Page Rank Algorithm Analysis

November 22, 2021

Anchal Gupta (2019chb1038) ,
Kaustubh Singh (2019meb1268) ,
Khushi Malviya (2019meb1270)

Instructor:
Dr. Anil Shukla

Teaching Assistant:
Simran Setia

Summary: The project aims to implement as well as discuss the page rank algorithm along with modern customization and improvements being involved. The algorithm was developed by Page and Brin. The implementation is modelled in C language entirely with list implementation and discusses the computation of page rank of every mapped node in a directed graph taking input from large graph data sets available over the web through a text file. In conclusion, the project has a in depth comparison of the page rank algorithm with the popular alternative of HITS algorithm as well. All the work is done under CS201 course project under the guidance of Dr Anil Shukla and Ms Simran Setia.

---

1. Introduction

The page rank algorithm was developed by Lawerence Page and Sergey Brin to rank the web pages in a designated web graph with the use of user like simulated behaviour. It is an algorithm that computes ranking scores for the nodes using the network created by the incoming edges in the graph. Thus it is intended for directed graphs, although undirected graphs can be treated as well by converting them into directed graphs with reciprocated edges (i.e. keeping the original edge and creating a second one going in the opposite direction). The edges on the graph will define the relevance of each node in the graph, reflecting this on the scores, meaning that greater scores will correspond to nodes with greater relevance.It uses the number of outgoing and incoming links that connect the web pages, in terms of graph the in-degree and the out-degree to determine the page rank of a web page. Hence, not only to the web pages the algorithm is suited to any data which can be mapped to a graph to gain insights about the relative proficiency of the nodes.The algorithm has been long used and its due course has also seen certain modification such as personalised page rank or nstar page rank. The project describes the mathematical and the implementation aspects of the algorithm with an in-depth dive into time complexity and comparison to existing alternatives as well.

2. Equations

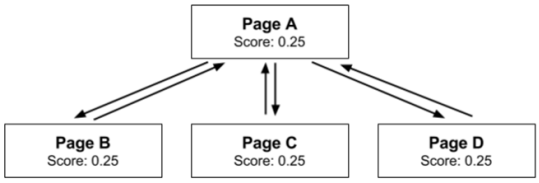
The formula for page rank of a vertex in a graph where each vertex represents a web page, given by Sergery Brin and Lawrence Page is as follows.

$$PR(A) = \frac{(1-d)}{N} + d(\frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)}) \tag{1}$$

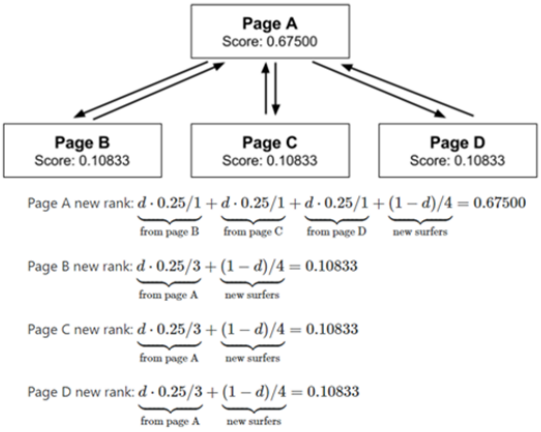
In the above equation, PR(T_i) stands for the page rank of a web page T_i. We assume that $T_1, T_2....., T_n$ are the web pages pointing to web page A, and $C(T_1), C(T_2).....C(T_n)$ are the number of outgoing links from web pages $T_1, T_2....., T_n$ respectively. Also, d is the damping factor which can have a value from 0 to 1 and is generally taken as 0.85.

