may be considered to have good hub characteristics as a result of the link relationship. By contrast, pages that are dangling because they produced a 404 or 403 response code should be considered to reflect poorly upon pages that link to them. There are two primary reasons for such a links to exist:

- the page existed at the time that the link was created, but was subsequently removed from the web, causing a broken link. In this case we may consider that the page containing the broken link is no longer being maintained, and is out of date.
- the page never existed, and the original link was created in error. In this case, the document containing the link should be considered to have been poorly authored, as the links were never checked for validity.

In both cases, pages that contain links to pages that with 403 or 404 HTTP return codes should be considered deficient in some way. We refer to pages that return a 403 or 404 HTTP code as *penalty pages*. This reflects a principle that has not previously been captured by PageRank, since PageRank computes a score for a page based on the pages that link to it, rather than based on features of pages that are linked to by the page.

There is anecdotal evidence that as the novelty wears off and the web matures, there is a growing trend toward "link rot", where links that worked at one time are broken by the removal of content on the web or change in the URL. Several studies [26, 31] have forecast the half-life of a URL at between four and five years, and in [26] it was found that more than half the pages being tracked in the .com domain disappeared in 24 months. In our crawl of over a billion pages we discovered that approximately 6% of all web pages that were linked to turned out to return a 404 code. These presumably reflect a fraction of pages that are no longer maintained or were poorly authored in the first place. As time passes we expect that this problem will only worsen as an increasing fraction of pages on the web fall into disrepair.

In our experiments we observed that as a crawl progresses, the percentage of penalty pages tends to increase. This is somewhat similar to the observation of Najork and Wiener [27] that a breadth-first order of crawling tends to find highly ranked pages early on in the crawl. In Figure 3 we show the rate at which penalty pages

were discovered during our crawl of 1.1 billion pages. The rate actually starts out high and then drops quickly, only to rise again as the crawl progresses. The reason for this is that this crawl was seeded with a large set of URLs from a previous crawl, many of which were now 404s. Once these 404s were attempted, the rate dropped and then started to rise again. Since the quality of pages encountered in the crawl tends to decrease, and the probability of having links to penalty pages also increases, there already appears to be a correlation between the two events.
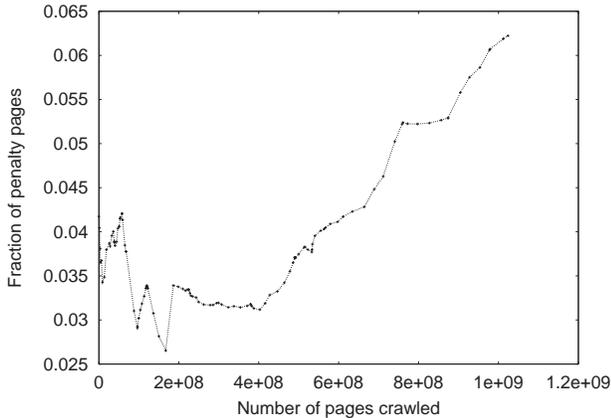

Figure 3: Rate at which penalty pages are encountered during a billion page crawl. The initially high rate was caused by the use of a large seed set from a previous crawl. After an initial dip, the rate climbs during the rest of the crawl.

In section 6 we describe some algorithms that can be used to explicitly discriminate against pages on the basis of whether they point to penalty pages. It is our hope that by incorporating more fine-grained information such as this into ranking, we can improve the quality of individual search results and better manage resources for crawling.

5.1 Another simple example

Let us now consider a case in which penalty nodes are significant. In the example of figure 4, there are four pages, with one of them being a dangling page with a link from page 3. If we compute pagerank with $\alpha = 0.85$, the reduced problem gives us scores of

$$s = [0.198684, 0.283124, 0.283124, 0.235068]$$

for the three strongly connected pages and the virtual page. Now suppose that node 3 has not one but four dangling links. The new ranks are:

$$s = [0.195954, 0.229266, 0.279234, 0.29554]$$

Note that the virtual node's rank has increased (as we might expect), but the rank of node 2 has significantly decreased. Thus dangling links, which may or may not be significant, can have a significant effect on the ranks of nearby pages. We might then consider how this influence is in turn affected by the presence of dead dangling links, such as 404s.

6. LINKS TO PENALTY PAGES

In this section we describe several modifications to the basic PageRank algorithm that can be used to adjust the ranks of pages
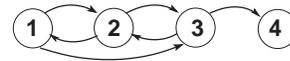
Figure 4: An extreme case for PageRank in which there are three pages that are strongly connected, and one dangling page.

with links to penalty pages. There are four basic methods described here, which we refer to as "push-back", "self-loop", "jump-weighting", and "BHITS".

