

# Local Methods for Estimating PageRank Values

Yen-Yu Chen      Qingqing Gan      Torsten Suel

CIS Department  
Polytechnic University  
Brooklyn, NY 11201

{yenyu, qq-gan}@photon.poly.edu, suel@poly.edu

## Categories and Subject Descriptors

H.3.5 [Information Storage and Retrieval]: Online Information Services

## General Terms

Algorithms, Performance

## Keywords

Pagerank, search engines, out-of-core, external memory algorithms, link-based ranking, link database

## ABSTRACT

The Google search engine uses a method called PageRank, together with term-based and other ranking techniques, to order search results returned to the user. PageRank uses link analysis to assign a global importance score to each web page. The PageRank scores of all the pages are usually determined off-line in a large-scale computation on the entire hyperlink graph of the web, and several recent studies have focused on improving the efficiency of this computation, which may require multiple hours on a workstation.

However, in some scenarios, such as online analysis of link evolution and mining of large web archives such as the Internet Archive, it may be desirable to quickly approximate or update the PageRanks of individual nodes without performing a large-scale computation on the entire graph. We address this problem by studying several methods for efficiently estimating the PageRank score of a particular web page using only a small subgraph of the entire web. In our model, we assume that the graph is accessible remotely via a link database (such as the AltaVista Connectivity Server) or is stored in a relational database that performs lookups on disks to retrieve node and connectivity information. We show that a reasonable estimate of the PageRank value of a node is possible in most cases by retrieving only a moderate number of nodes in the local neighborhood of the node.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM'04, November 8–13, 2004, Washington, D.C., USA.

Copyright 2004 ACM 1-58113-874-1/04/0011 ...\$5.00.

## 1. INTRODUCTION

As the World Wide Web has grown from a few thousand pages ten years ago to several billion pages at present, traditional term-based ranking techniques have become increasingly insufficient at returning good results. For this reason, a large number of studies have proposed new ranking techniques based on the analysis of the hyperlink structure of the Web. Probably the two best-known examples of link-based ranking methods are the HITS [18] and PageRank [22] algorithms. The PageRank algorithm, in particular, is used in the highly successful Google search engine.

PageRank is based on the idea of assigning a global importance score, called *PageRank value*, to each page on the web based on the number of hyperlinks pointing to the page and the importance of the pages containing those hyperlinks. PageRank values are usually computed in an off-line manner, during a large-scale iterative computation on the entire web graph. Several recent studies [3, 10, 13, 17] have looked at ways to optimize this computation, which may take many hours on a typical workstation for large graphs.

However, there are some situations in which a global computation on the entire graph is impractical, e.g., if the link information of the whole web graph is not easily accessible and we need a quick estimation for a particular web page. In particular, users may have only access to a limited subset of the web, or may have to access the graph via a remote connectivity server [5] such as the `link: query` facility in AltaVista, or via a similar interface to a local meta data repository in a disk-based relational database. For example, we might be interested in the evolution of the PageRank of a particular page tracked by the Internet Archive, but cannot afford multiple PageRank computations on billions of pages (We note that the Internet Archive does currently not yet provide the remote Connectivity Server functionality that would be needed in this case).

In this paper, we study local methods for determining reasonable estimates of the PageRank values of individual nodes. The basic approach in all our methods is to expand a small subgraph around the target node and to use this subgraph as the basis for our estimation. Our goal is to minimize the cost of our methods, as measured in terms of the size of the subgraph that is retrieved, while maximizing the accuracy of our estimation. We show that on average, a reasonable estimate of the PageRank of a node can be obtained by visiting a few dozen to a few hundred nodes. We compare several methods, and in the process also discuss the correlation between PageRank and simple statistical measures such as weighted and unweighted in-degrees. Our approach can

also be used to estimate current PageRank values based on slightly outdated values from an old graph, but under different assumptions than the work in [11], and we evaluate this approach as well. Our results assume that the graph is accessible remotely via a link database (such as the AltaVista Connectivity Server) or is stored in a relational database that performs lookups on disks to retrieve node and connectivity information, and thus the cost of the methods under this model is proportional to the number of nodes that are visited.

The remainder of the paper is organized as follows. We first provide a brief review of the PageRank method. Section 3 is the main section of this paper, and describes our local methods for estimating PageRank values and evaluates their performance on a large subgraph of the web. Section 4 discusses related work, and finally Section 5 provides some concluding remarks.

## 2. REVIEW OF PAGERANK

In this section, we review the PageRank technique. Recall that the Web can be viewed as a directed graph whose nodes are web pages and whose edges are the hyperlinks between pages. The *in-degree* of a node is the number of edges (hyperlinks) pointing to it, and the *out-degree* of a node is the number of distinct hyperlinks out of it. We use  $N$  to denote the number of nodes in the graph.

**The Basic Idea:** PageRank was proposed in [7, 22] as a ranking mechanism for the Google search engine that assigns to each page a global importance score based on link analysis. The basic idea is that if page  $u$  has a link to page  $v$ , then the author of  $u$  is implicitly saying that page  $v$  is somehow important to  $u$ . Thus, page  $u$  is conferring some amount of importance onto  $v$ , and this amount is determined by the importance of  $u$  itself and the number of outlinks in  $u$  over which it is divided. This recursive definition of importance can be described by the stationary distribution of a simple random walk over the graph, where we start at an arbitrary node and in each step choose a random outgoing edge from the current node. (Several papers [8, 6, 24] have proposed heuristics for assigning different weights to the outgoing edges in a node; we assume each edge is selected with equal probability though our techniques could be adjusted to other choices.) According to this random walk model, the importance  $r(p)$  of a page  $p$  is determined as:

$$r(p) = \sum_{q \rightarrow p} \frac{r(q)}{d(q)}, \quad (1)$$

where  $d(p)$  is the out-degree of page  $p$ . This can be rewritten in matrix form as follows. Let the nodes be labeled from 0 to  $n - 1$ , and define matrix  $L$  by  $L[i, j] = 1/d(i)$  if there is a link from node  $i$  to node  $j$  and 0 otherwise. (Thus,  $L$  is a "degree-scaled" version of the adjacency matrix of the graph.) Then we obtain

$$\vec{r} = L \cdot \vec{r},$$

where  $\vec{r}$  is the vector of rank values over all pages. This leads to the following iterative computation for approximating the rank vector  $r$  over all of the pages on the web. First, initialize  $r^{(0)}(p)$  to  $1/N$  for all pages  $p$ . Then, in each iteration, update the rank vector using

$$r^{(i)}(p) = \sum_{q \rightarrow p} \frac{r^{(i-1)}(q)}{d(q)}, \quad (2)$$

for  $i = 1, 2, \dots, n$ , or in matrix form  $\vec{r}^{(i)} = L \cdot \vec{r}^{(i-1)}$ . We continue the iterations until the rank vector stabilizes to within some threshold. The final vector is then used to influence the ranking of the search results returned by Google, though details on this aspect are not publicly disclosed.

**Some Technical Issues:** The above formulation assumes that each node has at least one outgoing edge. However, this is not true for web crawls or even the entire web, as there are many pages with out-degree zero (called *leaks*). Such pages would result in a loss of total rank value from the system. One solution is to prune the graph, by iteratively removing any leak nodes in  $G$ .

Even after complete pruning, the graph is still not strongly connected. There will be many smaller and even a few larger groups of pages that form separate strongly connected components, maybe with links entering the component but no links leaving it, that can act as rank sinks that "trap" large amounts of rank value. This can be dealt with by adding a dampening factor  $\alpha$ ,  $0 < \alpha < 1$ , to the random walk as follows: in each step, we choose a random outgoing link with probability  $\alpha$ , and jump to a random node in the graph with probability  $1 - \alpha$ . Thus we have

$$r^{(i)}(p) = \frac{1 - \alpha}{N} + \alpha \cdot \sum_{q \rightarrow p} \frac{r^{(i-1)}(q)}{d(q)}. \quad (3)$$

In the matrix formulation, we define a new matrix  $L'$  with  $L'[i, j] = \alpha/d(i)$  if there is a link from node  $i$  to node  $j$  and  $(1 - \alpha)/N$  otherwise, and then use  $L'$  instead of  $L$ . In our experiments, we use  $\alpha = 0.85$ .

**Weighted In-Degree:** One simplistic explanation of PageRank would be of the form "very similar to in-degree, except it matters where the pointers come from". This would suggest that the PageRank value of a node might be strongly correlated with its in-degree (which had been previously proposed as a ranking function [19]). It was observed in [23] that this is in fact not the case. One obvious reason is that the amount of PageRank sent across an edge depends on the out-degree of the sending node. This motivates us to define the *weighted in-degree* of a node  $p$  as:

$$wd(p) = \sum_{q \rightarrow p} \frac{1}{d(q)},$$

where  $d(q)$  is the outdegree of page  $q$ . A natural question is to what degree PageRank is related to weighted in-degree, and we will run into this question later on.

## 3. ALGORITHMS AND EXPERIMENTAL RESULTS

We start by describing the main idea underlying the methods that we propose. After introducing our experimental setup, we then describe and evaluate our methods one after the other.

### 3.1 The General Idea

In our problem setup, we assume that we have a *fetch* operation that allows us, given a URL or page ID, to retrieve the in-degree and out-degree of the corresponding page plus the sources and destinations of incoming and outgoing links, respectively. Our cost measure is the number of fetch operations performed. (In the case where there are many incoming links, our best methods actually do not require informa-

tion about all sources of links.) All our methods follow the same simple approach based on three phases:

- (1) **Expansion:** We build a subgraph starting from the *target node* for which we are interested in estimating the PageRank value, by expanding backwards from the target node following reverse hyperlinks. We stop this expansion phase after a while based on some criterion. The cost of this phase is equal to the size of the subgraph we build. An example of a subgraph is shown in Figure 1, with a target node on the right, three *internal nodes*, and five *boundary nodes* on the left side.
- (2) **Estimation:** We use a heuristic to estimate the PageRank of each boundary node, or the PageRank flowing into the boundary node from the rest of the graph. The simplest heuristic is to estimate it as the average PageRank value in the graph,  $1/N$ . This phase does not involve any fetch operations.
- (3) **Iteration:** We run the standard iterative algorithm for PageRank on the subgraph, in each step putting our estimated value into the boundary nodes, adding the random jump value of 0.15 to the internal nodes, and removing any flow leaving the subgraph. After some iterations, we use the PageRank in the target node as our estimate. This phase also does not involve any fetch operations.

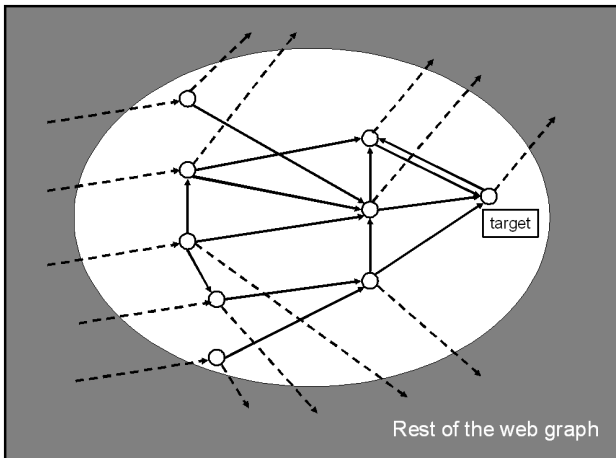


Figure 1: An expanded subgraph in our approach.

