DEPARTAMENTO DE COMPUTACION

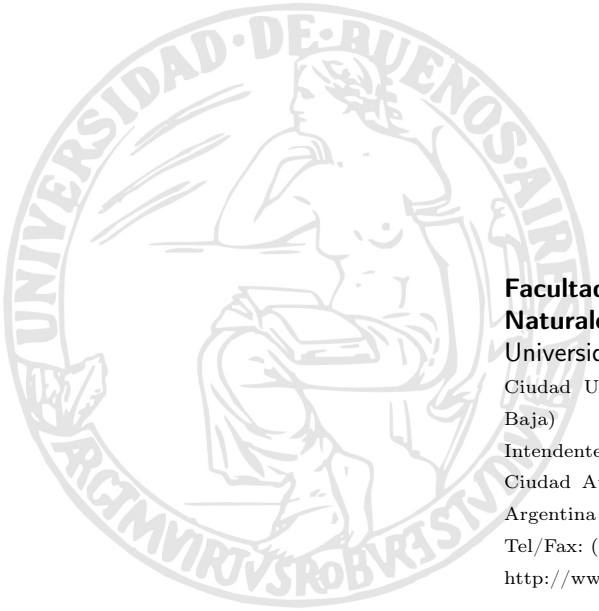
Facultad de Ciencias Exactas y Naturales - UBA

TP Métodos Númericos

Page Rank

6 de Septiembre de 2018 — Métodos Numericos

Integrante	LU	Correo electrónico
Carreira Munich, Tobías Agustín	278/17	tcarreira@dc.uba.ar
Martino, Maximiliano	123/17	maxii.martino@gmail.com
Nahmod, Santiago Javier	016/17	snahmod@dc.uba.ar
Torres, Edén	017/17	etorres@dc.uba.ar



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

Índice

1. Resumen

Este trabajo se centra en el estudio del algoritmo PageRank, propuesto en 1998 e implementado por el reconocido motor de búsqueda de Google, utilizado para clasificar sitios web por orden de relevancia. De acuerdo a Google:

> PageRank funciona contando el número y la calidad de los enlaces a una página web para determinar una estimación aproximada de la importancia de la misma. La suposición subyacente es que es probable que los sitios web más importantes reciban más enlaces de otros sitios web.

En el presente trabajo, se encontrarán los detalles de una posible implementación, incluyendo la elección de una estructura de datos apropiada para trabajar eficientemente con una gran cantidad de datos, bajo la asunción de ralidad.
También se llevará a cabo un exhaustivo análisis de la *performance* del algoritmo, teniendo en cuenta la calidad de los resultados que produce.
Además se lo utilizará para modelar distintos escenarios considerados de interés cualitativo, y posteriormente se analizará lo apropiado de los resultados.

2. Introducción

Dado un conjunto de páginas web, el objetivo de PageRank es establecer un orden de importancia o ranking entre ellas. Si pensamos en la estructura de la *internet*, resulta natural representarla mediante un grafo, cuyos nodos serán las páginas de la web y las aristas los enlaces (links) entre las mismas. Como veremos más adelante, la información contenida en dicho grafo posee un enorme potencial a la hora de establecer un orden de importancia entre sus nodos. Será de interés asignar a cada página un puntaje, basado en la cantidad de enlaces que apunten a ella y el peso de los mismos.

2.1. Matriz de conectividad

Esta información puede representarse mediante una **matriz de conectividad**, que llamamos W:

$$w_{ij} = \begin{cases} 0 & \text{si } i = j \text{ (no tenemos en cuenta los autolinks)} \\ 1 & \text{si } i \neq j \text{ y la página } j \text{ posee un link hacia la página } i \\ 0 & \text{si } i \neq j \text{ y la página } j \text{ no posee un link a la página } i \end{cases}$$

Se define el grado de cada página c_j como la cantidad de links salientes de la misma.

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

Se evaluará la *importancia* de úna página como la cantidada de links ponderados que reciba. De esta manera, una página que reciba pocos links de otras páginas *importantes* puede llegar a ser mas importante que otra que reciba muchos de páginas comunes.
Se define la importancia de una página como

$$x_i = \sum_{\substack{j=1 \\ c_j \neq 0}}^{n} \frac{x_j}{c_j} w_{ij}$$

Es importante notar que el puntaje de una página depende del puntaje de las otras.

2.2. Navegante aleatorio

Un modelo que aproxima mejor la realidad es el del **navegante aleatorio**.

Este modelo se basa en una idea similar a la anterior, tomando como criterio la **probabilidad** de saltar de una página hacia otra. De esta manera, dado un conjunto de n páginas, se define la probabilidad de saltar de la página j a la i como a_{ij} tal que:

$$a_{ij} = \begin{cases} \frac{1-p}{n} + p \cdot \frac{w_{ij}}{c_j} & \text{si } c_j \neq 0 \\ \frac{1}{n} & \text{si } c_j = 0 \end{cases}$$

y sea A $\in \mathbb{R}^{nxn}$ a la matriz de elementos a_{ij}. Entonces el *ranking de Page* es la solución del sistema:

$$A.x = x$$

que cumple $x_i \geq 0$ y $\sum_i x_i = 1$. Por lo tanto, el elemento i-ésimo $(Ax)_i$ es la probabilidad de encontrar al navegante aleatorio en la página i sabiendo que x_j es la probabilidad de encontrarlo en la página j, para $j = 1...n$;
Si definimos una matriz diagonal D con los elementos d_{jj} de la forma:

$$d_{ij} = \begin{cases} \frac{1}{c_j} & \text{si } c_j \neq 0 \\ 0 & \text{si } c_j = 0 \end{cases}$$

Un vector columna e de unos de dimensión n y z un vector columna cuyos componentes son:

$$z_j = \begin{cases} \frac{1-p}{n} & \text{si } c_j \neq 0 \\ \frac{1}{n} & \text{si } c_j = 0 \end{cases}$$

Podemos reescribir a la matriz A como:

$$A = p\mathbf{W}\ \mathbf{D} + \mathbf{e}\ \mathbf{z}^T$$

3. Marco teórico

El objetivo de esta sección es darle formalidad a las distintas a afirmaciones y propiedades que utilizamos asi como también una base teórica que fundamenta los métodos involucrados en el trabajo, junto con los métodos mismos.

3.1. Equivalencia de la matriz

Comenzaremos viendo que la matriz A definida anteriormente es equivalente a $p\mathbf{W}\ \mathbf{D}$+$\mathbf{e}\ \mathbf{z}^T$, lo cual puede resultarnos útil a la hora de resolver el sistema $A.x = x$ y de implementar una solción eficaz.

