Laboratorio de Métodos Numéricos

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

Trabajo Práctico Número 2

Ohhh solo tiran π-edras...

Integrante	LU	Correo electrónico
Ciruelos Rodríguez, Gonzalo	063/14	gonzalo.ciruelos@gmail.com
Costa, Manuel José Joaquín	035/14	manuc94@hotmail.com
Gatti, Mathias Nicolás	477/14	mathigatti@gmail.com

asdf

key1 **key2** **key3** **key4**

Índice

1. Introducción teórica

El objetivo del presente informe es resolver un problema práctico mediante el modelado matemático del mismo. Este problema consiste en modelar páginas web y ligas deportivas con cadenas de Markov, con el fin de obtener rankings para ellas.

Para esto, dada una cadena de Markov, construida de una cierta manera que será formulada más adelante, se va a considerar el modelo de navegante aleatorio. En este modelo, se comienza en un nodo cualquiera del diagrama y se va navegando a través de los links. Además podemos extrapolar esta idea, originalmente diseñada para páginas web, y utilizarla en ligas deportivas.

Entonces, la idea es, de alguna manera, rankear mejor aquellos nodos del diagrama de transición en los que el navegante aleatorio se encuentra más tiempo. Para ello, trataremos de encontrar un *estado estacionario*, pues este representará cual es la probabilidad de que el navegante se encuentre en cada nodo, si lo dejaramos recorriendo el diagrama infinito tiempo.

Volviendo a las cadenas de Markov, si la matriz de transición de la cadena es P (o sea, P_{ij} es la probabilidad de pasar del estado i al j), estamos buscando algun vector x tal que

$$x^t P = x^t$$

O equivalentemente,

$$P^t x = x$$

Como P es una matriz de transición, sus filas son vectores de probabilidad, por lo que $0 \leq P_i j \leq 1$ y las filas suman 1. En consecuencia, como se prueba en 5.1.1, el autovalor de mayor módulo es 1.

2. Desarrollo

2.1. Convenciones

2.2. Métodos numéricos usados

Como dijimos en la introducción, nuestro objetivo será, dada una matriz de transición P, encontrarle un autovector de autovalor asociado igual a 1 a su transpuesta. (Usamos la transpuesta por comodidad notacional).

$$P^t x = x$$

.

2.2.1. Método de la potencia

El método de la potencia, dada una matriz A, produce un autovalor λ y un autovector asociado a λ, v no nulo. El método es iterativo, y se puede encontrar una mejor explicación sobre él en [DB74, Cap. 5.8.1].

Consiste en tomar un $x^{(0)}$ inicial, y luego construir una sucesión $\{x^{(k)}\}$ de la siguiente manera:

$$x^{(i)} = \frac{Ax^{(i-1)}}{||Ax^{(i-1)}||}$$

Y entonces, bajo ciertas condiciones, si se toma k lo suficientemente grande, $x^{(k)} \to \overline{x}$, tal que $A\overline{x} = \lambda\overline{x}$, λ el autovalor de mayor módulo. Por ello establecemos como criterio de parada que la diferencia entre el vector generado en una iteración y su anterior sea lo suficientemente chica.

Como probamos en 5.1.1, el autovalor de máximo módulo en este caso es 1, aunque puede pasar también que $\lambda = -1$ también sea un autovalor. Sin embargo, comenzando con $x^{(0)} = (\frac{1}{n}, ..., \frac{1}{n})$ inicial, nos aseguramos de que las entradas sean siempre positivas, consiguiendo así un autovector asociado a autovalor $\lambda = 1$.

2.2.2. PageRank

PageRank más que un método de cómputo es un método de modelado. Dado un grafo cuyos nodos representan páginas webs y sus aristas representan links entre las páginas web, nos permitirá modelar un navegante aleatorio utilizando una cadena de Markov. Los detalles de la construcción de la cadena y la matriz asociada pueden encontrarse en [BP98].

Proveeremos una breve explicación de como se arma la matriz de transición utilizando un vector fila de la matriz P. P_i es la i-esima fila de la matriz, y su entrada j-ésima nos dice la probabilidad que habrá de ir de la página web i a la j. A priori una buena aproximación sería

$$P_{ij} = \begin{cases} \frac{1}{n_i} & \text{si hay un link de } i \text{ a } j \\ 0 & \text{si no} \end{cases}$$

Donde n_i es la cantidad de links salientes de la página i. El primer problema que se evidencia es que, en general, esta matriz no es de transición, porque si una página web no tiene links salientes, la matriz va a tener toda una fila de ceros. Por eso, en este caso, se agrega una fila que vale toda $(\frac{1}{n}, ..., \frac{1}{n})$.

