DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

# TP1 de Métodos Numéricos

11 de septiembre de 2022 Métodos Numéricos

**Grupo: 19**


| Estudiante | LU | Correo electrónico |
|---|---|---|
| Paula Pérez Bianchi | 7/20 | paulitapb@live.com.ar |
| Juan Ignacio Ponce | 420/21 | juaniponce0@gmail.com |
| Valeria Elizabeth Wodka | 39/20 | valewodka@gmail.com |





**Facultad de Ciencias Exactas y Naturales**
Universidad de Buenos Aires
Ciudad Universitaria
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

**Resumen**

En este trabajo analizamos el modelo PageRank, este es utilizado por los motores de búsqueda para ordenar las páginas web según su importancia. Este modelo determina el *ranking* de una página según la cantidad de links que tiene y la calidad de esos links. Pero solo considerar los links entre páginas no modela el comportamiento errático de un usuario que navega en la web. Por esto trabajamos con el modelo del Navegante Aleatorio que introduce la probabilidad de que un usuario decida saltar a una páginas linkeada por su página actual. Para obtener el PageRank de una red de páginas hay que resolver un sistema de ecuaciones lineales que involucra matrices ralas. En nuestra implementación exploramos dos estructuras de representación de matrices ralas: *Dictonary Of Keys* y *Compressed Spare Rows*. Además realizamos experimentos donde simulamos redes de páginas inspiradas en casos reales y observamos como varia el PageRank y la posición en el ranking de una página según cambia la probabilidad del navegante aleatorio.

**Palabras Claves:** *PageRank - Matrices ralas - Navegante Aleatorio*

# 1. Introducción

El modelo de *PageRank* fue introducido por Google en 1998 para estimar el orden de importancia de una página web usando sus links salientes y entrantes. Este modelo fue la clave del éxito de la compañía. Formalmente el *PageRank* de una red es una distribución de probabilidad que nos indica cual es la probabilidad de que una página sea visitada. Además a este modelo se le puede agregar la probabilidad de un Navegante Aleatorio. Esto nos permite modelar el comportamiento de un usuario que va saltando entre las páginas sin respetar los links. Esto nos define un mejor modelo pero sin grandes saltos de dificultad con respecto a la versión base. Si bien este modelo de *PageRank* no es utilizado actualmente por los motores de búsqueda su estudio es muy interesante ya que posee aplicaciones en otros campos de la ciencia como la biología y la neurociencia. Además que nos permite aplicar en un caso de estudio real los temas de la materia.

# 2. Desarrollo

Para calcular el PageRank de cada página necesitamos resolver un sistemas de ecuaciones. Veamos como se define ese sistema.

En primer lugar, vamos a querer representar los links entre las páginas. Estos nos definen un grafo, donde cada página es un nodo y los enlaces son los links entre las páginas. Esto lo podemos representar con una matriz de conectividad $\mathbf{W} \in \{0,1\}^{n \times n}$ definida como:

$$W_{ij} = \begin{cases} 1 & si\ la\ página\ j\ tiene\ un\ link\ a\ la\ página\ i \\ 0 & si\ no \end{cases}$$

En esta matriz cuadrada tenemos en la fila i están las páginas que apuntan a i y en la columna j están las páginas apuntadas por j. Tener en cuenta que no existen los autolinks, $w_{ii} = 0$

Luego, a cada página j le asignaremos su *grado*. Este está determinado por la cantidad de links salientes de la misma:

$$c_j = \sum_{i=1}^{n} w_{ij} \tag{1}$$

Una vez calculado el grado podremos calcular el *puntaje* de la página i como la suma del puntaje ponderado de las páginas j que apuntan a la página i:

$$x_i = \sum_{j=1}^{n} \frac{x_j}{c_j} w_{ij} \tag{2}$$

Entonces podemos definir la matriz de puntajes como $\mathbf{R} = \mathbf{WD}$ donde $\mathbf{D}$ es una matriz diagonal de la siguiente forma:

$$d_{j,j} = \begin{cases} \frac{1}{c_j} & si\ c_j \neq 0 \\ 0 & si\ no \end{cases}$$

Veamos como nos queda cada elemento de esta matriz:

$$WD = \begin{pmatrix} w_{11} & \dots & w_{1j} & \dots & \dots & w_{1n} \\ \vdots & \ddots & \vdots & & & \vdots \\ \vdots & & w_{jj} & & & \vdots \\ \vdots & & \vdots & \ddots & & \vdots \\ \vdots & & \vdots & & \ddots & \vdots \\ w_{n1} & \dots & w_{nj} & \dots & \dots & w_{nn} \end{pmatrix} \begin{pmatrix} d_{11} & \dots & 0 & \dots & \dots & 0 \\ \vdots & \ddots & \vdots & & & \vdots \\ \vdots & & d_{jj} & & & \vdots \\ \vdots & & \vdots & \ddots & & \vdots \\ \vdots & & \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & \dots & \dots & d_{nn} \end{pmatrix} =$$

$$\begin{pmatrix} \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ col_1(W)d_{11} & col_j(W)d_{jj} & col_n(W)d_{nn} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix}$$

Esta matriz $\mathbf{R}$ nos permite calcular el PageRank de todas las páginas de la forma $\mathbf{Rx} = \mathbf{x}$ donde $x_i$ es el puntaje de la $i$-ésima página y por ende el elemento $i$-ésimo $(Rx)_i$ corresponde al puntaje $x_i$ de la ecuación 2.

$$\begin{pmatrix} \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ col_1(W)d_{11} & col_j(W)d_{jj} & col_n(W)d_{nn} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{n} d_{jj} w_{1j} x_j \\ \vdots \\ \sum_{j=1}^{n} d_{jj} w_{jj} x_j \\ \vdots \\ \sum_{j=1}^{n} d_{jj} w_{nj} x_j \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{n} \frac{x_j}{c_j} w_{1j} \\ \vdots \\ \sum_{j=1}^{n} \frac{x_j}{c_j} w_{ij} \\ \vdots \\ \sum_{j=1}^{n} \frac{x_j}{c_j} w_{nj} \end{pmatrix}$$

Con lo cual, podemos ver que cada pagina tiene como puntuación efectivamente lo que habíamos planteado en la ecuación 2

## 2.1. Modelo del navegante aleatorio

El modelo anterior nos permite clasificar $n$ páginas, pero no tiene en cuenta la aleatoriedad del usuario al navegar en la red. En este modelo, el navegante aleatorio empieza en una página cualquiera del conjunto, y luego en cada página j que visita elige con probabilidad $p \in (0, 1)$ si va a seguir uno de sus links, o con probabilidad $1p$, si va a pasar a otra página cualquiera del conjunto. Una vez tomada la decisión, en caso de que se haya elegido una página dentro del conjunto de links que tenga esa página, la probabilidad es de $\frac{1}{c_{ij}}$, caso contrario la probabilidad será de $\frac{1}{n}$. Esto me define una matriz $\mathbf{A} \in \mathbb{R}^{n \times n}$ definida como:

$$a_{ij} = \begin{cases} \frac{(1-p)}{n} + \frac{p w_{ij}}{c_j} & si\ c_j \neq 0 \\ \frac{1}{n} & si\ c_j = 0 \end{cases} \tag{3}$$

Por ende para encontrar la solución al PageRank con navegación aleatoria se debe resolver el sistema:

$$\mathbf{Ax} = \mathbf{x}$$

con $x_i \geq 0 \wedge \sum_i x_i = 1$.

