Запропоновано алгебраїчний метод знаходження оцінок рангів PageRank для сторінок сайтів. Обсяг обчислень запропонованого методу не залежить від значення коефіцієнта демпфірування, що дозволяє отримувати більш точні оцінки рангів PageRank, в порівнянні з аналогами. Відмінною особливістю запропонованого методу є послідовне виконання обчислень одночасно з роботою алгоритму обходу графа. Проведений порівняльний аналіз алгоритмів обходу графів показав, що на відміну від алгоритму пошуку в глибину алгоритм пошуку в ширину дає більш упорядковану матрицю переходів, яка має вигляд блочно Хессенберговской. Використання цієї обставини дозволило істотно скоротити обсяг обчислень запропонованого методу. Отримані рівняння, що описують запропонований метод, мають блочну структуру, яка дозволяє ефективно розподіляти весь обсяг операцій на паралельні обчислювальні потоки. Виходячи з того, що основна частина обчислень може бути виконана під час виконання алгоритму обходу графа, визначені умови, при яких запропонований метод дозволяє отримати оцінку рангів PageRank швидше, ніж відомі ітераційні алгоритми. Областю застосовності розробленого методу в першу чергу є використання його при безпосередній перевірці достовірності розміщення рекламних матеріалів на відповідному веб-ресурсі, тому вона обмежена окремими сайтами або сегментами інтернету з кількістю сторінок не більше 104–105

Ключові слова: граф сайту, ранги сторінок, матриця переходів, коефіцієнт демпфірування, матриця телепортації

UDC 519.688: 519.177
DOI: 10.15587/1729-4061.2018.131275

AN ALGEBRAIC METHOD FOR CALCULATING PAGERANK

V. Vlasyuk
Manager
Vertamedia LLC branch
Admiralskyi ave., 34a, Odessa, Ukraine, 65009
E-mail: vladvlasiuk85@gmail.com
O. Galchonkov
PhD, Associate Professor*
E-mail: o.n.galchenkov@gmail.com
A. Nevrev
PhD, Associate Professor*
E-mail: a.i.nevrev@gmail.com
*Department of Information Systems
Institute of Computer Systems
Odessa National Polytechnic University
Shevchenko ave., 1, Odessa, Ukraine, 65044

1. Introduction

The constantly increasing use of the Internet in all areas of human activity has made it one of the most important places for the development of the advertising business [1]. However, when posting an advertisement, the advertiser does not directly interact with the owners of the websites that host the advertisement. On the way from the advertiser to the website, there is a large number of intermediaries, which include advertising agencies, ad networks [2], arbitrage networks (Trading desks and Ad Exchanges [3]), SSP and DSP platforms [4, 5]. On the one hand, it facilitates the activity of the advertiser, increases the effectiveness of advertising, and reduces its cost. On the other hand, there are additional chances for fraud.

Therefore, it is extremely topical to verify precisely the reliability of posting advertising materials on a relevant web resource [6–8]. As the structure of web resources dynamically changes, one of the primary tasks is to analyse the current structure of a site and the current ranks (importance) of its pages. This can help determine the effectiveness of advertisement posting in the next steps.

The questions of analysing the structure of sites and finding the ranks of their pages have been widely and fully covered in published studies, for example in [9, 10]. However, they are considered separately from each other there. It is assumed that the structure of a site is analysed first, and then the ranks of its pages are determined. This approach leads to the fact that when working with multicore or multiprocessor computers, not all computing powers are used. Therefore, it is important to develop algorithms for determining the ranks of site pages, the main amount of which can be done in parallel with the work of the algorithm for analysing the structure of the site. This will make it possible to use the calculator more fully and get the result faster.

One of the most well-known algorithms for ranking pages of websites is PageRank, proposed in [11]. However, due to the presence of pages with no outbound links, the task of estimating the ranks in accordance with this approach is degenerate. To solve the situation, various types of regularization are used. The greater the level of regularization, the easier it is to obtain a solution, but the greater the deviation of the estimates obtained from the idea proposed in [11]. Therefore, it is important to develop algorithms for finding PageRank estimates that differ from those known by the best ratio of the volume of necessary computations and the level of regularization.

2. Literature review and problem statement

The most effective way to represent the structure of a site is to represent it in the form of a graph the vertices of which are pages of the site, and the edges are defined by the links from one page to another. To determine the ranks of pages, there are many approaches, but the most famous of them is PageRank, proposed in [11]. The main idea of this approach is that the rank of a page u linearly depends on the ranks of the pages that refer to it:

$$PR(u)=\sum_{v\in B_u}\frac{PR(v)}{L(\mathrm{v})}, \tag{1}$$

where $PR(v)$ is the PageRank of a page v; $L(v)$ is the total number of outbound links on the page v; B_u is the set of pages that refer to the page u.

