Internet Mathematics Vol. 6, No. 2: 189–236

Random Alpha PageRank

Paul G. Constantine and David F. Gleich

Abstract. We suggest a revision to the PageRank random surfer model that considers the influence of a population of random surfers on the PageRank vector. In the revised model, each member of the population has its own teleportation parameter chosen from a probability distribution, and consequently, the ranking vector is random. We propose three algorithms for computing the statistics of the random ranking vector based respectively on (i) random sampling, (ii) paths along the links of the underlying graph, and (iii) quadrature formulas. We find that the expectation of the random ranking vector produces similar rankings to its deterministic analogue, but the standard deviation gives uncorrelated information (under a Kendall-tau metric) with myriad potential uses. We examine applications of this model to web spam.

1. Introduction

The PageRank modification for a Markov chain transforms any input Markov chain into an irreducible, aperiodic chain with a unique stationary distribution. Elements of this unique stationary distribution give the importance of the nodes in the state space of the input Markov chain. Brin and Page proposed the PageRank method to measure the *global* importance of web pages under the behavior of a *random surfer*, which can be interpreted as a Markov chain on the web graph [Page et al. 99]. We focus on this random surfer model and show that it contains a slight oversight when interpreted over a set of "surfers."

Let us begin by revisiting the putative random surfer. With probability α, the surfer follows the links of a web page uniformly at random. With probability

1542-7951/08 $0.50 per page

$1 - \alpha$, the surfer jumps to a different page according to a given probability distribution over web pages. (For the moment, we are assuming that dangling-node pages—those with no links—have been addressed in some fashion; see Section 2.)

Thus, the PageRank value for a web graph depends on two quantities: the parameter α and the given distribution over the pages. The effect of both of these quantities on the mathematics of the PageRank vector are fairly well understood [Langville and Meyer 06, Boldi et al. 09], but the choice of α is not well justified in the context of the random surfer model. Most existing PageRank calculations use a single value of α, and two choices stand out in the literature for web search: $\alpha = 0.5$ [Avrachenkov et al. 07, Chen et al. 07] and $\alpha = 0.85$ [Page et al. 99, Najork et al. 07]. Some other choices are discussed in Section 4.1 as well.

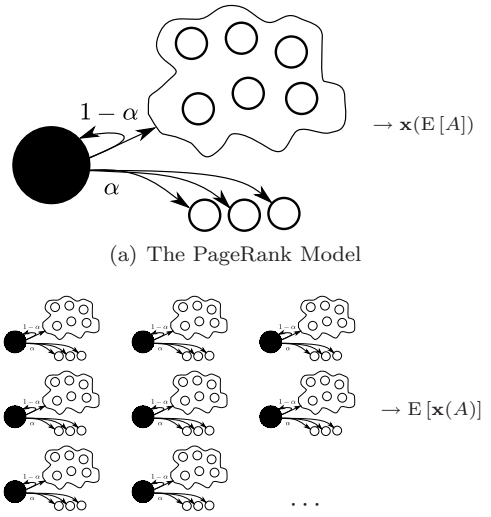
Rather than trying and testing arbitrary values of α, suppose we pick α to make the random surfer model accurate. Because α ought to be the probability of following a link on a web page, let us make it so.

Empirically measured browsing patterns on the web show that individual users, unsurprisingly, have different browsing behaviors [Huberman et al. 98, White and Drucker 07]. We also confirm this result in a recent paper [Gleich et al. 10a]. If we assume that all users have their own probability α_i of teleporting (i.e., all users follow links with different probabilities), then the random surfer model suggests that we should set $\alpha = \frac{1}{N}\sum_{i=1}^{N} \alpha_i$, i.e., the mean of these values.

More generally, if A is a random variable with a density function encoding the distribution of teleportation parameters among multiple (perhaps infinite) surfers, then the PageRank model suggests $\alpha = \mathrm{E}[A]$, where $\mathrm{E}[\cdot]$ is the expectation operator.

The flaw in PageRank is that using $\alpha = \mathrm{E}[A]$ still does not yield a PageRank vector that reflects all the surfer values α_i. We will justify this statement shortly; intuitively it arises because a single value of α condenses all surfers into a single über-surfer. Instead, we propose to give a small vote to the PageRank vector $\mathbf{x}(\alpha_i)$ corresponding to each random surfer and create a global metric that amalgamates this information. In other words, we want to examine the random surfer model with "$\alpha = A$," where A is a random variable modeling users' individual behaviors. Figure 1 gives a pictorial view of this change. If A is a random variable, then the PageRank vector $\mathbf{x}(A)$ is a random vector, and we can synthesize a new ranking measure from its statistics. We call this measure Random Alpha PageRank (RAPr); it is pronounced "wrapper."

In earlier work [Constantine and Gleich 07], we introduced a means of handling multiple surfers in PageRank. This manuscript extends those ideas by clarifying the presentation, expanding the computational algorithms, and compiling additional results. In particular, the previous paper used the polynomial chaos approach to investigate the behavior of multiple surfers algorithmically.



(a) The PageRank Model

(b) Our random α PageRank model

Figure 1. The PageRank model assumes a single random surfer representing an expected user. Our model assumes that each surfer is unique with a different value of α, which we represent as a random variable A. If the function $\mathbf{x}(\cdot)$ gives the PageRank vector for a deterministic or random α or A, respectively, we then compute the expected PageRank given the distribution for A.

In [Constantine et al., to appear], we showed that the polynomial chaos and quadrature methods are equivalent in the case of PageRank. The presentation here eliminates the discussion of polynomial chaos beyond this paragraph.

In what follows, we first present a short review of PageRank along with the notation we adopt in this paper (Section 2). We continue the discussion by outlining our high-level vision for how RAPr might be useful in a variety of situations (Section 3). Relationships between RAPr and other literature are discussed next (Section 4). In the following section, we present a theoretical comparison between PageRank and RAPr and how RAPr generalizes a few results about PageRank (Section 5). The following two sections present algorithms to approximate two important quantities in the RAPr model (Section 6) and analyze the convergence of each of these algorithms both theoretically and empirically (Section 7). Finally, we present a few applications of RAPr and show that it helps to improve a webspam classifier (Section 9).

All of our experimental code, which we provide to allow others to validate and repeat our experiments, is available online.[1]

[1]Available at http://stanford.edu/~dgleich/publications/2009/rapr.

In the exposition that follows, we also present our algorithms with MATLAB code instead of traditional pseudocode. While producing compact code introduces a few small restrictions and inefficiencies in the implementation, we believe that the advantage of cut-and-pastability of these implementations is considerable.

2. PageRank and Notation

In this section, we detail our choices for the PageRank model employed in the remainder of the paper. Given the row-oriented adjacency matrix of a web graph

$$W_{i,j} = \begin{cases} 1 & \text{page } i \text{ links to page } j, \\ 0 & \text{otherwise,} \end{cases}$$

let $\mathbf{P}$ be a *column*-stochastic transition matrix for $\mathbf{W}$,

$$P_{j,i} = \text{ probability of transitioning from node } i \text{ to node } j,$$

so that $\mathbf{e}^T\mathbf{P} = \mathbf{e}^T$. (Please note that we have transposed $\mathbf{P}$ from the row-oriented description of $\mathbf{W}$.) This transformation assumes that dangling nodes have been corrected in some manner.

For the experiments in this paper, we use the *strongly preferential* PageRank model [Boldi et al. 07], although we note that the analysis and algorithms apply to any column-stochastic matrix $\mathbf{P}$. We discuss only the strongly preferential framework because it seems to be the most common. Finally, let $(1-\alpha)$ be the teleportation probability and $\mathbf{v}$ the teleportation or personalization distribution. The PageRank model requires $0 \leq \alpha \leq 1$ and $v_i \geq 0$. Under these definitions, then, the PageRank vector $\mathbf{x}(\alpha)$ satisfies the linear system [Arasu et al. 02, Del Corso et al. 05]

$$(\mathbf{I} - \alpha\mathbf{P})\mathbf{x}(\alpha) = (1-\alpha)\mathbf{v}. \tag{2.1}$$

When $\alpha = 1$, then we define $\mathbf{x}(1)$ using the limiting value [Serra-Capizzano 05].

We summarize a subset of our notation for this paper, which we use consistently unless otherwise noted, in Table 1. All vectors are column oriented. Matrices and vectors are denoted by bold letters. Random variables are denoted by uppercase, nonsubscripted, roman letters. Only A, the random α, is used frequently. We use only the 1-norm, so $\|\mathbf{x}\| = \|\mathbf{x}\|_1$ throughout.

Given a continuous random variable A with a continuous density function $\rho(x)$ on the interval $[l, r]$ (such variables are quite special, but they suffice for this paper), we define the expectation of A:

$$\mathrm{E}[A] = \int_l^r \rho(x)\, dx.$$

Symbol	Meaning
A	a random variable for the teleportation parameter
α	a deterministic value for the teleporatation parameter
$\mathrm{Beta}(a, b, [l, r])$	the beta distribution with parameters a, b over $[l, r]$
$\mathrm{Beta}(\cdot, \cdot)$	the beta function of two parameters
$\mathrm{E}[\cdot]$	the expectation operator
$\bullet$	the Hadamard (or elementwise) product
$\mathbf{P}$	a column-stochastic matrix
$\|\cdot\|$	the one-norm ($\|\cdot\|_1$)
$\mathrm{Std}[\cdot]$	the standard deviation operator
$\mathbf{v}$	the teleportation distribution vector for PageRank
$\mathbf{x}(A)$	solution to the RAPr model
$\mathbf{x}(\alpha)$	the PageRank vector for teleportation parameter α
$x_i(\cdot)$	the ith element of $\mathbf{x}(\cdot)$

Table 1. Throughout this paper, we maintain the notation that a matrix is denoted by a bold, uppercase, roman letter; a vector by a bold, lowercase, roman letter; a random parameter by an uppercase letter; and the elements of a matrix or vector by nonbolded letters with subscripts.

Evaluating the expectation of a function corresponds to

$$\mathrm{E}[f(A)] = \int_l^r f(x)\rho(x)\,dx.$$

The standard deviation operator is defined in terms of the expectation operator:

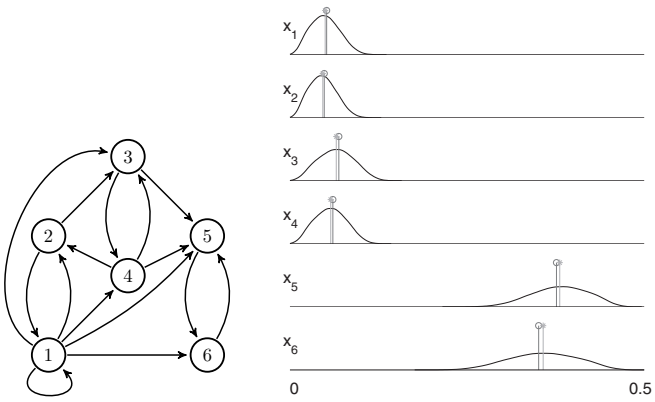
$$\mathrm{Std}[A] = \sqrt{\mathrm{E}[(A - \mathrm{E}[A])^2]}.$$

3. Vision

Let us return to the RAPr model and outline our vision for the new model before proceeding to discuss theoretical properties and algorithms. The PageRank vector $\mathbf{x}(\mathrm{E}[A])$ does not incorporate the surfing behavior of all users; we propose to use $\mathrm{E}[\mathbf{x}(A)]$ instead. Because the PageRank vector is a nonlinear function of α, we have $\mathrm{E}[\mathbf{x}(A)] \neq \mathbf{x}(\mathrm{E}[A])$, and Section 5 gives a formal counterexample. For reasonable distributions of A, however, we expect

$$\mathbf{x}(\mathrm{E}[A]) \approx \mathrm{E}[\mathbf{x}(A)].$$

We explore this behavior in Section 9. Despite this similarity, moving from the deterministic $\mathbf{x}(\alpha)$ to the random $\mathbf{x}(A)$ yields more information. For a given



(a) A small web graph (b) Density functions

Figure 2. (a) A simple web graph and (b) approximate probability density functions of the corresponding PageRank random variables. This shows that pages 5 and 6 have the highest variance (widest density function). These pages are traps from which the random surfer cannot leave. In this plot, $A \sim \text{Beta}(2, 16, [0, 1])$. In (b), the dark gray circle stems show the PageRank value for $\alpha = \text{E}[A] = 0.85$, whereas the light gray star stems show the expectation according to the PageRank density.

page, its "PageRank" is now a random variable. Figure 2 shows the probability density functions for the PageRank random variables on a small graph.

We can use the *standard deviation* of the random variables to help "quantify the uncertainty" in the PageRank value. The standard deviation is a measure of the variability in the PageRank induced by the variability in A. For the graph in Figure 2, the standard deviation vector is

$$\text{Std}[\mathbf{x}(A)] = \begin{bmatrix} 0.021 & 0.020 & 0.026 & 0.023 & 0.041 & 0.049 \end{bmatrix}^T.$$

This vector shows that x_5 and x_6 have the highest standard deviation. In a traditional PageRank context, these pages both are in a sink-component and accumulate rank from the largest connected component (x_1, x_2, x_3, and x_4).

We anticipate many problem-dependent uses for RAPr-derived quantities. For example, we outline the use of correlation coefficients in Section 4.5. Our experiments show that the standard deviation vector is uncorrelated (in a Kendall-τ sense) with the PageRank vector itself. (Kendall's τ measures the difference in *concordant* and *discordant* pairs between two lists relative to an identical ordering and an inverted ordering.) Because pages with a high standard deviation have highly variable PageRank values, the standard deviation vector could be an important input to a machine learning framework for web search or web page categorization. We evaluate this use in Section 9.4.

More generally, the PageRank model has become a key tool for network and graph analysis. It has been used to find graph cuts [Andersen et al. 06], infer missing values on a partially labeled graph [Zhou et al. 05], find interesting genes [Morrison et al. 05], and help match graph structures in protein networks [Singh et al. 07]. In all of these cases, the random surfer model does not directly apply. Each paper picks a particular value for α and computes a PageRank vector from that value. With RAPr, each case will have a natural random variable. For most, it may be a uniform distribution. Rather than reporting just a single number, the algorithms could use the standard deviation as a confidence measure representing uncertainty or global sensitivity in the resulting PageRank vector. The sensitivity using the standard deviation accounts for fluctuations in the function over a wider interval than the derivative (which we discuss further in Section 4.4).

4. Related Work

Our ideas have strong relationships with a few other classes of literature. Before delving into the details of the RAPr model, we would like to discuss these relationships.

4.1. Teleportation Parameters in the Literature

Algorithmic papers on PageRank tend to investigate the behavior of PageRank algorithms for multiple values of α or values of α larger than 0.85 [Kamvar et al. 03, Golub and Greif 06], whereas evaluations of the PageRank vector tend to use the canonical value $\alpha = 0.85$ [Najork et al. 07].

Katz used $\alpha = 0.5$ in a model related to PageRank [Katz 53]. Katz's model was $(\mathbf{I} - \alpha\mathbf{W}^T)\mathbf{k} = \alpha\mathbf{W}^T\mathbf{e}$ for an adjacency matrix $\mathbf{W}$. More recently, two papers suggested $\alpha = 0.5$ for PageRank. The first paper [Avrachenkov et al. 07] argues that $\alpha = 0.5$ is the right choice in using graph-theoretic techniques to examine the mass of PageRank in the largest strong component of the web graph. The second paper [Chen et al. 07] applies the random surfer model to a graph of literature citations. The authors claim, based on co-citation analysis, that literature networks contain very short citation paths of average length 2. This analysis then suggests $\alpha = 0.5$.

