

Randomized PageRank Algorithm

Efficiency Analysis

Project Final Report

Prepared by

Khoushik Raj Rasumalla

April 2024

Abstract:

This project explores a modified version of the traditional PageRank algorithm called Randomized PageRank algorithm. The main approach is incorporating random hops and random weights assignment based on the number of inbound links. The key findings show some promise of potential improvements on the efficiency. As a result, the Randomized PageRank algorithm outperforms the current PageRank algorithm when it is being applied on a small subset of data. But as the size of the dataset increases, the computation time of the Randomized PageRank increases as well. This study shows us some possibilities of having a better variation of the PageRank algorithm, which has ramifications for various applications such as web search engines, recommendation systems as well as network analysis.

Introduction:

The Page ranking is one of the basics in any search engine for making user tasks easier in browsing the right web page based on accuracy, quality of data and origin etc. The page Rank algorithm plays a crucial role in efficiently and accurately ranking the web pages [8]. In this study, we aim to analyze the computational complexity and the convergence behavior of the randomized PageRank algorithm. Building upon existing research on PageRank algorithms, we extend it by focusing on the efficiency aspect.

Our methodology involves conducting empirical experiments with a range of web graph datasets to thoroughly assess the efficiency of the algorithm. Through gaining insights into how the algorithm behaves in different scenarios, our objective is to provide guidance for making well-informed decisions in algorithmic design. Understanding the efficiency of the randomized PageRank algorithm has practical implications for web search engines and recommendation

systems. This study's findings could inform the development of more efficient ranking algorithms and enhance the performance of web-based applications.

Problem Definition:

In the digital age, the internet has become an essential source of information, with web page ranking algorithms serving as the foundation for search engines and recommendation systems. Among these algorithms, the PageRank algorithm is notable for its ability to provide an objective estimate of a web page's relevance based on its link structure [10]. However, as the volume and complexity of web data continue to increase exponentially, the scalability and efficiency of ranking algorithms become more important.

As a result, the challenge includes a thorough examination of the Randomized PageRank algorithm's efficiency, which is an extent of the PageRank algorithm. The main motivation behind the choice of examining the Randomized PageRank algorithm is to find out whether there exists a better version of the current PageRank algorithm. Our team will apply the methodologies and approaches that we learnt from the *Randomized Algorithm* course by introducing randomness to the original PageRank.

Furthermore, the challenge includes identifying potential trade-offs between computational efficiency and ranking accuracy, as well as investigating optimization ways to achieve a balance between the two. Furthermore, the evaluation of algorithmic efficiency extends beyond simple performance measurements to include larger implications for web search engines, recommendation systems, and network analysis. For instance, efficient ranking algorithms can improve the user experience by returning timely and relevant search results, enhancing user engagement.

Additionally, they can help with applications like social networks, e-commerce, and academic research that require more efficient information retrieval. Furthermore, gaining a grasp of the randomized PageRank algorithm's efficiency creates opportunities for cross-disciplinary innovation and cooperation. Our study could potentially lead to breakthroughs in network science, algorithmic design, and data analytics by bridging the gaps between computer science, mathematics, and information theory. The issue formulation ultimately highlights how complex assessing algorithmic efficiency is and how broad its consequences are for the digital world and beyond.

Related \York:

Understanding the concepts and performance characteristics of web page ranking algorithms is made easier by the vast research that has already been conducted on a variety of web page ranking algorithm topics. By using the web's link structure to determine a page's significance score, traditional algorithms like PageRank, which were first presented by Brin and Page [10], completely changed online search engines. Expanding on this basis, later research has concentrated on improving ranking algorithm accuracy and efficiency via computational and methodological advancements.

Personalized PageRank was first presented by Haveliwala which improves the relevancy of search results by customizing ranking results to each individual user's preferences. Personalized search engines and recommendation systems have benefited greatly from this personalization [5]. In a similar spirit, TrustRank, a PageRank version that includes trust values to strengthen ranking results' reliability and fight online spam [4]. Furthermore, a thorough analysis of the underlying mathematics and algorithms underpinning PageRank as well as its

applications in several disciplines may be found in Langville and Meyer's extensive work on Google's PageRank and beyond [6]. For those who are interested, especially researchers and practitioners, in learning more about web page ranking algorithms and their implications, the book mentioned can be a valuable resource.