6.1 The push-back algorithm

The principle behind this algorithm is that if a page has a link to a penalty page, then it should have its rank reduced by a fraction, and the excess rank from this page should be returned to the pages that pushed rank to it in the previous iteration. The end effect is that of limiting the "inflow" of rank to such pages.

To describe the push-back algorithm, let i be a page that contains a link to a penalty page p. Now looking at the equation for x_i above we have

$$x_i^{(k+1)} = \sum_{(j,i) \in \mathcal{E}} a_{ij} x_j^{(k)} \tag{13}$$

and we wish to return a portion (say β_i, where $0 < \beta_i < 1$) of that rank to the pages which point to it (i.e. the j such that $(j, i) \in \mathcal{E}$).

We may do this by modifying the PageRank calculation so that

$$x^{(k+1)} = BAx^{(k)}$$

where B, like A, is a column stochastic matrix. That is $e^T B = e^T A = e^T$ and e is a vector of all ones. This clearly preserves the column stochastic property, since $e^T(BA) = e^T A = e^T$.

We suggest that the penalized rank should be returned to its contributors in the same proportion as it was bestowed in (13). That is the penalized page i should retain a proportion $(1 - \beta_i)$ of its undiluted rank, and the remaining rank should be distributed in proportion $\beta_i a_{ij}$ to the pages which point to it (i.e. the j such that $(j, i) \in \mathcal{E}$). Naturally, these proportions must be normalized so that the total is 1.

In matrix terms this corresponds, in the case of a single penalized page—which we may, without loss of generality assume to be the first page — a B of the following form:

$$B = \begin{pmatrix} (1 - \beta_1)/\sigma & 0 \\ \beta_1 \bar{a}_1/\sigma & I \end{pmatrix} \tag{14}$$

where $\bar{a}_1^T$ is the 1st row of A (except for a_{11}), and

$$\sigma = (1 - \beta_1) + \beta_1 e^T \bar{a}_1$$

is a normalizing factor, so that B is column stochastic.

In the event that several pages are penalized we may extend this procedure in the obvious way to construct a B such that each such page "gives back" the fraction $(1 - \beta_i)$ of its standard rank.

In practice this modification involves an extra step at each iteration of the standard PageRank power iteration – the (sparse) post-multiplication of the rank vector by B.

A particular example of the above procedure might apply to pages which point to 404 (or other bogus) pages. Let g_i be the number of "good" links out of page i and b_i be the number of bad (penalty) links. Then we might penalize page i by setting

$$\beta_i = \frac{b_i}{g_i + b_i}$$

If we modify the example in Fig4 to have eight dangling links from page 3, where four are bad (404s) and four are good, and apply this technique, we obtain the new set of ranks (0.292287, 0.312162, 0.1666, 0.228948) for pages 1–3 and the virtual node. We now see that page 3 has a significantly reduced rank compared to pages 1 and 2.

6.2 The self-loop algorithm

Ordinarily at each step we would follow an outlink with probability α or jump to a random page with probability $1 - \alpha$. In the self-loop algorithm we augment each page with a self-loop link to itself, and with some probability γ_i follow this link (we assume that all self-loops have been removed from the link graph prior to augmentation). The probability γ_i should be smaller if the page has a large number of outlinks to penalty pages. In this way, a page that has no bad outlinks will retain some of its own rank by following a link to itself, whereas a page with only bad links will not retain any of its rank this way. One potential choice for γ_i is to choose a probability γ and use $\gamma_i = \gamma \cdot \frac{g_i}{b_i + g_i}$, where again b_i is the number of outlinks from i to penalty pages, and g_i is the number of outlinks to non-penalty pages. In order to create a stochastic matrix, we adjust the teleportation probability from $1 - \alpha$ to $1 - \alpha - \frac{\gamma b_i}{b_i + g_i}$.

There are several variations on this theme, including a simplified version where we simply add self loops to the page for every good outlink, and select a random outlink from the page (including the added self-loops) with equal probability each time. Alternatively, we could choose a parameter γ_i for each page, and with probability γ_i we follow the self loop, and with probability $1 - \gamma_i$ we follow the standard PageRank process. This would result in transition probabilities of γ_i for the self-loop, probability $(1 - \gamma_i)(1 - \alpha)$ for the teleportation step, and probability $\alpha(1 - \gamma_i)/g$ for following a non-penalty outlink from the page. This results in having no rank for penalty pages, but it can be modified in an obvious way to compute such ranks.