3. Figures, Tables and Algorithms

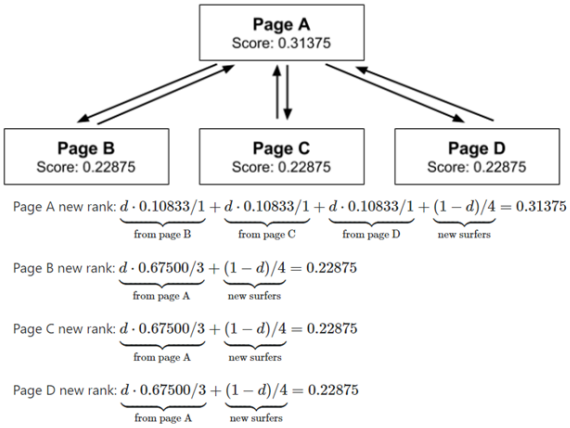
3.1. Figures



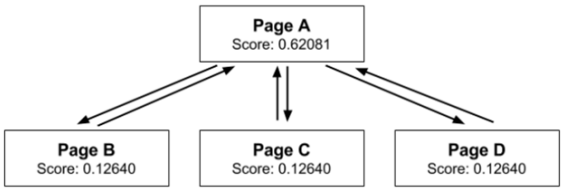
(a) Initialising page rank for shown graph



(b) Updated page rank after first iteration



(c) Updated page rank after second iteration



(d) Updated page rank after third iteration

(e) Final converged page rank after hundred iterations

Figure 1: Page Rank Algorithm Working Example

3.2. Tables

File Name	Description of Data set	Number of Vertices	Page Rank Algorithm Execution Time(secs)
dataset01.txt	Email data from a large European Research Institution	1005	3.172890E-317
dataset02.txt	Election Data for the post of Wikipedia Administrator	7115	3.172890E-317
dataset03.txt	Gnutella peer-to-peer file sharing network from August 2002	6301	3.172890E-317

3.3. Algorithms

Algorithm 1 Page Rank Algorithm

for *every* $v \in V$ **do**
  v.oldPR $= \frac{1}{N}$
  v.newPR $= 0$
end for
while *true* **do**
  for *every* $v \in V$ **do**
    if $v.links \neq 0$ **then**
      for *every* $u \in$ *adjacency list of* v **do**
        u.newPR $=$ u.newPR $+ \frac{v.oldPR \times d}{v.links}$
      end for
    end if
    if $v.links = 0$ **then**
      for *every* $s \in V$ **do**
        s.newPR $=$ s.newPR $+ \frac{v.oldPR \times d}{N}$
      end for
    end if
    v.newPR $=$ v.newPR $+ \frac{1-d}{N}$
  end for
  int flag = 0 , int change
  for *every* $v \in V$ **do**
    change $= |$(v.newPR - v.oldPR)$|$
    if $change > \epsilon$ **then**
      flag $=$ -1
      break;
    end if
  end for
  if $flag = 0$ **then**
    break;
  end if
else
  for *every* $v \in V$ **do**
    v.oldPR $=$ v.newPR
    v.newPR $= 0$
    flag $= 0$
  end for
end while

Brief Explanation of Page Rank Algorithm:

Firstly, the page ranks of all the vertices are initialised to 1/N i.e., v.oldPR = 1/N where N is the total number of vertices in the graph(lines 1-3).We set v.newPR = 0. Then we begin traversing the adjacency list of all the vertices of the graph one by one.
In case the vertex v in consideration has outgoing links to other vertices (i.e., v.links $\neq$ 0), the contribution of this vertex, v to the new page rank of all the vertices in it's adjacency list is $\frac{v.oldPR \times d}{v.links}$ as can be seen in the above algorithm(lines 7-9).

If in case the vertex, v under consideration has zero outgoing links, i.e., it is a dangling node, it contributes $\frac{v.oldPR \times d}{N}$ to the new page rank of all the vertices in the graph (lines 12-14).

Finally we add $\frac{1-d}{N}$ to the page ranks of all vertices(Line 12). In lines 20-21 we calculate the absolute change in page rank for every vertex with respect to its old page rank. If that value is greater than the ϵ (tolerance) we update the old page rank of the vertex to the calculated page rank, and set the new page rank to zero and repeat the above steps. If this is not the case, we break out of the loop and report the final page ranks (stored in v.newPR).

The algorithm can be better understood with the help of an example as shown in Fig 1: Page Rank Algorithm Working Example.

The above algorithm has been implemented for various data sets and the execution times of which can be found under section 3.2.

