Métodos Numéricos

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

Trabajo Práctico 2: Ranking de Page (¿y Plant?)

Integrante	LU	Correo electrónico
Ramiro Daniel Camino	264/06	ramirocamino@gmail.com
José Luis Escalante Catacora	822/06	joe.escalante@gmail.com
Juan Carlos Giudici	827/06	elchudi@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. Resumen

Dentro de la computación científica existen muchos campos de investigación, uno de ellos es el de la Teoría de las Comunicaciones. Internet forma parte de los temas que esta ciencia estudia y en esta oportunidad vamos a analizar un algoritmo famoso por aquellos que lo inventaron y famoso por sí mismo, el denominado "PageRank" o "Ranking de Page".

Dado a conocer por Larry Page y Sergey Brin en 1998, éste algoritmo forma parte principal del motor de búsqueda web Google y ha sido -y lo sigue siendo- uno de los pilares fundamentales en los cuales Google cimienta su éxito.

A lo largo de este trabajo desglosaremos este algoritmo usando el enfoque de la materia. Mostraremos la teoría en la cual se sustenta su resolución y ensayaremos distintas implementaciones del algoritmo, analizaremos su exactitud y performance con distintos casos de prueba, y evaluaremos los resultados para determinar qué implementación es mas conveniente bajo distintos criterios que daremos a conocer.

1.1. Palabras Clave

- **PageRank**
- **Random Walker**
- **QR**
- **Rotaciones de Givens**

2. Introducción Teórica

2.1. PageRank

El Page Rank (Ranking de Page) es un algoritmo de análisis de links (vinculos), fue nombrado asi por Larry Page. Es usado por el motor de búsqueda web Google, el cual asigna un peso numérico a cada elemento de un conjunto de datos relacionados entre sí. En este caso el conjunto es la web entera. El objetivo es medir la importancia relativa de cada elemento dentro del conjunto. El algoritmo puede ser aplicado a cualquier colección de entidades con referencias y citaciones recíprocas.

Podemos considerar la asignación de los pesos como una suma de votos: si una página P es referenciada por otras páginas, esos son votos positivos, y si P referencia a otras páginas, esos son votos negativos, pero además estos votos, no valen todos lo mismo (he aquí el gran truco) a mayor PageRank tenga P, mas pesado va a ser su voto. De esta manera puede ser posible que una página B sea referenciada por muchas páginas con peso bajo y estar en posición menor que otra C que es referenciada sólo por A que tiene mucho peso.

2.2. Rotaciones de Givens

Una rotación de givens es el proceso por el cual se multiplica una matriz A a izquierda por una matriz G para anular el elemento $A_{i,j}$ de A. La matriz G tiene la siguiente forma:

$$G(i,j) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

Donde $g_{i,j} = -s$, $g_{i,i} = c$, $g_{j,j} = c$ y $g_{j,i} = s$.
Además $c = A_{i,i}/r$, $s = A_{i,j}/r$ y $r = \sqrt{(A_{i,i})^2 + (A_{i,j})^2}$.

2.3. Descomposición QR

La descomposición QR de una matriz, es aquella que dada una matriz A la transforma al producto de dos matrices Q y R, donde Q es ortogonal y R es triangular superior obteniendo la siguiente igualdad $A = QR$. Una de las formas de lograr esta descomposición es utilizar las Matrices de Rotación de Givens aplicadas a A y Q se obtiene multiplicando el conjunto de matrices de rotación de Givens usadas para triangular a A.

El algoritmo de la descomposición tiene la siguiente forma:

QR($A: \mathbb{R}^{n \times n}$)→ $[Q, R]$:
1 $[Q, R] \leftarrow [I, A]$
2 **for** $i = 1, 2, \ldots, n-1$
3 Sea G_i matriz de Givens para eliminar el elemento $A_{i,i-1}$
4 $Q \leftarrow Q * G_i^t$
5 $R \leftarrow G_i * A$
6 **end for**
7
8 devolver $[Q, R]$
9 **end function**

2.4. Método de Jacobi

El método de Jacobi es un proceso iterativo con el cual se resuelve el sistema lineal $Ax = b$. Comienza con una aproximación inicial $x^{(0)}$ a la solucion x y genera una sucesión de vectores $x^{(k)}$ que convergen a x. Tiene como precondición que los elementos de la diagonal deben ser distintos de 0.

