TP 1: PageRank

September 11, 2022 Métodos Numéricos

Grupo 18

Integrante	LU	Correo electrónico
Vekselman, Natán	338/21	natanvek11@gmail.com
Arienti, Federico	316/21	fa.arienti@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 jerarquizar las páginas web del internet, o de un subconjunto de las páginas que lo componen. Holísticamente, se puede interpretar como una medición, al largo plazo, del porcentaje de tiempo que un navegante permancerá en cada uno de los sitios [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 matríz 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, Representación de Matrices*

Contenidos

1. Introducción Teórica

1.1. Aridad. Consideremos primero el dominio y la imágen de PageRank.

Dominio: 1. un conjunto de páginas web interconectadas a través de hipervínculos. Podemos considerar este conjunto como un grafo direccionado, 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 y x satisface que $x_i \geq 0\ \forall i : 0\ ...\ n$ y $\sum_{i=1}^{n} x_i = 1$.

Tenemos entonces:

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

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 $(i : 1\ ...\ n)$ en $g \in G_n$:

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

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

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

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 $Pr(j \longrightarrow i)$ se puede interpretar de la siguente 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 eligirá 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 [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*.

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

$$a_{ij} := Pr(j \longrightarrow i) \tag{4}$$

y proponemos que[1]:

$$\mathbf{A} = p\mathbf{W}\mathbf{D} + ez^t \tag{5}$$

donde $\mathbf{A} \in [0, \ 1]^{n \times n}$ y $\forall i, \ j : \ 1 \ ... \ n$ se satisface que:

$$e_i = 1$$

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

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

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

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 $g \in G_n$.

A partir de la Ecuación 5 podemos ver que:

[1]Una demostración de esta equivalencia se encuentra en 5.1.

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

donde $\gamma = z^t x$ es un escalar.

Dado que nuestro resultado deberá ser normalizado para cumplir con los requerimientos de la imágen, podemos asumir un γ conveniente [3], $\gamma = 1$, tal que el sistema a resolver sea:

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

Notar que La matriz $\mathbf{I} - p\mathbf{WD}$ es estocástica en columnas, por lo que permite la aplicación de la eliminación gaussiana sin permutación[2].

[2]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[3].
3) Resolver la matriz resultante mediante el algoritmo de sustitución inversa.
4) Normalizar el resultado.

En código:

```
proc PageRank(in g: G<n>, in p: real, out x: vector<n>) {
    /* Pre: 0 < p < 1 */
    B := construir(g, p) // B = I - pWD
    e := vector(n, 1)    // e = 1 de dimension n
    eliminacion_gaussiana(B, e)
    x := sustitucion_inversa(B, e)
    normalizar(x)
    /* Post: x[i] >= 0 para toda i y sum(x) = 1  */
}
```

Algoritmo 1. Pseudocódigo para *PageRank*.

De este algoritmo surgen las siguientes preguntas: ¿Cómo representamos las matrices[4]? ¿Cómo construimos $\mathbf{I} - p\mathbf{WD}$? ¿Cómo implementamos la eliminación gaussiana? y ¿Cómo implementamos la sustitución inversa?

2.1.1. *Matríz.* Una representación eficiente de matríz será una que permita aprovechar las cualidades de $\mathbf{I} - p\mathbf{WD}$, 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:

[3]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.

[4]Para este trabajo asumiremos que existe una implementación de vector. Sin embargo, notar que un vector puede ser representado por una matríz $n \times 1$ ó $1 \times n$.

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

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

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

Más allá de las posibles implementaciones —que veremos no hacen falta—, 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]. Para un conjunto lo suficientemente grande, podemos suponer que nuestra matríz estará principalmente vacía.

