Control and Cybernetics

vol. **40** (2011) No. 2

Accelerating PageRank computations*

by

Sławomir T. Wierzchoń[1,2], Mieczysław A. Kłopotek[1,3], Krzysztof Ciesielski[1], Dariusz Czerski[1] and Michał Dramiński[1]

[1] Institute of Computer Science, Polish Academy of Sciences
Ordona 21, 01-237 Warszawa, Poland
[2] Institute of Informatics, University of Gdańsk
Wita Stwosza 57, 80-952 Gdańsk, Poland
[3] Institute of Informatics, University of Natural and Human Sciences
3 Maja 54, 08-110 Siedlce, Poland

Abstract: Different methods for computing PageRank vectors are analysed. Particularly, we note the opposite behavior of the power method and the Monte Carlo method. Further, a method of reducing the number of iterations of the power method is suggested.

Keywords: PageRank, power method, Jacobi iterations, Monte Carlo.

1. Introduction

To improve the quality of search through a collection of Web documents (called also webpages) Brin and Page (1998) proposed an elaborated ranking system. Search results are ordered according to a mixture of content score and popularity score.

The rules used to assign each page its content score are top secret, but they are designed to characterize each document as precisely and exhaustively as possible. Particularly, they take into account the position of each query term in a document (title, anchor, URL, plain text large font, plain text small font, bold font, etc.), the number of times the query terms appear in a document, and so on – see Brin and Page (1998, Section 5). As noted by Langville and Meyer (2006b, Section 4) Google uses over a hundred such metrics to characterize a document from different points of view.

On the other hand, the popularity score reflects information that can be inferred about a document, but is not contained within it. Brin and Page (1998) assumed that a webpage is important if it is pointed to by other important webpages. This idea gave birth to the notion of PageRank, discussed later in

*Submitted: October 2010; Accepted: June 2011.

Section 2. From a formal point of view the process of assigning PageRank values qualifies as that of spectral ranking (Vigna, 2010), i.e. a technique that applies the theory of linear maps (eigenvectors in our case) to matrices that represent a relationships among entities (webpages in our case).

As these matrices are extremely large and sparse, we need special methods to process them in a manageable way. These techniques are reviewed in Section 3 while in Section 4 some considerations on parallelization of these computations are given. Section 5 concludes the paper.

Notation: Throughout the paper we will use the following notation. Lower case letters r, x, etc., denote column vectors. Particularly, e_n means an n-dimensional vector of ones. The symbol x_i denotes the i-th element of x and x' stands for the transpose of x, i.e. the row vector. $\|x\|$ is the L_1 norm of the vector x, i.e. $\|x\| = \sum_{i=1}^{n} |x_i|$.

Upper case letters A, H, etc. are used to denote square matrices. Particulary I_n stands for the identity (or unit) matrix of size n. If A is a matrix then a_{ij} denotes the element in the i-th row and j-th column. Consequently, A' stands for the transpose of A.

2. PageRank review

Let $\mathcal{N} = \{1, \ldots, n\}$ be the set of indices of webpages and let r stand for a real-valued vector with the components r_j interpreted as the importance weights of these pages. According to Page et al. (1998), the importance of a webpage is determined by the importance of pages linking to it. That is, if $out(j)$ stands for the number of outlinks, i.e. the number of links from page j, and $\mathcal{B}(i)$ is the set of pages pointing into webpage i then

$$r_i = \sum_{j \in \mathcal{B}(i)} \frac{r_j}{out(j)}. \tag{1}$$

To give precise and manageable meaning to such a roughly stated definition let us denote by A the adjacency matrix with the elements

$$a_{ij} = \begin{cases} 1 & \text{if there is a link } i \to j \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

Then the out-degree $out(i) = \sum_{j=1}^{n} a_{ij}$. Next, let us define the so-called hyperlink matrix H with the elements

$$h_{ij} = \begin{cases} a_{ij}/out(i) & \text{if } out(i) > 0 \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

Equation (1) can be written now in concise form $r' = r'H$, showing that r' is in fact a left eigenvector of H with the corresponding eigenvalue $\lambda = 1$. However, in general, r is not unique, i.e. algebraic multiplicity of $\lambda = 1$ is greater than

one, or the equation $r' = r'\lambda H$ cannot be satisfied for $\lambda = 1$. Therefore, H must be modified as described below.

If $out(i) > 0$ then h_{ij} can be interpreted as the probability of jumping from i to a webpage j. Under definition (3) we can imagine a surfer, who, sitting at page i, chooses randomly (with equal probability) one of the outgoing links. To extend this interpretation to the nodes with $out(i) = 0$, called dangling nodes, we replace H by the matrix S defined as

$$S = H + dw' \tag{4}$$

where the entries of vector d are $d_i = 1$ if i is the dangling node and $d_i = 0$ otherwise. The stochastic vector w is called the teleportation vector; it represents the probability with which the surfer jumps from the dangling node to another node $j \in \mathcal{N}$. Traditionally, $w = e_n/n$, i.e. w is the uniform probability vector.