Para ver la igualdad $\mathbf{A} = p\mathbf{W}\ \mathbf{D} + \mathbf{e}\ \mathbf{z}^T$ podemos ver que

$$\mathbf{A}_{ij} = (p\mathbf{W}\ \mathbf{D} + \mathbf{e}\ \mathbf{z}^T)_{ij} \quad \forall (1 \leq i, j \leq n)$$

que es lo mismo, por definición de suma de matrices, que ver que:

$$\mathbf{A}_{ij} = ((p\mathbf{W}\ \mathbf{D})_{ij} + (\mathbf{e}\ \mathbf{z}^T)_{ij}) \quad \forall (1 \leq i, j \leq n)$$

Por definición tenemos que:

$$\mathbf{A}_{ij} = \begin{cases} (pw_{ij})/c_{ij} + (1-p)/n & si \quad c_j \neq 0 \\ 1/n & si \quad c_j = 0 \end{cases}$$

Ahora miremos que pasa con $(p\mathbf{W}\ \mathbf{D})_{ij}$.
Tenemos que p es un escalar, por lo tanto $(p\mathbf{W}\ \mathbf{D})_{ij} = p(\mathbf{W}\ \mathbf{D})_{ij}$

Miremos $(\mathbf{W}\ \mathbf{D})_{ij}$:

Por definición de multiplicación de matrices tenemos que: $(\mathbf{W}\ \mathbf{D})_{ij} = \sum_{k=1}^{n} w_{ik} d_{kj}$
y como D es diagonal de la suma anterior solo va a sobrevivir los términos donde $k = j$ por lo tanto nos queda:

$$(\mathbf{W}\ \mathbf{D})_{ij} = w_{ij} d_{jj}$$

y por definición de d_{jj} tenemos que:

$$(\mathbf{W}\ \mathbf{D})_{ij} = \begin{cases} (w_{ij})/c_{ij} & si \quad c_j \neq 0 \\ 0 & si \quad c_j = 0 \end{cases}$$

donde luego multiplicamos por p:

$$(p\mathbf{W}\ \mathbf{D})_{ij} = p(\mathbf{W}\ \mathbf{D})_{ij} = \begin{cases} (pw_{ij})/c_{ij} & si \quad c_j \neq 0 \\ 0 & si \quad c_j = 0 \end{cases}$$

Ahora miremos $((\mathbf{e}\ \mathbf{z}^t))_{ij}$:
Por definición de multiplicación de matrices:

$$((\mathbf{e}\ \mathbf{z}^t))_{ij} = \sum_{k=1}^{n} e_{ik} (z^t)_{kj}$$

pero sabemos que e es un vector fila y z^t es un vector columna. Por lo tanto:

$$(\mathbf{e}\ \mathbf{z}^t)_{ij} = e_{ij}(z^t)_{jj}$$

Pero $e_i = 1 \quad \forall i$, luego $(\mathbf{e}\ \mathbf{z}^t)_{ij} = (z^t)_j$ y por definición de z_j tenemos que:

$$(\mathbf{e}\ \mathbf{z}^t)_{ij} = (z^t)_j = \begin{cases} (1-p)/n & si \quad c_j \neq 0 \\ 1/n & si \quad c_j = 0 \end{cases}$$

Por último, resumiendo todo, queda que:

$$(p\mathbf{W}\ \mathbf{D}+\mathbf{e}\ \mathbf{z}^t)_{ij} = ((p\mathbf{W}\ \mathbf{D})_{ij}+(\mathbf{e}\ \mathbf{z}^t))_{ij} = \begin{cases} (pw_{ij})/c_{ij} + (1-p)/n & si \quad c_j \neq 0 \\ 1/n & si \quad c_j = 0 \end{cases}$$

Es decir: $\mathbf{A}_{ij} = (p\mathbf{W}\ \mathbf{D} + \mathbf{e}\ \mathbf{z}^t)_{ij}\ \forall (1 \leq i, j \leq n)$ que es lo que queriamos ver.

3.2. Aplicabilidad de Eliminación Gaussiana sin pivoteo

Para resolver el Page Rank, utilizaremos Eliminación Gaussiana (EG) sin pivoteo, para resolver el siguiente sistema lineal:

$$(\mathbf{I} - p\ \mathbf{W}\ \mathbf{D})x = \gamma\ \mathbf{e}$$

donde $\gamma = \mathbf{z}^T\mathbf{x}$ y suponemos $\gamma = 1$.
Para esto, resulta fundamental probar que la forma de la matriz (**I** - p **W D**) garantiza la aplicabilidad de EG sin pivoteo.
Veamos:
Definimos la matriz M de la siguiente forma:

$$(p\mathbf{W}\ \mathbf{D})_{i,j} = \begin{cases} (pw_{ij})/c_j & si \quad c_j \neq 0 \\ 0 & si \quad c_j = 0 \end{cases}$$

$$(M)_{ij} = (\mathbf{I} - p\mathbf{W}\ \mathbf{D})_{i,j} = \begin{cases} (-pw_{ij})/c_j & si \quad i \neq j \quad y \quad c_j \neq 0 \\ 0 & si \quad i \neq j \quad y \quad c_j = 0 \\ 1 & si \quad i = j \end{cases}$$

Afirmacion: La matriz definida arriba es estrictamente diagonal dominante por columnas.

Demostración: Si queremos ver que es estrictamente diagonal dominante por columnas, lo que tenemos que probar es que para la matriz M:

$$\forall j = 1, ..., n \quad |m_{jj}| > \sum_{i=1\ i\neq j}^{n} |m_{ij}|$$

Por la forma en que definimos la matriz M, podemos concluir que los elementos de su diagonal siempre van a ser 1.

Además, cuando $i \neq j$, también se puede reemplazar $|m_{ij}|$ por $\frac{p.w_{ij}}{c_j}$ cuando $c_j \neq 0$ y por 0 cuando $c_j = 0$. De modo que lo que deberíamos probar es que simultáneamenete se cumple:

$$\forall j \in \{1, ..., n\} \ / c_j \neq 0 \quad 1 > \sum_{i=1 \ i \neq j}^{n} \left| \frac{p.w_{ij}}{c_j} \right| \tag{1}$$

$$\forall j \in \{1, ..., n\} \ / c_j = 0 \quad 1 > \sum_{i=1 \ i \neq j}^{n} 0 \tag{2}$$

Claramente la segunda de las sentencias que declaramos es válida, ya que $1 > 0$, por lo que solo resta analizar la primera.
Al remover las constantes de la sumatoria queda:

$$\forall j = 1, ..., n \ / c_j \neq 0 \quad 1 > \frac{p}{c_j} \sum_{i=1 \ i \neq j}^{n} w_{ij}$$

Recordando la definición de c_j:

$$\forall j = 1, ..., n \quad c_j = \sum_{i=1 \ i \neq j}^{n} w_{ij}$$

Observación: Nuestra sumatoria tiene un sumando menos (cuando $i = j$), pero es trivial que por la definición de W ese sumando sería 0, pues W tiene 0 en todo su diagonal.
Simplificando,

$$1 > p$$

Y como $p \in (0, 1)$, esto resulta cierto.
Luego, M es estrictamente diagonal dominante por columnas. Por lo tanto el algoritmo de EG no requiere permutaciones, es decir que la matriz $(I - p\mathbf{W}\ \mathbf{D})$ garantiza la aplicabilidad de EG sin pivoteo. Asimismo, esto nos garantiza estabilidad númerica, pues en cada paso del algoritmo de EG estamos dividiendo por el elemento de mayor tamaño, reduciendo al máximo posible los errores de precisión.

