



DEPARTAMENTO  
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

# TP 1: PageRank

September 10, 2022

Métodos Numéricos

## Grupo 18

Integrante	LU	Correo electrónico
Vekselman, Natán	338/21	natanvek11@gmail.com
Arienti, Federico	316/21	fa.arianti@gmail.com
Barcos, Juan Cruz	463/20	juancruzbarcos@hotmail.com



## Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (+54 +11) 4576-3300

<http://www.exactas.uba.ar>

## RESUMEN

El Ranking de Page, *PageRank* [1], es un método propuesto por Sergey Brin y Larry Page —co-fundadores de Google—, para establecer la importancia de una página web dentro del internet, o dentro de un subconjunto de las páginas que lo componen. Holísticamente, la relevancia de un sitio se interpretará como la fracción de tiempo, al largo plazo, que un navegante permanecerá en él [2].

Desde una perspectiva algorítmica, PageRank busca resolver un sistema lineal  $\mathbf{A}x = x$ , donde  $\mathbf{A}$  es una matriz estocástica en columnas [2] y cada una de sus posiciones  $a_{ij}$  representa la probabilidad que un usuario situado en la página  $j$  decida navegar a la página  $i$ .

Este trabajo propone una implementación eficiente del ranking a través del uso de una estructura de matriz acorde al problema, y el empleo de iteradores específicos, para reducir el costo espacial y temporal de la eliminación gaussiana, método utilizado para su resolución.

Se buscará dar una presentación teórica y una evaluación cuantitativa y cualitativa de los resultados de tanto el método propuesto, como de PageRank en si.

Palabras clave: *Ranking de Page, Eliminación Gaussiana, Matrices ralas*

## CONTENIDOS

1. Introducción Teórica	2
1.1. Aridad	2
1.2. El sistema	2
1.3. Representación matricial	3
2. Desarrollo	5
2.1. Implementación	5
2.1.1. Matriz	5
2.1.2. Construir(g, p)	7
2.1.3. Eliminación gaussiana	8
2.1.4. Sustitución inversa	9
2.1.5. normalizar	10
2.2. Análisis cuantitativo	11
2.2.1. Error relativo	11
2.2.2. Error absoluto	12
2.3. Análisis cualitativo	13
3. Resultados y Discusión	14
4. Conclusiones	15
5. Apéndice	16
5.1. A: $A = pWD + ez^t$	16
5.2. B: $I - pWD$ permite la eliminación gaussiana	17
5.3. C: Estructuras alternativas	19
Referencias	20

## 1. INTRODUCCIÓN TEÓRICA

1.1. **Aridad.** Consideremos primero el dominio y la imagen de PageRank.

DOMINIO: 1. un conjunto de páginas web interconectadas a través de hipervínculos. Podemos considerar este conjunto como un grafo direccional, donde los nodos son los sitios y los ejes, los links. 2. un parámetro de entrada  $p \in (0, 1)$ , que representa la probabilidad que un usuario decida navegar aleatoriamente a otra página en el grafo. Se puede interpretar como el parámetro de un variable aleatoria de Bernoulli.

IMÁGEN: un vector  $x \in [0, 1]^n$ , donde  $x_i$  representa el Ranking de Page para la  $i$ -ésima página del conjunto de entrada, donde  $x$  satisface que  $x_i \geq 0 \forall i : 0 \dots n$  y  $\sum_{i=1}^n x_i = 1$ .

Tenemos entonces:

$$(1) \quad \text{PageRank} : G_n \times (0, 1) \longrightarrow [0, 1]^n \quad \forall n \in \mathbb{N}$$

donde  $G_n$  refiere al conjunto de conjuntos de páginas web, interconectadas a través de hipervínculos, con cardinalidad  $n$ .

1.2. **El sistema.** PageRank propone resolver un sistema de ecuaciones para encontrar la relevancia de cada página  $i$  ( $i : 1 \dots n$ ) en  $g \in G_n$ :

$$(2) \quad x_i := \sum_{j=1}^n x_j \cdot \text{Pr}(j \longrightarrow i)$$

donde  $\text{Pr}(j \longrightarrow i)$  es la probabilidad que un usuario situado en la página  $j$  decida ir a la página  $i$ . Se define de la siguiente manera:

$$(3) \quad \text{Pr}(j \longrightarrow i) := \begin{cases} (1-p) \cdot \frac{1}{n} + p \cdot \frac{I_{ij}}{c_j} & \text{si } c_j \neq 0 \\ \frac{1}{n} & \text{si no} \end{cases}$$

con  $I_{ij} = 1$  si y sólo si existe un hipervínculo de  $j$  a  $i$ , con  $j \neq i$  —y nulo en caso contrario—, y  $c_j = \sum_{i=1}^n I_{ij}$ , la cantidad de links salientes de  $j$ . La restricción  $j \neq i$  será para evitar que se consideren auto-referencias en el ranking.

Notemos que  $\text{Pr}(j \longrightarrow i)$  se puede interpretar de la siguiente manera: un navegante situado en la página  $j$  decidirá con probabilidad  $p$  acceder a uno de los links del sitio y con probabilidad  $1-p$  saltar a otra página del conjunto. En ambos casos, deberá luego decidir uniformemente sobre el total disponible, y terminará eligiendo a  $i$  con una probabilidad de  $\frac{I_{ij}}{c_j}$  ó  $\frac{1}{n}$ , respectivamente, acorde a la primer decisión. Si no hay links en la página, siempre saltará de manera uniforme a otra página del conjunto, y elegirá a  $i$  con probabilidad  $\frac{1}{n}$ .

