# Google Page Rank

Rakesh Shivanand Margoor[1] and Ankit Srivastava[2]

[1] University of Colorado Boulder, CO, USA
rakesh.margoor@colorado.edu
[2] University of Colorado Boulder, CO USA
ankit.srivastava@colorado.edu

**Abstract.** Page rank algorithm is one of the important algorithms that are being used in the web search. It plays a vital role in deciding the importance of a webpage. The basic page rank algorithm involves large computation due to the involvement of millions of webpages resulting in a high computation time. In this paper, we discuss and implement several algorithms such as adaptive, filter-based adaptive, Aitken's extrapolation to reduce the time computation involved in calculation of page-ranks of webpages. We also discuss Weighted and Visits Of Links (VOL) algorithm which considers practical factors in the web-search to improve the accuracy of the page-ranks.

**Keywords:** Eigen vector, Extrapolation techniques, sparse matrix, Directed Acyclic Graph (DAG), Convergence

## 1 INTRODUCTION

In 1998, Brin and Page [1] came up with an algorithm called page rank algorithm to rank or distribute the importance of web pages based on the quality of inlinks associated with a page. The basic principle behind the above algorithm is based on citation, if a page is cited by large number of websites, then it the page is more important. This principle is defined by the below equation

$$PR(p_i) = (1 - \mathrm{d}) + d \sum_{p_j \in M(p_j)} \frac{PR(p_j)}{L(p_j)} \qquad (1)$$

where $p_1$ , $p_2$ , … $p_n$ are the webpages that are being considered, $M(p_j)$ are set of pages that link to ith page, PR being the page rank of a webpage, $L(p_j)$ indicates the number of outbound links from the page $p_j$ and d is the damping factor.

The damping factor represents the probability of a user switching from page i to page j. This ensures the convergence of the page ranks value in the above equation. The above convergence of the equation is associated with the power method where the inbound links form the eigen matrix and the page rank values are the eigen vector as shown in the below eqn.

$$y^{k+1} = Ay^k \quad (2)$$

where y is the eigen vectors which represents the page-rank value and A is the eigen matrix with inbound links and k represents the iteration number.

The above basic page rank algorithm can be optimized to reduce the time computation involved in calculation of page rank. In this paper, we have researched about some of the areas where the above algorithm can be achieved through lesser time complexity and are discussed in the below sections,

## 2 ADAPTIVE APPROACH

### 2.1 Adaptive Page rank algorithm

Page ranks of several webpages converge in lesser number of iterations than the others. Hence, the re-computation of the converged page rank at all further iterations till all page ranks converge, involves redundant computation. This concept is carefully utilized in the adaptive algorithm to avoid the redundant computation.

Consider the Eqn 2 where the eigen vectors or the page ranks are calculated for the k+1th iteration from the previous kth eigen vectors. Let's assume that C is the set of pages whose page ranks are already converged and N set of pages whose page ranks are not yet converged at the kth iteration. So with the above assumption, we can write the eigen vectors and the A matrix as

$$y^k = \begin{pmatrix} y_N^k \\ y_C^k \end{pmatrix} \quad (3)$$

$$\mathrm{A} = \begin{pmatrix} A_N \\ A_C \end{pmatrix} \quad (4)$$

Based on the above split in x and A matrices, we can write the Eqn 2 as

$$\begin{pmatrix} y_N^{k+1} \\ y_C^{k+1} \end{pmatrix} = \begin{pmatrix} A_N \\ A_C \end{pmatrix} \begin{pmatrix} y_N^k \\ y_C^k \end{pmatrix} \quad (5)$$

Now note that in the above equation, $y_C^{k+1}$ need not be re-calculated again, as it is equal to $y_C^k$. Hence the equation 5 represents the reduced computation to

$$y_N^{k+1} = A_N y^k \quad (6)$$

Since $A_N$ is smaller than the actual matrix A, the computation is definitely reduced. It would optimize efficiently if large number of pages converge much earlier than the few other pages.

Now, the disadvantage of the above algorithm is reordering the matrix A every time to converged and non-converged sub-matrix. Since A is quite a big matrix, the reordering of the matrix takes up a larger computation and hence is not efficient in all cases. We can reduce the number of times matrix A is re-ordered, but this compromises with avoiding redundant calculation on converged page rank. Now in order to find a better approach, we discuss another algorithm which is described in the next subsection.