Para iniciar el proceso se descompone la matriz A, de la siguente forma: $A = D - L - U$, donde D es la diagonal de A, y L y U son la parte triangular inferior y superior respectivamente. Luego operamos de la siguiente forma:

$$Ax = b;$$
$$(D - L - U)x = b;$$
$$Dx = (L + U)x + b;$$
$$x = D^{-1}(L + U)x + D^{-1}b$$

Ahora reemplazamos la x del miembro izquierdo por nuestro $x^{(0)}$ y el x del miembro derecho por $x^{(k)}$, el cual en cada iteración irá aproximando al valor exacto de x. La fórmula quedaría así:

$$x^{(k)} = D^{-1}(L + U)x^{(0)} + D^{-1}b$$

La convergencia del método esta asegurada cuando la matriz A es o bien es diagonal dominante o definida positiva.

3. Desarrollo

Pasaremos a explicar cada uno de los puntos importantes del trabajo. Para empezar debemos aclarar que tomaremos por asumido que al referirnos a la matriz, estamos hablando del sistema a resolver, asimismo tomaremos la dimensión como n y mencionaremos indistintamente cada uno de ellos.

3.1. Planteo de la ecuación para resolver el PageRank

Recordemos que la ecuación que devuelve el ranking según el enunciado es:

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

donde γ funciona como un factor de escala, $\mathbf{I}$ es la identidad, p es la probabilidad de que el *Random Walker* (el mismo que el del enunciado) salte a un link de la página en la que está, $\mathbf{W}$ es la *matriz de conectividad*, $\mathbf{D}$ es una matriz diagonal de la forma

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

donde c_j es la cantidad de links salientes de la página j y por último $\mathbf{e}$ es un vector columna de unos de dimensión n. Por sugerencia del enunciado asignamos: $\gamma = 1$ y $p = 0,85$.

3.2. Resolución del sistema

Algo que es importante notar es que la matriz $\mathbf{W}$ tiene la mayoría de sus posiciones en 0, esto es debido a que en nuestro caso, $\mathbf{W}$ representa los links que hay entre páginas, y es natural asumir que por cada página, los links salientes son mucho menores a la dimensión de la matriz. Por lo tanto al resolver el sistema, tenemos entre manos una matriz que podríamos considerar *rala*, este fue un factor determinante a la hora de elegir los métodos de resolución.

Vamos a mostrar dos tipos de soluciones, una usando un método directo y la otra iterativa. Cada una de estas soluciones, se adaptan mejor a distintos tipos de matrices, por lo cual en la sección de Pruebas y Resultados veremos en qué casos es conveniente aplicar cada uno.

3.2.1. Método Directo A = QR

En ésta sección $A = (\mathbf{I} - p\,\mathbf{W}\,\mathbf{D})$

La razón principal para elegir QR como método de resolución directa, es que que para triangular contamos con las Rotaciones de Givens. Este procedimiento nos permite preservar lo mas posible, la propiedad rala de la matriz, ya que para cada posición a_{ij} que queramos eliminar de A, solo veremos afectadas las filas i y j respectivamente. Dada nuestra implementación de matrices y la dimensión de éstas, es importante que tengamos matrices con la mayor cantidad de ceros posibles, pues esto implica un ahorro en memoria que en cada paso de la triangulación será muy importante (ver Detalles de Implementación).

La complejidad de éste método es cúbica, veamos:

- Al tener una matriz de dimensión n, es decir de $n \times n$, tendremos que poner en 0, $\frac{n(n-1)}{2}$ posiciones
- Hacer Givens, tiene un costo lineal n, pues multiplicar a izquierda por G_i^t solo afecta a 2 filas de la matriz.

  Por lo tanto, asi que como vemos tenemos que aplicar Givens $\frac{n(n-1)}{2}$ veces, lo que nos da un orden de $O(n^3)$.

3.2.2. Método Iterativo Jacobi

Al tener una matriz con elementos no nulos en la diagonal, pudimos tomarnos la libertad de elegir entre los métodos de Jacobi y Gauss-Seidel, pero elegimos el de Jacobi por sobre el de Gauss-Seidel ya que implementativamente el segundo requiere no pisar los elementos de $x^{(k)}$ para el cálculo de $x^{(k+1)}$, por supuesto, para la optimización de memoria esto es importante, por lo tanto nos inclinamos por Jacobi que carece de este problema.

Como vimos en la Introducción Teórica, el método de Jacobi garantiza convergencia si la matriz es definida positiva o es diagonal dominante. Nosotros tomaremos como criterio el que sea diagonal dominante.

Para comenzar con la iteración tomamos como $x^{(0)} = (1, 0, 0, \ldots, 0)$, además implementamos un criterio de parada, la cual es: Para la iteración k de nuestro método, dado un $\epsilon \in \mathbb{R}$ al cual llamaremos *residuo* y la norma de Frobenius $\| \ . \ \|_F$, si se cumple que $\| Ax^{(k)} - b \|_F \leq \epsilon$ entonces paramos y devolvemos $x^{(k)}$.