4. Time Complexity Analysis of Page Rank Algorithm:

Let's assume that the while loop (line 4) will take run for k iterations, where k is dependent on epsilon and the structure of the input graph, i.e., number of vertices, number of directed edges, and the way they are connected.

The time complexity of FOR loop (line 5):

1. Assuming, we go into the IF loop each time -
   The loop iterates over line 8 once for every edge. So, the time complexity will be the number of edges, i.e., O(E).
2. Assuming, we go into the ELSE loop each time-
   The outer loop (line 5) iterates for N times and the inner loop (line 10) will also iterate for N times. So, the time complexity will be O($N * N$) = O(N^2)

So, the final time complexity of the FOR loop (line 5) will be:

$$= max(O(E), O(N^2))$$
$$= max(O({}^{N}C_2), O(N^2))(worstcasewhenthegraphiscompletelyconnected, edges =^{N} C_2 \approx N^2)$$
$$= O(N^2)$$

The time complexity of the for loop at line 14 will be equal to O(N), where N is the number of vertices as the loop iterates in constant time over each vertex of the graph once.

Thus, the total time taken by each iteration of the while loop will be equal of O($N^2 + N$) $\approx$ O(N^2) Since the while loop runs k times, the total time complexity will be O($k * N^2$).

5. Comparison with alternatives

Hypertext Induced Topic Search (HITS)is also know as hubs and authorities algorithm is a link analysis algorithm developed by Jon Kleinberg. It was designed in 1998 to rate Web pages. It is a predecessor to the page rank algorithm and works upon the principle of search queries and processes its entire in-links and out-links to assign a page rank to the same. At the time of issuing of a search query from the end of the user, HITS first expands the list of relevant pages returned by a search engine and then produces two rankings of the expanded set of pages, authority ranking and hub ranking. The algorithm works upon a web page which is named as authority if the web page is pointed by many hyper links and a web page is named as HUB if the page point to various hyperlinks. Authorities and hubs exhibit a mutually reinforcing relationship: a better hub points to many good authorities, and a better authority is pointed to by many good hubs.The final hub-authority scores of nodes are determined after infinite repetitions of the algorithm. On applying the hub update rule and authority update rule directly and iteratively diverging values are obtained.The following is a depiction of the matrix implementation of the same:

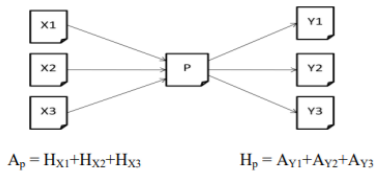


Figure 2: Portrayal of HITS algorithm.

The page rank algorithm offers a wide array of advantages such as:

1. **Reduced query time cost**: Page-Rank has an advantage over the HITS algorithm, as the query-time cost of incorporating the pre-computed Page-Rank importance score for a page is low.
2. **Susceptibility of localized links reduced**: Furthermore, as Page-Rank is generated using the entire Web graph, rather than a small subset, it is less susceptible to localized link spam.
3. **Increased efficiency**: The Page-Rank computes a single measure of quality for a page at its crawl time. This parameter is then combined with a traditional information retrieval score at query time. Compared with HITS, this has the advantage of much greater efficiency .
4. **Increased Feasibility**: In contrast to Hits algorithm the Page-Rank algorithm is more feasible in today's scenario since it performs computations at crawl time rather than query time

There are certain grey areas which can be evolved and need to be recognised as the drawbacks of the page rank algorithm as well:

1. **Dangling Links**: Dangling links problem arises when a page contains a link such that the hypertext points to a page with no outgoing links, this type of link is known as Dangling Link.
2. **Rank Sinks**: The Rank sinks problem occurs when in a network pages get in infinite link cycles. The following figure illustrates the same as well:

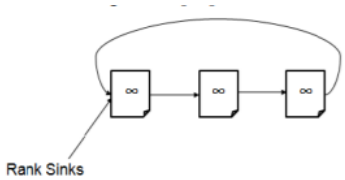


Figure 3: Rank Sink

3. **Spider Traps**: A group of pages is a spider trap if there are no links from within the group to outside the group.
4. **Dead Ends**: Dead Ends are pages with no outgoing links.
5. **Circular References** : The circle references in a website will reduce the front page's Page Rank.The following figure illustrates the same as well:

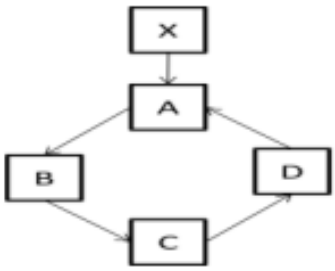


Figure 4: Circular References problem

6. Page Rank score of a page ignores whether or not the page is relevant to the query at hand.
7. Page Rank doesn't handle pages with no out edges very well, because they decrease the Page Rank overall.
8. **Additional Pages**: The addition of a web page to a website will increase the web page's rank by nearly 0.428. The problem with this method is that if the rank of the front page is increased by adding additional pages, than the rank of other pages will go down. The solution is to swap links with websites which have high Page Rank value. The easiest way to do this is to make a page with high Page Rank and link it to the front page.

The following table gives a clear and concise difference between the two:

Criteria	HITS	Page Rank
Basic Criteria	Link analysis algorithm	Link analysis algorithm based on random surfer model
General Algorithm	Web Structure Mining, Web Content Mining	Web Structure Mining
Reinforcement techniques	Mutual reinforcement is the key to HITS, between authority and hub web pages	Page Rank does not work on the distinction between hubs and authorities. It ranks pages just by authority
Influence of neighbouring pages	HITS is applied to the local neighborhood of pages surrounding the results of a query	Page Rank is applied to the entire web
Query Dependency	HITS is query dependent	Page Rank is query independent
Relative Stability	Unstable when changing a few links can lead to quite different rankings	Unstable when changing a few links can lead to quite different rankings.
Input Parameter	Content, Back and Forward links	Back links
Scope of analysis	Single Page	Single Page
Relevancy	Since this algorithm ranks the pages on the indexing time	More since this algorithm uses the hyperlinks to give good results and also consider the content of the page
Worst case time complexity analysis	O(k*N^2) where N represents the number of nodes and k is the number of iterations	O($k * E$) where E represents the number of edges and k is the number of iterations
Relative quality of obtained results	Relatively inferior than page rank algorithm	Relatively better than HITS
Efficiency	HITS invokes traditional search engine to retrieve set of pages relevant to it for a given query and then attempts to find hubs and authorities. Since this computation is carried out at query time, it is not feasible for today's search engines, which need to handle millions of queries per day	Page Rank computes a single measure of quality for a page at crawl time. This measure is then combined with a traditional information retrieval score at query time. The advantage is much greater efficiency

Merits	1. Hub and Authority values prioritise relevant and important pages while calculations 2. The retrieved data is ranked with HITS, a general algorithm used for calculating the authority and hubs. 3. The basic aim of that algorithm is to induce the Web graph by finding set of pages with a search on a given topic (query). 4. It is good in calculating the authority nodes and hubs.	1. Query-time cost of incorporating precomputed Page Rank importance score for a page is low 2. Since the algorithm does not use a small subset and instead derives the result from the whole web page, it is less susceptible to the localised links. 3. Page Rank may be used as a methodology to measure the impact of a community like the blogosphere on the overall Web itself. 4. It has frequent mentions in journals and citations and Google used the technology for ranking web pages.
Limitations	1. Irrelevant Hubs problem 2. Query Dependency 3. Topic Drift 4. Irrelevant authorities problem 5. Mutually reinforcing relationships between hosts problematic	1. Dangling Links 2. Additional Pages 3. Rank Sinks 4. Dead Ends 5. Circular References 6. Spider Traps