Figure 1 shows a subgraph with target node, internal nodes, and boundary nodes. Similar to [11], we can look at the rest of the graph as being abstracted into one supernode. There are three types of edges in our graph, *incoming edges* that come from outside the subgraph into a boundary node, *internal edges* within the subgraph, and *outgoing edges* leaving the subgraph. In addition, there is a known amount of value coming into each node via the random jump. The estimation error of our method will depend on how accurately we estimate the PageRank values of the boundary nodes, which itself depends on how much rank value enters over the incoming edges, and on the size of the subgraph since we hope that errors will largely cancel out given a sufficient number of boundary nodes.

### 3.2 Experimental Setup

Our algorithms were coded in Perl, with some subroutines in C for CPU-intensive computations. We used two web graphs of about 120 million pages crawled by us during October 2002 and May 2001. (The old graph was only used in one of the experiments.) Following Haveliwala [13], we pruned the graph twice by removing leak nodes; this resulted in a highly connected graph with 51,181,520 nodes and 783,407,543 links, for an average out-degree of 14.5 in the new graph. We then stored the link information for the graph in a Berkeley DB database which required about 20 GB storage space. This provides an adjacency-list type interface to the graph similar to connectivity servers [5] and search engines with support for `link`: queries, or relational web meta data repositories, with each *fetch* operation requiring a disk access. To measure the accuracy of our estimations, we also performed the global PageRank computation on the entire graph. We report our experimental results for a set  $S$  of 100 randomly chosen target nodes.

Two measurements are used to identify the accuracy of our methods. One is *relative error*  $e(t)$  of a target node  $t$ , defined as  $e(t) = |r'(t) - r(t)|/r(t)$ , where  $r'(t)$  and  $r(t)$  are the estimated and precise PageRank values, respectively. The other one is called *precision*, defined as  $p(t) = r'(t)/r(t)$ . Thus, precision tells us whether we tend to over- or underestimate the correct result. We typically plot our results in terms of the average relative error and standard deviation, and also show scatter plots for precision.

### 3.3 A Naive Method

In this method, we build the subgraph by simply expanding from the target node backwards for a fixed number of levels  $k$ , i.e., we include all nodes from which the target can be reached in at most  $k$  steps. We then estimate the PageRank of each boundary node as  $1/N$ , the average value in the graph. The results are shown in Figure 2, which shows the average error and cost per target node. Figure 3 gives a scatter plot of the precision and cost values for 6 levels. As we see, we eventually get average relative errors slightly below 10%, though at a high cost. For example, with three levels, we obtain an average relative error of 10.07% at an average cost of 2520 fetch operations per estimation.

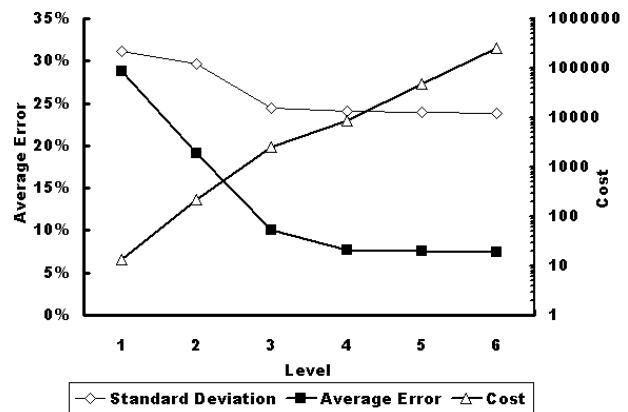


Figure 2: Error versus cost for different numbers of levels in the naive method.

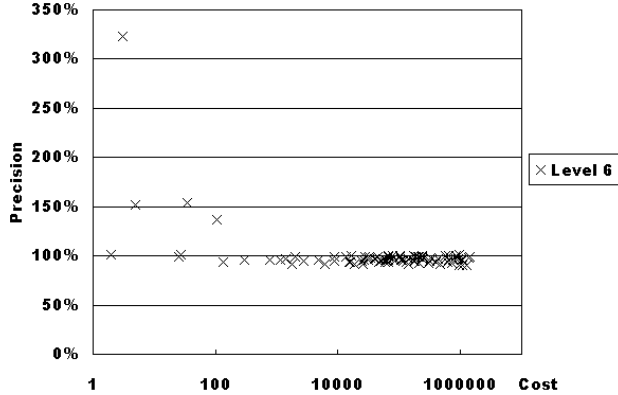


Figure 3: Precision versus cost for six levels in the naive method.

We note that the size of the subgraph explodes to about 250,000 nodes after six levels, but even at that size errors are 7.5%. In the following, we look at smarter techniques that do not expand the graph in a brute-force manner.

### 3.4 The Simple Influence Method

Considering the boundary nodes of the subgraph, we can ask about the impact that each boundary node has on the estimation at the target node. It should be obvious that for some nodes, the impact is low since most of the PageRank value entering the node will never arrive at the target node, while others will have more impact since most paths from this node go to the target node. We formally define the *influence*  $I_p$  of a node  $p$  as the fraction of PageRank value at this node that will eventually arrive at the target node without performing a random jump in between. We note that influence is related to the *inverse distance* measure defined by Jeh and Widom in [16]. The overall impact that a node has on the estimation error at the target depends on its influence and the absolute PageRank estimation error at the node (which we do not know). Formally, the absolute estimation error at the target is upper-bounded by the sum over all boundary nodes of the product of the absolute amounts of these two terms, though for large numbers of boundary nodes we expect errors with different signs to largely cancel each other out.