4.2. Usage Logs and Behavior Analysis

Huberman et al. studied the behavior of web surfers even before the original paper on PageRank, and they suggested a Markov model for surfing behavior

[Huberman et al. 98]. In contrast with the Brin and Page random surfer, the authors empirically measured the probability that surfers follow paths of length ℓ and then computed

$$\mathbf{n}_\ell = f_\ell \mathbf{P}^\ell \mathbf{n_0}$$

for the expected number of surfers on each page after ℓ transitions. They found that f_ℓ, the probability of following a path of length ℓ, is approximately an inverse Gaussian.

This model is strongly related to the path-damping models discussed next and in Section 5.3. An earlier study showed that the average path length of users visiting a site decayed quickly, but did not match the decay to a distribution [Catledge and Pitkow 95]. Both of these studies focused on the browsing behavior at a single site and not across the web in general. Subsequently, many papers have suggested measuring surfer behavior from usage logs to improve local site search [Wang 02, Xue et al. 03, Farahat et al. 06].

In the context of web search, a recent study identified two types of surfers: navigators and explorers [White and Drucker 07]. Navigators proceed roughly linearly, whereas explorers frequently branch their browsing patterns and revisit previous pages before going to new links. The former behavior corresponds to a larger value of α than the latter.

A recent patent from Yahoo! [Berkin et al. 08] describes a modification of the PageRank equations to build a "user-sensitive PageRank" system by incorporating observed page transitions and user segment modeling. The key idea in the patent is to modify the Markov chain transition probabilities to give higher weight to transitions observed and change the teleportation vector in light of the start points of observed transitions. These weights depend on a user segment. They also recognize the inaccuracy of a single teleportation coefficient, but model separate teleportation coefficients to and from each page on the web. Our approach differs by modeling a random Markov chain and its associated random stationary distribution. Researchers at Microsoft proposed a related system called BrowseRank that builds a continuous-time Markov chain from observed user transitions and holding times [Liu et al. 08].

We recently completed an investigation of empirical values of α [Gleich et al. 10a] and found that user values of α approximate a beta density with mean between 0.3 and 0.7 depending on the website.

4.3. Path Damping

While working on the mathematics of RAPr, we discovered a strong relationship with path-damping interpretations of the PageRank vector. Path-damping models weight each path of length ℓ in the graph with a set of coefficients that

sum to 1. Mathematically, they compute a ranking vector

$$\mathbf{r} = \sum_{\ell=0}^{\infty} \omega(\ell)\mathbf{P}^{\ell}\mathbf{v},$$

where $\sum_{\ell=0}^{\infty} \omega(\ell) = 1$ [Boldi 05, Baeza-Yates et al. 06]. As we show in Section 5.3, the value $\mathrm{E}[\mathbf{x}(A)]$ corresponds to a particular choice of $\omega(\ell)$.

4.4. The Derivative

In the vision section, we briefly alluded to the standard deviation as a sensitivity measure. A more local sensitivity measure for PageRank is the derivative with respect to α [Boldi et al. 05, Golub and Greif 06]. Whereas the derivative evaluates the sensitivity to small changes in α around a set value, the standard deviation reflects the stability of the PageRank value over the range of values for the random variable A. Thus, the derivative captures local sensitivity information about the PageRank function of α, and the standard deviation captures a more global sensitivity.

4.5. Spam Ranking

Zhang et al. use the PageRank vector at different values of α to infer spam pages [Zhang et al. 04]. Spam pages, the authors argue, ought to be sensitive to changes in α. Their goal is to trap the surfer and boost their rank. Thus, changing α will reveal them. After computing PageRank at a few α's, the authors measure the correlation between the function $1/(1-\alpha)$ and the PageRank value (for a page) on their small set of α's. Pages with high correlation are more likely to be spam pages. This idea is similar in practice to the Gauss quadrature algorithm of Section 6.4 when used to compute correlation coefficients between RAPr values. In RAPr, the random variable A has an associated quadrature rule for its expectation that specifies the α's at which to compute the PageRank vector. RAPr is also more general. It is not tied to just computing a correlation against a particular function but produces a correlation score between any pages. Measuring the correlation coefficient against known spam pages is yet another possible use of the RAPr model.

5. The Random Alpha PageRank Model

So far, we have discussed our vision for RAPr and the body of literature that surrounds our ideas. We now formally state and analyze RAPr.

Given a bounded random variable A distributed within the interval $[0, 1]$, the random alpha PageRank is the vector $\mathbf{x}(A)$ that satisfies

$$(\mathbf{I} - A\mathbf{P})\mathbf{x}(A) = (1 - A)\mathbf{v}, \tag{5.1}$$

where $\mathbf{I}$, $\mathbf{P}$, and $\mathbf{v}$ are as in (2.1). When we use this model, we often look at

$$\mathrm{E}\left[\mathbf{x}(A)\right] \quad \text{and} \quad \mathrm{Std}\left[\mathbf{x}(A)\right].$$

We address some theoretical implications of this model in the next few sections, and we defer the discussion of computation until Section 6.

From this definition, we can immediately show that our model generalizes the TotalRank algorithm [Boldi 05], which produces a vector $\mathbf{t}$ defined as

$$\mathbf{t} = \int_0^1 \mathbf{x}(\alpha)\, d\alpha.$$

If $A \sim U[0, 1]$ in RAPr (i.e., A is distributed according to a uniform $[0, 1]$ density), then

$$\mathrm{E}\left[\mathbf{x}(A)\right] = \int_0^1 \mathbf{x}(\alpha)\, d\alpha = \mathbf{t}.$$

A purported benefit of the TotalRank algorithm is that it eliminates picking an α in a PageRank computation. When compared to RAPr, however, it corresponds to a particular choice of the random variable A.

It behooves us to check that the expectation of RAPr is well defined. One concern is that $\mathrm{E}\left[\mathbf{x}(A)\right] = \int_0^1 \mathbf{x}(\alpha)\rho(\alpha)\, d\alpha$ touches the value $\mathbf{x}(1)$. Looking only at the linear system $(\mathbf{I} - \alpha\mathbf{P})\mathbf{x} = (1 - \alpha)\mathbf{v}$, we would conclude that $\mathbf{x}(1)$ may not be unique because the matrix is singular when $\alpha = 1$. If $\mathbf{x}(1)$ is not defined, then the expectation of RAPr will not exist. In fact, there is no difficulty for our formulation of PageRank because $\alpha = 1$ corresponds to a removable singularity of the function. Thus, we can extend the definition of $\mathbf{x}$ to $\alpha = 1$ with the limiting value.

5.1. Choice of Distribution

The first order of business for RAPr is to choose the distribution of A. While choosing a distribution seems more difficult than picking a single value α, having the right data makes it easy. The information for the empirical distribution of A is present in the logs from the surfer behavior studies discussed in Section 4.2. This point is illustrated in [Gleich et al. 10a], where we take browsing logs and compute a distribution for α.

Picking A based on browsing behavior, however, is yet another choice. It seems correct and natural for the random surfer derivation of PageRank. When the

PageRank or RAPr values are used in an application, the metrics of the application should drive the choice of α or A. We return to this point in Section 9.4.

We assume that A has a continuous distribution over $[l, r]$ with $0 \le l < r \le 1$. Two distributions with bounded, continuous support are the uniform distribution and the beta distribution. In fact, the uniform distribution is a special case of the beta distribution, and consequently, our "default" choice of A is a beta random variable with distribution parameters a and b, and support $[l, r]$. To denote this, we write $A \sim \text{Beta}(a, b, [l, r])$. The probability density function for this random variable is

$$\rho(x) = \frac{1}{(r-l)^{a+b+1}} \frac{(x-l)^b (r-x)^a}{\text{Beta}(a+1, b+1)}. \tag{5.2}$$

It reduces to a uniform distribution when $a = b = 0$. Later, we will derive our algorithms in the most general settings possible, but all computations are done with some version of the beta distribution. We find that empirical distributions of α are strikingly close to a beta with properly fitted parameters [Gleich et al. 10a].

5.2. Theoretical Properties

We continue to discuss the RAPr model by proving a few results that show how RAPr generalizes PageRank. All of the following theory reduces to known results about PageRank when A is a constant. We begin to discuss the theoretical properties of RAPr by computing a more tractable expression for the expectation of our random PageRanks. In the following theorems, we always assume that A is a beta random variable. While this assumption can be relaxed to a broader class for the results, it suffices for this paper.

Theorem 5.1. *If $A \sim \text{Beta}(a, b, [l, r])$ with $0 \le l < r \le 1$, then*

$$\text{E}\left[\mathbf{x}(A)\right] = \sum_{\ell=0}^{\infty} \text{E}\left[A^\ell - A^{\ell+1}\right] \mathbf{P}^\ell \mathbf{v}. \tag{5.3}$$

Proof. From (5.1) we have

$$\mathbf{x}(A) = (1-A)(\mathbf{I} - A\mathbf{P})^{-1}\mathbf{v}.$$

Because the spectral radius $\rho(A\mathbf{P})$ is less than 1 for any value of A in $[0, 1)$, we can expand $(\mathbf{I} - A\mathbf{P})^{-1}$ with its Neumann series [Meyer 00, p. 618]:

$$\mathbf{x}(A) = (1-A) \sum_{n=0}^{\infty} A^n \mathbf{P}^n \mathbf{v}.$$

Taking the expectation and rearranging gives

$$\mathrm{E}\left[\mathbf{x}(A)\right] = \mathrm{E}\left[\sum_{n=0}^{\infty}(A^n - A^{n+1})\mathbf{P}^n\mathbf{v}\right].$$

To interchange the expectation and sum, note that $0 \leq A^n\mathbf{P}^n\mathbf{v} \leq 1$ for all n. Because the summands are nonnegative, $(1-A)A^n \geq 0$, Fubini's theorem justifies this interchange. □

The previous theorem also holds when A is a constant between 0 and 1. Using this theorem, we can formally justify the claim that the expectation of RAPr is different from the PageRank vector computed with $\alpha = \mathrm{E}\left[A\right]$. The following pedagogic example restricts the claim to the case in which $A \sim \mathrm{Beta}(0, 0, [0, 1])$ (A is drawn from a uniform $[0, 1]$ distribution). Such a restriction allows us to use the expressions for the moments of A and compute the infinite sums exactly. Note that $\sum_{n=0}^{\infty}\mathrm{E}\left[A^n - A^{n+1}\right] = 1$ because the sums telescope.

Example 5.2. Set $\mathbf{P} = \begin{bmatrix} 0 & 0 & 0 \\ 1/2 & 0 & 0 \\ 1/2 & 1 & 1 \end{bmatrix}$, $\mathbf{v} = \begin{bmatrix} 1/3 & 1/3 & 1/3 \end{bmatrix}^T$. Then

$$\mathbf{P}^0\mathbf{v} = \mathbf{v}, \quad \mathbf{P}^1\mathbf{v} = \begin{bmatrix} 0 & 1/6 & 5/6 \end{bmatrix}^T, \quad \mathbf{P}^n\mathbf{v} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T, \quad n \geq 2.$$

To apply Theorem 5.1, we need $\mathrm{E}\left[A^n\right]$. If $A \sim \mathrm{Beta}(0, 0, [0, 1])$, then A is uniform over $[0, 1]$ and $\mathrm{E}\left[A^n\right] = \frac{1}{n+1}$. Finally,

$$\begin{aligned}\mathrm{E}\left[\mathbf{x}(A)\right] &= \frac{1}{2}\mathbf{v} + \frac{1}{2}\begin{bmatrix} 0 & 1/6 & 5/6 \end{bmatrix}^T + \sum_{n=2}^{\infty}\left(\frac{1}{n+1} - \frac{1}{n+2}\right)\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \\ &= \begin{bmatrix} 1/6 & 7/36 & 23/36 \end{bmatrix}^T.\end{aligned}$$

For $\mathbf{x}(\mathrm{E}\left[A\right]) = \mathbf{x}(1/2)$, we obtain

$$\begin{aligned}\mathbf{x}(\mathrm{E}\left[A\right]) &= \frac{1}{2}\mathbf{v} + \frac{1}{4}\begin{bmatrix} 0 & 1/6 & 5/6 \end{bmatrix}^T + \sum_{n=2}^{\infty}\left(\frac{1}{2^n} - \frac{1}{2^{n+1}}\right)\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \\ &= \begin{bmatrix} 1/6 & 5/24 & 5/8 \end{bmatrix}^T.\end{aligned}$$

Thus for this example, $\mathrm{E}\left[\mathbf{x}(A)\right] \neq \mathbf{x}(\mathrm{E}\left[A\right])$.

For this case, the RAPr solution satisfies $\mathbf{e}^T\begin{bmatrix} 1/6 & 7/36 & 23/36 \end{bmatrix} = 1$. This property is general, and we next show that—like PageRank—the vector $\mathrm{E}\left[\mathbf{x}(A)\right]$ is always a probability distribution.

Corollary 5.3. *If $A \sim \mathrm{Beta}(a, b, [l, r])$ with $0 \leq l < r \leq 1$ and probability density function ρ, then $\mathrm{E}\left[x_i(A)\right] \geq 0$ and $\|\mathrm{E}\left[\mathbf{x}(A)\right]\| = 1$.*

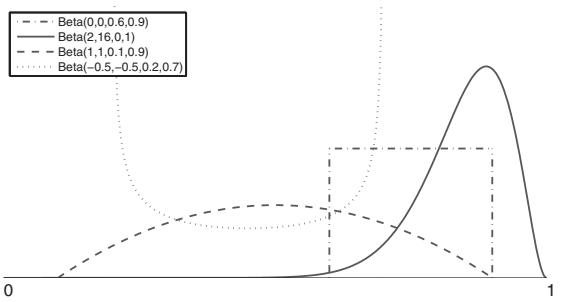


Figure 3. The four-parameter beta distribution.

Proof. First, $\mathrm{E}\left[x_i(A)\right] \geq 0$ because $0 \leq A \leq 1$ and $v_i \geq 0$. Then, we have

$$\|\mathrm{E}\left[\mathbf{x}(A)\right]\| = \mathbf{e}^T \int_0^1 \mathbf{x}(\alpha)\rho(\alpha)\, d\alpha = \int_0^1 \mathbf{e}^T\mathbf{x}(\alpha)\rho(\alpha)\, d\alpha = 1,$$

because $\mathbf{e}^T\mathbf{x} = 1$ for each α and $\int_0^1 \rho(\alpha)\, d\alpha = 1$. □

Finally, we show that for a certain class of pages, the expectation of RAPr is *equal* to that of PageRank with $\alpha = \mathrm{E}\left[A\right]$.

Theorem 5.4. *Let* $A \sim \mathrm{Beta}(a, b, [l, r])$ *with* $0 \leq l < r \leq 1$. *If* i *is the index for a node with no in-links, then* $\mathrm{E}\left[x_i(A)\right] = x_i(\mathrm{E}\left[A\right])$ *and* $\mathrm{Std}\left[x_i(A)\right] = v_i\,\mathrm{Std}\left[A\right]$.