$x_i$ , por su parte, también recibe una interpretación particular: es la probabilidad que para algún momento  $k > K$ , el navegante se encuentre situado en la página  $i$ . Para un  $K$  lo suficientemente grande, esta probabilidad es única [4]. Notar que esto es equivalente a decir que  $x_i$  representa la fracción de tiempo, al largo plazo, que un navegante permanecerá en la página  $i$ .

A este modelo se lo conoce como el *modelo del navegante aleatorio*. El mismo asume lo siguiente: Dado un link de la página  $j$  a la página  $i$ , la página  $j$  le conferirá importancia a la página  $i$  de manera proporcional a la relevancia de  $j$  e inversamente proporcional al número de páginas a las que apunta  $j$  [4]. Esto se puede ver en que  $x_i$  es la suma de toda otra importancia  $x_j$  ponderada por  $Pr(j \rightarrow i)$ , que incluye en su definición una división sobre la cantidad de links salientes  $c_j$ .

**1.3. Representación matricial.** Dado que estamos trabajando con un sistema lineal, será de utilidad considerar la matriz asociada ‘ $\mathbf{A}$ ’ y resolver, equivalentemente,  $\mathbf{A}x = x$ , donde  $x = (x_1, \dots, x_n)^t$ . Definimos entonces:

$$(4) \quad a_{ij} := Pr(j \rightarrow i)$$

y proponemos que<sup>1</sup>:

$$(5) \quad \mathbf{A} = p\mathbf{W}\mathbf{D} + ez^t$$

donde,  $\forall i, j : 1 \dots n$ , se satisface que:

$$e_i = 1$$

$$z_j = \begin{cases} (1-p)/n & \text{si } c_j \neq 0 \\ 1/n & \text{si no} \end{cases}$$

$$w_{ij} = \begin{cases} 1 & \text{si } i \neq j \wedge j \xrightarrow{l} i \\ 0 & \text{si no} \end{cases}$$

$$d_{ij} = \begin{cases} 1/c_j & \text{si } i = j \wedge c_j \neq 0 \\ 0 & \text{si no} \end{cases}$$

La notación  $j \xrightarrow{l} i$  representa que existe un link de la página  $j$  a la página  $i$ , y las filas y columnas de  $\mathbf{W}$ , denominada *matriz de conectividad*, representan —indexadas por posición— las páginas de una web  $g \in G_n$ .

---

<sup>1</sup>Una demostración de esta equivalencia se encuentra en 5.1.

A partir de la ecuación (5) podemos ver que:

$$\begin{aligned}
 \mathbf{A}x &= x \\
 (p\mathbf{WD} + ez^t)x &= x \\
 p\mathbf{WD}x + ez^tx &= x \\
 x - p\mathbf{WD}x &= ez^tx \\
 (\mathbf{I} - p\mathbf{WD})x &= \gamma e
 \end{aligned}$$

donde  $\gamma = z^tx$  es un escalar.

Dado que nuestro resultado deberá ser normalizado para cumplir con los requerimientos de la imagen (con la intuición que, si cada  $x_i$  representa una probabilidad disjunta, entonces la suma de las  $x_i$  deberá ser igual a uno, ya que su unión representa la probabilidad del espacio entero), podemos asumir un  $\gamma$  conveniente [3],  $\gamma = 1$ , tal que el sistema a resolver sea:

$$(6) \quad (\mathbf{I} - p\mathbf{WD})x = e$$

Notar que La matriz  $\mathbf{I} - p\mathbf{WD}$  permite la aplicación de la eliminación gaussiana sin permutación <sup>2</sup>.

---

<sup>2</sup>Una demostración de este enunciado se encuentra en 5.2.

## 2. DESARROLLO

**2.1. Implementación.** A partir del sistema planteado en la ecuación (6), proponemos el siguiente método para la resolución de PageRank:

- 1) Construir la matriz  $\mathbf{I} - p\mathbf{WD}$  a partir de alguna representación de  $g \in G_n$  (recordar que  $g$  es un conjunto de páginas web con cardinalidad  $n$ ).
- 2) Triangular la matriz extendida  $(\mathbf{I} - p\mathbf{WD} \mid e)$  mediante eliminación gaussiana, sin pivoteo<sup>3</sup>.
- 3) Resolver la matriz triangular resultante mediante el algoritmo de sustitución inversa.
- 4) Normalizar el resultado.

Tenemos entonces:

---

```

1 proc PageRank(in g: G<n>, in p: real, out x: vector<n>) {
2     /* Pre: 0 < p < 1 */
3     B := construir(g, p) // B = I - pWD
4     e := vector(n, 1)    // e = 1 de dimension n
5     eliminacion_gaussiana(B, e)
6     x := sustitucion_inversa(B, e)
7     normalizar(x)
8     /* Post: |Bx - x| < 1e-4 */
9 }

```

---

ALGORITMO 1. Pseudocódigo para *PageRank*.