6.3 The jump-weighting approach

We previously treated penalty pages (such as 404s) as dangling nodes in the web graph, and collapsed them into the virtual node along with the legitimate dangling nodes, in equation (4). The standard procedure then redistributes the rank of the virtual node evenly (or to a chosen seed set). We propose an alternative procedure of biasing the redistribution so that penalized pages receive less of this rank. Using the above notation for good and bad pages, a straightforward choice is to weight the link from the virtual node to an unpenalized node in $\mathcal{C}$ (or the seed set) by ρ and to a penalized node by $\rho g_i/(g_i + b_i)$, where ρ is chosen so that the sum of all these edge weights is unity.

6.4 The BHITS algorithm

In this section we describe an algorithm that resembles the HITS[7] algorithm, in the sense that it is derived from a random walk in both the forward and backward directions. In contrast to the HITS algorithm, the BHITS algorithm does not depend on a query, and ranks all pages together. In this way it is more similar to the hub walk of SALSA [25], or to the method describe in [30]. As in the other algorithms, the intent is that pages pointing to penalty pages should be degraded somewhat.

The algorithm is most easily described by the random walk, which consists of two steps, namely a forward step using ordinary PageRank, followed by a "backward" step for dangling nodes. For all non-dangling nodes, the backward step consists only of a self-loop. We distinguish two cases for the backward step from a dangling node. In the case of a penalty page, we forward all of its score to

the virtual node. In the case of a non-penalty page, the backward step would divide the current score of the page by the number of inlinks, and propagate its score equally among all of the backward links. We assume that all pages have an inward link, which is certainly true of pages that we discover by crawling, but we must also assume that the seed pages have known inward links in order to run the algorithm for them. Without loss of generality we could treat any page with no inlinks as a penalty page for this process. For a penalty page, instead of traversing an inlink in the reverse direction, we take a step to a randomly selected seed node as in PageRank. The effect of this is to "return" the rank of pages that point to non-penalty pages, but to redistribute the rank that is given to penalty pages.

If P denotes the matrix representing a PageRank Markov process (as described in section 4, then the matrix describing the BHITS algorithm is simply BP, where B is the matrix that encodes the backwards step. More specifically, order the pages so that the penalty pages are at the end, and for a non-penalty page j, let $\delta(j)$ denote the indegree of a node j. The let $b_{ij} = \frac{1}{\delta(j)}$ be the probability of going from j to i. In this case we get the matrix

$$B = \left[\ b_{ij}\ \middle|\ \frac{\mathbf{1}}{m}\ \right]$$

to describe the backwards step. Here $\mathbf{1}$ denotes a matrix of 1's, but this could easily be replaced with a personalized distribution that favors some pages over others, as can also be done in PageRank. One might for example exclude the penalty pages from the redistribution of weight, which if there are p penalty pages, would produce a matrix as:

$$B = \left[\begin{array}{c} B_{11} \\ B_{21} \end{array}\ \middle|\ \begin{array}{c} \frac{\mathbf{1}}{m-p} \\ 0 \end{array}\right]$$

The matrix BP is stochastic, because it is the product of two stochastic matrices. This Markov chain will produce a unique stationary probability distribution on the pages, but it is evident that for pages that link to penalty nodes, their probability will generally be less than it is in standard PageRank. Note that this algorithm can be cast in a unified framework as was done in [13].

6.5 Implementation Considerations

Considering the size of the web, the feasibility of efficient implementations for any ranking algorithm are essential. For the first three methods, that are based on the PageRank algorithm, we compare the methods we propose for penalizing page scores for pages with broken links to the standard ranking method as described in Section 4. In all of these methods, the modification to the standard calculation is expressed as a modification of the matrix A. Looking at the particulars of the three methods, one can immediately notice that those modifications are local in nature. i.e., apart for the requirement to keep the matrix normalized, the changes to an entry in the ith row of the matrix are only dependent on the number of penalty and "good" outlinks from the ith node, b_i and g_i. This allows either the modified matrix to be pre-computed in linear time (given the vectors b and g), or the required modifications to be executed on the fly along with the power iteration computation. We therefore conclude that these three methods require minimal computational overhead.

The BHITS algorithm is slightly more involved to implement. The main complication, however, is independent from the modification required to penalize pages with broken links. Rather than allowing the collapse of all dangling nodes into a virtual node, as described in Section 4, the forward/backward approach requires an eigen-system that actually includes all of the dangling nodes. In our sample crawl we had roughly one billion crawled pages (some of

them dangling), but nearly five billion total URLs discovered. This means that a system that includes dangling nodes is more than five times bigger than that required for a "forward-only" process based on PageRank. Therefore the BHITS algorithm is not as suitable for large scale implementations as the others.