The matrix S is row-stochastic, i.e. $\sum_{i=1}^{n} s_{ij} = 1$ for each row i. Unfortunately, stochasticity of S does not guarantee the uniqueness of r. Thus, the last modification relies upon replacing S by the matrix

$$G = \alpha(H + dw') + (1 - \alpha)e_n v' \tag{5}$$

where v is a stochastic vector called personalization vector, and $\alpha \in (0, 1)$ is the so-called damping factor. Note that $e_n v'$ is a matrix where each row is v'.

The matrix G, called the Google matrix, models a random surfer, who applies with probability α the already described scenario of moving to subsequent pages by randomly choosing an outlink from a current webpage[1], and, if bored, i.e. with probability $1-\alpha$, jumps to a node $j \in \mathcal{N}$ with probability v_j.

According to Boldi et al. (2008) we distinguish between the strongly preferential model, where $w = v$, and the weakly preferential model with $w \neq v$. There is a subtle difference in formal properties of both models. The standard Google matrix, as proposed by Page et al. (1998) is such that $w = v = e_n/n$, and the damping factor $\alpha = 0.85$. In the sequel we will focus primarily on such a matrix.

Note also that the matrix G is[2]: (a) irreducible, i.e. every page is directly connected to every other page, (b) aperiodic, and (c) primitive, because $g_{ij} > 0$ for all $i, j \in \mathcal{N}$. Thus, according to the Perron theorem, there exists the unique and positive eigenvector r such that $r' = r'G$, or equivalently $r = G'r$, with the corresponding eigenvalue $\lambda = 1$ (consult Chapter 8 of Meyer, 2000, for details). Moreover, $\lambda = 1$ is the dominating eigenvalue of G, i.e. $|\lambda_i| < \lambda_1 = 1$ for $i = 2, \ldots, n$. Thus, r is the dominating eigenvector and normalized r is said to be the PageRank. Treating G as the transition matrix of a Markov chain we conclude that the PageRank is nothing but the stationary distribution of this chain (see e.g. Section 8.4 of Meyer, 2000, for details).

[1] Note that if i is the dangling node, then $out(i) = n$.

[2] An unacquainted reader is referred to Langville and Meyer (2006a) or Meyer (2000) for an explanation of these terms.

3. Computing PageRank

In this section we briefly review basic methods used to compute the PageRank vector.

3.1. Power method

It is the simplest and most popular method of computing the dominant eigenvector of the large sparse matrices, see e.g. Section 7.3 of Meyer (2000). Other methods are described e.g. in Chapter 7 of Golub and van Loan (1996). It is an iterative technique. The successive approximations $r^{(k)}$ are computed according to the equation

$$r^{(k+1)} = G' r^{(k)}, \qquad k = 0, 1, \ldots \tag{6}$$

and $r^{(0)}$ is a starting vector with positive entries, such that $\|r^{(0)}\| = 1$. The formula is valid, because in our case $\lambda_1 = 1$. In general, the speed of convergence is linear with coefficient proportional to (λ_2/λ_1), that is the method behaves poorly when $\lambda_2 \approx \lambda_1$. Fortunately, in the case of the Google matrix its second eigenvalue does not exceed the damping factor, i.e. $|\lambda_2| \leq \alpha$ (Haveliwala and Kamvar, 2003). This means that we can control the speed of convergence by choosing the appropriate value of α.

3.1.1. Power method with the Google matrix

To take the advantage of sparsity of matrix H, rewrite the equation (6) as

$$r^{(k+1)} = \alpha H' r^{(k)} + \alpha w \Big[d' r^{(k)} \Big] + (1-\alpha) v \Big[e_n' r^{(k)} \Big]. \tag{7}$$

The i-th component of r takes the form

$$\begin{aligned} r_i^{(k+1)} &= \alpha \textstyle\sum_{j=1}^{n} h_{ji} r_j^{(k)} + \alpha w_i \sum_{j \in \mathcal{D}} r_j^{(k)} + (1-\alpha) v_i \\ &= \alpha s_1 + \alpha s_2 + (1-\alpha) s_3 \end{aligned} \tag{8}$$

where $\mathcal{D}$ is the set of dangling nodes. We see that the third summand must be computed only once, the second component s_2 is computed once for each cycle k, and the first component must be computed for each element $r_i^{(k+1)}$ individually. To get further simplification of the above formula, note two, almost obvious, identities (Gleich and Zhukov, 2005):

$$\begin{aligned} e_n' r^{(k)} &= \|r^{(k)}\| \\ d' r^{(k)} &= \|r^{(k)}\| - \|H' r^{(k)}\|. \end{aligned} \tag{9}$$

Substituting (9) into (7), noting that $\|r^{(k)}\| = 1$ and making the generally used assumption of $v = w$ we obtain

$$r^{(k+1)} = \alpha H' r^{(k)} + v \Big[1 - \alpha \|H' r^{(k)}\| \Big] \tag{10}$$

what justifies the Algorithm 1.