De este algoritmo surgen las siguientes preguntas: ¿Cómo representamos las matrices<sup>4</sup>? ¿Cómo construimos  $(\mathbf{I} - p\mathbf{WD} \mid e)$ ? ¿Cómo implementamos la eliminación gaussiana? y ¿Cómo implementamos la sustitución inversa?

**2.1.1. Matriz.** Una representación eficiente de matriz será una que permita aprovechar las cualidades de  $(\mathbf{I} - p\mathbf{WD} \mid e)$ , que tenga un costo mínimo de mantenimiento respecto a sus operaciones elementales y que sea eficiente en el uso de la memoria.

Lo primero a notar son las operaciones fundamentales de la estructura. Desde un punto de vista abstracto, estas son aquellas que la definen como un espacio vectorial y, en el caso de  $\mathbb{R}^{n \times n}$ , como un álgebra:

---

<sup>3</sup>Es decir, nuestra solución no buscará reducir el error numérico de la aritmética de punto flotante —en particular por cancelación catastrófica— por medio de las técnicas de pivoteo parcial o pivoteo completo que se utilizan en algunas implementaciones del algoritmo.

<sup>4</sup>Para este trabajo asumiremos que existe una representación de vector. Sin embargo, notar que un vector puede ser representado por una matriz  $n \times 1$  ó  $1 \times n$ .

$$\mathbf{A} + \mathbf{B} := (a + b)_{ij} = a_{ij} + b_{ij} \quad \forall i : 1 \dots n, j : 1 \dots m$$

$$\mathbf{A} \cdot \mathbf{B} := (ab)_{ij} = \sum_{k=1}^m a_{ik} b_{kj} \quad \forall i : 1 \dots n, j : 1 \dots q$$

$$\lambda \cdot_{\mathbb{R}} \mathbf{A} := (\lambda a)_{ij} = \lambda a_{ij} \quad \forall i : 1 \dots n, j : 1 \dots m$$

Más allá de las posibles implementaciones —que no se requerirán—, importa destacar el fuerte carácter iterativo (las operaciones y la acción actúan sobre todas las posiciones) y observar que cuando  $a_{ij} = 0$  ó  $b_{ij} = 0$ , el resultado es invariante respecto a al menos uno de los operandos. Desde un punto de vista algorítmico, esto significa que estos casos son redundantes y posiblemente se los pueda omitir (por ejemplo, iterando sólo sobre los elementos no nulos).

Esto es particularmente importante para PageRank, donde se espera que una página web tenga pocos links salientes (en relación al total de los sitios) y donde se puede demostrar una tendencia a la localidad de las relaciones [3]. Entonces, para un conjunto lo suficientemente grande, podemos suponer que nuestra matriz estará principalmente vacía.

A partir de este breve análisis, proponemos las siguientes estructuras<sup>5</sup>:

$$(7) \quad \text{matriz} := \left\{ \begin{array}{l} n, m : \mathbb{N} \\ \text{datos} : \text{vector} < \text{vector} < \text{par} := \left\{ \begin{array}{l} \text{posición} : \mathbb{N}_0, \\ \text{elemento} : \mathbb{R} \end{array} \right. >> \\ \text{at} : \mathbb{N} i \times \mathbb{N} j \longrightarrow \mathbb{R} \quad \{0 \leq i < n \wedge 0 \leq j < m\} \\ \text{set} : \mathbb{N} i \times \mathbb{N} j \times \mathbb{R} \longrightarrow \text{matriz} \quad \{0 \leq i < n \wedge 0 \leq j < m\} \end{array} \right.$$

$$(8) \quad \text{iterador} := \left\{ \begin{array}{l} i, j, \text{pos} : \mathbb{N}_0 \\ p : * \text{matriz} \\ \text{at} : \longrightarrow \mathbb{R} \quad \{\text{en\_rango}()\} \\ \text{set} : \mathbb{R} \longrightarrow \text{iterador} \quad \{\text{en\_rango}()\} \\ \text{proximo\_fila} : \longrightarrow \text{iterador} \\ \text{proximo\_columna} : \longrightarrow \text{iterador} \\ \text{en\_rango} : \longrightarrow \text{bool} \end{array} \right.$$

donde *vector* se refiere a un arreglo de tamaño variable en memoria contigua y *\** designa un puntero.

---

<sup>5</sup>Una evaluación en más detalle de la implementación y las alternativas consideradas se encuentra en (5.3).

Esta estructura de matriz garantizará que el tamaño del vector externo siempre equivaldrá a  $n$ , que el tamaño de cada vector interno estará acotado por  $m$ , que los pares internos estarán ordenados por posición y que para cada uno se satisfará que  $0 \leq \text{posición} < n$  y que el elemento es no nulo.

La estructura asociada al iterador, por su parte, satisfará que el iterador es válido si y sólo si  $0 \leq i < n$  y  $0 \leq \text{pos} < \text{largo}(\text{datos}[i])$ , y que un iterador válido siempre estará sobre un elemento no nulo<sup>6</sup>.

Entre ambas podremos iterar por los elementos no nulos de cada fila en  $\Theta(1)$  —ya que estarán en orden sucesivo en el vector interno— y será eficiente en espacio<sup>7</sup>. Como contrapartida, no será eficiente en la inserción: en el peor caso requerirá mover todos los elementos en una fila (por un costo en  $\Theta(m)$ ).