Assuming that there is at least one reference to other pages on each page, equation (1) is a particular case of the vector equation [12]:

$$Av=v, \tag{2}$$

where v is the vector of the ranks of site pages:

$$A=P^{\mathrm{T}},$$

P is the matrix of transitions of a dimension $N\times N$, a non-negative matrix with coefficients:

$$p_{ij}=\begin{cases}1/\deg(i), \text{ if the i-th page has} \\ \text{a link to the j-th page;} \\ 0, \text{ if the i-th page does not have} \\ \text{a link to the j-th page;}\end{cases}$$

$\deg(i)$ is the number of outbound links on the page i,

$$p_{ij}\geq 0 \text{ and } \sum_j p_{ij}=1,$$

N is the number of pages on the site.

The solution of equation (2) is considered either as finding the eigenvector of the matrix A, corresponding to the maximum eigenvalue of this matrix equal to 1, or as a solution of the system of equations [12, 13]:

$$(I-A)v=0, \tag{3}$$

where I is an identity matrix.

From the theory of Markov homogeneous chains, where a similar graph model is used, it is known that equation (2) has a solution, and it is unique (within a multiplier) only if the graph is strongly coupled and aperiodic [9]. The second condition for web graphs is automatic. And the first condition is not fulfilled if the site has the so-called 'hanging' pages – pages that do not have links to other pages. Hanging pages give zero rows in the matrix of transitions P and, accordingly, lead to its degeneracy.

The most popular approach for solving the problem of hanging pages is to modify the matrix P in such a way as if hanging pages contain transitions to all pages [9, 12, 13]:

$$P^{'}=P+\frac{1}{N}D, \tag{4}$$

where N is the total number of pages on the site; D is a matrix of the dimension $N\times N$, whose i-th rows contain ones, and the remaining rows are zero for all i-numbered pages.

In addition to solving the problem of hanging pages to ensure strong coherence of the graph, another modification of the matrix of transitions is made – 'teleportation.' Physically, it is interpreted in the way as if a user who is on any page of the site follows, with a probability α, one of the links available on this page, and with a probability $(1-\alpha)$, goes to any other page:

$$P^{''}=\alpha P^{'}+\frac{(1-\alpha)}{N}J, \tag{5}$$

where J is a matrix with all elements equal to 1.

As a result, rank estimates are found from the solution of the modified equation (2):

$$A^{''}v=v, \tag{6}$$

where v is the vector of the ranks of the site pages, $A^{''}=(P^{''})^{\mathrm{T}}$.

The whole set of approaches to solving equation (6) can be divided into two groups [14] – algebraic methods and iterative methods.

Iterative methods assume a consistent approximation to the desired estimate of the vector of ranks, producing iterations. One of the most popular iterative methods is the power method [15]:

$$v^{k+1}=(A^{''}v^{k})/\left\|A^{''}v^{k}\right\|_1, \tag{7}$$

where v^k is the estimate of the vector of ranks v at the k-th iteration;

$$\|X\|_1=\sum_i |x_i|$$

is the norm of the vector X, equal to the sum of the moduli of its components x_i.

As the initial approximation, we take the vector v^0, all elements of which are equal to $1/N$. Calculations stop after reaching

$$\left\|v^{k+1}-v^{k}\right\|_1\leq\delta, \tag{8}$$

where δ is a small value.

To obtain faster estimates of ranks, either linear algebra methods [15, 16], such as LU decomposition and QR algorithm, or the Jacobi, Lanczos, Arnoldi [17] and Gauss-Seidel methods are used. Or heuristic methods are involved. For example, the extrapolation method, which is based on the power-law method [18], or the Arnoldi method [19]; however, instead of computations on some iterations, there is simply extrapolation to obtain estimates of the next step. Or an adaptive method [20, 21] is used, which is based on the fact that the elements of the vector v^k converge at different rates. Therefore, for the elements that have already practically converged and do not change, no calculations are made. In addition, a large number of studies are devoted to the acceleration of obtaining estimates by iterative methods due to their modification for calculating on multicore or multiprocessor computers [22–27].