Algorithm 1 Computing PageRank by the power method

1: $k = 0$, $r^{(k)} = v$, $\epsilon = 10^{-8}$
2: **repeat**
3: $\quad r^{(k+1)} = \alpha H' r^{(k)}$
4: $\quad w = 1 - \|r^{(k+1)}\|$
5: $\quad r^{(k+1)} = r^{(k+1)} + wv$
6: $\quad \delta = \|^{(k+1)} - r^{(k)}\|$
7: $\quad k = k + 1$
8: **until** $\delta < \epsilon$
9: **return** $r^{(k+1)}$

The condition $\delta < \epsilon$ in line 8 is a commonly used stopping criterion. Other criteria are discussed in Subsection 3.1.2.

As H is sparse, the time and space complexity of the multiplication $\alpha H' r^{(k)}$ can be reduced substantially by representing H as the product $O^{-1}A$ where O is a diagonal matrix, such that $o_{ii}^{-1} = 1/out(i)$ if i is a nondangling node and $o_{ii}^{-1} = 0$ otherwise, see Langville and Meyer (2006a, p. 76). It suffices to store the main diagonal of O only. That is, we need a real-valued vector deg with n entries: $deg(i) = o_{ii}^{-1}$. To avoid permanent multiplications by the damping factor, we can set $deg(i) = \alpha \cdot o_{ii}^{-1}$. Lastly, A can be stored, e.g., as a vector; its i-th element is a list of integers representing the outgoing links. That is, we need to store only $nnz(A)$ integers, where $nnz(A)$ stands for the number of nonzero elements of A. Such a representation requires only n multiplications and $nnz(A)$ additions (instead of $O(n^2)$, required by a standard setting).

To be more concrete, assume that the matrix A is stored as a two-dimensional array of integers with varying number of columns in each row. `Java`, for instance, allows such a representation: `int[][] A = new int[n][]`. Also let deg be defined as the array `double[] deg = new double[n]` with entries `deg[i] =` α`/A[i].length` if `A[i].length>0` and 0 otherwise. Now, the matrix-vector multiplication $y = A'r$ is implemented (in `Java`) as

```
for (int i=0; i<n; i++) {
    double mult = r[i]*deg[i];
    for (int j=0; j<A[i].length; j++) {
        int k = A[i][j];
        y[k] = y[k] + mult;
    }
}
```

3.1.2. How many iterations?

PageRank values are used to order webpages. We say that the page p_i precedes page p_j, iff $r_i \geq r_j$. Formally, to obtain PageRank values with p significant

digits we need (Stewart, 1994, p. 156)

$$k_{max} = \left\lceil \frac{-p}{\log_{10} \alpha} \right\rceil \tag{11}$$

iterations. For instance, if $\alpha = 0.85$ we need 15 iterations per digit. Usually the power method halts if $\|r^{(k+1)} - r^{(k)}\| < 10^{-p}$. Such a criterion allows for reduction of the number of iterations, e.g. if $p = 8$ then $k_{max} = 141$, but using the later criterion in case of `stanford.edu` dataset[3] we need only 91 iterations. Haveliwala (1999) noted also that the exact values of PageRank are not so important as the correct ordering of the webpages assessed by this vector. On some datasets as few as 10 iterations guarantee a good approximate ordering.

To illustrate this phenomenon consider the dataset `stanford.dat`. To compare the approximated ordering with the exact one we used two measures based on Kendall τ rank correlation, see Section 5.1 in Fogaras and Rácz (2004) and Fagin et al. (2004) for review of other measures.

Let $T(l)$ denote the set of pages with l highest PageRank values in vector r and let $\widehat{T}(l)$ be the set of pages with l highest approximated PageRank values in vector $\widehat{r}$ (returned, e.g., by the power method after $k \ll k_{max}$ iterations). The relative aggregated goodness, RAG, introduced by Singitham et al. (2004), measures how well the approximate top-l set performs in finding the set of pages with high values of PageRank. It is defined as

$$RAG(l) = \frac{\sum_{j \in \widehat{T}(l)} r_j}{\sum_{j \in T(l)} r_j}. \tag{12}$$

The second index measures the precision of returning the top-l set in classical information retrieval terminology

$$Prec(l) = \frac{1}{l} |\widehat{T}(l) \cap T(l)|. \tag{13}$$

Fig. 1 shows how well the set $\widehat{T}(l)$ approximates the exact set $T(l)$ for different values of l. To make the figure readable, we present the results for $l \leq 2000$ only, but obviously, the two indices tend to 1 when l approaches n. The two sets $\widehat{T}(l)$ were obtained by halting the power method after 10 and 15 iterations. The exact set was obtained after $k_{max} = 91$ iterations. In this case the elapsed time is 11.782 seconds. On the other hand, 15 iterations take only 2.078 seconds. This shows that almost fivefold reduction of the time produces almost satisfactory results.

