**NANYANG TECHNOLOGICAL UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

A close up of a logo

Description automatically generated

**SC4052 / CE4052 / CZ4052 Assignment**

**AY 2023-2024**

**Assignment 2: Google PageRank Algorithm & MapReduce**

Submitted by:

**Dion Toh Siyong**

**U2021674D**

# Introduction

The aim of this study is to explore the PageRank algorithm and its parameters. Graphs with smaller number of nodes will be used first to show the implementation of my PageRank algorithm, showing that there is indeed convergence. Followed by testing on graphs with a larger number of nodes. Finally, we will experiment with PageRank and MapReduce for parallel computing.

## What is PageRank

PageRank is an algorithm developed by Larry Page and Sergey Brin. It is a method that is used by search engines to rank web pages in their search results. It assigns a numerical weight to each element of a hyperlinked set of documents, such as web pages, with the purpose of measuring its relative importance within the set. The algorithm works by analysing the links between pages, considering both the quantity and quality of links. Pages that are considered more important by PageRank receive a higher weight or rank. This creates a system where the most relevant and authoritative pages are prioritized in search engine results.

## How is importance measured?

PageRank considers both the quantity and quality of links pointing to a particular page to determine its importance within a network of interconnected pages.

* **Quantity of Links:** A page is considered more important if it has a higher number of incoming links from other pages.
* **Quality of Links:** Not all incoming links are treated equally. PageRank also considers the quality of the pages linking to the target page. Links from highly reputable or authoritative pages carry more weight in determining the importance of the linked page.

# PageRank Algorithm Experimentation

## Iterative PageRank Implementation

I implemented 2 different Iterative PageRank algorithms. Simplified PageRank and Modified PageRank. Simplified PageRank follows the basic principles of the PageRank algorithm without considering additional factors such as dampening or dead ends (dangling nodes). Modified PageRank however, introduces a dampening factor, as well as a boredom distribution, representing the probability of a user getting bored and jumping to another node directly.

Both PageRank algorithms have a default threshold value of 1e-13, which is used to check for convergence. Both algorithms also have a max iteration of 1000. The boredom distribution in the Modified PageRank algorithm has a default value of for each node, where is the total number of nodes in the graph. This can be changed to give certain nodes a higher probability of a user directly jumping to that node.

## Closed Form PageRank Implementation

The closed form solution is calculated using the following equation:

|  |  |
| --- | --- |
|  | 2.1 |

Where is the PageRank Vector, is the transition matrix representing the web graph, is the identity matrix and is the damping factor.

## Spider Trap Example (Lecture 7 Page 53)

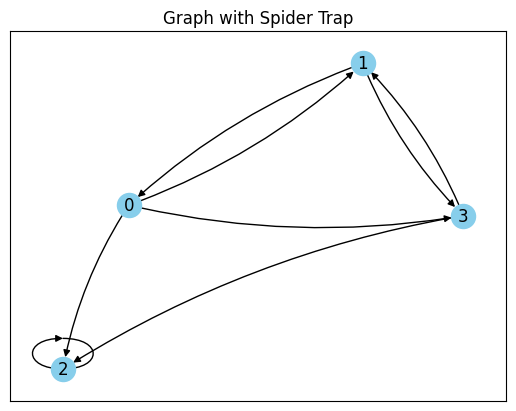


Figure 1: Example Graph given in Lecture Notes 7, Page 53

Figure 1 shows the graph of the example given in the Lecture Notes. This graph contains a spider trap at node 2, where it has 2 incoming links, and only 1 outgoing link back to itself. We expect the Simplified PageRank algorithm to not be able to handle the calculation of PageRanks in this graph due to the spider trap. Table 1 shows the results of both PageRank algorithms.

