Internet Mathematics Vol. 1, No. 3: 277-304

Link Evolution: Analysis and Algorithms

Steve Chien, Cynthia Dwork, Ravi Kumar, Daniel R. Simon, and D. Sivakumar

Abstract. We anticipate that future web search techniques will exploit *changes* in web structure and content. As a first step in this direction, we examine the problem of integrating observed changes in link structure into static hyperlink-based ranking computations.

We present a very efficient algorithm to incrementally compute good approximations to Google's PageRank [Brin and Page 98], as links evolve. Our experiments reveal that this algorithm is both fast and yields excellent approximations to PageRank, even in light of large changes to the link structure.

Our algorithm derives intuition and partial justification from a rigorous sensitivity analysis of Markov chains. Consider a regular Markov chain with stationary probability π, and suppose the transition probability into a state j is increased. We prove that this can only cause

- π_j to *increase*—adding a link to a site can only cause the stationary probability of the target site to increase;
- the rank of j to *improve*—if the states are ordered according to their stationary probabilities, then adding a link to a site can only cause the rank of the target site to improve.

This analysis formalizes why the intuition that drives Google *never* fails.

1. Introduction

With rapid advances in storage and processing technology, we anticipate a next generation of web algorithms, where the *evolution* of content and link structure

1542-7951/04 $0.50 per page

in the web plays a crucial role. Algorithms that exploit evolution will therefore assume increasing significance.

For the present, link analysis ([Brin and Page 98, Kleinberg 99] and many variants) is recognized as the state-of-the-art tool in web search algorithm design. However, for the purposes of a large-scale and general-purpose search engine, such as Google, the fact that the web is an evolving object already poses serious algorithmic and engineering challenges. Computations of measures such as PageRank are typically too massive to be performed frequently. Therefore, the design of algorithms that lend themselves to efficient *incremental* implementations is already important.

The web is frequently viewed as a directed graph whose nodes are the static pages and whose edges are the hyperlinks. Recall that the PageRank of a site a is the stationary probability of a node corresponding to a in a Markov chain defined by the web graph ([Brin and Page 98], details below).

The PageRank update algorithm. We obtain a fast method to provide approximate incremental updates to PageRank. This demonstrates two phenomena: despite the holistic nature of PageRank, it admits incremental updates to a large extent; and the fact that links evolve is therefore not necessarily an obstacle for PageRank (it could, in fact, be an advantage). To the best of our knowledge, this is the first effort where a large-scale web application is tailored to suit the evolutionary nature of the web.

At a high level, the algorithm (Section 3) works in the following natural way. Given a set of link changes, we identify a small portion of the web graph in the vicinity of these changes, and model the rest of the web as a single node in this small graph. We then compute a version of PageRank on this small graph and suitably transfer these results to the original graph.

Our experiments on large crawls of the web as well as on synthetically generated graphs demonstrate that our algorithm performs exceedingly well both in speed and quality. The algorithm also appears robust to various types of link modifications—small or large number of changes, random or correlated link additions—and with respect to the density of the underlying slice of the web graph.

Though our algorithm is fairly natural, it is not a priori clear why it would work well, given the sensitivity of eigenproblems to even small perturbations. We present a theoretical analysis that yields intuition for the high quality of our experimental results. We now describe the two monotonicity results for Markov chains obtained in this context.

This paper addresses link updates only. For a discussion of node updates, see Section 6.

Monotonicity results for Markov chains and PageRank. Let $P = \{p_{ij}\}$ be the transition matrix of a regular Markov chain.[1] Let π be the stationary probability distribution, i.e., $\pi P = \pi$. We are interested in the question of how the stationary probability π_j of a state j changes if some p_{ij} is increased (and other transitions are decreased) to obtain a new regular Markov chain $\tilde{P}$.

The connection to PageRank and link evolution is as follows. The web, with n nodes, defines a natural Markov chain, with each node (page) corresponding to a unique state, and with transition matrix $M = \{m_{ij}\}$, as follows. For each node i, let k be the number of children (that is, out-neighbors) of i. Then $m_{ix} = 0$ if x is not a child of i and $m_{ix} = 1/k$ otherwise.[2] In their seminal work, Brin and Page [Brin and Page 98] considered a *random web surfer*, who starts at an arbitrary web page and moves on to other pages according to the transition probability matrix M. Such a surfer might wind up at a web page with no children and so, nowhere to go. To avoid this, they tweaked the random process so that, at every step, with some fixed probability ε (usually, set to 1/7), the surfer chooses a random page uniformly from the set of all pages and jumps to the chosen page. This process yields the celebrated PageRank Markov chain [Brin and Page 98], defined by $P = (1-\varepsilon)M + (\varepsilon/n)J$, where J is the $n \times n$ matrix of all ones. Let π denote the stationary distribution of P, i.e., $\pi P = \pi$. The stationary probability π_i of a page i is its *PageRank*.

Changes in link structure correspond to addition of an "error matrix" E so that $\tilde{P} = P + E$ denotes the new Markov chain. Let $\tilde{\pi}$ denote the stationary distribution of $\tilde{P}$. Consider the special case in which E is all zeroes except in one row, say, i, and in which only one entry is positive, say, $e_{ij} > 0$. This corresponds to increasing the probability of a transition from state i to state j. We ask the natural *monotonicity* question: can this cause the stationary probability of state j to *decrease*? We show that the answer is negative, i.e., $\tilde{\pi}_j > \pi_j$. This answers an open question posed by Moni Naor in the affirmative—PageRank is *monotonic*, in that adding a link to a site cannot decrease that site's PageRank.

The term "Page*Rank*" is a bit confusing. The PageRank of a page is *not* its position in a ranking, rather, it is just a positive real number. Of course, pages can be ordered according to their stationary probabilities, but the fact that a stationary probability has increased *does not* mean that the state has maintained its rank ordering; there are examples to the contrary. We therefore ask a second natural monotonicity question: in the ordering of states according to their stationary probabilities, can the rank ordering of state j decrease? The answer is again negative. Thus, if pages are ordered by their PageRanks, and

[1] A chain with transition matrix P is *regular* if $\exists k > 0$ such that $P^k > 0$.

[2] We adopt the convention that if A is a matrix, then the ith row of A is denoted A_{i*}, the jth column of A is denoted A_{*j}, and the (i, j)th entry of A is denoted a_{ij}.

a link is added to page j, then j's relative popularity cannot decrease. In fact, the rank ordering of page j will outperform that of every page it outperformed before the link was added.

These monotonicity results are extremely intuitive, and we were startled not to find them in the literature. Indeed, were it not for Braess' Paradox, which says that adding a route can cause overall traffic delays to worsen, we would not have even suspected there was anything to prove.

Experiments. We first conducted a series of experiments based on the real web to demonstrate that our algorithm can be successfully used in a real-world setting. We imagine a situation in which we have PageRank accurately computed for our current description of the web. As we discover, via web crawl, that edges have been added and deleted, we apply our algorithm to periodically update PageRank on the fly with the hope that its final approximation of PageRank is close enough.

To test this, we studied two crawls of the web generated by the web crawler Mercator [Heydon and Najork 99]. The first crawl was done in May 2000, and the portion we used contained more than 61 million pages and more than 259 million edges whose source and destination pages were in different domains. (Local edges within the same domain are frequently ignored for this type of work [Broder et al. 00] since they tend to be navigational or purposely introduced to take advantage of ranking algorithms.) The second crawl was conducted in November 2000 and had substantial overlap with the first. We found a total of more than 17 million edge changes between the two crawls.

Beginning with the correct value of PageRank for the first crawl, we used our algorithm to successively update PageRank on these 17 million changes. The end result was extremely successful; the difference (in the L_1 metric) between our final approximation and the correct value was on the order of 5.9×10^{-5}, while the actual change in PageRank was 0.12. The total cost of our updates was much less than similarly frequent updates using a full-blown PageRank algorithm.

We also ran several experiments in which small numbers ($\leq$ 10,000) of edges were randomly added to the web graph to test how well we do in other situations. In each case, our results were similarly accurate while the costs of our updates were again very low.

Our second set of experiments are conducted with random graphs that are designed to possess properties of the web graph [Kumar et al. 00]. Our goal in experimenting with synthetic graphs is to obtain a finer understanding of the robustness of our algorithms to graphs of different edge densities and various edge update scenarios. Here, we observe that the difference between our approximation and the true PageRank is typically of the order of 10^{-10}, compared with an actual change in PageRank around 10^{-1}. As in the previous experiment, our algorithm is far more efficient than running the full-blown version of PageRank.