3.3. Detalles de Implementación

En principio para implementar la matriz, se pensó en hacer un vector de vectores, donde en cada cada vector del vector principal serían las filas de la matriz. Sin embargo, nos dimos cuenta que estábamos desperdiciando espacio en memoria, pues como bien el enunciado lo dice (y nosotros también recalcamos), la matriz $(\mathbf{I} - p\ \mathbf{W}\ \mathbf{D})$ es rala, por lo tanto podriamos ahorrar memoria por cada lugar nulo.

A la hora de implementar una matriz rala, luego de investigar y analizar las posibilidades, decidimos que era mas práctico tener una estructura que se encargue de contruir progresivamente una matriz rala, y otra que represente a la matriz rala propiamente dicha, con una estructura interna que favorece las operaciones matriciales. Sin embargo, dada la complejidad de alterar esta estructura interna, diseñamos las matrices ralas para que sean inmutables.

Por otro lado, agregamos la opción de ordenar los elementos no nulos de la matriz rala de dos formas: considerando que la fila de una posición tiene mas prioridad que la columna, o al revez. Los algoritmos que se utilizan para acceder a las estructuras internas de la matriz rala varían levemente dependiendo de su orden, así que para una mayor claridad, en esta sección consideramos que siempre están ordenadas por filas, es decir, si definimos una posición como $P = (fila, columna)$, dados dos elementos distintos de cero, cada uno con posición

$P1 = (x1, x2)$ y $P2 = (y1, y2)$ entonces $P1 < P2$ si y sólo si $x1 < y1$ o bien si se da al mismo tiempo que $x1 = y1$ y $x2 < y2$.

3.3.1. Creador de Matrices Ralas

Esta estructura esta implementada en *C++* sobre un *map* de la *STL*, usando las posiciones como claves y el valor de la matriz en esa posicion justamente como valor del map. Además, cuando se inserta una nueva entrada, queda ordenada internamente, ya sea dandole mas importancia a la fila o a la columna de la posición. De esta manera, cuando se desea crear la matriz rala, es fácil iterar los elementos no nulos con el orden mencionado anteriormente.

Agregar o sacar elementos no nulos con el creador, según la documentación del *map* de la *STL*, tiene un orden logarítmico en la cantidad de elementos no nulos guardados. Crear la matriz tiene un orden lineal en función a la cantidad de elementos distintos de cero.

3.3.2. Estructura de la Matriz Rala

La idea básica de nuestra implementación es tener todos los elementos distintos de cero guardados de manera "plana" y algunos datos adicionales para indicar sus posiciones. Mas específicamente, los datos se organizan de la siguiente manera:

- Un arreglo Z con todos los elementos distintos de cero ordenados adecuadamente.
- Un entero z que representa la cantidad de elementos distintos de cero.
- Un entero n que representa la cantidad de filas de la matriz.
- Un entero m que representa la cantidad de columnas de la matriz.
- Un arreglo F de z elementos que indica la fila de cada elemento distinto de cero, es decir, que para $1 \leq i \leq z$, el elemento Z_i pertenece a la fila F_i de la matriz.
- Un arreglo C de z elementos que indica la columna de cada elemento distinto de cero, es decir, que para $1 \leq i \leq z$, el elemento Z_i pertenece a la columna C_i de la matriz.
- Un arreglo J de $n + 1$ elementos, donde para $1 \leq i \leq n$, el entero J_i indica donde inicia la fila i dentro de Z. De este modo, el primer elemento de la fila i es Z_{J_i} y el último es $Z_{J_{i+1}-1}$. El valor de J_{n+1} simplemente ayuda a que el invariante anterior se mantenga para la última fila. *Todo lo dicho en este punto es análogo para las columnas si la matriz está ordenada por columnas.*

Entonces, por cada elemento en la matriz distinto de cero hay un elemento en el arreglo Z, otro en F y otro en C, y por otro lado, hay un elemento en J por cada fila (o columna) en la matriz (mas uno). Luego, comparando con $n * m$ que es la cantidad de memoria que utiliza una matriz normal:

$$3 * z + (n + 1) < n * m \Leftrightarrow 3 * z < n * m - n - 1 \Leftrightarrow z < (n * (m - 1) - 1)/3$$

Podemos decir entonces que en cuestiones de memoria vale la pena utilizar esta matriz rala solo si la cantidad de elementos no nulos de la matriz cumple la desigualdad:

$$z < (n * (m - 1) - 1)/3$$

3.3.3. Iterador de Matrices Ralas

Esta estructura permite recorrer facilmente una matriz dependiendo del orden de la misma. Por ejemplo, si esta está ordenada por filas, entonces primero recorre por filas y luego por columnas. Sin embargo, mas allá del sentido en el cual avanza, solo pasa por los elementos no nulos de la matriz.