Luego, se introduce el concepto de teletransportación. La idea es que, con una cierta probabilidad $1 - c$, el navegante aleatorio puede saltar a cualquier página de toda la red sin importar en cual esté actualmente. Todo esto, nuevamente, esta correctamente explicado en [BP98] y [Kam+03].

Luego, puede utilizarse el algoritmo descripto anteriormente para encontrar un autovector de autovalor asociado igual a 1.

En este trabajo en particular, utilizaremos una versión mejorada del método de la potencia, adaptada para este problema en particular, propuesta por [Kam+03]. Este consiste en separar el único paso del método de la potencia en 3 pasos separados, de tal manera de acelerar el cómputo, aprovechándonos de que la matriz de transición (sin agregarle el factor de teletransportación) es esparsa.

De esta manera, se pueden obtener importantes ganancias en lo que respecta a la performance.

2.2.3. Método GeM

El método GeM, propuesto en [GMA08], tiene como objetivo adaptar el algoritmo de PageRank para ligas deportivas. La idea es simple, al igual que en el algoritmo original de PageRank, la idea es armar una cadena de Markov y modelar un navegante aleatorio.

En este modelo, se representa una temporada (o una fecha, o un periodo de tiempo cualquiera) como un grafo dirigido y pesado, al igual que en el modelo de PageRank. Sin embargo, en este caso, los pesos de la primera matriz no valen 0 o 1, sino que son el valor absoluto de los puntajes de cada partido.

De esta manera, si el equipo i perdió contra el equipo j por p puntos, en la primera matriz A, valdra que $A_{ij} = p$.

Luego, al igual que en PageRank, las filas de esta matriz que valgan 0 (eso significa que el equipo está invicto hasta el momento) serán completadas y además se agregará el factor de teletransportación, haciendo que todas las entradas de la matriz P sean distintas de 0.

Al igual que antes, nuestro objetivo es encontrar un autovector de autovalor 1 para P^t, y para ello utilizaremos el método de la potencia común y corriente.

Es importante notar que este método es, a diferencia de el anterior, un método que nos indica *cómo* modelar, mientras que el método propuesto en [Kam+03] lo que hace es tomar un modelo conocido e intentar mejorar su velocidad de cómputo.

2.2.4. Otros métodos

Además de los métodos mencionados enteriormente, para cada problema (rankeo de páginas web y de ligas deportivas), utilizaremos un método alternativo.

En el caso de las páginas webs será el conocido como IN-DEGREE, que rankea mejor a aquellas páginas que son mas linkeadas y peor a las que son menos linkeadas. En la parte de experimentación se comparará estos métodos con la requerida profundidad.

En el caso de los rankings de ligas deportivas, utilizaremos el método standard de rankeo de cada deporte. En el caso del fútbol, este consiste en caso de empate un punto a cada equipo, y en otro caso darle 3 puntos al equipo ganador y 0 al perdedor.

2.3. Estructuración del código

Para el modelado del problema diseñamos tres módulos: Matriz, MatrizDep y Problema.

2.3.1. Matriz

El módulo Matriz es el que se usará para representar a las matrices de conectividad de redes de páginas web.

Representación interna Como la matriz de conectividad es en general esparsa (dado que cada página web se conecta en proporción con muy pocas páginas), es conveniente utilizar una representación que aproveche esto. Para eso, vamos a usar una representación conocida como Compressed Row Storage (CRS). Para más información sobre este formato puede consultarse [Bar+].

Elegimos este formato porque será especialmente cómodo a la hora de hacer el producto iterativo del método de la potencia, dado que si queremos hacer $P^t x$, es conveniente poder acceder a P^t por filas fácilmente.

Además, nos guardamos la cantidad de links que entran y salen de cada nodo. El primer dato será util para calcular la métrica IN-DEGREE, y el segundo dato será util para saber cuánto valdra P^t_{ij}, dado que es $\frac{1}{n_j}$, donde n_j es la cantidad de nodos salientes del nodo j.