2. Sensitivity Analysis

Let P be the transition matrix of an n-state, regular, Markov chain. The stationary distribution of P is the unique vector π satisfying $\pi = P\pi$, $\sum_{j=1}^{n} \pi_j = 1$. Suppose P is perturbed by an error matrix E to obtain $\tilde{P} = P + E$. Letting $\tilde{\pi}$ denote the stationary distribution of $\tilde{P}$, the fundamental question in sensitivity analysis is how to express $\pi - \tilde{\pi}$ in terms of E; typically, $||\pi - \tilde{\pi}|| \leq \kappa ||E||$ for a *condition number* κ, for standard norms such as the L_1, L_2, or L_∞ norms.

We are interested in the special case in which E reflects the addition of a single edge from state i to state j in the web graph. This can be described as a perturbation E in which $e_{ij} > 0$ and $e_{ik} \leq 0$ for all $k \neq j$, with all entries not in row i being zero. Although there is an extensive literature on the sensitivity of analysis of Markov chains and perturbation theory, the question of how this affects the stationary distribution of π_j seems not to have been explicitly addressed.[3] We remark that these questions become much simpler to answer if the underlying Markov chain is reversible.

2.1. Basics

We assume we have a set of states $\{1, 2, \ldots, n\}$ and transition probabilities defined by a matrix P:

$$P \stackrel{\text{def}}{=} \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix}.$$

Naturally, $\forall i, \sum_j p_{ij} = 1$.

A Markov chain is *regular* if $\exists k > 0$ such that $P^k > 0$. This means that every state communicates with every other and that the chain is *aperiodic*, that is, $\forall i, \min d$: d divides all k such that $p_{ii}^{(k)} > 0$ is 1.

The fundamental theorem for regular chains (see, for example, [Kemeny and Snell 60, Theorem 4.1.4]) says there exists a probability distribution, called the *stationary distribution*, $\pi = (\pi_1, \ldots, \pi_n)$ such that

[3] Experimentally, Ng, Zheng, and Jordan [Ng et al. 01a, Ng et al. 01b] compared the *stability* of PageRank and HITS [Kleinberg 99] under massive changes to a graph derived from the *Cora* citation database [McCallum et al. 00].

1. $\forall i, \pi_i \geq 0$ and $\sum_{i=1}^{n} \pi_i = 1$;
2. $\pi P = \pi$;
3. $\forall i, j, \pi_j = \lim_{k\to\infty}(P^k)[i,j]$.

The following elementary theorem is proved, for example, in [Kemeny and Snell 60, Theorem 1.11.1].

Theorem 2.1. *If* $\lim_{k\to\infty} M^k = 0$*, then* $(I - M)^{-1}$ *exists and*

$$(I - M)^{-1} = \sum_{i=0}^{\infty} M^i.$$

Define

$$B \stackrel{\text{def}}{=} \lim_{k\to\infty} P^k = \begin{pmatrix} \pi_1 & \cdots & \pi_n \\ \vdots & & \vdots \\ \pi_1 & \cdots & \pi_n \end{pmatrix}.$$

Note that $B^k = B$ for all $k \geq 1$ and $\pi B = (\pi_1 \sum_i \pi_i, \pi_2 \sum_i \pi_i, \ldots, \pi_n \sum_i \pi_i) = \pi$. The following are immediate.

Lemma 2.2. $BP = PB = B$.

Corollary 2.3. $\forall k \geq 1, P^k B = PB^k = B^k P = B$*, and* $(P - B)^k = P^k - B$.

2.2. The Fundamental Matrix

The *fundamental matrix* Z of P is defined as

$$Z \stackrel{\text{def}}{=} (I - (P - B))^{-1}. \tag{2.1}$$

Kemeny and Snell provide a beautiful treatment of the fundamental matrix [Kemeny and Snell 60, Section 4.3], and we will use several facts proved in their book. Z has the following natural interpretation [Kemeny and Snell 60, page 77]. Given two states i and j, let $r_{ij}^{(k)}$ be the expected number of times that P will be in state j in the first k steps when started from state i (including the initial state as one step). Then

$$\lim_{k\to\infty} r_{ij}^{(k)} - k\pi_j = z_{ij} - \pi_j.$$

Thus, in the limit, the number of "extra" steps that P will spend in state j when started from state i can be described as $z_{ij} - \pi_j$.

Theorem 2.4. [Kemeny and Snell 60] *The fundamental matrix of a regular Markov chain with transition matrix P always exists, and furthermore,*

$$Z = I + \sum_{k=1}^{\infty} (P - B)^k. \tag{2.2}$$

For states i and j of a Markov chain, let m_{ij} denote the *mean first passage time* from i to j, that is, the expected number of steps before entering state j from state i. The mean first passage time is always positive.

We will use the following important properties of Z.

Lemma 2.5.

(i) The matrix of mean first passage times can be expressed in terms of Z and π, as follows:

$$M = (I - Z + JZ_{\text{dg}})D, \tag{2.3}$$

where D is diagonal with $d_{ii} = 1/\pi_i$, J is all 1s, and Z_{dg} has 0s off the diagonal and z_{ii} in the (i,i)th position, for all $1 \leq i \leq n$;

(ii) $\pi Z = \pi$;

(iii) $\tilde{\pi} = \tilde{\pi} E Z + \pi$;

(iv) Z is diagonally dominant over columns and for all $j \neq i$, $z_{jj} - z_{ij} = m_{ij}\pi_j$;

(v) $\forall j,\ z_{jj} > 0$.

Proof.

(i) is proved in [Kemeny and Snell 60, Theorem 4.4.7].

(ii) follows from Theorem 2.4 and Corollary 2.3 as

$$\pi Z = \pi I + \sum_{k=1}^{\infty} \pi (P - B)^k = \pi + \sum_{k=1}^{\infty} \pi (P^k - B) = \pi.$$

(iii) is a major result due to Schweitzer [Schweitzer 68].

For (iv), note that by Equation 2.3, we have $M - D = (JZ_{\text{dg}} - Z)D$. The rows of JZ_{dg} are all equal to $(z_{11}, z_{22}, \ldots, z_{nn})$, the rows of $JZ_{\text{dg}}D$ are all $(z_{11}/\pi_1, z_{22}/\pi_2, \ldots, z_{nn}/\pi_n)$, and, for every i, the ith row of ZD is $(z_{i1}/\pi_1, z_{i2}/\pi_2, \ldots, z_{in}/\pi_n)$. So for all $i \neq j$, since $d_{ij} = 0$ we have $m_{ij} - d_{ij} =$

$m_{ij} = z_{jj}/\pi_j - z_{ij}/\pi_j$. Now, (iv) follows by arithmetic and the fact that mean first passage times are strictly positive.

To see (v), notice that from (iv) we have $\forall i \neq j, z_{jj} > z_{ij}$. Suppose for contradiction that $z_{jj} < 0$. Then for all i, $z_{ij} < 0$. Also, from (ii), $\sum_i \pi_i z_{ij} = \pi_j$. The z_{ij}s are all negative, and the π_is are all positive, yielding the contradiction. □

2.3. Monotonicity of Stationary Probability

Our first monotonicity result says that the stationary probability of j can only increase.[4]

Theorem 2.6. *Let P be the transition matrix of a finite-state regular Markov chain and let i, j be arbitrary states of P (not necessarily distinct). Let E be a matrix that is zero everywhere except in row i, $e_{ij} > 0$ is the only positive entry, and such that $\tilde{P} = P + E$ is also the transition matrix of a regular Markov chain. Let $\tilde{\pi}$ denote the stationary distribution of $\tilde{P}$. Then $\tilde{\pi}_j > \pi_j$.*

Proof. The entries in the ith row satisfy $e_{ij} > 0$, $\sum_k e_{ik} = 0$, and $\forall k \neq j, e_{ik} \leq 0$, corresponding to increasing transition probability from i into j, and decreasing (some) other probabilities from i accordingly. We wish to show that $\tilde{\pi}_j > \pi_j$.

Using the fact that $\tilde{\pi} = \tilde{\pi} E Z + \pi$ by Lemma 2.5(iii), and the special form of E, we have,

$$\begin{aligned}\tilde{\pi} - \pi &= (\tilde{\pi}_1, \ldots, \tilde{\pi}_n) \begin{pmatrix} 0 & \cdots & 0 \\ E_{i*}Z_{*1} & \ldots & E_{i*}Z_{*n} \\ 0 & \cdots & 0 \end{pmatrix} \\ &= (\tilde{\pi}_i E_{i*} Z_{*1}, \ldots, \tilde{\pi}_i E_{i*} Z_{*n}), \end{aligned} \tag{2.4}$$

whence,

$$\tilde{\pi}_k - \pi_k = \tilde{\pi}_i E_{i*} Z_{*k}.$$

Since $\tilde{P}$ is regular, we have that $\tilde{\pi}_i > 0$. Hence, to finish the proof, we need only show that $E_{i*}Z_{*j} > 0$. To see this, note that $E_{i*}Z_{*j} = e_{ij}z_{jj} + \sum_{k \neq j} e_{ik} z_{kj}$. By Lemma 2.5(iv) and 2.5(v), we have that $z_{jj} > 0$ and z_{jj} strictly dominates each of the other z_{kj}. Since $e_{ij} > 0$, and $e_{ij} = -\sum_{k \neq j} e_{ik}$, the result follows. □

[4] A natural approach to prove this result might be to collapse the original chain to a two-state chain and analyze the stationary probability of this collapsed chain. However, it is unclear if this would be easy since the relationship between the perturbed original chain and the perturbed collapsed chain is not straightforward.