2.1.2. *Construir*( $g, p$ ). Dada nuestra representación de matriz, el siguiente paso será construir, a partir de un conjunto de páginas web  $g$  y un valor  $p$ , la matriz  $\mathbf{I} - p\mathbf{W}\mathbf{D}$ . Para ello, vamos a adoptar una representación particular de  $g$ :

$$(9) \quad g \in G_n := \left\{ \begin{array}{l} \text{\#páginas : } n, \\ \text{relaciones : } \text{vector} < \text{eje} := \{ i, j : \mathbb{N} \} > \end{array} \right.$$

donde cada eje representa un hipervínculo de la página  $j$  a la página  $i$  y se satisface que  $0 \leq i, j < n$ .

Proponemos el siguiente algoritmo:

---

```

1 proc construir(in g: G<n>, in p: real, out B: matriz<n, n>) {
2   n := g.#paginas
3   B := identidad(n)    // B = I de dimension n x n
4   ponderar(B, g, p)    // B = I - pWD
5 }
```

---

ALGORITMO 2. Pseudocódigo para *construir*.

Notamos que la matriz  $\mathbf{D}$  es diagonal y, por ende, la multiplicación a derecha con  $\mathbf{W}$  equivaldrá a escalar los elementos de cada columna  $W_j$  por  $d_{jj}$ . En el caso de los elementos nulos, la operación será invariante —algo que también sucederá con la multiplicación escalar por  $-p$ —.

---

<sup>6</sup>Sobre el iterador, debemos aclarar que éste se situará en la columna no nula más cercana a la pedida, que el método PROXIMO\_COLUMNNA situará al iterador sobre la columna no nula más próxima (en relación a la que se usó para inicializar el iterador originalmente) de la siguiente fila, o continuará si esta es vacía, y PROXIMO\_FILA situará al iterador sobre el próximo elemento no nulo en la misma fila.

<sup>7</sup>En relación a nuestras expectativas sobre la realidad de PageRank.



Tenemos entonces:

---

```

1 proc identidad(in n: natural, out B: matriz<n, n>) {
2     B := matriz(n, n, 0) // B = 0 de dimension n x n
3     i := 0
4     while i < n {
5         B.set(i, i, 1)
6         i := i + 1
7     }
8 }

```

---

ALGORITMO 3. Pseudocódigo para *identidad*.

---

```

1 proc ponderar(inout B: matriz<n, n>, in g: G<n>, in p: real) {
2     grado := vector(n, 0) // grado = 0 de dimension n
3     i = 0
4     while i < largo(g.relaciones) {
5         eje := g.relaciones[i]
6         grado[eje.j] := grado[eje.j] + 1
7     }
8     while i < largo(g.relaciones) {
9         eje := g.relaciones[i]
10        if eje.i != eje.j {
11            B.set(eje.i, eje.j, -p / grado[eje.j])
12        }
13    }
14 }

```

---

ALGORITMO 4. Pseudocódigo para *ponderar*.

De ambos algoritmos podemos notar que la complejidad de CONSTRUIR estará en  $\Theta(n+r \cdot c)$  donde  $r$  representa la cantidad total de relaciones y  $c$  es el costo de SET. Para nuestra representación, esto equivaldrá a  $\Theta(n + r \cdot n)$  y, como  $r \leq n^2$ , tendremos un peor caso en  $\Theta(n^3)$ . Sin embargo, si  $r \ll n^2$ , se puede esperar un comportamiento mejor que el método ‘directo’ (cuyo costo fijo es  $\Theta(n^3)$  por el producto de matrices).

2.1.3. *Eliminación gaussiana*. El siguiente paso será triangular  $(\mathbf{I} - p\mathbf{WD} \mid e)$  para obtener un sistema fácil de resolver. Proponemos una versión optimizada de la eliminación gaussiana —para nuestro uso— que se vale de las siguientes observaciones:

- Dada una matriz  $\mathbf{A}$  que permite la eg., el paso  $i$ -ésimo del algoritmo, que modifica la fila  $j$  ( $j > i$ ), es necesario sólo si  $a_{ji} \neq 0$ , ya que sino la operación  $A_j \leftarrow A_j - \frac{a_{ji}}{a_{ii}} \cdot A_i$  es invariante.
- Similarmente,  $A_j \leftarrow A_j - \frac{a_{ji}}{a_{ii}} \cdot A_i$ , requiere operar sólo sobre los elementos no nulos de  $A_i$ .

---

```

1 proc eliminacion_gaussiana(inout A: matriz<n, n>, inout b: vector<n>) {
2     i := 0
3     while i < A.n - 1 {
4         jt := iterador(A, i + 1, i)
5         while jt.en_rango() {
6             if (jt.j == i) { // solo si B.at(i + 1, i) != 0
7                 mij := jt.at() / A.at(i, i)
8                 sumar_fila(A, b, i, jt.i, mij)
9             }
10            jt.proximo_columna()
11        }
12        i := i + 1
13    }
14 }

```

---

ALGORITMO 5. Pseudocódigo para *eliminacion\_gaussiana*.

---

```

1 proc sumar_fila(inout A: matriz<n, n>, inout b: vector<n>,
2               in f1, f2: natural, in c: real) {
3     it := iterador(A, f1, f1)
4     while it.en_rango() {
5         val := B.at(f2, it.j) - c * it.at()
6         B.set(f2, it.j, val)
7         it.proximo_fila()
8     }
9     b[f2] := b[f2] - mij * b[f1]
10 }

```

---

ALGORITMO 6. Pseudocódigo para *sumar\_fila*.