Another interesting feature of the power method is observed when we analyze the absolute error $\delta_j = |\widehat{r}_j - r_j|$ — see left Fig. 2. Here, both vectors are sorted

[3]It contains about 2.3 million links among 281903 pages. This dataset can be downloaded from the webpage `http://kamvar.org/personalized_search`.

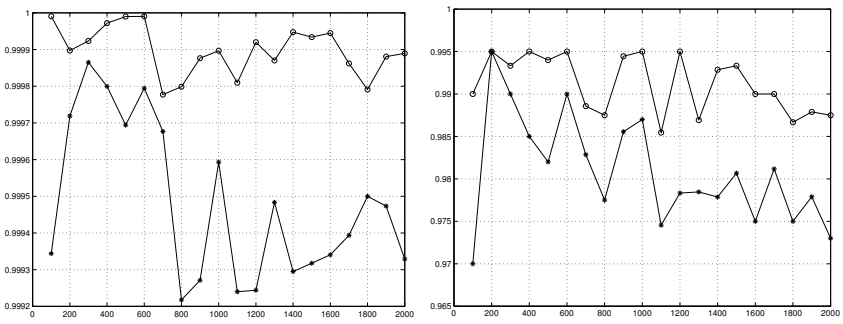

Figure 1. Relative aggregated goodness (left) and precision (right) of returning the top-l set for $l = 100, 200, \ldots, 2000$. The approximate sets $\widehat{T}(l)$ were deduced from the vectors $\widehat{r}$ obtained after $k = 10$ (marked with $*$) and $k = 15$ (marked with $\circ$) iterations of the power method

in descending order imposed by the exact vector r. A lesson taken from this picture is that the elements with smaller values of PageRank converge faster to exact values than those corresponding to larger values. This observation gave an impulse to the so-called adaptive methods, developed by Kamvar et al. (2004). However, the experiments reported by Thorson (2004) show that the adaptive method takes more iterations to converge to a fixed threshold. Good news is that the average cost in computation time for each iteration of the adaptive method is lower for large webs.

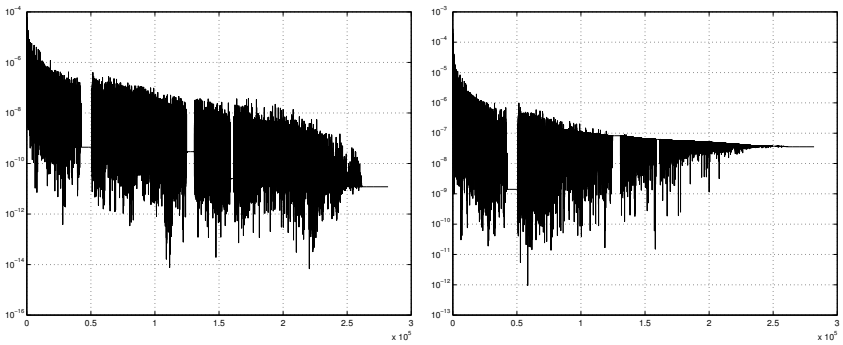

Figure 2. Absolute error of PageRank vector approximation by the vector $\widehat{r}$ returned after 15 iterations of the power method (left) and Jacobi method (right)

3.1.3. Some generalizations of PageRank

A vital problem when computing PageRank values is that of choosing appropriate value of the damping factor. If $\alpha = 0$, then the link structure is totally ignored and r becomes the uniform probability distribution (under standard Google setting). On the other hand, if $\alpha \to 1$ then all the importance is assigned to few pages, forming so-called "rank sinks". To avoid the problem of choosing appropriate value of α, Boldi (2005) proposed the so-called TotalRank

$$T = \int_0^1 r(\alpha) d\alpha. \tag{14}$$

A manageable formula for computing T refers to the Maclaurin expansion of PageRank for a given value α

$$r(\alpha) = \sum_{k=0}^{\infty} \Delta_k(\alpha_0)\alpha^k = r^{(0)}(\alpha_0) + \sum_{k=1}^{\infty} \Delta_k(\alpha_0)\alpha^k \tag{15}$$

where the coefficients $\Delta_k = [r^{(k)}(\alpha_0) - r^{(k-1)}(\alpha_0)]/\alpha_0^k$, $\Delta_0 = r^{(0)}(\alpha_0) = e_n/n$, are generated through successive iterations of the power method, and $\alpha_0 < 1$ is a fixed (but arbitrary) value of the damping factor. With such a representation we easily state that

$$T = \sum_{k=0}^{\infty} \int_0^1 \Delta_k(\alpha_0)\alpha^k d\alpha = \sum_{k=0}^{\infty} \frac{\Delta_k(\alpha_0)}{k+1}. \tag{16}$$

A possible application of the formula (15) is that we compute $r(\alpha_0)$ first. If α_0 is close to zero, this takes few iterations only. Next, the values $r(\alpha)$ for the appropriate value of the damping factor can be easily determined. In fact, all these computations can be performed simultaneously with an extra vector to accumulate total ranks. One should be cautious, however. Suppose that $r(\alpha)$ is computed as the sum of K components of (15). Then, even if $\alpha_0 = \alpha$, then $r(\alpha) = r^{(K)}(\alpha)$. It means that the final Maclaurin series produces an error even if $\alpha_0 = \alpha$. However, PageRank approximations obtained in this way are more accurate than obtained by lowered iterations of the power method. Fig. 3 illustrates this conjecture.

