Using PageRank to compute the importance of airports

Patrik Palviainen Marc Parcerisa
Facultat d'Informàtica de Barcelona (FIB), UPC, 08034 Barcelona, Spain

***Abstract*—This work applies the PageRank algorithm to create a relative measure of importance of airports in the Open Flights dataset. By modeling airports as nodes and flight routes as directed edges, we construct a sparse transition matrix and compute PageRank through power iteration. A central challenge is the presence of dead-end nodes (airports with no outgoing routes), which can disrupt probability conservation. We therefore compare three different strategies for handling dead-end nodes. We further analyze the impact of damping factors on convergence behavior and PageRank distribution, highlighting non-trivial dependencies. Overall, we demonstrate that PageRank is an effective method for ranking airports by structural importance while analyzing the role of dead-end handling in producing probability vectors.**

I. Introduction

In the early days of the internet, when it was catalogs, travel blogs, and a chatroom or two, a new problem started to arise: Searching through the massive amounts of information that constituted the entire world wide web. And whoever solved this problem gained the great power (and the great responsibility) to control the information that the population had access to, and how they did so.

The initial challenges of scaling web information led to varied early solutions: Yahoo explored curated directories, a method that quickly proved unsustainable as web content exploded [1]; others, like AltaVista, relied on early full-text search engines, which often struggled with spam and ranking quality. The solution emerged with Google's innovative approach, which utilized the PageRank algorithm to leverage the web's hyperlink structure [2,3].

In this report, we apply the same algorithm *Google* proposed for ranking web pages to compute the importance of each airport in a dataset taken from Open Flights [4].

The PageRank algorithm is based on the concept of a random surfer navigating the world wide web. Each time a surfer lands on a web page, it has a high, uniform, probability of following any hyperlink on it, and a small one to "teleport" to any other web page. The rank of each page is then determined as the probability of a web page appearing on this random surfer's path. One can easily model this stochastic simulation with a Markov chain through all the nodes in the web with the following transition matrix:

$$G = \lambda M^T + \frac{1-\lambda}{n}\mathbf{1}_{n\times n}, \tag{1}$$

where λ is the damping factor, the probability of following a link instead of jumping to any other random page, M is the transition matrix (adjacency matrix, divided by the out degree of each node, or, if the adjacency matrix is weighted, by the sum of all the weights of the outgoing edges), n is the number of nodes, $\mathbf{1}_{n\times n}$ is a square matrix filled with ones, and G is known as the *Google Matrix*.

It can be proven that, thanks to the right hand side of Equation 1, the *Google Matrix* has a principal eigenvector which is found by either solving the following expression

$$p = Gp,$$

or by treating it as a dynamic system which converges to said solution, which is known as the *Power Method*. This last method's complexity scales with the complexity of the matrix multiplication of G times p, which given the sparse topology of the world wide web, can be proven to be extremely effective.

Note, however, that when the original transition matrix M has some rows filled with zeros (dead-end nodes, with no outgoing links), the resulting G matrix is *substochastic* (some of its columns sum less than one, meaning that with probability λ, we loose PageRank mass in that node), in which case its principal eigenvector might not add up to one. In such case, one must implement some bespoke strategies for handling dead-ends. We explore the following two:

A. Always Teleport from Dead Ends

When a dead end is reached, instead of loosing the PageRank mass, we always teleport to any other node at random. This is equivalent to adding a row of ones to the adjacency matrix (or a row of $\frac{1}{n}$ to the transmission matrix) on the rows of the sink nodes. However, if one does so, the resulting matrix M ends up not being sparse, and so the algorithm slows on an order of $O(N_{\text{dead-ends}} \times n)$. A more intelligent solution is to, after each iteration of the *Power Method*, add to all elements of the vector p what would result from the probability mass flux from all dead-end nodes,

$$p = Gp + \sum_{j\in\{\text{dead-ends}\}} \frac{p_j}{n}\mathbf{1}_n.$$

The pseudo-code for this implementation would be as follows:

$A_{ij} := 1$ if $i \to j$ 0 otherwise
$K_i^{out} := \sum_j A_{ij}$
$M_{ij} := A_{ij}/K_i^{out}$
$p_i^0 := 1/n$
do until convergence:

$$\begin{aligned} \text{DEmass} &:= \textstyle\sum_{j\in\{\text{dead-ends}\}} \frac{p_j^t}{n} \\ p^{t+1} &:= M^T p^t + \tfrac{1-\lambda}{n}\mathbf{1}_n + \text{DEmass}\cdot\mathbf{1}_n \end{aligned}$$

where one can see that each iteration's complexity is $O(|\{\text{dead-ends}\}|) + O(M^T p)$, which, if using *Tensorflow* for sparse matrix multiplications, is really fast [5].

B. Always Stay on Dead Ends

When a dead end is reached, instead of losing the PageRank mass, we feed it back to itself. Such a node can only be exited by teleportation, i.e., with the probability of $1 - \lambda$.

This strategy is equivalent to adding a single one to the diagonal of the adjacency matrix on the rows of the sink nodes. This doesn't break the sparsity of the matrix, and so can be implemented out of the box into any *Power Iteration* solution.

II. Methodology

As mentioned in the introduction, all data was taken from the Open Flights Dataset, and processed using *Tensorflow*, for quick, easy and fast sparse matrix multiplications. Note that, on the back-end, all sparse matrix multiplications are exactly equivalent to "raw" *Power Iterations*, in which we would compute each value of the $t + 1$-th p vector as a sum over values of the t-th one.

The proposed implementations can be found attached to this report, or in our GitHub repository [6]. In them, a main file can be found, PageRank.py, which reads the airports and routes from the given csv files, parses them, then computes the sparse adjacency matrix of the whole graph, and performs several iterations of the *Power Method*, stopping when either the maximum number of iterations is reached, or when the maximum difference in probability betwee p^{t+1} and p^t is under some tolerance ε. The *damping*, *tolerance*, *maximum number of iterations*, as well as the *dead end strategy* used are all configurable by command-line arguments. This script, alongside the resulting ranking of airports and their pagerank value, also logs every middle step into a log file, which is then used to generate *convergence plots*, as well as animations and diagrams, shown in the following section.

III. Results

We split this section into three different experiments: *Raw PageRank* (without any dead-end strategy), *Always teleport form dead ends*, and *Always stay in dead ends*. For each dead end strategy, several different *damping* factors were tested, and the results are showed in their respective sections.

A. Raw PageRank

As outlined previously, the raw implementation yields vectors that, although they do order airports with respect to their importance, they cannot be interpreted as probability masses, as they don't add up to one. Figure 1 shows that the sum of the values of the probability vectors rapidly stabilize into something less than 1 for a run of the *Power Method* with damping equal to 0.85. However, Figure 2 shows that each nodes' probability correctly converges to some PageRank value.

Interestingly, we can compare the dependence of the 1-norm to which the vector converges with respect to the damping factor, which can be seen in Figure 3, decreases as the damping factor increases. This can be easily explained, as the damping factor controls what portion of the PageRank mass from the dead-ends gets lost on them. In the extreme case, $\lambda = 1$, we expect that all the probability mass is lost on them, and the resulting 1-norm should be equivalent to the fraction of non-dead-end nodes in the dataset.

Finally, the resulting sorting of the airports, with their PageRank can be seen in the summary Table I.

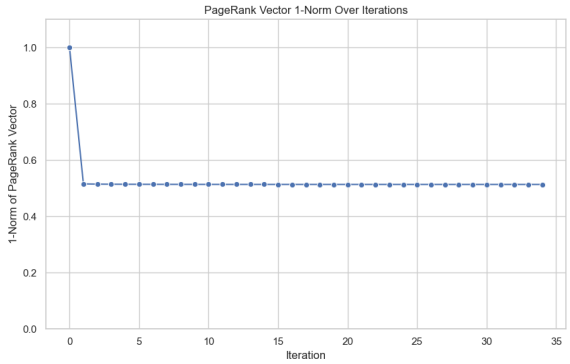

Fig. 1: Evolution of the 1-norm of the vector of probabilities over several iterations of the *Power Method* algorithm for PageRank when ranking airports without any dead end strategy, with a damping factor of 0.85.