Para no complicar el análisis, podemos estimar que el algoritmo costará  $O(n^2 \cdot k^2)$  en el caso promedio. El  $n^2$  corresponde a la iteración de cada fila  $j$  sobre cada fila  $i$ , dado  $0 \leq i < j < n$ , y  $k$  corresponde a la cantidad de elementos no nulos en la matriz<sup>8</sup>. Dado  $k \leq n$ , el costo de peor caso será  $O(n^4)$ . Sin embargo, mientras  $k < \sqrt{n}$  este método funcionará mejor que la forma tradicional de la eliminación (cuyo costo teórico es  $\Theta(n^3)$ ).

2.1.4. *Sustitución inversa*. Al igual que en la eliminación gaussiana, aprovecharemos el hecho que  $A_j \leftarrow A_j - \frac{a_{ji}}{a_{ii}} \cdot A_i$  requiere operar sólo sobre los elementos no nulos de  $A_i$ , para lograr un algoritmo en  $\Theta(n \cdot k)$ , con  $k$  la cantidad de elementos no nulos de la matriz:

---

<sup>8</sup>Se estima  $k^2$  por SUMAR\_FILA, que itera sobre los elementos no nulos de una fila e incluye una operación de SET.

---

```

1 proc sustitucion_inversa(in A: matriz<n, n>, in b: vector<n>,
2                               out x: vector<n>) {
3     x := vector(A.n, 0) // x = 0 de dimension n
4     i := A.n - 1
5     while i >= 0 {
6         s := 0
7         it := iterador(A, i, i + 1)
8         while it.en_rango() {
9             s := s + it.at() * x[it.j]
10            it.proximo_fila()
11        }
12        x[i] := (b[i] - s) / A.at(i, i)
13        i := i + 1
14    }
15 }

```

---

ALGORITMO 7. Pseudocódigo para *sustitucion\_inversa*.

Nuevamente, el peor caso se dará cuando  $k = n$ .

2.1.5. *normalizar*. Por último, presentaremos un algoritmo estándar para normalizar un vector de tamaño  $n$  en  $\Theta(n)$ :

---

```

1 proc normalizar(inout x: vector<n>) {
2     s := 0
3     i := 0
4     while i < largo(x) {
5         s += x[i]
6         i := i + 1
7     }
8     i := 0
9     while i < largo(x) {
10        x[i] = x[i] / s
11        i := i + 1
12    }
13 }

```

---

ALGORITMO 8. Pseudocódigo para *normalizar*.

**2.2. Análisis cuantitativo.** Se procederá a evaluar una implementación de PageRank en C++ acorde a los algoritmos propuestos.

**2.2.1. Error relativo.** Medimos el error  $|\mathbf{A}x - x|_1$  en función del valor de  $p$  para cien grafos generados aleatoriamente. En total, obtuvimos 10,000 mediciones<sup>9</sup>.

**METODOLOGÍA.** Se calculó  $x = \text{PageRank}(g, p)$  y se midió el error relativo  $|\mathbf{A}x - x|_1$  para cada uno de los grafos sobre cada valor de  $p$  en el intervalo  $(0, 1)$  de a saltos de 0.01.

Cada caso, representable por una matriz de conectividad  $W \in \{0, 1\}^{100 \times 100}$ , se generó a través del siguiente procedimiento<sup>10</sup>:

- 1) Se eligió la cantidad de ejes ( $e$ ) del sistema de manera uniforme sobre el intervalo  $[0, R \cdot T)$ , donde  $T = 100^2 - 100$  representa el máximo de ejes posibles en un grafo de cien nodos sin auto-direccionamiento y  $R = 1/4$  es un valor arbitrario definido para imitar las características de realidad esperables en un conjunto de páginas web.
- 2) Se pobló una matriz  $W_0 \in \{0, 1\}^{99 \times 100} = 0$  con unos en las primeras ‘e’ posiciones y se utilizó el algoritmo de shuffle de numpy, sobre el rng PCG64, para generar una permutación aleatoria.
- 3) Se expandió la misma con ceros en la diagonal para lograr la matriz  $W \in \{0, 1\}^{100 \times 100}$ .

**OBSERVACIONES.** El experimento tiene como limitaciones principales el tamaño de la muestra (cien grafos distintos) y el método de generación de casos —los mismos no provienen de muestras reales—, que incluye la elección arbitraria del valor  $R$ . Sin embargo, contempla con cierta granularidad todo el espectro de valores posibles para  $p$ , tal que permite conocer el error relativo de los resultados en función de su parámetro ‘libre’.

**RESULTADOS.** El cuadro 1. resume los resultados obtenidos.

mediciones	10000
error relativo promedio	0.00084646
desviación estándar	0.00063764
mínimo	1.7347e-16
25%	0.00034960
50%	0.00075431
75%	0.00121044
máximo	0.00376671

Cuadro 1. datos de resumen del experimento.

Podemos observar que el error relativo fue en promedio menor a  $1e - 3$ . De distribuirse uniformemente, esto nos permite suponer que el error relativo de cada puntaje debe estar en el orden de  $1e - 5$ .