[Models/Algorithms/Measures:](#)

As most of us may already know, the current PageRank algorithm developed by Google already has some randomness incorporated [3]. For instance, the damping factor is one of the things that make the current PageRank algorithm random. The term damping factor refers to the factor that simulates random surfing in the PageRank algorithm which can help prevent dead ends during the score calculation. Dead ends can happen when a web surfer gets trapped in cases where there are no outbound links from the current page which can also be known as dangling nodes.

The default damping factor for the original PageRank algorithm is 0.85 which tells us that there is an 85% chance that the web surfer will continue following the outbound links on the current page. The other 15% is the chance that they will jump to a completely random or unrelated page [9]. This was chosen by the team at Google who tested out different dangling factors and tried to find the one with the best balance and would give a great result.

The current PageRank algorithm calculates the PageRank score through an iterative approach, starting off by assigning equal scores to each page and recalculating the score based on both the number and the quality of inbound links of each web page [2]. This process continues until the score converges to a stable point or in other words, the algorithm stops when the PageRank scores assigned to each web page no longer changes significantly between each

iteration. The rate at which it takes for the PageRank algorithm to reach the stable point can be called *Convergence rate*. This rate can be affected by various factors such as the size of the web graph, the damping factor and the link distribution [1].

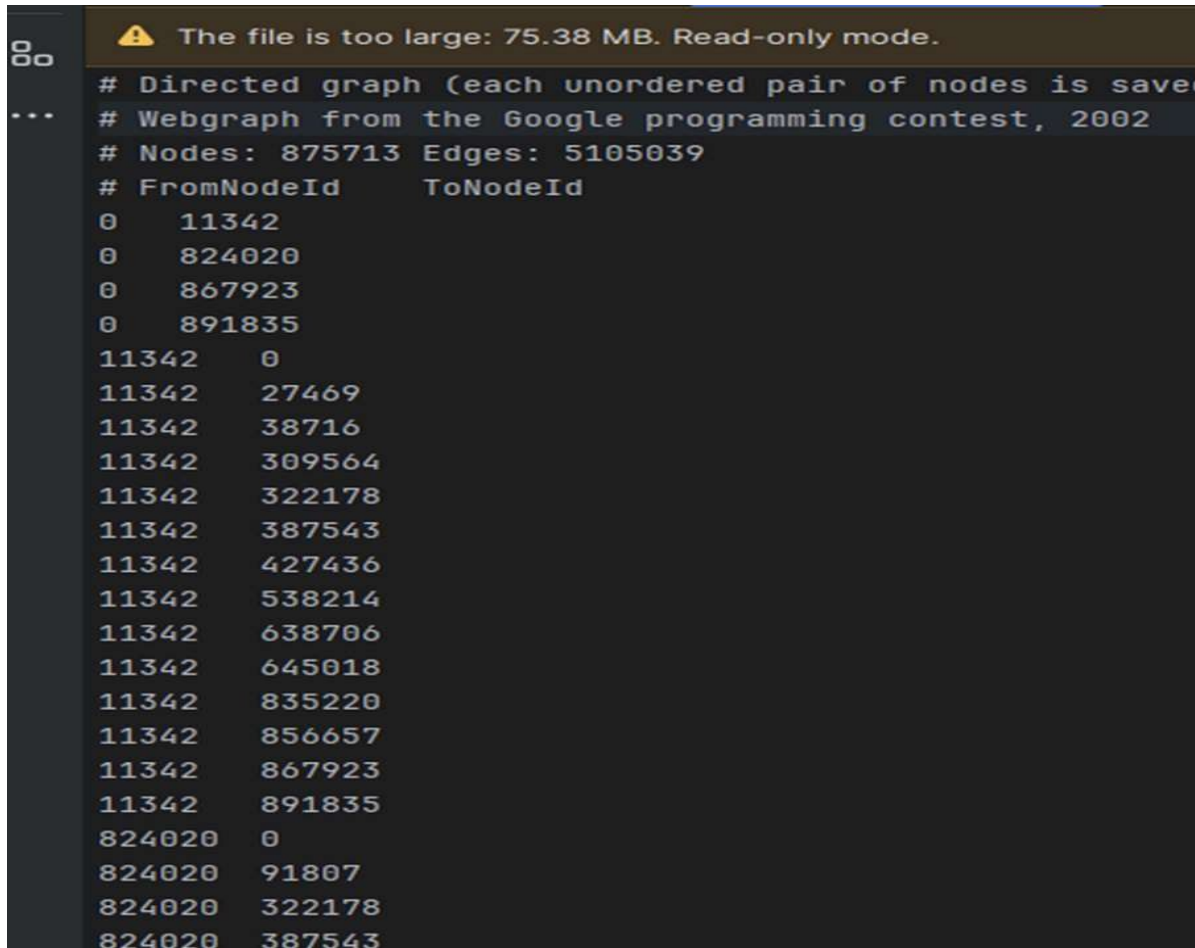
For this project, our team worked on modifying the current deterministic PageRank algorithm by incorporating randomness through random hops and random weights assignment. This means that the random surfer would have to hop to a completely random web page that is not related at all instead of following the outgoing link from the current web page every time. Instead of following a uniform probability distribution, this random hop follows the probability distribution that is calculated based on the number of incoming links toward each node. In other words, the web pages with more in-degree nodes are more likely to get chosen randomly since they are considered to be more important.