### 2.2 Filter based Adaptive Page rank algorithm

Continuing with the disadvantage of the basic adaptive algorithm, we notice through Eqn 6 that the sub-matrix $A_C$ is not used in the computation of the page rank of the k+1th iteration. So, if we are not using it, we could remove it, which would reduce the computation in reordering the matrix A. To match the dimensions of matrix multiplication, we assign 0 values to the submatrix $A_C$ thereby increasing the sparse of the matrix A. Thus, the approach here is to make the row of the converged page to completely 0 as shown in the below equations. Let us define a matrix $\mathrm{A}'$ and $y_C'$

$$\mathrm{A}' = \begin{pmatrix} A_N \\ 0 \end{pmatrix} \quad (7)$$

$$y_C'^{k} = \begin{pmatrix} y_N^k \\ y_C^k \end{pmatrix} \quad (8)$$

With the above filtering, we can now write the equation for the page rank of k+1th iteration as below

$$y^{k+1} = \mathrm{A}' y^k + y_C'^{k} \quad (9)$$

Note that, the size of the matrix $\mathrm{A}'$ is same as that of the original matrix A, Hence the general time complexity is same. However, since $\mathrm{A}'$ is sparser, the computation cost is less, as the computation of multiplication is considered only when both the variables are non-zero. This method also removes the ex-

tra computation of reordering the matrix as done in the basic adaptive algorithm. To filter out all the converged rows, the worst case would take a linear scan on the matrix, thereby reducing the computation time significantly.

### 2.3 Advanced Adaptive algorithm

The above two algorithms focus on reducing the computation by not re-computing the page rank of the pages which are already converged. Along with this, an additional optimization can be performed by avoiding the computation of the partial page rank weights given by the converged pages to the non-converged pages. Since the converged page's page-ranks are fixed, then re-computation of those to the non-converged pages can be easily reduced as shown by the below equations.

Let $A_{NN}$ be the links from non-converged pages to the non-converged page, $A_{CN}$ be the links from the converged pages to the non-converged pages. $A_{CC}$ and $A_{NC}$ are the vice-versa of the above two. Now, with these sub-matrices we can write a new matrix $A^*$ as

$$A^* = \begin{pmatrix} A_{NN} & A_{NC} \\ A_{CN} & A_{CC} \end{pmatrix} \quad (10)$$

With the above matrix, we can rewrite the equation for calculating the page rank of the k+1th iteration as below.

$$y_N^{k+1} = A_{NN} y_N^k + A_{CN} y_C^k \quad (11)$$

Now, the idea behind the algorithm is to not re-compute the second term in the Eqn 11. The matrix $A^*$ is not calculated or reordered on every iteration, rather it is done periodically after certain fixed size iterations. During this fixed number of iterations, the second term has to be computed only once and can be reused till the next periodic change of the matrix $A^*$. Thus, through the eqn 11 we can not only reduce the computation of converged page ranks but can also avoid page rank distribution of the converged pages to the non-converged pages.

In this subsection, we have discussed three adaptive algorithms which focus on reducing the work done in a single iteration. Apart from this, we can also optimize the equation2 by reducing the number of iterations in the convergence. The next section discusses some of the methods that we researched to increase the convergence of the page rank algorithm.

# 3 EXTRAPOLATION APPROACH

## 3.1 Aitken's delta-square method

The idea of increasing the convergence rate of a sequence made us research in extrapolation methods. We have researched several extrapolation methods that can be applied to converge the page rank algorithm quicker. Some of them are Aitken's delta-square method and Quadratic extrapolation method.

The algorithm is built by considering the iterate $y^{k-2}$ as a linear combination of the first two eigen vectors. The same concept is used to derive a new sequence called $y^*$ which converge quickly. Based on the power method, we can define $y^*$ with the previous two eigen vectors as shown in the below equations.

$$g = (y^{k-1} - y^{k-2})^2$$

$$h = (y^k - 2y^{k-1} + y^{k-2})$$

$$y^* = y^k - \frac{g}{h} \qquad (12)$$

The above new sequence $y^*$ is calculated periodically with the previous eigen vectors from the power method for better convergence. The above equations are called usually or every 10$^{th}$ iteration for better results. The above aitken's equation can be integrated with the power method as depicted in the below pseudo-code. Here ε indicates the threshold used for convergence, and the function *Aitken* performs the set of equations as described in Eqn 12.

```
Function y^n = AitkensPowerMethod(){
k=1;
repeat;
  y^k = A y^(k-1);
  ∂ = abs(y^k - y^(k-1));
  Periodically, y^k = Aitken(y^(k-2), y^(k-1), y^k);
  k=k+1
until ∂ < ε;
}
```

All the above discussed algorithms, focus on reducing the computation time for calculating the page-ranks of web-pages. The next sections considers, the practical scenario for generating accurate page-ranks.