A partir de este breve análisis, proponemos las siguientes estructuras[5]:

$$(7) \qquad \text{matriz} := \begin{cases} \text{n, m} : \ \mathbb{N} \\ \text{datos} : \textit{vector} < \textit{vector} < \text{par} := \begin{cases} \text{posición} : \mathbb{N}_0, \\ \text{elemento} : \mathbb{R} \end{cases} >> \\ \\ \text{at} : \mathbb{N}\ i \times\ \mathbb{N}\ j \ \longrightarrow\ \mathbb{R} & \{0 \leq i < n \wedge 0 \leq j < m\} \\ \text{set} : \mathbb{N}\ i \times\ \mathbb{N}\ j\ \times \mathbb{R} \longrightarrow\ \text{matriz} & \{0 \leq i < n \wedge 0 \leq j < m\} \end{cases}$$

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

---

[5]Una evaluación en más detalle de la implementación y las alternativas consideradas se encuentra en el Apéndice 5.3.

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

Esta estructura de matríz mantendrá las siguientes garantías: el tamaño del vector externo siempre equivaldrá a n, el tamaño de cada vector interno estará acotado por m, los pares internos estarán ordenados por posición y 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 $0 \leq pos < largo(datos[i])$, que $j = (*p).datos[i][pos].posicion$, y que un iterador válido siempre estará sobre un elemento distinto a cero[6].

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[7]. Como contrapartida, no será eficiente en la inserción: en el peor caso requerirá mover todos los elementos en una fila (por un costo en $\Theta(m)$).

Una decisión importante a mencionar es que consideraremos nulo a cualquier valor menor a $1e-4$. Este valor permite mejores tiempos en la ejecución y mantiene un error en el orden de $1e-5$ coordenada a coordenada[8].

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

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

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:

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

ALGORITMO 2. Pseudocódigo para *construir*.

[6]Sobre el iterador, debemos aclarar que éste se situará en la columna no nula más cercana a la pedida, que el método PROXIMO_COLUMNA 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 ésta es vacía, y PROXIMO_FILA situará al iterador sobre el próximo elemento no nulo en la misma fila.
[7]En relación a nuestras expectativas sobre la ralidad de PageRank.
[8]La medición de los resultados se encuentra en 3.1.

Notamos que la matríz $\mathbf{D}$ 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 multiplación escalar por $-p$—.

Tenemos entonces:

```
proc identidad(in n: natural, out B: matriz<n, n>) {
    B := matriz(n, n, 0) // B = 0 de dimension n x n
    i := 0
    while i < n {
        B.set(i, i, 1)
        i := i + 1
    }
}
```

Algoritmo 3. Pseudocódigo para *identidad*.

```
proc ponderar(inout B: matriz<n, n>, in g: G<n>, in p: real) {
    grado := vector(n, 0) // grado = 0 de dimension n
    i = 0
    while i < largo(g.relaciones) {
        eje := g.relaciones[i]
        grado[eje.j] := grado[eje.j] + 1
    }
    while i < largo(g.relaciones) {
        eje := g.relaciones[i]
        if eje.i != eje.j {
            B.set(eje.i, eje.j, -p / grado[eje.j])
        }
    }
}
```

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 $r << n^2$, 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 matríz $\mathbf{A}$ 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.

```
proc eliminacion_gaussiana(inout A: matriz<n, n>, inout b: vector<n>) {
    i := 0
    while i < A.n - 1 {
        jt := iterador(A, i + 1, i)
        while jt.en_rango() {
            if (jt.j == i) { // solo si B.at(i + 1, i) != 0
                mij := jt.at() / A.at(i, i)
                sumar_fila(A, b, i, jt.i, mij)
            }
            jt.proximo_columna()
        }
        i := i + 1
    }
}
```

Algoritmo 5. Pseudocódigo para *eliminacion_gaussiana*.

```
proc sumar_fila(inout A: matriz<n, n>, inout b: vector<n>,
                in f1, f2: natural, in c: real) {
    it := iterador(A, f1, f1)
    while it.en_rango() {
        val := B.at(f2, it.j) - c * it.at()
        B.set(f2, it.j, val)
        it.proximo_fila()
    }
    b[f2] := b[f2] - mij * b[f1]
}
```

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 matríz[9]. 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 $\Theta(n^3)$).