A precise computation of the influence  $I_p$  of a node  $p$  on target node  $t$  is quite expensive, and thus we approximate this value as follows. We place one unit of rank value onto  $p$ , and then simulate PageRank for a while until most of the rank value has either left the subgraph (including random jumps) or arrived at the target. In particular, we adopt the OPIC approach recently proposed by Abiteboul et al. in [1] for this problem. In this approach, we select a node currently containing some rank value, and *fire* it, resulting in its rank value being pushed out of the node to other nodes. We remove any rank arriving at the target, leaving the subgraph, or taking a random jump, and terminate this process when the amount of rank still in the subgraph has decreased below some threshold. We then estimate the influence by the total amount of value that has arrived at the target. The process can be made more efficient by firing in each step the

node that contains the most rank value (based on a heap structure), and thus after  $O(s \log_\alpha(\epsilon))$  firings the influence value can be estimated within an additive term  $\epsilon$ , where  $s$  is the size of the subgraph.

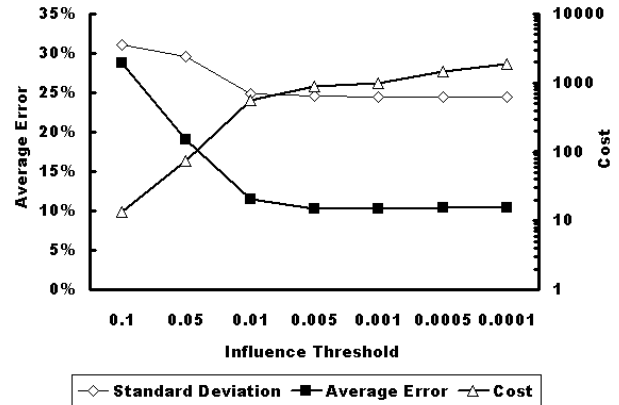


Figure 4: Error versus cost for simple influence method.

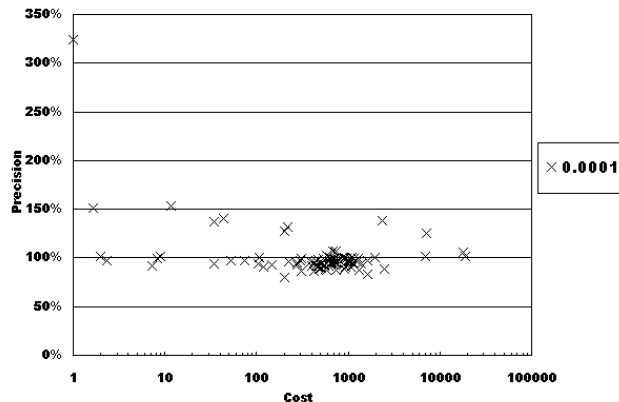


Figure 5: Precision at threshold 0.0001 versus cost for simple influence method.

In our second method, we attempt to improve efficiency by selectively expanding the subgraph only at boundary nodes that have influence above some threshold  $c$ . In the following experiment, we set  $c$  to 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, and 0.0001. Figure 4 shows results for the various thresholds. When a large threshold is used, only few nodes are expanded, but the error is very high. With a threshold of 0.005 we get an average relative error around 10% with a cost of 1000, which is much better than before. However, the standard deviation is still significantly higher than the average error due to a few estimates that are way off.

### 3.5 The Indegree-Based Influence Method

The influence test provides useful information for choosing when to stop expanding. However, some nodes with large

in-degree can create problems. If their influence is above the threshold, then expanding them would be very expensive as we have to fetch all nodes pointing to them. Not expanding them and just estimating their value as  $1/N$  is also a problem, potentially even if their influence is low, since such nodes typically have a large absolute PageRank. The impact of such a node on the estimation at the target is given by the product of its influence and its (in this case large) absolute estimation error.

Thus, we need to refine our approach. In particular, we use the following rule: If the influence of a node divided by its in-degree is greater than the threshold  $c$ , we expand the node by fetching its predecessors; otherwise, we stop. This solves the first of the above issues. To solve the second issue, we experiment with better heuristics for estimating the PageRank of the boundary nodes. We compared the following approaches:

- (a) estimate the PageRank as  $1/N$  as before.
- (b) estimate the PageRank by assuming that each edge from outside the subgraph into the boundary node transmits a PageRank value of  $\alpha/E$ , where  $E$  is the total number of hyperlinks in the graph. Rank values due to random incoming jumps and from other nodes in the subgraph can be directly computed and do not need to be estimated.
- (c) estimate the PageRank using the weighted in-degree defined in Section 2. We note that this would assume that this measure is precomputed and stored in the database, which may not be realistic.

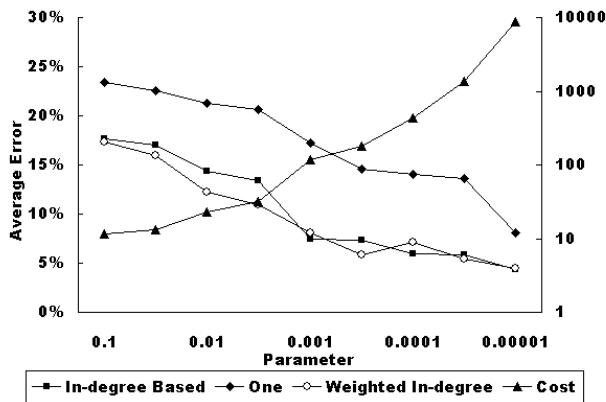


Figure 6: Average relative error versus cost for in-degree based influence methods.