For example, in [22], massive parallelism of GPUs is used to calculate PageRank, and in [23], a partitioning and compact representation of a site graph is proposed to match the memory size of single processors in an array of GPUs. In [24], a parallel algorithm for computing PageRank is proposed, based on the use of the distributed programming concept MapReduce and the list of contiguities in multiprocessor arrays. The load balancing capabilities in comparison with memory-based methods in the multiprocessor CUDA architecture are examined in [25]. In [26], a method is proposed to accelerate the calculation of PageRank in the multiprocessor CUDA architecture by searching for special structures in the site graphs that allow parallelizing the computations to the maximum number of threads. In work [27], it is suggested to accelerate the calculation of PageRank on distributed multiprocessor computers due to the use of the MapReduce concept and the Hadoop framework.

Algebraic methods presuppose an exact solution of equation (6) in the form [14] of

$$v=\left[I-\alpha\left(P^T+\frac{1}{N}D^T\right)\right]^{-1}\cdot\frac{(1-\alpha)}{N}e, \qquad (9)$$

where e is a vector of a dimension N, all elements of which are equal to 1; I is the identity matrix of the size $N\times N$.

In search programs with very large transition matrices, with N of the order of 10^7, iterative solutions of (2) are used with α=0.85...0.9, where α is also called the damping coefficient. The smaller the value of α, the higher the convergence rate of the iterative methods, but the less the accuracy of the solution for the page ranks in terms of formula (1) [12, 13, 28].

When verifying advertising companies, a particular web resource (site) is analysed; N usually does not exceed 10^4 ... 10^5. It is, therefore, desirable to obtain the most accurate solution of formula (1). In addition, it should be noted that the solution of equation (6) by iterative methods usually begins only after the structure of the web resource is completely determined, that is, when the matrix P is known. At the same time, the bypass of the site graph (definition of its structure) is iteratively performed, and it takes a significant amount of time that could be used to calculate page ranks. This is especially true when all operations are performed on a powerful calculator containing a large number of cores or processors. Therefore, it is of interest to develop a method for calculating page ranks by an algebraic method [14] the volume of operations for which does not depend on the proximity of α to 1 and which takes into account the features of the graph traversal algorithm.

3. The aim and objectives of the study

The aim of the study is to develop a method for calculating page ranks with an algebraic approach that takes into account the features of the graph traversal algorithm.

To achieve the aim, the following tasks are set and solved:

– to analyse the features of the algorithms for traversing graphs and to identify opportunities to reduce the amount of calculations, as well as to produce part of the calculations while the graph traversal algorithm is being used;

– to take into account the structural features of the transition matrix when constructing a step-by-step calculation of the ranks;

– to determine the order of the operations for the method obtained and to find the conditions under which the operations' volume is smaller than with the iterative methods of determining the ranks.

4. Analysis of the features of graphs traversal algorithms

As the structure of websites dynamically changes with time, when verifying advertising companies, it is considered that this structure is a priori unknown and it is necessary to determine it, starting from the main page of a website.

Let us consider two main approaches to the construction of algorithms for traversing graphs [10]. These are a depth-first search (DFS) algorithm and a breadth-first search (BFS) algorithm.

We represent the graph of the target site in the form of

$$G=(V,E),$$

where V is the set of all pages of the site, or the set of all vertices of the graph G; E is the set of the pairs (u, v), where $u,v\in V$ are connected by an edge of the graph G, that is, on the page u there is a link to the page v.

We assume that we know the initial vertex of the graph G $v_{0,0}$ as the main page of the site.

The DFS algorithm is aimed at a maximally rapid movement into the interior of the graph G. Upon falling into some vertex of the graph w_j, the algorithm forms a set of the vertices of the graph that it has already by passed, D_j:

$$D_j=D_{j-1}+\{w_j\},\quad j=1,2,3,\ \dots \qquad (10)$$

$$D_0=\{w_0\},\ \text{where}\ w_0=v_{0,0}.$$

Next, we find the set of the vertices of the graph G to which there are links on the page w_j:

$$B_j=\{b_{j,1},b_{j,2},\ \dots\}. \qquad (11)$$

From this set, we remove the vertices which the DFS algorithm has already bypassed:

$$W_j=B_j-(B_j\cap D_j). \qquad (12)$$

From the set W_j, by some rule (for example, the first outgoing link on the page), we select one vertex w_{j+1}, into which the algorithm proceeds in the next step.

If the set W_j is empty, one step back is made to form a new set:

$$W_{j-1}^1=B_{j-1}-(B_{j-1}\cap D_j), \qquad (13)$$

and the vertex w_{j+1} is selected from it.

If the set W_{j-1}^1 is also empty, then it is necessary to roll back one more step and so on. The traversal of the graph is completed when it is impossible to form a nonempty set W_n^m and to roll back to the initial vertex of the graph.