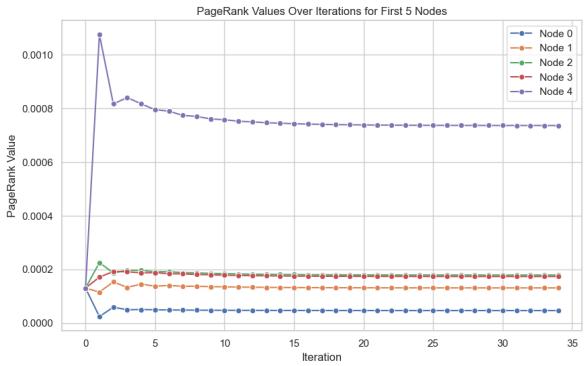

Fig. 2: Evolution of the p values of the first 5 nodes in the list of airports over several iterations of the *Power Method* algorithm for PageRank when ranking airports without any dead end strategy, with a damping factor of 0.85.

B. Always teleport from dead ends

With this strategy, the 1-norm of the vector correctly held across all iterations to numerical precision. Figure 4 shows the dependency of the evolution of the PageRank values of the first five airports with respect to the damping factor. It is shown that not only the final value of all the nodes is dependent on the damping factor, but also the amount of iterations needed

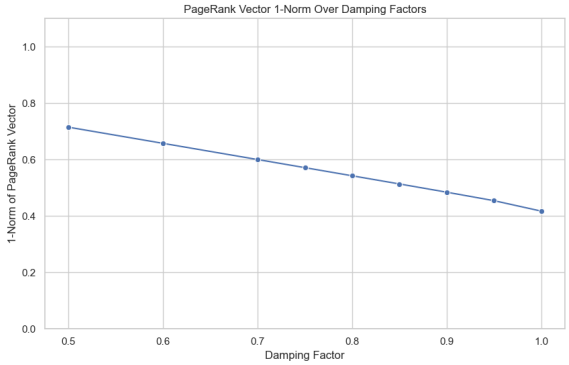

Fig. 3: Dependency of the final norm of the p vector in the *Power Method* algorithm for PageRank with the damping factor λ, when ranking airports without any dead end strategy.

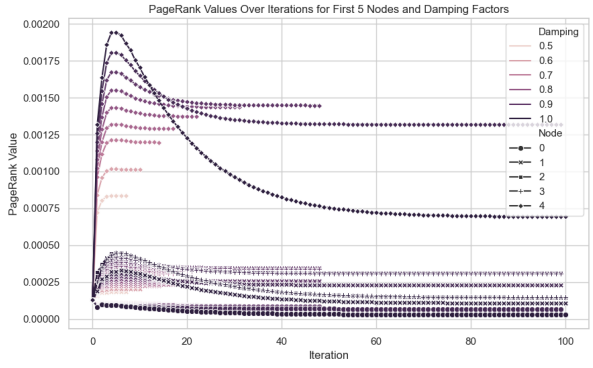

Fig. 4: Evolution of the p values of the first 5 nodes in the list of airports over several iterations of the *Power Method* algorithm for PageRank when ranking airports with an *always teleport from dead ends* strategy. The final convergence value of the different nodes depends on the damping factor, as well as the amount of iterations needed for them to converge.

for convergence. Unexpectedly, there seems to be a non-trivial dependency on the final PageRank value of each node with the damping factor that is shown in Figure 5.

The resulting sorting of the airports, with their PageRank can be seen in the summary Table I.