7. THE HOSTRANK ALGORITHM

Up until this point we have concentrated on the issue of dangling pages, but there are numerous other structural features of the web that can be incorporated into PageRank. The original paradigm was to model the web as a set of pages that readers can navigate through, occasionally jumping to a random page as they lost interest or decided to investigate a different topic. This basic paradigm makes many simplifying assumptions, and in particular one might object that it is impossible for a user to choose a page uniformly at random (or even to know what URLs there are to choose from). There have been several recent studies suggesting that a large percentage of web browsing sessions start by a visit to a search engine, expressing a query for their need, and following links suggested by the search engine. The duration of the session will usually then consist of following links from the search results for a while, possibly returning to the list of results or reformulating the query, until the information need is satisfied or the user gives up. When the next session starts, users don't simply jump to a random page, but instead return to the search engine and formulate a new query to describe their need. This behavior is quite different from the model of the PageRank paradigm, and in particular when a user takes a random jump, they are presented with a set of possible choices that are profoundly nonuniform in their distribution.

Another variation on the model for a web surfer that was suggested in PageRank involved having the user periodically jump to a random page selected from some smaller set of "trusted pages". This has further been refined as a means of producing a personalized or topic-sensitive page rank [21, 28]. This would be a good model for a user who jumps to a URL selected from a set of bookmarks, or perhaps to a portal. On the other hand, portals now tend to change their content rapidly in order to encourage return visits from users, so once again the random jumps follow a nonuniform distribution. Moreover, each time the user visits the portal the links are likely to be to content that is *new*, and did not even exist during their prior visit. Hence the jump distribution changes over time, and is even directed to the frontier.

In [29] it was observed that the probability values produced by the PageRank algorithm decay according to a power law, and they incorporate this into a model of how the web evolves. They assumed that teleportation takes place to a page chosen uniformly at random. In Figure 5 we show the distribution of pagerank values computed using teleportation to a randomly selected page vs. teleportation with probability 0.5 to one of two highly ranked sites (www.microsoft.com and www.yahoo.com). The distribution of the pagerank values is quite different, and while our observations confirm the results of [29], it points out that the teleportation strategy has a profound effect on the hypothesized probability distribution for a user to end up at a page. In particular, if the link structure of the web was hierarchical, in which every page at a given level linked to every page below it, then the distribution would decay exponentially as we descended the hierarchy, and the tail of the distribution shows something closer to this than a power law distribution. In passing we note that it is perhaps tempting to assign advertising value to pages on the basis of a PageRank value, since it was intended to model the probability that a web surfer will end up at the page. These computations at least show that it makes a critical difference how one computes PageRank.
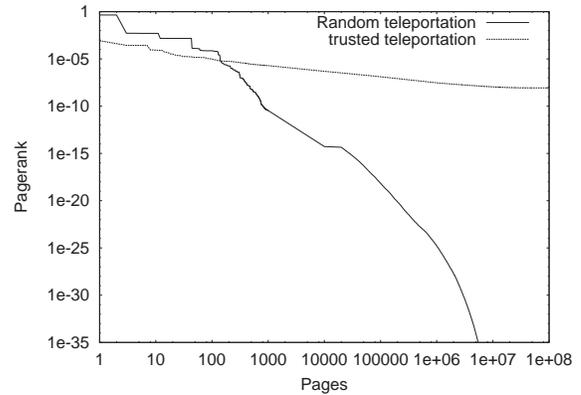

Figure 5: Distribution of pagerank values in the case where teleportation is done to a randomly selected site or one of two trusted seed sites.

7.1 Exploiting hierarchical structure

One suggestion that was made in the original PageRank paper [28] was to jump to a randomly selected top-level page of a site. In fact this probably models very well how people enter a site, since there is a strong human tendency for information to be organized hierarchically [15]. Moreover, from our crawl of over a billion pages we found that 62.4% of all links were internal to a site, and that links within a site tend to show a high degree of locality [15]. We have also found that when links are external to a site, they tend to link to the top level of the site (see Figure 6). These structural features of

Depth	fraction
0	0.705
1	0.089
2	0.035
3	0.017
4	0.005
5	0.002
6	0.001

Figure 6: Links from outside a site tend to link to the top level of a site. The depth here is the number of levels of hierarchy below the top for the destination URL. Approximately 14% were to dynamic URLs, and if we ignore those then the top level of sites receives 82% of all external links.

the web have also shown up in other ways, allowing very high levels of compression for the link graph and enabling a block-oriented approach to accelerate the convergence of PageRank [23].