Remark 2.7. The statement of the theorem requires that both P and $\tilde{P}$ have stationary distributions. We used regularity of $\tilde{P}$ only at the very end of the proof, in asserting that $\tilde{\pi}_i \neq 0$. If $\tilde{\pi}_i = 0$, then the proof shows that $\tilde{\pi}_j \geq \pi_j$.

Remark 2.8. In Section 7, we provide four additional proofs of Theorem 2.6, presented in the order in which they became known to us. The first was our initial proof—a more complicated version of the one presented above. The second relies on a theorem of Ipsen and Meyer [Ipsen and Meyer 94]. The third is due to Tim Roughgarden, and the fourth, due to an anonymous referee, relies on Lemma 2.10 below.

2.4. Monotonicity of Rank Ordering

In this section, we show that if the states are ordered according to their stationary probabilities, then the recipient of increased transition probability can only rise in the ordering.

We prove the theorem by showing that for any states j and k of P, if $\pi_j \geq \pi_k$ and we increase the transition probability into j from some state i, then $\tilde{\pi}_j \geq \tilde{\pi}_k$.

If we increase the transition probability from state i to state j, then we must decrease the transition probability from i to some other states, $w_1, w_2, \ldots w_y$, since the sum of the probabilities must be 1. We want to argue that there is no state k whose stationary probability starts out below that of j and whose final stationary probability exceeds that of j.

Theorem 2.9. *Let P be a finite-state regular Markov chain. Let i, j be arbitrary states of P (not necessarily distinct), and let $w_1, \ldots, w_y \neq j$ be arbitrary. Let $\tilde{P}$ be the Markov chain obtained by decreasing $p_{iw_1}, \ldots, p_{iw_y}$ by amounts $\delta_1, \ldots, \delta_y$ and increasing p_{ij} by $\sum_{\ell=1}^{y} \delta_\ell$; assume $\tilde{P}$ is regular. Let π and $\tilde{\pi}$ denote the stationary distributions of P and $\tilde{P}$, respectively. Then for all states $k \neq j$, if $\pi_j > \pi_k$, then $\tilde{\pi}_j > \tilde{\pi}_k$. Moreover, $\pi_j = \pi_k \Rightarrow \tilde{\pi}_j \geq \tilde{\pi}_k$.*

Proof. We first present the proof for the case $y = 1$ and then explain how to generalize to arbitrary y.[5] For simplicity, we write δ for δ_1 and p_{iw} for p_{iw_1}.

For any two states a, b, let flow(a, b) be the probability that a random walk starting at a reaches b before returning to a. For technical reasons, we define flow$(a, a) = 1$. We first show the following simple, but important, lemma.

[5] One might expect a hybrid argument here, that is, a series of steps in which transition probability is shifted from one e_{wi} to e_j, where one shows that at each step j cannot fall in the ranking. However, this would require a proof that the intermediate Markov chains are all regular.

Lemma 2.10. $\forall a, b, \pi_a = \pi_b \cdot \frac{\text{flow}(b,a)}{\text{flow}(a,b)}$

Proof. Define random variables τ and σ as follows. Beginning in state a, perform a random walk on the chain. Let τ be the first time the walk returns to a and let σ be the first time the walk returns to a *after* visiting b. Clearly, $\tau \leq \sigma$ and flow(a, b) is the probability that $\tau = \sigma$.

If $\tau < \sigma$, then *after* the first τ steps, the expected number of steps needed to get from a to b and back to a is $E[\sigma]$. Therefore, the expected difference between σ and τ is zero times the probability that $\sigma = \tau$ plus $E[\sigma]$ times the probability that they are not equal. This yields the equation

$$E[\sigma - \tau] = (1 - \text{flow}(a, b)) \cdot E[\sigma].$$

Since $E[\sigma - \tau] = E[\sigma] - E[\tau]$, we have that flow$(a, b) = E[\tau]/E[\sigma]$. Now, $E[\tau] = M_{aa}$ is the mean return time for state a and $M_{aa} = 1/\pi_a$. Also, $E[\sigma]$ is the *commute time* between a and b—expected time to go from a to b and back to a; it is clear that the commute time is a symmetric quantity. Piecing these together, we have $\pi_a \,\text{flow}(a, b) = \frac{1}{E[\sigma]} = \pi_b \,\text{flow}(b, a)$, and the lemma follows. □

We have the following immediate corollary.

Corollary 2.11. $\forall a, b, a \neq b, \pi_a > \pi_b$ *if and only if* flow$(a, b) <$ flow(b, a).

Assume the conditions of the theorem. Let $k \neq j$ be arbitrary. By Corollary 2.11, $\pi_j > \pi_k$ if and only if flow$(j, k) <$ flow(k, j). To prove the theorem, we examine how the two flows are affected by the change to P. We show that flow(j, k) cannot increase and flow(k, j) cannot decrease. Note that the following lemma holds *independent* of whether or not $\pi_j > \pi_k$.

Lemma 2.12. *For all* k, $\text{flow}_P(j, k) \geq \text{flow}_{\tilde{P}}(j, k)$ *and* $\text{flow}_P(k, j) \leq \text{flow}_{\tilde{P}}(k, j)$.

If $k = j$, then all the quantities involved are 1, by definition, and the lemma holds.

Let e be the edge from i to w, with transition probability p_{iw}. We conceptually "break" e into two edges, e_w and e'_w, with transition probabilities δ and $p_{iw} - \delta$, respectively. We will model reducing the transition probability p_{iw} by changing the probability along e_w to 0. Similarly, we "add" an edge e_j from i to j, initially with transition probability 0. We will model increasing the transition probability from i to j by changing the probability on e_j to δ and changing the probability on e_w to zero. At all times, only one of edges e_j and e_w will have nonzero transition probability δ, while the other will have transition probability zero.

Consider a random walk beginning at j and ending at the first of j and k to be reached (not counting the initial state). Let $[\text{flow}(j,k)|(e_j \vee e_w)]$ (respectively, $[\text{flow}(j,k)|\neg(e_j \vee e_w)]$) denote the conditional probability that such a random walk ends at k given that the walk passes (respectively, does not pass) through at least one of e_j or e_w:

$$\text{flow}(j,k) = [\text{flow}(j,k)|(e_j \vee e_w)]\,\Pr[(e_j \vee e_w)] + [\text{flow}(j,k)|\neg(e_j \vee e_w)]\,\Pr[\neg(e_j \vee e_w)]\,. \tag{2.5}$$

The second term in (2.5) does not change as a result of the modifications, since the modifications are only to the probabilities on edges e_j and e_w, so we need only consider the first term. Since e_j and e_w are mutually exclusive (only one is nonzero at any time), we write

$$[\text{flow}(j,k)|(e_j \vee e_w)]\,\Pr[(e_j \vee e_w)] = [\text{flow}(j,k)|e_j]\,\Pr[e_j] + [\text{flow}(j,k)|e_w]\,\Pr[e_w].$$

In the original chain, we have

$$[\text{flow}(j,k)|e_j]\,\Pr[e_j] + [\text{flow}(j,k)|e_w]\,\Pr[e_w] = \\ 0 \cdot 0 + \text{ (nonnegative) } \cdot \text{ (nonnegative)}\,,$$

while in the modified chain, we have

$$[\text{flow}(j,k)|e_j]\,\Pr[e_j] + [\text{flow}(j,k)|e_w]\,\Pr[e_w] = \\ 0 \cdot \text{ (nonnegative) } + \text{ (nonnegative) } \cdot 0,$$

and so $\text{flow}(j,k)$ cannot increase.