[9]Se estima k^2 por SUMAR_FILA, que itera sobre los elementos no nulos de una fila e incluye una operación de SET.

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 algorítmo en $\Theta(n \cdot k)$, con k la cantidad de elementos no nulos de la matríz:

```
proc sustitucion_inversa(in A: matriz<n, n>, in b: vector<n>,
                         out x: vector<n>) {
    x := vector(A.n, 0) // x = 0 de dimension n
    i := A.n - 1
    while i >= 0 {
        s  := 0
        it := iterador(A, i, i + 1)
        while it.en_rango() {
            s := s + it.at() * x[it.j]
            it.proximo_fila()
        }
        x[i] := (b[i] - s) / A.at(i, i)
        i := i + 1
    }
}
```

Algoritmo 7. Pseudocódigo para *sustitucion_inversa*.

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

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

```
proc normalizar(inout x: vector<n>) {
    s := 0
    i := 0
    while i < largo(x) {
        s += x[i]
        i := i + 1
    }
    i := 0
    while i < largo(x) {
        x[i] = x[i] / s
        i := i + 1
    }
}
```

Algoritmo 8. Pseudocódigo para *normalizar*.

3. Resultados y Discusión

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

3.1.1. *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[10].

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

Cada caso, representable por una matriz de conectividad $\mathrm{W} \in \{0,\ 1\}^{100\times100}$, se generó a través del siguiente procedimiento[11]:

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 carácteristicas de ralidad esperables en un conjunto de páginas web.
2) Se pobló una matriz $W_0 \in \{0,\ 1\}^{99\times100} = 0$ 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 matríz $W \in \{0, 1\}^{100\times100}$.

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 expectro 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.

[10] El script asociado se puede encontrar en *./experimentos/error_relativo.py*

[11] Se utilizó un valor semilla para facilitar la reproducibilidad.

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 $1e-5$.

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

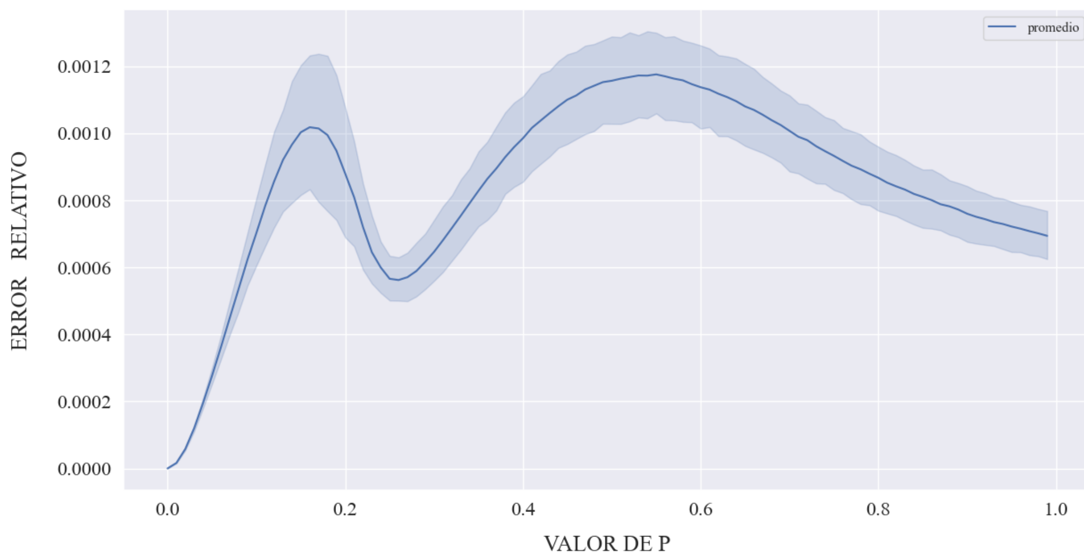

FIGURA 1. Error relativo promedio -en base a una norma L1- e intervalo de confianza del 95% para una muestra aleatoria de cien grafos en función de p.