The hierarchical structure of information on a site and the fact that most external links are to *sites* rather than individual pages suggests that we consider all pages on a site as a single body of information, and assign them all a rank based on the collective value of information on that site. This suggestion has been made in the past (going back at least as far as [3]), and is sometimes referred to as computing a *hostrank* rather than a PageRank. Formally, we can think of the hostnames as representing nodes in a graph, and construct a directed edge from hostname S to D if there is a URL on S that has a link to a URL on D. It is further possible to assign weights to the edges between hosts, where the weight reflects the number of links from URLs on the source to URLs on the destination. It is then natural to scale these edge weights so that the total weight emanating from a hostname sums to 1. These weights

then replace the entries $a_{ij} = 1/d_j$ in the original PageRank calculation. The inclusion of weights on the edges means that we require an additional piece of information for each edge of the graph, but the weights can improve the quality of results by reflecting the strength of the connection. As an alternative we might use weights consisting of the number of distinct destination URLs.

The computation of hostrank is much simpler than that of computing PageRank, because the graph is much smaller. In our data set there were approximately 48 million hostnames, although for various reasons, many of the hostnames are uncrawlable. Note that this approach to reducing the amount of computation for PageRank can be combined with a number of other optimization approaches, including more sophisticated linear algebra [3, 23, 22, 24] better I/O management [20, 8], and approximations that use only local information [1].

8. THE DIRRANK ALGORITHM

One might argue that lumping all pages on a host into a single entity is too coarse a level of granularity. There are certainly examples such as ISPs, and notably `www.geocities.com`, for which it is implausible to consider all pages on the same host as having equivalent rank. Moreover, the distribution of the number of URLs per host has been observed to have a heavy tail distribution. Thus while there are some hosts with millions (perhaps billions or trillions?) of possible URLs on them, most hosts have relatively few URLs. The hostrank algorithm would have difficulty in distinguishing between these.

It has previously been observed that web information tends to have a hierarchical information reflected in URLs [15, 16]. This is due in part to the fact that many web servers simply export a file system, and it has become a common practice for humans to group related files together in a single directory, and for administrative delegation of authorship to be done at the directory level. This hierarchical structure is even present on servers whose content is not stored in a hierarchical file system, as the URL standard was originally designed to incorporate hierarchical structure when it is natural to do so [4].

In [16] it was further observed that URLs can often be grouped into "compound documents" that represent a single unit of information, and that such compound documents tend to consist of URLs that agree up to the last '/' character. Thus it is perhaps natural to group together URLs that agree up to the last delimiter as a single information node, and construct a *DirRank* graph. The nodes in this case correspond to URL prefixes up to the last '/' character (or other delimiter), and there is an edge from one node to another if there is a link from a URL in the source virtual directory to a URL in the destination virtual directory. In effect this groups URLs at a finer level of granularity than entire hostnames, but still often conforms to a human-designed hierarchical organization of information.

So-called "dynamic" URLs containing a '?' character tend not to follow this hierarchical organization of information, but such URLs are often an indicator for the existence of an underlying database capable of serving an enormous number of URLs, and these are often still closely related to each other and it is natural to group them. Thus we can extend the hierarchical grouping to include all URLs that agree up to the last '/' or the last '?', whichever occurs first.

As we mentioned in the previous section, the distribution of the number of URLs per hostname tends to have a heavy tail distribution, and the same is true for the number of URLs in a virtual directory [15]. In our crawl of a billion URLs, over 79% contained five or fewer URLs, and over 87% contained fewer than ten URLs. Whenever a directory contains a large number of URLs, it tends to

be either the archive of a mailing list or else a database-driven set of dynamic URLs.

It should be noted that the URL hierarchy extends into the hostname (although the order is reversed at this point). Thus we might also consider grouping all URLs that agree up to some point in the complete hierarchy, thus assembling some multiple hostnames into a single node in cases where it makes sense to do so. In particular, we discovered second level domains that had more than a million hostnames in them.

9. EXPERIMENTAL RESULTS

Comparison of ranking methods for web pages is complicated by the fact that there is no universally recognized measure of quality for a static ranking. Moreover, as we mentioned before, PageRank is a paradigm rather than a well defined algorithm, so it is not clear what to compare with. In reality, search engines use a large number of factors to rank results relative to a query, a user, and the context in which the search is performed. The problem of producing a single linear static ordering on web pages is an over-simplification of the search ranking problem, though it has proved to be an extremely important tool for the original problem.