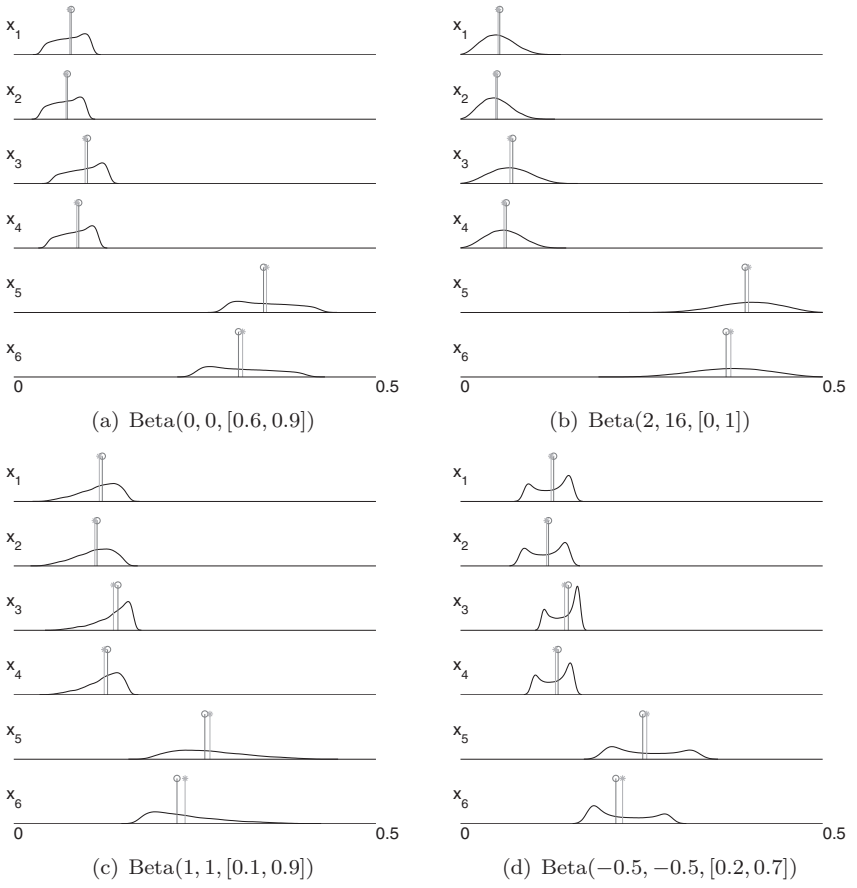
Proof. For a page with no in-links, $\mathbf{e}_i^T\mathbf{P}^n = 0$, $n > 0$, where $\mathbf{e}_i$ is the vector with a 1 in the ith component. Taking the Neumann series for $\mathbf{x}(A)$ gives

$$x_i(A) = \mathbf{e}_i^T \sum_{j=0}^{\infty} (A^j - A^{j+1})\mathbf{P}^j\mathbf{v} = \mathbf{e}_i^T(A^0 - A^1)\mathbf{v} = (1 - A)v_i.$$

Equality of the statistics follows from the linearity of the expectation operator. □

While Theorem 5.4 yields one condition in which the expectation is the same for the random and deterministic models, the result may not be useful. Given many of the standard corrections for dangling nodes (including the methods used in this paper, where all dangling nodes link according to $\mathbf{v}$ and $\mathbf{v} = 1/n\mathbf{e}$), a graph with any dangling nodes will induce an effective graph in which all nodes have an in-link.

At this point, we have finished our theoretical survey of the model. In what follows, we present one example of how these theoretical contributions allow us to interpret the resulting model.



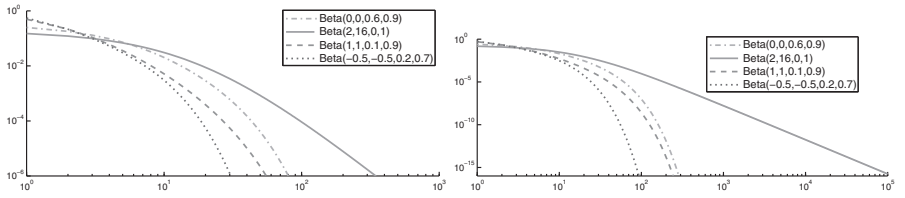
(a) Beta$(0, 0, [0.6, 0.9])$ (b) Beta$(2, 16, [0, 1])$

(c) Beta$(1, 1, [0.1, 0.9])$ (d) Beta$(-0.5, -0.5, [0.2, 0.7])$

Figure 4. The four-parameter beta distribution is quite flexible and exhibits a range of behaviors as a function of α, β, l, and r. The four density plots correspond to the graph from Figure 2 with A drawn from the beta distribution in the caption. When $\alpha, \beta < 0$, the resulting PageRank density functions are bimodal. (PageRank densities are computed with a kernel density estimator applied to 10,000 random samples.)

5.3. A Path-Damping and Browse Path View

Although we derived the RAPr model by replacing the deterministic coefficient α with a random variable A, the resulting model has strong connections with other generalizations of PageRank based on *path-damping coefficients* [Baeza-Yates et al. 06]. From the Neumann series for $\mathrm{E}[\mathbf{x}(A)]$, (5.3), the coefficient on the jth power of $\mathbf{P}$ is the weight placed on a path of length j in the Markov chain. Because these coefficients tend to decrease as j increases, they "damp"



(a) $\mathrm{E}\left[A^n - A^{n+1}\right]$ (b) $\mathrm{E}\left[A^n - A^{n+1}\right]$ for larger n

Figure 5. These two figures show the path-damping coefficients for the distributions from Figure 4 drawn with the same legend.

longer paths in the Markov chain. Figure 5 shows the path-damping coefficients for the distributions from Figures 3 and 4.

As shown in Figure 5, the path-damping view of RAPr provides interesting information about the impact of different distributions. For details on the algorithmic implications of the path-damping view, see Section 6.3.

6. Algorithms

We now move to a discussion of how to compute the vectors $\mathrm{E}[\mathbf{x}(A)]$ and $\mathrm{Std}[\mathbf{x}(A)]$. We define all the algorithms in this section before proceeding to analyze their behavior and cost in the next section. Specifically, we describe three methods for approximating the statistics $\mathrm{E}[\mathbf{x}(A)]$ and $\mathrm{Std}[\mathbf{x}(A)]$ of the RAPr model. Each of the following methods has a unique history. In the last decade, each has been successfully applied to partial differential equation models with stochastic input parameters; this has transpired in the burgeoning field of *uncertainty quantification*, where the goal is to obtain statistics—such as expectation, standard deviation, and approximate density function—of the stochastic output quantities of interest in models with stochastic inputs. Applications in the recent literature include fluid dynamics [Mathelin et al. 05], structural mechanics [Ghanem and Red-Horse 99], and petroleum engineering [Christie et al. 02]. To the best of our knowledge, ours is the first application to data mining.

In this section, we illustrate all our codes using working MATLAB implementations rather than pseudocodes. The intention is to exhibit a fully working code, albeit with a few simplifications and inefficiencies dictated by the desire for a compactly written code. These implementations consume only slightly more space than a pseudocode description of each algorithm. Furthermore, the theoretical derivation of each algorithm allows readers to move from our specific implementations to more general implementations, if necessary.

6.1. PageRank

One key component of these algorithms is a solver for a deterministic PageRank problem with $\alpha < 1$. For this task, we use two solvers: a direct method and an inner–outer method [Gleich et al. 10b]. Let us explicitly state that the choice of PageRank solver is arbitrary. In two of our methods, we use *any* PageRank solver.

For computational efficiency, all the MATLAB programming uses row substochastic matrices. These matrices are all implicitly corrected in the strongly preferential sense. We make this choice because it seems to be the most common, although we note that all the methods can be adapted to the weakly preferential case.

The direct method uses the "backslash" solve in MATLAB. Given a row substochastic matrix $\bar{\mathbf{P}}$—produced, for example, by a random walk normalization of a graph—we solve

$$(\mathbf{I} - \alpha\bar{\mathbf{P}}^T)\mathbf{y} = \mathbf{v}, \qquad \mathbf{x}(\alpha) = \mathbf{y}/\left\|\mathbf{y}\right\|.$$

This procedure solves (2.1) when $\mathbf{P} = \bar{\mathbf{P}}^T + \mathbf{v}\mathbf{d}^T$ and $\mathbf{d} = \mathbf{e} - \bar{\mathbf{P}}\mathbf{e}$ [Langville and Meyer 06]. The inner–outer iteration requires only matrix–vector products, which makes it a natural choice for a generic algorithm. Using Gauss–Seidel iterations [Arasu et al. 02] or a graph decomposition algorithm [Eiron et al. 04, Ipsen and Selee 07, Langville and Meyer 06] is typically faster, but these algorithms require access to the graph structure and its manipulation.

Our code works with a native Matlab sparse matrix structure and does not perform any manipulations of the graph structure. We note that some of these standard manipulations may accelerate our codes.

6.2. Monte Carlo

An enticingly straightforward method to compute the expectations, standard deviations, and density functions of RAPr is to use a Monte Carlo method (see [Asmussen and Glynn 07] for a modern treatment). To wit, first generate N realizations of A from a chosen distribution, and then solve each resulting PageRank problem. With the N different realizations of $\mathbf{x}(\alpha_i)$, $i = 1, \ldots, N$, we can compute unbiased estimates for $\mathrm{E}\left[\mathbf{x}(A)\right]$ and $\mathrm{Std}\left[\mathbf{x}(A)\right]$ with the formulas

$$\mathrm{E}\left[\mathbf{x}(A)\right] \approx \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}(\alpha_i) \equiv \hat{\mu}_{\mathbf{x}},$$

$$\mathrm{Std}\left[\mathbf{x}(A)\right] \approx \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}(\alpha_i) - \hat{\mu}_{\mathbf{x}})^2}.$$

```
function [ex,stdx] = mcrapr(P,N,ba,bb,bl,br)
tol=1e-9; maxterms=500; n=size(P,1); v=1/n;
alphas = betarnd(bb+1,ba+1,N,1)*(br-bl) + bl;
ex=zeros(n,1); stdx=zeros(n,1);
for i=1:N
    % solve the PageRank system
    x = inoutpr(P,alphas(i),v,tol,2*ceil(log(tol)/log(alphas(i))));
    % update the running solution sum and variance sum formulas
    ex = ex+x; if i>1, stdx = stdx + (1./(i*(i-1))).*(i*x-ex).^2; end
end
ex = ex./N; stdx=sqrt(stdx./(N-1)); % compute the mean and std
```

Figure 6. A Monte Carlo code in MATLAB to estimate the expectation and standard deviation of the RAPr model. The function `inoutpr` solves a strongly preferential PageRank problem for a row substochastic matrix `P` and takes parameters α, $\mathbf{v}$, tolerance, and maximum iteration.

Unfortunately, as with any Monte Carlo method, these estimates converge as $1/\sqrt{N}$ [Asmussen and Glynn 07], which makes this approach prohibitively expensive for large graphs such as the web graph. The real advantage of the Monte Carlo method is its beautiful simplicity. The short code in Figure 6 is our entire implementation of the Monte Carlo method, including a numerically stable method to update the running variance computation [Chan et al. 83].

6.3. Path Damping

As discussed in Sections 4.3 and 5.3, *path-damping* algorithms for PageRank are not novel. RAPr simply provides a large set of functions that generate the path-damping coefficients. In this section, we will discuss using these ideas to compute $\mathrm{E}[\mathbf{x}(A)]$ and $\mathrm{Std}[\mathbf{x}(A)]$.

Recall the Neumann series from Theorem 5.1:

$$\mathrm{E}[\mathbf{x}(A)] = \sum_{\ell=0}^{\infty} \mathrm{E}\left[A^{\ell} - A^{\ell+1}\right] \mathbf{P}^{\ell}\mathbf{v}.$$

If we truncate this series to a finite value N, then an algorithm for $\mathrm{E}[\mathbf{x}(A)]$ immediately follows:

$$\mathrm{E}[\mathbf{x}(A)] \approx \mathbf{x}^{(N)} = \sum_{\ell=0}^{N} \mathrm{E}\left[A^{\ell} - A^{\ell+1}\right] \mathbf{P}^{\ell}\mathbf{v} + \underbrace{\left(1 - \sum_{\ell=0}^{N} \mathrm{E}\left[A^{\ell} - A^{\ell+1}\right]\right)}_{=\mathrm{E}[A^{N+1}]} \mathbf{P}^{N+1}\mathbf{v}. \tag{6.1}$$

The final term in this summation ensures that $\mathbf{e}^T\mathbf{x}^{(N)} = 1$ by completing the telescopic series. The previous equation produces an algorithm because it involves only matrix–vector products with $\mathbf{P}$, assuming that we have some way to compute the difference in moments. We will return to that point shortly.

To compute $\mathrm{Std}\left[\mathbf{x}(A)\right]$ using the path-damping equations we compute $\mathrm{E}\left[\mathbf{x}(A) \bullet \mathbf{x}(A)\right]$ (where $\bullet$ is the elementwise or Hadamard product) and then compute

$$\mathrm{Std}\left[\mathbf{x}(A)\right] = \sqrt{\mathrm{E}\left[\mathbf{x}(A) \bullet \mathbf{x}(A)\right] - \left(\mathrm{E}\left[\mathbf{x}(A)\right] \bullet \mathrm{E}\left[\mathbf{x}(A)\right]\right)},$$

where $\sqrt{\ }$ is the elementwise square-root function. Based on the Neumann expansion,

$$\mathrm{E}\left[\mathbf{x}(A) \bullet \mathbf{x}(A)\right] = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \mathrm{E}\left[A^{i+j} - 2A^{i+j+1} + A^{i+j+2}\right] \left(\mathbf{P}^i\mathbf{v}\right) \bullet \left(\mathbf{P}^j\mathbf{v}\right).$$

And again, we truncate this series to a common term in both i and j:

$$\mathrm{E}\left[\mathbf{x}(A) \bullet \mathbf{x}(A)\right] \approx \mathbf{s}^{(N)} = \sum_{i,j}^{N} \mathrm{E}\left[A^{i+j} - 2A^{i+j+1} + A^{i+j+2}\right] \left(\mathbf{P}^i\mathbf{v}\right) \bullet \left(\mathbf{P}^j\mathbf{v}\right).$$

Note that we do not apply any correction to the sum to ensure a summation property of the solution as in the case for $\mathrm{E}\left[\mathbf{x}(A)\right]$.

```
function m=beta_moments(N,a,b,l,r)
c = l; s = (r-l); m=zeros(N+1,1);            % c is the offset, s is the scale
uk=1; k=0; sk=1; m(1) = uk;                  % uk are the Beta(a,b,0,1) moments
for i=1:N, k = k+1; uk=s*uk*((b+k)/(a+b+k+1)); m(i+1) = uk; end
% form the shifted and scaled moments       % m are the Beta(a,b,l,r) moments
if c ~= 0, for i=1:N, m(i+1:end) = c*m(i:(end-1)) + m((i+1):end); end, end
```

(a) Moment computation

```
function [ex,stdx] = pdrapr(P,N,a,b,l,r)
tol=1e-9; maxterms=500; n=size(P,1); v=1/n;
ms = beta_moments(2*(maxterms+1),a,b,l,r);              % setup the moments
i=0; delta=2; ex=zeros(n,1); y = zeros(n,1) + v; s=0;   % setup vectors
while i<maxterms && ms(i+2)>tol
    Ptiv=y; ex = ex + (ms(i+1)-ms(i+2))*Ptiv;
    y = P'*(Ptiv); y = y + (1-norm(y,1)).*v; i=i+1; end % update P^i v
ex = ex + ms(i+2)*y;                                    % adjust with the last term
% compute stdx with same number of terms of sequence
nterms=i; ex2=zeros(n,1); Ptiv = zeros(n,1); Ptiv=Ptiv+v; Ptjv=Ptiv;
for i=0:nterms, for j=0:nterms
    ex2 = ex2 + (ms(i+j+1)-2*ms(i+j+2)+ms(i+j+3))*(Ptiv.*Ptjv);
    y = P'*(Ptjv); Ptjv = y + (1-norm(y,1)).*v;
  end                                                   % now update Ptiv and reset Ptjv
  y = P'*(Ptiv); Ptiv = y + (1-norm(y,1)).*v; Ptjv(:)=0; Ptjv=Ptjv+v;
end                                                     % finish by update ex2 to be stdx
stdx = sqrt(ex2-ex.^2);
```

(b) Path-damping computation

Figure 7. The first figure presents a compact algorithm to compute the moments of a beta distribution. Next, we present our implementation of the path-damping algorithms using the moments. Just as for the PageRank case, we perform our computations with $\mathtt{P} = \mathbf{P}^T$ for efficiency.