Notar la progresiva disminución del error a medida que p se aproxima a 1. Los máximos locales se encuentran en $p = 0.16$ (0.00101828) y $p = 0.55$ (0.00117609). El mínimo local en $p = 0.26$ (0.00056232)[12].

Si bien el tamaño de la muestra y su método de generación no permiten sacar conclusiones fuertes al respecto de los resultados, sí podemos notar que el valor de p influencia el error relativo. Desde un punto de vista numérico esto tiene sentido, dado que p reduce la magnitud de los valores sobre los que trabaja el algoritmo. Sin embargo, la forma particular de la distribución del error en función de p resultó sorpresiva.

A modo de consideración para futuros análisis, podemos mencionar que nuestro método de 'corte' —cualquier valor menor a $1e-4$ se anula— también debe influir en el error de los resultados.

[12]El cálculo de los picos locales se puede encontrar en *./experimentos/resultados/error_relativo*

3.1.2. *Error absoluto.* Medimos el error $|x - \hat{x}|_1$ para los casos de test provistos por la cátedra[13].

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.

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.

[13]El script asociado se puede encontrar en *./experimentos/error_tests.py*. Los resultados coordenada a coordenada se pueden observar en *./experimentos/resultados/error_tests*.

3.2. **Análisis cualitativo.** Para entender en profundidad las propiedades de *PageRank*, decidimos evaluar distintos escenarios 'simples' de grafos. Comenzamos por los casos triviales.

3.2.1. *Escenario 1: Sin Links.* Ninguna página tiene un vínculo a ninguna otra. Este es el caso base que el *modelo del navegante aleatorio* busca resolver respecto a modelos anteriores.

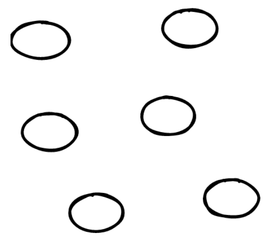

FIGURA 2. Una web con seis páginas desvinculadas.

METODOLOGÍA. Se generaron cien conjuntos de páginas web sin links donde se varió la cantidad total de sitios (de uno a cien). Se midió el puntaje de cada uno en los rankings obtenidos.

RESULTADOS. Observamos que el puntaje de todas las páginas fue igual en cada caso, pero fue decrementando en función de la cantidad total de nodos.

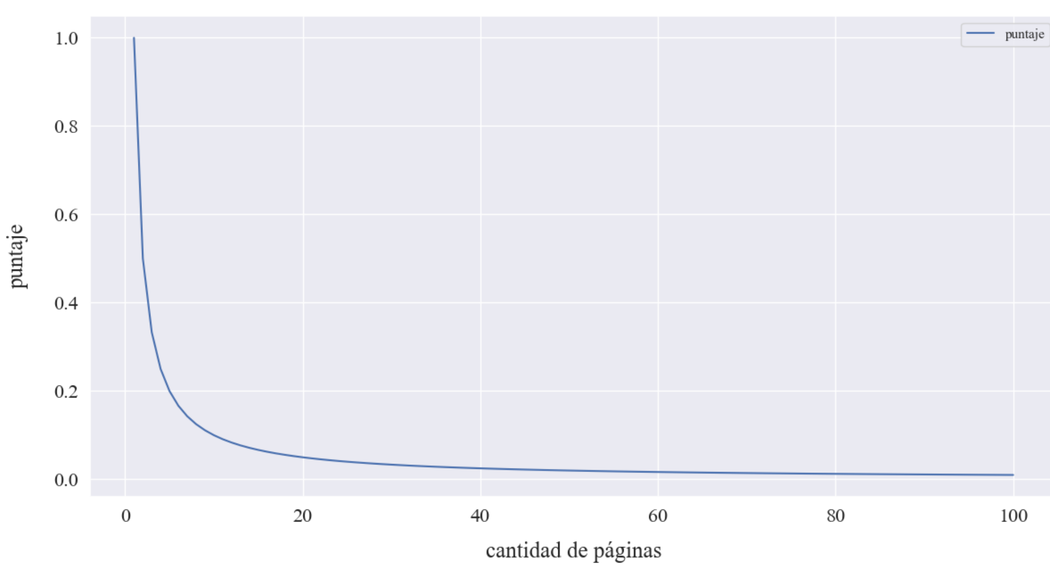

FIGURA 3. El puntaje de una página testigo en función de la cantidad de páginas para conjuntos sin links.

