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.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 $x = (x_1,\ \ldots\ ,\ x_n)^t$. Definimos entonces:

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

y proponemos que[1]:

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

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

```
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: x[i] >= 0 para toda i y sum(x) = 1  */
9 }
```

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 $1 \times n$.

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

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

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

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]:

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

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

[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:

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

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:

```
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*.

[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 2.2.

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:

```
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 $r << n^2$, 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 **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 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:

```
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 algóritmo 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[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.

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 $1e-5$.

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

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

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

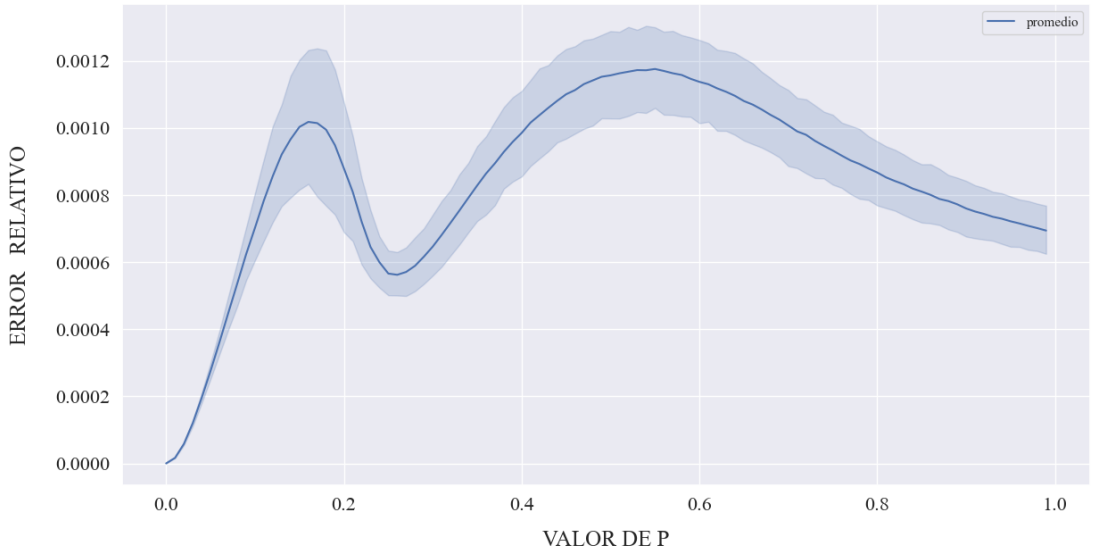

FIGURA 1. Error relativo promedio -en base a una norma L1- e intervalo de confianza del 95% para una muestra aleatoria de cien grafos en función de p.

Notar la progresiva disminución del error a medida que p se aproxima a 1. Los máximos locales se encuentran en $p = 0.16$ (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 extensión 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.

2.2.2. *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.

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

[13]Los resultados coordenada a coordenada se pueden observar en *./experimentos/resultados/error_tests*.

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.

2.3. **Análisis cualitativo.**

3. Resultados y Discusión

4. Conclusiones

5. Apéndice

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

demostración. Recordemos que:

$$e_i = 1$$

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

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

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

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

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

■

5.2. **B: $I - pWD$ permite la eliminación gaussiana.**

Demostración. Demos primero una definición formal del enunciado:

Eliminación Gaussiana. $\forall \mathbf{A} \in \mathbb{R}^{n\times n},\ \exists!\ m \in \mathbb{N} :\ 1 \leq m \leq n$ tal que:

$\forall k \in \mathbb{N} :\ 1 \leq k < m$

$$eg_1(\mathbf{A})_{ij} = a_{ij}$$
$$eg_{k+1}(\mathbf{A})_{ij} = \begin{cases} eg_k(\mathbf{A})_{ij} & \text{si} \quad i < k+1 \ \vee \ j < k+1 \\ eg_k(\mathbf{A})_{ij} - \frac{eg_k(\mathbf{A})_{ik}}{eg_k(\mathbf{A})_{kk}} \cdot \ eg_k(\mathbf{A})_{kj} & \text{si no} \end{cases}$$