4. Desarollo

Para resolver el sistema lineal anteriormennte propuesto:

$$(\mathbf{I} - p\ \mathbf{W}\ \mathbf{D})x = \gamma\ \mathbf{e}$$

donde $\gamma = \mathbf{z}^T\mathbf{x}$ y suponemos $\gamma = 1$. Utilizaremos el algoritmo de Eliminación Gaussiana sin pivoteo. Para implementarlo, tuvimos en cuenta que la matriz que define el sistema tiene muchos espacios en 0 en casos de la vida real, ya que la cantidad de links que tiene una página web es pequeña en comparación con la cantidad de páginas existentes. Es por esto que implementamos una clase de Matriz Rala o Dispersa, que guarde solamente los elementos distintos a 0 y ahorrar espacio y optimizar las operaciones. Lo guardamos en un vectores de vectores donde cada elemento se guarda en un vector fila y el elemento contiene el valor y la posición en la fila. Además cada elemento de las filas estan ordenados por su posición en la matriz real. También tuvimos en cuenta que la matriz **D** es diagonal, para implementar la multiplicación **W D** de forma óptima.

Para facilitar la busqueda de errores en el código, implementamos primero las operaciones básicas de matriz (multiplicación, resta), creamos tests para estas (utilizado el framework *Google Test*), y luego pasamos a implementar la EG, testeandola con los archivos provistos por la cátedra.

Desde un comienzo, utilizamos *long double* para almacenar los elementos de las matrices de la forma más precisa posible. Posteriormente, probamos si realmente el uso de *long double* prevenía problemas de precisión o no, frente a *double* y *float*.

Dado que se nos pide normalizar el vector **x** de forma que $\sum_i x_i = 1$, dividimos a cada elemento de **x** por la sumatoria de sus elementos. Esto lleva a problemas de precisión, dada la suma de números que potencialmente tendrían distinta magnitud. Es por eso que probamos con distintos métodos:

- Suma normal
- Suma normal, ordenando antes los números de forma ascendente
- Suma con algoritmo de Kahan

Algunas de las preguntas teóricas levantaron interés sobre posibles resultados de la aplicación del Page Rank. En particular, en una de las preguntas teóricas se concluye que el número de condición de la matriz crece de forma condicional, a medida que también aumente p, la probabilidad de elegir un link en la página actual (en vez de elegir otra aleatoriamente) Dado este resultado teórico, y el hecho de que nuestro algoritmo trabaja (inevitablemente) con precisión finita, pusimos a prueba la hipótesis de que a medida que aumenta p, también aumenta el error de nuestro algoritmo.

Pasando a la experimentación cuantitativa, creamos un script en C++ que nos provea de la matriz **A**, aprovechando el código anteriormente implementado. Luego, utilizamos *Octave* para ver el error producido al aproximar *x'* obtenida ($|Ax' - x'| \approx 0$). Para ver el error absoluto, volvimos a usar *Octave* para ver la distancia entre nuestras soluciones y las dadas por la cátedra.

Para la experimentación cualitativa, primero pensamos casos no triviales de conjuntos de páginas y sus links que nos parecian de interés (estos casos seran

descriptos extensamente en Resultados) Los graficamos usando una herramienta web. Luego, generamos los archivos .txt que representan cada uno de los casos, en el formato de entrada con el que trabaja nuestro ejecutable de C++. Definimos experimentos usando estos casos, y para correrlos utilizamos la ayuda de *Octave*. Los resultados de cada experimento se visualizan de diferentes formas, usando gráficos de diferente naturaleza. Esta tarea se logro realizar de forma muy directa, ya que *Octave* permite generar gráficos con facilidad y ejecutar comandos de consola, por lo que pudimos llamar a nuestro ejecutable de C++ de forma dinámica. Esto permitió cumplir con lo definido en los experimentos, ya que algunos requerian correr el algoritmo muchas veces antes de llegar a un resultado.

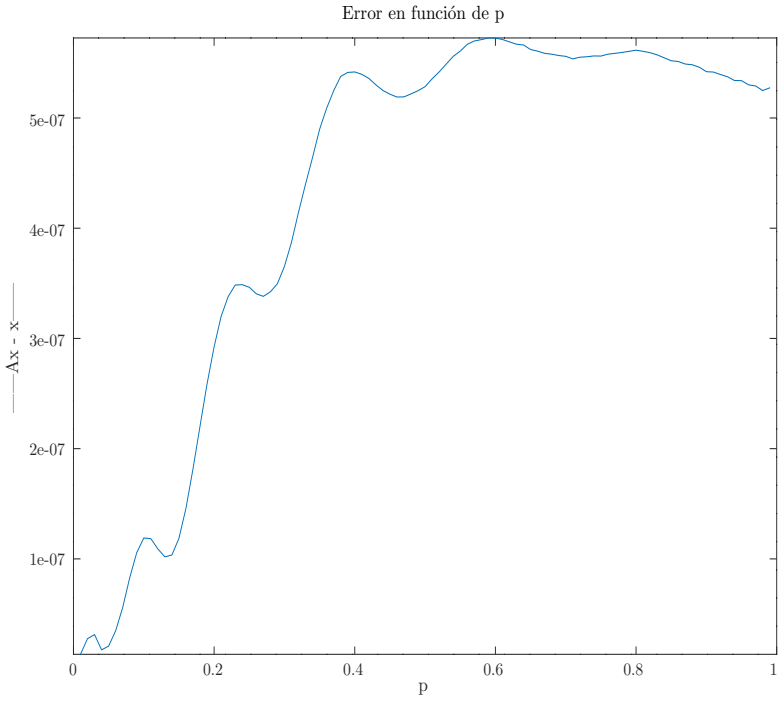
5. Resultados

5.1. Experimentación sobre predicados teóricos

5.1.1. El valor de p y el error del sistema

En la Figura 1 se ejecuta el Page Rank con un archivo de entrada habilitado por la catedra, "Test 15 Segundos", con 100 probabilidades distintas.

Figura 1



5.2. Experimentación cuantitativa

5.2.1. Aproximación de la solución

Calculamos la aproximación de la solución x' obtenida con $|Ax' - x'|$:

Tabla 1

Test	$\|Ax' - x'\|$
trivial	0
sin links	6.2063e-17
completo	5.5511e-17
aleatorio	7.6163e-07
aleatorio desordenado	5.3216e-07
15 segundos	5.4199e-07
30 segundo	7.9482e-07

5.2.2. Error absoluto con respecto a los tests de la cátedra

Calculamos la distancia en norma 2 entre nuestra solución x' y la provista por la cátedra:

Tabla 2

Test	Error
trivial	0
sin links	0
completo	0
aleatorio	0
aleatorio desordenado	0
15 segundos	5.8256e-07
30 segundo	5.8.273e-07

5.3. Experimentación sobre distintas implementaciones

5.3.1. Error con distintos tipos de datos para almacenar punto flotante

Para ver el impacto de usar *float*, *double* o *long double*, calculamos el error al hacer $|Ax' - x'|$ usando cada uno de estos tipos de datos.