Note that with such a method the maximal absolute error is 0.000111 while the approximation obtained by 20 iterations of the power method produces the absolute error of 0.009607. Further, the first method requires almost the same time (1797 ms) as the second (1610 ms).

The notion of TotalRank was further extended by Constantine and Gleich (2009). They assumed that each user has his/her individual damping factor α_i, i.e. α can be modeled as a random variable $\mathcal{A}$. Hence, the PageRank vector $r(\mathcal{A})$ is a random vector, and a new ranking measure `RAPr` (Random Alpha PageRank) can be synthesized from its statistics. Particularly, if $\mathcal{A}$ is the uniform random variable, then RAPr is identical with TotalRank.

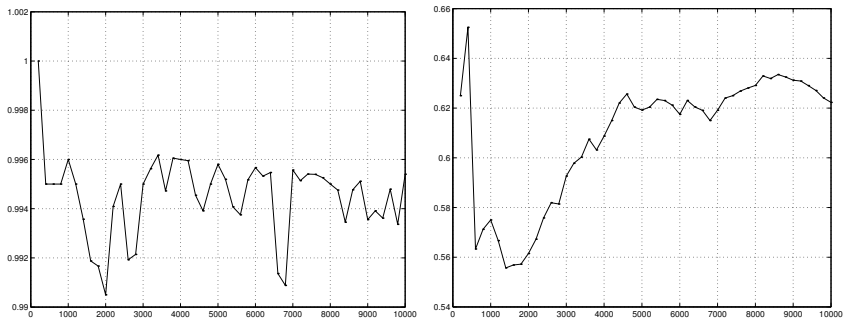

Figure 3. Precision of returning the top-l set for $l = 200, 400, \ldots, 10000$. Left: PageRank values are approximated by the Laurent series of the 20^{th} order. Right: PageRank values produced after 20 iterations of the power method.

3.2. Linear algebra

Langville and Meyer (2006a) show in Section 7.3 of their monograph that solving the linear system

$$(I_n - \alpha H')x = v \tag{17}$$

and letting $r = x/\|x\|$ produces the PageRank vector. A similar result was proven by DelCorso et al. (2005).

Such a result offers another interpretation for the PageRank. To answer the question of existence of x when solving (17) we should note that the matrix $M = (I_n - \alpha H')$ is strictly diagonally dominant[4], i.e. $|m_{ii}| > \sum_{j \neq i} |a_{ij}|$ for each row i. Thus, it is nonsingular.

Interestingly, for small problems the direct method (17) is much faster than the power method. Hence, there is hope that by introducing smart representation and using a "good" iterative method it is possible to obtain satisfactory results for even larger problems. Other methods of potential use in finding the dominating eigenvector of a stochastic matrix are reviewed by Stewart (1994).

3.2.1. PageRank problem as the sum of geometric series

Noting that the inverse of $(I_n - \alpha H')$ can be computed as the Neumann series, see e.g. Meyer (2000) p. 126, equation (17) can be rewritten in the form

$$x = (I_n - \alpha H')^{-1} v = \sum_{j=0}^{\infty} \left[\alpha^k (H')^k\right] v. \tag{18}$$

As $(\alpha H')^k v = (\alpha H') \cdot [(\alpha H')^{k-1} v]$, $k \geq 1$, we obtain a simple algorithm of computing the PageRank

[4]Consult examples 4.3.3 and 7.1.6 in Meyer (2000) for a deeper discussion of this notion.

Algorithm 2 Computing PageRank as a sum of the power series

1: $q = v$, $x = v$
2: **for** $i = 1$ to k_{max} **do**
3: $\quad x = (\alpha H')x$
4: $\quad q = q + x$
5: **end for**
6: **return** $r = q/\|q\|$

Chung (2007) treats (18) as the definition of PageRank and introduces "improved" definition of a heat kernel pagerank

$$\rho = e^{-t} \sum_{k=0}^{\infty} \frac{t^k}{k!} (H')^k v \tag{19}$$

where $t > 0$ is a parameter (temperature). With this concept he introduces then a graph partitioning algorithm for which the running time is proportional to the size of the targeted volume (and not to the size of the whole graph).

3.2.2. PageRank problem as a linear system

Rewriting equation (17) as $r = \alpha H' r + v$ we reformulate PageRank problem as that of solving a set of linear equations. The condition number of matrix M (in L_1 norm) is $\kappa_1(M) \leq (1+\alpha)/(1-\alpha)$; this inequality is strict iff from each node there is a direct path to a dangling node – consult Section 4 in DelCorso et al. (2005) for a rigorous proof of these statements. This shows that (17) becomes "harder" when $\alpha \to 1$.