Entonces, el elemento i-ésimo $(\mathbf{A}x)_i$ es la probabilidad de encontrar al navegante aleatorio en la

página i sabiendo que $x_j$ es la probabilidad de encontrarlo en la página j. Entonces, la matriz A puede reescribirse como:

$$\mathbf{A} = p\mathbf{W}\ \mathbf{D} + ez^t$$

Donde **WD** son las que definimos en el modelo anterior y $e \in \mathbb{R}^n$ es un vector columna de unos y z es un vector fila cuyos componentes son:

$$z_j = \begin{cases} \frac{(1-p)}{n} & si\ c_j\ \neq\ 0 \\ \frac{1}{n} & si\ c_j\ =\ 0 \end{cases} \tag{4}$$

### 2.1.1. Justificación A = p W D + e $z^t$

Veamos que la descomposición de A es válida.
Primero, escribamos el resultado de hacer $ez^t$:

$$ez^t = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \begin{pmatrix} z_1 & \dots & z_n \end{pmatrix} = \begin{pmatrix} z_1 & \dots & z_n \\ \vdots & \vdots & \vdots \\ z_1 & \dots & z_n \\ \vdots & \vdots & \vdots \\ z_1 & \dots & z_n \end{pmatrix}$$

Es decir que $ez^t$ nos da una matriz de $\mathbb{R}^{n\times n}$, donde cada fila es el vector $z^t$
Por otro lado, como **D** es diagonal:

$$p\mathbf{W}\ \mathbf{D} = \begin{pmatrix} \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ p\ col_1(W)d_{11} & p\ col_1(W)d_{jj} & p\ col_1(W)d_{nn} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix}$$

Por lo que entonces $p\mathbf{W}\ \mathbf{D} + ez^t$ es una matriz de la pinta:

$$p\mathbf{W}\ \mathbf{D} + ez^t = \begin{pmatrix} \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ p\ col_1(W)d_{11} & p\ col_1(W)d_{jj} & p\ col_1(W)d_{nn} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix} + \begin{pmatrix} z_1 & \dots & z_n \\ \vdots & \vdots & \vdots \\ z_1 & \dots & z_n \\ \vdots & \vdots & \vdots \\ z_1 & \dots & z_n \end{pmatrix} =$$

$$\begin{pmatrix} pw_{11}d_{11} + z_1 & \dots pw_{1j}d_{jj} + z_j & \dots & pw_{1n}d_{nn} + z_n \\ \vdots & \vdots & & \vdots \\ pw_{i1}d_{11} + z_1 & \dots pw_{ij}d_{jj} + z_j & \dots & pw_{in}d_{nn} + z_n \\ \vdots & \vdots & & \vdots \\ pw_{n1}d_{11} + z_1 & \dots pw_{nj}d_{jj} + z_j & \dots & pw_{nn}d_{nn} + z_n \end{pmatrix}$$

Con lo cual, $\forall\ i,j \in \mathbb{R}\ /\ 1 \leq i,j \leq n \Rightarrow (p\mathbf{W}\ \mathbf{D} + ez^t)_{ij} = pw_{ij}d_{jj} + z_j$. Pero este número en realidad presenta dos valores diferentes dependiendo de $c_j$, ya que los valores de $z$ como de $d_{ii}$ dependen de esta condición. Por lo que nos termina quedando:

$$(p\mathbf{W}\ \mathbf{D} + ez^t)_{ij} = \begin{cases} \frac{pw_{ij}}{c_j} + \frac{(1-p)}{n} & si\ c_j\ \neq\ 0 \\ \frac{1}{n} & si\ c_j\ =\ 0 \end{cases} = a_{ij}$$

La matriz $\mathbf{A}$ tiene propiedades especiales que nos permiten resolver el sistema $\mathbf{A}x$ usando Eliminación Gaussiana sin pivoteo. Exploremos esto.

### 2.1.2. Justificación Eliminación Gaussiana sin pivoteo

Para garantizar la eliminación gaussiana sin pivoteo primero queremos ver que la matriz del sistema es **estrictamente diagonal dominante (edd)**, ya que de serlo sabemos que tiene factorización LU, por ende tiene eliminación Gaussiana sin pivoteo. Entonces resta demostrar:

$$A \in \mathbb{R}^{n\times n}, |a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ji}|$$

para todo $1 \leq i \leq n$. En particular, la matriz $\mathbf{A} = \mathbf{I} - p\mathbf{WD}$ la podemos escribir como:

$$\mathbf{I} - p\mathbf{WD} = \begin{pmatrix} 1 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ \vdots & & 1 & & \vdots \\ \vdots & & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 1 \end{pmatrix} - \begin{pmatrix} \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ pcol_1(W)d_{11} & pcol_1(W)d_{jj} & pcol_1(W)d_{nn} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix} =$$

$$\begin{pmatrix} 1 - pw_{11}d_{11} & \dots & -pw_{1j}d_{jj} & \dots & -pw_{1n}d_{nn} \\ \vdots & & \vdots & & \vdots \\ -pw_{j1}d_{11} & \dots & 1 - pw_{jj}d_{jj} & \dots & -pw_{jn}d_{nn} \\ \vdots & & \vdots & & \vdots \\ -pw_{n1}d_{11} & \dots & -pw_{nj}d_{jj} & \dots & 1 - pw_{nn}d_{nn} \end{pmatrix}$$

Luego, veamos que se cumple la definición de **edd**. Sabemos que $|a_{ii}| = |1 - pw_{ii}d_{ii}| = 1$ ya que $w_{ii} = 0$ y sabemos también que $|a_{ji}| = |-pw_{ij}d_{ii}|$. Entonces queremos ver:

$$1 > \sum_{i=1, i\neq j}^{n} |-pw_{ij}d_{ii}|$$

Veamos como acotar la sumatoria:

$$\sum_{i=1, i\neq j}^{n} |-pw_{ij}d_{ii}| =$$

Como sabemos que $w_{ij} \in \{0,1\}$, $p \in (0,1)$ y $d_{jj} \in (0,1)$ por como están definidos podemos sacar el modulo.

$$= \sum_{i=1, i\neq j}^{n} d_{ii}w_{ij}p$$

Luego, $\sum_{i=1,i\neq j}^{n} w_{ij}$ es la suma de la columna j de $\mathbf{W}$ menos el elemento de $w_{ii}$ y $d_{ii} = \frac{1}{c_i}$ si $c_i \neq 0$ El caso donde $c_i = 0$ lo vemos al final.

$$= p\frac{1}{c_i}(\sum col_j(\mathbf{W}) - w_{ii})$$

$$= p\frac{1}{c_i}(\sum_{i=1}^{n} w_{ij} - w_{ii})$$

Pero usando la definición de $\mathbf{D}$ sabemos que $d_{ii} = \frac{1}{c_i} = \frac{1}{\sum_{k=1}^{n} w_{ki}}$, el cual haciendo un cambio de variables, nos queda $\frac{1}{c_j} = \frac{1}{\sum_{i=1}^{n} w_{ij}}$

$$= p\frac{1}{\sum_{i=1}^{n} w_{ij}}(\sum_{i=1}^{n} w_{ij} - w_{ii})$$

Ahora como $\mathbf{W}$ tiene ceros en la diagonal, $w_{ii} = 0$

$$= p\frac{1}{\sum_{i=1}^{n} w_{ij}}\sum_{i=1}^{n} w_{ij}$$

$$= p$$

Volviendo a la desigualdad que queremos probar tenemos:

$$|a_{ii}| = 1 > p$$

Ahora eso sabemos que siempre vale pues $p \in (0, 1)$.

Si $c_i = 0$ luego por definición $d_{ii} = 0$ y la suma nos queda:

$$\sum_{i=1,i\neq j}^{n} | - pw_{ij}d_{ii}| = 0$$

Entonces la desigualdad queda:

$$|a_{ii}| = 1 > 0$$

Finalmente podemos concluir que $\mathbf{A}$ es **edd** y por ende tiene eliminación gaussiana sin pivoteo.

Para poder calcular el *PageRank* debemos triangular la matriz A y resolver el sistema de ecuaciones:

$$\mathbf{A}x = \gamma e$$

donde $\gamma = z^t x$ es un factor de escala que valdra 1 y finalmente debemos normalizar el vector $x$. Gracias a los resultados anteriores, sabemos que podemos triangular A sin necesidad de realizar pivoteo.

## 2.2. Implementación

Para calcular los ranking realizamos una implementación en C++ que resuelve el sistema de ecuaciones ya mencionado. Además debido a que las matrices que definimos son ralas, es decir que la mayoría de sus elementos son ceros, no tiene sentido usar la representación clásica de `vector<vector<double> >`. Para poder hacer un uso eficiente de memoria y tiempo decidimos representar a las matrices con estructuras que solo mantengan los elementos no nulos de la matriz.

### 2.2.1. DOK (Dictionary of Keys)

La primer estructura que consideramos fue **DOK**. Esta consiste en tener un `diccionario(fila, diccionario (columna, valor))`. Para implementar esto en `C++` usamos un `map<map<int, double> >`. Esto internamente se implementa usando Arboles Binarios de Búsqueda.

Estos nos permiten iterar en orden by key de forma eficiente y acceder a un elemento en $\mathcal{O}(logn)$. Esta estructura nos permite solo mantener en memoria los elementos no nulos

Para su implementación decidimos crear una nueva clase la cual llamamos `MatrizRala`. En esta clase, definimos variables privadas las cuales nos dan la dimensión de la matriz, una variable que me devuelve la matriz entera y una última llamada `épsilon` que usamos en las comparaciones para no guardar elementos que sean casi 0.