Given the moments of the distribution A,

$$\mu_k(A) = \mathrm{E}\left[A^k\right], \quad 0 \le k \le 2N+2,$$

the previous summation expressions become algorithms. For $A \sim \mathrm{Beta}(a, b, [0, 1])$, the values μ_k are known analytically [Zwillinger et al. 96]:

$$\mu_0 = 1, \quad \mu_k = \frac{b+k}{a+b+k+1}\mu_{k-1} = \prod_{j=1}^{k} \frac{b+j}{a+b+j+1}, \quad k \ge 1. \tag{6.2}$$

Figure 7 gives a simple implementation of the path-damping algorithms and an implementation of a code to compute the moments of the general four-parameter beta density. This latter code, described in Section 11, is based on a dynamic programming approach.

6.4. Gaussian Quadrature

RAPr has only one random parameter $A \sim \mathrm{Beta}(a, b, [l, r])$, so we can employ the one-dimensional interpolation and integration formulas—commonly called quadrature—to produce highly accurate statistics. In this section we discuss their application to RAPr.

For a modern reference on Gaussian quadrature, see [Gautschi 02]. In an N-point quadrature rule, we approximate

$$\int_l^r f(x)\, dw(x) \approx \sum_{i=1}^{N} f(z_i) w_i, \tag{6.3}$$

where z_i are the N nodes or points of a quadrature rule and w_i are the corresponding weights. In Gaussian quadrature, these nodes and weights are chosen to make the integration exact if f is a polynomial of degree less than $2N$, and there are efficient ($O(N \log N)$ or $O(N^2)$) and stable algorithms to compute these rules [Glaser et al. 07, Golub and Welsch 69]. Note that the quadrature rule changes if the integration endpoints change, or if the weight function w changes.

With the points and weights of the Gaussian quadrature formula, we first solve N deterministic PageRank problems

$$(\mathbf{I} - z_i\mathbf{P})\mathbf{x}_i = (1 - z_i)\mathbf{v} \tag{6.4}$$

using methods described in Section 6.1. Then we can compute statistics of RAPr with the quadrature formulas

```
function [ex,stdx] = gqrapr(P,N,a,b,l,r)
% first run these commands to get the OPQ codes
% urlwrite('http://www.cs.purdue.edu/archives/2002/wxg/codes/gauss.m','gauss.m')
% urlwrite('http://www.cs.purdue.edu/archives/2002/wxg/codes/r_jacobi.m','r_jacobi.m')
% urlwrite('http://www.cs.purdue.edu/archives/2002/wxg/codes/r_jacobi01.m','r_jacobi01.m')
tol=1e-9; maxit=1000; n=size(P,1); v=1/n;
ab=r_jacobi01(N,a,b); xw=gauss(N,ab); xw(:,2) = (1./beta(b+1,a+1))*xw(:,2);
xw(:,1) = (r-l).*xw(:,1)+l;          % generate the quadrature rule by scale and shift
ex = zeros(n,1); stdx = zeros(n,1); % initialize running sums
for i=1:N
    % solve the PageRank system
    x = inoutpr(P,xw(i,1),v,min(tol./xw(i,2),1e-2), ... % adjust tol and maxit
      2*ceil(log(min(tol./xw(i,2),1e-2))/log(xw(i,1))));% for mult by xw(i,2)
    ex = ex+xw(i,2).*x; stdx = stdx+xw(i,2).*(x.^2);
end
stdx = sqrt(stdx - ex.^2);           % convert to stdx
```

Figure 8. Using Gautschi's OPQ codes, `r_jacobi01.m` and `gauss.m`, a MATLAB quadrature implementation is quite easy. The function `inoutpr` solves a strongly preferential PageRank problem for a row substochastic matrix `P` and takes parameters α, $\mathbf{v}$, tolerance, and maximum iteration.

$$\mathrm{E}\left[\mathbf{x}(A)\right] \approx \sum_{i=1}^{N} \mathbf{x}_i w_i, \tag{6.5}$$

$$\mathrm{Std}\left[\mathbf{x}(A)\right] \approx \sqrt{\sum_{i=1}^{N} (\mathbf{x}_i \bullet \mathbf{x}_i) w_i - \left(\sum_{i=1}^{N} \mathbf{x}_i w_i\right) \bullet \left(\sum_{i=1}^{N} \mathbf{x}_i w_i\right)}.$$

These two procedures—compute $\mathbf{x}_i$ and a weighted sum—constitute the algorithm for Gaussian quadrature implemented in Figure 8.

For the quadrature rule (6.3), the nodes z_i are known to lie on the interior of the integration region, $l < z_i < r$. Furthermore, the weights w_i are strictly positive. The first property is essential to using quadrature with PageRank when the right endpoint is 1 (i.e., $r = 1$). It states that we do not have to compute a PageRank vector at $\alpha = 1$. Many other quadrature rules, such as Clenshaw–Curtis, Gauss–Radau, and Gauss–Lobatto, all utilize a function value at one or both of the endpoints. For PageRank, computing the limit vector $\mathbf{x}(1)$ efficiently is still an open problem, and hence these alternatives are not appropriate.

As Figure 8 shows, implementing the Gaussian quadrature algorithm is easy using the OPQ routines [Gautschi 02]. A more efficient code for computing the Gaussian quadrature rule is available in the ***chebfun*** package's function `jacpts.m` [Trefethen et al. 09]. In the code, we adjust the solution tolerance of the linear system based on the weights of the final quadrature summation. We call this a weighted tolerance τ.

7. Algorithm Analysis

In this section, we compare and analyze the algorithms using theoretical and numerical techniques. As with many such comparisons, the theoretical analysis does not treat every case, and the numerical comparisons are always limited to the chosen experiments. Nevertheless, the combined examination yields strong suggestions for the choice of algorithm and implementation when applied to RAPr. For a compact summary of the properties of the methods, see Table 2. The twin objectives of the analysis are work and accuracy. Both are typically proportional to N, the number of terms used in the approximate statistics. For the Monte Carlo, N is number of samples (i.e., the number of PageRank systems solved). For the path-damping approach, N is the number of terms of the Neumann series (i.e., the number of matrix–vector products). For the Gaussian quadrature algorithm, N is the number of quadrature points (i.e., the number of PageRank systems solved). As we shall show in the following sections, each algorithm produces a sequence of vectors as a function of N that converges to the exact answer as N goes to infinity. One of the key quantities listed in the table is the *rate* at which these algorithms converge. In the three following sections, we establish the convergence rate and work for each of these three algorithms.

7.1. Monte Carlo

The convergence behavior for Monte Carlo is well known [Asmussen and Glynn 07]. The question we address here regards the work that this procedure requires. In Monte Carlo, we have to solve N PageRank systems at random values of α. Each PageRank problem is solved iteratively and matrix–vector multiplications with $\mathbf{P}$ dominate the work. We cannot find a precise number of matrix–vector multiplications because the work varies between runs. Instead, we compute the expected (or average) number.

Let $\mathbf{x}^*$ be the exact solution of a PageRank problem. We begin our formal analysis by noting that the kth iterate from the power method on the PageRank system satisfies

$$\left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\| \leq 2\alpha^k. \tag{7.1}$$

Thus, when $k > \log(\varepsilon)/\log(\alpha)$, it has error bounded by 2ε, which takes at most k iterations (matrix multiplications). Empirically, the inner–outer method uses fewer iterations than the power method, so we use the upper bound from the latter.

Method	Nonintrusive	Update	Storage
Monte Carlo	+	+	`prsolve` + 2 n-vectors
Path Damping	−	+	`P mult` + 5 n-vectors
Gaussian Quadrature	+	−	`prsolve` + 2 n-vectors

(a) Algorithm properties

Method	Conv.	Work Required	What is N?
Monte Carlo	$\frac{1}{\sqrt{N}}$	N PageRank systems	number of samples from A
Path Damping (without Std $[\mathbf{x}(A)]$)	$\frac{r^{N+2}}{N^{1+a}}$	$N+1$ matrix–vector products	terms of the Neumann series
Gaussian Quadrature	r^{2N} γ^{-N}	N PageRank systems	number of quadrature points

(b) Convergence analysis

Table 2. A brief summary of our results about each method. A nonintrusive method uses only an existing PageRank solver. The Monte Carlo and path-damping algorithms can be updated from N to $N+1$ with no more work than another iteration, whereas the Gaussian quadrature routines produce different instances when N is incremented. For storage, `prsolve` is the storage required to solve a PageRank problem, and `P mult` is the storage required for the matrix $\mathbf{P}$. The convergence results show how the norm of the error decays as a function of N, the intrinsic parameter of the method, and both a and r, the parameters from the beta distribution. For quadrature, we list two bounds. The first applies when $r < 1$, while the second is a generic bound on a quadrature method in terms of $\gamma > 1$, which is related to the region of analyticity of the components of the PageRank function. We have geometric convergence in either case.

Using the bound (7.1), we can estimate the expected number of iterations in the Monte Carlo method given that we are taking N samples:

$$\mathrm{E}[M] = \sum_{i=1}^{N} \mathrm{E}[M_i] \le N \underbrace{\int_l^r \frac{\log(\varepsilon)}{\log(\tau)} \rho(\tau)\, d\tau}_{\text{expected iterations for one sample}}, \tag{7.2}$$

where M is the total number of matrix–vector multiplications and M_i is the number of iterations in the ith random sample.

In the case that $\rho(\tau)$ corresponds to a beta distribution over $(0,1)$ with integer $a > 0$, $b > 0$, we can solve the integral analytically. The following theorem summarizes the result, which follows simply by solving (7.2) when ρ is from the appropriate distribution.

Theorem 7.1. *If $A \sim \text{Beta}(a, b, [0, 1])$ with integers $a > 0$ and $b > 0$, then approximating $\text{E}[\mathbf{x}(A)]$ with an N-sample Monte Carlo method takes*

$$N \frac{\log \varepsilon}{\text{Beta}(a+1, b+1)} \sum_{k=0}^{a} (-1)^k \binom{a}{k} \log(b+k+1) \tag{7.3}$$

matrix multiplications in expectation.

One problem with this theorem is that it does not handle $a = 0$, $b = 0$, the case in which A is uniformly distributed. Computing this expectation exactly is impossible in this case because the improper integral $\int_0^1 \log^{-1}(\tau)\, d\tau$ does not converge. To handle this case, we would need to use an algorithm in which the number of iterations does not go to ∞ as $\alpha \to 1$. The bound for the power method, unfortunately, is insufficient for this task.

7.2. Path Damping

When $A \sim \text{Beta}(a, b, [0, r])$, $r \leq 1$, we can explicitly bound the convergence of the path-damping algorithm for $\text{E}[\mathbf{x}(A)]$. Recall the path-damping approximation from (6.1):

$$\text{E}[\mathbf{x}(A)] \approx \mathbf{x}^{(N)} = \sum_{\ell=0}^{N} \text{E}\left[A^\ell - A^{\ell+1}\right] \mathbf{P}^\ell \mathbf{v} + \left(1 - \sum_{\ell=0}^{N} \text{E}\left[A^\ell - A^{\ell+1}\right]\right) \mathbf{P}^{N+1} \mathbf{v}.$$

Note that

$$\left(1 - \sum_{\ell=0}^{N} \text{E}\left[A^\ell - A^{\ell+1}\right]\right) = \text{E}\left[A^{N+1}\right].$$

Thus we have

$$\begin{aligned}
\left\| \mathbf{x}^{(N)} - \mathbf{x}^* \right\| &= \left\| \text{E}\left[A^{N+2}\right] \mathbf{P}^{N+1} \mathbf{v} - \sum_{\ell=N+2}^{\infty} \text{E}\left[A^\ell - A^{\ell+1}\right] \mathbf{P}^\ell \mathbf{v} \right\| \\
&\leq \text{E}\left[A^{N+2}\right] + \mathbf{e}^T \sum_{\ell=N+2}^{\infty} \text{E}\left[A^\ell - A^{\ell+1}\right] \mathbf{P}^\ell \mathbf{v} \\
&\leq 2\, \text{E}\left[A^{N+2}\right].
\end{aligned}$$

We could have removed the final normalization term $\text{E}\left[A^{N+1}\right] \mathbf{P}^{N+1} \mathbf{v}$ in the summation and bounded the result by $\text{E}\left[A^{N+1}\right]$ instead. However, the iteration we outlined in the algorithms section gives us better performance in practice.

For $l = 0$ and $r \leq 1$, the moments are scaled modifications of (6.2). Using these moments shows that

$$\begin{aligned} \mathrm{E}\left[A^{N+2}\right] &= r^{N+2}\frac{\Gamma(b+N+3)\Gamma(a+b+2)}{\Gamma(b+1)\Gamma(a+b+N+4)} \\ &\leq r^{N+2}\frac{\Gamma(a+b+2)}{\Gamma(b+1)}\frac{1}{(b+N+3)^{a+1}}, \end{aligned}$$

from which we conclude that the path-damping algorithm for computing $\mathrm{E}\left[x(A)\right]$ converges like r^{N+2}/N^{a+1}.

7.3. Error Bounds on Gaussian Quadrature

Quadrature methods are old tools, and many excellent error analysis techniques exist. For example, [Davis and Rabinowitz 84] devotes an entire chapter to their study. Let $\mathbf{x}^{\mathrm{GQ}(N)}$ be the approximation to $\mathrm{E}\left[\mathbf{x}(A)\right]$ using an N-point quadrature rule. We can achieve only an error bound for any component of the solution and thus use the upper bound

$$\left\|\mathrm{E}\left[\mathbf{x}(A)\right] - \mathbf{x}^{\mathrm{GQ}(N)}\right\| \leq n \max_i \left|\mathrm{E}\left[x_i(A)\right] - x_i^{\mathrm{GQ}(N)}\right|.$$

This bound is terrible for large n, and we do not expect it to be tight. Instead, we focus on the error decay—how much the error drops when N increases.

Computing an explicit bound on a component is possible and is given by the following theorem.

Theorem 7.2. *Let A be a random variable with a continuous density function supported in $[0, r]$, where $r < 1$. Let $x_i^{GQ(N)}$ be an N-point quadrature approximation to the ith component of $\mathbf{x}(A)$. The error in the Gaussian quadrature approximation of $\mathrm{E}\left[\mathbf{x}(A)\right]$ is bounded above by*

$$\left|\mathrm{E}\left[x_i(A)\right] - x_i^{\mathrm{GQ}(N)}\right| \leq \frac{32\omega r}{15(1-\rho^{-2})\rho^{2N+2}},$$

where N is the number of points in the Gaussian quadrature rule,

$$\omega = \sqrt{1+\frac{1}{r}}, \quad \textit{and} \quad \rho = \frac{1}{r} + \sqrt{\frac{1}{r^2} - 1}.$$

In the interest of space, we defer the proof, which involves some intriguing facts about the PageRank function with a *complex* value for α, to Section 11. Thus, the quadrature codes converge to the exact solutions as $N \to \infty$ when $r < 1$.

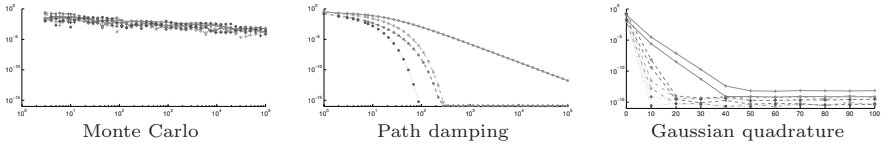
If the goal, however, is not an explicit bound, then we can use classical results about the region of analyticity to determine the asymptotic convergence rate of the quadrature rule for $r \leq 1$. Namely, let γ denote the sum of the semimajor and semiminor axes of the largest ellipse for which the components of $x(\alpha)$ are analytic. Then the error in a quadrature rule converges asymptotically at a rate of $O(\gamma^{-N})$ [Davis and Rabinowitz 84]. For PageRank, this result immediately shows that the quadrature rule always converges at a geometric rate governed by the closest singularity to the region $[l, r]$. The singularities of the PageRank function of α are $1/\lambda$ for each eigenvalue λ of $\mathbf{P}$ that is not $\lambda = 1$. In our definition of PageRank, we assume that $\mathbf{x}(1)$ assumes the limiting value. Thus there is no singularity at $\alpha = 1$. Without knowing the eigenvalues, we cannot bound the rate precisely.