3.3.4. Acceso aleatório en Matrices Ralas

Acceder a una posición (i, j) de una matriz rala M no es muy recomendable sin el uso del iterador, ya que primero debe ubicar la fila dentro del arreglo F de la matriz M usando búsqueda binaria, y luego hacer lo mismo para la columna en el arreglo C de la matriz M. Es decir, que la complejidad de esta operación tiene un orden $O(log(n) + log(m))$. En una matriz normal esta operación es de orden constante.

3.3.5. Suma y Resta de Matrices Ralas

Para sumar o restar dos matrices ralas A y B, ambas con dimensión $n * m$, se necesita recorrer todos los elementos distintos de cero de cada una y agregarlos al resultado, teniendo en cuenta de que cuando la posición de un elemento distinto de cero de una de las dos matrices coincide con la posición de un elemento distinto de cero de la otra matriz entonces hay que operar entre esos elementos. El peor caso se da cuando no coinciden ningunas de las posiciones no vacías de las matrices, y por lo tanto, el orden de estas operaciones es $O(A_z + B_z)$, siendo A_z y B_z la cantidad de elementos no nulos de cada matriz. Para una matriz normal hay que operar con cada uno de sus elementos y por lo tanto el orden es $O(n + m)$, siendo n y m las dimensiones las matrices.

3.3.6. Multiplicación de Matrices Ralas

Multiplicar dos matrices ralas A y B es un poco mas complicado. Primero se iteran las filas de A, y por cada una se iteran las columnas de B. Luego, se inicializa un acumulador y se van iterando "a la par" las columnas de A con las columnas de B, ajustando a cada paso los iteradores cuando no coinciden los indices. Si la columna de A coincide con la fila de B se realiza la multiplicación entre el elemento actual de A y el de B y se suma en el acumulador. Al finalizar el ciclo en el cual los dos iteradores avanzan a la par, se guarda el acumulador en el lugar que corresponde. En el peor de los casos se recorren todos los elementos no nulos de A y de B una vez por cada fila con al menos un elemento no nulo de A, y esto empeora cuando hay al menos un elemento distinto de cero en cada una de las filas de A. Es decir, la complejidad

tiene un orden $O(n*(A_z+B_z))$. Si A y B son matrices normales, con dimensiones $n*k$ y $k*m$ respectivamente, la complejidad sería $O(n*k*m)$.

3.3.7. Otras operaciones con Matrices Ralas

Para trasponer una matriz rala de $n*m$ o para multiplicarla por un escalar solo se requiere recorrer cada uno de sus elementos distintos de cero una vez, y por lo tanto, estas operaciones tienen $O(z)$. Para una matriz normal hay que operar con cada uno de sus elementos y por lo tanto el orden es $O(n+m)$.

4. Pruebas y Resultados

Para poner a prueba nuestras implementaciones, lo que haremos en principio, será probar con casos pequeños, donde esperamos tener resultados previsibles. Luego iremos aumentando la cantidad de páginas y veremos la performance de la implementación.

4.1. Testeando con Casos Pequeños

Veamos algunos casos chicos:

- **Caso con 4 páginas** En este caso tenemos 4 páginas relacionadas de esta manera:

  1. $a \rightsquigarrow b$
  2. $a \rightsquigarrow c$
  3. $a \rightsquigarrow d$
  4. $b \rightsquigarrow c$
  5. $b \rightsquigarrow d$

  Como vemos tenemos c y d como las páginas mas referenciadas, por lo cual es de esperar pensar que el resultado tiene que dar que estos son los que tienen mayor ranking. Veamos pues el resultado luego de aplicar el método directo:

  - $a = 0{,}168327$
  - $b = 0{,}216019$
  - $c = 0{,}307827$
  - $d = 0{,}307827$