In contrast to the depth-first search (DFS) algorithm, the breadth-first search (BFS) algorithm involves sequential traversal of the graph over layers. By the i-th layer, we mean a subset V_i of vertices of the graph G spaced at the shortest distance to i edges from the vertex $v_{0,0}$. If any of the j-th vertices of the i-th layer of the graph $v_{i,j}$ is hit, the BFS algorithm generates a subset of the vertices of the graph that it has already bypassed – $P_{i,j}$:

$$P_{i,j}=\{v_{i,j}\}+P_{i,j-1},\quad j=1,2,\ \dots\ , \qquad (14)$$

where

$$P_{i,0}=P_{i-1,j\max i},\quad i=1,2,\ \dots\ ,\quad P_{0,0}=v_{0,0},$$

$J\max i$ is the maximum index j for the vertex $v_{i-1,j}$ in the layer V_{i-1}.

Further, the BFS algorithm finds all vertices of the graph G that are separated from the vertex $v_{i,j}$ at a distance of one edge:

$$F_{i,j} = \{f_{i,j1}, f_{i,j2}, \dots\}, \tag{15}$$

the (i+1)-th layer of the vertices of the graph is made up of new detected vertices, separated from the vertices of the i-th layer at a distance of one edge:

$$V_{i+1} = \left\{ (F_{i,j} - (F_{i,j} \cap P_{i,j})) : \forall v_{i,j} \in V_i \right\}. \tag{16}$$

Bypassing of the graph by the BFS algorithm is terminated when the next set V_{i+1} is empty.

From the comparison of the DFS and BFS algorithms, it is clear that the breadth-first algorithm provides more ordered subsets of the graph's vertices – layers. Moreover, proceeding from the rules of layer formation in the BFS algorithm, it is clear that the vertices of the i-th layer have references only to the vertices of the (i+1)-th layer and do not have references to the vertices of the (i+2), (i+3) or other layers. Therefore, when the vertices of the graph are ordered in layers by the BFS algorithm, the transition matrix P will be of a lower blockwise Hessenberg matrix type [29]. An example of the matrix P for eight layers is the following:

$$P = \begin{pmatrix} A_{11} & A_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ A_{21} & A_{22} & A_{23} & 0 & 0 & 0 & 0 & 0 \\ A_{31} & A_{32} & A_{33} & A_{34} & 0 & 0 & 0 & 0 \\ A_{41} & A_{42} & A_{43} & A_{44} & 0 & 0 & 0 & 0 \\ A_{51} & A_{52} & A_{53} & A_{54} & A_{55} & A_{56} & 0 & 0 \\ A_{61} & A_{62} & A_{63} & A_{64} & A_{65} & A_{66} & A_{67} & 0 \\ A_{71} & A_{72} & A_{73} & A_{74} & A_{75} & A_{76} & A_{77} & A_{78} \\ A_{81} & A_{82} & A_{83} & A_{84} & A_{85} & A_{86} & A_{87} & A_{88} \end{pmatrix}. \tag{17}$$

The size of the matrix P is equal to $N \times N$, where N is the total number of pages on the site (the number of vertices in the site graph). A_{ij} denotes blocks containing links between the vertices of the i-th layer and the vertices of the j-th layer. The size of each block A_{ij} is equal to $n_i \times n_j$, where n_i is the number of vertices in the i-th layer of the graph, n_j is the number of vertices in the j-th layer of the graph. Since the matrix of transitions for the BFS algorithm has a block view, and the A_{ij} blocks, for which $j \geq (i+2)$, contain only zero elements, this can be used to develop an algorithm for calculating the page rank of the target site. In addition, it should be noted that the block view of the transition matrix P creates prerequisites for all calculations in the rank estimation algorithm to have a block view. This helps parallelize efficiently the computations to threads for their processing by multicore and multiprocessor computing structures.

5. Development of a modified algorithm for estimating PageRank

We assume that the BFS algorithm is used to bypass the graph of the target site. As a result, we obtain a matrix of transitions P structured with layers.

To ensure the stability of the subsequent computational process as to (4) and (5), we similarly modify the transition matrix P as follows:

$$P^{''} = \alpha P + \frac{\alpha}{N} D + \frac{(1-\alpha)}{N} J, \tag{18}$$

where $0 < \alpha < 1$; J is a matrix of teleportation, with the size $N \times N$, corresponding to the matrix of transitions P; all its elements standing in the places of blocks A_{ij} $(j \leq i+1)$, are equal to 1, and all its elements standing in the places of zero blocks $j > i+1$, are equal to 0; D is a matrix with the size $N \times N$, which corresponds to the 'hanging' pages; for each 'hanging' page (not having outgoing links) with a number m, the corresponding m-th row of the matrix D contains ones in the places corresponding to the blocks A_{ij} with $j \leq i+1$, and zeros in the places corresponding to the zero blocks $(j > i+1)$, and all other elements of the matrix D are equal to zero.