We now carry out a similar argument for $\text{flow}(k,j)$, showing that it cannot decrease:

$$\begin{aligned}\text{flow}(k,j) = {} & [\text{flow}(k,j)|(e_j \vee e_w)]\,\Pr[(e_j \vee e_w)] \\ & + [\text{flow}(k,j)|\neg(e_j \vee e_w)]\,\Pr[\neg(e_j \vee e_w)]\,.\end{aligned}$$

Again, the second term is unchanged by the modifications, so we focus on the first term:

$$[\text{flow}(k,j)|(e_j \vee e_w)]\,\Pr[(e_j \vee e_w)] = [\text{flow}(k,j)|e_j]\,\Pr[e_j] + [\text{flow}(k,j)|e_w]\,\Pr[e_w].$$

In the original chain, we have

$$[\text{flow}(k,j)|e_j]\,\Pr[e_j] + [\text{flow}(k,j)|e_w]\,\Pr[e_w] = 1 \cdot 0 + \text{ (nonnegative) } \cdot \Pr_P[e_w],$$

where $\Pr_P[e_w]$ is the probability in the original chain (P) that edge e_w is taken at least once in the random walk starting at k and ending at the first of j and k to be reached. In the modified chain, we have

$$[\text{flow}(k,j)|e_j]\,\Pr[e_j] + [\text{flow}(k,j)|e_w]\,\Pr[e_w] = 1 \cdot \Pr_{\tilde{P}}[e_j] + \text{ (nonnegative) } \cdot 0,$$

where $\Pr_{\tilde{P}}[e_j]$ is the probability in the modified chain $(\tilde{P})$ that edge e_j is taken at least once in the random walk starting at k and ending at the first of j or k

to be reached. We have defined e_j and e_w so that $\rho = \Pr_{\tilde{P}}[e_j] = \Pr_P[e_w]$, and so flow$(k, j)$ cannot decrease: the quantity ρ has coefficient 1 in the modified chain, but only some nonnegative coefficient bounded by 1 in the original chain. This completes the proof for the case $y = 1$.

To extend the proof to the case of arbitrary y, we make the following changes. Letting e_ℓ be the edge from i to w_ℓ, $\ell = 1, \ldots, y$, we conceptually break each e_ℓ into two edges e_{w_ℓ} and e'_{w_ℓ} with transition probabilities δ_ℓ and $p_{iw_\ell} - \delta_\ell$, respectively. The "added" edge e_j from i to j again initially has transition probability 0. We will model increasing the transition probability from i to j by changing the probability on e_j to $\sum_{\ell=1}^{y} \delta_\ell$ and changing the probabilities on the e_{w_ℓ} to zero. We let $e_w = e_{w_1} \vee \cdots \vee e_{w_y}$ denote both the set of edges $w_1, \ldots, w_y$ and, when used in a conditional probability, the condition that at least one of those edges is taken. The "transition probability" of e_w is the sum of the transition probabilities on the edges e_{w_ℓ}. As in the proof for the case $y = 1$, at all times, only one of e_j and e_w will have nonzero transition probability δ, while the other will have transition probability zero. That is, either e_j will have nonzero transition probability or each of the e_{w_ℓ} will have nonzero transition probability, but not both.

These changes imply the following interpretation of $[\text{flow}(j, k)|(e_j \vee e_w)]$: this expression now denotes the conditional probability that a random walk, starting at j and ending at the first of j and k to be reached, ends at k, given that the walk passes through at least one of $e_j, e_{w_1}, \ldots, e_{w_y}$. We reinterpret $[\text{flow}(j, k)|\neg(e_j \vee e_w)]$ analogously. The rest of the proof follows, *mutatis mutandi*, interpreting $\Pr_P[e_w]$ as the probability in the original chain (P) that one of the edges $e_{w_1}, \ldots, e_{w_y}$ is taken at least once in the random walk starting at k and ending at the first of j and k to be reached. □

3. The Algorithm

In this section, we first describe our incremental PageRank algorithm. First, we describe a special case of the algorithm for the case where a single edge has been added to the web graph—this algorithm derives motivation from our monotonicity results. This is followed by a description of how the algorithm generalizes to the case of multiple edge changes. We conclude with an analysis of several implementation issues as well as some theoretical analysis as to why the algorithm should be effective.

Consider the addition of an edge (i, j) to the web graph W. Again, let P and $\tilde{P}$ be the respective PageRank Markov chains associated with W before and after the new edge is added. We will construct a small subgraph G of the web graph that contains a small neighborhood around i and j; loosely speaking, this is the area of the web graph affected by the change. We will model the rest of the web graph $W \setminus G$ by a single "supernode" Ω; this defines a new, much smaller Markov chain T whose states are the pages of G and the supernode Ω. Our algorithm will compute the stationary distribution of T to find the new PageRank of the pages in G, while the pages in Ω will be assumed to keep their original PageRank.

Our algorithm thus implicitly assumes that the stationary distribution of nodes in the supernode is relatively unperturbed by the addition of the edge (i, j). The intuition for this is that while the stationary distribution of a Markov chain can be very sensitive to even small changes, the "random reset step" in PageRank ensures that any substantive changes in this distribution will indeed be relatively local to i and j. (Recall that random reset refers to the fact that at each step, the PageRank Markov chain will make a transition to a random node with a fixed probability ϵ.) If this intuition is correct, then the old and new stationary distributions π and $\tilde{\pi}$ will differ almost entirely on the nodes in G, and will be very close on the nodes in the supernode. Moreover, P and $\tilde{P}$ are identical on the rows corresponding to nodes in the supernode (indeed, they are identical on all rows except row i). Thus, modeling the supernode using π for the nodes in Ω should give a good approximation of the real behavior of the rest of the graph.

3.1. Details of the Single Edge Insertion Algorithm

As outlined above, we will construct a small subgraph G and a supernode Ω that models $W \setminus G$. We first outline how to proceed with the approximate PageRank algorithm after G has been constructed; we will then return to the construction of G. Where it is clear from context, we will denote by Ω both a single node in the new graph, and a set of nodes in the large graph W.

The transition matrix T for the new, smaller, Markov chain is defined as follows:

1. For two pages k and ℓ within G, we keep the transition probability from $\tilde{P}$ by setting $t_{k\ell} = \tilde{p}_{k\ell}$.

2. The transition probabilities from Ω and the pages in G are computed using π and P. For any $k \in G$, we set $t_{\Omega k} = \sum_{s \in \Omega} \frac{\pi_s}{Q} p_{sk}$, where $Q = \sum_{s \in \Omega} \pi_s$. Ω's self-loop probability is the complement of the sum of these probabilities.

3. The transition probabilities from the pages in G to Ω are computed similarly. For any $k \in G$, we have $t_{k\Omega} = \sum_{s \in \Omega} \tilde{p}_{ks}$.

We then compute the stationary distribution τ of T. For all $s \in \Omega$, our "approximate PageRank" $\hat{\pi}$ will be defined to be $\pi(s)$, and for all $k \in G$, it will be $\tau(k)$. These quantities are suitably normalized to obtain a probability distribution.

We now turn to the construction of the graph G, given (i, j). Our basic idea is to explore the web graph W around the nodes i and j, and collect a small set of nodes whose PageRank are likely to be most affected by the insertion of (i, j). To do this, we assign the node i a weight of 1, and then allow this weight to "dissipate" according to the PageRank dampening factor. Thus after one step, all its children (including j) will have weight $(1 - \epsilon)/d_i$, where d_i is the (new) out-degree of i. A node is chosen to be in the subgraph if its total weight (over all time-steps considered) exceeds a threshold δ. Note that there are two sources of approximation here: if we do not place any bound on the number of time-steps and if we take $\delta = 0$, we recover a truthful recomputation of PageRank for the entire graph W. Our idea is that a reasonable bound placed on these two quantities should still give a graph G so that the resulting approximation is sufficiently good, and G isn't too large.

3.2. Generalization and Implementation Issues

We will use the basic method outlined above extended in the following way. Suppose we have a collection of t edges $(i_1, j_1), \ldots, (i_t, j_t)$ that are being added/deleted. We perform a BFS around these nodes, again applying the weighting method using the PageRank dampening factors. The small graph G will then be the union of all nodes discovered in this process (carried out to a certain number of time steps and using a threshold δ); the supernode will contain all nodes not in G.

In our implementations, we use the following variant of PageRank (concerning the random reset step). If a node has out-degree zero, then its self-loop probability is $(\epsilon/n) + 1 - \epsilon$ and its other transitions have mass ϵ/n, where n is the number of nodes.

Because the web graph is so large, the only feasible method for computing PageRank is power iteration, which corresponds to simulating the Markov chain step by step. An initial starting probability distribution x is chosen, and we iterate by repeatedly multiplying $x \leftarrow xP$. A straightforward PageRank update algorithm would start with π, the stationary distribution for P, and perform power iteration with $\tilde{P}$ until convergence to the new stationary distribution $\tilde{\pi}$.

The cost of doing this can be very expensive. Since the edges are numerous but relatively sparse, they are usually stored in a compressed, sparse form such as in the Connectivity Server built at Compaq [Randall et al. 01]. In this model, each vector-matrix multiplication requires $O(n)$ calls to the connectivity server to retrieve a specific page's neighbors and $O(m)$ floating-point multiplications, where n is the number of pages and m the number of edges in the web graph.