El puntaje obtenido para cada página fue de $\frac{1}{n}$. La figura 3. grafica los resultados para un caso testigo. Esto se explica teóricamente: si no hay links en la web, entonces la matriz de conectividad $\mathbf{W}$ es nula y $\mathbf{I} - p\mathbf{W}\mathbf{D} = \mathbf{I}$. Como $\mathbf{I}x = 1 \implies x_i = 1$. Al normalizar nos queda que $x_i = \frac{1}{n}$ $\forall i : 1 \ldots n$.

El resultado es coherente con la interpretación intuitiva del ranking: si ninguna página redirige a ninguna otra, entonces un navegante solo podrá acceder a los sitios del conjunto de manera aleatoria. El modelo define esta probabilidad de manera uniforme sobre el total de las páginas. A mayor cantidad de páginas, menor la probabilidad que el navegante aleatorio termine en un sitio en particular.

3.2.2. *Escenario 2: Simetría.* Otro escenario trivial es en el que todas las paginas tienen links a todas las demás.

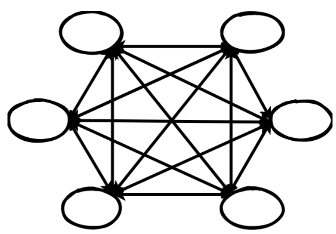

Figura 4. Una web simétrica dónde cada nodo apunta al resto.

Metodología. Para analizar esta situación, se generaron cien casos distintos donde se varió la cantidad de nodos (entre uno y cien) y se mantuvo la simetría de las relaciones. En el primero caso, se compuso una web con un único nodo. Luego, con dos nodos apuntándose mutuamente y así progresivamente hasta llegar a cien nodos donde cada uno apuntó al resto.

Resultados. Observamos que el puntaje de cada una de las páginas fue igual para cada caso particular[14] pero que este valor fue decrementando a medida que aumentó la cantidad de sitios. Luego de observar el resultado notamos que para cada n el mismo fue equivalente al escenario ”sin links”.

Esto tiene sentido ya que si todos los nodos se apuntan entre sí, el sistema es totalmente simétrico, y sería ilógico que alguno tenga más importancia que los demás.

Pudimos notar la misma clase de relación con otras estructuras simétricas. Por ejemplo, en estructuras de referencias circulares[15].

3.2.3. *Escenario 3: Todos con uno.* Para esta estructura, tal y como su nombre indica, todas las páginas tendrán un enlace con una única página. Para referir a esta página apuntada, se la referirá durante esta subsección como *’uno’*. Además, cualquier otro tipo de relación será nula. Los enlaces que unen al ’uno’ serán únicamente enlaces de ’ida’, por lo que, si bien el *’uno’* recibirá n enlaces, este no tendrá ninguno señalando a otro.

[14]Se puede ver el resultado de los experimentos en $./experimentos/resultados/todos\_con\_todos$.

[15]Ver $./experimentos/resultados/circular$

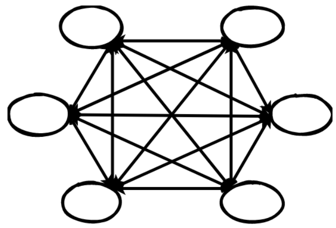

Figura 5. Una web centralizada, donde todos señalan a la misma página.

Metodología. Para analizar esta situación, se generaron cien casos distintos donde se varió la cantidad de enlaces que conectaban con el *'uno'*. De esta forma, veremos la evolución de una web a medida que el *'uno'* va recibiendo todos los enlaces.