Interfaz La interfaz de Matriz provee las siguientes operaciones:[1]

- Matriz(**ifstream&** in): constructor de la matriz. Recibe un archivo abierto del cual parseará todos los datos que necesita, en el formato adecuado (SNAP).
- **vector<double>** multiplicar(**vector<double>** x): El propósito de esta función es autoexplicativo. Devuelve el resultado del producto Ax, donde A es la matriz de la clase. El algoritmo que utilizamos es el standard para multiplicar por matrices representadas en

[1] Cuando se escribe la aridad de la función la misma puede no coincidir con la notación usada en C++. Esto está bien pues lo único que se busca aquí es dar una orientación de lo que hace cada función y no código preciso.

CRS, y además, como dijimos anteriormente, dividimos cada entrada por la cantidad de nodos salientes del nodo que corresponda.

2.3.2. MatrizDep

El módulo MatrizDep es el que se usará para representar a las matrices de conectividad de ligas deportivas.

Representación interna Como la matriz de conectividad en este caso, a diferencia del anterior, no suele ser esparsa (de hecho, en el caso general podría aproximarse bastante al grafo completo), entonces no fue necesario utilizar ninguna estructura compleja, y utilizamos simplemente la clásica representación de vector de vectores.

Además fue necesario utilizar otra estructura para almacenar los puntajes de acuerdo a otros métodos de rankeo (por ejemplo, el standard) de la liga deportiva correspondiente.

Interfaz La interfaz de MatrizDep provee las siguientes operaciones:

- MatrizDep(**ifstream&** in): constructor de la matriz. Recibe un archivo abierto del cual parseará todos los datos que necesita, en el formato adecuado (el explicitado en el tp).
- **vector<double>** multiplicar(**vector<double>** x): autoexplicativa. Devuelve el resultado del producto Ax, donde A es la matriz de la clase. El algoritmo que utilizamos es el común y corriente, el mismo que se utiliza al hacer cuentas a mano con matrices.

2.3.3. Problema

Problema es un módulo que engloba todo lo relacionado al modelado y a la resolución de cada caso particular.

Adicionalmente, tenemos algunas funciones auxiliares:

Interfaz La interfaz de Problema provee funciones utilizadas tanto para el modelado de rankings de páginas web como para el modelado de rankings en ligas deportivas.

Las siguientes operaciones en lo que concierne a la resolución de problemas relacionados con el modelado de páginas web:[2]

- **vector<double>** pagerank(**Matriz** p_trans, **double** c, **double** tolerancia) : Se ocupa de aplicar el algoritmo de PageRank sobre la matriz `p_trans`, utilizando el algoritmo descripto por [Kam+03, Algoritmo 1]. Además, se testea luego de cada iteración si la diferencia (en norma 1[3]) es menor que la tolerancia. En ese caso, se termina de iterar y se considera que el vector resultante de la iteración actual es el resultado.

[2] Cuando se escribe la aridad de la función la misma puede no coincidir con la notación usada en C++. Esto está bien pues lo único que se busca aquí es dar una orientación de lo que hace cada función y no código preciso.

[3] $||x||_1 = \sum_i |x_i|$

- **vector<uint>** indeg(**Matriz** p_trans): Devuelve el resultado de aplicarle el método de rankeo indegree a la red del problema.

  A continuación siguen los métodos utilizados para el modelado de rankings de ligas deportivas

- **vector<double>** metodopot(**MatrizDep** p, **double** tolerancia): Método de la potencia *vanilla*. Es el que se utilizará para resolver el problema de las ligas deportivas. Obtiene una sucesión de vectores, y cuando su diferencia de norma 1 es menor que la tolerancia pasada como parámetro se termina de iterar.

2.4. Experimentación

3. Resultados y discusión

3.1. PageRank y páginas web

Ahora procederemos a analizar, experimentar y discutir los métodos que conciernen al rankeo de páginas Web. Recordemos que en este trabajo se utilizó el modelo PageRank para modelar el rankeo de páginas web utilizando cadenas de Markov y se implementó una versión optimizada especialmente para este problema, presentada en [Kam+03]

3.1.1. Convergencia de PageRank

Para los experimentos de velocidad de convergencia y rendimiento en general de PageRank utilizamos los datasets de [Sna]. Para eso, elegimos tres datasets de diferentes tamaños y coeficiente de conectividad [4], web-Stanford, web-NotreDame y web-Google.