In Figure 6 we show the cost of the method and the average relative errors for all three estimation approaches (the cost stays the same), for various thresholds  $c$ . Also, Figure 8 compares the naive and simple influence methods from the previous subsections to the method (b) above in terms of the error/cost tradeoff. We see from Figure 6 that using the number of links together with the influence to decide whether to expand decreases the cost for a given threshold value  $c$  compared to the simple influence method, and that methods (b) and (c) perform much better than (a) in terms of error. Also, we see from Figure 8 that (b) achieves

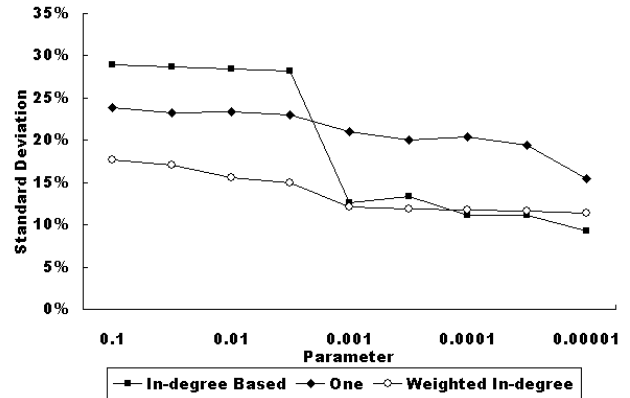


Figure 7: Standard deviation versus cost for in-degree based influence methods.

an overall better error/cost tradeoff than our earlier approaches, while Figure 7 shows improvements in the standard deviation due to fewer outliers.

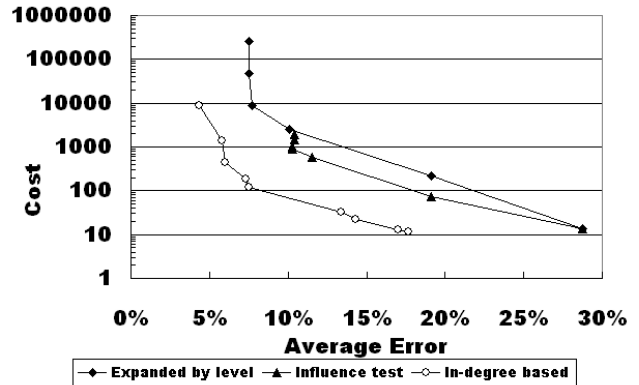


Figure 8: Comparison of error/cost tradeoffs for three methods from Subsections 3.3, 3.4, and 3.5.

Concerning absolute numbers, using method (b) we get an average relative error below 8% with an average cost of 118 fetch operations. We note that the cost of the influence computations and of the PageRank iterations in the third phase are insignificant, and thus the number of fetches provides a reasonable model.

### 3.6 Estimating Updated PageRank Values

To check the correctness of our code, we also ran an experiment where we estimated the boundary nodes using their exact PageRank values, to see if we would also obtain the exact value at the target node (we did). This test raised the following question for us: Suppose we have outdated PageRank values of some or all nodes available from a previous crawl, and we want to estimate the current value, can we do better by following our approach and estimating the

rank values of the boundary nodes using the old values? To test this hypothesis, we used two different crawls that we performed about 15 months apart using the Polybot web crawler [25] on the same set of start pages and crawl policies. Thus, both crawls have 120 million pages before pruning, and about 50 million afterwards. The two snapshots have about 13.6 million pages and one million sites in common. We now briefly discuss the observed changes between the two graphs.

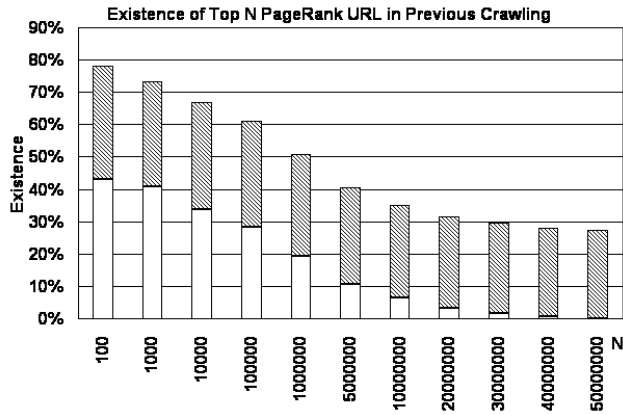


Figure 9: Observed changes in PageRank between old and new web graph (page level).

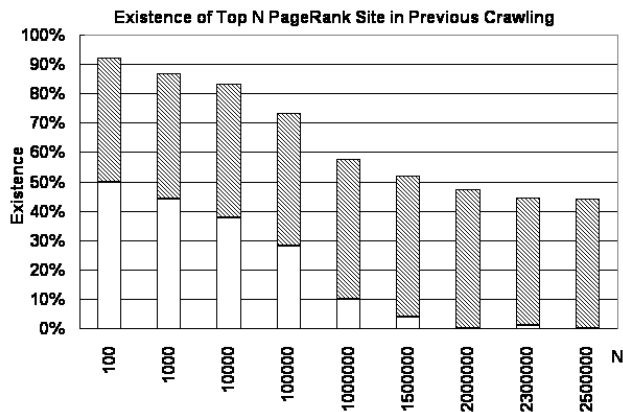


Figure 10: Observed changes in PageRank between old and new web graph (site level).