There are different factors and measures that we should take into account when analyzing the performance efficiency of both the current PageRank and the Randomized PageRank such as *computation time, accuracy, convergence rate, scalability and memory usage*. Due to the time constraint for this project, our team decided to focus mainly on the computation time and the convergence rate of both algorithms.

Implementation/Analysis:

Our team has decided to conduct an experiment on the *Google Web Graph* dataset from the Stanford Large Network Dataset collection [11]. This dataset includes a directed graph with:

- 875,713 nodes, representing the web pages
- 5,105,039 edges representing the hyperlinks between the web pages



The screenshot shows a text editor window with a warning message at the top: "The file is too large: 75.38 MB. Read-only mode." Below the warning, the file content is displayed in a monospaced font. The content starts with a comment: "# Directed graph (each unordered pair of nodes is save". This is followed by another comment: "# Webgraph from the Google programming contest, 2002". Then, the number of nodes and edges are listed: "# Nodes: 875713 Edges: 5105039". The main part of the file is a list of edges, each represented by two numbers: "FromNodeId" and "ToNodeId". The first few lines of the edge list are: "0 11342", "0 824020", "0 867923", and "0 891835". This is followed by a series of edges starting with "11342" as the FromNodeId, pointing to various ToNodeIds like 0, 27469, 38716, etc. The list continues with edges starting with "824020" as the FromNodeId, pointing to 0, 91807, 322178, and 387543.

```

# Directed graph (each unordered pair of nodes is save
# Webgraph from the Google programming contest, 2002
# Nodes: 875713 Edges: 5105039
# FromNodeId    ToNodeId
0      11342
0      824020
0      867923
0      891835
11342    0
11342    27469
11342    38716
11342    309564
11342    322178
11342    387543
11342    427436
11342    538214
11342    638706
11342    645018
11342    835220
11342    856657
11342    867923
11342    891835
824020    0
824020    91807
824020    322178
824020    387543

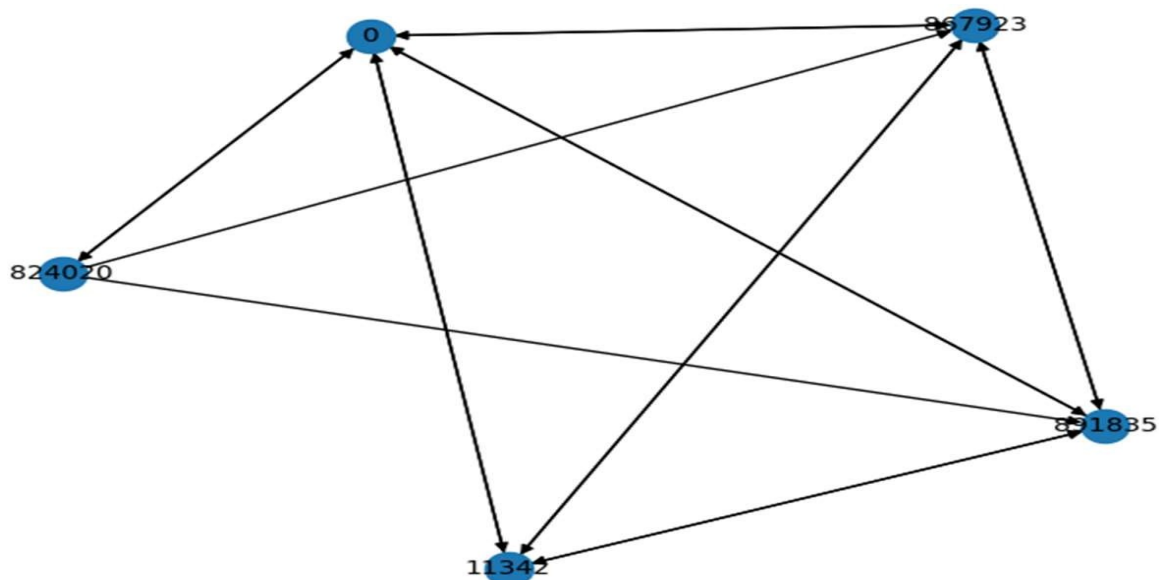
```

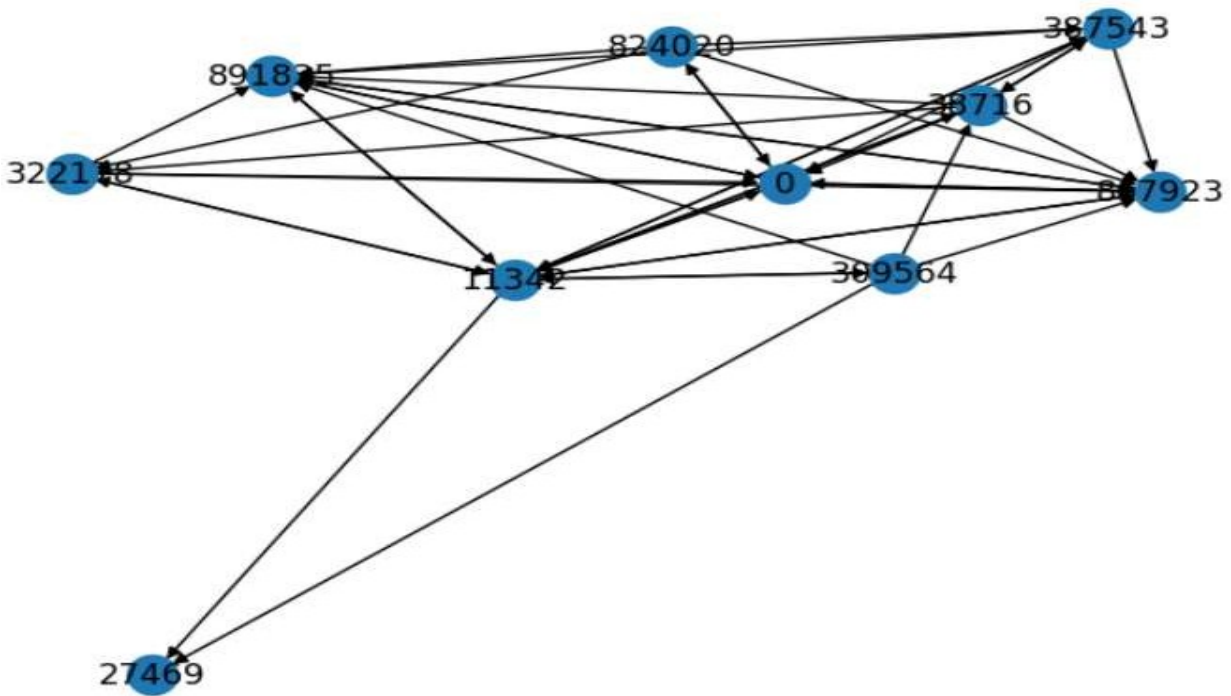
The visualization above illustrates the structure of the chosen dataset. The nodes in *FromNodeId* have links pointing toward nodes in *ToNodeId*. As we can see, the dataset is large and shows complex but important connections between web pages, especially the inbound and outbound links for each node. Our team has decided to apply both the original and the modified PageRank algorithm on subsets of the data instead of the whole dataset to compute the PageRank scores and compare their computation time due to the dataset's large size, which might result in a high computation time for this project.