# 4 ADVANCED PAGE RANK ALGORITHM

## 4.1 Weighted algorithm

Weighted Page Rank Algorithm is an advanced version of basic page-rank algorithm. In basic page-rank algorithm, each web page equally contributes its page rank to all of the outbound links. The weighted algorithm involves a slight modification in the basic page rank algorithm.
The basic algorithm does not account the popularity of the web-page. Consider a graph where web page A has an outbound link to web-pages B and C and A having more impact on webpage B due to the high popularity of B, With this assumption, B has to get more weightage from page A than C, However the basic page algorithm fails to distribute the weights unequally, The weighted approach considers this scenario and instead of contributing equally, contributes its rank based on weight of links which is directly proportional to the popularity of web pages. Google has currently integrated its analytics which adopts the relative importance of two pages. Based on this, the weights are assigned to each of the links in the considered graph.

The above algorithm can be represented by below equations. Consider the popularity from the number of inlinks and outlinks as $W^{in}_{(v,u)}$ and $W^{out}_{(v,u)}$ respectively, where $W^{in}_{(v,u)}$ is the weight of link(v, u) calculated based on the number of inlinks of page u and the number of inlinks of all reference pages of page v. $W^{out}_{(v,u)}$ is the weight of link(v, u) calculated based on the number of outlinks of page u and the number of out-links of all reference pages of page v.

$$W^{in}_{(v,u)} = \frac{I_u}{\sum_{p \in R(v)} I_p} \quad (13)$$

where $I_u$and $I_p$represent the number of inlinks of page u and page p, respectively. R(v) denotes the reference page list of page v.

$$W^{out}_{(v,u)} = \frac{O_u}{\sum_{p \in R(v)} O_p} \quad (14)$$

where $O_u$and $O_p$represent the number of outlinks of page u and page p, respectively. R(v) denotes the reference page list of page v.
With the above equations, we can modify the basic page rank algorithm as

$$PR(u) = (1-d) + d \sum_{v \in B(u)} PR(v)\, W^{in}_{(v,u)} W^{out}_{(v,u)} \quad (15)$$

### 4.2 Visits Of Links(VOL) based Algorithm

The VOL algorithm modifies the basic page rank algorithm by considering the semantics and the user behavior in the web-search. The number of hits on a given page and the number of visits from page A to page B is recorded. These recorded values are used to distribute the page rank as shown in the below Eqn 16.

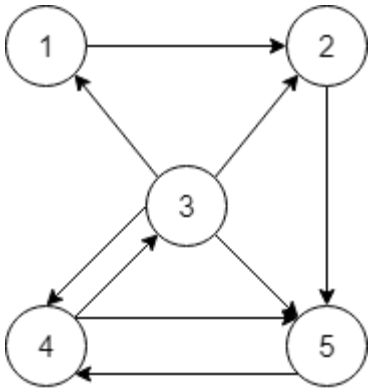
$$WPR_{vol}(u) = (1-d) + d \sum_{v \in B(u)} \frac{L_u WPR_{\text{vol}}(v) W^{in}_{(v,u)}}{TL(v)} \qquad (16)$$

Where B(u) is the set of pages that point to page u, $WPR_{vol}(u)$ and $WPR_{vol}(v)$ are rank scores of page u and v respectively, Lu is the number of visits of link which is pointing page u from v and TL (v) denotes total number of visits of all links.

The above algorithm is very useful to display most valuable pages on the top of the result list on the basis of user browsing behavior, which reduce the search space to a large scale.