By comparison, our algorithm will perform power iteration on the much smaller Markov chain T, using as its starting point the stationary distribution π for P, projected onto T in the natural way. Now each iteration requires only $O(n_T)$ calls to the connectivity server and $O(m_T)$ multiplications for each iteration, where n_T and m_T are the number of pages and edges in the new Markov chain, respectively. Typically these will be much smaller than n and m. In addition, we must pay a one-time cost to compute the transition probabilities from and to the supernode. Each of these preprocessing computations requires $O(n_T)$ calls to the connectivity server and a number of floating-point operations proportional to the number of edges entering or leaving G. Each preprocessing computation therefore requires at most the same cost as one full PageRank iteration, but typically far less than that. Thus, the total cost of running our algorithm is only a fraction of that of running full power iteration. Furthermore, among eigenvector techniques, power iteration converges rather slowly. In our case, however, if G is small enough, we might conceivably use other methods.

3.3. Analysis of the Single Edge Insertion Algorithm

We analyze how well the supernode created in our algorithm models the behavior of the nodes in $W \setminus G$. If, in creating the supernode, we were to use the (not yet known) values $\tilde{\pi}(x)$ for x in the supernode, then the modeling would be perfect, so the problem is to estimate the difference between the new values and the old.

Now, inserting an edge (i, j) into the web graph corresponds to increasing the transition probability p_{ij} from ε/n to something depending on the degree of the source. Let d_i denote the out-degree of node i in the (original) web graph. For each child k of i in the original graph,

$$p_{ik} = \frac{1-\varepsilon}{d_i} + \frac{\varepsilon}{n},$$

while for each nonchild x of i, including $x = j$, $p_{ix} = \varepsilon/n$. After the edge insertion, we have

$$\tilde{p}_{ij} = \frac{1-\varepsilon}{1+d_i} + \frac{\varepsilon}{n} \qquad \text{and for child } k \text{ of } i, \qquad \tilde{p}_{ik} = \frac{1-\varepsilon}{1+d_i} + \frac{\varepsilon}{n},$$

while for each nonchild x of i, now *excluding* $x = j$, $p_{ix} = \varepsilon/n$. Thus,

$$e_{ij} = \frac{1-\varepsilon}{1+d_i}.$$

For each child k of i in the original graph,

$$e_{ik} = -\frac{1-\varepsilon}{d_i(d_i+1)}$$

and for each x not a child of i in the new graph, $e_{ix} = 0$.

Now fix a page s in the supernode. We now bound the change in PageRank of s as follows, where we use (2.4) in the first line below and Lemma 2.5(i) in the second line (recall that Z is the fundamental matrix and M is the matrix of mean first passage times associated with P):

$$\begin{aligned}
\tilde{\pi}(s) - \pi(s) &= \tilde{\pi}(i) \sum_k e_{ik} z_{ks} \\
&= \tilde{\pi}(i) \sum_k e_{ik} \left(z_{ss} - m_{ks} \pi(s) \right) \\
&= \tilde{\pi}(i) \left(\sum_k e_{ik} z_{ss} - \sum_k e_{ik} m_{ks} \pi(s) \right) \\
&= \tilde{\pi}(i) \left(- \sum_k e_{ik} m_{ks} \pi(s) \right).
\end{aligned}$$

The last line holds because $\sum_k e_{ik} = 0$ (because both P and $\tilde{P}$ are stochastic).

Since $e_{ik} \neq 0$ only for children of i, this result states that $\tilde{\pi}(s) - \pi(s)$ depends on the difference between the mean first passage times to s from the different children of i.

Intuitively, if there is no short "natural" path in P from i to s—that is, a path using only web links and not random jumps—then the mean first passage times to s from any two of is children should not differ by much, since the chain is likely to experience a random jump before any long natural path can be explored. Thus, if the graph G contains all nodes reachable from i by relatively short (compared to $1/\varepsilon$) natural paths, then if s is not among them, the mean first passage times to s from any child of i will be close and hence $\tilde{\pi}(s) - \pi(s)$ will be very close to 0.

4. Experiments on the Real Web

The experiments in this section are meant to illustrate the performance of our algorithm in a real-world setting. En route, we also study our algorithm when there is a very small number of edge additions. Section 5 contains experiments on synthetically generated graphs.

Data. We used a web crawl produced at Compaq Systems Research Center. The crawl we used ran in May 2000 and produced a graph containing 61,272,004 pages and 259,411,961 edges. (Of these pages, 55,764,359 were explicitly crawled; the rest were assumed to exist because they were pointed to by the crawled pages.) We used this as our starting graph and computed its PageRank, π, by running power iteration for 50 iterations starting from the uniform distribution.

Small changes. We first tested our algorithm's basic competence on the cases justified by our theoretical analysis, namely small numbers of edge additions. We find that our algorithm performs as well as predicted.

We ran a series of experiments in which we added 1, 100, 1000, and 10000 edges to our graph. The cases of 100, 1000, and 10000 added edges used randomly chosen edges; the case of the single edge added an edge from www.yahoo.com to www.cs.berkeley.edu. In these experiments, we used π as our starting PageRank and our threshold δ while constructing the subgraph was 10^{-6}.

The results are described in Table 1. The first column indicates the number of edges added. The "Change in PageRank" column is the effect of the added edges on PageRank, as measured in the L_1 distance, or $\|\tilde{\pi} - \pi\|$. Subgraph coverage is the amount of this change that occurs in the constructed subgraph G, and subgraph size is the number of nodes in the subgraph G. Finally, "Our error" is the L_1 distance from the correct PageRank $\tilde{\pi}$ to our algorithm's approximation $\hat{\pi}$, or $\|\hat{\pi} - \tilde{\pi}\|$, and the last column is a measure of how much of the change in PageRank is "corrected" by our algorithm, or $1 - \frac{\|\hat{\pi} - \tilde{\pi}\|}{\|\tilde{\pi} - \pi\|}$, expressed as a percentage.

We see that after building relatively small subgraphs, our algorithm gives excellent approximations to the correct PageRanks in each case. The pages in the subgraphs that are built are responsible for nearly all of the change in PageRank, and this change is accounted for very effectively.

Edges	Change in PageRank	Subgraph coverage	Subgraph size	Our error	Correction
1	8.184×10^{-4}	99.94%	8,119	4.781×10^{-7}	99.94%
100	4.586×10^{-6}	96.99%	428,090	1.447×10^{-7}	97.85%
1000	4.446×10^{-5}	99.80%	1,311,395	1.050×10^{-7}	99.86%
10000	4.959×10^{-4}	99.99%	3,077,152	4.331×10^{-8}	99.99%

Table 1. Performance of our algorithm on a real web graph for small number of changes.

Large changes. Our theoretical analysis shows that our algorithm should work well on individual edge changes, and the above results indicate that we can also handle a small number of additions. Here we demonstrate empirically that our algorithm

Partition	Page changes	Edge additions	Edge deletions
1	127,415	443,015	423,250
10	90,701	295,063	285,197
20	40,560	122,287	114,493
30	29,924	78,743	63,274
40	22,929	66,617	56,836
50	19,569	53,866	45,552

Table 2. The number of changes in some sample partitions. The number of page changes is the number of pages whose set of outgoing edges changed.

also works well even on a sequence of successive updates, each containing tens or hundreds of thousands of edge changes.

In our model, we are a search engine that maintains a graph of the web and its associated PageRank. As we crawl the web and discover changes to our graph, we will use our algorithm to repeatedly update our computed PageRank to reflect the new changes. The following experiment shows that this works very well, and that the cumulative error from our updates is very small. In fact, we were able to account for 99.95% of the change in PageRank.

The set of edge changes was found by comparing the 61-million-page graph described at the beginning of this section to the graph produced by a second crawl conducted in November 2000. This second crawl did not completely cover the pages in the first, but the intersection was sufficient to discover a total of 8,930,307 edge additions and 8,433,037 edge deletions.

We then needed to simulate discovering these edge changes by a web crawl. To do this, we sequenced the 55 million crawled pages in our graph in the order they were crawled in May 2000, and sequenced the edges accordingly: the edge change e_1 is discovered before the edge change e_2 if e_1's source page was crawled before e_2's source page in the original crawl.

To simulate periodic updates, we partitioned the ordered list of more than 55 million crawled pages into 56 partitions, each containing 1,000,000 pages (except the last one, naturally). For each of these partitions, we ran our algorithm on the edge updates whose source pages were in that partition. In each case, we used a threshold value of 10^{-6} in building our subgraph. To compute PageRank within the subgraph, we ran power iteration for 30 steps.

Table 2 gives some indication as to the scale of the changes in each partition. We see that in the heaviest case, our algorithm is being asked to handle more than 800,000 changes to more than 100,000 source pages. Table 3 reports the sizes of the subgraphs built for each partition, and the number of internal edges they contain. These numbers should be compared with 61 million pages and

Partition	Pages in subgraph	Edges in subgraph	Change in PageRank
1	5,590,123	29,229,732	0.01752
10	5,302,006	26,848,170	0.00125
20	4,081,733	20,530,431	0.00065
30	3,520,523	17,751,694	0.00049
40	3,308,829	17,146,822	0.00030
50	3,089,939	15,545,991	0.00020

Table 3. The size of the subgraphs built for some sample partitions.