Esta clase posee una serie de funciones, entre ellas las que nos devuelven estas variables privadas, y otras como `dameValor`, que dados dos índices $i < n$ y $j < m$, nos devuelve el valor del elemento $w_{ij}$, y `m_real` que nos devuelve la cantidad de elementos no nulos de una fila $i < n$. Luego tenemos `asignarValor` y `agregarColumna` que nos permiten agregar un valor en una posición válida de la matriz rala y una columna al final de todo respectivamente. Por último, contamos con `print_matriz` que nos imprime por pantalla nuestra matriz de la siguiente manera: Dada una matriz W

$$W = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Entonces el *output* por pantalla será:

```
0: 0:1 3:1
1: 2:1
3: 1:1
```

Además de estas funciones dentro de la clase, existen funciones creadas con el objetivo de realizar operaciones sobre estas matrices. Entre estas funciones se encuentran:

- `multiplicarRalas`: Nos permite multiplicar dos matrices siempre que den las dimensiones.
- `multiplicarEscalar`: Nos permite multiplicar una matriz con un $\lambda \in \mathbb{R}$.
- `restarRalas`.
- `elimGauss` (Eliminación Gaussiana): Permite triangular una matriz siempre y cuando no necesite pivoteo.
- `backwardSust` (Backward Sustitution): Permite resolver un sistema de ecuaciones dada una matriz triangulada superiormente.

Esta representación nos resulto bastante intuitiva y fácil de implementar. Pero no encontramos una forma sencilla de iterar las columnas sin tener que tener una estructura simétrica. Esto nos duplica la cantidad de memoria a utilizar. Y iterar por todos los valores posibles lleva demasiado tiempo para los test más grandes. Por eso decidimos cambiar la representación.

## 2.3. CSR (Compressed Spare Rows)

La segunda estructura que consideramos fue **CSR**. Esta estructura representa a la matriz usando tres vectores de la siguiente manera:

- `A`: Es un vector cuyo contenido son los elementos no nulos de la matriz.
- `JA`: Vector de enteros donde el i-ésimo elemento corresponde al índice de la columna del i-esimo valor en A.
- `IA`: El intervalo formado por los valores IA[i] y IA[i+1] representa los índices de los vectores A y JA que pertenecen a la i-ésima fila.

Es decir, dada $B \in \mathbb{R}^{n \times n}$:

$$B = \begin{pmatrix} 1 & 0 & 0 & 7 \\ 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 9 \\ 0 & 3 & 0 & 0 \end{pmatrix}$$

La representamos con los siguientes vectores:

$$\begin{gathered} \text{A} = [1, 7, 2, 3, 9, 3] \\ \text{JA}= [0, 3, 1, 2, 3, 1] \\ \text{IA}= [0, 2, 4, 5, 6] \end{gathered}$$

Para facilitar las funciones mencionadas anteriormente realizamos los siguientes auxiliares:

- `dameValor`: Dados un i,j devuelve el valor correspondiente a ese valor en la matriz. Para esto iteramos dentro de los índices correspondientes a la fila i y para cada uno de ellos buscamos el elemento cuya columna sea j.
- `dameFila`: Dada un i devuelve la i-ésima fila de la matriz como tuplas `(columna, valor)`. Para su implementación usamos los valores del vector `IA` que se encuentran en lo s índices $i$ e $i+1$. Luego, iteramos sobre los elementos de `A` y los de `JA` que se encuentran entre esos índices tomados de `IA` y los retornamos como la tupla `(columna, valor)`.
- `dameColumna`: Dada una j devuelve la j-esima columna de la matriz como tuplas `(fila, valor)`. Para ello, iteramos dentro de cada intervalo de índices correspondiente a una fila i con it, y para cada uno de sus elementos verificamos el valor de su columna en `JA[it]`, si este corresponde a la columna j, agregamos el elemento como la tupla `(i, A[IA[it]])`

Esta estructura nos permite iterar en orden los elementos no nulos las filas y las columnas de forma sencilla y eficiente.

### 2.4. Elección de estructura

Tras implementar ambas estructuras optamos por utilizar CSR, ya que resulto mucho más rápida corriendo los test grandes. Ahora, es necesario considerar que la implementación de CSR fue mucho más compleja. Esto lo analizamos en detalle en la sección de experimentos 3.1.3

## 3. Experimentación

Se realizará una evaluación de los aspectos cuantitativos y cualitativos relacionados a los métodos de asignación de *pageRank* .

Para la evaluación cuantitativa haremos un análisis a partir del error absoluto de los test de ranking de Page y con la aproximación de la solución $x'$ obtenida: $|Ax' - x'| \approx 0$.

Para la evaluación cualitativa del algoritmo decidimos analizar distintas estructuras de grafos para poder entender como influye la probabilidad del navegante aleatorio ($p$) en el PageRank. Además exploramos grafos inspirados en un ejemplo real de la web.

## 3.1. Análisis cuantitativo

### 3.1.1. Error Absoluto respecto a los test de la cátedra

En este experimento queremos ver cual es el error absoluto de los rankings que calcula nuestra implementación versus los rankings provistos por la cátedra.

Suponemos que el error absoluto va ser del orden de $10^{-5}$ pues ese es el épsilon que usamos para comparar dos valores dentro de la clase MatrizRala.

Para medir el error absoluto implementamos una función que nos permite chequear si los puntajes calculados para cada test son 'iguales' con un margen de error de $10^{-4}$. Luego esa función calcula:

$$ErrorAbsoluto(test_j) = \frac{\sum_{i=1}^{n} |p_i - \tilde{p_i}|}{n}$$

Con $p_i$ el valor del puntaje de la página i obtenidos luego de correr el test_j y $\tilde{p_i}$ el valor del puntaje de la página i según los resultados de la cátedra.

**Resultados**

Al realizar el cálculo de error absoluto obtuvimos los siguientes resultados para cada test de la cátedra:

| Test | Error Absoluto |
|---|---|
| 15 segundos | 0.00000183 |
| 30 segundos | 0.00000171 |
| Aleatorio desordenado | 0.00000012 |
| Aleatorio | 0.00002000 |
| Completo | 0.00000000 |
| Sin links | 0.00000000 |
| Trivial | 0.00000000 |

La varianza para cada uno de los test nos da 0 ya que al calcular el error absoluto varias veces para cada test nos termina dando siempre el mismo valor. Esto quiere decir que el resultado que obtuvimos realizando el cálculo está bien definido.

### 3.1.2. Aproximación del error

Queremos ver si los puntajes obtenidos son una buena solución aproximada del sistema. Es decir queremos ver que $|Ax' - x'| \approx 0$ con $x'$ la solución obtenida por nuestra implementación.

Creemos que los puntajes obtenidos va a ser una buena solución al sistema, ya que solo redondeamos los valores cercanos a cero con un épsilon de $10^{-5}$. Para medir esto calculamos $\mathbf{A} = p\mathbf{W}\ \mathbf{D} + ez^t$ y luego simplemente realizamos la cuenta y chequeamos que el vector resultante sea aproximadamente nulo con un épsilon de $10^{-5}$.

**Resultados**

Tras realizar el experimento para los tests dados por la cátedra, obtuvimos que la solución a $|Ax' - x'|$ fue 0 con una precisión de $10^{-5}$ en todos los casos, por lo que, el resultado fue exitoso. Como todas las mediciones nos dieron 0 para una precisión de $10^{-5}$ luego la varianza es 0.

### 3.1.3. Comparación de los tiempos de ejecución representando las matrices ralas con DOKs vs CSR

En este experimento medimos cuánto tardan los test de la cátedra y otros que generamos nosotros usando las distintas estructuras de representación de matrices ralas que implementamos. Todos los tests que agregamos nosotros menos `completo25x25.txt` son los que utilizamos en la sección 3.2 para otros experimentos, allí se detalla cómo fueron generados. El test `completo25x25.txt` es un grafo completo de 25 nodos.

Nuestra hipótesis es que para los test pequeños y con matrices que no sean ralas la diferencia no será significativa. Pero para el resto de los casos CSR será mucho más eficiente.

| Test | # de Páginas | # de links | DOK (seg) | CSR (seg) |
|---|---|---|---|---|
| 15 segundos | 2000 | 12000 | 460.53332351 | 11332.67122160 |
| Aleatorio desordenado | 5 | 12 | 0.00063999 | 0.00066232 |
| Aleatorio | 5 | 12 | 0.00052481 | 0.00063373 |
| Completo | 5 | 20 | 0.00072163 | 0.00086125 |
| Sin links | 5 | 0 | 0.00031827 | 0.00030714 |
| Trivial | 1 | 0 | 0.00025630 | 0.00019251 |
| Antisupernodo 100 | 100 | 99 | 0.02314249 | 0.13611629 |
| Supernodo 100 | 100 | 99 | 0.01683984 | 0.11263090 |
| Wikipedia 10 | 10 | 15 | 0.00453100 | 0.02168011 |
| Wikipedia 50 | 50 | 99 | 0.00510254 | 0.02203562 |
| Completo 25 | 25 | 625 | 0.00583659 | 0.04206059 |

Efectivamente se confirmó lo que creíamos. Para las matrices ralas corre mucho más rápido la implementación CSR. Y se puede ver que para los test grandes como `test_15_segundos.txt` la diferencia en los tiempos de ejecución de ambas estructuras es muy grande. Esto ocurre porque nuestra implementación de DOKS tiene complejidad de $\mathcal{O}(n^3)$ para la multiplicación de matrices y eliminación gaussiana, mientras que la implementación de CSR tiene complejidad de $\mathcal{O}(\tilde{n}^3)$ donde $\tilde{n}$ es la cantidad de elementos no nulos de la matriz.