## 5 IMPLEMENTATION AND RESULTS

In order to verify the discussed algorithms in section two, three and four, we choose the below directed acyclic graph with convergence threshold as 0.001.



| Algorithm | PR 1 | PR 2 | PR 3 | PR 4 | PR5 | Total time taken(ms) | # Iterations |
|---|---|---|---|---|---|---|---|
| Basic PageRank | 0.3304 | 0.6111 | 0.8494 | 1.6465 | 1.5491 | 10 | 35 |
| Adaptive Algorithm | 0.3259 | 0.6044 | 0.8444 | 1.6362 | 1.5374 | 8 | 31 |
| Advanced Adaptive Algorithm | 0.3259 | 0.6044 | 0.8444 | 1.6362 | 1.5374 | 7.6 | 31 |
| Aiken's Method | 0.3165 | 0.6164 | 0.8243 | 1.6521 | 1.5391 | 12 | 26 |
| Weighted Method | 0.3152 | 0.5645 | 0.7673 | 1.3457 | 1.8921 | 11 | 32 |

Based on the above results, we can interpret many aspects of our algorithms. As you can see the adaptive algorithm takes lesser time to compute the page-ranks as compared with the basic page-rank. Apparently, the number of iterations is 31 which is less than that of the basic page rank, this is just a coincidence, as the adaptive algorithm does not try to converge. Also, the note that the advanced and adaptive algorithm page-rank values are same, as we are not re-computing the page rank values of the converged pages.
Also, in the Aitken's method, the number of iterations reduced to 26, however its total time is more than the basic due to the Aitkens extra computation. The Aitkens method would take up smaller time when the graph is very huge, which would result in larger reduction in the number of iterations.
Also, in the weighted method, we assigned a higher weight value to the page 5 than the other. Thus with a larger weight for the edges (2,5) and (3,5) we were able to see, that 5 being assumed as the popular page is assigned a higher page-rank value than its previous values with the basic and the adaptive based page-rank algorithm. The same can be abstracted for a larger dataset, where an analytics would measure the popularity of the pages and the weights can be associated proportionally.

## 6 CONCLUSION

The research in this paper helped us in understanding the mathematical analysis behind the Google's page rank algorithm. A thorough research has been done on algorithms that can be applied to reduce the computation involved in the page rank calculation. The paper also discusses about certain extrapolation methods that can be applied to reduce the number of iterations thereby achieving faster convergence in the page-rank algorithm. The paper further discusses about the practical and semantic factors such as visits of links, popularity of pages that affect the importance of a web-page. All the above approaches have been formulated and analyzed thoroughly. Overall, the research on page-

rank algorithm was very enjoyable and interesting. We would like to thank Professor Ashutosh for giving us such an opportunity to work on one of the interesting projects.

## 7 FUTURE WORK

Currently, the algorithms that we have discussed were tested on a small graph with less than 10 nodes. One of the future works would be to scale our code to incorporate the data from the Google's test data. This would require sufficient memory and a distributed cluster for better and faster computation. There are certain extrapolation methods such as higher polynomial extrapolation and multi-grid algorithms which can be applied to the page-rank algorithm. We would like to incorporate these algorithms and compare with the current algorithms. Also, currently for the VOL algorithm we have mocked the user data, in future we can create an application that records the user's visits and apply the algorithm in run time. This would make our algorithm robust and real time.

## 8 ACKNOWLEDGEMNT

We acknowledge that the introduction section was discussed and implemented by both the Authors. The research on section 2, 3 and their implementation is done by Rakesh Shivanand Margoor. The research on section 4 and its implementation is done by Ankit Srivastava.

## 9 REFERENCES

[1] S. Brin, and Page L., "The Anatomy of a Large ScaleHypertextual Web Search Engine", Computer Network and ISDN Systems, Vol. 30, Issue 1-7, pp. 107-117, 1998.
[2] Wenpu Xing and Ghorbani Ali, "Weighted PageRank Algorithm", Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR "04), IEEE, 2004
[3] Neelam Tyagi, Simple Sharma, "Weighted Page Rank Algorithm Based on Number of Visits of Links of Web Page", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-3, July 2012.
[4] Sepandar Kamvar, Taher Haveliwala, and Gene Golub, "Adaptive Methods for the Computation of PageRank", Stanford University.

[5] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning and Gene H. Golub, “Extrapolation Methods for Accelerating PageRank Computations”.
[6] A. Aitken. On Bernoulli’s numerical solution of algebraic equations. Proc. Roy. Soc. Edinburgh, 46:289–305, 1926.
[7] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin. “PageRank computation and the structure of the web: Experiments and algorithms”, Eleventh International World Wide Web Conference, Poster Track, 2002