Hipótesis. Como suposición para esta estructura tenemos la idea de que se comportará de la siguiente forma:

- La *'uno'* tendrá un ranking muy cercano a 1, que aumentará de tamaño de manera proporcional a la cantidad de páginas existentes. Estimamos que el crecimiento adopte una curva similar a una logarítmica.
- Las demás páginas tendrán un ranking muy cercano a 0, que disminuirá de manera según el número de páginas existentes. Estimamos que el decrecimiento adopte una curva similar a una '$\frac{1}{n}$'.

Resultados. Observamos que los resultados obtenidos distan un poco de los especulados[16], ya que, si bien el decrecimiento del resto de páginas fue de la forma que se suponía, el comportamiento del *'uno'* no se comportó de manera logarítmica, sino que parece tener una forma algo distinta. Conversando sobre esta curva concluímos en que simila a alguna variante de $\sqrt{n}$. Si bien este podría ser uno de los comportamientos esperados, no es por el que nos habíamos inclinado en primer lugar, por lo que llamó nuestra atención. Pudimos notar la misma clase de relación con otras estructuras simétricas. Por ejemplo, en estructuras de referencias circulares a la hora de hablar de las demás páginas.[17].

3.2.4. *Escenario 4: Uno con todos.* Para esta estructura, veremos el caso inverso de la estructura anterior. En esta ocasión, una única página, que será referenciada como *'uno'*, tendrá enlaces salientes hacia todas las demás. Además, cualquier otro tipo de relación será nula. Los enlaces de *'uno'* serán únicamente salientes, por lo que no recibirá ningún enlace.

Metodología. Para analizar esta situación, se generaron cien casos distintos donde se varió la cantidad de enlaces que salían del *'uno'*. De esta forma, veremos la evolución de una web a medida que el *'uno'* va conectando con las demás páginas. Para evaluar los resultados de la evolución, graficamos las variaciones de 3 páginas. La *'uno'*, la primer página con la que conecta y la última con la que lo hace. Así, podremos ver la evolución desde ambas perspectivas.

[16]Se puede ver el resultado de los experimentos en *./experimentos/resultados/todos_con_todos*.

[17]Ver *./experimentos/resultados/circular*

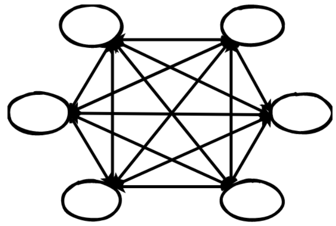

Figura 6. Una web muy simétrica, donde todos, excepto *'uno'* poseen el ranking máximo.

Hipótesis. Como suposición para esta estructura tenemos la idea de que se comportará de la siguiente forma:

- La *'uno'* tendrá un ranking significativamente menor que todas las demás páginas, disminuyendo a medida que aumenta el número de páginas. Estimamos que el decrecimiento adopte una curva similar al $\frac{1}{n}$.
- Las demás páginas irán disminuyendo su ranking hasta cierto punto, donde alcanzarán un máximo, cuando *'uno'* las conecte. Luego, seguirán disminuyendo su ranking hasta que todas posean el mismo, excepto *'uno'*.

Resultados. Observamos que los resultados difieren en gran manera de lo especulado para lo que es el *'uno'*[18], la disminución de su ranking fue abrupta ni bien se conecta el primer enlace. Por otro lado, las suposiciones hechas para los demás son tal cual se esperaban.

3.2.5. *Escenario 5: Transitividad en Cadena.* Cómo se transfiere el puntaje de un sitio a otro a medida que se extiende la cadena de páginas intermedias entre ambos.

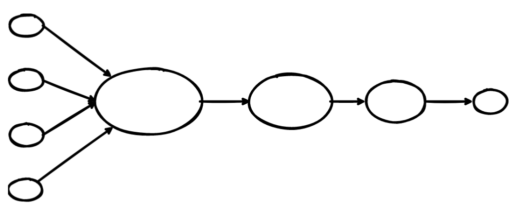

Figura 7. Una cadena de nodos.