Tabla 3

Test	float	double	long double
trivial	0	0	0
sin links	6.2063e-17	6.2063e-17	6.2063e-17
completo	5.5511e-17	5.5511e-17	5.5511e-17
aleatorio	7.6163e-07	7.6163e-07	7.6163e-07
aleatorio desordenado	5.3216e-07	5.3216e-07	5.3216e-07
15 segundos	5.5818e-07	5.4199e-07	5.4199e-07
30 segundo	8.1439e-07	7.9482e-07	7.9482e-07

5.3.2. Error con distintos tipos de sumatoria

Vemos el impacto de sumar normalmente, sumar ordenando ascendentemente los números, o usar el algoritmo de Kahan, calculando el error al hacer $|Ax' - x'|$ usando cada uno de estos algoritmos (siempre almacenando los datos en *long double*)

Tabla 4

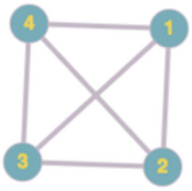
Test	Normal	Ordenando	Kahan
trivial	0	0	0
sin links	6.2063e-17	6.2063e-17	6.2063e-17
completo	5.5511e-17	5.5511e-17	5.5511e-17
aleatorio	7.6163e-07	7.6163e-07	7.6163e-07
aleatorio desordenado	5.3216e-07	5.3216e-07	5.3216e-07
15 segundos	5.4199e-07	5.4199e-07	5.4199e-07
30 segundo	7.9482e-07	7.9482e-07	7.9482e-07

5.4. Experimentación cualitativa sobre distintas estructuras de grafos

Para analizar los resultados obtenidos, queremos ver como influye la estructura del grafo y el valor de p. Decidimos plantear las siguientes estructuras:

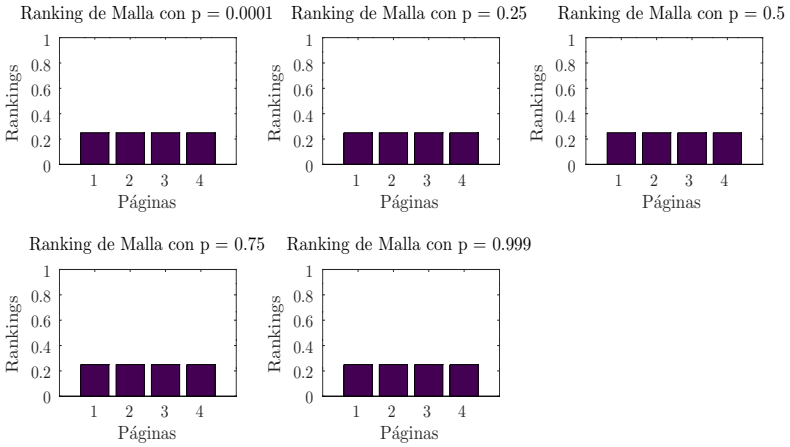
5.4.1. Malla

La estructura que proponemos son 4 nodos conectados todos entre si de manera bidireccional.



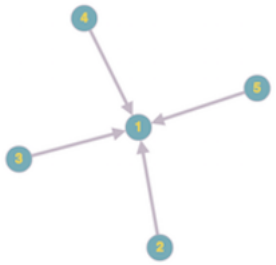
Se caulcula el Page Rank sobre distintos p.

Figura 2



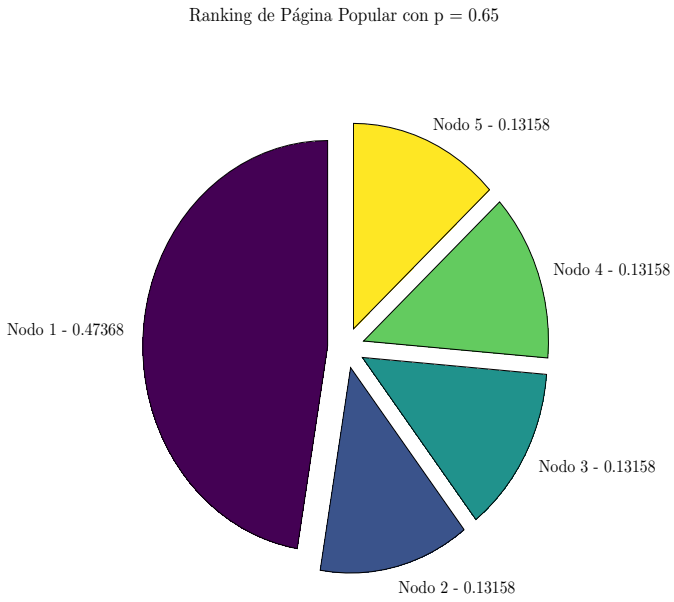
5.4.2. Página popular

Todas las páginas estan conectadas a una central.



Se calcula el Page rank para $p = 0{,}65$.[1]

Figura 3

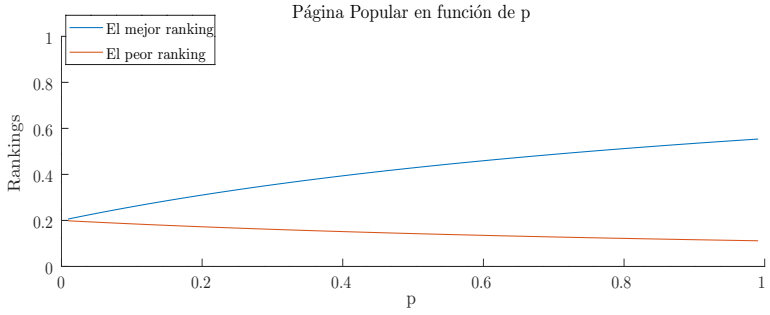


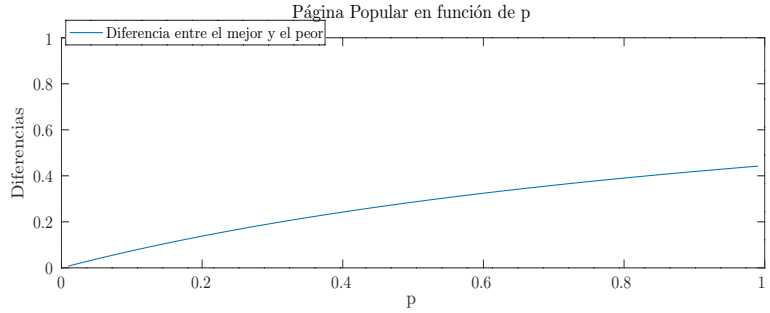
Calculamos el ranking de la páginas más y menos populares del sistema, y su variación en función de p.[2]

Figura 4

[1] Siempre que calculamos el Page Rank de un sistema, a primera instancia usamos $p = 0{,}65$ porque es lo suficientemente alto para dar a luz las particularidades del sistema.

[2] Siempre que graficamos rankings en función de p, se calcula el Page Rank para el sistema con 100 valores de p distintos espaciados de forma uniforme.