As matrix M is very large and sparse, again iterative techniques are of primary interest. Generally, we distinguish between older and simpler stationary iterative methods, like Jacobi or Gauss-Seidel methods, and much efficient but more complicated non-stationary methods – consult Barrett et al. (1994) for a comprehensive review. Gleich and Zhukov (2005) state that when choosing an appropriate technique we should consider if it works with nonsymmetric matrices and how easily parallelizable it is.

Recent experiments with parallel PageRank computations show that "although the nonstationary methods have the highest average convergence rate and fastest convergence by number of iterations, on some graphs, the actual run time can be longer than the run time for simple power iterations" (Gleich and Zhukov, 2005, Sect. 5). In general, the convergence of nonstationary methods strongly depends on the Web graph structure and is non-monotonic. On the other hand the power and Jacobi methods have approximately the same rate and the most stable convergence pattern (Gleich and Zhukov, 2005).

Thus, of the stationary methods, Jacobi iterations are most frequently used. The method is simple in implementation and easy parallelizable. If the diagonal

elements of matrix H are equal zero[5], it takes very attractive form shown in Algorithm 3. Here the fast method of matrix-vector multiplication from section 3.1.1 can be used again.

Algorithm 3 Computing PageRank by the Jacobi method

1: $k = 0$, $x = v$, $\epsilon = 10^{-8}$
2: **repeat**
3: $y = \alpha H' x + v$
4: $\delta = \|y - x\|$
5: $x = y$
6: $k = k + 1$
7: **until** $\delta < \epsilon$
8: **return** $r = x/\|x\|$

Like the power method it quickly converges to correct values – see the right Fig. 2, although absolute error (in case of `stanford.dat` dataset) is slightly larger. While with power method it varies in the range $[8.1854 \cdot 10^{-9}, 9.2394 \cdot 10^{-5}]$, now it belongs to the interval $[8.6370 \cdot 10^{-8}, 2.7194 \cdot 10^{-4}]$. However, comparing both pictures we see that the absolute error vanishes "smoother" in the case of Jacobi iterations than in the case of power iterations.

Further acceleration of the Jacobi method is possible by lumping dangling nodes. Kamvar et al. (2003) observed that, e.g., a sample of the Web containing 290 million pages had only 70 million nondangling nodes. Thus, a smart decomposition of H (equivalently of A) allows further saving of computations.

Namely, let $\mathcal{ND} = \mathcal{N} \backslash \mathcal{D}$ denote the set of nondangling nodes and let us permute the rows and columns of H so that the rows corresponding to dangling nodes are at the bottom of the matrix

$$H = \begin{bmatrix} H_{11} & H_{12} \\ 0 & 0 \end{bmatrix}. \tag{20}$$

Here the square matrix H_{11} of size $m = |\mathcal{ND}|$ describes connections among nondagling nodes while rectangular matrix H_{12} of size $m \times (n-m)$ contains the links from nondangling to dangling nodes. Similarly, the personalization vector v is partitioned into nondangling (v_1) and dangling (v_2) sections. Now to find r we must solve the equation, see Langville and Meyer (2006a, Sect. 8.4)

$$(I_n - \alpha H'_{11}) x_1 = v_1 \tag{21}$$

and compute the vector

$$x_2 = \alpha H'_{12} x_1 + v_2. \tag{22}$$

By concatenating x_1 and x_2 into a single vector x we compute $r = r/\|x\|$.

[5]This is not so strange a requirement as self-references are removed before matrix H is formed.

3.3. Monte Carlo methods

Let us rewrite the formula (7) as

$$\pi = (1-\alpha)ve_n'(I_n - \alpha H) \tag{23}$$

with π being the stationary distribution, i.e. $\pi = x^{(k)}$ for all $k \geq K$, where K is a sufficiently large number. This formula suggests a simple way of sampling from PageRank distribution (Avrachenkov et al., 2007). Namely, we construct a random walk (r.w. for brevity) $\{X\}_{t\geq 0}$ that starts from a randomly chosen page. At each step the r.w. terminates with probability $(1-\alpha)$, and makes a transition according to the matrix S with probability α. The end-point of such r.w. has the distribution π.

Careful analysis of such a simple procedure led the authors referred to, to further simplifications. First, to reduce the unnecessary randomness in experiments, each r.w. starts m times from each page. That is, we perform $N = mn$ random walks. Second, to extract full information from all the experiments, π_j is defined as the total number of visits to page j (η_j) multiplied by $(1-\alpha)/N$. Third, when an r.w. reaches a dangling node it jumps with the uniform (in standard model) probability to an arbitrary node. Thus, it is more efficient to terminate the r.w. in such a case. That is, the r.w. is governed by the H matrix and not by the S matrix.

Let $\{Y_t\}_{t\geq 0}$ stand for such a random walk. It can be terminated at each step either with probability $(1-\alpha)$ or when it reaches a dangling node. This leads to Algorithm 4. It counts the number of total visits and the number of visits at each node.