In Figure 9, we show how many of the pages ranked in the top  $N$  in the new graph were present in the old graph (total height of column) and how many were also ranked in the top  $N$  in the old graph (shaded part of column), for values of  $N$  from 100 to the total number of nodes. In Figure 10, we show results for a site-level view, where each site is represented by its highest-ranked URL (with PageRank computed on a page-level graph as before). We see clearly that pages that are highly ranked in the new graph were also likely to have existed and been crawled in the old graph, and in fact often

were already ranked high in the old graph. This tendency is even stronger at the site level since in some cases sites move their most highly ranked page to a different URL. We note that according to [20], a breadth-first crawl of sufficiently large size is likely to visit most pages with large PageRank. We also refer to [4] for additional discussion of PageRank in the context of web evolution.

Now we look at what happens if we directly plug old PageRank values into boundary nodes instead of approximating them with our other techniques. For those nodes where no old PageRank values are available, we use approach (b) in Subsection 3.5, i.e., estimation by in-degree. Figures 11 and 12 show the results for this method. The cost is the same as in Subsection 3.5, but the average error is slightly worse.

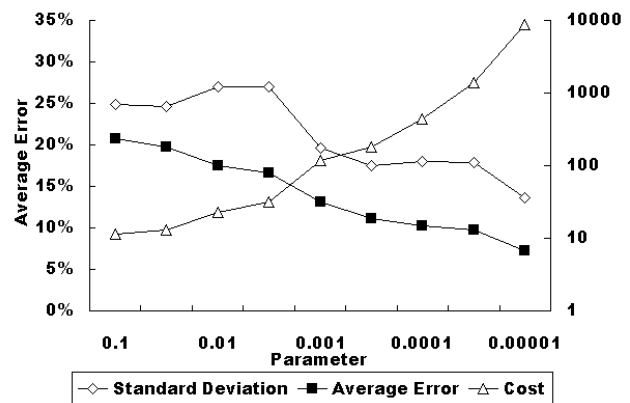


Figure 11: Error versus cost when using old PageRank values.

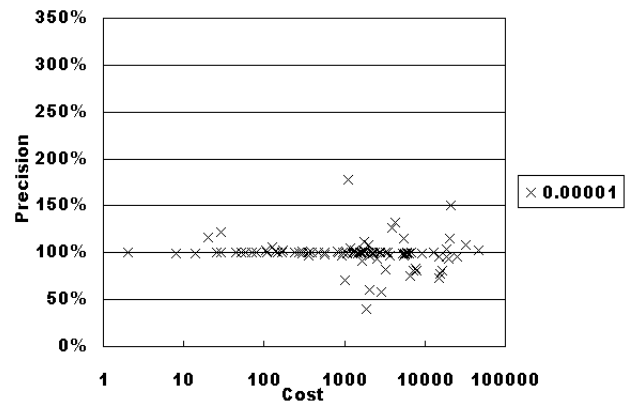


Figure 12: Precision versus cost when using old PageRank values.

There seem to be two problems with this approach. First, the old graph is quite old at 15 months and this limits the quality of the results. Second, on closer examination we find that many of the nodes with high PageRank, which usually already existed in the old graph, have significantly

increased their in-degree and PageRank value. On the other hand, some nodes have lost significant numbers of incoming links. Thus, if such nodes are boundary nodes in our expanded web graph, which is quite common due to our pruning condition involving in-degree, this could result in significant estimation errors.

This problem could be addressed by trying to correct the old PageRank value for changes in in-degree, and we experimented with two ideas. If  $d_{new}$  and  $d_{old}$  are the new and old in-degrees of some boundary node, respectively, then we could estimate the new PageRank  $r_{new}$  of such a node as  $\frac{d_{new}}{d_{old}} \cdot (r_{old} - (1-\alpha)) + (1-\alpha)$ , i.e., by normalizing the PageRank due to incoming edges by the change in the number of edges. Alternatively, we could add or subtract  $\alpha/E$ , the average flow over an edge in the graph, from the old PageRank for each change in degree, i.e.,  $r_{new} = r_{old} + (d_{new} - d_{old}) \cdot \frac{\alpha}{E}$ . We refer to these correction rules as Rule1 and Rule2, respectively. A third rule, Rule3, uses the first approach when the new degree is smaller than the old degree, and the second approach otherwise; the intuition behind this is that deletions are at random from the current incoming edges, while new incoming edges are from random nodes in the graph. (This approach can be further refined if we know all the URLs or IDs of the old and new incoming links.)

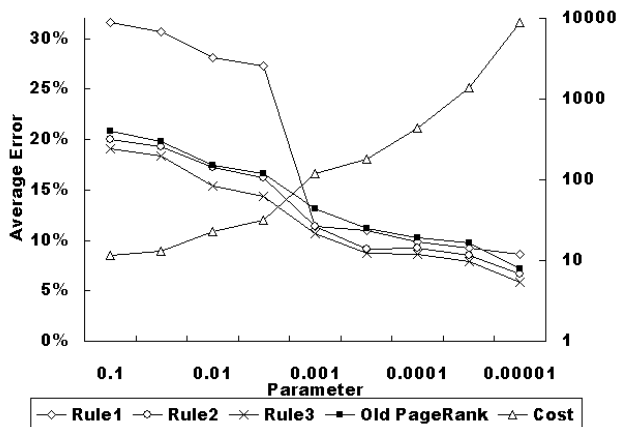


Figure 13: Error versus cost when using correction methods in combination with old PageRank values.