The PageRank vector v of the ranks is defined as the dominant eigenvector of the matrix where $A = \left(P^{''}\right)^T$:

$$Av = v. \tag{19}$$

Taking into account (14), equation (15) is equivalent to the following:

$$\left[I - \alpha \left(P^T + \frac{1}{N} D^T \right) \right] v = \frac{(1-\alpha)}{N} e, \tag{20}$$

where e is a vector of a dimension N, all elements of which are equal to 1; I is the identity matrix of the size $N \times N$.

The algebraic solution of (20) is defined as [12]

$$v = \left[I - \alpha \left(P^T + \frac{1}{N} D^T \right) \right]^{-1} \cdot \frac{(1-\alpha)}{N} e. \tag{21}$$

We introduce the notation for the matrix beeing inverted, obtained at the k-th step of the operation of the BFS algorithm as follows:

$$B_k = I_k - \alpha \left(P_k^T + \frac{1}{N_k} D_k^T \right), \tag{22}$$

where the matrices B_k, I_k, P_k, and D_k have the dimensions of $N_k \times N_k$; N_k is the total number of vertices of the graph obtained by the BFS algorithm in k steps.

$$N_k = \sum_{i=1}^{k} n_i. \tag{23}$$

The matrix B_k, similarly to the form of the matrix P in (17) and taking into account the transposition, is the upper blockwise Hessenberg matrix:

$$B_k = \begin{pmatrix} C_{11}^k & C_{12}^k & C_{13}^k & . & . & . & C_{1(k-2)}^k & C_{1(k-1)}^k & C_{1k}^k \\ C_{21}^k & C_{22}^k & C_{23}^k & . & . & . & C_{2(k-2)}^k & C_{2(k-1)}^k & C_{2k}^k \\ 0 & C_{32}^k & C_{33}^k & . & . & . & C_{3(k-2)}^k & C_{3(k-1)}^k & C_{3k}^k \\ . & . & . & . & & & . & . & . \\ . & . & . & & . & & . & . & . \\ . & . & . & & & . & . & . & . \\ 0 & 0 & 0 & . & . & . & C_{(k-2)(k-2)}^k & C_{(k-2)(k-1)}^k & C_{(k-2)k}^k \\ 0 & 0 & 0 & . & . & . & C_{(k-1)(k-2)}^k & C_{(k-1)(k-1)}^k & C_{(k-1)k}^k \\ 0 & 0 & 0 & . & . & . & 0 & C_{k(k-1)}^k & C_{kk}^k \end{pmatrix}. \tag{24}$$

Then the matrix B_{k+1} can be represented in the block form

$$B_{k+1}=\begin{pmatrix} B_k & \Phi_{k+1} \\ \Theta_{k+1} & C^{k+1}_{(k+1)(k+1)} \end{pmatrix}, \qquad (25)$$

where the matrix Θ_{k+1} has a dimension of $n_{k+1}\times N_k$, and the matrix Φ_{k+1} has a dimension of $N_k\times n_{k+1}$:

$$\Theta_{k+1}=(0\ 0\ 0\ \ldots\ C^{k+1}_{(k+1)k}), \qquad (26)$$

$$\Phi^T_{k+1}=\left(C^{k+1}_{1(k+1)}\ C^{k+1}_{2(k+1)}\ \ldots\ C^{k+1}_{k(k+1)}\right)^{\mathrm{T}}. \qquad (27)$$

In accordance with the Frobenius theorem [29],

$$B^{-1}_{k+1}=\begin{pmatrix} B_k^{-1}+B_k^{-1}\Phi_{k+1}H^{-1}_{k+1}\Theta_{k+1}B_k^{-1} & B_k^{-1}\Phi_{k+1}H^{-1}_{k+1} \\ -H^{-1}_{k+1}\Theta_{k+1}B_k^{-1} & H^{-1}_{k+1} \end{pmatrix}, \qquad (28)$$

where

$$H_{k+1}=C^{k+1}_{(k+1)(k+1)}-\Theta_{k+1}B_k^{-1}\Phi_{k+1}.$$

Thus, expression (28) makes it possible to calculate, step-by-step, the matrix B^{-1}_{k+1} and the grades of the graph vertices that have already been bypassed by the BFS algorithm, using equation (21). At each (k+1)-th step, it is necessary to invert only one matrix H_{k+1} of the size $n_{k+1}\times n_{k+1}$. All other operations are block multiplications and additions of the matrices. The total number of multiplication and addition operations to be performed at the (k+1)-th step is given in Table 1.