In spite of the fact that we have no reasonable methodology to quantitatively measure the quality of the rankings, we can still compare the different ranking schemes on a number of factors, including the computational resources required, the similarity between different ranking methods, and subjective judgments about highly ranked pages.

In order to make some comparisons, we calculated several variations of PageRank of the graph induced by the first 100 million pages from our billion page crawl. We also calculated a standard PageRank for 1.08 billion pages using teleportation to pages selected uniformly at random. Finally we computed DirRank and hostrank on our billion-page crawl.

For the DirRank calculation, once we collapsed the URLs into directories, we were left with approximately 114 million nodes representing the directories, and 15 billion edges representing links between pages in different directories. Thus we were able to reduce the number of nodes in the graph by nearly an order of magnitude, but the number of edges dropped only by a small factor. It is natural to expect that the ranking induced by computing DirRank on this graph will interpolate between the results obtained with hostrank and the results using PageRank. The major differences are that it breaks URLs into more fine-grained detail than hostrank, but still saves an order of magnitude on the the number of nodes in the PageRank graph. By contrast, the hostlink graph turned out to only involve 19.7 million hosts, with 1.1 billion edges between them. This represents a savings of nearly another order of magnitude in the number of nodes over the DirRank graph.

In calculating PageRank on the billion pages, we discovered very early on that single precision calculations were inadequate without careful attention to ordering of operations or some other method of controlling errors. We therefore had to compute these in double precision, though we did not encounter this requirement when computing PageRank on only a hundred million pages. We chose to implement a fairly naive method using simple power iteration, although we could have employed more sophisticated implementations (e.g.,[20]).

9.1 Subjective judgments and link spamming

Web search has come to be recognized as the primary control point for electronic commerce on the web, and as a consequence, high ranking in search engines is now perceived to have high value. This has given rise to an industry that assists merchants and adver-

tisers in manipulating the ranking of web pages in search engines, and has increased the need for ranking methodologies that are resistant to manipulation. This turned out to be a major consideration in evaluating the results for different algorithms.

The competition for high ranking in search engines has made PageRank a direct target for manipulation, and this was apparent in our experiments. Among the top 20 URLs in our 100 million page PageRank calculation using teleportation to random pages, 11 were pornographic, and they appear to have all been achieved using the same form of link manipulation. The specific technique that was used was to create many URLs that all link to a single page, thereby accumulating the PageRank that every page receives from random teleportation, and concentrating it into a single page of interest. This can be cascaded through trees of increasing concentration, using only moderate indegree for the intermediate nodes in the tree. All that is required is to get as many pages crawled as possible.

By contrast, the PageRank that used teleportation to only two trusted sites had no pornography in the top 1000 pages. There has been relatively little published on this problem of rank manipulation, although there has been some anecdotal suggestions to assign greater weight to non-nepotistic links, and some work has been published on recognizing them [12]. Our subjective evaluation suggests that teleportation to trusted sites can help considerably.

We also computed hostrank using a weighted graph and teleportation to random nodes. After reviewing the top 100 hostnames in this ranking, we found only 14 to be questionable, including two porn sites, six sites offering link exchange services, and five fake companies involved in link spamming. Though there was still evidence of manipulation in the host rankings, we judged them to be less affected. The manner in which manipulation took place was similar to that used in manipulating PageRank, and in fact we found that the second-level .com domain with the most hostnames was a pornographic content provider for which we had discovered 1,442,013 distinct hostnames. There is no apparent reason to maintain such an unusual DNS server other than to manipulate hostrank. The technique of aggregation in DirRank can be carried over into the domain name space in order to combat this type of rank manipulation.

9.2 Comparison of rankings

Given two ranking methods, we can measure how different they are from each other using a measure such as Kendall or Spearman distance, or measures from [5]. When we began this study, we hypothesized that hostrank and DirRank would be good approximations for PageRank, and because they were so much cheaper to compute, they would be preferable. In fact every form of measurement that we applied showed them to result in very different rankings. This is not in itself a bad thing, because our observations on PageRank with uniform teleportation show that it is very subject to manipulation and does not provide a particularly good ranking any more. The computational advantages of hostrank and DirRank, and their enhanced resistance to rank manipulation are probably enough to make them attractive candidates for ranking.

