SpeedSiteRank: PageRank algorithm distributed in websites

Saint-Jean A. O. Djungu[1] and Pierre Manneback[2]

[1] Mathematics and Informatics Department, Faculty of Sciences, University of Kinshasa

Kinshasa, R.D. Congo

[2] Computer and Management Engineering Department, Faculty of Engineering, University of Mons

Mons, Belgium

Abstract

A global and centralized classification of web pages requires a fairly high computation cost and therefore does not favor a regular updating of the index database. To overcome this problem, we propose in this article an asynchronous parallel algorithm, called *SpeedSiteRank*, capable of calculating the PageRank vector by site. The results of tests carried out on a cluster made up of 10 bi-opteron nodes demonstrated the efficiency of our algorithm.

Keywords*: PageRank, SpeedSiteRank, website*

1. Introduction

Exploiting the structure of the hypertext links between the different web pages has made it possible to considerably improve the performance of modern search engines. Designed in 1998 by Larry Page and Sergey Brin, the famous Google search engine classifies web pages using a combination of multiple factors, the most famous of which is called PageRank. The latter uses the number of links pointing to a web page to assign it a popularity index. PageRank is also one of the measures used to allow crawlers to crawl web content [1].

Systematic exploration of the web has become a delicate task subject to strong constraints. Indeed, the technical aspects of the web such as network traffic and bandwidth are the factors penalizing in the repatriation of web pages. In addition, the current size of the web, estimated at several billion pages, greatly impedes the progression of crawlers by significantly extending the time necessary for the completion of an exploration cycle [6]. This involves considerable time for updating the index as well as calculating the popularity index for each web page [13].

It has become difficult to implement crawlings intended to visit all of the web pages. Faced with these constraints, we saw in [7] that classical parallelization, based on index centralization, has shown its limits. Hence the need to focus on distributed and collaborative systems in terms of crawling, indexing and information retrieval [2, 4].
In this article, we take advantage of the natural decomposition of the web into blocks (servers, domains, directories, etc.) to approximate the PageRank vector, so that the update of the index base of a site is done without demand to crawl the entire web.

In this contribution, we propose an approach based on the Vantilborgh aggregation / disaggregation method [5], which allows to calculate the PageRank vector by site. It differs from previous work, [3, 10, 12, 14], in which the estimation of the PageRank vector is done first by site and then refined by taking into account information outside the site. In our approach, the calculation of the PageRank vector integrates all information outside the site beforehand. This is to avoid calculating the PageRank vector twice per site.

This article is organized as follows: Section 2 presents a number of definitions and notations. In section 3, we give the wording of SpeedSiteRank, indicating how the load balancing is done as well as the communications between processors. Section 4 is devoted to the presentation of experimental results and performance analysis. Finally we conclude and present perspectives for future work.

2. Notations and definitions

In this section, we group together a number of definitions and notations that will be used throughout this article.

Definition 1 [9]: A *website* is a set of interrelated web pages identified by a common name (domain, server, directory, etc.).

Let $\delta = (S_1, ..., S_k)$ be a set of k sites, with $k>1$. The size of each site S_i is equal to n_i and $\sum_{i=1,...,k} n_i = n$.

Definition 2: The *extended site graph* S_i is defined by the couple $G_i = (V_i, E_i)$, where V_i is a set of n_i web pages of S_i and E_i the set of pairs of web pages whose first component (element of V_i) has a hypertext link to the second component (element of V_i or not). The transition matrix associated with the graph G_i is a matrix $A_{i*} \in M_{n_i,n}(\Re)$ defined by equation (1) below.

$$A_{i*} = (a_{u,v}),\ with\ \ a_{u,v} \begin{cases} \frac{1}{d_u} & if\ u \to v \\ 0 & else \end{cases} \qquad (1)$$

where d_u is the number of outgoing links on the page $u \in V_i$.

Definition 3: Let A_{ii} be the *intra-site matrix* S_i and $A_{ij}\ (j \neq i)$ the *inter-site matrix* S_i to S_j. The matrix A_{i*} can still be written as:

$$A_{i*} = [A_{i1}, ..., A_{ii}, ..., A_{ik}] \qquad (2)$$

By aggregating site graphs, we get the web graph $G = (V, E)$, where all of the web pages are $V = \bigcup_{i=1}^{k} V_i$ and the hyperlinks is $E = \bigcup_{i=1}^{k} E_i$. The transition matrix associated with the graph G is therefore

$$A = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1k} \\ A_{21} & A_{22} & \dots & A_{2k} \\ \cdot & \cdot & & \cdot \\ \cdot & & \cdot & \cdot \\ \cdot & & & \cdot \\ A_{k1} & A_{k2} & \dots & A_{kk} \end{bmatrix} \qquad (3)$$

Let x be a vector of size n. Taking into account the sizes of the different sites, the vector x can be written as follows:

$$x = (x_1, ..., x_k)^T \qquad (4)$$

where each x_i is a vector of size n_i, $i = 1, ..., k$.