To verify the properties of such an algorithm we simulated $m = 4$ cycles with matrix H specified by the `stanford.dat` dataset. Fig. 4 presents the values of $RAG(l)$ and $Prec(l)$ computed for $l = 10, 20, \ldots, 1000$ after four cycles of the Monte Carlo algorithm. As both indices are almost equal 1 for $l \leq 200$, we can say that even small number of Monte Carlo runs allows for quite exact ordering of the top-l items, with l of reasonable amplitude. This hypothesis was first stated by Avrachenkov et al. (2007).

4. Parallelization

Gleich and Zhukov (2005) experimented with both stationary and nonstationary methods. They compared two classical algorithms, i.e. power and Jacobi iterations with few nonstandard methods, particularly with BiCGSTAB (Biconjugate Gradient Stabilized) and GMRES (Generalized Minimum Residual). In summary they state that their parallel implementation was able to compute the PageRank vector for a 1.5 billion node Web graph on a distributed memory 140 processor RLX cluster in less than 6 minutes.

In this section we briefly describe another idea of parallelization developed within the "cloud computing" formalism.

Algorithm 4 Computing PageRank by the Monte Carlo method

```
1: visited = 0, total = 0
2: for i = 1 to m do
3:   for j = 1 to n do
4:     current = i
5:     done = false
6:     while (not done) do
7:       visited[current] + +
8:       total + +
9:       if ((random() < 1 − α) or (deg[current] = 0)) then
10:        done = true
11:      else
12:        choose new node w among the nodes pointed by current node and
           set current = w
13:      end if
14:    end while
15:  end for
16: end for
17: return r = visited/total
```

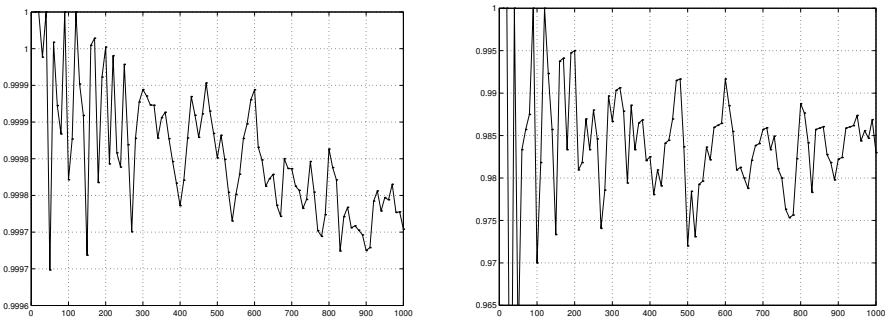

Figure 4. Relative aggregated goodness (left) and precision (right) of returning the top-l set for $l = 10, 20, \ldots, 1000$. The approximate sets $\widehat{T}(l)$ were deduced from the vectors $\widehat{r}$ obtained after four cycles of the Monte Carlo simulations

4.1. MapReduce

As stated by its inventors, Dean and Ghemawat (2008), "MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a `map` function that processes a key/value pair to generate a set of intermediate key/value pairs, and a `reduce` function that merges all intermediate values associated with the same intermediate key".

The "mapper" and "reducer" can be described symbolically as follows:

$$\begin{aligned} \texttt{map:} &\quad <k_1, v_1> \rightarrow [<k_2, v_2>] \\ \texttt{reduce:} &\quad <k_2, [v_2]>] \rightarrow [<k_3, v_3>] \end{aligned}$$

Here $<k_1, v_1>$ denotes the key-value pair and $[\cdot]$ indicates a list of elements.

A real advantage of MapReduce implementation is that a programmer writes only these functions and he/she does not worry about system-level issues such as synchronization, data exchange or fault tolerance.

Hadoop (Lam and Warren, 2009) is an open source implementation (available from `http://hadoop.apache.org`) of this methodology.

4.2. Computing PageRank with MapReduce

To implement Algorithm 1 in the MapReduce formalism, we must apply a double set of map-reduce functions. The first pair realizes matrix-vector multiplication as described in Subsection 3.1.1. First, knowing webpage identifier i, its current PageRank r_i, and the list of outlinks A_i, the mapper returns the set of pairs $<j, w_j>$, where j is a page linked from i and $w_i = r_i/o_i$, with o_i being the length of A_i.

$$\texttt{map:} <i; (r_i, A_i)> \rightarrow [<j, r_i/o_i>]$$

Now the reducer aggregates partial information

$$\texttt{reduce:} <j, [\rho_i])> \rightarrow <j, \sum_i \rho_i>$$

The second pair finishes the job, i.e. the pairs $<j, \sum_i \rho_i>$ are transmitted by the mapper to the reducer that computes their sum (i.e. finds the norm used in line 4 of Algorithm 1 and returns the pairs $<i, r_i>$ computed according to the line 5 of this algorithm).

In a similar fashion we can implement two other algorithms.

5. Final remarks

Various methods of computing PageRank values were discussed in this paper. Interestingly, using the power method or Jacobi iterations small PageRank values converge faster than high values. On the contrary, when using Monte Carlo sampling an opposite phenomenon can be observed. Thus, a smart combination of these two approaches can produce satisfactory results with rather small number of computations.