Dataset	Vértices	Aristas
web-Stanford	281.903	2.312.497
web-NotreDame	325.729	1.497.134
web-Google	916.428	5.105.039

Lo primero que se puede decir de este Datasets es que la diferencia de cantidad de vértices de cada uno nos va a permitir analizar como varía según el tamaño de la red.

Además, notemos como el dataset web-Stanford está mucho más conectado que el dataset web-NotreDame, dado que tiene menos vértices pero más aristas. Creemos que esto va a impactar negativamente en la performance, dado que sobre una red poco conectada el algoritmo va a converger más rápido.

Eso se puede notar fácilmente si se piensa en el grafo que es totalmente disconexo. En ese caso, todas las entradas de la matriz van a valer $\frac{1}{n}$, y el método de la potencia va a converger en un paso.

Otra cosa que podemos pronosticar (en parte porque en [Kam+03] pasa eso) es que entre más cercano sea c a 1 (es decir, el navegante aleatorio se teletransporta menos), más lento va a converger.

Al igual que antes, para entender intuitivamente que está pasando, podemos analizar el caso extremo $c = 0$. En este caso, la probabilidad de que el navegante se teletransporte es siempre 1, por lo que volvemos al caso anterior en el que todos los coeficientes de la matriz van a valer $\frac{1}{n}$, por lo que se convergerá en un paso.

Concluidas las hipótesis relacionados a la convergencia del método, procedamos a ver los resultados de los experimentos.

[4] Con esto nos referimos a que tan conectado está el grafo, esto puede medirse por ejemplo como $\frac{2e}{v(v-1)}$, donde e son las aristas del grafo y v los vertices, dado que el grafo completo tiene $\frac{v(v-1)}{2}$ aristas.

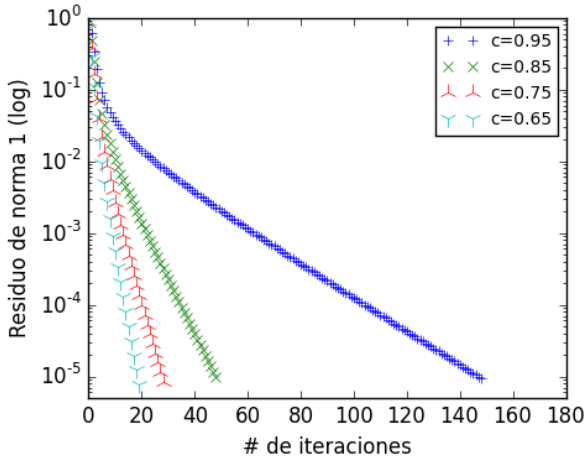

Figura 1: Comparación de la velocidad de convergencia del el método para el dataset web-Stanford.

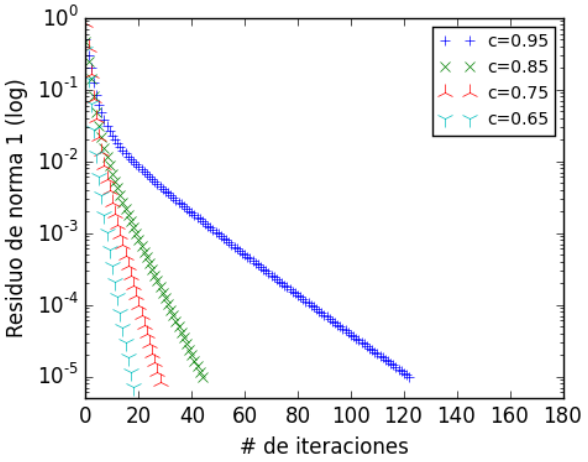

Figura 2: Comparación de la velocidad de convergencia del el método para el dataset web-NotreDame.

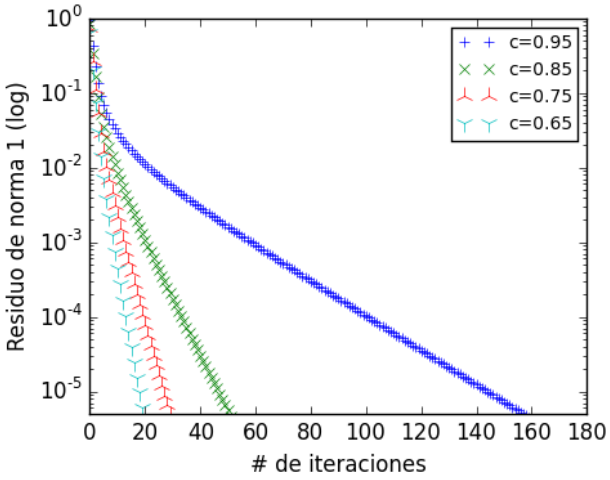

Figura 3: Comparación de la velocidad de convergencia del el método para el dataset web-Google.