Vamos también los valores de la varianza para este experimento:

| Test | # de Páginas | # de links | DOK (Var) | CSR (Var) |
|---|---|---|---|---|
| 15 segundos | 2000 | 12000 | $3{,}5139377025 \times 10^{-8}$ | 0,004828058913535843 |
| Aleatorio desordenado | 5 | 12 | $2{,}177751874 \times 10^{-10}$ | $6{,}564561060 \times 10^{-10}$ |
| Aleatorio | 5 | 12 | $1{,}513178885311 \times 10^{-10}$ | $1{,}74765501775 \times 10^{-10}$ |
| Completo | 5 | 20 | $3{,}420250044975 \times 10^{-10}$ | $1{,}445502328 \times 10^{-10}$ |
| Sin links | 5 | 0 | $6{,}9090275731 \times 10^{-11}$ | $1{,}204474203263 \times 10^{-10}$ |
| Trivial | 1 | 0 | $4{,}88849262839 \times 10^{-11}$ | $1{,}910062760158 \times 10^{-10}$ |
| Antisupernodo 100 | 100 | 99 | $1{,}517480255094 \times 10^{-05}$ | $2{,}01521483889 \times 10^{-06}$ |
| Supernodo 100 | 100 | 99 | $7{,}857205106407 \times 10^{-05}$ | $1{,}91621776675 \times 10^{-06}$ |
| Wikipedia 10 | 10 | 15 | $3{,}6078730791 \times 10^{-08}$ | $1{,}593330136316 \times 10^{-08}$ |
| Wikipedia 50 | 50 | 99 | $6{,}029400367437 \times 10^{-06}$ | $2{,}483206673564 \times 10^{-07}$ |
| Completo 25 | 25 | 625 | $1{,}34299209298 \times 10^{-05}$ | $4{,}200983222052 \times 10^{-07}$ |

## 3.2. Análisis cualitativo

**Generación de casos de test** Para generar las matrices para cada uno de los experimentos usamos un script de Python. El código utilizado se encuentra en el Notebook entregado. Allí se explica en detalle la generación de cada test, de todas formas todas las topologías de los casos de test están explicadas en cada sección.

### 3.2.1. Experimento 1: Análisis de casos extremos Supernodo vs Antisupernodo

En este experimento nos proponemos analizar como se comporta el *PageRank* en 2 tipos de redes de páginas particulares. Una es una red que tiene un supernodo, es decir, tiene una página que es linkeada por todas las demás. En contraste, la otra red tiene una página que cita a todo el resto de las páginas pero no es citada. El resto de las páginas no tienen links a otros nodos. Si bien estas redes no son lo que uno esperaría encontrar en la web su análisis nos permite entender mejor como se asigna el *pageRank* cuando tenemos muchos links de páginas "malas". Esto nos interesa para ver si podemos encontrar maneras de tratar de 'romper' el *pageRank*. Es decir imaginemos que tenemos una página que no tiene un buen ranking y queremos ver si podemos crear páginas nuevas que la linkeen de alguna forma específica para poder mejorar su ranking. Esto último lo exploramos en los Experimentos de la sección 3.2.2.

**Supernodo: Página muy citada**

Nos proponemos analizar la fluctuación del *pageRank* de una página que es citada por todas las demás a medida que variamos la probabilidad $p$ del navegante aleatorio. Además el resto de las páginas no tienen links entrantes.

Esto lo hacemos con el objetivo de ver en cuánto influye la probabilidad del navegante aleatorio en el *pageRank* del supernodo.

Nuestra hipótesis es que a mayor $p$, el supernodo tendrá un mejor ranking debido a que si hay más probabilidad de que el usuario navegue a los links de la página actual luego va a terminar con mayor probabilidad en el supernodo pues todas las páginas lo linkean.

Detalles del experimento: Generamos una matriz de $100 \times 100$ donde la fila 0 representa al supernodo. El resto de las filas solo tienen 0s.

**Resultados**

Primero veamos cómo varía el ranking del supernodo a medida que incrementamos la probabilidad del navegante aleatorio:

En la figura 1 podemos ver como el *pageRank* del supernodo va creciendo a medida de que aumenta $p$. Esto confirma lo que suponiamos pero ver el valor del *pageRank* no nos da información de como se posiciona el supernodo con respecto a las demás páginas. Y dado a que eso es lo que nos interesa decidimos ver como varia su posición en el ranking de todas las páginas. Es decir, el índice del supernodo en el vector de puntajes ordenado en orden decreciente para un $p$ arbitrario. Donde la posición 1 es la mejor. Esto lo vemos en la figura 2.

Como podemos apreciar en este gráfico, la posición en el ranking de nuestro supernodo se mantiene constante a medida que $p$ aumenta. Además el supernodo se mantiene en la posición 1 (que es la mejor posición posible). Esto corresponde con lo que pensamos que iba a suceder.

Otra cosa que nos preguntamos es si al resto de las páginas que no son el supernodo se les asigna el mismo *pageRank* o no. Nuestra hipótesis es que si. Para ver esto comparamos los *pageRank* de los

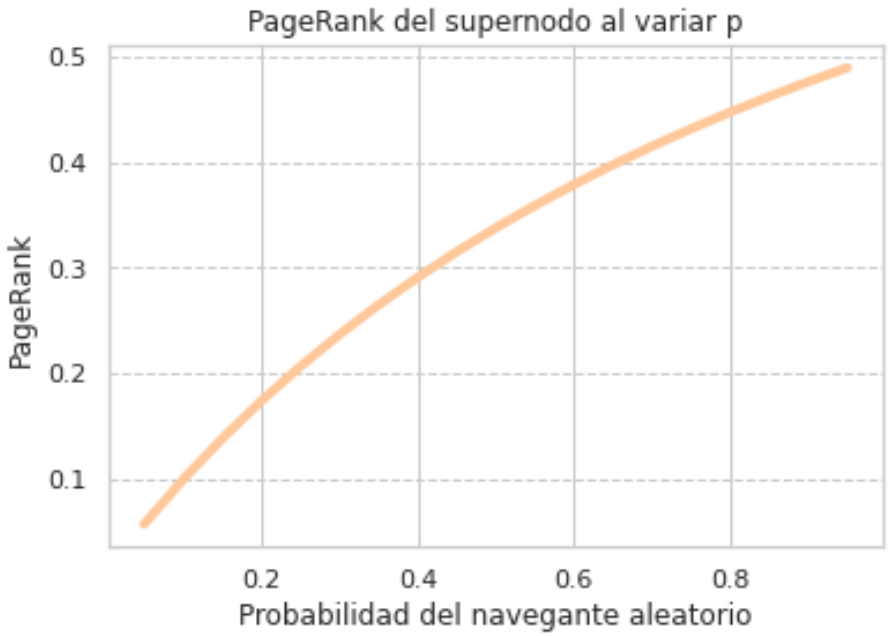


Figura 1: Grafico del pageRank del supernodo dependiendo de $p$ en una red de 100 nodos


Figura 2: Gráfico de la posición del supernodo en el ranking de todas las páginas

nodos entre si para cada $p$. Pudimos ver que efectivamente todos los nodos que no son el supernodo reciben igual *pageRank*. En la figura 3 graficamos los *pageRank* de todos los nodos para un $p$ arbitrario (ya que la situación es igual para todos los $p$). En este gráfico el área de cada punto coincide con el *pageRank* de la página y se encuentran separado el supernodo del resto de los nodos.