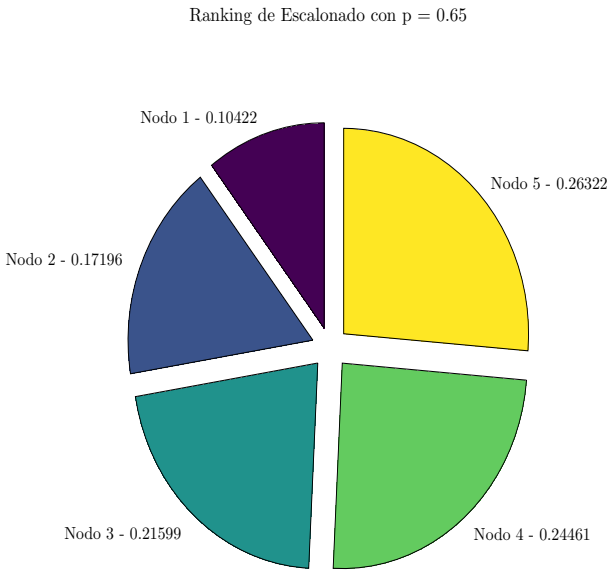
5.4.3. Escalonado

«Todos los caminos conducen a Roma. (O la Página 5)»

Esta estructura son 5 nodos con la forma de una lista enlazada. De forma que la Página 5 es el final de la lista.

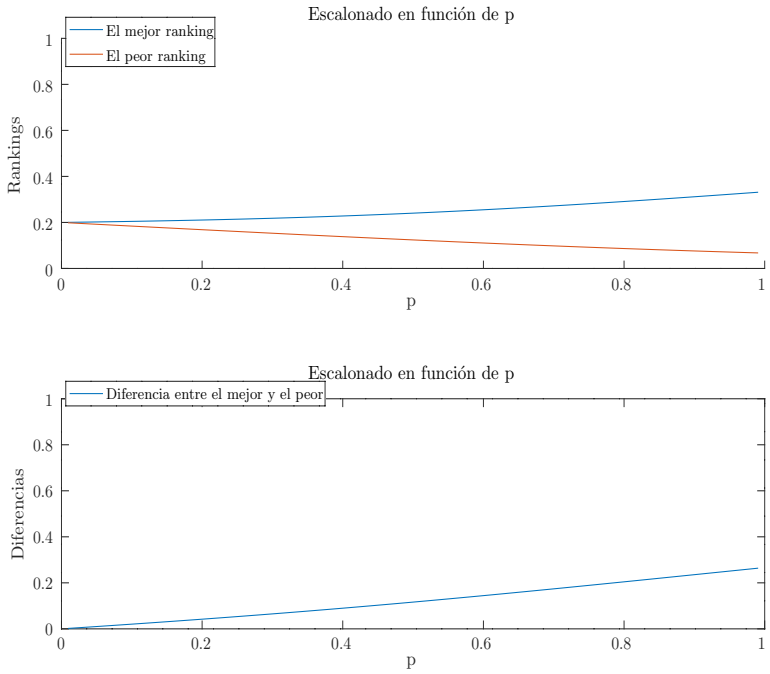
Se calcula el Page Rank con $p = 0{,}65$.

Figura 5



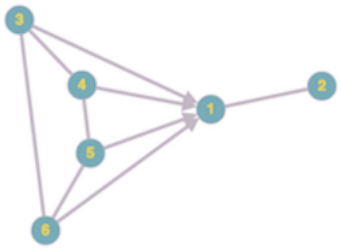
Calculamos el ranking de la páginas más y menos populares del sistema, y su variación en función de p.

Figura 6



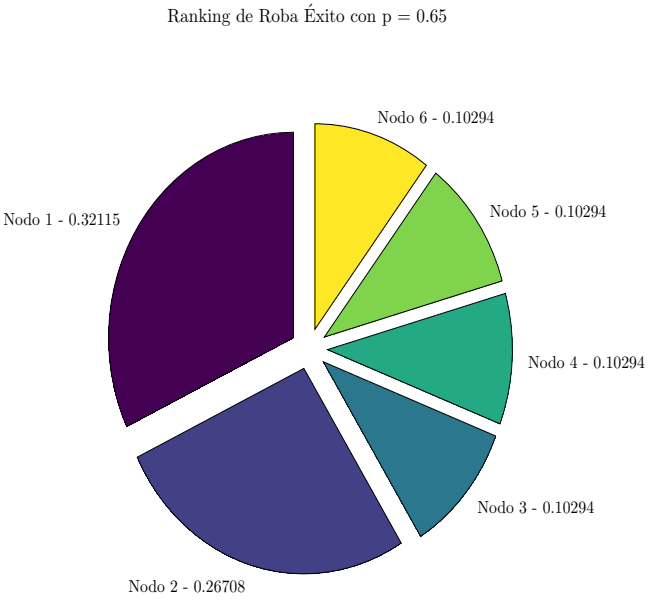
5.4.4. Roba éxito

Todas la páginas tienen un link a la página 1. Luego las paginas 3, 4, 5 y 6 estan contectadas entre si y la página 1 solo tiene un link a la página 2.



Se calcula el Page Rank con $p = 0{,}65$.

Figura 7


Ranking de Roba Éxito con p = 0.65
Nodo 6 - 0.10294
Nodo 5 - 0.10294
Nodo 4 - 0.10294
Nodo 3 - 0.10294
Nodo 2 - 0.26708
Nodo 1 - 0.32115

6. Discusión

6.1. Experimentación sobre predicados teóricos

6.1.1. El valor de p y el error del sistema

En la Figura 1 se puede observar claramente que a medida que p crece, el error del sistema aumenta. Sin embargo, nos sorprendió que si bien el error máximo es 5 veces mayor al mínimo, estábamos esperando una diferencia mucho más drástica. De forma que si bien el valor de p afecta la precisión de la solución, nuestro algoritmo se sigue comportando bien y dentro de los parámetros aceptables en todo el espectro de p.

6.2. Experimentación cuantitativa

6.2.1. Aproximación de la solución

Como vemos en la Tabla 1, el nuestro programa hace una buena aproximación a la solución, siendo perfecta en el caso trivial. En los tests más simples (sin links y completo), se ve que la implementación se encuentra con pocos problemas, teniendo un error del orden de 10^{-17}. Con tests más complejos, este error aumenta, pero incluso en los tests intensivos de 15 y 30 segundos se comporta dentro de lo esperable con errores del orden de 10^{-7}.

6.2.2. Error absoluto con respecto a los tests de la cátedra

En la Tabla 2 podemos ver que nuestra implementación da resultados iguales a los de la cátedra en la mayoría de los casos. La única diferencia se encuentra en los tests más intensivos (de 15 y 30 segundos), donde se encuentra una pequeña discrepancia del orden de 10^{-7}.

6.3. Experimentación sobre distintas implementaciones

6.3.1. Error con distintos tipos de datos para almacenar punto flotante

Al realizar la implementación, intentamos evitar distintas fuentes de imprecisión, una de ellas siendo los tipos de datos, por lo cual desde el inicio usamos *long double*. Sin embargo, como se puede ver en la Tabla 3, no hay diferencia en el error al usar este tipo o *double*. Lo que si se puede diferenciar es entre estos dos y *float*, teniendo este errores ligeramente mayores, pero casi insignificantes.

6.3.2. Error con distintos tipos de sumatoria

Al probar distintos tipos de sumatorias para normalizar el resultado, esperábamos que este fuera una fuente de imprecisiones. Sin embargo, como se ve en la Tabla 4, tanto la suma más simple, como la que ordena antes los números, y la que usa el algoritmo de Kahan, dan un error exactamente igual, por lo que vemos que no es un paso crítico en la pérdida de precisión.