Table 1: Results of PageRank algorithms against graph in Figure 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PageRank Algorithm | Iterative PageRank | Closed Form PageRank | Number of Iterations | Sum of PageRank |
| Simplified PageRank | [0. 0. 1. 0.] | [0.1014 0.1284 0.6419 0.1284] | 93 | 1 |
| Modified PageRank | [0.1014 0.1284 0.6419 0.1284] | [0.1014 0.1284 0.6419 0.1284] | 55 | 1 |

The results shown in Table 1 shows that the Simplified PageRank algorithm did not converge to the closed form result, while the Modified PageRank algorithm did. This highlights the importance of considering additional factors such as damping factors when implementing PageRank algorithms, especially in scenarios where issues like spider traps may arise. While the Simplified PageRank algorithm may suffice for basic applications, the Modified PageRank algorithm offers better robustness and performance in dealing with complex web graphs. Diagrams of the individual algorithm’s convergence can be found in Figure 2.

|  |  |
| --- | --- |
| A graph with a green line  Description automatically generated | A graph of a number of different colored lines  Description automatically generated |
| Simplified PageRank | Modified PageRank |

Figure 2: PageRank Convergence

## Parameter Exploration and Tuning

Going forward with the rest of the experiments, we will be using a new PageRank algorithm that is able to handle dead-end nodes. This algorithm will be simply referred to as Updated PageRank algorithm.

### Evaluation Metrics

To evaluate our algorithm with various parameter configurations, the following evaluation metrics will be used:

* **Iterations:** How quickly the algorithm converges to stable PageRank scores. This can be assessed by tracking the change in PageRank scores across iterations.
* **Max Error from Closed Form Solution:** The error between the iterative approach against the closed form solution.
* **Scalability:** Measure the algorithm's scalability with respect to the size of the graph.

### Handling of Dead-End Nodes

A graph with arrows and points

Description automatically generated

Figure 3: Example of a Graph with a Dead-End node

Dead-end nodes are nodes with incoming links, but no outgoing links. Figure 3 shows an example of a graph containing a Dead-End node, Node 2. The Updated PageRank algorithm is the Modified PageRank algorithm with additional checks implemented to handle such situations.

Before commencing convergence calculation, checks on the adjacency matrix is first done to identify any dead-end nodes. This can be done by find a column of 0. This indicates 0 outgoing links from the node. We will then replace this column with a column that contains values of , where is the total number of nodes. By doing this, users who end up at the dead-end node can ‘teleport’ away to another node, with all nodes having equal probability.

### Graph Generation

To further test our PageRank algorithm, a function was created to generate a graph with n nodes. It first creates a square matrix with 0, then iteratively goes through the matrix, and randomly assign edges. It takes in the following parameters:

* **n:** The number of nodes in the graph.
* **p:** Probability that there exists a directed edge between 2 nodes.

A diagram of a network

Description automatically generated

Figure 4: Example of a Generated Graph with 25 Nodes and 463 Edges

### Dampening Factor Tuning and Evaluation

For the following tests, we generated a graph with a total of 10,000 nodes and 75,000,810 edges.

Table 2: Results from Dampening Factor Tuning

|  |  |  |
| --- | --- | --- |
| Dampening Factor | Iterations | Max Error from Closed Form Solution (e-07) |
| 0.50 | 6 | 7.7639 |
| 0.55 | 6 | 6.6556 |
| 0.60 | 6 | 5.5470 |
| 0.65 | 6 | 4.4381 |
| 0.70 | 6 | 3.3290 |
| 0.75 | 6 | 2.2196 |
| 0.80 | 6 | 1.1099 |
| 0.85 | 6 | 0.0 |
| 0.90 | 6 | 1.1102 |
| 0.95 | 6 | 2.2207 |

From Table 2, we can see that the number of iterations till convergence across all Dampening Factors remained the same at 6. However, it is evident that the Dampening Factor of 0.85 is the best due to its lowest error, with it exactly converging to the Close Form Solution.

### Scalability

Using the Dampening Factor of 0.85, we will evaluate its scalability.

Table 3: Scalability Evaluation Results

|  |  |  |
| --- | --- | --- |
| Nodes / Edges | Iterations | Max Error from Closed Form Solution (e-07) |
| 10,000 / 75,000,810 | 6 | 0.0 |
| 20,000 / 299,996,289 | 6 | 0.0 |
| 30,000 / 674,992,682 | 6 | Out of memory |

I was able to evaluate up to 20,000 nodes and 299,996,289 edges. Upon trying to get the Close Form solution for a graph with 30,000 nodes, I face an “Out of memory” error. However, with 20,000 nodes, it was still able to converge to the Closed Form solution in 6 iterations, which is pretty impressive. This could be better evaluated with more memory in my system.

# Parallel Programming with MapReduce

# Conclusion

# References

# Appendix