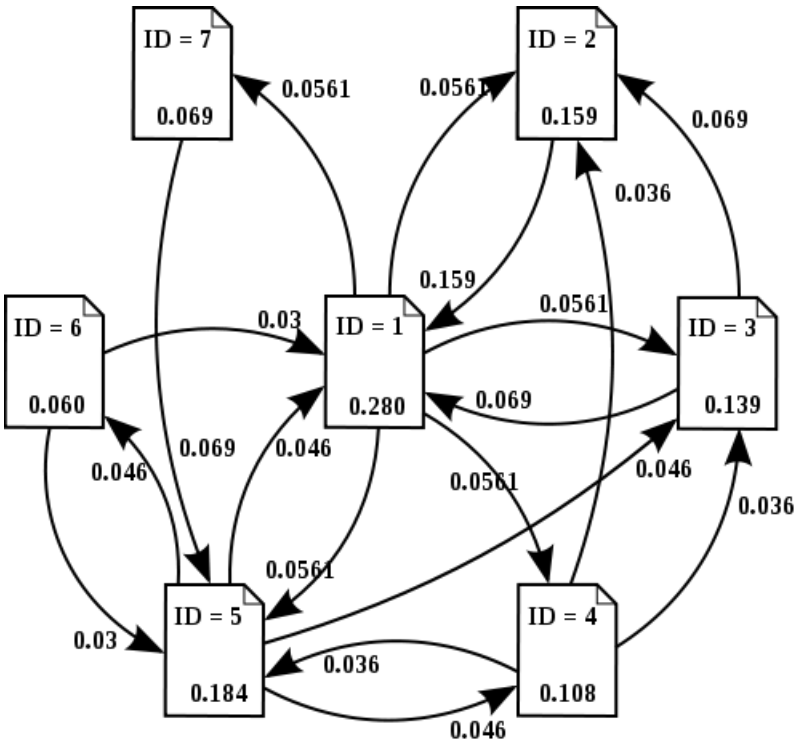
  Ahora veamos los resultados con el método iterativo, con 10 iteraciones y $k = 0{,}001$:

  - $a = 0{,}168327$
  - $b = 0{,}216019$
  - $c = 0{,}307827$
  - $d = 0{,}307827$

  También con 1000 iteraciones terminó dando lo mismo.

- **Caso con 7 páginas**

  Aca tenemos un conjunto de 7 páginas y 18 links, relacionadas como vemos en el gráfico:



Como bien se ve en el gráfico esperamos que la página de ID = 1, sea la que tenga mayor ranking. Aplicamos el método directo y nos quedó:

- **ID 1** = 0,280288
- **ID 2** = 0,158764
- **ID 3** = 0,138882
- **ID 4** = 0,10822
- **ID 5** = 0,184198
- **ID 6** = 0,0605707
- **ID 7** = 0,0690775

Luego aplicamos el iterativo, con 10 iteraciones y residuo 0,001:

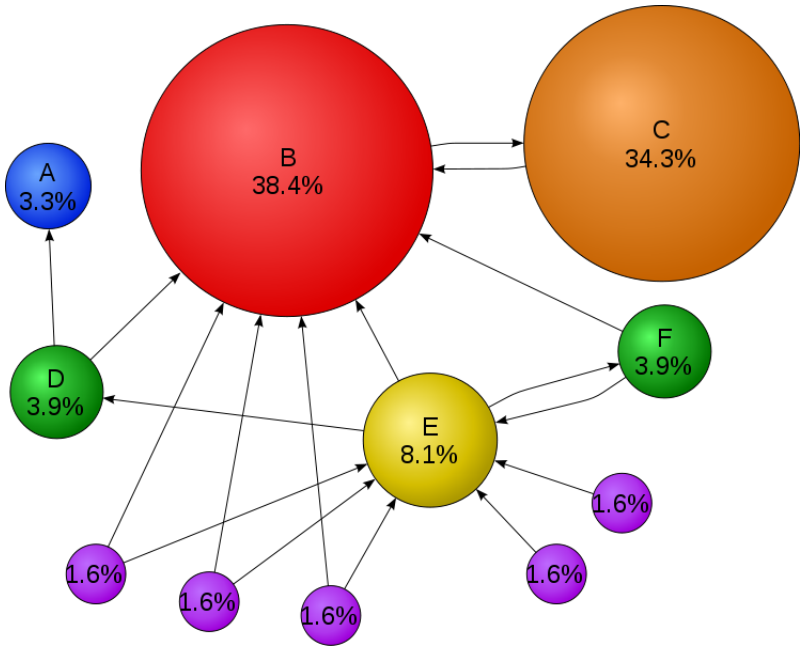
- **ID 1** = 0,274751
- **ID 2** = 0,157001
- **ID 3** = 0,138479
- **ID 4** = 0,108889
- **ID 5** = 0,18545
- **ID 6** = 0,0643551
- **ID 7** = 0,0710757

Ahora con 100 iteraciones, mismo residuo:

- **ID 1** = 0,280281
- **ID 2** = 0,158762
- **ID 3** = 0,138881
- **ID 4** = 0,10822
- **ID 5** = 0,1842
- **ID 6** = 0,0605753
- **ID 7** = 0,0690799

Con 1000 iteraciones nos quedó el mismo resultado.