6. Improvements and further customisation

1. **Weights**:
   Edge weights often change the relative values of pages in between a networked graph, how each link contributes to the final outcome. In the Page Rank algorithm, all edges are given a uniform value of one by default. We can use this attribute to have the network or the corresponder to pass less value through certain defined edge types. Link positions can be used to assign link scores/parameter, which we can use for weights. The assignment allows a user to label certain link types, such as footer links and other boilerplate links, as low-value internal links. The parameter/weight can also incorporate for the deteriorating values links experience as the in-link count goes up. The user can reduce the weight of edges that go to a page with an extreme in-link count.
2. **Personalization**:
   The personalization factor assigns a weight or parameter to every node that influences a new random surfer walk restart. It influences the walk of the model towards certain desired node(s) whose probability has risen from 1/N. This can then be implemented reflect external link value such that the nodes with more external links will have a greater probability of being the site's entry point (the starting point of a random walk).
   The personalization modification also allows the user to measure the centrality relative to a specific node or subset of nodes. A subset of nodes can be labelled and given personalization values. Outside of Search Engine Optimisation (SEO), this could be used to implement ideas of recommendation systems, spam detection, and fraud detection by finding which nodes are most discoverable or vulnerable relative to a specified subset.

3. **N-Start**:
   Every node in the graph is assigned a starting page rank value other than. Without the nstart modification, all nodes start with a uniform value of 1/N, N being the number of nodes in a graph. This customisation does not change the outcome of final obtained values, as Page Rank should still converge on the same value as it would without. However, this can speed up the time it takes to calculate Page Rank if the initial values are closer to the final value than the default uniform distribution.

Theorem 6.1. *The page rank iteration is said to converge if* $||r^{(t)} - r^{(t-1)}|| < \epsilon$ *for small* ϵ *where* $r^{(t)}$ *represents the page rank vector after iteration t i.e., the row vector containing page ranks of all vertices after iteration t. In other words, the convergence criteria for page rank iterations is the Euclidean distance between two successive page rank vectors.*

Theorem 6.2. *Let f(N) and g(N) be some arbitrary real valued functions of N and let* T_1 *and* T_2 *be the time complexity of f(N) and g(N) respectively, i.e.,* $T_1(N) = O(f(N))$ *and* $T_2(N) = O(g(N))$
Then, $T_1(N) + T_2(N) = max(\ O(f(N)),\ O(g(N))\ )$
and $T_1(N) * T_2(N) = O(f(N) * g(N))$

7. Conclusions

Collecting information and gathering insight from data is an ever increasing business opportunity, data gathering and machine learning can prove to be very complex processes, using graph based solutions and algorithms like Page rank can provide valid alternatives. Their strength is in simplicity: classic machine learning data might be very high dimensional and obtaining it can be an obstacle itself whereas graphs are simple and can be easily built with inferred or public information. The project concluded with subsequent in-depth analysis of how the page rank algorithm works out along with its mathematical and theoretical intricacy. The comparison of page rank and HITS suggests the methodology to interchange between them under different situations. Apart from fundamental algorithm and analysis the further innovative methods to customise the page rank with parameters such as weights of edges and personalisation are methods that are used to model real world SEO techniques with appropriate optimisation as well.

8. Bibliography and citations

Various sources including online articles and websites were referred to for obtaining necessary information and certain charts, images, and table were referenced and used for page rank algorithm. The list of these references can be found below under the reference section.
[1**?** –9]

Acknowledgements

We thank Dr Anil Shukla for providing us the opportunity and Ms Simran Setia for their extremely helpful guidance and innovative insights on our project which have helped us throughout the course to build, ammend and present the project in a better form.

References

[1] Jamie Arians. The google pagerank algorithm.

[2] Briggsby. Personalized pagerank with edge weights, 2021.

[3] Washington Edu. Implementing pagerank.

[4] Adnan Burak Gurdag. A parallel implementation of a link–based ranking algorithm for web search engines. *Journal of Applied Mathematics*, 2013.

[5] Oracle Labs. Pagerank algorithms.

[6] Ritika Wason Nidhi Grover. Comparative analysis of pagerank and hits algorithms. *International Journal of Engineering Research Technology (IJERT)*, 2012.

[7] Stack Overflow. What is pageranks big-o complexity?, 2021.

[8] Wikipedia. Pagerank.

[9] Chun Wen De-An Wu Yue Xie, Ting-Zhu Huang. An improved approach to the pagerank problems. *Journal of Applied Mathematics*, 2013.