In Figure 13 we show the results, for the basic approach of using the raw old PageRank value and the three correction rules. We see that Rule1 does worse for high thresholds but later does slightly better than the basic method. The other two rules consistently, though only slightly, outperform the basic method, with Rule3 performing best and achieving about 5.85% error in the best case. Thus, some improvements are possible but we are limited by the age of the graph.

### 3.7 Correlations between PageRank and In-Degree Measures

In one of our two best methods, we used the in-degree of the boundary nodes to estimate their PageRank, by multiplying the in-degree by the average edge flow in the entire graph. In the other one, we used the weighted in-degree with slightly better results. It was observed in [23] that the

in-degree is in fact not strongly correlated to the PageRank value (although there is a slight correlation that makes it at least better than estimating the value by  $1/N$  for our purposes). On the other hand, the weighted in-degree measure was not studied in [23]. This leads us to look at the correlation between iterated versions of the weighted and unweighted in-degree, shown in Table 3.1 below. We see that while there is some correlation with in-degree (I), not surprisingly this correlation largely disappears for two-level (I-I) and three-level (I-I-I) iterated in-degree. On the other hand, for weighted in-degree (WI), the correlation is much higher and increases with more levels (WI-WI and WI-WI-WI) to close to 1. Of course, computing the weighted in-degree over several levels is really similar (but not identical) to expanding a subgraph by several levels (our naive method) and thus not really a more efficient alternative to our methods. For comparison, the correlation between new and old PageRank values for those nodes that exist in both graphs is 0.609.

	I	I-I	I-I-I
correlation coefficient	0.37665	0.031919	0.003822
	WI	WI-WI	WI-WI-WI
correlation coefficient	0.5305	0.67166	0.948538

Table 3.1: Correlations of in-degree (upper part) and weighted in-degree (lower part) based measures to PageRank.

### 3.8 Discussion of the Remaining Error

By optimizing our techniques step by step, we were able to obtain average errors below 5% with a few thousand fetches. We noticed that most of the target nodes have very small errors, while a few others have fairly large errors. Recall that in all our methods, if we put the correct PageRank value into the boundary nodes, we get the correct PageRank value at the target node. Thus, the error at the target node comes from over- or underestimation of the PageRanks of boundary nodes. More precisely, the influence of a boundary node multiplied by the estimation error at the boundary node give us an upper bound on the impact on the target node estimation, though over- and underestimates at different boundary nodes may cancel each other out. In particular, we have

$$|err(t)| \leq \sum_{b \in B} |err(b)| \cdot inf(b),$$

where  $err(t)$  is the error at the target node,  $B$  is the set of boundary nodes, and  $inf(b)$  is the influence of a boundary node  $b$ .

In our measurements, we found that boundary nodes with large influence, which due to our expansion criteria must have large in-degrees, also tend to have large absolute estimation errors under our various heuristics. Thus, these boundary nodes tend to limit the precision that we can achieve with our approach. One solution would of course be to expand backwards one more level from such a boundary node, but this would increase the cost significantly. Another approach would be try to better approximate the PageRank of such a boundary node by selectively sampling and expanding to some of the nodes outside the subgraph that have links to the boundary nodes. This would work if a lot of the incoming edges carry above or below average amounts

of PageRank, maybe due to certain characteristics of a particular region of the web. However, this approach would not work if the estimation error at the boundary node is due to a small number of incoming links that come from nodes with very high PageRank and that thus carry a much larger than average amount of PageRank. (Formally, sampling cannot reliably estimate the sum or average of a large set of values if that sum is dominated by a few very large values.) Thus, while such selective expansion might help in improving the precision/cost trade-off in some cases, it seems that in other cases we still need to expand a very substantial subgraph in order to reliably and precisely estimate the PageRank of the target node.

#### 4. DISCUSSION OF RELATED WORK

Link analysis techniques are playing an increasingly important role in web search engines, and a large number of techniques have been proposed. The two best known approaches probably are PageRank, proposed in [7] and used by Google, and the HITS method proposed by Kleinberg [18]. A number of extensions of these approaches were subsequently described; see, e.g., [14, 24, 8, 16, 21]. We refer to [2, 9] for an overview of link ranking techniques. There is also a lot of recent interest in efficient graph computations in the database community, for scenarios such as graph data mining and processing of ranked queries in database systems, where large graph data is often stored on disk in a format similar to our approach.

Recent work by Chien et al. [11] studies the problem of maintaining the results of a PageRank computations under changes in the graph structure. Their technique is similar to ours in that they also expand a small subgraph around a point of interest, in this case a change to a link or node in this graph, and collapse the rest of the graph into one large supernode. Then resulting changes in the PageRank values are propagated only within this graph. An important difference is that in [11] the graph is expanded by following forward links from the point of interest, while we expand backwards until some appropriate stopping criterion is met. As discussed, our approach can also be used to estimate updated from outdated PageRank values, but in contrast to [11] no knowledge of the precise changes in the web graph is required.

Since the early days of search engines, counting the in-degree of a node to estimate the page importance has been used as a form of link-based ranking [19], and there is a wealth of research that studies the in-degree properties of the web graph. Simply counting the in-degree of a node encourages spamming and is thus not considered a good ranking method. However, it is interesting to try to relate in-degree to PageRank and more advanced methods. In this context, Pandurangan, Raghavan, and Upfal [23] studied the relationship between PageRank and in-degree and discuss random graph models for capturing the PageRank, in-degree, and out-degree distributions in the real Web. Their results showed that there is only very weak correlation between PageRank values and node degrees, which is relevant to our problem of estimating PageRanks of boundary nodes discussed in Section 3.

The notion of the influence between two nodes also plays a role in the work on *Personalized Pagerank* by Jeh and Widom [16], which studies the efficient computation of personalized PageRank values based on a combination of pre-computation and online computation on small subgraphs.