6.4. Experimentación cualitativa sobre distintas estructuras de grafos

6.4.1. Malla

Lo que esperábamos con esta estructura es que el valor de p no importe, ya que todas las páginas tienen la misma importancia en la red.
En la Figura 2 podemos observar que independientemente de la probabilidad de saltar a otra página del conjunto, el ranking se mantiene igual en todas las páginas. Siendo este un caso muy sencillo, podemos fácilmente ver que el algoritmo de Page Rank está haciendo lo que debe. En este caso muy particular donde ninguna página es mejor que otra, es esperable y también es efectivamente el resultado, que el ranking sea el mismo para todas.

6.4.2. Página popular

En este caso, pretendíamos que la primer página siempre tenga el Page Rank más alto y que las otras página tengan igual ranking, pero menores. Aun así, esperábamos que cuanto más cerca p este de 0, todos las páginas tiendan a tener el mismo rank de 1/n.

Con la Figura 3 podemos notar que la Página 1, nuestra página popular, efectivamente se lleva el primer lugar en el Ranking de Page. Más aún, todas las otras páginas tienen exactamente el mismo ranking, están todas empatadas en el segundo lugar. Esto tiene mucho sentido, dado que la Página 1 recibe links de todas las páginas del conjunto, mientras todas las demas, no reciben links. Sus puntajes dependen exclusivamente del valor de p. La única razon por la que sus valores no son despreciables, es que todavia tienen probabilidad de ser visitadas espontaneamente gracias a que la probabilidad de elegir una página del sistema de forma aleatoria es $1 - p$, y para este experimento $p = 0{,}65$. Por este motivo, en el siguiente experimento se realizamos distintas mediciones modificando p.

Con los gráficos de la Figura 4, se puede observar que a medida que p crece, el mejor ranking es aún mejor, y de la misma forma, el peor ranking es aún peor. También se puede ver en el segundo gráfico que la diferencia entre ambos gráficos crece. Pero lo que resulta más importante para este experimento, es que cuando p toma un valor muy cercano a 0, la diferencia entre el mejor ranking y el peor (o los peores, esperando que los 4 peores sean iguales como sucedió con $p = 0{,}65$) también toma un valor cercano a cero. Con estos resultados podemos concluir que nuestra hipótesis es cierta. Esto se explicaría con que, si la probabilidad de saltar a otra página del sistema es muy alta $(1 - p)$, entonces la probabilidad de caer en cualquier página tiende a ser equiprobable, y por lo tanto el ranking es (casi) un empate.

6.4.3. Escalonado

Lo que esperábamos es que la página 5 gane siempre el PageRank y que la 1 siempre sea la última, ya que la página 1 no recibe ningún link y la número 5 esta indirectamente apuntada por todas las páginas del sistema. Además esperamos que cuanto más cerca p esté de 0, todos las páginas tengan el mismo

ranking de 1/n.

Como se puede ver en la Figura 5, el resultado fue el esperado. La página 5 es el final de la cadena, de cierta forma todos los links del sistema terminan eventualmente en esta página. Por lo que era esperable que esta sea la página con mejor ranking. Más aún, la página 1 es el inicio de la cadena, ningún link la apunta, por lo cual también tiene sentido que sea la página con peor ranking. Sin embargo, si bien el orden de los rankings fue el predicho, esperábamos que los valores estén aún más inclinados al final de la cadena. Esto nos dio pie a experimentar con distintos valores para p. Tal vez $p = 0{,}65$ era un valor muy alto (o muy bajo).

Con el resultado de este experimento en la Figura 6 podemos concluir que cuando p toma un valor muy pequeño, arruina el ranking del sistema. Ya que los valores del ranking resultan muy similares, implicando que existe una uniformidad en la probabilidad de caer en cualquier página.

6.4.4. Roba éxito

Lo que esperábamos es que la página 1 siempre tenga mayor ranking pero que sea bastante parecido a la página 2, ya que la página 1 tiene un link a la página 2. Además pretendemos que las páginas 3, 4, 5, 6 tengan pagerank iguales. El proposito de esta estructura, es verificar un aspecto importante del algoritmo. El PageRank pretende que sea mejor recibir un solo link de una página con muy buen ranking, a recibir muchos links de páginas poco populares. La idea del experimento fue corroborar que si bien la Página 2 recibe un solo link, su ranking aún así debería ser alto, dado que que este link proviene de la Página 1, que es (o esperábamos) la página más popular del sistema.

Tal como muestra la Figura 7, los resultados de este experimento fueron los esperados. La Página 1 es la más popular, seguida de la Página 2, y en tercer puesto con un cuadruple empate, el resto de las páginas. Es un resultado muy satisfactorio, dado que demuestra este aspecto tan importante del algoritmo

7. Conclusiones

Se nos pidio realizar un sistema de ranking de páginas web que utilice el algoritmo de page rank explicado en el enunciado del trabajo práctico.

Escrbimos el algoritmo en C++ usando nuestra propia implentación de matriz rala para ahorrar espacio en memoria y tiempo al ejecutar el programa. Támbien utilizamos el algoritmo de eliminación gaussiana visto en clase para poder resolver el problema.

Pudimos resolver las pruebas planteadas por la catedra con mejor aproximación de la que se pedia y en un tiempo similar.

Igualmente quisimos realizamos experimentos para encontrar posibles mejoras y poner a prueba nuestro programa. Decidimos por un lado, probar lo que aprendimos en las clases de laboratorio sobre precisión de datos y por otro ver si se cumple lo que el algoritmo de page rank propone.

En las clases de laboratorio vimos distintos tipos de representación de datos como float, double y long double. Quisimos experimentar sobre que tipo de dato produce menos error en nuestro programa. Llegamos a la conclusión, que si bien long double y double tienen un error parecido conviene usar double, ya que ocupa menos lugar en memoria.

También vimos distintas formas de sumar valores no enteros. Quisimos probar cual producía menos error en nuestro programa. Como mencionamos anteriormente, no encontramos ningun algoritmo que se destaque más que otro.

Por último hicimos experimentaciones sobre el algoritmo de page rank. Page rank propone que puede calcular la importancia de una página contando la cantidad de links y la cálidad de cada uno. Es decir, los links de páginas importantes valen más. Pudimos concluir que esto es cierto en el experimento roba éxito que planteamos. Además experimentamos como influye el modelo del navegante aleatorio en el proceso de obtener el page rank de una página. Siempre cuando p tiende a 0 todas las páginas tienen aproximadamente un mismo ranking. Luego el valor de p influye de manera distinta a cada sistema dependiendo de su estructura.

8. Apéndices

8.1. Apéndice A: Enunciado

Métodos Numéricos
2^o Cuatrimestre 2018
Trabajo Práctico 1: PageRank

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Sistemas de ecuaciones lineales

Introducción

El motor del buscador de Google, desde hace más de 20 años, utiliza el *ranking de Page* o *PageRank* [1] como uno de los criterios para ponderar la importancia de los resultados de cada búsqueda. Calcular este ranking requiere simplemente resolver un sistema de ecuaciones lineales donde la cantidad de ecuaciones e incógnitas del sistema es igual al número de páginas consideradas. ¿Simplemente?