Table 1

The number of operations of multiplication and addition at the (k+1)-th step

Operation	Multiplication	Addition
Φ_{k+1}	$\sum_{i=1}^{k}(n_i\times n_{k+1})$	–
$C^{k+1}_{(k+1)(k+1)}$	$n_{k+1}\times n_{k+1}$	n_{k+1}
Θ_{k+1}	$n_k\times n_{k+1}$	–
$\Theta_{k+1}B_k^{-1}$	$n_{k+1}\times n_k\times\left(\sum_{i=1}^{k}n_i\right)$	$n_{k+1}\times(n_k-1)\times\left(\sum_{i=1}^{k}n_i\right)$
$\Theta_{k+1}B_k^{-1}\Phi_{k+1}$	$\left(\sum_{i=1}^{k}n_i\right)\times n^2_{k+1}$	$\left(\sum_{i=1}^{k}n_i-1\right)\times n^2_{k+1}$
H_{k+1}	–	n^2_{k+1}
H^{-1}_{k+1}	$O(n^3_{k+1})$	$O(n^3_{k+1})$
$\Phi_{k+1}H^{-1}_{k+1}$	$\left(\sum_{i=1}^{k}n_i\right)\times n^2_{k+1}$	$\left(\sum_{i=1}^{k}n_i\right)\times n_{k+1}\times(n_{k+1}-1)$
$B_k^{-1}\Phi_{k+1}H^{-1}_{k+1}$	$\left(\sum_{i=1}^{k}n_i\right)^2\times n_{k+1}$	$\left(\sum_{i=1}^{k}n_i-1\right)\times\left(\sum_{i=1}^{k}n_i\right)\times n_{k+1}$
$-H^{-1}_{k+1}\Theta_{k+1}B_k^{-1}$	$\left(\sum_{i=1}^{k}n_i\right)\times n^2_{k+1}$	$\left(\sum_{i=1}^{k}n_i\right)\times n_{k+1}\times(n_{k+1}-1)$
$B_k^{-1}\Phi_{k+1}H^{-1}_{k+1}\Theta_{k+1}B_k^{-1}$	$\left(\sum_{i=1}^{k}n_i\right)^2\times n_{k+1}$	$\left(\sum_{i=1}^{k}n_i\right)^2\times(n_{k+1}-1)$
$B_k^{-1}+B_k^{-1}\Phi_{k+1}H^{-1}_{k+1}\Theta_{k+1}B_k^{-1}$	–	$\left(\sum_{i=1}^{k}n_i\right)^2$
v_{k+1}	$\left(\sum_{i=1}^{k+1}n_i\right)^2$	$\left(\sum_{i=1}^{k+1}n_i-1\right)\times\left(\sum_{i=1}^{k+1}n_i\right)$

We also assume that $k\gg 1$ and $N_{k+1}\gg n_{k+1}$. Then the total number of multiplication operations at the (k+1)-th step is of the following order:

$$U_{k+1}=O(N^2_{k+1}\times(2n_{k+1}+1)). \qquad (29)$$

If we assume for simplicity that all n_i are equal to each other and equal to n, then the total number of multiplications in the steps from 1 to k for calculating B_k^{-1} is of the following order:

$$\sum_{j=1}^{k}U_j=O\left(n^3\times\frac{k\times(k+1)\times(2k+1)}{3}\right)=O\left(\frac{2}{3}N^3_{k+1}\right). \qquad (30)$$

The total number of additions has the same order.

For comparison, the total number of multiplications (similarly, additions, too) of fast iterative methods [12, 28, 30] is of the order

$$U_{FPR}=O\left(L_\delta\times N^2_{k+1}\right), \qquad (31)$$

where L_δ is the number of iterations that must be done to achieve

$$\left\|v_{L_\delta}-v_{L_\delta-1}\right\|\le\delta, \qquad (32)$$

where v_j is the estimate of the vector v at the j-th iteration.

For the typical values of $\delta=10^{-7}$, $\alpha=0.85$ and $N_{k+1}\sim 10^{10}$, the required number of iterations for iterative methods [30] is

$$L_\delta=50\ \ldots\ 100.$$

However, this number increases significantly as α approaches 1 and the necessary δ decreases [30].

The exact PageRank value for the page u is determined by equation (1). The parameter α is introduced in (5) to regularize the resulting solution, and the closer it is to 1, the resulting solution is closer to the ideal value.