Actual change in PageRank	Our error	Correction
0.12056	5.9552×10^{-5}	99.95%

Table 4. Our results for real web graph.

259 million edges in the original graph. The last column indicates the change in PageRank for this partition in the L_1 metric as computed by our algorithm. We can see that each update requires far fewer computations than would be needed for the full 61 million pages and 259 million edges.

At the end of all the updates, we have the results described in Table 4. If we let π be the original PageRank, $\tilde{\pi}$ be the new correct PageRank, and $\hat{\pi}$ the approximation given by our algorithm, we have that $\|\pi - \tilde{\pi}\| = 0.12056$ while $\|\hat{\pi} - \tilde{\pi}\| = 5.9552 \times 10^{-5}$, meaning that our algorithm has corrected for 99.95% of the change in PageRank. In fact, there is a good chance our results are better than this, as the new PageRank was computed by running 50 additional steps of power iteration from the original PageRank. At the last step, the change in L_1 distance was still 7.04×10^{-6}, meaning that much of our error may result from inaccuracy of the new PageRank. This suggests that our algorithm performs well not only on small numbers of changes, but over a sequence of very large updates as well.

5. Experiments on Synthetic Graphs

We present the results of our experiments on random graphs designed to model the structure of the web graph. For generating synthetic graphs, we use variants of the stochastic model proposed by [Kumar et al. 00].

Before we present the model, we'll say a few words about why this experiment is interesting. With a synthetic graph, we have an opportunity to study variations in the parameters of interest. Specifically, we wish to obtain insights into the behavior of our approximation algorithm when we consider very large changes, e.g., when the number of edge changes is about 5% of all edges. Similarly, we could understand the way in which our algorithm performs at various densities of web-like graphs. This is important because there is evidence [Dill et al. 02] that parameters of the web graph at various scales (corporate intranets, topic-specific web page collections, etc.) are *similar* up to suitable scaling. In particular, different topic-specific collections tend to exhibit different densities of graphs. Therefore, we would like to know if the quality of approximation we produce is sensitive to the density parameter. Finally, we would like to understand the dependence of the quality of approximation on the *nature* of these changes—how random or correlated these changes are. Again, there is model-based evidence that the web evolves via mechanisms that tend to be a mixture of correlated and random changes, and the degree of the correlation depends on the slice of the web (and its scale) one considers. To the extent that these hypotheses hold true, we wish to gain an understanding of their influence on our algorithm.

We now proceed to the details. The copying model of [Kumar et al. 00] suggests the following evolution mechanism. For simplicity, we consider the model with one parameter β. At each time-step, with probability β, a new node is created, and with probability $1-\beta$, a new edge (i, j) is created among existing nodes (we choose $\beta = 0.3, 0.5, 0.7$ in our experiments). Note that β controls the density of the graph. For an edge addition, j is chosen to be uniform among existing nodes with probability 0.15 and chosen proportional to the current in-degree of existing nodes with probability 0.85. A similar procedure is applied to choose i—it is chosen to be uniform among existing nodes with probability 0.23 and chosen proportional to the current out-degree of existing nodes with probability 0.77. For a node addition, we choose a node among existing nodes and copy each of its links with probability 0.9. (The particular numerical values were chosen so that the in- and out-degree distribution implied by these values match those empirically observed (for instance, [Broder et al. 00]).)

We generated several synthetic graphs and the results from these experiments are in Table 5 and Table 6. We use the following convention in the tables. The first column (R) indicates the ratio of the number of added edges to the total number of edges of the original graph, expressed as a percentage. The second column ($|G|/|W|$) shows the number of nodes in the graph G constructed by our algorithm as a fraction of the number of nodes in the original graph. The third column (PR diff.) shows the true L_1 distance between the original PageRank π and the correct new PageRank $\tilde{\pi}$. The fourth column (Our error) shows the L_1 distance between the output $\hat{\pi}$ of our algorithm and the correct PageRank $\tilde{\pi}$.

R (%)	$\frac{\lvert G\rvert}{\lvert W\rvert}$	PR diff.	Our error	$\frac{\lvert G\rvert}{\lvert W\rvert}$	PR diff.	Our error	$\frac{\lvert G\rvert}{\lvert W\rvert}$	PR diff.	Our error
$\beta = 0.3$									
	60,421 nodes; 1,051,245 edges			300,013 nodes; 5,782,599 edges			750,063 nodes; 15,199,330 edges		
0.1	0.28	0.001	1×10^{-9}	0.28	0.001	3×10^{-09}	0.28	0.002	2×10^{-9}
0.2	0.34	0.002	6×10^{-10}	0.35	0.003	7×10^{-10}	0.35	0.004	1×10^{-9}
0.5	0.46	0.005	1×10^{-10}	0.47	0.009	3×10^{-10}	0.48	0.010	5×10^{-10}
1.0	0.59	0.013	5×10^{-11}	0.61	0.015	2×10^{-11}	0.62	0.021	3×10^{-10}
2.0	0.74	0.029	3×10^{-11}	0.76	0.049	5×10^{-12}	0.77	0.049	3×10^{-12}
5.0	0.92	0.102	4×10^{-14}	0.93	0.104	6×10^{-13}	0.94	0.091	3×10^{-13}
$\beta = 0.5$									
	99,930 nodes; 1,224,637 edges			499,924 nodes; 6,635,776 edges			1,250,850 nodes; 17,264,650 edges		
0.1	0.18	0.001	1×10^{-9}	0.18	0.004	4×10^{-9}	0.18	0.001	1×10^{-9}
0.2	0.23	0.002	5×10^{-10}	0.24	0.008	7×10^{-10}	0.24	0.003	1×10^{-9}
0.5	0.34	0.004	1×10^{-10}	0.35	0.018	4×10^{-10}	0.36	0.026	5×10^{-10}
1.0	0.47	0.018	4×10^{-11}	0.48	0.056	8×10^{-11}	0.49	0.049	6×10^{-11}
2.0	0.63	0.064	2×10^{-11}	0.64	0.073	5×10^{-12}	0.65	0.064	1×10^{-11}
5.0	0.84	0.121	1×10^{-13}	0.86	0.140	1×10^{-12}	0.87	0.130	7×10^{-13}
$\beta = 0.7$									
	139,971 nodes; 1,300,133 edges			699,401 nodes; 6,997,618 edges			1,749,248 nodes; 18,143,039 edges		
0.1	0.11	0.006	1×10^{-9}	0.11	0.002	8×10^{-10}	0.12	0.005	9×10^{-10}
0.2	0.16	0.008	7×10^{-10}	0.17	0.003	4×10^{-10}	0.17	0.007	4×10^{-10}
0.5	0.26	0.017	2×10^{-10}	0.27	0.012	2×10^{-10}	0.28	0.015	2×10^{-10}
1.0	0.38	0.020	1×10^{-10}	0.39	0.048	1×10^{-10}	0.40	0.027	1×10^{-10}
2.0	0.54	0.047	1×10^{-10}	0.55	0.095	1×10^{-10}	0.56	0.158	9×10^{-11}
5.0	0.77	0.097	9×10^{-11}	0.79	0.179	9×10^{-11}	0.80	0.223	9×10^{-11}

Table 5. Performance of our algorithm on three synthetic graphs of different sizes for $\beta = 0.3, 0.5, 0.7$ with $\delta = 1 \times 10^{-6}$.

From the tables, we see that our algorithm performs extremely well on the synthetic graphs and is robust to the various choices of β, the density of the graph. To give an idea, even when 1% more edges are added, the number of nodes in the graph created by the algorithm is only 40% of the original graph, while the error in PageRank is only of the order of 10^{-10}. Moreover, the error in the algorithm is only of the order of 10^{-4} even with δ as high as 10^{-2}; recall that a higher choice of δ means that the graph constructed by the algorithm is smaller, rendering our algorithm more efficient.

6. Conclusions and Future Work

We have developed a method for computing PageRank incrementally when given a collection of link changes. Our method appears to be robust, fast, and accurate.