Modelado del problema

Para un determinado conjunto de n páginas web definamos la *matriz de conectividad* $\mathbf{W}$ poniendo $w_{ij} = 1$ si la página j tiene un link a la página i y $w_{ij} = 0$ si no. Además $w_{ii} = 0$ pues ignoramos los *autolinks*. De esta forma, la matriz $\mathbf{W}$ puede resultar extremadamente rala y muy grande de acuerdo al tamaño del conjunto.

Para cada página $j = 1 \dots n$, definimos su *grado* como:

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

Es decir, la cantidad de links *salientes* de j. Típicamente c_j es un número mucho menor que n.

Se busca que el ranking sea mayor en las páginas *importantes*. Heurísticamente, una página es importante cuando recibe muchos "votos" de otras páginas, es decir, links. Pero no todos los links pesan igual: los links de páginas más importantes valen más. Pocos links de páginas importantes pueden valer más que muchos links de páginas poco importantes. Y los links de páginas con muchos links valen poco. Por lo tanto, una forma de calcular la importancia o *puntaje* x_i de la página i es:

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

Es decir, la página j le aporta a i su puntaje ponderado por cuántos links salientes tiene. También, se puede definir la matriz de puntajes $\mathbf{R} = \mathbf{WD}$, donde $\mathbf{D}$ es una matriz diagonal con elementos d_{jj} de la forma:

$$d_{jj} = \begin{cases} 1/c_j & \text{si} \quad c_j \neq 0 \\ 0 & \text{si} \quad c_j = 0 \end{cases},$$

[1] Denominado así por Larry Page, uno de los fundadores de Google, otrora joven científico actualmente devenido en multimillonario. Ver artículo original del 1998 (citado más de 17400 veces) en [1]

Lo cual nos permite calcular el ranking de todas las páginas como:

$$\mathbf{R}\ \mathbf{x} = \mathbf{x} \tag{3}$$

donde $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n)^T$. Luego, la ecuación 2 corresponde al elemento i-ésimo $(\mathbf{R}\ \mathbf{x})_i$.

Este modelo tiene un problema y es que no logra capturar el comportamiento errático del usuario mientras surfea la red redes.

Modelo del navegante aleatorio

Un enfoque alternativo es considerar el modelo del *navegante aleatorio*. El navegante aleatorio empieza en una página cualquiera del conjunto, y luego en cada página j que visita elige con probabilidad $p \in (0, 1)$ si va a seguir uno de sus links, o con probabilidad $1 - p$, si va a pasar a otra página cualquiera del conjunto. Una vez tomada esa decisión, si decidió seguir un link de la página j elige uno al azar con probabilidad $1/c_j$, mientras que si decidió pasar a otra página cualquiera entonces elige una al azar con probabilidad $1/n$). Cuando la página j no tiene links salientes, es decir $c_j = 0$, elige al azar una página cualquiera del conjunto. Por lo tanto, se espera que luego de mucho surfear el navegante aleatorio va a estar en páginas importantes con mayor probabilidad.

Formalmente, la probabilidad de pasar de la página j a la página i es:

$$a_{ij} = \begin{cases} (1-p)/n + (p\ w_{ij})/c_j & \text{si} \quad c_j \neq 0 \\ 1/n & \text{si} \quad c_j = 0 \end{cases}, \tag{4}$$

y sea $\mathbf{A} \in \mathbb{R}^{n \times n}$ a la matriz de elementos a_{ij}. Entonces el *ranking de Page* es la solución del sistema:

$$\mathbf{A}\ \mathbf{x} = \mathbf{x} \tag{5}$$

que cumple $x_i \geq 0$ y $\sum_i x_i = 1$.

Por lo tanto, el elemento i-ésimo $(\mathbf{A}\mathbf{x})_i$ es la probabilidad de encontrar al navegante aleatorio en la página i sabiendo que x_j es la probabilidad de encontrarlo en la página j, para $j = 1 \ldots n$;

Luego, la matriz $\mathbf{A}$ puede reescribirse como:

$$\mathbf{A} = p\ \mathbf{W}\ \mathbf{D} + \mathbf{e}\ \mathbf{z}^T,$$

donde $\mathbf{D}$ es una matriz diagonal de la forma

$$d_{jj} = \begin{cases} 1/c_j & \text{si} \quad c_j \neq 0 \\ 0 & \text{si} \quad c_j = 0 \end{cases},$$

$\mathbf{e}$ es un vector columna de unos de dimensión n y $\mathbf{z}$ es un vector columna cuyos componentes son:

$$z_j = \begin{cases} (1-p)/n & \text{si} \quad c_j \neq 0 \\ 1/n & \text{si} \quad c_j = 0 \end{cases}.$$

Así, la ecuación (5) puede reescribirse como

$$(\mathbf{I} - p\ \mathbf{W}\ \mathbf{D})\ \mathbf{x} = \gamma\ \mathbf{e}, \tag{6}$$

donde $\gamma = \mathbf{z}^T\ \mathbf{x}$ funciona como un factor de escala. Se puede demostrar[2] que resolver este sistema lineal mediante el siguiente procedimiento brinda un método eficaz para calcular el ranking de Page.

Cálculo del ranking de Page

El procedimiento para calcular el ranking de Page consiste en:

1. Suponer $\gamma = 1$.
2. Resolver el sistema lineal de la ecuación (6).
3. Normalizar el vector $\mathbf{x}$ de manera que $\sum_i x_i = 1$.

Enunciado

Se debe implementar un programa en `C` o `C++` que realice el cálculo del ranking de Page según el procedimiento descripto anteriormente.

Como parte **obligatoria** se pide implementar lo siguiente:

1. El método de Eliminación Gaussiana (EG) *sin* pivoteo.
2. Una estructura de matrices ralas que sea eficiente en espacio y en tiempo para la tarea que se busca realizar.
3. Herramientas de entrada/salida para la lectura de archivos con los datos de los conjuntos de páginas y escritura del ranking de Page. Se debe respetar el formato que se describe más abajo.

Previamente, **se deberá** estudiar las características de la matriz involucrada y responder a lo siguiente:

1. ¿Por qué la matriz $\mathbf{A}$ definida en (4) es equivalente a $p\ \mathbf{W}\ \mathbf{D} + \mathbf{e}\ \mathbf{z}^T$? Justificar adecuadamente.
2. ¿Cómo se garantiza la aplicabilidad de EG sin pivoteo? ¿Qué tipo de matriz resulta $(\mathbf{I} - p\ \mathbf{W}\ \mathbf{D})$?
3. OPCIONAL ¿Calcular el número de condición de $(\mathbf{I} - p\ \mathbf{W}\ \mathbf{D})$? ¿Cómo influye el valor de p ?

[2] Esto se puede ver en el artículo [4] pero no es objetivo de este trabajo práctica.

Además, **se pide** realizar un informe utilizando como guía las pautas de laboratorio de la materia conteniendo la experimentación pedida en la siguiente sección. Es importante incluir en la sección desarrollo del informe del trabajo práctico, las alternativas consideradas y descartadas para cada uno de los métodos utilizados.

Se **recomienda** fuertemente en todos los casos comparar los resultados intermedios utilizando Matlab/Octave.

Casos de test

La cátedra proveerá un conjunto de tests con archivos de entrada y salida esperada. Para aprobar el trabajo, los mismos **deberán** funcionar correctamente en sus implementaciones con una tolerancia máxima de 10^{-4} medida en error absoluto con respecto a los valores de ranking proporcionados.