Our team also tested out using different damping factors on the algorithm. We know that the lower the damping factor, the slower the convergence rate and the higher the computation

time [9]. But that would also lead to a more accurate result. There is typically a tradeoff between efficiency and accuracy which means that we have to find a balance between the two. So we started off by reading the chosen dataset and created a directed graph from it using the NetworkX Python library before we went on to the next step which is modifying the current PageRank algorithm to introduce random hops and random weights assignment. For our experiment with the modified PageRank algorithm, we chose ten to be the number of max iterations and a damping factor of 0.8 since we believe that it is a good balance between the accuracy and the efficiency.

The iterative process of computing the PageRank score for the modified PageRank algorithm stays the same as the current one. The main difference is the random hops and the random weights assignment. We calculated the hopping probabilities based on the number of incoming links to a node, meaning that the nodes with higher hopping probabilities or those that are seen to be more important, are more likely to be chosen as a node to hop to. Similarly, the random weights are also assigned based on the number of inbound links. Our team believes that this approach would give a better and meaningful result than the uniform distribution approach.





The two visual representations above show the Google Web Graph with both five and ten nodes respectively. The complexity of the graph increases proportionally with the size of the nodes or web pages. As we can imagine, the directed graph would become even more complex as the number of the total web pages increases and this makes the task of visualizing and ranking websites on the search engine complicated.

Results & Discussion:

As we can see from the output after running both the algorithms on the selected subset of data, the Randomized PageRank outperforms the current PageRank algorithm on a data subset, with faster computation time. This tells us that our modified PageRank algorithm does not work well as the complexity of the web graph increases. As the size of the dataset increases, we can see a decline in its performance. While introducing more randomness into the current PageRank

algorithm has shown some promise in an improved performance, there are still some limitations and room for improvements in our approach and methodology.

Running the algorithms on a graph with **100** nodes (or web pages)

```
def main():
    data_file = './web-Google.txt'
    # Read the Google web graph dataset and create a directed graph from it
    G = nx.read_adjlist(data_file, create_using=nx.DiGraph())
    sub_nodes = list(G.nodes())[:100]

main x
```

"C:\Users\Both\Desktop\Spring 2024 (Master's)\CPTS580\PageRank\RandomizedPageRank\
 Computation time for Original PageRank: 0.09730744361877441 s
 Computation time for Randomized PageRank: 0.059621334075927734 s

- We applied both algorithms to a subset of the original Google Web graph dataset, in which only 100 nodes were selected. We can see that the computation time for the Original PageRank algorithm is slightly higher than the computation time for the Randomized PageRank algorithm.