R (%)	$\frac{\lvert G\rvert}{\lvert W\rvert}$	PR diff.	Our error	$\frac{\lvert G\rvert}{\lvert W\rvert}$	PR diff.	Our error	$\frac{\lvert G\rvert}{\lvert W\rvert}$	PR diff.	Our error
	99,930 nodes; 1,224,637 edges			499,924 nodes; 6,635,776 edges			1,250,850 nodes; 17,264,650 edges		
				$\delta = 1 \times 10^{-4}$					
0.1	0.15	0.001	6×10^{-7}	0.16	0.004	5×10^{-7}	0.16	0.001	2×10^{-7}
0.2	0.22	0.002	3×10^{-7}	0.22	0.008	3×10^{-7}	0.22	0.003	2×10^{-7}
0.5	0.34	0.004	2×10^{-7}	0.35	0.018	1×10^{-7}	0.35	0.026	1×10^{-7}
1.0	0.47	0.018	9×10^{-8}	0.48	0.056	6×10^{-8}	0.49	0.049	4×10^{-8}
2.0	0.62	0.064	3×10^{-8}	0.64	0.073	1×10^{-8}	0.65	0.064	1×10^{-8}
5.0	0.84	0.122	8×10^{-10}	0.86	0.140	9×10^{-10}	0.87	0.130	9×10^{-10}
				$\delta = 1 \times 10^{-2}$					
0.1	0.08	0.001	2×10^{-4}	0.08	0.004	5×10^{-4}	0.08	0.001	1×10^{-4}
0.2	0.14	0.002	2×10^{-4}	0.14	0.008	6×10^{-4}	0.14	0.003	1×10^{-4}
0.5	0.26	0.004	2×10^{-4}	0.26	0.018	5×10^{-4}	0.27	0.026	2×10^{-4}
1.0	0.40	0.018	2×10^{-4}	0.41	0.056	2×10^{-4}	0.42	0.049	2×10^{-4}
2.0	0.58	0.064	2×10^{-4}	0.59	0.073	2×10^{-4}	0.60	0.064	2×10^{-4}
5.0	0.83	0.122	5×10^{-5}	0.85	0.140	4×10^{-5}	0.85	0.130	5×10^{-5}

Table 6. Performance of our algorithm on three synthetic graphs of different sizes for $\delta = 1 \times 10^{-4}, 1 \times 10^{-2}$ with $\beta = 0.5$.

In fact, it is significantly faster than naïve recomputation. Further work includes developing better heuristics to create the small graph G. We believe that better heuristics for this will substantially improve the running time and the quality of approximation. Another direction will be to study the effect of *node* additions and deletions. Recently, Langville and Meyer [Langville and Meyer 02] have obtained a PageRank update algorithm that can handle node additions/deletions; their algorithm is based on the well-known Iterative Aggregation/Disaggregation (IAD) method for nearly completely decomposable Markov chains [Stewart 94].

7. Appendix: Four Additional Proofs of Theorem 2.6

7.1. A Proof Based on Group Inverse

Let I be the $n \times n$ identity matrix and let $A = I - P$. The *group inverse* $A^\sharp$ of A is the matrix satisfying

$$AA^\sharp A = A, \quad A^\sharp AA^\sharp = A^\sharp, \quad AA^\sharp = A^\sharp A.$$

Let $W = I - AA^\sharp$.

Fact 7.1. [Meyer 75, Theorem 2.3] *If P is regular, then W is the $n \times n$ matrix in which* every *row is π^T.*

Fact 7.2. [Meyer 75, Theorem 2.4] *If P is the transition matrix of a regular Markov chain, then*

$$A^\sharp = \sum_{k=0}^{\infty}(P^k - W). \tag{7.1}$$

Note the connection to the fundamental matrix—for the chains under consideration here, $W = B$ and $A^\sharp = Z - B$.

Proposition 7.3.

1. [Cho and Meyer 00] $A^\sharp$ *is diagonally dominant over the columns. In particular, for all i, j,*

   $$a^\sharp_{ji} = a^\sharp_{ii} - m_{ji}\pi_i, \tag{7.2}$$

   where m_{ji} is the mean first passage time from j to i in P. Since the mean first passage times and the stationary probabilities cannot be negative, no entry in a column can exceed the diagonal element.

2. [Cho and Meyer 01] $\forall i, a^\sharp_{ii} > 0$.

Cho and Meyer [Cho and Meyer 00, Proposition 2.1] show that for irreducible P and $\tilde{P} = P + E$,

$$\tilde{\pi} - \pi = \tilde{\pi} E A^\sharp. \tag{7.3}$$

Note that since both P and $\tilde{P}$ are transition matrices, the entries in E_{i*} sum to zero. So if the transition probability from i to j increases, other transition probabilities out of i must decrease. Thus, by definition of E, we have $\sum_a e_{ia} = 0$, $e_{ij} > 0$, and for all $k \neq j$ $e_{ik} \leq 0$, and $\sum_{k \neq j} e_{ik} = -e_{ij}$.

We now analyze Equation 7.3 to prove monotonicity. We do this by determining the signs of the expressions. We have that

$$\begin{aligned}
\tilde{\pi} - \pi &= (\tilde{\pi}_1, \ldots, \tilde{\pi}_n)\begin{pmatrix} 0 & \cdots & 0 \\ E_{i*}A^\sharp_{*1} & \ldots & E_{i*}A^\sharp_{*n} \\ 0 & \cdots & 0 \end{pmatrix} \\
&= (\tilde{\pi}_i E_{i*} A^\sharp_{*1}, \ldots, \tilde{\pi}_i E_{i*} A^\sharp_{*n}),
\end{aligned}$$

and hence for all k,

$$\tilde{\pi}_k - \pi_k = \tilde{\pi}_i E_{i*} A^\sharp_{*k}.$$

As in the proof of Theorem 2.6, since $\tilde{P}$ is regular, we can conclude that $\tilde{\pi}_i > 0$. Hence we need only to show that $E_{i*}A^\sharp_{*j} > 0$. To see this, note that

$$E_{i*}A^\sharp_{*j} = e_{ij}a^\sharp_{jj} + \sum_{k\neq j} e_{ik}a^\sharp_{kj}.$$

By Proposition 7.3, $a^\sharp_{jj}$ is positive and strictly dominates each other $a^\sharp_{kj}$. Recalling that $e_{ij} > 0$, and $e_{ij} = -\sum_{k\neq j} e_{ik}$ finishes the proof.

7.2. A Proof Based on a Theorem of Ipsen and Meyer

An *M-matrix* is any matrix of the form $A = sI - B$, where $s > 0$, $B \geq 0$, and $s \geq \rho(B)$ ($\rho(B)$ is the *spectral radius* of B, $\rho(B) = \max\{|\lambda| \ : \ x^T B = \lambda x, x \neq 0\}$).

We will use the following lemma about nonnegative matrices in $\mathbf{R}^{n\times n}$.

Lemma 7.4. [Berman and Plemmons 94, Lemma 6.2.1] *Nonnegative matrix* $T \in \mathbf{R}^{n\times n}$ *is convergent; that is,* $\rho(T) < 1$, *if and only if* $(I-T)^{-1}$ *exists and*

$$(I-T)^{-1} = \sum_{i=1}^{\infty} T^i.$$

Let P be the transition matrix of a regular n-state Markov chain, and define $A = I - P$. A is a singular M-matrix of rank at most $n-1$ (all the rows are orthogonal to the all ones vector). Let A_j (respectively, P_j) be the principal submatrix of A (respectively, P) obtained by deleting the jth row and column from A (respectively P) so that $A_j = I - P_j$. P_j is convergent[6] and so, by Lemma 7.4, A_j^{-1} exists. Thus, A_j is a nonsingular M-matrix (as noted in [Ipsen and Meyer 94, page 1064]. By [Berman and Plemmons 94, page 137, statement N$_{38}$], A_j is inverse positive, i.e., A_j^{-1} exists and $A_j^{-1} > 0$ (also noted in [Ipsen and Meyer 94]).

Ipsen and Meyer show [Ipsen and Meyer 94, Theorem 4.1]

$$\frac{\pi_j - \tilde{\pi}_j}{\pi_j} = \tilde{\pi}^T E^{(j)} A_j^{-1} e, \tag{7.4}$$

where $E^{(j)}$ denotes the matrix obtained by deleting the jth column of E and e is the column vector of all 1s. Now, $E^{(j)}$ contains no positive entries, while it does

[6] See, for instance, [Berman and Plemmons 94, Lemma 8.3.20] for a proof. This is intuitive, however, since P_j represents transitions to all states but j (think of a chain with a single absorbing state, otherwise completely connected); every walk eventually ends up in the absorbing state, so $\lim_k (P_j)^k = 0$.

contain some negative entries (transition probability from i to some state(s) k must have been reduced in order for the transition probability from i to j to have been increased). Everything else on the right-hand side of Equation 7.4 is nonnegative, so the entire right-hand side is nonpositive. Thus the expression on the left is nonpositive, whence π_j, the old steady state probability of state j, cannot exceed $\tilde{\pi}_j$, that is, $\tilde{\pi}_j \geq \pi_j$. Moreover, if $\tilde{P}$ is regular, so that $\forall k, \tilde{\pi}_k \neq 0$, then the right-hand side is strictly negative, so $\tilde{\pi}_j > \pi_j$.

7.3. Roughgarden's Proof

This proof was communicated to us by Tim Roughgarden.