<sup>9</sup>El script asociado se puede encontrar en `./experimentos/error_relativo.py`

<sup>10</sup>Se utilizó un valor semilla para facilitar la reproducibilidad.

La Figura 1. muestra la distribución de los resultados en función del parámetro  $p$ .

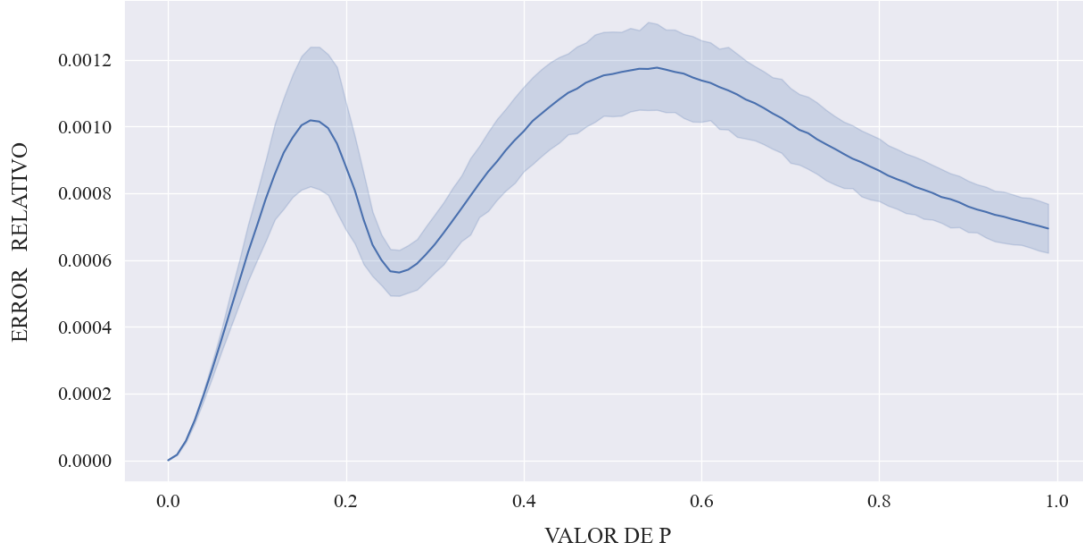


FIGURA 1. Error relativo promedio -en base a una norma L1- e intervalo de confianza del 95% para una muestra aleatoria de cien grafos en función de  $p$ .

Notar la progresiva disminución del error a medida que  $p$  se aproxima a 1. Los máximos locales se encuentran en  $p = 0.16$  y  $p = 0.55$ , y el mínimo local en  $p = 0.26$ .

2.2.2. *Error absoluto*. Medimos el error  $|x - \hat{x}|_1$  para los casos de test provistos por la cátedra<sup>11</sup>.

El Cuadro 2. resume los resultados.

test	error
test_15_segundos	0.0291137
test_30_segundos	0.0229572
test_aleatorio	6.176e-07
test_aleatorio_desordenado	6.176e-07
test_completo	0.0
test_sin_links	0.0
test_trivial	0.0

Cuadro 2. Error absoluto en base a la -norma L1- de los tests.

<sup>11</sup>Los resultados coordenada a coordenada se pueden observar en `./experimentos/resultados/error_tests`.

Podemos observar que hay cierta correlación entre la cantidad de páginas y el error. Esto tiene sentido dado que la norma L1 es la suma del valor absoluto de las coordenadas. Mientras mayor sea la dimensión, mayor será la cantidad de errores a sumar.

### 2.3. **Análisis cualitativo.**

### 3. RESULTADOS Y DISCUSIÓN

#### 4. CONCLUSIONES



## 5. APÉNDICE

5.1. **A:**  $A = pWD + ez^t$ .

*demostración.* Recordemos que:

$$e_i = 1$$

$$z_j = \begin{cases} (1-p)/n & \text{si } c_j \neq 0 \\ 1/n & \text{si no} \end{cases}$$

$$w_{ij} = \begin{cases} 1 & \text{si } i \neq j \wedge j \xrightarrow{l} i \\ 0 & \text{si no} \end{cases}$$

$$d_{ij} = \begin{cases} 1/c_j & \text{si } i = j \wedge c_j \neq 0 \\ 0 & \text{si no} \end{cases}$$

A partir de estas definiciones, vemos que, como **D** es diagonal, el producto a derecha **WD** escala cada columna  $w_j$  por el factor  $d_{jj}$ , tal que:

$$(\mathbf{WD})_{ij} = \begin{cases} w_{ij}/c_j & \text{si } c_j \neq 0 \\ 0 & \text{si no} \end{cases}$$

Como  $p$  es un escalar, sigue entonces que:

$$(p\mathbf{WD})_{ij} = \begin{cases} p \cdot w_{ij}/c_j & \text{si } c_j \neq 0 \\ 0 & \text{si no} \end{cases}$$

Además,  $e \in \mathbb{R}^{n \times 1} \wedge z^t \in \mathbb{R}^{1 \times n} \implies ez^t \in \mathbb{R}^{n \times n}$ , y:

$$(ez^t)_{ij} := \sum_{k=1}^1 e_{ik} \cdot z_{kj}^t = e_i \cdot z_j^t = 1 \cdot z_j^t = z_j$$