7.4. Implementation Correctness and Convergence

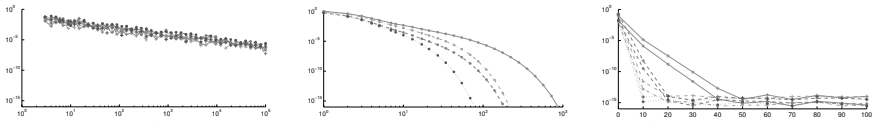
In this section, we present empirical results pertaining to the accuracy and convergence of our implementations. This type of analysis is important because numerical experimentation allows us to explore broader ranges of parameter values than may be feasible in the theoretical analysis. Additionally, it provides strong evidence that we have correctly implemented all the algorithms in this manuscript. To begin, we use three experiments to verify that our algorithms are convergent when implemented with and without approximate solutions of the linear algebra problems. Each of our algorithms has a parameter N that controls the degree of approximation. Theoretically, all the algorithms are convergent as $N \to \infty$.

We first test this convergence by comparing with a semianalytical solution. Using the symbolic toolbox inside MATLAB, we compute the PageRank vector as a rational function of α on the `har500cc` graph, a 335-node connected component. Using Mathematica, we then numerically integrate (6.5) for the expectation and standard deviation in 32-digit arithmetic. This process resolves the solution when converted to a double-precision number. Finally, we track convergence of each algorithm to these semianalytical solutions in Figure 9(a).

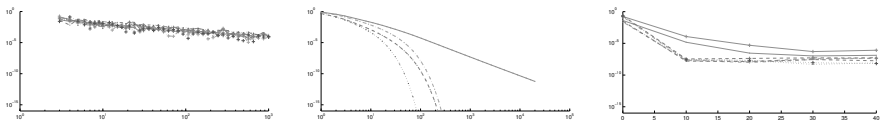
As the respective N increases, all methods demonstrate convergence to the exact solution. For the same graph, we also analyze another measure we term *stepwise convergence* by tracking the 1-norm change when incrementing N to $N + 1$: $\left\|\mathbf{y}^{(N+1)} - \mathbf{y}^{(N)}\right\|$. See Figure 9(b); this measure is tractable to compute for an unknown solution, and we hope to see convergence here as well. Both of these results use a direct method to solve any linear system that arises. Finally, we replace `har500cc` with `cnr-2000`, a 325,557-node graph, and use the inner–outer iteration to solve the PageRank systems with a tolerance of 10^{-8}. In all



(a) Convergence to analytical solutions (N vs. $\left\|\mathbf{y}^{(N)} - \mathbf{y}^{\star}\right\|$) on `har500cc`



(b) Stepwise convergence (N vs. $\left\|\mathbf{y}^{(N+1)} - \mathbf{y}^{(N)}\right\|$) with direct methods on `har500cc`



(c) Stepwise convergence (N vs. $\left\|\mathbf{y}^{(N+1)} - \mathbf{y}^{(N)}\right\|$) with iterative methods on `cnr-2000`

Figure 9. All of our implementations converge with iterative methods and direct methods in a stepwise sense for $\mathbf{y}^{(N)} \approx \mathrm{E}\left[\mathbf{x}(A)\right]$ (dotted points) and $\mathbf{y}^{(N)} \approx \mathrm{Std}\left[\mathbf{x}(A)\right]$ ("+" points). Computing the standard deviation with path damping was too inefficient to include. The colors correspond to distributions from Figure 4.

of these cases, the algorithms are convergent.

We now make a few additional observations:

- The Monte Carlo method has similar convergence behavior for all distributions and does not achieve better than expected $1/\sqrt{N}$ accuracy for all tests.
- The Beta$(2, 16, [0, 1])$ problem (solid line) requires the largest N for all methods except Monte Carlo.
- The accuracy of the standard deviation is less than the accuracy of the expectation.
- Using stepwise convergence as a proxy for analytical convergence in path damping can produce significant errors.

The last statement merits further comment. A simple calculation shows that stepwise convergence of the path-damping expression is

$$\left\|\mathbf{x}_{PD}^{(N)} - \mathbf{x}_{PD}^{(N+1)}\right\| = \mathrm{E}\left[A^{N+2}\right] \left\|\mathbf{P}^{N+2}\mathbf{v} - \mathbf{P}^{N+1}\mathbf{v}\right\|,$$

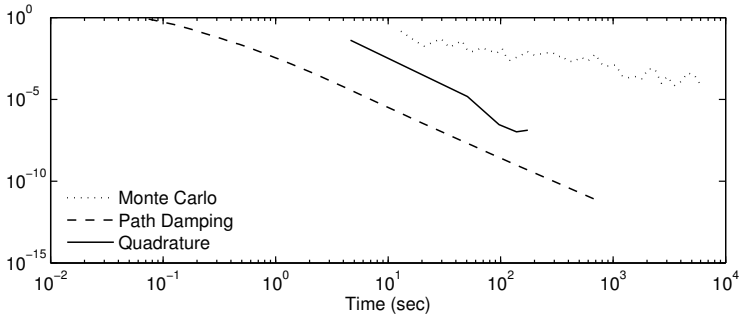


Figure 10. The time required to compute the results from Figure 9(c) for $\mathrm{E}\left[\mathbf{x}(A)\right]$, $A \sim \mathrm{Beta}(2, 16, [0, 1])$.

which is how we compute the values for the figures. The theoretical bound is much weaker with $\|\mathbf{P}^{N+2} - \mathbf{P}^{N+1}\|$ replaced by the trivial value 2. When the vectors $\mathbf{P}^N\mathbf{v}$ reach a small value, stepwise convergence is no longer a good bound. Consequently, our final code for the path damping formulation uses $\mathrm{E}\left[A^{N+2}\right]$ to test convergence instead.

Next, we examine the runtime for these methods in the hard case of the $\mathrm{Beta}(2, 16, [0, 1])$ distribution—the case $r = 1$ has the slowest convergence for all the methods. Figure 10 displays the values of Figure 9(c) against the time they took to compute. Again, the standard deviation was not computed for the path-damping algorithm. These timings include all computations of moments and eigenvalues for path damping and Gaussian quadrature.

Based on these experiments, we advise the following. Path damping is the algorithm of choice when $r \ll 1$ or the standard deviation is not required. Otherwise, the best method for computing both the expectation and standard deviation for reasonably accurate ($\approx 10^{-4}$–10^{-8}) solutions is Gaussian quadrature with about 33 points. Using 33 quadrature points may seem like a lot to those accustomed to integrating smooth functions. With PageRank, there is a singularity near the region of integration and we need to use many points.

8. Data

Before discussing applications of our method, we first summarize the data we use in Table 3. For both the `har500cc` and `us2004cc`, we precomputed the largest strong component of the original graph and used the strong component itself. The other graphs were translated from their native form to a MATLAB sparse matrix and then either saved in a MATLAB file or converted into a BVGraph file

Name	Source	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{C}\rvert$	$\max\lvert\mathcal{C}_i\rvert$	
har500cc	[Moler 04]	335	1,963	1	335	100 %
cnr-2000	[Boldi et al. 04, Boldi and Vigna 05]	325,557	3,216,152	100,977	112,023	34 %
uk-2006-host	[Castillo et al. 06]	11,402	730,774	2,935	7,945	70 %
uk-2007-host	[Castillo et al. 06]	114,529	1,836,441	54,822	59,160	52 %
nz2006	[Thelwall 03]	604,913	3,777,080	455,317	144,020	24 %
eu-2005	[Boldi et al. 04, Boldi and Vigna 05]	862,664	19,235,140	90,768	752,725	87 %
us2004cc	[Thelwall 03]	1,084,200	11,554,007	1	1,084,200	100 %
enwiki-2008	Section 8, [Various 08]	4,982,964	63,242,904	1,799,291	3,175,527	64 %
indochina	[Boldi et al. 04, Boldi and Vigna 05]	7,414,866	194,109,311	1,749,052	3,806,327	51 %
uk2005	[Thelwall 03]	10,037,216	47,993,341	7,338,097	2,604,479	26 %
uk-2006-05	[Castillo et al. 06]	77,741,046	2,965,197,340	10,789,143	49,710,330	64 %
generank	[Morrison et al. 05]	4,047	339,596	10	4,026	99 %

Table 3. The data sets used in our experiments span a wide range of sides. The two host databases are weighted host graphs, whereas the `generank` matrix is a weighted undirected graph. Let $\mathcal{V}$ denote the set of vertices, $\mathcal{E}$ the set of edges, and $\mathcal{C} = \{\mathcal{C}_i\}$ the set of connected components.

[Boldi and Vigna 04]. The `uk-2006-host`, `uk-2007-host`, and `generank` graphs are weighted. On these graphs, we used the natural weighted-degree random walk instead of a strict-degree-based random walk.

Wikipedia provides complete copies of their user-edited encyclopedia for download [Various 08]. We downloaded a database of their current page text from January 3, 2008, and converted it into a link graph. To keep the encyclopedia as the main focus, we only used pages in the "main" (the encyclopedia articles), "category" (a taxonomic organization of the articles), and "portal" (entrance guides to groups of articles) namespaces.

9. Applications

Thus far, we have theoretically examined the RAPr model, given algorithms to compute its statistics, and analyzed those algorithms; we have yet to address applications of this model. We do so now. Our first finding is that although the expected value of the RAPr model appears to order nodes like the deterministic PageRank vector at the expected α, the standard deviation vector orders nodes differently. We demonstrate this behavior for a range of graphs and distributions of A. Then, we show similar observations on a large web graph and discuss the intersection similarity of the standard deviation vector for this graph. Next, we

present an example of our model outside the web graph domain and observe that this ranking behavior of the standard deviation vector holds for a gene-ranking application. Finally, we show that using the standard deviation information aids a spam classification task. All of these experiments show that the standard deviation of RAPr ought to be useful in other applications, precisely because it reveals something *different* from PageRank.

9.1. PageRank

To begin our empirical analysis of RAPr, we present Table 4. In this table, we attempt to understand how the information from RAPr compares with standard PageRank. Our hope is that some of the information is *different* and that this information might aid a machine-learning framework. For the four beta distributions we have examined throughout this manuscript, the table presents the 1-norm, Kendall's τ correlation coefficient, and a truncated-τ correlation coefficient between $\mathbf{x}(\mathrm{E}[A])$, $\mathrm{E}[\mathbf{x}(A)]$, and $\mathrm{Std}[\mathbf{x}(A)]$. The 1-norm difference is rescaled to be related to a correlation coefficient when applied to probability distribution vectors. The τ value is 1 for identical lists and -1 for inverted lists. It is a common measure of correlation between the ranks induced by a vector, rather than the values, and has been used to study differences among PageRank variations [Baeza-Yates et al. 06]. The truncated-τ or τ_ε measure removes digits less than ε before computing τ. Formally,

$$\tau_\varepsilon(\mathbf{y}, \mathbf{z}) = \tau(\varepsilon\,\mathrm{round}(\mathbf{y}/\varepsilon), \varepsilon\,\mathrm{round}(\mathbf{z}/\varepsilon)),$$

where the "round function" rounds to the nearest integer. The τ_ε measure is motivated by inconsistencies with the τ measure and inaccurate computation [Boldi et al. 07]. The expectation and standard deviation were computed with a 33-point quadrature rule and each PageRank system solved to a weighted 10^{-9} tolerance (see Section 6.4).

From the table we make the following observations:

- The PageRank vector $\mathbf{x}(\mathrm{E}[A])$ and the expected value in the random model $\mathrm{E}[\mathbf{x}(A)]$ are numerically similar and induce similar orderings of the pages.
- The standard deviation vector $\mathrm{Std}[\mathbf{x}(A)]$ is neither numerically similar nor similar in either τ metric to $\mathbf{x}(\mathrm{E}[A])$.
- Using τ_ε can give different results.
- The behavior of the standard deviation vector is not consistent between graphs and distributions.

Graph	Beta				$\mathbf{y} = \mathbf{x}(\mathrm{E}[A]), \mathbf{z} = \mathrm{E}[\mathbf{x}(A)]$			$\mathbf{y} = \mathbf{x}(\mathrm{E}[A]), \mathbf{z} = \mathrm{Std}[\mathbf{x}(A)]$		
	a	b	l	r	$f(\mathbf{y},\mathbf{z})$	$\tau(\mathbf{y},\mathbf{z})$	$\tau_\varepsilon(\mathbf{y},\mathbf{z})$	$f(\mathbf{y},\mathbf{z})$	$\tau(\mathbf{y},\mathbf{z})$	$\tau_\varepsilon(\mathbf{y},\mathbf{z})$
uk-2006-host	0	0	0.6	0.9	0.972	0.995	0.995	0.173	0.200	0.196
	2	16	0	1	0.943	0.994	0.994	0.231	0.599	0.597
	1	1	0.1	0.9	0.963	0.984	0.983	0.229	−0.421	−0.418
	−0.5	−0.5	0.2	0.7	0.970	0.983	0.982	0.210	−0.457	−0.454
uk-2007-host	0	0	0.6	0.9	0.971	0.997	0.993	0.176	−0.071	−0.072
	2	16	0	1	0.944	0.996	0.995	0.232	0.498	0.455
	1	1	0.1	0.9	0.961	0.987	0.987	0.221	−0.578	−0.557
	−0.5	−0.5	0.2	0.7	0.969	0.986	0.975	0.201	−0.586	−0.563
nz2006	0	0	0.6	0.9	0.984	0.995	0.978	0.114	−0.546	−0.333
	2	16	0	1	0.976	0.996	0.966	0.135	0.027	−0.192
	1	1	0.1	0.9	0.975	0.981	0.980	0.143	−0.620	−0.506
	−0.5	−0.5	0.2	0.7	0.980	0.981	0.950	0.125	−0.614	−0.527
eu-2005	0	0	0.6	0.9	0.975	0.993	0.987	0.174	0.318	0.286
	2	16	0	1	0.952	0.992	0.982	0.214	0.517	0.524
	1	1	0.1	0.9	0.962	0.976	0.975	0.267	−0.536	−0.518
	−0.5	−0.5	0.2	0.7	0.968	0.975	0.974	0.251	−0.621	−0.604
us2004cc	0	0	0.6	0.9	0.971	0.989	0.990	0.173	0.179	0.177
	2	16	0	1	0.947	0.985	0.986	0.225	0.436	0.461
	1	1	0.1	0.9	0.960	0.969	0.973	0.247	−0.395	−0.364
	−0.5	−0.5	0.2	0.7	0.967	0.969	0.974	0.230	−0.489	−0.468
enwiki-2008	0	0	0.6	0.9	0.981	0.996	0.995	0.180	0.240	0.159
	2	16	0	1	0.975	0.995	0.994	0.189	0.381	0.184
	1	1	0.1	0.9	0.961	0.986	0.984	0.277	−0.444	−0.406
	−0.5	−0.5	0.2	0.7	0.966	0.986	0.984	0.262	−0.578	−0.222
indochina	0	0	0.6	0.9	0.975	0.993	0.968	0.165	0.189	0.229
	2	16	0	1	0.946	0.991	0.972	0.217	0.479	0.569
	1	1	0.1	0.9	0.966	0.974	0.958	0.250	−0.542	−0.284
	−0.5	−0.5	0.2	0.7	0.973	0.973	0.949	0.235	−0.613	−0.358
uk2005	0	0	0.6	0.9	0.985	0.997	0.903	0.110	−0.519	−0.199
	2	16	0	1	0.974	0.997	0.967	0.134	0.065	−0.034
	1	1	0.1	0.9	0.977	0.985	0.947	0.144	−0.596	−0.080
	−0.5	−0.5	0.2	0.7	0.981	0.984	0.916	0.128	−0.598	−0.137

Table 4. The function $f(\mathbf{y},\mathbf{z}) = 1 - \|\mathbf{y} - \mathbf{z}\|$ shifts the difference in norm to $[-1, 1]$ to make the values comparable to the other correlation coefficients; τ is Kendall's τ correlation coefficient; and τ_ε is τ with $\mathbf{y}$ and $\mathbf{z}$ truncated to eight digits. Values near 0 indicate places where the vectors are uncorrelated.