One aspect in which the hostrank calculation seems to differ in from PageRank is the effect of adversarial link manipulation. We believe much of this difference can be explained by examining the distribution of rank in the virtual node in both cases as was described in Section 4. Teleportation is also formulated by jumping first to this virtual node. This virtual node distributes its score uniformly at random to all other nodes. It turns out that because of the size of the frontier of the web, the score that the virtual node receives is rather significant. In fact, in our 100 million page Page-

Rank calculation, the weight of the virtual node was .82. We believe much of the link manipulation that is used in boosting the page rank of certain sites, involves using a large collection of pages that all link to the page whose rank is to be boosted. This collection of pages receives a non-negligible fraction of the probability of the virtual node (which is distributed uniformly among all pages) and passes it on to just one page. Hostrank cancels out the effect of this construction because it transfers probability from the virtual node to a host independent of the number of pages in that host. Furthermore, because the host graph is smaller, and its frontier is smaller (since it is better connected), the virtual node receives a significantly lower probability itself. In our experiments, the virtual node probability for hostrank was a relatively low .17.

9.3 The effects of dangling nodes

Many major newspaper sites prohibit crawling of most of the pages on their sites, thus turning all pages on those sites into dangling nodes. Often home pages of a site will accumulate rank by aggregating the PageRank score flowing into individual pages on the site to a top level homepage through a site's internal navigational link structure. For a site that is not crawlable, this structure does not exist, and the internal aggregation of score cannot take place. This has a very noticeable effect on the rank of the home page. Among major newspapers, the Chicago Tribune was the highest ranking in our one billion node PageRank calculation (its home page ranked 87th). By contrast, newspapers that are both considered more authoritative, and have much higher circulation such as the Wall Street Journal or the New York Times ranked much lower because their sites are mostly blocked from crawling. In fact, of the 55 newspapers that we examined, we found 24 newspapers with higher rank than www.nytimes.com under PageRank with uniform teleportation. In all, major newspapers whose sites are crawlable clearly dominated major newspapers whose sites cannot be crawled, which illustrates another effect from dangling nodes.

10. CONCLUSION

The original suggestion for PageRank was a heuristic based on a *random surfer* model to evaluate the probability that a page would be viewed. In this work we have found significant evidence that PageRank with uniform teleportation has been targeted for manipulation, and the results are now less than satisfactory. The original paper on PageRank had multiple variations that overcome this problem, and we have found evidence that these variations are either helpful or else have not come under direct attack.

We have suggested the hostrank and DirRank methods as improvements to the basic PageRank algorithm, and our results show that they are both cheaper to compute and currently result in rankings that display less effect from manipulation. Whether these techniques will continue to resist rank manipulation is open, but we believe that aggregation is a useful technique for tuning rank algorithms, and is well motivated from the hierarchical structure that is evident in the web. There remain many opportunities for research in new ranking methods that are better tuned to models of user behavior and interests.

The problem of dealing with "dangling" pages on the frontier of the crawled web has been largely ignored in previous discussion of PageRank. We have provided several different alternatives for rigorously and efficiently ranking these pages. Moreover, we have distinguished between different types of dangling links, and proposed four techniques for penalizing pages which point to illegitimate pages, thus reducing the bias afforded to the ranking process by these bad links.

Up until this time the analysis of ranking algorithms for the web

has remained largely in the realm of trade secrets and economic competition. It is our hope that this paper will elevate the discussion to a more scientific one, and that future notions of fairness and balance will emerge to place the problem of web page ranking on a firmer scientific foundation.

11. REFERENCES

[1] Serge Abiteboul, Mihai Preda, and Grgory Cobena. Adaptive on-line page importance computation. In *Proc. 12th World Wide Web Conference*, pages 280–290, 2003.

[2] Gianni Amati, Iadh Ounis, and Vassilis Plachouras. The dynamic absorbing model for the web. Technical Report TR-2003-137, University of Glasgow, April 2003.

[3] Arvind Arasu, Jasmine Novak, Andrew S. Tomkins, and John A. Tomlin. Pagerank computation and the structure of the web: Experiments and algorithms. In *Poster Proc. WWW2002*, Honolulu, 2002.

[4] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifiers (URI): Generic syntax. http://www.ietf.org/rfc/rfc2396.txt. RFC 2396.

[5] Allan Borodin, Gareth O. Robers, Jeffrey S. Rosenthal, and Panayiotis Tsaparas. Finding authorities and hubs from link structures on the world wide web. In *Proc. 10th World Wide Web Conference*, pages 415–429, 2001.

[6] Sergey Brin, Rajeev Motwani, Lawrence Page, and Terry Winograd. What can you do with a web in your pocket? *Data Engineering Bulletin*, 21:37–47, 1998.