**Antisupernodo: Página muy citadora**

De igual forma que en el experimento anterior vamos analizar el *pageRank* a medida que varia $p$ pero ahora del antisupernodo. Un antisupernodo es una página que cita a todas las demás pero que no es citada por ninguna. El resto de las páginas no tiene links salientes. Esto lo haremos con

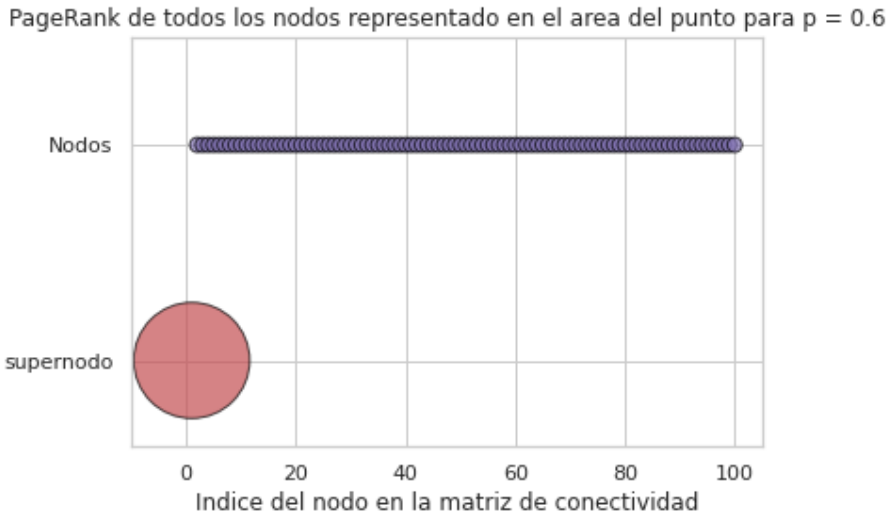


Figura 3: Gráfico del *pageRank* de todos los nodos. El área de cada punto es su *pageRank*

el objetivo de analizar una estructura totalmente opuesta a la anterior y ver qué tan distinto se comporta en comparación.

Nuestra hipótesis es que el ranking va a disminuir en proporción directa a $p$. Pues debería pasar lo contrario que en el experimento anterior.

**Resultados**

Al calcular el *pageRank* del antisupernodo para cada $p$ confirmamos nuestra hipótesis. Esto lo podemos ver en la figura 4.

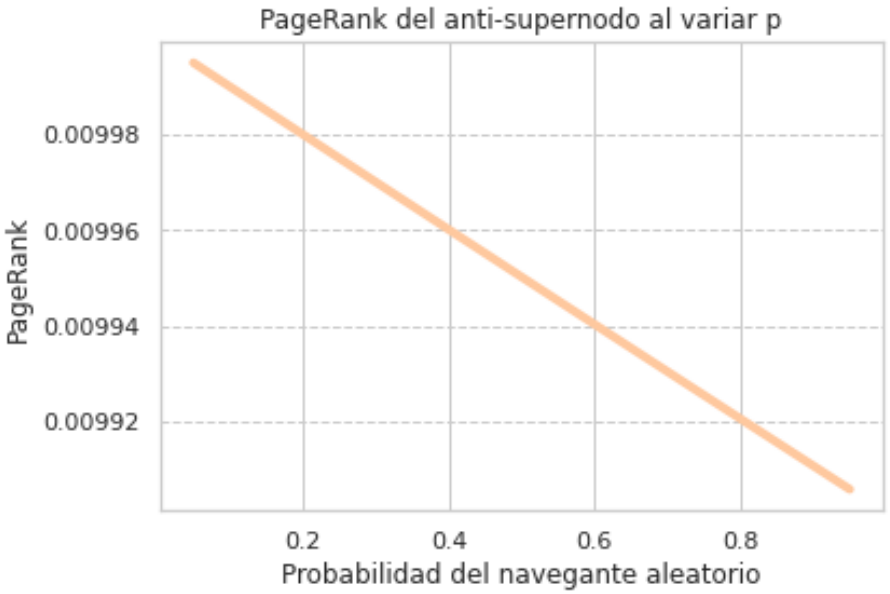


Figura 4: Gráfico del *pageRank* del antisupernodo variando $p$ en una red de 100 nodos.

Al igual que en experimento anterior nos interesa ver qué pasa con la posición en el ranking del antisupernodo. Esto lo vemos en la figura 5.


Figura 5: Gráfico de la posición en el ranking del antisupernodo variando $p$

Observamos que el antisupernodo se mantiene en la última posición del ranking para todo $p$. Lo cual tiene mucho sentido pq el ranking de los demás nodos aumenta porque el antisupernodo los linkea.

**PageRank del antisupernodo a medida que le agregamos ejes**

En este experimento decidimos jugar con la cantidad de ejes que tiene el antisupernodo y ver cómo se comporta su *pageRank* para un $p$ arbitrario. Tomamos un antisupernodo vacio, es decir, sin links y fuimos agregando uno a uno links desde el supernodo a las demás páginas.

Nuestra hipotesis es que el ranking del antisupernodo va a empeorar a medida de que agregamos ejes que apunten a los demás nodos. Y su posición en el ranking también sera peor.

**Resultados**

Primero miramos como varia el *pageRank* del antisupernodo a medida de que agregamos ejes. Esto lo podemos ver en la figura 6. Nos llamo la atención de que el *pageRank* se mantiene constante en un mismo valor. Llegamos a la conclusión de que el *pageRank* de los demás nodos es lo que se redistribuye a medida de que aumentan los links.

Luego miramos como se comporta la posición del ranking del antisupernodo. Esto lo mostramos en la figura 7. Como esperabamos el ranking del antisupernodo empeora a medida de que agregamos links salientes.

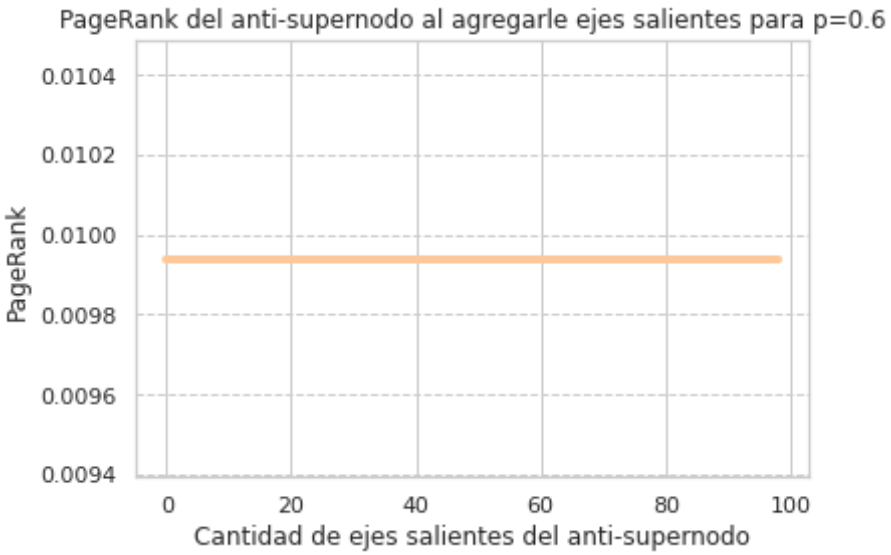


Figura 6: Gráfico de la posición en el ranking del antisupernodo variando $p$

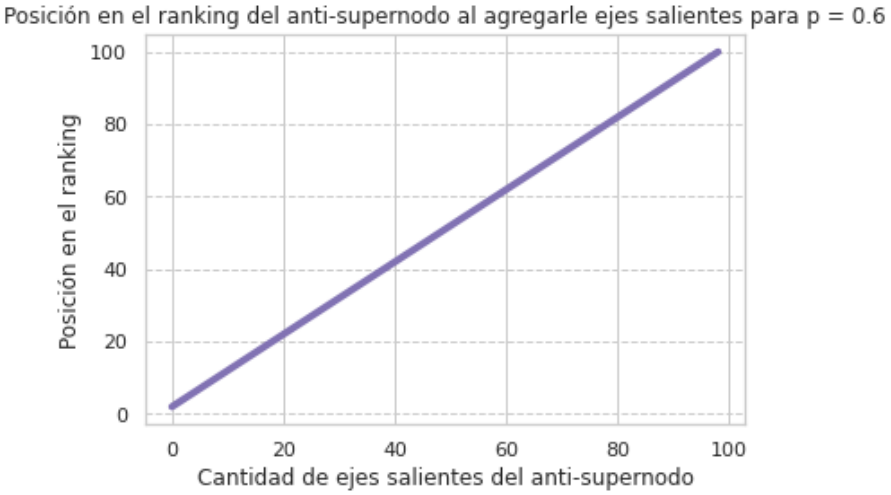


Figura 7: Gráfico de la posición en el ranking del antisupernodo variando $p$

### 3.2.2. Experimento 2: Caso Wikipedia

En este experimento nos interesa analizar que pasa con el *pageRank* en una red de páginas inspirada en un caso real. Para esto vamos a simular una red de links que contiene a una entrada de Wikipedia.