At the same time, the volume of calculations by the proposed algebraic method does not depend on α, and the accuracy of finding the ranks of the pages is determined only by the accuracy of the calculations.

It should be noted that iterative methods begin calculations when the site graph is fully known. At the same time, in accordance with the proposed algebraic method, the amount of calculations specified in (30) can be made while the graph of the site is traversed by the BFS algorithm. After traversing the graph, only U_{k+1} operations remain (formula (29)). Proceeding from a comparison of formulas (29) and (31), we obtain that for

$$L_\delta>(2n_{k+1}+1), \qquad (33)$$

the proposed algebraic method requires less computation at the last iteration, which allows getting the necessary PageRank values faster.

6. Discussion of the developed algebraic method for calculating PageRank values

The results obtained in this study are based on the fact that, in contrast to known works, the calculation of ranks is carried out simultaneously with using the algorithm for traversing the graph in breadth. Taking into account the structural features of the transition matrix obtained at each step of the algorithm for traversing the graph in breadth, it is possible to construct a step-by-step algorithm for calculating the ranks.

The advantage of the developed method of determining ranks is the independence of the volume of calculations from the damping coefficient. The proposed step-by-step calculation of rank estimates is oriented to multicore and multiprocessor architectures and allows more efficient use of the computing device.

The applicability of the developed method is limited to Internet sites or segments with the number of pages not exceeding 10^4 or 10^5. The spread of this method to segments of the Internet with a large number of pages requires further investigation of the peculiarities of the transition matrix in particular cases.

Estimates for the volumes of computations given in formulas (29) through (31) are upper estimates, since they assume that all the matrices involved in the computations are completely filled. On real websites, the number of links on one page usually does not exceed 5–10. Therefore, most of the elements in the matrices $C^{k+1}_{(k+1)k}$, $C^{k+1}_{(k+1)(k+1)}$, and Φ_{k+1} have zero values, which drastically reduces the number of floating point operations that need to be performed.

It is also noteworthy that the proposed algebraic method of calculating PageRank at each step contains operations of addition and multiplication of matrix blocks of a small size and only one operation of inverting a matrix, again of a small size. This makes it easy to split the entire number of computations into parallel computational threads using multicore or multiprocessor computers.

7. Conclusions

1. The analysis of the algorithm for traversing the graphs of sites in depth did not reveal any features of the resulting matrix of transitions. At the same time, the algorithm for traversing the graphs of sites in breadth arranges the pages of the site into layers, which reduces the matrix of transitions to the blockwise Hessenberg type. The presence of a large number of zero blocks in such a matrix was taken into account when developing the algorithm for calculating ranks.

2. The algorithm for calculating the PageRank values is based on a step-by-step estimation of the inverse modified transition matrix on the basis of the Frobenius theorem on inversion of block matrices. At each step of the algorithm for traversing the site graph in breadth, there is a next layer of graph vertices and, accordingly, a step of the algorithm for computing the inverse transition matrix is performed.

3. The comparison of the computational volume for the new algebraic method and iterative methods has shown an advantage of the proposed method in situations where only one site or a relatively small segment of the Internet is analysed, and it is required to use a damping factor close to 1.

References

1. Understanding the Detection of View Fraud in Video Content Portals / Marciel M., Cuevas R., Banchs A., González R., Traverso S., Ahmed M., Azcorra A. // Proceedings of the 25th International Conference on World Wide Web – WWW '16. 2016. doi: 10.1145/2872427.2882980
2. Alternative Ad Networks to Open Up New Channels of Growth in 2018. URL: https://www.singlegrain.com/blog-posts/pay-per-click/44-ad-networks-will-help-open-new-channels-growth/
3. Agency Trading Desks. URL: https://adexchanger.com/Agency_Trading_White_Paper.pdf
4. What's the Difference Between Using a DSP and an Agency Trading Desk? URL: http://weevermedia.com/uncategorised/whats-difference-using-dsp-agency-trading-desk/
5. Thompson J. What Is a DSP, SSP and Ad exchange, and how do they fit together? URL: https://www.bannerconnect.net/what-is-a-dsp-ssp-and-ad-exchange/
6. Scott S. The $8.2 Billion Adtech Fraud Problem That Everyone Is Ignoring // TechCrunch. URL: https://techcrunch.com/2016/01/06/the-8-2-billion-adtech-fraud-problem-that-everyone-is-ignoring/
7. Reichow J. TUNE Fraud Series: Types of advertising fraud // TUNE. 2017. URL: https://www.tune.com/blog/types-of-advertising-fraud/
8. Ricci M. Why it's Time to Take a Stand Against Ad Fraud // MartechAdvisor. 2017. URL: https://www.martechadvisor.com/articles/ads/why-its-time-to-take-a-stand-against-ad-fraud/
9. Leskovec J., Rajaraman A., Ullman J. D. Mining of Massive Datasets. Cambridge University Press, 2014. doi: 10.1017/cbo9781139924801
10. Introduction to Algorithms / Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. 3rd ed. MIT press and McGraw-Hill, 2009. 1313 p.
11. Brin S., Page L. The anatomy of a large-scale hypertextual Web search engine // Computer Networks and ISDN Systems. 1998. Vol. 30, Issue 1-7. P. 107–117. doi: 10.1016/s0169-7552(98)00110-x
12. Polyak B. T., Tremba A. A. Regularization-based solution of the PageRank problem for large matrices // Automation and Remote Control. 2012. Vol. 73, Issue 11. P. 1877–1894. doi: 10.1134/s0005117912110094
13. Del Corso G. M., Gullí A., Romani F. Fast PageRank Computation via a Sparse Linear System // Internet Mathematics. 2005. Vol. 2, Issue 3. P. 251–273. doi: 10.1080/15427951.2005.10129108