The first column group of the table justifies the first statement. All the values are near 1, which indicates a close correlation between two measures. The marked reduction in shading in the second column group explains the second, and the seemingly random values in this column group justify the last statement. Interestingly, four graphs behave nearly the same: `uk-2006-host`, `uk-2007-host`, `eu-2005`, and `us2004cc`. With the exception of `uk-2007-host`, these graphs have the highest percentage of nodes in the largest strong component.

The graph `uk2005` demonstrates the largest discrepancy between τ and τ_ε. This relatively large difference may signify that it differs characteristically from the other graphs. However, most of its standard deviation values are less than 10^{-8}, so truncating the τ metric with $\varepsilon = 10^{-8}$ may lose important information. Another explanation for the discrepancy is that more than half of the nodes in this graph have no links.

9.2. PageRank on a Large Graph

The graphs in the previous section are small compared with the size of the true web graph. Now we address computing the quantities on a graph with 78 million nodes and just under 3 billion edges: the `uk-2006` web spam test graph [Castillo et al. 06]. Even this graph is tiny compared with the real web graph, which is known to be over 150 billion pages [Cuil 09]. Our distributions of interest are $A_1 \sim \text{Beta}(2, 16, [0, 1])$ and $A_2 \sim \text{Beta}(1, 1, [0, 1])$. We chose the former because $\text{E}[A_1] = 0.85$, the canonical value of α, and the latter because $\text{E}[A_2] = 0.5$, a recently proposed alternative value of α. Both of these distributions have small a and support that extends all the way to 1. This makes computing the solution with path damping a difficult proposition, so we chose to use Gaussian quadrature. For A_1 we used a 25-point rule, and for A_2 we used a 10-point rule. The error bounds on quadrature state that these results may have considerable error from the quadrature approximation. But for big problems, running hundreds of Gauss points is not feasible (in Section 10, we discuss a few ideas to make the codes more scalable).

While the MATLAB codes given throughout this manuscript handle this graph through the *bvgraph* package, working in MATLAB yields roughly half the speed of an optimized computation. Consequently, we used a C++ implementation of the inner–outer iteration to solve the PageRank systems and compute the aggregated solution using a *bvgraph* structure to hold the graph in memory [Boldi and Vigna 04].

The time required for our deterministic solves (tolerance 10^{-12}) was

$$\begin{aligned} \alpha = 0.85, &\quad 204 \text{ minutes}, \\ \alpha = 0.5, &\quad 51 \text{ minutes}. \end{aligned}$$

Computing the expectation and standard deviation in the RAPr model required

$$\begin{aligned} A_1, &\quad 6199 \text{ minutes}, \\ A_2, &\quad 1569 \text{ minutes}. \end{aligned}$$

Our codes solved each PageRank vector to a tolerance of 10^{-12}. This accuracy is far more than required when given the intrinsic error in the quadrature approximation mentioned above. Nevertheless, we might as well get something accurate with these computations when we can.

To analyze the output, we used two schemes. First, we applied the truncated τ measure to the expectation and standard deviation vectors (Table 5). The comparison shows that $\text{E}[\mathbf{x}(A)] \approx \mathbf{x}(\text{E}[A])$ in terms of ranking and that the standard deviation vectors behave differently under this measure. Interestingly, the standard deviation vector for A_2 appears to invert the orderings of all other measures, and the magnitude of its anticorrelation is much stronger than for A_1.

				z		
y	$\mathbf{x}(0.85)$	$\mathbf{x}(0.95)$	$\mathrm{E}[\mathbf{x}(A_1)]$	$\mathrm{E}[\mathbf{x}(A_2)]$	$\mathrm{Std}[\mathbf{x}(A_1)]$	$\mathrm{Std}[\mathbf{x}(A_2)]$
$\mathbf{x}(0.5)$	0.850	0.765	0.845	*0.956*	0.412	*-0.538*
$\mathbf{x}(0.85)$		0.910	*0.967*	0.891	*0.294*	−0.675
$\mathbf{x}(0.95)$			0.916	0.808	0.219	−0.706
$\mathrm{E}[\mathbf{x}(A_1)]$				0.892	*0.287*	−0.675
$\mathrm{E}[\mathbf{x}(A_2)]$					0.378	*-0.577*

Table 5. The truncated τ values ($\tau_\varepsilon(\mathbf{y}, \mathbf{z})$ with $\varepsilon = 10^{-10}$) again show that the standard deviation vectors produce different rankings from the expectation vectors for the graph `uk-2006` with 77 million vertices and 2.2 billion edges. Here A_1 is a Beta$(2, 16, [0, 1])$ random variable with statistics computed using a 25-point quadrature rule, and the parameter A_2 is a Beta$(1, 1, [0, 1])$ random variable computed using a 10-point quadrature rule.

The second comparison metric is the intersection similarity metric [Boldi 05]. Whereas the τ metric counts interchanges and uses the same score for transpositions at the head and tail of the ranking, the intersection similarity shows the regions in which the two ranked lists differ. Given two ordered sequences of items $\mathcal{A}$ and $\mathcal{B}$, let $\mathcal{A}_k$ (respectively $\mathcal{B}_k$) be the top k items in $\mathcal{A}$ (respectively $\mathcal{B}$). Then

$$\mathrm{isim}_k(\mathcal{A}, \mathcal{B}) = \frac{1}{k} \sum_{j=1}^{k} \frac{|\mathcal{A}_j \Delta \mathcal{B}_j|}{2j},$$

where Δ is the symmetric difference operator between two sets. The intersection similarity is the average of the normalized symmetric differences for all top-j lists with $j \leq k$. If the two orderings are identical, then $\mathrm{isim}_k = 0$ for all k.

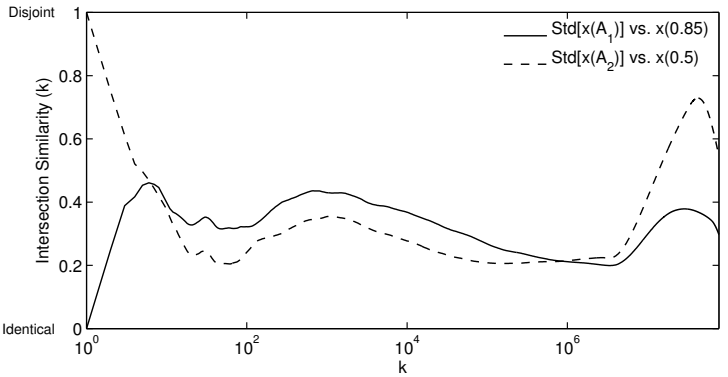


Figure 11. The intersection similarity metric for the `uk-2006` graph shows that the standard deviation vector is unlike the PageRank vector under this measure. The computations were done for $A_1 \sim \mathrm{Beta}(2, 16, [0, 1])$ with a 25-point quadrature rule and for $A_2 \sim \mathrm{Beta}(1, 1, [0, 1])$ with a 10-point quadrature rule.

If the two sequences have disjoint items, then $\text{isim}_k = 1$. Figure 11 displays this value for the standard deviation vectors. For A_1, the intersection similarity hovers around 0.3 with increases at 10, 1000, and 10,000,000 pages. In contrast, $\text{Std}\left[\mathbf{x}(A_2)\right]$ has a higher intersection similarity for the first 10^6 pages and orders the tail quite differently, resulting in a peak past 10^6 pages. This final peak is perhaps indicative of the negative τ correlation between $\text{Std}\left[\mathbf{x}(A_2)\right]$ and $\mathbf{x}(0.5)$.

These results support our claim that the standard deviation of RAPr reveals characteristically new information about the underlying graph. Now we explore an application far from working with web graphs.

9.3. Gene Regulatory Networks

Recently, many authors have used PageRank-type equations as measures on arbitrary graphs. Among these measures are GeneRank [Morrison et al. 05] for identifying important genes in a regulatory network, ProteinRank [Freschi 07] for identifying important proteins, and IsoRank [Singh et al. 07] for identifying important edges in a graph-isomorphism-like problem. We will demonstrate the results of RAPr on the GeneRank problem using the data published for that paper.

In this context, we cannot interpret RAPr as representing a hypothetical random surfer. Instead, the GeneRank vector is used with a single choice of α to infer important genes. We propose using the standard deviation vector as another set of important genes, or as "confidence bounds" on the actual importance values for a gene. GeneRank uses an undirected graph of known relationships between genes instead of the directed web graph in PageRank and specifies a teleportation vector $\mathbf{v}$ based on the expression level for each gene in a microarray experiment. In our experiments, we look at the τ correlation between $\mathbf{x}(\text{E}\left[A\right])$, $\text{E}\left[\mathbf{x}(A)\right]$, and $\text{Std}\left[\mathbf{x}(A)\right]$ for a range of parameters when $A \sim \text{Beta}(0, 0, [l, r])$ and when $A \sim \text{Beta}(a, b, [0, 1])$. The expectation and standard deviation vectors are computed with a 50-point quadrature rule with a direct solution method.

Table 6 presents the τ correlations. We note a few interesting observations. Again, the τ difference between the expectation and PageRank vector is negligible, whereas the standard deviation vector does produce a value of τ much closer to zero. For all the tests, τ is positive, and as the mass of the Beta distribution shifts closer to 1, the τ values become larger. We hypothesize that these effects are due to the symmetric nature of the initial GeneRank graph, which has a stationary distribution proportional to the weighted degree of a node. From these experiments, we believe that looking at the standard deviation vector would be useful in this application.

	r				
l	0.2	0.4	0.6	0.8	1.0
0.0	0.999	0.996	0.988	0.973	0.935
0.2		0.999	0.994	0.980	0.944
0.4			0.998	0.988	0.954
0.6				0.996	0.967
0.8					0.984

(a) Values of $\tau(\mathbf{x}(\mathrm{E}[A]), \mathrm{E}[\mathbf{x}(A)])$, $A \sim U(l, r)$

	r				
l	0.2	0.4	0.6	0.8	1.0
0.0	0.166	0.212	0.261	0.317	0.389
0.2		0.256	0.305	0.356	0.414
0.4			0.342	0.381	0.413
0.6				0.382	0.381
0.8					0.326

(b) Values of $\tau(\mathbf{x}(\mathrm{E}[A]), \mathrm{Std}[\mathbf{x}(A)])$, $A \sim \mathrm{U}(l, r)$

	b					
a	1	4	7	10	13	16
1	0.964	0.965	0.970	0.975	0.979	0.982
4	0.990	0.985	0.984	0.985	0.985	0.986
7	0.995	0.992	0.990	0.990	0.990	0.990
10	0.997	0.995	0.994	0.993	0.993	0.993
13	0.998	0.996	0.995	0.995	0.995	0.994
16	0.999	0.997	0.997	0.996	0.996	0.995

(c) Values of $\tau(\mathbf{x}(\mathrm{E}[A]), \mathrm{E}[\mathbf{x}(A)])$, $A \sim \mathrm{Beta}(a, b, 0, 1)$

	b					
a	1	4	7	10	13	16
1	0.378	0.410	0.386	0.362	0.344	0.331
4	0.263	0.362	0.395	0.399	0.392	0.383
7	0.217	0.305	0.355	0.382	0.392	0.394
10	0.194	0.268	0.319	0.352	0.373	0.385
13	0.180	0.244	0.291	0.326	0.350	0.367
16	0.170	0.226	0.269	0.303	0.329	0.349

(d) Values of $\tau(\mathbf{x}(\mathrm{E}[A]), \mathrm{Std}[\mathbf{x}(A)])$, $A \sim \mathrm{Beta}(a, b)$

Table 6. For the generank matrix, the Kendall-τ correlation coefficient shows that the PageRank and the expected PageRank order the genes similarly, whereas the standard deviation vector produces a different ordering under a wide range of parameters of the beta distribution.

9.4. Spam Classification

Thus far, the evaluations of RAPr have been speculative. We have seen that the standard deviation vector *differs* from the standard PageRank vector. However, the proof is in the pudding and for RAPr, the pudding is spam.

Web spam occurs when a web site consists primarily of misleading content or links designed to draw visitors to generate ad revenue or inflate another site's importance. Web spam is distinguished by this artificiality. Identifying these sites is a growing problem, and one technique is pure link analysis. Hypothetically, spam sites have different linking patterns from those of organic (nonspam) sites.

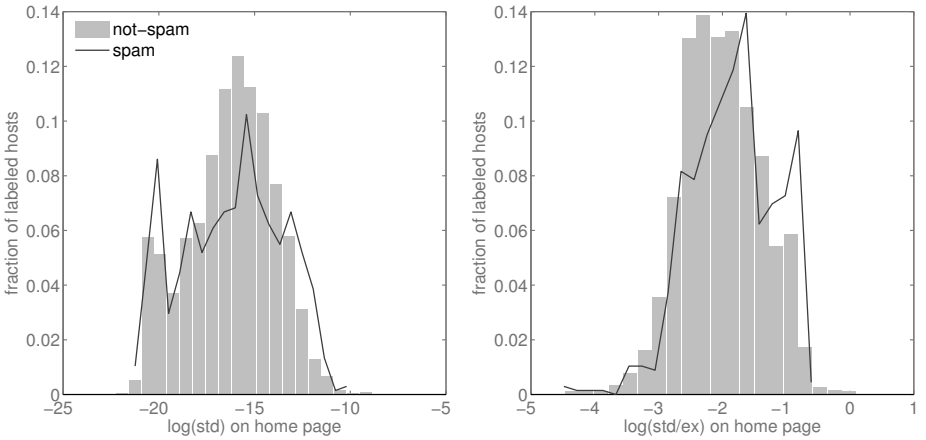


Figure 12. The background histogram displays (log) standard deviation scores for nonspam hosts when $A \sim \text{Beta}(2, 16, [0, 1])$. The foreground (black line) plot shows the same data for spam hosts. Each host is represented by its home page score and the statistics are computed with a 21-point quadrature rule. The second figure shows the same data for the (log) ratio of standard deviation over expectation.

In [Castillo et al. 06] and [Becchetti et al. 08], the authors investigate identifying web spam purely from link analysis. They labeled around 7,500 hosts from the `uk-2006` graph as follows:

Label	Training	Test
spam	674	1250
nonspam	4948	601
no label	5780	9551

The data have a training and test subset, although only the training subset is used in [Becchetti et al. 08] and in the following experiments. In the remainder of our own experiment, we continue following the methodology of [Becchetti et al. 08], and add the standard deviation vector from RAPr as an additional feature for a spam classification task. The fact that the authors have released their data makes experimentation straightforward.

Figure 12 shows that the standard deviation information identifies some spam pages. In particular, a high standard deviation relative to PageRank (the right-hand side of the right-hand figure) is a reasonably strong indicator. Ironically, a low standard deviation also appears to be an indicator.

Feature vectors for each host are included with the data from [Becchetti et al. 08]. These features are numerical results that may have an impact on the

"spamminess" of the pages on that web host and include TrustRank [Gyöngyi et al. 04], PageRank, and Truncated PageRank [Becchetti et al. 08] among others. Thus, pure PageRank ideas are already included. To support our statement that the standard deviation of RAPr is different, we must be able to *improve* upon the performance with all these features present.

Although measures such as PageRank, TrustRank, and RAPr produce one or two scores for each page, the previous study found that computing a few variations on these features aided the classification task. Thus, for RAPr on each host, we produce the following, where the RAPr scores are from the host home page and the page with largest PageRank on the host:

- log of RAPr expectation
- log of (RAPr expectation / log of outdegree)
- log of (RAPr expectation / log of indegree)
- standard deviation of RAPr expectation on in-links
- log of (standard deviation of RAPr expectation on in-links / PageRank)
- log of RAPr standard deviation
- log of (RAPr standard deviation / log of outdegree)
- log of (RAPr standard deviation / log of indegree)
- standard deviation of standard deviation on in-links
- log of (standard deviation of RAPr standard deviation on in-links / PageRank)
- log of (standard deviation of RAPr / RAPr expectation)

In total, we produce 22 features (= 11 from the list $\times 2$ for the different host pages) from the RAPr statistics.