Supongamos que buscamos en nuestro navegador favorito `PageRank`. Nos van a aparecer muchas páginas entre ellas la entrada de Wikipedia de PageRank 5. También algunos papers que Wikipedia cita como fuente. Finalmente van a aparecer otras páginas como tutoriales y blogs. Para simplificar el modelo supongamos que todas estas páginas citan a Wikipedia y que no se citan entre si. Luego podemos dividir este grupo en: Las páginas que citan a papers y a Wikipedia y las páginas que solo citan a Wikipedia. Vamos a hacer que las páginas que citan papers citen a todos los papers. La estructura general de esta red la podemos ver en la Figura 8.

**Cambios en el PageRank de la entrada de Wikipedia a medida de que cambia la probabilidad del navegante aleatorio**

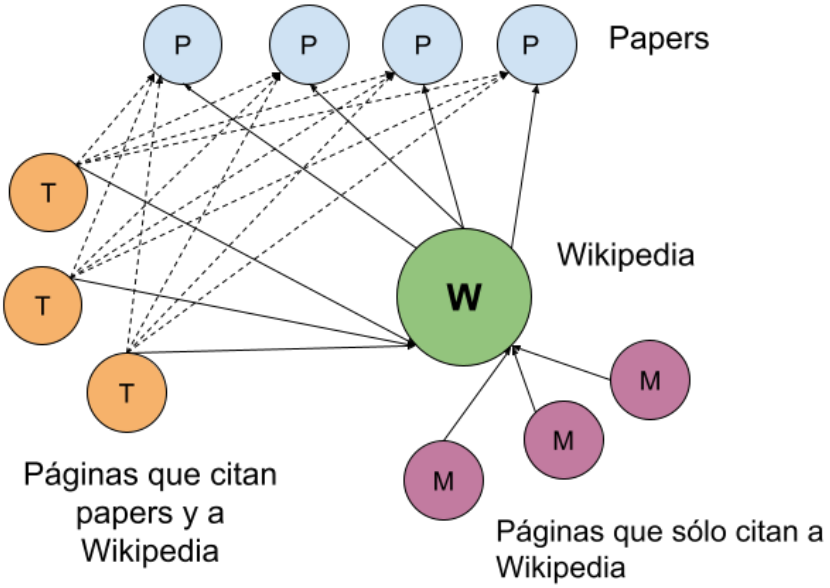


Figura 8: Estructura general de una red de links que contiene una página de Wikipedia

Para este experimento generamos una red con la estructura que ya describimos y que se representa gráficamente en la figura 8. La red generada para este experimento particular tiene 50 nodos de los cuales 5 son papers, 10 son páginas que citan a Wikipedia y a los papers y 34 son páginas que citan sólo a Wikipedia.

Queremos analizar como cambia el *PageRank* de la entrada de Wikipedia a medida de que varia la probabilidad del navegante aleatorio y como ese valor la posiciona en el ranking final.

Nuestra hipótesis es que el *pageRank* va a aumentar a medida de que aumenta la probabilidad del navegante aleatorio pues cada vez hay más chances de que la siguiente página a elegir sea la entrada de Wikipedia pues tiene muchos links. Además Wikipedia se va a ubicar en la mejor posición del raking.

**Resultados**

Para este experimentos tomamos todos los valores de $p \in [0{,}05, 1)$ con saltos de 0.05. Para estos valores de $p$ graficamos el *pageRank* de Wikipedia en la figura 9. En este gráfico se puede observar que el ranking de la página va mejorando mientras el $p$ aumenta.

Además nos intereso ver como queda posicionada la entrada de Wikipedia dado su *pagerank* para cada $p$. Es decir el índice en el vector de puntajes ordenado de forma decreciente. La posición 1 es la mejor. Graficamos esto en la figura 10

Como preveíamos Wikipedia se encuentra siempre en la primer posición del ranking. Es decir que será el primer resultado que aparezca en la búsqueda lo que se corresponde con lo que queremos que pase si el sistema de *pageRank* rankea bien a las páginas. Pero en este modelo tenemos otros nodos como los papers que van a tener un buen ranking. Pero, ¿Cómo se compara su con el de Wikipedia? Para ver esto armamos un gráfico de puntos donde representamos a todos los nodos separados por si son Wikipedia, papers, páginas que citan papers o páginas que solo citan a Wikipedia. Luego el área del punto nos indica el *pageRank* de cada nodo. Este gráfico corresponde con la figura 11


Figura 9: Gráfico del *pageRank* de Wikipedia variando $p$


Figura 10: Gráfico de la posición en el ranking de Wikipedia variando $p$

Como esperabamos los papers también tienen un muy buen *pageRank* y en particular vale lo mismo para todos. Esto pasa porque todos los papers tienen los mismos links salientes y entrantes. Además el *pageRank* de Wikipedia en bastante superior a el de todos los demás nodos. Algo que nos

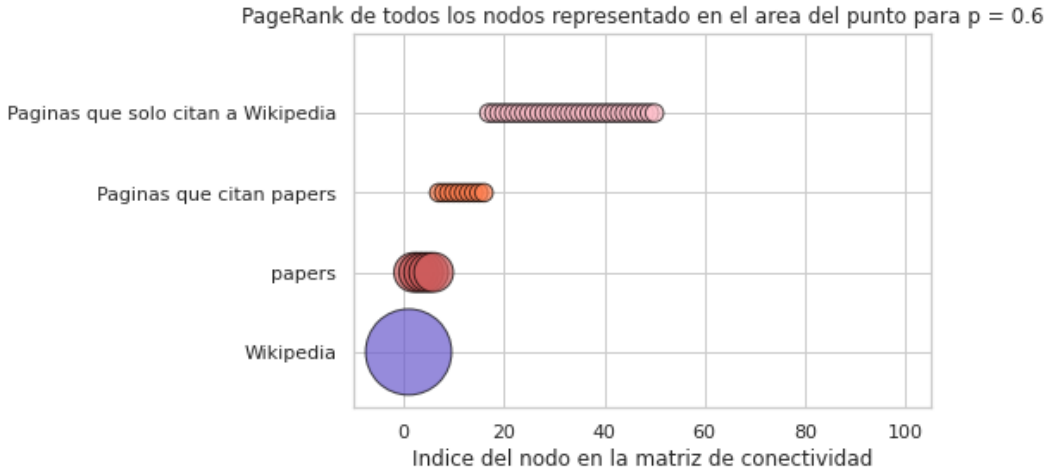


Figura 11: Gráfico del *PageRank* de todos los nodos en experimento Wikipedia. El área de cada punto es su *pageRank*

llamo la atención es que el *pageRank* de las páginas que solo citan a Wikipedia y las que citan a Wikipedia y a todos los papers es el mismo. Esto nos pareció interesante ya que esperábamos que las páginas que citan a los papers tuvieran mejor ranking. Pero analizándolo más en detalle nos dimos cuenta de que tiene sentido lo que se ve en el gráfico ya que los links influyen mucho el *pageRank* de nodo al que apuntan y no en el del nodo del que sale el link. Esto es clave para que el modelo sea robusto pues sino podríamos crear una página ficticia hacerla que apunte a muchas páginas buenas y así mejorar su ranking.

**Cambios en el PageRank de un paper a medida de que cambia la probabilidad del navegante aleatorio**

Para este experimento realizamos un análisis de los valores del PageRank y de su posición el el ranking de uno de los papers que es linkeado por Wikipedia. Usamos la misma red en el caso anterior 3.2.2 . Tomamos un paper cualquiera ya que todos tienen los mismos links por lo que su *pageRank* es igual.

Nuestra hipótesis sobre este experimento es que el *PageRank* va a aumentar a medida de que la probabilidad del navegante aleatorio aumente. Pues como los papers son citados por Wikipedia entonces a medida de que aumenta la probabilidad del que navegante siga un link de la página actual este va a terminar con mayor probabilidad en los papers. Además creemos que los papers se van a mantener en las primeras 6 posiciones del ranking pero sin sobrepasar a Wikipedia.

**Resultados**

Primero miramos como varia el *pageRank* de un paper cualquiera. Se confirma lo que pensabamos pues el *pageRank* del paper aumenta a medida de que aumenta $p$. Esto lo vemos en la figura 12.

Luego nos intereso ver en qué posición del ranking queda este paper. Esto está graficado en la figura 13. El paper se mantiene en la posición 6 del ranking. Y esto pasa para todos los papers porque como el *pageRank* de todos es el mismo nuestro método de calcular la posición en el ranking te da la peor posición. Esto es algo particular de las funciones de Python que usamos para calcular la posición en el ranking.