- **Caso con 11 páginas** Aquí tenemos 11 páginas y un total 17 links, en el gráfico ilustrada, aparecen sus relaciones:



Acá esperamos que las páginas B y C, sean las que mas ranking tengan y seguida por la E. Aplicando el directo:

- $A = 0{,}0327815$
- $B = 0{,}384401$
- $C = 0{,}34291$
- $D = 0{,}0390871$
- $E = 0{,}0808857$
- $F = 0{,}0390871$
- $G = 0{,}0161695$
- $H = 0{,}0161695$

- $I = 0{,}0161695$
- $J = 0{,}0161695$
- $K = 0{,}0161695$

Luego el iterativo con 10 iteraciones y residuo 0,001:

- $A = 0{,}0399942$
- $B = 0{,}361587$
- $C = 0{,}30572$
- $D = 0{,}0476854$
- $E = 0{,}0986861$
- $F = 0{,}0476854$
- $G = 0{,}0197284$
- $H = 0{,}0197284$
- $I = 0{,}0197284$
- $J = 0{,}0197284$
- $K = 0{,}0197284$

Con 100 iteraciones mismo residuo:

- $A = 0{,}0327861$
- $B = 0{,}384386$
- $C = 0{,}342886$
- $D = 0{,}0390926$
- $E = 0{,}0808971$
- $F = 0{,}0390926$
- $G = 0{,}0161718$
- $H = 0{,}0161718$
- $I = 0{,}0161718$
- $J = 0{,}0161718$
- $K = 0{,}0161718$

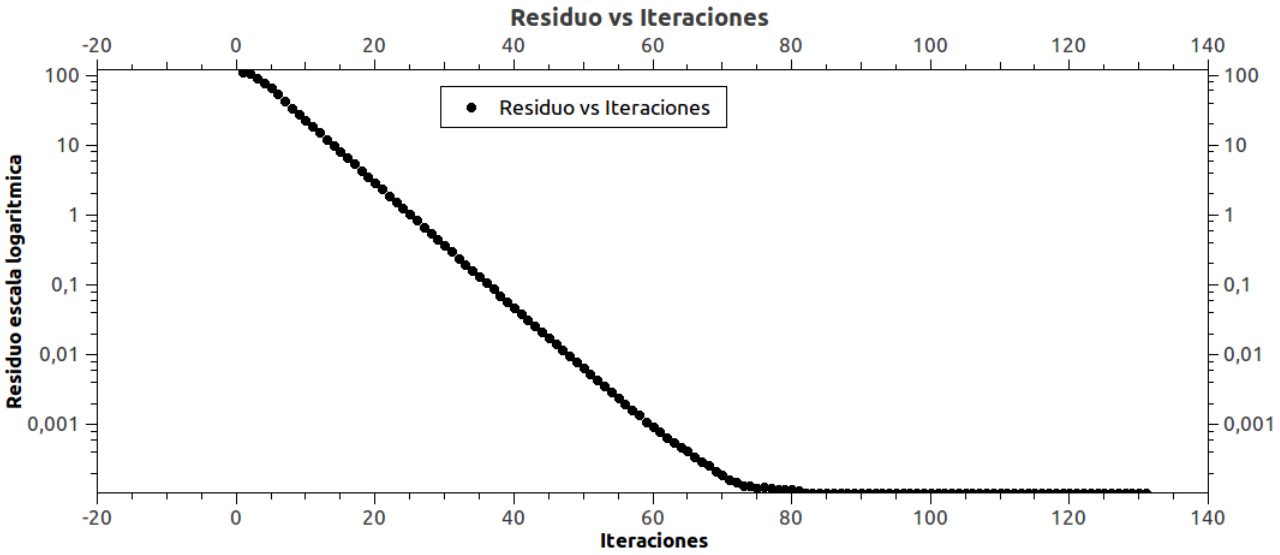
Con 1000 iteraciones, quedaron los mismos resultados.

También propusimos un caso de test en el cual todos las paginas estan relacionadas entre sí, es decir un grafo completo. Como era de esperarse tanto para el iterativo como para el directo, el ranking de cada uno de las páginas quedó 0,05. En esto nuestro cantidad de páginas analizadas fueron 20.

4.2. Iteraciones vs. Residuo

A continuación estudiaremos la relacion entre la cantidad de iteraciones y el residuo resultante, este estudio nos permitira evaluar la velocidad de convergencia del método iterativo.

Como caso de estudio, elegimos el caso provisto por la catedra de la wikipedia en Interlingua.