Let be a vector

$$x^* = (x_1^*, ..., x_k^*)^T \qquad (5)$$

with $x_i^* = x_i / \|x_i\|_1$ a line vector of size n_i.

Let $L \in M_{n,k}(\Re)$ and $S \in M_{n,k}(\Re)$ be two matrices associated with the operators of communications between the sites, defined respectively as follows:

$$L(x^*) = \begin{bmatrix} x_1^* & 0 & \dots & 0 \\ 0 & x_2^* & \dots & 0 \\ \cdot & & \cdot & \cdot \\ \cdot & & \cdot & \cdot \\ \cdot & & & \cdot \\ 0 & 0 & \dots & x_k^* \end{bmatrix} \qquad (6)$$

and

$$S = \begin{bmatrix} e_1 & 0 & \dots & 0 \\ 0 & e_2 & \dots & 0 \\ \cdot & & \cdot & \cdot \\ \cdot & & \cdot & \cdot \\ \cdot & & & \cdot \\ 0 & 0 & \dots & e_k \end{bmatrix} \qquad (7)$$

Let $B(x^*)$ be the inter-site transition matrix: for two given sites i and j, we want $B_{i,j}(x^*)$ to measure the probability that, starting from site i according to the distribution x_i^*, we reach j by randomly following an edge of G_i. Using relations (6) and (7), we obtain the expression of the matrix $B \in M_{k,k}(\Re)$ using the matrices $A \in M_{n,n}(\Re)$, $L \in M_{n,k}(\Re)$ and $S \in M_{n,k}(\Re)$.

$$B(x^*) = L(x^*)^T AS \qquad (8)$$

Since A is both stochastic and irreducible, we deduce that the matrix B is stochastic and irreducible [11].

3. SpeedSiteRank algorithm

In general, the matrix A defined by the relation (3) is not irreducible. To force irreducibility, we consider the normalization of A by taking the weighted average by of A and by $(1 - \alpha)$ of ez^T.

$$A_\alpha = \alpha A + (1-\alpha) e z^T \quad (9)$$

where z is a uniform zap distribution and α the zap factor. The distribution in sites of the matrix A_α is written as indicated in (10).

$$(A_\alpha)_{ii} = \alpha A_{ii} + (1-\alpha) e_i z_i^T$$
$$(A_\alpha)_{ij} = \alpha A_{ij} + (1-\alpha) e_i z_j^T \quad (10)$$

where e_i and z_i are vectors of size n_i. The $(A_\alpha)_{ii}$, $i = 1, \cdots, k$, are the block matrices of the main diagonal of the matrix A_α.

Proposition 1

Let z be a distribution of zap of size n, e^* a unit vector of size k (number of sites) and $b = [\|z_1\|_1, \ldots, \|z_k\|_1]^T$. Let $V = e^* b^T$ be a square matrix of order k.

1. The expression of the inter-site matrix is given by:

$$B^* = \alpha B + (1-\alpha) V \quad (11)$$

2. The relative importance of sites is a solution vector of equation (12).

$$w = \text{SpeedRank}(B, \alpha, (1-\alpha)b) \quad (12)$$

where $w = \text{SpeedRank}(B, \alpha, (1-\alpha)b)$ comes from $w = \alpha B^T w + (1-\alpha) b$.

Proposition 2

An approximation of the PageRank vector, x_i, relating to the web pages of the site S_i is given by:

$$\begin{cases} y_i = \text{SpeedRank}(A_{ii}, \alpha, r_i) \\ x_i = w_i y_i \end{cases} \quad (13)$$

where

$$r_i = (1-\alpha) + (2-w_i) z_i + \alpha \sum_{j \neq i} w_j x_j^{*T} A_{ji}$$

is a line vector of size n_i.

We have now all the ingredients to describe our new algorithm, which we propose to name *SpeedSiteRank*. The parallelism required by *SpeedSiteRank* is better exploited if the number of sites (k) is equal to the number of processors (p). In practice, the number of processors is often less than the number of sites. Hence the need to group the calculation of PageRanks of sites on a processor.

Let $F(i)_{1, 2, \ldots p}$ be the frequency of the i^{th} processor and $Nnz(j)_{1, 2, \ldots k}$ the number of non-zero elements in the matrix of the j^{th} site. Based on the principle of dynamic programming, algorithm 1 makes it possible to determine the optimal static distribution of the computational load of PageRanks on p processors. The greedy algorithm 1 makes it possible to determine the optimal static distribution of the load of computation of PageRanks on p heterogeneous processors. The case where the processors are homogeneous is a special case.