14. PageRank // Wikipedia. URL: https://en.wikipedia.org/wiki/PageRank
15. Eigensystems for Large Matrices / Harris L., Papanikolaou I., Shi J., Strott D., Tan Y., Zhong C., Changhui T. // AMSC460. 2015.
16. Sargolzaei P., Soleymani F. PageRank Problem, Survey And Future Research Directions // International Mathematical Forum. 2010. Vol. 5, Issue 19. P. 937–956.
17. Wu G., Wei Y. A Power–Arnoldi algorithm for computing PageRank // Numerical Linear Algebra with Applications. 2007. Vol. 14, Issue 7. P. 521–546. doi: 10.1002/nla.531
18. Kamvar S. D., Haveliwala T. H., Golub G. H. Extrapolation methods for accelerating PageRank computations // Technique Report SCCM 03-02.2003. Stanford University, 2003.
19. Wu G., Wei Y. An Arnoldi-Extrapolation algorithm for computing PageRank // Journal of Computational and Applied Mathematics. 2010. Vol. 234, Issue 11. P. 3196–3212. doi: 10.1016/j.cam.2010.02.009
20. Kamvar S., Haveliwala T., Golub G. Adaptive methods for the computation of PageRank // Linear Algebra and its Applications. 2004. Vol. 386. P. 51–65. doi: 10.1016/j.laa.2003.12.008
21. Yin J.-F., Yin G.-J., Ng M. On adaptively accelerated Arnoldi method for computing PageRank // Numerical Linear Algebra with Applications. 2011. Vol. 19, Issue 1. P. 73–85. doi: 10.1002/nla.789
22. Kim M.-S. Towards Exploiting GPUs for Fast PageRank Computation of Large-Scale Networks // Proceeding of the third International Conference on Emerging Databases. 2012.
23. Rungsawang A., Manaskasemsak B. Fast PageRank Computation on a GPU Cluster // 2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing. 2012. doi: 10.1109/pdp.2012.78
24. Liu C., Li Y. A Parallel PageRank Algorithm with Power Iteration Acceleration // International Journal of Grid and Distributed Computing. 2015. Vol. 8, Issue 2. P. 273–284. doi: 10.14257/ijgdc.2015.8.2.24
25. Parallel PageRank Algorithms: A Survey / Srivastava A. K., Srivastava M., Garg R., Mishra P. K. // International Journal on Recent and Innovation Trends in Computing and Communication. 2017. Vol. 5, Issue 5. P. 470–473.
26. Parallel and Improved PageRank Algorithm for GPU-CPU Collaborative Environment / Choudhari P., Baikampadi E., Patil P., Gadekar S. // International Journal of Computer Science and Information Technologies. 2015. Vol. 6, Issue 3. P. 2003–2005.
27. Yang P., Zhou L. Research on PageRank Algorithm parallel computing Based on Hadoop // Proceedings of the 2016 4th International Conference on Mechanical Materials and Manufacturing Engineering. 2016. doi: 10.2991/mmme-16.2016.40
28. Gleich D., Zhukov L., Berkhin P. Fast Parallel PageRank: A Linear System Approach. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.592.64&rep=rep1&type=pdf
29. Gantmakher F. Theory of matrices. Moscow: Nauka, 2004. 581 p.
30. Wright G. Probability, linear algebra, and numerical analysis: the mathematics behind Google’sTM PageRankTM. URL: http://math.boisestate.edu/~wright/courses/m297/google_talk.pdf