[7] Soumen Chakrabarti, Byron Dom, David Gibson, Jon M. Kleinberg, Prabhakar Raghavan, and Sridhar Rajagopalan. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proc. 7th World Wide Web Conference*, pages 65–74, 1997.

[8] Yen-Yu Chen, Qingqing Gan, and Torsten Suel. I/O efficient techniques for computing pagerank. In *CIKM 2002*, pages 549–557, McLean, Virgina, 2002.

[9] Steve Chien, Cynthia Dwork, Ravi Kumar, and D. Sivakumar. Towards exploiting link evolution. In *Workshop on Algorithms and Models for the Web Graph*, 2001.

[10] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. Efficient crawling through url ordering. In *Proc. of 7th World Wide Web Conference*, 1998.

[11] Nick Craswell, David Hawking, and Stephen E. Robertson. Effective site finding using link anchor information. In *Proc. of the 24th annual international ACM SIGIR conference on research and development in information retrieval*, pages 250–257, New Orleans, Louisiana, USA, September 2001. Association for Computing Machinery.

[12] Brian D. Davison. Recognizing nepotistic links on the web. In *Artificial Intelligence for Web Search*, pages 23–28. AAAI Press, July 2000.

[13] Chris Ding, Xiaofeng He, Parry Husbands, Hongyuan Zha, and Horst Simon. Pagerank, hits and a unified framework for link analysis. In *Proc. of 25th ACM SIGIR*, pages 353–354, Tampere, Finland, 2002.

[14] Nadav Eiron and Kevin S. McCurley. Analysis of anchor text for web search. In *Proc. of 26th ACM SIGIR*, pages 459–460, 2003.

[15] Nadav Eiron and Kevin S. McCurley. Locality, hierarchy, and bidirectionality in the web. In *Workshop on Algorithms and Models for the Web Graph*, Budapest, May 2003.

[16] Nadav Eiron and Kevin S. McCurley. Untangling compound documents on the web. In *Proc. 14th ACM Conf. on Hypertext*, pages 85–94, 2003.

[17] Ronald Fagin, Ravi Kumar, Kevin S. McCurley, Jasmine Novak, D. Sivakumar, John A. Tomlin, and David P. Williamson. Searching the workplace web. In *Proc. 12th World Wide Web Conference*, Budapest, 2003.

[18] Gene H. Golub and Charles van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 3rd edition, 1996.

[19] Siegfried Handschuh, Steffen Staab, and Raphael Volz. On deep annotation. In *Proc. 12th World Wide Web Conference*, pages 431–438, Budapest, 2003.

[20] Taher Haveliwala. Efficient computation of pagerank. Technical report, Stanford University, 1999.

[21] Taher H. Haveliwala. Topic-sensitive pagerank. In *Proc. 11th World Wide Web Conference*, pages 517–526, Honolulu, 2002.

[22] Sepandar Kamvar, Taher Haveliwala, and Gene Golub. Adaptive methods for the computation of pagerank. Technical report, April 2003.

[23] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub. Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University, 2003.

[24] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub. Extrapolation methods for accelerating pagerank computations. In *Proc. 12th World Wide Web Conference*, 2003.

[25] Ronny Lempel and Shlomo Moran. SALSA:the stochastic approach for link-structure analysis. *ACM Transactions on Information Systems*, 19(2):131–160, 2001.

[26] John Markwell and David W. Brooks. Link rot limits the usefulness of web-based educational materials in biochemistry and molecular biology. *Biochem. Mol. Biol. Educ.*, 31:69–72, 2003.

[27] Marc Najork and Janet L. Wiener. Breadth-first search crawling yields high-quality pages. In *Proc. 10th World Wide Web Conference*, pages 114–118, 2001.

[28] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998. Paper SIDL-WP-1999-0120 (version of 11/11/1999).

[29] Gopal Pandurangan, Prabhakar Raghavan, and Eli Upfal. Using pagerank to characterize web structure. In *COCOON 2002*, pages 330–339, Singapore, 2002. Springer-Verlag. LNCS 2387.

[30] Davood Rafiei and Alberto Mendelzon. What is this page known for? computing web page reputations. In *Proc. 9th World Wide Web Conference*, Amsterdam, 2000.

[31] Diomidis Spinellis. The decay and failures of web references. *Comm. ACM*, 46(1):71–77, 2003.

[32] John A. Tomlin. A new paradigm for ranking pages on the world wide web. In *Proc. 12th World Wide Web Conference*, pages 350–355, Budapest, May 2003.