C. Always stay on dead ends

Again, with this strategy, the 1-norm of the vector holds across all iterations up to numerical precision. Figure 6 shows the evolution of the values of p over the iterations for the first 5 nodes in the airport list with its dependency on the damping factor. A more turbulent evolution seems to be present towards the start of the algorithm, with a final convergence value much lower than the one in the previous section. The reason for this drop in probability is because in this case, dead end nodes are hoarding pagerank values. For comparison, Figure 7 shows the dependency of the average page rank mass of the dead end nodes and the damping factor, compared between the *always teleport from dead ends* strategy and the *always stay*

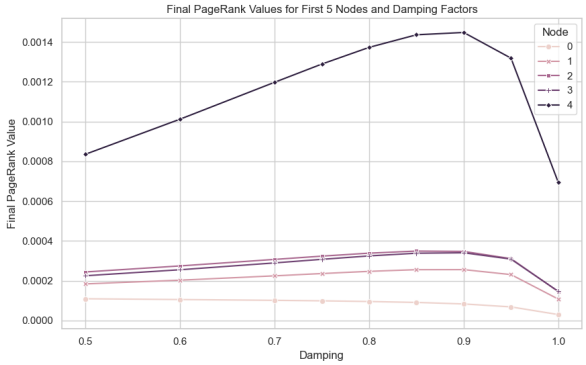

Fig. 5: Dependency of the final PageRank value of the first 5 nodes in the list of airports and the damping used on the *Power Method* algorithm with an *always teleport from dead ends* strategy. A non-trivial dependency can be seen, with a clear maxima around a factor of 0.9.

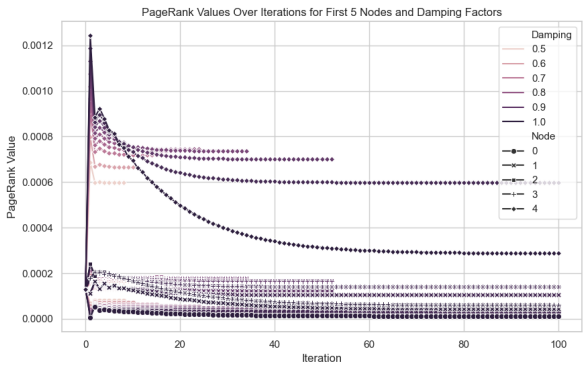

Fig. 6: Evolution of the p values of the first 5 nodes in the list of airports over several iterations of the *Power Method* algorithm for PageRank when ranking airports with an *always stay on dead ends* strategy. The final convergence value of the different nodes depends on the damping factor, as well as the amount of iterations needed for them to converge.

in dead ends one. Obviously, when allowing sink nodes to hoard probability mass, their response to the damping is really small, if any, and dead end nodes end up having much higher rankings than they should.

The resulting sorting of the airports, with their PageRank can be seen in the summary Table I.

D. Final Results

Setting a common damping factor of 0.9 for all strategies, Table I shows the final 10 top-ranked airports, and their PageRank values, compared between strategies. The top ranking doesn't seem to change with the strategy used, but the values to which the algorithm converges do seem to change.

IV. Conclusions

In this report, we successfully applied the PageRank algorithm to rank airports by importance using Open Flights data.

Raw		**Always Teleport**		**Always Stay**	
Airport	PageRank	Airport	PageRank	Airport	PageRank
Los Angeles Intl (LAX)	0.00285	Los Angeles Intl (LAX)	0.00590	Los Angeles Intl (LAX)	0.00285
Chicago Ohare Intl (ORD)	0.00284	Chicago Ohare Intl (ORD)	0.00589	Chicago Ohare Intl (ORD)	0.00284
Denver Intl (DEN)	0.00274	Denver Intl (DEN)	0.00567	Denver Intl (DEN)	0.00274
Heathrow (LHR)	0.00232	Heathrow (LHR)	0.00481	Heathrow (LHR)	0.00232
Charles De Gaulle (CDG)	0.00225	Charles De Gaulle (CDG)	0.00466	Charles De Gaulle (CDG)	0.00225
Capital Intl (PEK)	0.00222	Capital Intl (PEK)	0.00459	Capital Intl (PEK)	0.00222
Frankfurt Main (FRA)	0.00219	Frankfurt Main (FRA)	0.00453	Frankfurt Main (FRA)	0.00219
Changi Intl (SIN)	0.00215	Changi Intl (SIN)	0.00445	Changi Intl (SIN)	0.00215
Hartsfield Jackson Atlanta Intl (ATL)	0.00215	Hartsfield Jackson Atlanta Intl (ATL)	0.00444	Hartsfield Jackson Atlanta Intl (ATL)	0.00215
John F Kennedy Intl (JFK)	0.00203	John F Kennedy Intl (JFK)	0.00419	John F Kennedy Intl (JFK)	0.00203