Write $\pi_j = 1/E[T_{jj}^P]$, where T_{jj}^P is the return time of a random walk started at node j of the chain with transition matrix P. Write

$$\begin{aligned} E[T_{jj}^P] &= \sum_{k\geq 1} k \cdot \Pr_P[\text{first return in exactly } k \text{ steps}] \\ &= \sum_{k\geq 1} \Pr_P[\text{first return in} \geq k \text{ steps}] \\ &= \sum_{k\geq 1} \Pr_P[\text{fail to return in first } k-1 \text{ steps}]. \qquad (7.5) \end{aligned}$$

Similarly,

$$E[T_{jj}^{\tilde{P}}] = \sum_{k\geq 1} \Pr_{\tilde{P}}[\text{fail to return in first } k-1 \text{ steps}]. \qquad (7.6)$$

The claim is that $\pi_j \leq \tilde{\pi}_j$, which is equivalent to $E[T_{jj}^P] \geq E[T_{jj}^{\tilde{P}}]$. The derivation above shows that to prove this it suffices to show that the probability of failing to return to j in a random walk in P in the first $k-1$ steps is at least as great as the probability of failing in a random walk in $\tilde{P}$ to return in the first $k-1$ steps, for each k.

Let S (respectively, $\tilde{S}$) denote the walks in P (respectively, $\tilde{P}$) that start at j, go exactly $k-1$ steps, and fail to return to j. Let us conceptually view the increase in probability of the (i, j) transition as the addition of a new edge with the extra transition probability. The key observation is that S and $\tilde{S}$ are the same set, and contain no walks that include the new (i, j) edge (any walk that uses the new edge in $\tilde{P}$ necessarily returns to j, and so cannot be in $\tilde{S}$).

The probabilities in Equations 7.5 and 7.6 are just sums over the walks in S and $\tilde{S}$, respectively, of the probability of walking a given walk. For example, the probability of failing to return to j in a random walk in P in the first $k-1$ steps is the sum over all walks $w \in S$ of the probability of taking walk w. Note that for any $w \in S$ (equivalently, $w \in \tilde{S}$, since the sets are the same), $\Pr_S[w] \geq \Pr_{\tilde{S}}[w]$:

adding the new edge (i, j) decreases the probability of other edges out of i and leaves other relevant probabilities unaffected. Since the sums in the expressions for $E[T^P_{jj}]$ and $E[T^{\tilde{P}}_{jj}]$ are over the same index set of ws, and for each such w, $\Pr_S[w] \geq \Pr_{\tilde{S}}[w]$, the proof is complete.

7.4. A Proof Based on Lemma 2.10

This proof was sent to us by an anonymous referee.

Recall that by Lemma 2.10, $\forall a, b$,

$$\pi_a = \pi_b \frac{\text{flow}(b, a)}{\text{flow}(a, b)},$$

and by Lemma 2.12, $\forall b$, $\text{flow}_P(j, b) \geq \text{flow}_{\tilde{P}}(j, b)$ and $\text{flow}_P(b, j) \leq \text{flow}_{\tilde{P}}(b, j)$, so

$$\frac{\text{flow}_{\tilde{P}}(b, j)}{\text{flow}_{\tilde{P}}(j, b)} \geq \frac{\text{flow}_P(b, j)}{\text{flow}_P(j, b)} .$$

Since $\sum_b \tilde{\pi}_b = 1$, we have

$$\pi_j = \sum_b \tilde{\pi}_b \pi_j = \sum_b \tilde{\pi}_b \left(\pi_b \frac{\text{flow}_P(b, j)}{\text{flow}_P(j, b)} \right),$$

and since $\sum_b \pi_b = 1$, we have

$$\tilde{\pi}_j = \sum_b \pi_b \tilde{\pi}_b \frac{\text{flow}_{\tilde{P}}(b, j)}{\text{flow}_{\tilde{P}}(j, b)} = \sum_b \tilde{\pi}_b \pi_b \frac{\text{flow}_{\tilde{P}}(b, j)}{\text{flow}_{\tilde{P}}(j, b)}.$$

Thus,

$$\tilde{\pi}_j = \sum_b \tilde{\pi}_b \pi_b \left(\frac{\text{flow}_{\tilde{P}}(b, j)}{\text{flow}_{\tilde{P}}(j, b)} \right) \geq \sum_b \tilde{\pi}_b \pi_b \left(\frac{\text{flow}_P(b, j)}{\text{flow}_P(j, b)} \right) = \pi_j,$$

which proves the theorem.

Acknowledgments. We thank Compaq SRC and especially Janet Wiener for their support. We thank Michael Mitzenmacher, Moni Naor, Tim Roughgarden, and Salil Vadhan for helpful discussions. We are grateful to Jerry Kazdan both for his enthusiasm and for simplifying the calculations in our first monotonicity proof. Some of the first and second authors' work was performed while they were visiting the Compaq Systems Research Center. The first author was supported by NSF grants CCR-9820951 and CCR-0121555.

References

[Berman and Plemmons 94] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Philadelphia: SIAM, 1994

[Brin and Page 98] S. Brin and L. Page. "The Anatomy of a Large-Scale Hypertextual (Web) Search Engine." *Computer Networks* 30:1–7 (1998) 107–117.

[Broder et al. 00] A. Broder, F. Maghoul, R. Kumar, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. "Graph Structure in the Web." *Computer Networks* 33:1–6 (2000), 309–320.

[Cho and Meyer 01] G. Cho and C. D. Meyer. "Comparison of Perturbation Bounds for the Stationary Distribution of a Markov Chain." *Linear Algebra and Its Applications* 335:1–3 (2001), 137–150.

[Cho and Meyer 00] G. Cho and C. D. Meyer. "Markov Chain Sensitivity Measured by Mean First Passage Times." *Linear Algebra and Its Applications* 316 (2000), 21–28.

[Dill et al. 02] S. Dill, R. Kumar, K. McCurley, S. Rajagopalan, D. Sivakumar, and A. Tomkins. "Self-Similarity in the Web." *ACM Trans. Internet Techn.* 2:3 (2002), 205–223.

[Heydon and Najork 99] A. Heydon and M. Najork. "Mercator: A Scalable, Extensible Web Crawler." *World Wide Web* 2:4 (1999), 219–229.

[Ipsen and Meyer 94] I. Ipsen and C. Meyer. "Uniform Stability of Markov Chains." *SIAM Journal on Matrix Analysis and Applications* 15:4 (1994), 1061–1074.

[Kemeny and Snell 60] J. Kemeny and J. L. Snell. *Finite Markov Chains*, New York: D. Van Nostrand Company, Inc., 1960.

[Kleinberg 99] J. Kleinberg. "Authoritative Source in a Hyperlinked Environment." *J. ACM* 46:5 (1999), 604–632.

[Kumar et al. 00] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. "Stochastic Models for the Web Graph." In *Proc. 41st IEEE Symposium on Foundations of Computer Science*, pp. 57–65. Los Alamitos, CA: IEEE Computer Society, 2000.

[Langville and Meyer 02] A. N. Langville and C. D. Meyer. "Updating PageRank Using the Group Inverse and Stochastic Complementation." Technical Report, *NCSU CRSC #CRSC-TR02-32*, 2002.

[McCallum et al. 00] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. "Automating the Construction of Internet Portals with Machine Learning." *Information Retrieval Journal* 3 (2000), 127–163.

[Meyer 75] C. D. Meyer, Jr. "The Role of the Group Generalized Inverse in the Theory of Finite Markov Chains." *SIAM Review* 17:3 (1975), 443–464.

[Ng et al. 01a] A. Y. Ng, A. X. Zheng, and M. I. Jordan. "Stable Algorithms for Link Analysis." In *Proc. 24th International Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 258–266. New York: ACM Press, 2001.

[Ng et al. 01b] A. Y. Ng, A. X. Zheng, and M. I. Jordan. “Link Analysis, Eigenvectors, and Stability.” In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 903–910. San Francisco, CA: Morgan Kaufmann, 2001.

[Randall et al. 01] K. H. Randall, R. Stata, R. G. Wickremesinghe, and J. L. Wiener. “The Link Database: Fast Access to Graphs of the Web.” Research Report 175, Compaq Systems Research Center, Palo Alto, CA, 2001.

[Schweitzer 68] P. J. Schweitzer. “Perturbation Theory and Finite Markov Chains.” *Journal of Applied Probability* 5:3 (1968), 401–404.

[Stewart 94] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton, NJ: Princeton University Press, 1994.

Steve Chien, University of California at Berkeley, Berkeley, CA 94720
(schien@cs.berkeley.edu)

Cynthia Dwork, Microsoft Research–Silicon Valley, 1065 L’Avenida, Mountain View, CA 94043 (dwork@microsoft.com)

Ravi Kumar, IBM Almaden Research Center, 650 Harry Rd., San Jose, CA 95120
(ravi@almaden.ibm.com)

Daniel R. Simon, Microsoft Research, One Microsoft Way, Redmond, WA 98052
(dansimon@microsoft.com)

D. Sivakumar, IBM Almaden Research Center, 650 Harry Rd., San Jose, CA 95120
(siva@almaden.ibm.com)

Received June 17, 2003; accepted December 2, 2003.