Hosts, with all of their features, are then input to a machine-learning framework that attempts to learn a decision rule about spam based on these features. Just as in the original work, we use a bagged J48 tree classifier in Weka [Witten and Frank 05] with 10 bags. Bagging a classifier produces a new classifier whose label is the consensus of a bag of independent classifiers. On the training data, we conducted 50 independent tenfold cross-validation experiments to estimate the performance of the classifier, and Table 7 displays the results. For each classifier, we show the following:

	Precision	Recall	f-score	False Pos.	False Neg.
Baseline	0.694	0.558	0.618	0.034	0.442
Beta$(1.5, 0.5, 0, 0.99)$	0.692	0.557	0.617	0.034	0.443
Beta$(-0.5, -0.5, 0.3, 0.99)$	0.698	0.564	0.624	0.033	0.436
Beta$(0.5, 1.5, 0, 0.99)$	0.695	0.561	0.621	0.034	0.439
Beta$(10, 10, 0.3, 0.7)$	0.690	0.560	0.620	0.034	0.442
Beta$(1, 1, 0, 1)$	0.698	0.562	0.622	0.033	0.438
Beta$(2, 16, 0, 1)$	0.699	0.562	0.623	0.033	0.438

Table 7. Our performance baseline includes all the features from [Becchetti et al. 08]. Each row represents adding features from RAPr based on a particular beta distribution. The results are averaged over fifty repetitions of tenfold cross validation with a 10-bag J48 decision tree classifier. After adding features based on RAPr, we observe an improvement in the f-score. Consequently, these features uncover new information in the graph that is not expressed by PageRank.

- *precision:* fraction of spam pages corrected labeled as spam;
- *recall:* fraction of total spam pages identified;
- *fscore:* harmonic mean of precision and recall;
- *false positive:* fraction of nonspam pages mislabeled as spam;
- *false negative:* fraction of spam pages mislabeled as nonspam.

The RAPr features improve the performance of the classifier! It is a small improvement, only a few tenths of a percent. We obtain the best classification performance using features from the Beta$(-0.5, -0.5, [0, 3, 099])$ distribution. In some sense, this distribution represents the least likely surfer behavior. In contrast to many of the other metrics investigated in the baseline performance, there is no tuning of the RAPr metrics for spam ranking. If we combined RAPr and TrustRank, for example, it may be possible to achieve even better performance.

10. Conclusion

By incorporating information from multiple random surfers simultaneously, the RAPr model increases the flexibility of PageRank models considerably. It generalizes the many properties of the PageRank vector in a straightforward way. The expectation statistic generalizes other PageRank variants including the TotalRank algorithm.

We have presented three algorithms to compute the expectation and standard deviation for the RAPr setup. Two of these algorithms just use PageRank

solutions at multiple values of α. These algorithms can take advantage of any future acceleration in PageRank solvers [Karande et al. 90] or methods that exploit graph-theoretic optimizations [Eiron et al. 04, Ipsen and Selee 07, Lin et al. 09, Langville and Meyer 06, Lee et al. 07]. All methods converge both theoretically and empirically. Thus, computing these quantities is not a problem.

We observe that the standard deviation vector is uncorrelated with the PageRank vector under the Kendall-τ correlation coefficient. This observation holds for a wide variety of graphs, including a large (three-billion-link) graph, and a gene association graph. The RAPr statistics also improve a spam classification task. Thus we conclude that the standard deviation of RAPr represents a new and useful metric for an importance ranking on a graph.

While this paper presents an investigation into RAPr, we suggest two directions for further research.

First, although Gaussian quadrature is the fastest method to compute the expectation and standard deviation in the RAPr model, faster approximations are likely to be even more useful. Currently, we need to solve many PageRank problems at values of α that are close to 1. These vectors take considerable computation time. Gauss–Turán quadrature [Gautschi 04] uses derivatives in a quadrature rule with a derivative substituting for another point. Computing PageRank derivatives can be done at the *same* value of α, and thus switching to this new quadrature rule seems a promising idea to speed the RAPr computations. However, computing the Gauss–Turán rules themselves is far more difficult than computing a Gaussian quadrature rule.

Second, an equivalent way to write the RAPr system with quadrature is to solve

$$(\mathbf{I}_N \otimes \mathbf{I} - \mathbf{T}_N \otimes \mathbf{P})\mathbf{x} = (\mathbf{I}_N - \mathbf{T}_N)\mathbf{e}_1 \otimes \mathbf{v}$$

for a Jacobi matrix $\mathbf{T}_N$ given by the Gaussian quadrature rule. This system is a PageRank system with α replaced by the matrix $\mathbf{T}_N$. The vector $\mathbf{x}$ above is also a permutation of the PageRank function applied to a matrix parameter. That is, it is related to $\mathbf{x}(\mathbf{C})$, where $\mathbf{C}$ is a square matrix. (Again, PageRank is a rational function, and the PageRank function of a matrix is well defined.) Examining PageRank as a function of a matrix might lead to other generalizations of PageRank and some interesting connections with the derivative of PageRank with respect to α.

Finally, our code is available for experimentation.[2]

[2]Available at http://stanford.edu/~dgleich/publications/2009/rapr.

11. Appendix A: Four-Parameter Beta Moments

In this appendix, we derive an algorithm to compute the moments of a four-parameter $\text{Beta}(a, b, [l, r])$. To handle the general case, define

$$\hat{\mu}_j \equiv \mu_j(A), \quad \text{where } A \sim \text{Beta}(a, b, [0, 1]).$$

For $A \sim \text{Beta}(a, b, [l, r])$,

$$\begin{aligned}\text{E}\left[A^k\right] &= \int_l^r \zeta^k \rho^{(l,r)}_{\text{Beta}(a,b)}(\zeta)\, d\zeta = \int_0^1 ((r-l)\tau + l)^k \rho^{(0,1)}_{\text{Beta}(a,b)}(\tau)\, d\tau \\ &= \sum_{j=0}^{k} \binom{k}{j} \hat{\mu}_j (r-l)^j l^{k-j},\end{aligned}$$

and we can compute the moments of $A \sim \text{Beta}(a, b, [l, r])$ by scaling and shifting those of $A \sim \text{Beta}(a, b, [0, 1])$. Figure 7 gives an implementation of the recurrence

$$\begin{aligned}\mu_k(A) &= \mu^{(0,k)}, \\ \mu^{(i,j)} &= \sum_{m=i}^{j} \binom{j-i}{m-i} \hat{\mu}_m (r-l)^{m-i} l^{j-m} = (r-l)\mu^{(i,j-1)} + l\mu^{(i+1,j)},\end{aligned}$$

to compute the moments $\mu_k(A)$.

The implementation follows from organizing the moments into a matrix

$$\begin{bmatrix} \mu^{(0,0)} & \mu^{(0,1)} & \cdots & \mu^{(0,k)} \\ & \mu^{(1,1)} & \cdots & \mu^{(1,k)} \\ & & \ddots & \vdots \\ & & & \mu^{(k,k)} \end{bmatrix}$$

and filling in the entries $\mu^{(0,1)}, \ldots, \mu^{(0,k)}$ from the initially specified diagonal. At every step in the implementation, we compute a new diagonal.

12. Appendix B: Quadrature Convergence

We prove the following theorem.

Theorem 12.1. *Let A be a random variable with a continuous density function supported in $[0, r]$, where $r < 1$. The error in the Gauss quadrature approximation of $\text{E}[\mathbf{x}(A)]$ is bounded above by*

$$\left|\text{E}\left[x_i(A)\right] - x_i^{\text{GQ}(N)}\right| \leq \frac{32\omega r}{15(1-\rho^{-2})\rho^{2N+2}},$$

where N is the number of points in the Gauss quadrature rule,

$$\omega = \sqrt{1 + \frac{1}{r}},$$

and

$$\rho = \frac{1}{r} + \sqrt{\frac{1}{r^2} - 1}.$$

Proof. There are many statements for the error in Gauss quadrature, and we begin with a modern statement from [Trefethen 08, Theorem 4.5]. Consider $I = \int_{-1}^{1} f(x)\,dx$ for an analytic function f. Let I_N be the N-point Gauss quadrature approximation to I. Then

$$|I - I_N| \leq \frac{64\omega}{15(1 - (\rho_a + \rho_b)^{-2})(\rho_a + \rho_b)^{2N+N}}, \tag{12.1}$$

where $|f(z)| \leq \omega$ for all z in the ellipse with foci ± 1 and semimajor axis $\rho_a > 1$ and semiminor axis ρ_b. Figure 13(a) illustrates the construction.

Note that we are approximating the integral

$$\mathrm{E}\left[x_i(A)\right] = \int_0^r x_i(\alpha)\,d\alpha$$

with an N-point quadrature rule. Each PageRank component is a rational function with no poles in the interval of integration, which is a special case of an analytic function. In the remainder of the proof, we go through the details of applying the bound from (12.1) precisely. First, we transform the problem to the integration region $[-1, 1]$ by a change of variables α to z. In this z-space, we build an ellipse in the complex plane where $x_i(z)$ is analytic. To study the function magnitude ω, we transform the ellipse back to α-space and examine the magnitude of PageRank as a function of α when α is complex.

Let

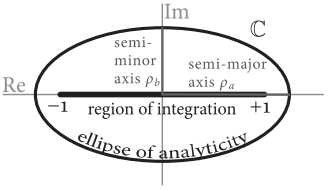
$$z = \frac{2\alpha}{r} - 1 \quad \Longleftrightarrow \quad \alpha = \frac{r}{2}(z + 1)$$

be the change of variables between α and z. Consider $z = z_R + iZ_i$ for z in the ellipse with foci ± 1. The ellipse satisfies

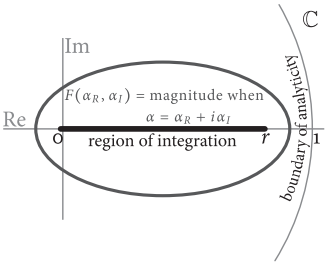
$$\frac{z_R^2}{\rho_a^2} + \frac{z_I^2}{\rho_b^2} = 1,$$

and the constraint on the foci implies that $\rho_a^2 = 1 + \rho_b^2$. Both ρ_a and ρ_b live in z-space, so for

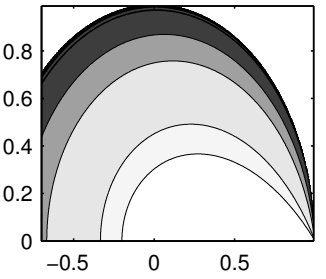
$$\rho_a = \frac{1}{r},$$



(a) The framework for Gauss quadrature error analysis



(b) The Gauss quadrature error analysis applied to PageRank



(c) PageRank magnitude for a complex damping parameter

Figure 13. The ellipse of analyticity provides bounds on the error in a Gauss quadrature rule. Roughly, $|\text{error}| \leq (\rho_a + \rho_B)^{2N}$. When integrating $x_i(\alpha)$, we use the ellipse given in (b) to bound the error in a Gauss quadrature approximation. Note that $x_i(\alpha)$ is clearly analytic in this region, since it is enclosed inside $|\alpha| < 1$. In this final plot, we show an upper bound on $\|\mathbf{x}(\alpha)\|$ when $\alpha \in \mathbb{C}$. Darker gray indicates larger magnitude, and white indicates a magnitude near zero. The magnitudes increase as α veers off the real line, or when the real component is negative.

we consider an ellipse in α-space with a right endpoint $r/2+1/2$, halfway between r and 1.[3] The function $x_i(\alpha(z))$ is analytic inside this ellipse. The right endpoint (in α-space) is less than 1, and $x_i(\alpha)$ is analytic for all complex α with $|\alpha| < 1$. See [Horn and Serra-Capizzano 07] for the first study of PageRank with complex α. Thus,

$$\rho_a + \rho_b = \frac{1}{r} + \sqrt{\frac{1}{r^2} - 1}$$

slips into (12.1) for the PageRank case.

Now that we have $\rho_a + \rho_b$, let us find ω.

[3]This choice of ρ_a may not be optimal, but other choices increase the difficulty of the computations considerably. In particular, we tried using a right endpoint of $\gamma r + (1 - \gamma)$, but could compute only the upper bound ω when $\gamma = 1/2$.

In α-space where $\alpha = \alpha_R + i\alpha_I$, the ellipse is

$$\frac{(r/2 - \alpha_R)^2}{(1/2)^2} + \frac{\alpha_I^2}{(\frac{1}{2}\sqrt{1-r^2})^2} = 1.$$

This ellipse is centered at $r/2$ with semimajor axis length $1/2$, as illustrated in Figure 13(b).

In (12.1), the value of ω is an upper bound on $f(z)$ inside the ellipse. Thus, we must bound the magnitude of PageRank components for a complex α. First,

$$\mathbf{x} = \alpha\mathbf{P}\mathbf{x} + (1-\alpha)\mathbf{v} \quad \text{gives} \quad \|\mathbf{x}\| \leq |\alpha|\,\|\mathbf{x}\| + |1-\alpha|.$$

For complex α, this bound yields

$$|x_i(\alpha)| \leq \|\mathbf{x}(\alpha)\| \leq \frac{|1-\alpha|}{1-|\alpha|} = \frac{\sqrt{(1-\alpha_R)^2 + \alpha_I^2}}{1 - \sqrt{\alpha_R^2 + \alpha_I^2}} \equiv F(\alpha_R, \alpha_I).$$

When $\alpha_I = 0$, this bound respects the property that $x_i(\alpha) \leq 1$ for $0 \leq \alpha_R < 1$. When $\alpha_I \neq 0$, the bound is considerably more interesting. In Figure 13(c), we see that as α_I increases, F increases. Analytically, we find that $\partial F/\partial \alpha_I > 0$ for $\alpha_I > 0$ and $\partial F/\partial \alpha_I < 0$ for $\alpha_I < 0$. Consequently, the maximum ω is going to occur on the boundary of the ellipse. In this case,

$$\alpha_I^2 = \frac{1}{4}(1-r^2)\left(1 - \frac{(r/2-\alpha_R)^2}{(1/2)^2}\right).$$

Let $F_R(\alpha_R) = F(\alpha_R, \alpha_I(\alpha_R))$ be the value of F on the ellipse. The critical points of F are

$$\alpha_R = \frac{r^2-1}{2r}, \quad \frac{r^2-2r-3}{2r}, \quad \frac{r^2+2r-1}{2r}$$

with respective values at these points

$$F_R(\cdot) = \sqrt{1 + \frac{1}{r} - r}, \quad -i\sqrt{\frac{3}{r} - 1}, \quad \sqrt{1 + \frac{1}{r}}.$$

Only

$$\alpha_R = \frac{r^2+2r-1}{2r}$$

is inside the region of integration, and thus

$$\omega = \sqrt{1 + \frac{1}{r}}.$$

There is one more step:

$$\left|\mathrm{E}\left[x_i(A)\right] - x_i^{\mathrm{GQ}(N)}\right| \leq \left|\frac{d\alpha}{dz}\right| \left|\mathrm{E}\left[x_i(z)\right] - x_i^{\mathrm{GQ}(N)}\right|.$$

The initial bound (12.1) now applies to the second expression with

$$\rho_a + \rho_b = \frac{1}{r} + \sqrt{\frac{1}{r^2} - 1}$$

and

$$\omega = \sqrt{1 + \frac{1}{r}}.$$

□

Acknowledgments. We would like to acknowledge many individuals for their assistance as we worked on this manuscript. First, the ICME students put up with countless hours of discussion, debate, and excitement as we worked on this paper. Second, the Center for Computational Approaches to Digital Stewardship at Stanford University, headed by Margot Gerritsen and Amin Saberi, graciously allowed us to use one of their computers for our numerical experiments. After presenting a less-refined version of these ideas at the 2007 Workshop on Algorithms for the Webgraph, we received much helpful feedback. We would like to thank the audiences of the Stanford Linear Algebra and Optimization Seminar as well as the Uncertainty Quantification Group for additional feedback. Finally, we thank Michael Saunders for a careful reading of this manuscript.