Running the algorithms on a graph with **5000** nodes (or web pages)

```
83 def main():
84     data_file = './web-Google.txt'
85     # Read the Google web graph dataset and create a directed graph from it
86     G = nx.read_adjlist(data_file, create_using=nx.DiGraph())
87     sub_nodes = list(G.nodes())[:5000]
88     # Generate the subgraph of the selected node
89     subgraph = G.subgraph(sub_nodes)

un main x
```

"C:\Users\Both\Desktop\Spring 2024 (Master's)\CPTS580\PageRank\RandomizedPageRa
 Computation time for Original PageRank: 0.09012365341186523 s
 Computation time for Randomized PageRank: 0.3132972717285156 s

- Similarly, we applied the two algorithms to a subset of the original Google Web graph dataset, but this time 5000 nodes were selected. As a result, the computation time for the Randomized PageRank algorithm is higher than that of the Original PageRank in this case.

Conclusion and future work:

By incorporating both random hops and random weights assignment, the Randomized PageRank algorithm shows an improved performance, in terms of computation time, on smaller subsets of selected web pages. But unfortunately as the size and complexity of the web graph increases, the performance of the Randomized PageRank algorithm declines. Our team was hoping that we would get better results but it did not turn out that way. Either way, we learned a lot from the study and experiments.

There are more factors and parameters in the algorithm that can be taken into account for future research directions to improve the performance as well as scalability. Some alternative methods for introducing randomness into the current PageRank algorithm, to potentially get a better result, includes *Sampling methods, Graph partitioning and compression* [7]. These explorations can be done in future work or research. It would be a great achievement if we could figure out a way to improve the performance of the current PageRank algorithm since the algorithm is not only important in ranking web pages for search engines but also in various fields such as *Computer systems, Neuroscience, Physics, Chemistry, Bio and Bioinformatics* [3].

Appendix:

Github repo to our Python Code for this project:

<https://github.com/Phearakbothbunna/CPTS591-RandomizedPageRank>

Bibliography

- [1] C. F. Ipsen, Ilse, and Kirkland, Steve. *Convergence Analysis of PAGERANK ...*,
<https://epubs.siam.org/doi/abs/10.1137/S0895479804439808>. Accessed 25 Apr. 2024.

- [2] GeeksforGeeks. “Page Rank Algorithm and Implementation.” *GeeksforGeeks*, 6 Sept. 2022,
www.geeksforgeeks.org/page-rank-algorithm-implementation/. Accessed 18 Apr. 2024.

- [3] Gleich, David. *PageRank Beyond the Web*, <https://epubs.siam.org/doi/10.1137/140976649>.
 Accessed 18 Apr. 2024.

- [4] Gyöngyi, Zoltán, and Jan Pedersen. *Combating Web Spam with TrustRank*,
www.researchgate.net/publication/262407049_Combating_Web_Spam_with_TrustRank.
 Accessed 22 Apr. 2024.

- [5] Haveliwala, Taher. *An Analytical Comparison of Approaches to Personalizing ...*,
nip.stanford.edu/pubs/haveliwala2003personalizing.pdf. Accessed 23 Apr. 2024.

- [6] Langville, Amy N., and Carl Dean Meyer. *Google’s Pagerank and beyond. The Science of Search Engine Rankings*. Princeton University Press, 2012,
press.princeton.edu/books/ebook/9781400830329/googles-pagerank-and-beyond.

- [7] Leskovec, Jure, and Faloutsos, Christos. *Sampling from Large Graphs - CMU School of Computer...*, www.cs.cmu.edu/~jure/pub/sampling-paper.pdf. Accessed 21 Apr. 2024.

- [8] Roberts, Eric. *The Google Pagerank Algorithm*,
web.stanford.edu/class/cs54n/handouts/24-GooglePageRankAlgorithm.pdf. Accessed 19 Apr. 2024.
- [9] Santini, Massimo. *PageRank as a Function of the Damping Factor*,
[df](#). Accessed 22 Apr. 2024.
- [10] Sergey, Brin and Lawrence, Page. “*The Anatomy of a Large-Scale Hypertextual Web Search Engine.*” *Google Research*,
research.google/pubs/the-anatomy-of-a-large-scale-hypertextual-web-search-engine/.
 Accessed 25 Apr. 2024.
- [11] *Stanford Large Network Dataset Collection*, snap.stanford.edu/data/#web. Accessed 17 Apr. 2024.