In our computation of the influence, we adapt a technique proposed by Abiteboul, Preda, and Cobena [1] for the purpose of approximating rank values during a crawl. Finally, we note that PageRank estimation during a crawl based on the already crawled part of the web is also performed in [15, 12]. However, these methods do not attempt to optimize the precision of their estimates by expanding a subgraph in a controlled fashion, but instead simply use the available subgraph.

#### 5. CONCLUSIONS AND FUTURE WORK

In this paper, we have described and evaluated several heuristic algorithms for estimating PageRank values of individual pages without a global computation over the entire graph. As we have shown, a reasonable estimate of the value is possible in a few seconds on a typical workstation based on maybe a few hundred disk or remote server accesses to retrieve an appropriate subgraph. On the other hand, even a larger amount of work does not result in a truly reliable estimate due to the structure of the graph and in particular due to the difficulty of dealing with boundary nodes with large PageRank that can have a large impact on the target node. For future work, it would be nice to evaluate the proposed methods under a formal web graph model. We are also interested in applying the methods to the analysis of large archives of web data with multiple versions of pages and an evolving link structure.

We are currently performing a more detailed comparison of the structure of our two large web graphs and plan to report these results in the final version of this paper. We are also evaluating how our estimation methods fare when we are only interested in the relative ordering of the PageRank values of the different pages, rather than the absolute PageRank values.

#### Acknowledgements:

This work was supported by NSF CAREER Award NSF CCR-0093400 and the New York State Center for Advanced Technology in Telecommunications (CATT) at Polytechnic University, and by equipment grants from Sun Microsystems and Intel Corporation. Yen-Yu Chen was also supported by a Sun Foundation (Taiwan, R.O.C.) Fellowship.

#### 6. REFERENCES

- [1] S. Abiteboul, M. Preda, and G. Cobena. Adaptive on-line page importance computation. In *Proc. of the 12th Int. World Wide Web Conference*, May 2003.
- [2] A. Arasu, J. Cho, H. Garcia-Molina, and S. Raghavan. Searching the web. *ACM Transactions on Internet Technologies*, 1(1), June 2001.
- [3] A. Arasu, J. Novak, Tomkins A, and J. Tomlin. Pagerank computation and the structure of the web: Experiments and algorithms. In *Poster presentation at the 11th Int. World Wide Web Conference*, May 2002.
- [4] R. Baeza-Yates, F. Saint-Jean, and C. Castillo. Web dynamics, age and page quality. In *String Processing and Information Retrieval (SPIRE)*, September 2002.
- [5] K. Bharat, A. Broder, M. Henzinger, P. Kumar, and S. Venkatasubramanian. The connectivity server: Fast

- access to linkage information on the web. In *7th Int. World Wide Web Conference*, May 1998.
- [6] K. Bharat and M. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proc. 21st Int. Conf. on Research and Development in Inf. Retrieval (SIGIR)*, August 1998.
- [7] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proc. of the Seventh World Wide Web Conference*, 1998.
- [8] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *Proc. of the 7th Int. World Wide Web Conference*, May 1998.
- [9] S. Chakrabarti, B. Dom, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, David Gibson, and J. Kleinberg. Mining the web's link structure. *IEEE Computer*, 32(8):60–67, 1999.
- [10] Y. Chen, Q. Gan, and T. Suel. I/O-efficient techniques for computing pagerank. In *Proc. of the 11th International Conf. on Information and Knowledge Management*, pages 549–557, November 2002.
- [11] S. Chien, C. Dwork, R. Kumar, D. Simon, and D. Sivakumar. Link evolution: Analysis and algorithms. In *Workshop on Algorithms and Models for the Web Graph*, 2002.
- [12] J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through URL ordering. In *7th Int. World Wide Web Conference*, May 1998.
- [13] T.H. Haveliwala. Efficient computation of pagerank. Technical report, Stanford University, October 1999. Available at <http://dbpubs.stanford.edu:8090/pub/1999-31>.
- [14] T.H. Haveliwala. Topic-sensitive pagerank. In *Proc. of the 11th Int. World Wide Web Conference*, May 2002.
- [15] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. On near-uniform URL sampling. In *Proc. of the 9th Int. World Wide Web Conference*, May 2000.
- [16] G. Jeh and J. Widom. Scaling personalized web search. In *12th Int. World Wide Web Conference*, 2003.
- [17] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Extrapolation methods for accelerating pagerank computations. In *Proc. of the 12th Int. World Wide Web Conference*, May 2003.
- [18] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
- [19] Y. Li. Toward a qualitative search engine. *IEEE Internet Computing*, August 1998.
- [20] M. Najork and J. Wiener. Breadth-first search crawling yields high-quality pages. In *10th Int. World Wide Web Conference*, 2001.
- [21] A. Ng, A. Zheng, and M. Jordan. Stable algorithms for link analysis. In *Proc. of the 24th Annual SIGIR Conf. on Research and Development in Information Retrieval*, 2001.
- [22] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Computer Science Department, Stanford University, 1999.
- [23] G. Pandurangan, P. Raghavan, and E. Upfal. Using pagerank to characterize web structure. In *Proc. of the 8th Annual Int. Computing and Combinatorics Conference (COCOON)*, 2002.
- [24] M. Richardson and P. Domingos. The intelligent surfer: Probabilistic combination of link and content information in pagerank. In *Advances in Neural Information Processing Systems*, 2002.
- [25] V. Shkapenyuk and T. Suel. Design and implementation of a high-performance distributed web crawler. In *Proc. of the Int. Conf. on Data Engineering*, February 2002.