***Algorithm 1**: function DC = loadDistribution (Nnz, F, k, p)*

Start

Sort Nnz in descending order

tempsEx (i) = 0, for i = 1, ..., p

For i = 1 to k do

$$j = \arg\min_{1 \leq t \leq p} (tempsEx(j) + \frac{Nnz(i)}{F(t)})$$

DC (i) = j

$$\text{tempsEx}(j) = tempsEx(j) + \frac{Nnz(i)}{F(j)}$$

End

End

Step 7 of algorithm 2 shows how communication is done between the sites. The site S_i, for example, sends to all the other sites a vector $K_{i,*}$. The latter contains all the information concerning the hypertext links between the S_i site and all the other sites. The exploitation of the master-slave paradigm, for example, requires the redistribution of vectors w and f (steps 9 and 10) on the slave processors.

***Algorithm** 2: Parallel approximation of SpeedSiteRank*

Data

** A set of sites = (S_1, ..., S_k) and $n_i = |S_i|$;*

** Transition matrix A associated with $\mathcal{S}$;*

** Matrix S indicating the pages belonging to the sites;*

** A distribution of zap z;*

** A coefficient of zap $\alpha \in]0,1[$;*

** DC load distribution*

Result

** An estimation of the PageRank vector by sites*

Start

1. $x^{(0)} = (\frac{z_1}{\|z_1\|_1}, \ldots, \frac{z_k}{\|z_k\|_1})^T$
2. $b = (\|z_1\|_1, \ldots, \|z_k\|_1)$
3. $B = O_{k \times k}$
4. *For each site S_i of $\mathcal{S}$ according to DC do*
5. $K_{i,*} = x_i^{(0)T} A_{i*}$
6. *End*
7. *K = GlobalCollection ($K_{i,*}$)*
8. *B = KS*
9. $w = \text{SpeedRank}(B, \alpha, (1-\alpha)b)$
10. *f = wK*
11. *For each site S_i of $\mathcal{S}$ according to DC do*
12. $r_i = (1-\alpha) + (2-w_i)z_i + \alpha \sum_{j \neq i} w_j x_j^{*T} A_{ji}$
13. *y_i = SpeedRank(A_{ii}, α , r_i)*
14. *$x_i = w_i y_i$*
15. *End*

End

4. Digital experiences

4.1. Test matrices

To test our algorithms, we considered matrices from the Web-Graph project (http://law.dsi.unimi.it/). The partitioning into sites of these matrices was made on the basis of a lexicographic sorting on the urls. We have partitioned into servers or domains. Table 1 shows the main characteristics of 2 of these matrices. Written in C and MPI, our code has been tested on a Linux cluster with 10 bi-opteron 244 nodes clocked at 1.8 GHz interconnected in Gigabit Ethernet. Each node has 2 GB of RAM and 76 GB of HDD in 10,000 RPM SCSI. Only 8 nodes were used to do the tests.

Table 1: Characteristics of the matrices

	#Pages	**# Links**	**Lvl.**	**#Sites**
cnr-2000	325557	3216152	server	143
in-2004	1382908	16538959	domain	13

4.2. Performance analysis

Given the asynchronous nature of *SpeedSiteRank* and the fact that the site graphs do not have the same number of hypertext links, the execution times per node are in an interval *[timemin, timemax]*. To allow us to appreciate the scalability of our results, we offer a presentation where only the average time *((max time - min time) / 2)* is indicated (see Table 2).

Table 2: Average SpeedSiteRank execution time compared to the number of nodes ($\alpha = 0.85$)

#Links	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
cnr-2000 (in sec)	53	30	22	19	17	14	13	12
in-2004 (in sec)	256	131	88	67	56	48	43	38

Figure 1 shows the variation of the execution time as a function of the number of nodes used. It is noted that the scalability obtained is a function of the size and the nature of the matrix used. The *in-2004* matrix has the best scalability compared to *cnr-2000*.

4.3. Correlation between PageRank and SpeedSiteRank

In this article, we have used the normalized Kendall distance to compare the rankings obtained using the

reference PageRank and our new *SpeedSiteRank* algorithm, respectively.

Definition 4: The normalized Kendall distance between two lists X and Y of size n is given by the relation (14).

$$\mathrm{KDist}(X,Y) = \frac{\sum_{\{i,j\}\in P} K_{ij}(X,Y)}{n(n-1)/2} \qquad (14)$$

where

- P is the set of disordered pairs of distinct elements in X and Y
- $K_{ij} = 0$ if i and j are in the same order in X and Y
- $K_{ij} = 1$ if i and j are in opposite order in X and Y.