Consideramos de interés ver cómo se ve afectado el puntaje de una página al ser apuntada por otra 'importante', y cómo varía a medida que empiezan a haber más intermediarios en la relación.

Metodología. Se generaron cien instancias de test con 200 nodos cada una —para controlar las variaciones del puntaje relacionadas a la inserción de nuevos nodos[19]— y se estableció que las páginas indexadas en el rango (100, 200] apunten a la página 1 y que el resto esté desvinculada.

[18] Se puede ver el resultado de los experimentos en *./experimentos/resultados/todos_con_todos*.
[19] Como se comprobó sucede en el Escenario 1.

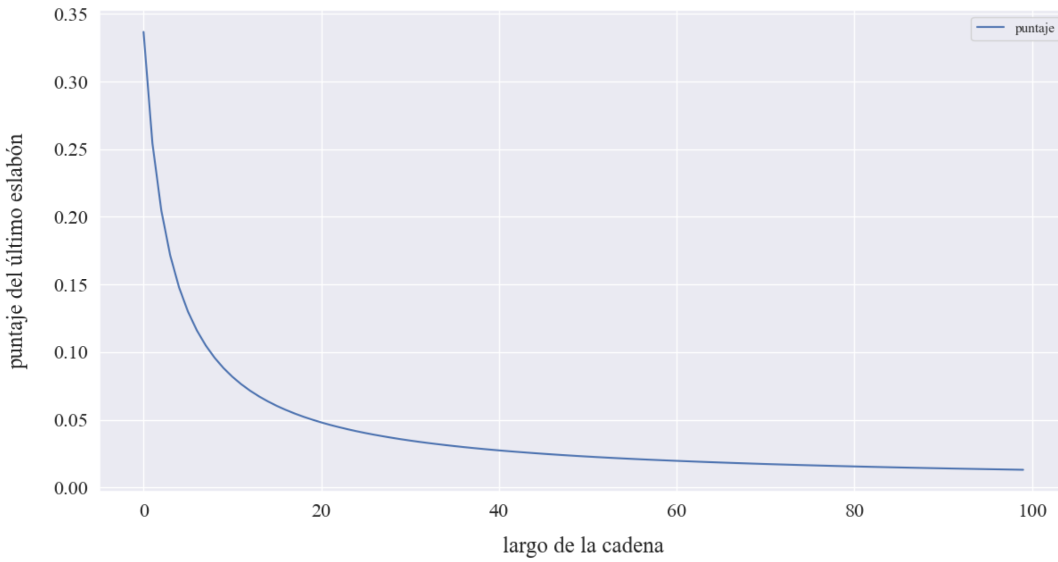

Figura 8. El puntaje del último eslabón en una cadena de vínculos, en función de su largo. Salvo la primer página, el resto de las páginas en la cadena no tiene otras relaciones.

Se procedió a generar una cadena cuyo tamaño se incrementó en uno en cada instancia sucesiva del test: en la primera, el nodo '1' apuntó al nodo '2'; en la segunda, se extendió la cadena y el nodo '2' apuntó al nodo '3'; se procedió de igual forma hasta llegar al último nodo desvinculado. En cada instancia, se midió el puntaje del último nodo en la cadena.

Resultados. Como se ve en la Figura 8. el puntaje del último eslabón se hace más chico a medida que la cadena se hace más larga. De igual manera, el resultado se acerca progresivamente a $1/l$, donde l es el largo de la cadena.

Podemos ver que la influencia de la página 'importante' es fuerte al principio, pero se diluye rápidamente en la cadena. Consideramos que el comportamiento es deseable: Si una página importante apunta a otra, entonces es esperable que aumente la probabilidad que un navegante decida acceder a éste segundo sitio. Pero si agregamos un tercer eslabón en la cadena, entonces la probabilidad que se acceda a ésta tercera página debería ser la intersección de las probabilidades anteriores, y en consecuencia más improbable.

3.2.6. *Escenario 6: Referencias Valiosas.* Cómo varía el peso de una página a medida que cambia la importancia de las páginas que la referencian.