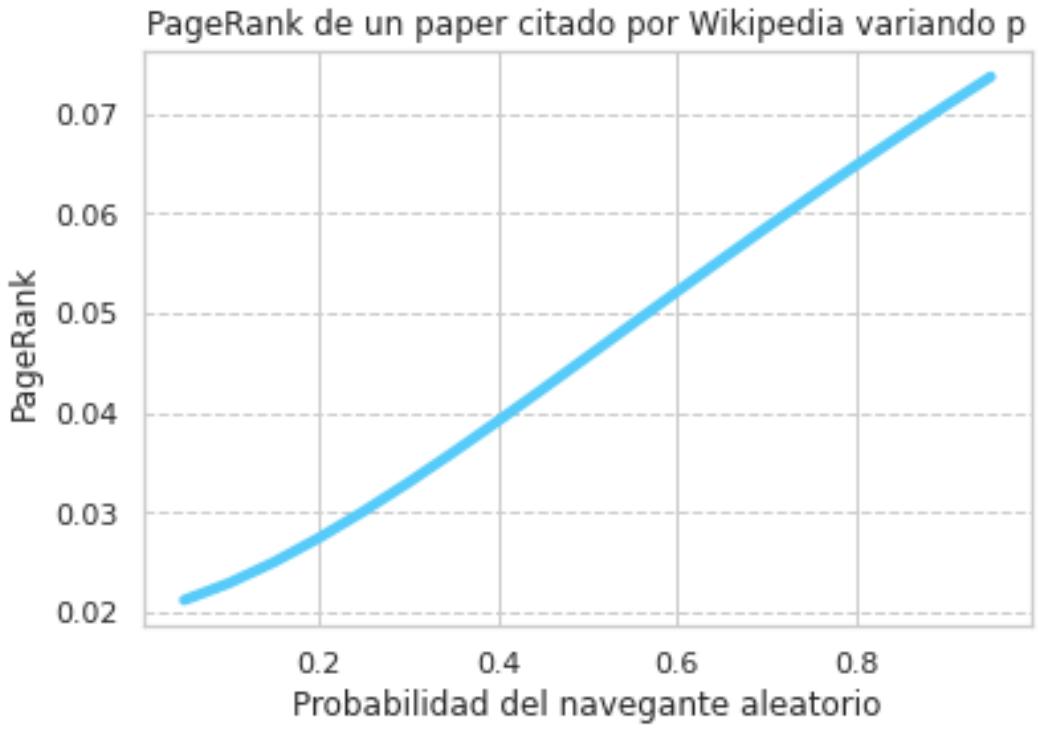


Figura 12: Gráfico del *PageRank* de un paper variando $p$


Figura 13: Gráfico de la posición de un paper variando $p$

**Cambios en el *pageRank* de Wikipedia si variamos la cantidad de papers para un $p$ arbitrario**

En este experimentos decidimos investigar cómo se comporta el *pageRank* de Wikipedia si cam-

biamos la estructura de la red agregando papers. Nuestro objetivo es poder encontrar alguna configuración de la red donde Wikipedia no tenga el mejor ranking. Generamos redes con la estructura de la figura 8 de 50 nodos. Variamos la cantidad de papers entre 5 y 35. Dejamos fija la cantidad de páginas que citan papers y a Wikipedia en 10. El resto de los nodos son los que solo citan a Wikipedia.

Nuestra hipótesis es que al sumar cantidad de papers Wikipedia va a perder puntaje en la red pues su *pageRank* se va a distribuir hacia los papers. Además al aumentar la cantidad de papers le estamos sacando links entrantes a Wikipedia porque dejamos fija la cantidad total de nodos y la cantidad de páginas que citan papers y a Wikipedia.

**Resultados**

Calculamos el *pageRank* de Wikipedia para un $p$ arbitrario al variar la cantidad de papers. Esto se muestra en la figura 14. Como esperábamos el *pageRank* de Wikipedia disminuye pues se distribuye hacia los papers. Ahora, ¿Esto hace que Wikipedia pierda su posición en el ranking total? No. Esto lo vemos en la figura 15. Wikipedia mantiene la mejor posición en el ranking. Esto tiene que ver con que Wikipedia sigue siendo la página con mayor cantidad de links entrantes.

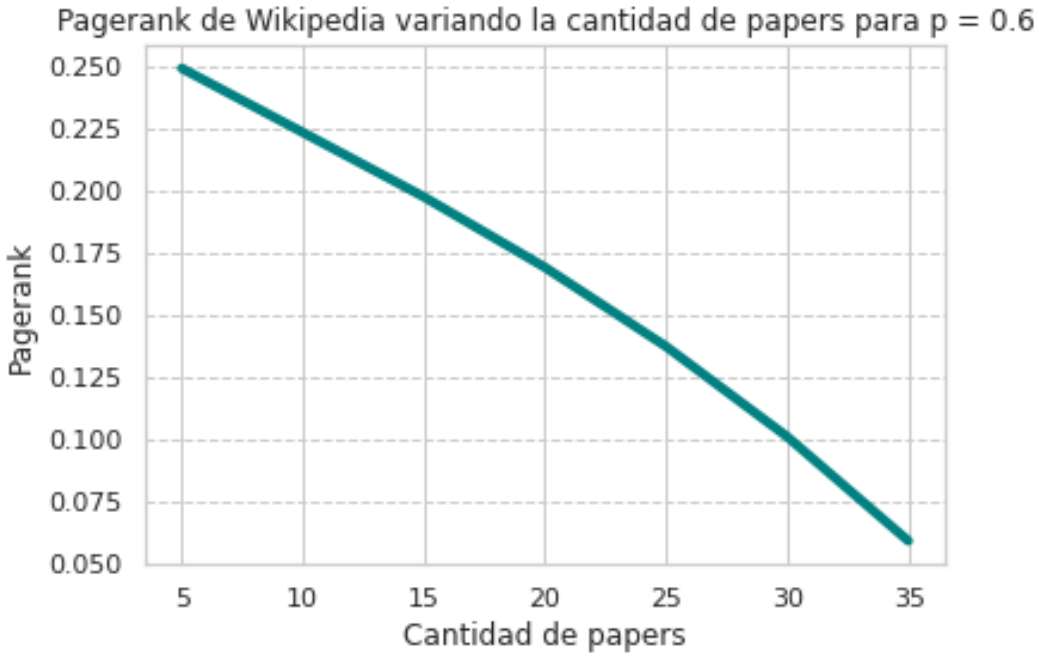


Figura 14: Gráfico del PageRank de Wikipedia variando la cantidad de papers


Figura 15: Gráfico de la posición del ranking de Wikipedia variando la cantidad de papers

**Cambios en el *pageRank* de Wikipedia si variamos la cantidad de páginas que citan papers para un *p* arbitrario**

Ahora vamos a modificar la red pero variando la cantidad de páginas que citan papers. Generamos redes con la estructura de la figura 8 de 50 nodos. Variamos la cantidad de páginas que citan papers entre 5 y 35. Dejamos fija la cantidad papers en 5. El resto de los nodos son los que solo citan a Wikipedia.

Nuestra hipótesis es que Wikipedia va a perder su posición en el ranking y que los papers va a quedarse con los primeros puestos. Además el *pageRank* de Wikipedia va a caer pues va a distribuirse hacia los papers.

**Resultados**

Primero miramos qué le pasa al *pageRank* al variar la cantidad de páginas que citan papers y a Wikipedia. Esto lo mostramos en la figura 16. Como suponiamos el *pageRank* de Wikipedia disminuye. Ahora la prueba de fuego es ver que pasa con su posición en el ranking. Esto lo graficamos en la figura 17. Finalmente en contra de lo que suponíamos Wikipedia mantiene su posición en el ranking. Esto pasa porque Wikipedia gana la misma cantidad de links que los papers entonces la relación entre los *pageRank* se mantiene.