TABLE I: Comparison of the top 10 ranked airports using PageRank with **A)** raw PageRank, **B)** *always teleport from dead ends* strategy, and **C)** *always stay in dead ends* strategy. Even though the resulting values are completely different, the resulting ranking stays consistent across strategies. The algorithm has correctly identified some of the most busy airports in the world.

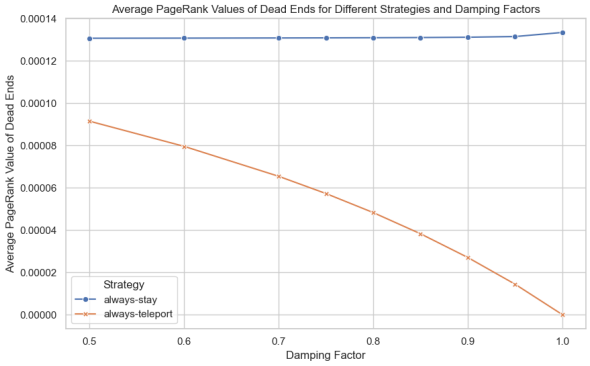

Fig. 7: Dependency of the average PageRank value of the dead-end nodes and the damping factor when solving with either the *always teleport from dead ends* strategy, or the *always stay on dead ends* one. The latter one allows sink nodes to hoard PageRank mass similar to when the damping factor is 1 (no damping).

Three strategies were tested for handling dead-end nodes:

- Raw PageRank: Ranks airports but results in a PageRank vector with a 1-norm less than one, meaning the values aren't true probabilities.
- Always Teleport from Dead Ends: Maintains the vector's 1-norm at one, making the values interpretable as probabilities. This is the more theoretically sound strategy.
- Always Stay on Dead Ends: Also maintains the 1-norm at one, but causes dead-end nodes to "hoard" PageRank mass, resulting in lower PageRank values for non-dead-end nodes.

Crucially, the relative ranking of the top airports remained consistent across all three strategies, successfully identifying many of the world's busiest airports.

V. Future Work

We leave for future work exploring the actual dependencies of the final PageRanks with the damping factors, which can probably be expressed analytically to find the theoretical optimal damping factor, as well as finding a theoretical explanation for the curves of the dependency between the Average Page Rank value of the dead ends with respect to the damping factor.

References

[1] H.-L. Lee and H. A. Olson, "Hierarchical navigation: An exploration of yahoo! directories," *KO KNOWLEDGE ORGANIZATION*, vol. 32, no. 1, pp. 10–24, 2005.

[2] L. Page *et al.*, "The pagerank citation ranking: Bringing order to the web." Stanford infolab, Tech. Rep., 1999.

[3] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[4] OpenFlights, "OpenFlights Airport, Airline, and Route Data," Online Data Resource, January 2017, accessed: November 19, 2025. [Online]. Available: https://openflights.org/data

[5] A. Abboud *et al.*, "The time complexity of fully sparse matrix multiplication," 2023. [Online]. Available: https://arxiv.org/abs/2309.06317

[6] AimbotParce, "MDS-IRRS-Lab5: Code for "Using PageRank to compute the importance of airports"," GitHub repository, 2025, archived Version/Commit: v1.0. [Online]. Available: https://github.com/AimbotParce/MDS-IRRS-Lab5