4.3. Testeando casos grandes

Dados los casos de tamaño mediano a grande, nos encontramos con lo siguiente, el método directo tarda un tiempo no razonable, dado el tamaño de los mismos, sin embargo el método iterativo para los casos intermedios se comporta lento, pero de forma aceptable, dando algùn resultado aproximado.

Vimos que mostrar los resultados de los casos intermedios a grandes, no aporta a la discusiòn, por lo tanto, mostramos que casos de los provistos por la catedra pudieron ejecutar como máximo los diferentes algoritmos.

- Directo: Wikidump - Extremeño (2395 páginas, 19084 links)
- Directo: Obama transition (páginas 30002, 379871 links)

5. Discusión

En los casos de prueba cuyos resultados son conocidos o el caso es relativamente chico que se puede ver a ojo si el ranking esta bien calculado obtuvimos los mismos resultados que los conocidos, las paginas más importantes son las que más son apuntadas y menos reparten su importancia.

En cuanto a la convergencia del metodo iterativo, podemos observar gracias al grafico de residuo contra iteraciones, que a mayor cantidad de iteraciones el algoritmo converge y la velocidad de convergencia es de orden logaritmico.

El método directo, si bien garantiza la solución es sumamente lento ya que tiene que revisar toda los elementos distintos debajo de la diagonal de la matriz a factorizar y hacer multiplicaciones de las matrices de givens por cada elemento, si los casos son muy grandes podemos observar que el método no termina en un tiempo razonable.

6. Conclusiones

6.1. Matriz Rala

Dado los resultados obtenidos en ambos algoritmos, nuestra implementación de la matriz rala nos resulto mal encaminada, la decisión de hacerla estática nos jugó en contra al momento de hacer la factorización QR con rotaciones de givens, ya que la matriz a factorizar se tiene que cambiar por cada 0 que se pone debajo de la diagonal, lo cual implica volver a crear la matriz por cada paso demorando el cálculo del mismo.

De todas formas, nuestra algoritmo iterativo con respecto a la matriz rala, aumenta la capacidad de procesamiento en cuanto al tamaño del caso, pero perdiendo precisión en los cálculos.

6.2. Método Iterativo

El método iterativo se presenta como una alternativa a resolver los casos que el método directo no puede abordar gracias al tamaño del mismo, solamente tenemos que tener en cuenta que si queremos una mayor precisión tenemos que sacrificar tiempo, por lo tanto, si el tiempo no es problema, podemos hacer mas iteraciones obteniendo un mejor resultado. Como la velocidad de convergencia es relativamente rapida al principio, se pueden obtener buenos resultados con pocas iteraciones, logrando una un resultado aceptable relativamente rapido.

6.3. Método Directo

El método directo se vio directamente afectado por nuestra implementación de la matriz rala, como bien dijimos antes, nuestra implementación nos obligaba a reconstruir la matriz a factorizar por cada paso de la factorización QR. De todas formas en los casos en que pudimos correr el método, obtuvimos los resultados esperados y el algoritmo funciona correctamente.

Este método es el ideal para casos chicos de prueba, o una vez mejorada en un futuro la matriz rala, poder abarcar casos mas grandes.

A. Enunciado

Introducción

El motor del buscador de Google utiliza el denominado ranking de Page[1] como uno de los criterios para ponderar la importancia de los resultados de cada bsqueda. Calcular este ranking requiere simplemente resolver un sistema de ecuaciones lineales... donde la cantidad de ecuaciones e incgnitas del sistema es igual al nmero de pginas consideradas. Simplemente?

Para un determinado conjunto de n pginas web definamos la *matriz de conectividad* $\mathbf{W}$ poniendo $w_{ij} = 1$ si la pgina j tiene un link a la pgina i y $w_{ij} = 0$ si no. Adems $w_{ii} = 0$ pues ignoramos los "autolinks". De esta forma, la matriz $\mathbf{W}$ puede resultar extremadamente rala y muy grande de acuerdo al tamao del conjunto. Por ejemplo, cada pgina j tiene $c_j = \sum_i w_{ij}$ links (salientes), que tpicamente es un nmero mucho menor que n.

Se busca que el ranking sea mayor en las pginas "importantes". Heursticamente, una pgina es importante cuando recibe muchos "votos" de otras pginas, es decir, links. Pero no todos los links pesan igual: los links de pginas ms importantes valen ms. Pocos links de pginas importantes pueden valer ms que muchos links de pginas poco importantes. Y los links de pginas con muchos links valen poco, por ser "poco especficos".