Unfortunately, the power method is almost insensitive to the initial vector. In other words, improving vector r obtained after a small cycles of Monte Carlo simulations takes almost the same time as starting the power method from an arbitrary vector $r^{(0)}$.

Hopefully, approximating PageRank by the Maclaurin series (15) offers a substantial time reduction.

Acknowledgment

This work is partly supported by Polish State budget scientific research funds under grant POIG.01.01.02-14-013/09 "Adaptive system supporting solving problems based on content analysis of available electronic source".

References

Avrachenkov, K., Litvak, N., Nemirovsky, D. and Osipova, N. (2007) Monte Carlo methods in PageRank computation: When one iteration is sufficient. *SIAM Journal of Numerical Analysis*, **45**(2), 890–904.

Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C. and van der Vorst, H. (1994) *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA.

Boldi, P. (2005) TotalRank: Ranking without damping. In: *Poster Proc. of the 14-th International Conference on the World Wide Web WWW05*. ACM, New York, 898–899.

Boldi, P., Posenato, R., Santini, M. and Vigna, S. (2008) Traps and pitfalls of topic-biased PageRank. In: W. Aiello, A. Broder, J. Janssen, and E. Milios, eds., *Proc. WAW 2006. 4-th Workshop on Algorithms and Models for the Web-Graph*, **LNCS 4936**, Springer, 107—116.

Brin, S. and Page, L. (1998) The anatomy of a large-scale hypertextual Web search engine. In: *Proc. 7-th International World-Wide Web Conference, WWW 1998*. ACM, New York.

Chung, F. (2007) The heat kernel as the PageRank of a graph. *PNAS*, **105**(50), 19735–19740.

Constantine, P.G. and Gleich, D.F. (2009) Random Alpha PageRank. *Internet Mathematics*, **6**(2), 189–236.

Dean, J. and Ghemawat, S. (2008) MapReduce: simplified data processing on large clusters. *Communications of the ACM*, **51** (1), 107–113.

DelCorso, G.M., Gulli, A. and Romani, F. (2005) Fast PageRank computation via a sparse linear system. *Internet Mathematics*, **2**(3), 251–273.

Fagin, R., Kumar, R. and Sivakumar, D. (2004) Comparing top k lists. *SIAM Journal on Discrete Mathematics*, **17**(1), 134 – 160.

Fogaras, D. and Rácz, B. (2004) Towards scaling fully personalized PageRank. In: S. Leonardi, ed., *Proc. of the 3rd Workshop on Algorithms and Models for the Web-Graph (WAW 2004)*, **LNCS 3243**, Springer, 105–117.

Gleich, D. and Zhukov, L. (2005) Scalable Computing for Power Law Graphs: Experience with Parallel PageRank. In: *SuperComputing, SC|05*. ACM, New York.

Golub, G.H. and van Loan, Ch.F. (1996) *Matrix Computation. Third edition*. Hindustan Book Agency, New Delhi, India.

Haveliwala, T. (1999) Efficient Computation of PageRank. Technical Report 1999-31, Stanford InfoLab, `http://ilpubs.stanford.edu:8090/386/`.

Haveliwala, T.H. and Kamvar, S.D. (2003) The Second Eigenvalue of the Google Matrix. Technical report, Stanford University.

Kamvar, S.D., Haveliwala, T.H. and Golub, G.H. (2004) Adaptive methods for the computation of PageRank. *Linear Algebra and its Applications*, **386**, 51–65.

Kamvar, S.D., Haveliwala, T.H., Manning, C.D. and Golub, G.H. (2003) Exploiting the block structure of the Web for computing PageRank. Technical report, Stanford University.

Lam, Ch and Warren, J. (2009) *Hadoop in Action*, Manning.

Langville, A.N. and Meyer, C.D. (2006a) *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press.

Langville, A.N. and Meyer, C.D. (2006b) Information Retrieval and Web Search. In: L. Hogben, R. Brualdi, A. Greenbaum, and R. Mathias, eds., *Handbook of Linear Algebra*, chapter 63, 63–1, CRC Press.

Meyer, C.D. (2000) *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia.

Page, L., Brin, S., Motwani, R. and Winograd, T. (1998) The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project.

Singitham, P.K.C., Mahabhashyam, M.S. and Raghavan, P. (2004) Efficiency-quality tradeoffs for vector score aggregation. In: M.A. Nascimento, M.T. Özsu, D. Kossmann, R.J. Miller, J.A. Blakeley, and K.B. Schiefer, eds., *Proc. of the 30-th International Conf. on Very Large Data Bases*, **30**, 624–635, Morgan Kaufmann, Toronto, Canada.

Stewart, W.J. (1994) *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, Princeton.

Thorson, K. (2004) Modeling the Web and the computation of PageRank. Undergraduate thesis, Hollins University.

Vigna, S. (2010) Spectral ranking. *ArXiv e-prints* (0912.0238).