Primero notemos algo más o menos sorprendente: la cantidad de iteraciones requeridas por el método no varía demasiado, sobre todo para valores chicos de c. Creemos que esto se debe a propiedades generales del método de la potencia, que sin embargo son díficiles de analizar y están fuera del alcance de este trabajo.

Se puede notar además que para los valores más pequeños de c la cantidad de iteraciones correspondientes a cada dataset es prácticamente igual.

Una cosa importante para decir, de la cual vamos a hablar en profundidad más adelante, es que aunque la cantidad de iteraciones sea parecida, el costo de cada iteración es mayor a más grande es la matriz y a más valores no nulos tiene, por lo que las figuras 1, 2 y 3 no deben intepretarse como el tiempo que requiere el algoritmo para correr.

Sin embargo, una cosa interesante a analizar es qué c usar. Como vimos, usar un c pequeño disminuye el tiempo de cómputo requerido. Sin embargo, disminuir el c también puede impactar en el resultado. Dado que el factor de teletransportación $1 - c$ es más alto, de alguna manera el resultado es menos significativo, debido a que se hace más uniforme, como explicamos

antes. En [Cha+02], por ejemplo, se sugiere que c debería ser elegido basado en la conectividad del grafo.

Otro experimento muy interesante que realizamos es cambiar el criterio de parada del algoritmo iterativo. En particular, lo que hicimos fue cambiar la norma con la que medimos la diferencia entre dos vectores de sucesivas iteraciones. Para ello utilizamos las normas $||-||_1$, $||-||_2$ y $||-||_\infty$.

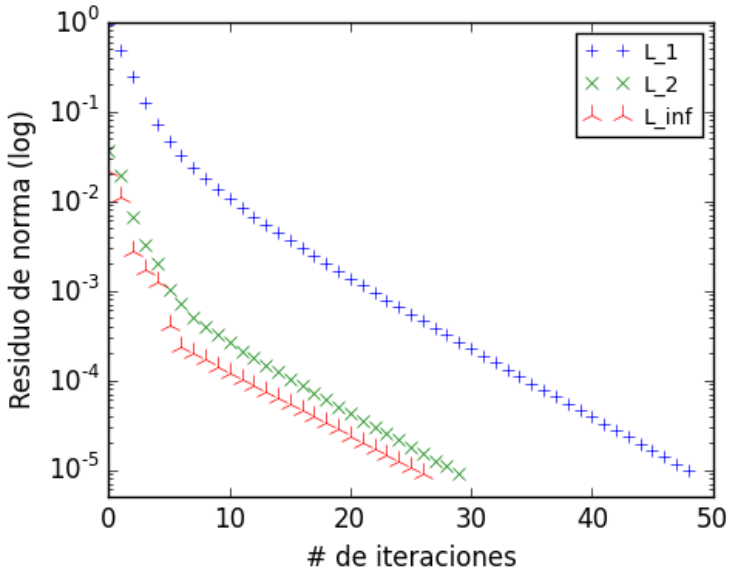

Figura 4: Comparación de la velocidad de convergencia del el método para el dataset web-Stanford, variando la norma del criterio de parada.

Como se observa en la figura 4, cuando se utiliza la norma 1 para el criterio de parada, se requieren más iteraciones para terminar. Esto se debe a que la norma 1 es la más *exigente*.

Se puede probar fácilmente que para todo vector x, $||x||_1 \geq ||x||_2 \geq ||x||_\infty$ y esto explica claramente los resultados del experimento.

3.1.2. Rendimiento de PageRank

A continuación analizaremos el rendimiento del algoritmo de PageRank implementado en este trabajo, es decir, el de [Kam+03]. Para ello, utilizaremos los mismos datasets que en la sección anterior.

Dataset	Vértices	Aristas	Tiempo (segundos)
web-Stanford	281.903	2.312.497	7.81882
web-NotreDame	325.729	1.497.134	3.38163
web-Google	916.428	5.105.039	23.7367

Figura 5: Comparación de los tiempos que requiere el algoritmo para distintos datasets. Fue ejecutado con $c = 0{,}85$ y $tolerancia = 0{,}00001$. Se tomó el promedio de 20 mediciones.