donde $m < n$ es el mínimo valor que satisface que $eg_m(\mathbf{A})_{mm} = 0$. Si la matríz $\mathbf{A}$ es tal que $m = n$, entonces $\mathbf{A}$ permite la eliminación gaussiana.

Probaremos que para la matríz $\mathbf{B} \in \mathbb{R}^{n\times n} = \mathbf{I} - p\mathbf{WD}\ ,\ \forall k \in \mathbb{N} : 1 \leq k \leq n$:

$$eg_k(\mathbf{B})_{kk} \neq 0$$

y en consecuencia $m = n$.

Para ello demostraremos el siguiente enunciado:

$$|b^k_{jj}| \ > \sum_{i=k,\ i\neq j}^{n} |b^k_{ij}| \qquad \forall k,\ j \in \mathbb{N} : 1 \leq k \leq j \leq n$$

donde $b^k_{ij} = eg_k(\mathbf{B})_{ij}$.

Es decir, para todo paso k de la eliminación gaussiana se satisface que el valor absoluto de los elementos de la diagonal mayores o iguales a k son mayor estricto al valor absoluto de la suma de los elementos de su columna desde la posición k, descontando la diagonal.

Lo demostraremos por inducción:

$\forall k,\ j \in \mathbb{N} : 1 \leq k \leq j \leq n$

$$|b^k_{jj}| \ > \sum_{i=k,\ i\neq j}^{n} |b^k_{ij}| \implies |b^{k+1}_{jj}| \ > \sum_{i=k+1,\ i\neq j}^{n} |b^{k+1}_{ij}|$$

Caso base ($k = 1$):

Notemos primero que

$$(\mathbf{B})_{ij} = (\mathbf{I} - p\mathbf{WD})_{ij} = \begin{cases} 1 & \text{si} \quad i = j \\ -p/c_j & \text{si} \quad w_{ij} = 1 \ \wedge \ c_j \neq 0 \\ 0 & \text{si no} \end{cases}$$

Dado que todo elemento de cada columna j de $\mathbf{WD}$ está normalizado por $1/c_j$, tal que su suma da uno, entonces la suma de la columna $(-p\mathbf{WD})_j = -p$, con $0 < p < 1$.

Observamos también que la diagonal es nula para esta matríz, tal que su suma por $\mathbf{I}$ da $(\mathbf{I} - p\mathbf{WD})_{jj} = 1$. Este tipo de matríz se conoce como estocástica en columnas y su transpuesta es diagonal dominante.

En consecuencia, es inmediato que $|b^1_{jj}| = |b_{jj}| = 1 \ > \ \sum^n_{i=k,\ i\neq j} |b^1_{ij}| = |-p| \ \forall j \in \mathbb{N} : 1 \leq j \leq n$.

□

Paso inductivo:

$\forall j \in \mathbb{N} : 1 \leq k \leq j \leq n$

$$|b^{k+1}_{jj}| \ > \ \sum_{i=k+1,\ i\neq j}^{n} |b^{k+1}_{ij}|$$

$$\iff |b^k_{jj} - \frac{b^k_{jk} \ \cdot \ b^k_{kj}}{b^k_{kk}}| \ > \ \sum_{i=k+1,\ i\neq j}^{n} |b^k_{ij} - \frac{b^k_{ik} \ \cdot \ b^k_{kj}}{b^k_{kk}}|$$

dado que $|a-b| \geq |a|-|b| \ \wedge \ |a|+|b| \geq |a+b|$ por desigualdad triangular, entonces podemos demostrar equivalentemente, por teorema del sandwich, que:

$$|b^k_{jj}| - |\frac{b^k_{jk} \ \cdot \ b^k_{kj}}{b^k_{kk}}| \ > \ \sum_{i=k+1,\ i\neq j}^{n} |b^k_{ij}| \quad + \sum_{i=k+1,\ i\neq j}^{n} |\frac{b^k_{ik} \ \cdot \ b^k_{kj}}{b^k_{kk}}|$$

Como:

$$\sum_{i=k+1,\ i\neq j}^{n} |b^k_{ij}| \ = \ (\sum_{i=k,\ i\neq j}^{n} |b^k_{ij}|) - |b^k_{kj}|$$

y por hipótesis inductiva:

$$|b^k_{jj}| \ > \ \sum_{i=k,\ i\neq j}^{n} |b^k_{ij}| \implies |b^k_{jj}| - |b^k_{kj}| \ > \ (\sum_{i=k,\ i\neq j}^{n} |b^k_{ij}|) - |b^k_{kj}|$$

entonces podemos volver a acotar tal que:

$$|b_{jj}^k| - |b_{kj}^k| \ + \sum_{i=k+1,\ i\neq j}^{n} |\frac{b_{ik}^k \ \cdot \ b_{kj}^k}{b_{kk}^k}| \ >_{hi} \sum_{i=k+1,\ i\neq j}^{n} |b_{ij}^k| \ + \sum_{i=k+1,\ i\neq j}^{n} |\frac{b_{ik}^k \ \cdot \ b_{kj}^k}{b_{kk}^k}|$$

y considerar que:

$$|b_{jj}^k| - |\frac{b_{jk}^k \ \cdot \ b_{kj}^k}{b_{kk}^k}| \ \geq \ |b_{jj}^k| - |b_{kj}^k| + \sum_{i=k+1,\ i\neq j}^{n} |\frac{b_{ik}^k \ \cdot \ b_{kj}^k}{b_{kk}^k}|$$

$$\iff |b_{kj}^k| - |\frac{b_{jk}^k \ \cdot \ b_{kj}^k}{b_{kk}^k}| \ \geq \ \frac{|b_{kj}^k|}{|b_{kk}^k|} \sum_{i=k+1,\ i\neq j}^{N} |b_{ik}^k|$$

Podemos acotar una última vez, tal que:

$$|b_{kj}^k| - |\frac{b_{jk}^k \ \cdot \ b_{kj}^k}{b_{kk}^k}| \ \geq \ \frac{|b_{kj}^k|}{|b_{kk}^k|}(|b_{kk}^k| - |b_{jk}^k|) \ >_{hi} \ \frac{|b_{kj}^k|}{|b_{kk}^k|} \sum_{i=k+1,\ i\neq j}^{N} |b_{ik}^k|$$