Además, cada test **deberá** correr en las computadoras del laboratorio en un tiempo de similar en orden de magnitud al indicado.

Experimentación

Se deberá realizar tanto un análisis **cualitativo** como **cuantitativo** de los métodos vistos en el trabajo.

Para el análisis cuantitativo, se **pide**, analizar los errores obtenidos en el ranking de Page utilizando:

1. La aproximación de la solución x' obtenida: $|Ax' - x'| \approx 0$
2. El error absoluto con respecto a los test de la cátedra.

Para el análisis cualitativo se **deberán** estudiar los rankings obtenidos, en función de la estructura del grafo, y del valor de p. Para esto, cada grupo **deberá** proponer instancias de prueba no triviales, que crean pertinentes para analizar el método (entregando además los archivos correspondientes). Para guiar el análisis, responder las siguientes preguntas:

1. ¿Cómo es el ranking obtenido en cada caso de acuerdo a la estructura del grafo páginas?
2. ¿Qué conclusiones pueden sacar de la interpretación de los resultados?
3. Respecto del ranking de Page: ¿Funciona cómo era esperado? ¿Hubo sorpresas? ¿Qué pueden concluir sobre su significado? ¿Dirían que es un buen ranking?
4. ¿Cómo influye el valor de p en la calidad de los rankings?

Programa ejecutable y entrada/salida

Los archivos de entrada y salida serán de texto plano, y deberán respetar el siguiente formato:

archivo entrada: En la primera línea un entero N, la cantidad total de páginas. En la segunda línea un entero M, la cantidad total de links. Luego siguen M líneas, cada una con dos enteros i j separados por un espacio ($1 \leq i, j \leq N$), indicando que hay un link de la página i a la página j.

archivo de salida: La primera línea deberá contener el valor de p utilizado. Luego, deberán seguir N líneas, donde la línea i contiene el valor del ranking de Page para la página i.

En el Cuadro 1 se pueden ver las primeras 8 líneas de un archivo de entrada que contiene una instancia con 30 páginas y 178 links entre ellas, y su salida obtenida como resultado de correr el programa con $p = 0{,}8$.

30	0.8
178	0.00042152
1 3	0.00026892
1 2	0.000230783
1 19	0.000329963
2 26	0.000117996
3 12	0.000359948
3 4	0.000367727
...	...

Cuadro 1: Izquierda: Ejemplo de un archivo de entrada. Derecha: archivo de salida.

El programa ejecutable **deberá** cumplir con el siguiente formato de uso:

```
./tp1 archivo p
```

Donde `archivo` es la ubicación del archivo de entrada a leer, y `p` es el valor de p a usar. El archivo de salida deberá ser escrito en la ubicación `archivo.out`.

Además, se **deberá** entregar un archivo `README` conteniendo la instrucciones de compilación y ejecución, y también ejemplos de invocación del programa.

Fechas de entrega

- *Formato Electrónico:* jueves 6 de septiembre hasta las 23.59 hs, enviando el trabajo (informe + código) a la dirección `metnum.lab@gmail.com`.

  - El subject del email debe comenzar con el texto `[TP1]` seguido de la lista de apellidos de los integrantes del grupo separados por punto y coma `;`. Ejemplo: `[TP1] Lennon; McCartney; Starr; Harrison`

  - Se ruega no sobrepasar el máximo permitido de archivos adjuntos de 20MB. Tener en cuenta al realizar la entrega de no ajuntar bases de datos disponibles en la web, resultados duplicados o archivos de backup.

- Formato físico: viernes 7 de septiembre en la clase de laboratorio.

  Se debe entregar solamente la impresión del pdf del informe que se entregó por mail. Ambas versiones **deben** coincidir.

- Pautas de laboratorio:
  `https://campus.exactas.uba.ar/pluginfile.php/109008/course/section/16502/pautas.pdf`

Importante: El horario es estricto. Los correos recibidos después de la hora indicada serán considerados re-entrega.

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. *SIAM review*, 48(3):569–581, 2006.

[3] 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.

[4] Amy N Langville and Carl D Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2004.

8.2. Apéndice B: Código fuente relevante numéricamente

Agregamos el código usado para resolver el sistema (método resolver de MatrizRala).

```
vector<ld> MatrizRala::resolver(ll gamma) {
    vector<ld> b(this->filas.size(), gamma);

    // Eliminacion gaussiana
    for (unsigned int i = 0; i < this->filas.size() - 1; i++) {
        for (unsigned int j = i + 1; j < this->filas.size(); j++) {
            this->restarFila(i, j, b);
        }
    }

    // Sustituyo para atras
    vector<ld> x;
    x.push_back(b.back() / this->elem(this->filas.size() - 1, this->filas.size() - 1));
    for (int i = this->filas.size() - 2; i >= 0; i--) {
        ld sum = 0;

        for (unsigned int j = i + 1; j < this->filas.size(); j++) {
            sum += this->elem(i, j) * x[this->filas.size() - j - 1];
        }
        x.push_back((b[i] - sum) / this->elem(i, i));
    }

    reverse(x.begin(), x.end());
    this->normalizar(x);
    return x;
}
```

8.3. Apéndice C: Demostración A e.d.d. $\Rightarrow$ A inversible

Si $A = ((a_{i,j})_{i,j\in[1,n]})$ es una matriz de diagonal estrictamente dominante, entonces A es inversible

Demostración:

Por contrarrecíproco: Supongamos que A no es inversible, entonces su núcleo no se reduce a cero, existe entonces un vector: $X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \neq 0$ tal que $AX = 0$.

Entonces, se tiene que:

$$\forall i \in [1,n], \sum_{j=1}^{n} a_{i,j}x_j = 0$$

Como $X \neq 0$, existe $x_{i_0} \neq 0$ tal que $|x_{i_0}| = \text{máx}\{|x_i|, i \in [1,n]\}$.

Tenemos: $-a_{i_0,i_0}x_{i_0} = \sum_{\substack{j=1 \\ j\neq i_0}}^{n} a_{i_0,j}x_j$, de donde $|a_{i_0,i_0}x_{i_0}| \leq \sum_{\substack{j=1 \\ j\neq i_0}}^{n} |a_{i_0,j}x_j|$,

y como : $\forall j \in [1,n], \frac{|x_j|}{|x_{i_0}|} \leq 1$,

se obtiene:

$$|a_{i_0,i_0}| \leq \sum_{\substack{j=1 \\ j\neq i_0}}^{n} |a_{i_0,j}| \frac{|x_j|}{|x_{i_0}|} \leq \sum_{\substack{j=1 \\ j\neq i_0}}^{n} |a_{i_0,j}|$$

Finalmente, $|a_{i_0,i_0}| \leq \sum_{\substack{j=1 \\ j\neq i_0}}^{n} |a_{i_0,j}|$, lo cual es absurdo por hipótesis, absurdo que provino de suponer que A era no inversible.

9. Referencias

- C++ Docs - http://www.cplusplus.com/reference/
- Octave Docs - https://octave.org/doc
- Stack Overflow - https://stackoverflow.com/
- Herramienta web para graficar grafos - graphonline.ru/en/