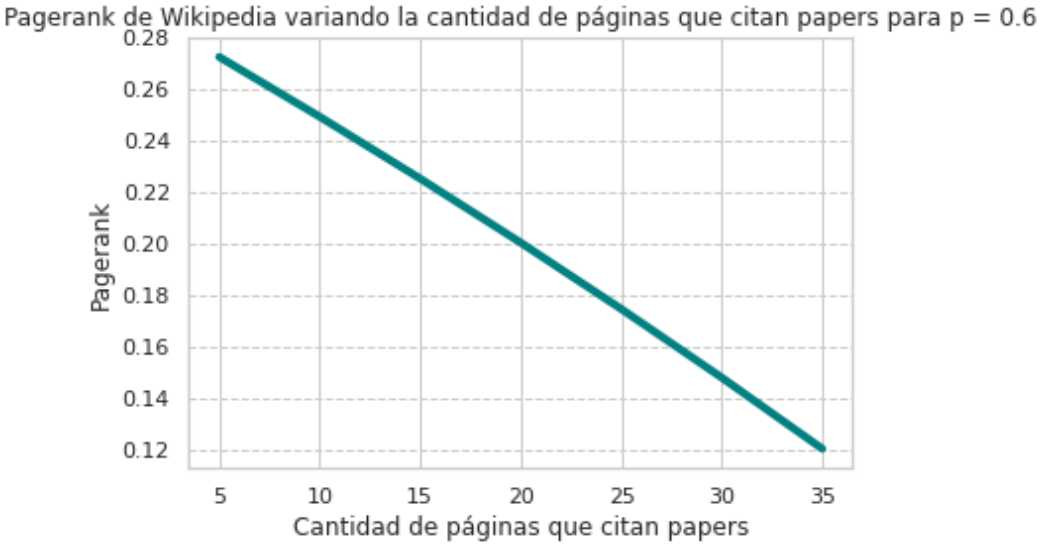


Figura 16: Gráfico del PageRank de Wikipedia variando la cantidad de paginas que citan a los papers y a Wikipedia

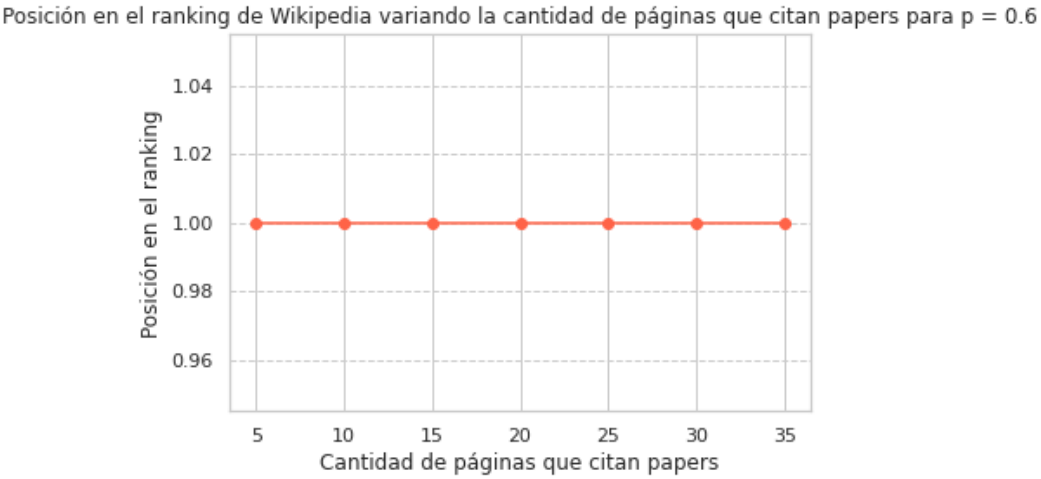


Figura 17: Grafico de la posición del ranking de Wikipedia variando la cantidad de paginas que citan a los papers

**PageRank de Wikipedia si agregamos un supernodo a la red**

En este experimento queremos ver si podemos romper el *pageRank* de Wikipedia. Supongamos que tenemos una página web que se llama PageRank donde vendemos teclados. Queremos aumentar el *pageRank* de nuestra página PageRankTeclados agregando páginas que la linkeen.

A la red de Wikipedia descripta en le agregamos $m$ páginas que forman un grafo supernodo como los de la sección 3.2.1. Es decir que la nueva red va a tener $50+m$ nodos. Y queremos ver cuál es el m talque Wikipedia deja de estar en el primer puesto del ranking.

Nuestra hipótesis es que para ganarle a Wikipedia el $m$ debe estar entre 40 y 50. Pues el supernodo debe tener la misma o más cantidad de links entrantes que Wikipedia para ganarle.

**Resultados** Probamos agregando grafos supernodos de distintos tamaños a la red. Descubrimos que al agregar exactamente 37 nodos el supernodo se ubica en la primer posición del ranking y desplaza a Wikipedia al segundo lugar. Pero para esos valores la diferencia en el *pageRank* de ambos es muy pequeña. Esto lo podemos ver en la figura 18.

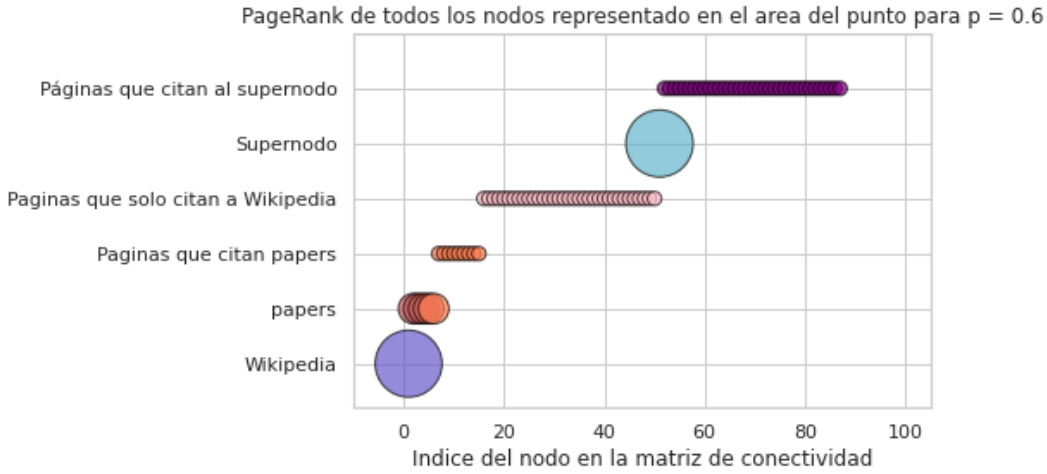


Figura 18: Gráfico del *pageRank* representado en el área del punto de todos los nodos de la Red Wikipedia+Supernodo

Lo que vimos en este experimento es muy interesante pues Wikipedia tiene en total 44 links entrantes y el Supernodo tiene 36. Aún así el Supernodo le gana a Wikipedia en el ranking. De todas formas el modelo de *pageRank* se mantiene robusto en el mundo real. Porque en la web Wikipedia va a tener millones de links entrantes entonces tener que agregar una cantidad parecida de links para mejorar el *pageRank* de nuestra página va a tener un costo demasiado grande. Por otro lado, sería muy interesante analizar cuál es la razón por la que pasa esto.

# 4. Conclusión

A partir de los experimentos y los resultados obtenidos conseguimos comprender mejor algunos conceptos tanto teóricos como prácticos de la materia así como también entender el algoritmo PageRank. Además pudimos experimentar con distintas estructuras de representación de matrices ralas. Llegamos a las siguientes conclusiones:

- Nuestra implementación de matrices ralas con DOK funciona rápidamente para casos donde las matrices son pequeñas, pero al utilizar el mismo algoritmo para matrices de gran tamaño, este se vuelve demasiado lento porque no itera sobre la matriz de manera eficiente. En cambio la implementación que usa CSR es mucho mejor aunque hay muchas mejoras que pueden acelerar su ejecución. Si bien las funciones dameFila y dameColumna hacen muy simple la implementación sería mejor poder iterar directamente sobre la estructura y no sobre los vectores que devuelven esas funciones. Esto nos permitiría tener una mejor performance en las operaciones más costosas que son la multiplicación de matrices y la eliminación gaussiana.

- El modelo de PageRank es muy sensible a la cantidad de links entrantes que tiene un nodo pero citar a páginas con buen *pageRank* no mejora el ranking de la página citadora. Esto lo podemos ver en el experimento 3.2.2 donde el supernodo le gana a Wikipedia en el ranking. Esto pasa pues los links entrantes de Wikipedia son equivalentes a los links al supernodo pues ninguno es citado.

- Pudimos observar como al aumentar la probabilidad de que el navegante aleatorio se mueva a un link de la página donde está parado el PageRank de la página con mayor cantidad de links

aumenta. Esto ocurre pues cuando aumenta $p$ luego el navegante tiene más chances de caer en las páginas con más cantidad de links.

- Pudimos romper el *pageRank* de una red agregando un grafo supernodo y así conseguir que el supernodo sea el nodo con mejor *pageRank* . Pero esto no significa que el modelo no sea robusto en la vida real pues por las dimensiones que tiene la web el costo de romper el *pageRank* con nuestro método es demasiado alto. De todas formas sería interesante investigar en profundidad ese caso.

# 5. Referencias


- `https://en.wikipedia.org/wiki/PageRank`

- `https://en.wikipedia.org/wiki/Binary_search_tree`

- Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. Computer networks and ISDN systems, 30(1- 7):107117, 1998.