Un enfoque alternativo es considerar al "navegante aleatorio". El navegante aleatorio empieza en una pgina cualquiera del conjunto, y luego en cada pgina j que visita elige con probabilidad p si va a seguir uno de sus links, o (con probabilidad $1-p$) si va a pasar a una pgina cualquiera del conjunto. Usualmente $p = 0{,}85$. Una vez tomada esa decisin, si decidi seguir un link elige uno al azar (probabilidad $1/c_j$), mientras que si decidi pasar a una pgina cualquiera entonces elige una al azar (probabilidad $1/n$). Cuando la pgina j no tiene links salientes ($c_j = 0$) elige al azar una pgina cualquiera del conjunto. Heursticamente, luego de muchos pasos el navegante aleatorio estar en pginas importantes con mayor probabilidad.

Formalmente, la probabilidad de pasar de la pgina j a la pgina i es

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

y sea $\mathbf{A}$ a la matriz de elementos a_{ij}. Entonces el **ranking de Page** es la solucin del sistema

$$\mathbf{A}\,\mathbf{x} = \mathbf{x} \tag{1}$$

que cumple $x_i \geq 0$ y $\sum_i x_i = 1$. Si pensamos que x_j es la probabilidad de encontrar al navegante aleatorio en la pgina j, tenemos que $(\mathbf{Ax})_i$ es la probabilidad de encontrarlo en la pgina i luego de un paso. Y el ranking de Page es aquella distribucin de probabilidad que resulta "estable".

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

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

[1] Por Larry Page, uno de los fundadores de Google, otrora joven cientfico actualmente devenido multimillonario.

e es un vector columna de unos de dimensin n y **z** es un vector columna cuyos componentes son

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

As, la ecuacin (**??**) puede reescribirse como

$$(\mathbf{I} - p\ \mathbf{W}\ \mathbf{D})\ \mathbf{x} = \gamma\ \mathbf{e}, \tag{2}$$

donde $\gamma = \mathbf{z}^T\ \mathbf{x}$ funciona como un factor de escala.

De esta manera, un procedimiento para calcular el ranking de Page consiste en:

1. Suponer $\gamma = 1$.
2. Resolver la ecuacin (**??**).
3. Normalizar el vector **x** de manera que $\sum_i x_i = 1$.

Enunciado

El objetivo de este trabajo es programar el clculo del ranking de Page segn el procedimiento descripto anteriormente. Para la resolucin del sistema de ecuaciones resultante debern implementar por lo menos

- un mtodo directo y
- un mtodo iterativo,

que resulten apropiados.

Previamente debern estudiar las caractersticas de la matriz involucrada. Cmo se garantiza la aplicabilidad de cada mtodo? En el caso directo, la inversibilidad de $(\mathbf{I} - p\ \mathbf{W}\ \mathbf{D})$. Est bien condicionada? Y en el caso iterativo, la convergencia del mtodo.

Debern proponer 3 casos de prueba (conjuntos de pginas y sus links) de acuerdo a criterios establecidos por el grupo, para probar las implementaciones.

Los programas debern leer los archivos con los datos de los conjuntos de pginas (segn el formato que se describe ms abajo) y calcular el ranking. Las diferentes implementaciones debern utilizarse para calcular el ranking de pginas de varios conjuntos disponibles en la pgina de la materia propuestos por la ctedra y por los dems grupos. Podrn, adems, analizarse otros conjuntos elegidos por el grupo.

En base a los ensayos realizados:

- Interpretar los resultados. Cmo es el ranking obtenido en cada caso de acuerdo a las estructuras de los conjuntos? Qu conclusiones pueden sacar de la interpretacin de los resultados?
- Respecto del ranking de Page. Funciona cmo era esperado? Hubo sorpresas? Qu pue-den concluir sobre su significado?

- Respecto de los 2 ms mtodos implementados para la resolucin de los sistemas de ecuaciones lineales Cmo es el desempeo de cada uno? Considerar error numrico, costo computacional en tiempo, cantidad de operaciones, memoria y cantidad de iteraciones (cuando corresponda).

Datos de entrada

Cada conjunto de pginas ser descripto por 2 archivos de texto plano, a saber:

archivo de pginas: conteniendo en la primera lnea la cantidad de pginas, y luego para cada pgina una lnea con: el nmero y la *url* de la misma, separados por espacios; y

archivo de links: conteniendo en la primera lnea la cantidad total de links, y luego para cada link una lnea con: el nmero de la pgina de origen y el nmero de la pgina de destino (en ese orden), separados por espacios.

Entregas parciales en papel, no ms de una carilla de texto:

1 de octubre: ideas y soluciones propuestas, plan de implementacin, casos de prueba

8 de octubre: implementacin y plan de experimentacin.

Fecha de entrega final: 22 de octubre de 2010