Pero:

$$|b_{kj}^k| - |\frac{b_{jk}^k \ \cdot \ b_{kj}^k}{b_{kk}^k}| \ \geq \ \frac{|b_{kj}^k|}{|b_{kk}^k|}(|b_{kk}^k| - |b_{jk}^k|) \iff |b_{kj}^k| - |\frac{b_{jk}^k \ \cdot \ b_{kj}^k}{b_{kk}^k}| \geq |b_{kj}^k| - |\frac{b_{jk}^k \ \cdot \ b_{kj}^k}{b_{kk}^k}|$$

Lo que es trivialmente cierto.

□

Demostramos ahora por absurdo que $b_{kk}^k \neq 0 \ \ \forall k \in \mathbb{N} : 1 \leq k \leq n$.

Supongamos que existe algún k tal que $b_{kk}^k = 0$. Entonces:

$$|b_{kk}^k| \ = \ 0 \ > \sum_{i=k,\ i\neq k}^{n} |b_{ij}^k| \qquad \forall j \in \mathbb{N} : 1 \leq k \leq j \leq n$$

Pero $\sum_{i=k,\ i\neq k}^{n} |b_{ij}^k|$ es necesariamente una suma de valores positivos ó 0. Es decir:

$$0 \ > \sum_{i=k,\ i\neq k}^{n} |b_{ij}^k| \ \geq \ 0 \implies \ 0 > 0$$

Absurdo!

■

5.3. **C: Evaluación de estructuras.** A la hora de decidir cómo representar las matrices se evaluaron las siguientes alternativas:

- Como estructura 'base', utilizar un vector de n vectores de tamaño fijo n. VENTAJAS: se puede acceder y modificar los elementos en $\Theta(1)$. DESVENTAJAS: requiere iterar sobre todos los elementos de la matríz, es ineficiente en espacio y —para matrices muy grandes— probablemente requiera más accesos a disco.

- Una estructura 'alt' con un vector de n vectores de tamaño variable donde en cada lugar se guarda la tupla $< posicion, elemento >$. VENTAJAS: es más eficiente en espacio, permite iterar sobre los elementos no nulos de cada fila y acceder a estas en $\Theta(1)$. DESVENTAJAS El acceso a cada columna —con búsqueda binaria— tiene un peor caso en $\Theta(log(n))$ y la inserción tiene un peor caso en $\Theta(n)$.

- La misma estructura pero con listas enlazadas. VENTAJAS: permite la inserción en $\Theta(1)$ e iterar sobre los elementos no nulos de cada fila con la misma complejidad. DESVENTAJAS el acceso es en $\Theta(n)$ y no mantiene memoria contigua, por lo que hay mayor probabilidad de cache-miss.

- Una 'matriz enlazada' donde cada nodo apunta al próximo elemento no nulo de la misma fila y columna. VENTAJAS: permite iterar sobre los elementos no nulos de cada fila y columna en $\Theta(1)$. DESVENTAJAS: las mismas que la lista enlazada.

- Utilizar alguna estructura de diccionario como $< map >$. Las ventajas y desventajas son similares a la de la estructura alternativa si se mantienen las claves en orden y se pueda iterar sobre ellas en $\Theta(1)$.

Todas estas posibilidades motivaron la elaboración de un código genérico que permitiera el intercambio de las representaciones. La clase $matriz < R >$ definida en $./implementacion/$ provee una interfáz de matríz completa junto a la implementación de muchos de sus métodos.

Una representación R deberá definir sólo aquellas operaciones que conciernen el acceso a datos. Para lograr esta modularización se tomaron dos decisiones importantes:

- Se optó por la transparencia referencial de todas las operaciones. Para las representaciones contempladas, generar una nueva instancia es más rápido que modificar la estructura actual.
- Todos los algoritmos están implementados para evitar las operaciones redundantes. Par ello, se optó por que cada representación implemente iteradores específicos.

Estas decisiones tienen una clara desventaja: el código provisto por $matriz < r >$ podría ser optimizado para cada representación subyacente. Sin embargo, se priorizó la facilidad de la experimentación sobre la optimalidad del código.

Siguiendo esta metodología, se implementaron dos de las alternativas propuestas: 'base' y 'alt'.

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.