Nuestra hipótesis es que inicialmente el peso del nodo '1' va a ser mayor que el del nodo '0', basándonos en que '0' no tiene páginas que la referencien y '1' tiene una referencia por ende va a ser mayor. Pero a medida que más y más nodos apunten a '0' el peso de éste va a superar al del nodo '1'. Sosteniéndonos en los resultados que obtuvimos a partir de

"Transitividad en Cadena" donde se observa que la importancia de un nodo se transmite a los nodos que apunta, consideramos que el peso de ambos va a ser creciente.

Metodología. La manera en la que planteamos esta experimentación fue, generar 100 instancias de 102 nodos cada uno. Inicialmente el nodo '0' apunta al nodo '1' y todos los demás nodos permanecían aislados. En cada iteración del test se seleccionó uno de los nodos aislados y se agregó una referencia de este último al nodo '0', siguiendo esta dinámica hasta que no haya mas nodos aislados. En cada una de estas iteraciones almacenamos por un lado el peso de '0' y por otro el peso de '1'.

Resultados.

El resultado que obtuvimos cumplía parcialmente con lo esperado pero cambio nuestra percepción de PageRank, ya que inicialmente si se cumplió que el nodo '1' era mayor que '0' y que ambos pesos crecen a medida que '0' toma relevancia. Este comportamiento queda claramente plasmado en el gráfico donde se puede apreciar que el peso de ambos es creciente. Por otro lado, lo que no se cumplió y nos tomó por sorpresa, fue que el peso de '0' nunca superó al peso de '1', como se observa en el gráfico la curva que representa a '1' siempre esta por encima de la curva que representa el peso de '0'.

Concluimos entonces que tener referencias valiosas es esencial para aumentar el peso en el ranking y que la influencia que tienen las páginas de mayor importancia puede generar cambios radicales en la valoración de PageRank.

3.2.7. *Escenario 7: Referenciador Importante.* Cómo varía el peso de un nodo importante a medida que referencia a más nodos.

Metodología. Planteamos un test donde había 20 nodos. Inicialmente todos los nodos apuntan a '0' y en cada iteración se agrega un vínculo de '0' a alguno de los nodos. En cada una de estas iteraciones almacenamos el peso de '0'.

Resultados. El resultado no fue el esperado, ya que como se ve en el gráfico la curva de color xxx, el peso de '0' de no referenciar a nadie a referenciar a un nodo baja pero luego se mantiene constante, como si el impacto que genera en el peso de un nodo particular referenciar a una cantidad arbitraria de nodos fuera comparable con referenciar a un único nodo.

Pensamos que quizás en un ambiente más 'real' con relaciones aleatorias entre los demás nodos del sistema podría alterar nuestros resultados, por ello reestructuramos el test para que además de las relaciones previamente mencionadas los nodos [1 a 20] estén conectados entre sí de manera aleatoria. Para obtener resultados significativos que no esten 'contaminados' por la aleatoreidad corrimos el test 10 veces, sumamos el resultado de cada una de las iteraciones y calculamos su promedio. El resultado obtenido fue similar al resultado previo por ende deducimos que no referenciar a ningún nodo es la estrategia óptima y que a partir de referenciar a uno o más nodos en un ambiente aleatorio genera, en promedio, el mismo impacto en el peso de la la página que referencia.

4. Conclusiones

5. Apéndice

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

demostración. Recordemos que:

$$e_i = 1$$

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

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

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

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:

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

Como p es un escalar, sigue entonces que:

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

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:

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

Por lo que:

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

$$= \begin{cases} (1-p) \cdot \frac{1}{n} + p \cdot \frac{w_{ij}}{c_j} & \text{si} \quad c_j \neq 0 \\ \frac{1}{n} & \text{si no} \end{cases}$$

pero:

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

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 concluímos que $a_{ij} = (p\mathbf{WD} + ez^t)_{ij}$, $\forall i, j :\ 1 \ldots n$, 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:

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

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

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

$$CB : (k = 1)$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 saltear 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.