Por lo que:

$$\begin{aligned} (p\mathbf{WD} + ez^t)_{ij} &= \begin{cases} p \cdot w_{ij}/c_j + z_j & \text{si } c_j \neq 0 \\ z_j & \text{si no} \end{cases} \\ &= \begin{cases} (1-p) \cdot \frac{1}{n} + p \cdot \frac{w_{ij}}{c_j} & \text{si } c_j \neq 0 \\ \frac{1}{n} & \text{si no} \end{cases} \end{aligned}$$

pero:

$$a_{ij} := Pr(j \longrightarrow i) = \begin{cases} (1-p) \cdot \frac{1}{n} + p \cdot \frac{I_{ij}}{c_j} & \text{si } c_j \neq 0 \\ \frac{1}{n} & \text{si no} \end{cases}$$

Como  $I_{ij} = 1$  si y sólo si existe un hipervínculo de  $j$  a  $i$ , con  $j \neq i$  —y nulo en caso contrario—, entonces  $I_{ij} = w_{ij}$  y concluimos que  $a_{ij} = (p\mathbf{WD} + ez^t)_{ij}$ ,  $\forall i, j : 1 \dots n$ , lo que implica que:

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

■

**5.2. B:  $I - pWD$  permite la eliminación gaussiana.** Definimos la eliminación Gaussiana de la siguiente manera:

$$\mathbf{A} := \mathbf{I} - p\mathbf{WD}$$

$$\begin{aligned} \mathbf{Eg}_1(\mathbf{A}) &= \mathbf{A} \\ \mathbf{Eg}_{k+1}(\mathbf{A})_{ij} &= \begin{cases} \mathbf{Eg}_k(\mathbf{A})_{ij} & \text{si } i < k+1 \vee j < k+1 \\ \mathbf{Eg}_k(\mathbf{A})_{ij} - \frac{\mathbf{Eg}_k(\mathbf{A})_{ik} \cdot \mathbf{Eg}_k(\mathbf{A})_{kj}}{\mathbf{Eg}_k(\mathbf{A})_{kk}} & \text{si no} \end{cases} \end{aligned}$$

$$HI : (\forall j \in \mathbb{N})(k \leq j \leq N) \longrightarrow_L |\mathbf{Eg}_k(\mathbf{A})_{jj}| \geq \sum_{i=k, i \neq j}^N |\mathbf{Eg}_k(\mathbf{A})_{ij}|$$

$$CB : (k = 1)$$

$$(\forall j \in \mathbb{N})(1 \leq j \leq N) \longrightarrow_L |\mathbf{A}_{jj}| \geq \sum_{i=1, i \neq j}^N |\mathbf{A}_{ij}|$$

$$(\forall j \in \mathbb{N})(1 \leq j \leq N) \longrightarrow_L 1 \geq \sum_{i=1, i \neq j}^N |(1/C_j \vee 0)|$$

$$(\forall j \in \mathbb{N})(1 \leq j \leq N) \longrightarrow_L 1 \geq (|p/C_j| \vee 0) \cdot C_j$$

$$(\forall j \in \mathbb{N})(1 \leq j \leq N) \longrightarrow_L 1 \geq |p| > (|p/C_j| \vee 0) \cdot C_j$$

$$\text{sabemos} : 0 < p < 1 \rightarrow 1 \geq |p|$$

$$(\forall j \in \mathbb{N})(1 \leq j \leq N) \longrightarrow_L \text{True}$$

$$QVQ : (\forall j \in \mathbb{N})(k+1 \leq j \leq N) \longrightarrow_L |\mathbf{Eg}_{k+1}(\mathbf{A})_{jj}| \geq \sum_{i=k+1, i \neq j}^N |\mathbf{Eg}_{k+1}(\mathbf{A})_{ij}|$$

$$\longrightarrow_L |\mathbf{Eg}_k(\mathbf{A})_{jj} - \frac{\mathbf{Eg}_k(\mathbf{A})_{jk} \cdot \mathbf{Eg}_k(\mathbf{A})_{kj}}{\mathbf{Eg}_k(\mathbf{A})_{kk}}| \geq \sum_{i=k+1, i \neq j}^N |\mathbf{Eg}_k(\mathbf{A})_{ij} - \frac{\mathbf{Eg}_k(\mathbf{A})_{ik} \cdot \mathbf{Eg}_k(\mathbf{A})_{kj}}{\mathbf{Eg}_k(\mathbf{A})_{kk}}|$$

$$\begin{aligned} \longrightarrow_L |\mathbf{Eg}_k(\mathbf{A})_{jj}| - \left| \frac{\mathbf{Eg}_k(\mathbf{A})_{jk} \cdot \mathbf{Eg}_k(\mathbf{A})_{kj}}{\mathbf{Eg}_k(\mathbf{A})_{kk}} \right| &\geq \sum_{i=k+1, i \neq j}^N |\mathbf{Eg}_k(\mathbf{A})_{ij}| + \sum_{i=k+1, i \neq j}^N \left| \frac{\mathbf{Eg}_k(\mathbf{A})_{ik} \cdot \mathbf{Eg}_k(\mathbf{A})_{kj}}{\mathbf{Eg}_k(\mathbf{A})_{kk}} \right| \\ \sum_{i=k+1, i \neq j}^N |\mathbf{Eg}_k(\mathbf{A})_{ij}| &= \sum_{i=k, i \neq j}^N (|\mathbf{Eg}_k(\mathbf{A})_{ij}|) - |\mathbf{Eg}_k(\mathbf{A})_{kj}| \end{aligned}$$