References

[Andersen et al. 06] Reid Andersen, Fan Chung, and Kevin Lang. "Local Graph Partitioning Using PageRank Vectors." In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pp. 475–486. Washington, DC: IEEE Computer Society, 2006.

[Arasu et al. 02] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin. "PageRank Computation and the Structure of the Web: Experiments and Algorithms." In *Poster Proceedings of the 11th International Conference on the World Wide Web*. Available at http://www2002.org/CDROM/poster/173.pdf, 2002.

[Asmussen and Glynn 07] Søren Asmussen and Peter W. Glynn. *Stochastic Simulation: Algorithms and Analysis*. New York: Springer, 2007.

[Avrachenkov et al. 07] Konstantin Avrachenkov, Nelly Litvak, and Kim Son Pham. "Distribution of PageRank Mass among Principle [sic] Components of the Web." In *Algorithms and Models for the Web-Graph: 5th International Workshop, WAW 2007, San Diego, CA, USA, December 11–12, 2007, Proceedings*, Lecture Notes in Computer Science 4863, edited by Anthony Bonato and Fan Chung Graham, pp. 16–28. New York: Springer, 2007.

[Baeza-Yates et al. 06] Ricardo Baeza-Yates, Paolo Boldi, and Carlos Castillo. "Generalizing PageRank: Damping Functions for Link-Based Ranking Algorithms." In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 308–315. New York: ACM Press, 2006.

[Becchetti et al. 08] Luca Becchetti, Carlos Castillo, Debora Donato, Ricardo Baeza-Yates, and Stefano Leonardi. "Link Analysis for Web Spam Detection." *ACM Trans. Web* 2:1 (2008), 1–42.

[Berkin et al. 08] Pavel Berkhin, Usama M. Fayyad, Pabhakar Raghavan, and Andrew Tomkins. "User-Sensitive PageRank." United States Patent Application 20080010281, 2008.

[Boldi 05] Paolo Boldi. "TotalRank: Ranking without Damping." In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web* , pp. 898–899. New York: ACM Press, 2005.

[Boldi and Vigna 04] Paolo Boldi and Sebastiano Vigna. "The Webgraph Framework I: Compression Techniques." In *Proceedings of the 13th International Conference on the World Wide Web*, pp. 595–602. New York: ACM Press, 2004.

[Boldi and Vigna 05] Paolo Boldi and Sebastiano Vigna. "Codes for the World Wide Web." *Internet Mathematics* 2:4 (2005), 407–429.

[Boldi et al. 04] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. "UbiCrawler: A Scalable Fully Distributed Web Crawler." *Software: Practice & Experience* 34 (2004), 711–726.

[Boldi et al. 05] Paolo Boldi, Massimo Santini, and Sebastiano Vigna. "PageRank as a Function of the Damping Factor." In *Proceedings of the 14th International Conference on the World Wide Web*. New York: ACM Press, 2005.

[Boldi et al. 07] Paolo Boldi, Roberto Posenato, Massimo Santini, and Sebastiano Vigna. "Traps and Pitfalls of Topic-Biased PageRank." In *Algorithms and Models for the Web-Graph: Fourth International Workshop, WAW 2006, Banff, Canada, November 30–December 1, 2006. Revised Papers*, Lecture Notes in Computer Science 4936, pp. 107–116. New York: Springer, 2007.

[Boldi et al. 09] Paolo Boldi, Massimo Santini, and Sebastiano Vigna. "PageRank: Functional Dependencies." *ACM Trans. Inf. Syst.* 27:4 (2009), 1–23.

[Castillo et al. 06] Carlos Castillo, Debora Donato, Luca Becchetti, Paolo Boldi, Stefano Leonardi, Massimo Santini, and Sebastiano Vigna. "A Reference Collection for Web Spam." *SIGIR Forum* 40:2 (2006), 11–24.

[Catledge and Pitkow 95] Lara D. Catledge and James E. Pitkow. "Characterizing Browsing Strategies in the World-Wide Web." *Computer Networks and ISDN Systems* 27:6 (1995), 1065–1073.

[Chan et al. 83] Tony F. Chan, Gene H. Golub, and Randall J. LeVeque. "Algorithms for Computing the Sample Variance: Analysis and Recommendations." *The American Statistician* 37:3 (1983), 242–247.

[Chen et al. 07] P. Chen, H. Xie, S. Maslov, and S. Redner. "Finding Scientific Gems with Google's PageRank Algorithm." *Journal of Informetrics* 1:1 (2007), 8–15.

[Christie et al. 02] M. Christie, S. Subbey, and M. Sambridge. "Prediction under Uncertainty in Reservoir Modeling." *Journal of Petroleum Science and Engineering* 44:1–2 (2004), 143–153.

[Constantine and Gleich 07] Paul G. Constantine and David F. Gleich. "Using Polynomial Chaos to Compute the Influence of Multiple Random Surfers in the PageRank Model." In *Algorithms and Models for the Web-Graph: 5th International Workshop, WAW 2007, San Diego, CA, USA, December 11–12, 2007, Proceedings*, Lecture Notes in Computer Science 4863, edited by Anthony Bonato and Fan Chung Graham, pp. 82–95. New York: Springer, 2007.

[Constantine et al., to appear] Paul G. Constantine, David F. Gleich, and Gianluca Iaccarino. "Spectral Methods for Parameterized Matrix Equations." *SIAM Journal on Matrix Analysis and Applications*, to appear.

[Cuil 09] Cuil. "Cuil FAQs (Item 10)." Available at http://www.cuil.com/info/faqs/, 2009.

[Davis and Rabinowitz 84] P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration*. New York: Academic Press, 1984.

[Del Corso et al. 05] Gianna M. Del Corso, Antonio Gullí, and Francesco Romani. "Fast PageRank Computation via a Sparse Linear System." *Internet Mathematics* 2:3 (2005), 251–273.

[Eiron et al. 04] Nadav Eiron, Kevin S. McCurley, and John A. Tomlin. "Ranking the Web Frontier." In *Proceedings of the 13th International Conference on the World Wide Web*, pp. 309–318. New York: ACM Press, 2004.

[Farahat et al. 06] Ayman Farahat, Thomas LoFaro, Joel C. Miller, Gregory Rae, and Lesley A. Ward. "Authority Rankings from HITS, PageRank, and SALSA: Existence, Uniqueness, and Effect of Initialization." *SIAM Journal on Scientific Computing* 27:4 (2006), 1181–1201.

[Freschi 07] Valerio Freschi. "Protein Function Prediction from Interaction Networks Using a Random Walk Ranking Algorithm." In *Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering*, pp. 42–48. Los Alamitos, CA: IEEE Press, 2007.

[Gautschi 02] Walter Gautschi. "OPQ: A MATLAB Suite of Programs for Generating Orthogonal Polynomials and Related Quadrature Rules." Available at http://www.cs.purdue.edu/archives/2002/wxg/codes/OPQ.html, 2002.

[Gautschi 04] *Orthogonal Polynomials: Computation and Approximation*. Oxford: Oxford University Press, 2004.

[Ghanem and Red-Horse 99] R. G. Ghanem and J. Red-Horse. "Propagation of Uncertainty in Complex Physical Systems Using a Stochastic Finite Elements Approach." *Physica D* 133 (1999), 137–144.

[Glaser et al. 07] Andreas Glaser, Xiangtao Liu, and Vladimir Rokhlin. "A Fast Algorithm for the Calculation of the Roots of Special Functions." *SIAM Journal on Scientific Computing* 29:4 (2007), 1420–1438.

[Gleich et al. 10a] David F. Gleich, Paul G. Constantine, Abraham Flaxman, and Asela Gunawardana. "Tracking the Random Surfer: Empirically Measured Teleportation Parameters in PageRank." In *Proceedings of the 19th International Conference on World Wide Web*, pp. 381–390. New York: ACM Press, 2010.

[Gleich et al. 10b] David F. Gleich, Andrew P. Gray, Chen Greif, and Tracy Lau. "An Inner–Outer Iteration for PageRank." *SIAM Journal of Scientific Computing* 32:1 (2010), 349–371.

[Golub and Greif 06] Gene H. Golub and Chen Greif. "An Arnoldi-Type Algorithm for Computing PageRank." *BIT Numerical Mathematics* 46:4 (2006), 759–771.

[Golub and Welsch 69] Gene H. Golub and John H. Welsch. "Calculation of Gauss Quadrature Rules." *Mathematics of Computation* 23:106 (1969), 221–230.

[Gyöngyi et al. 04] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. "Combating Web Spam with TrustRank." In *Proceedings of the 30th International Conference on Very Large Databases*, pp. 576–587. Toronto: VLDB Endowment, 2004.

[Horn and Serra-Capizzano 07] Roger A. Horn and Stefano Serra-Capizzano. "A General Setting for the Parametric Google Matrix." *Internet Mathematics* 3:4 (2007), 385–411.

[Huberman et al. 98] Bernardo A. Huberman, Peter L. T. Pirolli, James E. Pitkow, and Rajan M. Lukose. "Strong Regularities in World Wide Web Surfing." *Science* 280:5360 (1998), 95–97.

[Ipsen and Selee 07] Ilse C. F. Ipsen and Teresa M. Selee. "PageRank Computation, with Special Attention to Dangling Nodes." *SIAM Journal on Matrix Analysis and Applications* 29:4 (2007), 1281–1296.

[Kamvar et al. 03] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. "Extrapolation Methods for Accelerating PageRank Computations." In *Proceedings of the 12th International Conference on the World Wide Web*, pp. 261–270. New York: ACM Press, 2003.

[Karande et al. 90] Chinmay Karande, Kumar Chellapilla, and Reid Andersen. "Speeding Up Algorithms on Compressed Web Graphs." In *WSDM '09: Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pp. 272–281. New York: ACM Press, 2009.

[Katz 53] Leo Katz. "A New Status Index Derived from Sociometric Analysis." *Psychometrika* 18:1 (1953), 39–43.

[Langville and Meyer 06] *Google's PageRank and Beyond: The Science of Search Engine Rankings.* Princeton: Princeton University Press, 2006.

[Lee et al. 07] Chris P. Lee, Gene H. Golub, and Stefanos A. Zenios. "A Two-Stage Algorithm for Computing PageRank and Multistage Generalizations." *Internet Mathematics* 4:4 (2007), 299–327.

[Lin et al. 09] Yiqin Lin, Xinghua Shi, and Yimin Wei." "On Computing PageRank via Lumping the Google Matrix." *Journal of Computational and Applied Mathematics* 224:2 (2009), 702–708.

[Liu et al. 08] Yuting Liu, Bin Gao, Tie-Yan Liu, Ying Zhang, Zhiming Ma, Shuyuan He, and Hang Li. "BrowseRank: Letting Web Users Vote for Page Importance." In *SIGIR '08: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 451–458. New York: ACM Press, 2008.

[Mathelin et al. 05] Lionel Mathelin, M. Yousuff Hussaini, and Thomas A. Zang. "Stochastic Approaches to Uncertainty Quantifcation in CFD Simulations." *Numer. Algorithms* 38 (2005), 209–236.

[Meyer 00] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra.* Philadelphia: SIAM, 2000.

[Moler 04] Cleve B. Moler. *Numerical Computing with Matlab.* Philadelphia: SIAM, 2004.

[Morrison et al. 05] Julie L. Morrison, Rainer Breitling, Desmond J. Higham, and David R. Gilbert. "GeneRank: Using Search Engine Technology for the Analysis of Microarray Experiments." *BMC Bioinformatics* 6:1 (2005), 233.

[Najork et al. 07] Marc A. Najork, Hugo Zaragoza, and Michael J. Taylor. "HITS on the Web: How Does It Compare?" In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 471–478. New York: ACM Press, 2007.

[Page et al. 99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. "The PageRank Citation Ranking: Bringing Order to the Web." Techinical report, Stanford University, 1999.

[Serra-Capizzano 05] Stefano Serra-Capizzano. "Jordan Canonical Form of the Google Matrix: A Potential Contribution to the PageRank Computation." *SIAM Journal on Matrix Analysis and Applications* 27:2 (2005), 305–312.

[Singh et al. 07] Rohit Singh, Jinbo Xu, and Bonnie Berger. "Pairwise Global Alignment of Protein Interaction Networks by Matching Neighborhood Topology." In *Research in Computational Molecular Biology: 11th Annual International Conference, RECOMB 2007, Oakland, CA, USA, April 21–25, 2007, Proceedings*, Lecture Notes in Computer Science 4453, pp. 16–31. New York: Springer, 2007.

[Thelwall 03] Mike Thelwall. "A Free Database of University Web Links: Data Collection Issues." *International Journal of Scientometrics, Informetrics and Bibliometrics* 6/7:1 (2003), paper 2.

[Trefethen 08] Lloyd N. Trefethen. "Is Gauss Quadrature Better Than Clenshaw–Curtis?" *SIAM Review* 50:1 (2008), 67–87.

[Trefethen et al. 09] L. N. Trefethen, N. Hale, R. B. Platte, T. A. Driscoll, and R. Pachón. "Chebfun Version 3." Available at http://www.maths.ox.ac.uk/chebfun/, 2009.

[Various 08] Various. "Wikipedia XML Database Dump from April 2, 2007." Accessed from http://en.wikipedia.org/wiki/Wikipedia:Database_download, 2008.

[Wang 02] Minhua Wang. "A Significant Improvement to Clever Algorithm in Hyperlinked Environment." In *Paper Proceedings of the 11th International Conference on World Wide Web*. Available at http://www2002.org/CDROM/poster/171.pdf, 2002.

[White and Drucker 07] Ryan W. White and Steven M. Drucker. "Investigating Behavioral Variability in Web Search." In *Proceedings of the 16th International Conference on World Wide Web*, pp. 21–30. New York: ACM Press, 2007.

[Witten and Frank 05] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco: Morgan Kaufmann, 2005.

[Xue et al. 03] Gui-Rong Xue, Hua-Jun Zeng, Zheng Chen, Wei-Ying Ma, HongJiang Zhang, and Chao-Jun Lu. "User Access Pattern Enhanced Small Web Search." Poster presented at the 12th International Conference on World Wide Web, Budapest, Hungary, May 20–24, 2003.

[Zhang et al. 04] Hui Zhang, Ashish Goel, Ramesh Govindan, Kahn Mason, and Benjamin Van Roy. "Making Eigenvector-Based Reputation Systems Robust to Collusion." In *Algorithms and Models for the Web-Graph: Third International Workshop, WAW 2004, Rome, Italy, October 16, 2004, Proceeedings*, Lecture Notes in Computer Science 3243, pp. 92–104. New York: Springer, 2004.

[Zhou et al. 05] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. "Learning from Labeled and Unlabeled Data on a Directed Graph." In *ICML '05: Proceedings of the 22nd International Conference on Machine Learning*, pp. 1036–1043. New York: ACM Press, 2005.

[Zwillinger et al. 96] Daniel Zwillinger, Steven G. Krantz, and Kenneth H. Rosen, editors. *Standard Mathematical Tables and Formulae*. Boca Raton, FL: CRC Press, 1996.

Paul G. Constantine, Sandia National Labs, New Mexico, P.O. Box 5800, Albuquerque, NM 87185-1318 (pconsta@sandia.gov)

David F. Gleich, Sandia National Labs, New Mexico, P.O. Box 5800, Albuquerque, NM 87185-9159 (dfgleic@sandia.gov)

Received January 21, 2010; accepted May 24, 2010.