Starting from definition 4, we found 0.0189 and 0.0306 respectively as mean normalized Kendall distances for the matrices cnr-2000 and in-2004. Thus, the rankings obtained using PageRank and SpeedSiteRank are correlated to more than 98% for cnr-2000 and 97% for in-2004.

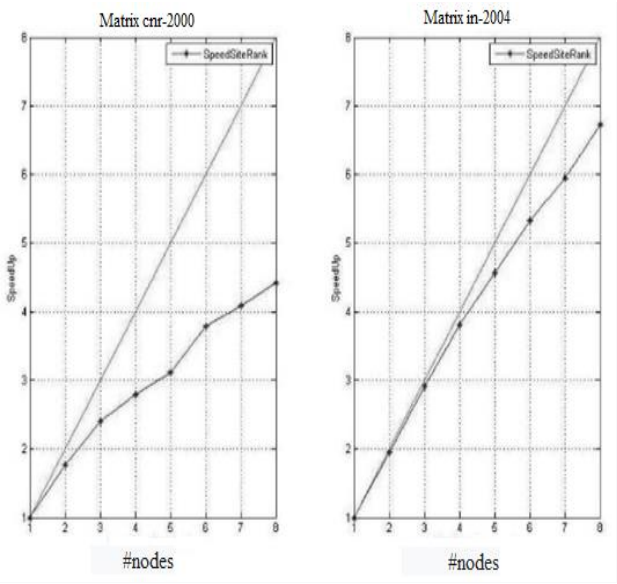

Figure 1: speed-up of SpeedSiteRank for the cnr-2000 and in-2004 matrices

5. Conclusion

The objective of this article was to find a non-centralized way to calculate the PageRank vector. The SpeedSiteRank algorithm is not only a response to the initial concern, but is a powerful way to allow the webmaster to update his index base without trying to heckle the whole web.

Our future work will consist in seeking to validate our algorithm on very large web matrices obtained on the basis of a decentralized crawling in sites (servers, domains, tld, etc.). We can also take advantage of the results obtained in [7, 8] to use EramRank or GmersRank as input parameters for SpeedSiteRank instead of SpeedRank.

References

[1] S. Abiteboul, M. Preda and G. Cobena. Adaptive on-line page importance computation. In Proceedings of the twelfth international conference on World Wide Web, ACM Press, 2003, pp. 280-290.
[2] K. Aberer and J. Wu. A framework for decenralized ranking in web information retrieval. In Web Technologies and Appliations : Proceedinfs of th Asia-Pacific Web Conference, APWeb 2003 volume LNCS 2642, Xi'an, China, September 2003. Springer-Verlag. September 27-29, 2003, pp. 213-226.
[3] A. Broder, R. Lempel, F. Maghoul, and J. Pedersen. Efficient pagerank approximation via graph aggregation. In Proc. of the WWW'04 Conf., 2004, pp. 484 - 485.
[4] J. Cho and H. Garcia-Molina. Parallel crawlers. In Proceedings of the eleventh international conference on World Wide Web, Honolulu, Hawaii, USA, ACM Press. May 2002, pp. 124-135.
[5] W. Cao and W. J. Stewart. Iterative aggregation/disaggregation techniques for nearly uncoupled markov chains, ACM Press. 1985, pp. 702 - 719.
[6] J. Cho and S. Roy. Impact of Web search engines on page popularity. In Proceedings of the Thirteenth International WWW Confernce, 2004.
[7] S.-J. Djungu et P. Manneback. Mise en œuvre parallèle sur cluster d'algorithmes itératifs de PageRank. Dans les Actes de 17ème Rencontres francophones du Parallélisme. Perpignan/France, du 3 au 6 octobre 2006.
[8] S.-J Djungu. Conception et mise en œuvre parallèle d'algorithmes de PageRanking. Editions Universitaires Européennes, Allemagne, 2014.

[9] C. Kohlschutter, P. Chirita, and W. Nejdl. Efficient Parallel Computation of PageRank, 2006.
[10] S. Kamvar, T. Haveliwala, C. Manning and G. Golub. Exploiting the Block Structure of the Web for Computing PageRank, Technical Report, Stanford University, 2003.
[11] I. Marek and P. Mayer. Convergence analysis of an aggregation / disaggregation iterative method for

computation stationary probability vectors of stochastic matrices, Numer. Linear Algebra Appl. 5, 1998, pp. 253-274.
[12] F. Mathieu and L. Viennot. Aspects of the Global Ranking of Web Pages. In Proceedings of I2CS 2006, 2006.
[13] L. Page and S. Brin. The anatomy of a large-scale hypertextualWeb search engine. Computer Networks and ISDN Systems, 1998, 33(3) : pp. 107-117.
[14] Y. Wang and D. J. DeWitt. Computing PageRank in a distributed internet search system. In Proceedings of the 30th VLDB Conference, 2004.