$$\text{Por HI: } |\mathbf{Eg}_k(\mathbf{A})_{jj}| - |\mathbf{Eg}_k(\mathbf{A})_{kj}| \geq \sum_{i=k, i \neq j}^N (|\mathbf{Eg}_k(\mathbf{A})_{ij}|) - |\mathbf{Eg}_k(\mathbf{A})_{kj}|$$

$$|\mathbf{Eg}_k(\mathbf{A})_{jj}| - \left| \frac{\mathbf{Eg}_k(\mathbf{A})_{jk} \cdot \mathbf{Eg}_k(\mathbf{A})_{kj}}{\mathbf{Eg}_k(\mathbf{A})_{kk}} \right| \geq |\mathbf{Eg}_k(\mathbf{A})_{jj}| - |\mathbf{Eg}_k(\mathbf{A})_{kj}| + \sum_{i=k+1, i \neq j}^N \left| \frac{\mathbf{Eg}_k(\mathbf{A})_{ik} \cdot \mathbf{Eg}_k(\mathbf{A})_{kj}}{\mathbf{Eg}_k(\mathbf{A})_{kk}} \right|$$

$$|\mathbf{Eg}_k(\mathbf{A})_{kj}| - \left| \frac{\mathbf{Eg}_k(\mathbf{A})_{jk} \cdot \mathbf{Eg}_k(\mathbf{A})_{kj}}{\mathbf{Eg}_k(\mathbf{A})_{kk}} \right| \geq \frac{|\mathbf{Eg}_k(\mathbf{A})_{kj}|}{|\mathbf{Eg}_k(\mathbf{A})_{kk}|} \sum_{i=k+1, i \neq j}^N |\mathbf{Eg}_k(\mathbf{A})_{ik}|$$

$$\sum_{i=k+1, i \neq j}^N |\mathbf{Eg}_k(\mathbf{A})_{ik}| = \sum_{i=k+1}^N (|\mathbf{Eg}_k(\mathbf{A})_{ik}|) - |\mathbf{Eg}_k(\mathbf{A})_{jk}|$$

$$\text{Por HI: } |\mathbf{Eg}_k(\mathbf{A})_{kk}| - |\mathbf{Eg}_k(\mathbf{A})_{jk}| \geq \sum_{i=k+1}^N (|\mathbf{Eg}_k(\mathbf{A})_{ik}|) - |\mathbf{Eg}_k(\mathbf{A})_{jk}|$$

$$|\mathbf{Eg}_k(\mathbf{A})_{kj}| - \left| \frac{\mathbf{Eg}_k(\mathbf{A})_{jk} \cdot \mathbf{Eg}_k(\mathbf{A})_{kj}}{\mathbf{Eg}_k(\mathbf{A})_{kk}} \right| \geq \frac{|\mathbf{Eg}_k(\mathbf{A})_{kj}|}{|\mathbf{Eg}_k(\mathbf{A})_{kk}|} (|\mathbf{Eg}_k(\mathbf{A})_{kk}| - |\mathbf{Eg}_k(\mathbf{A})_{jk}|)$$

$$|\mathbf{Eg}_k(\mathbf{A})_{kj}| - \left| \frac{\mathbf{Eg}_k(\mathbf{A})_{jk} \cdot \mathbf{Eg}_k(\mathbf{A})_{kj}}{\mathbf{Eg}_k(\mathbf{A})_{kk}} \right| \geq |\mathbf{Eg}_k(\mathbf{A})_{kj}| - \left| \frac{\mathbf{Eg}_k(\mathbf{A})_{jk} \cdot \mathbf{Eg}_k(\mathbf{A})_{kj}}{\mathbf{Eg}_k(\mathbf{A})_{kk}} \right|$$

$$(\forall j \in \mathbb{N})(1 \leq j \leq N) \longrightarrow_L \text{True}$$

QED que  $(\forall j \in \mathbb{N})(k \leq j \leq N) \longrightarrow_L |\mathbf{Eg}_k(\mathbf{A})_{jj}| \geq \sum_{i=k, i \neq j}^N |\mathbf{Eg}_k(\mathbf{A})_{ij}|$

Por ende el elemento  $|\mathbf{Eg}_k(\mathbf{A})_{kk}| = 0 \leftrightarrow \sum_{i=k+1}^N |\mathbf{Eg}_k(\mathbf{A})_{ik}| = 0$  O sea que el elemento de la diagonal en cada paso de la eliminacion gaussiana es 0 si y solo si todos los elementos que tiene por debajo en su columna son 0 o sea que se podria saltar ese paso

### 5.3. C: Estructuras alternativas.

## REFERENCIAS

- [1] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30 (1-7):107–117, 1998.
- [2] Kurt Bryan and Tanya Leise. The \$25,000,000,000 eigenvector: The linear algebra behind google. *T SIAM review*, 48 (3):569–581, 2006.
- [3] Amy N Langville and Carl D Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2004.
- [4] Sepandar D Kamvar; Taher H Haveliwala; Christopher D Manning and Gene H Golub. Extrapolation methods for accelerating pagerank computations. *In Proceedings of the 12th international conference on World Wide Web*, pages 261-270. ACM, 2003.