Podríamos empezar a sacar conclusiones apresuradas sobre estos datos, pero para que

sean más significativos, preferimos realizar otro experimento antes de comenzar el análisis. Este experimento consiste en dividir el algoritmo en 2: primero el armado de la matriz (recordemos que como representamos la matriz en CRS este no es un paso trivial) y luego el paso de método de la potencia optimizado presentado en [Kam+03].

Esto nos permitirá ver mejor el problema y tener datos más detallados, con el objetivo de poder analizar mejor que es lo que sucede.

Dataset	Tiempo total	Armado de matriz	Método de la potencia
web-Stanford	7.81882	1.8968	5.91098
web-NotreDame	3.38163	0.978923	2.42743
web-Google	23.7367	4.30584	18.9483

Figura 6: Comparación de los tiempos que requiere el algoritmo para distintos datasets. Fue ejecutado con $c = 0{,}85$ y $tolerancia = 0{,}00001$. Se tomó el promedio de 20 mediciones. Todos los tiempos son en segundos. Las sumas probablemente no den bien porque fueron tomadas en diferentes mediciones.

Ahora, teniendo toda la información que nos proveen la figura 5, 6 y los experimentos de la sección anterior, podemos hacer un análisis completo y riguroso.

Podemos concluir primero que el paso más costoso del algoritmo es el método de la potencia. Justamente por eso [Kam+03] se ocupa de intentar optimizar ese paso, dado que optimizarlo va a impactar fuertemente en el rendimiento general de la aplicación.

Otro aspecto interesante para ver es, como aunque las iteraciones requeridas para cada dataset son bastante similares, el costo de cada operación es muy distinto, pues el tamaño de los vectores varía, y las entradas no nulas de la matriz son distintas.

De hecho, este experimento nos permite confirmar que el paso más costoso es realizar el producto de la matriz y el vector $P^t x^{(k)}$, ya que se puede ver como el método de la potencia en el caso de web-NotreDame tarda menos que en el caso de web-Stanford, aunque los vectores sean más largos.

Esto se debe a que la matriz de web-NotreDame es más esparsa, y al multiplicar Ax con A representada en CRS, el costo resulta lineal en la cantidad de elementos no nulos de la matriz. Esta última está intimamente relacionada con la cantidad de aristas del grafo. De esta manera se explica fácilmente el comportamiento a priori extraño de el algoritmo sobre estos datasets.

3.2. PageRank y ligas deportivas

4. Conclusiones

5. Apéndices

5.1. Proposiciones

5.1.1. Proposición 1

Si $P \in R^{n \times n}$ es una matriz de transición, es decir $0 \leq P_{ij} \leq 1$, y $\sum_j P_{ij} = 1 \; \forall i \in \{1, ..., n\}$, entonces el mayor autovalor en módulo de P^t es 1 o -1.

Demostración Primero veamos que si λ es autovalor de P^t, entonces $|\lambda| \leq 1$.

Vale que $\rho(P^t) \leq ||P^t||$, $\rho(P^t)$ el radio espectral y $||-||$ cualquier norma inducida. En particular, si tomamos la norma 1, $||P^t||_1 = 1$, pues todas las columnas suman 1, pues P es de transición. Entonces $|\lambda| \leq \rho(P^t) \leq 1$.

Ahora, como las filas de P suman 1, si multiplico $P(1, ..., 1)^t = (1, ..., 1)^t$. Es decir que 1 es autovalor de P. Entonces, como P y P^t tienen los mismos autovalores, listo.

Referencias

[Bar+] Richard Barrett y col. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. URL: http://www.netlib.org/templates/templates.pdf.

[BF11] R. Burden y D. Faires. *Numerical Analysis*. Brooks/Cole, 2011.

[BL06] Kurt Bryan y Tanya Leise. "The Linear Algebra behind Google". En: *SIAM Review* 48.3 (2006), págs. 569-581.

[BP98] Sergey Brin y Lawrence Page. "The anatomy of a large-scale hypertextual Web search engine". En: *Computer Networks and ISDN Systems* 30.1-7 (abr. de 1998), págs. 107-117. ISSN: 01697552. DOI: 10.1016/S0169-7552(98)00110-X. URL: http://linkinghub.elsevier.com/retrieve/pii/S016975529800110X.

[Cha+02] Soumen Chakrabarti y col. "The Structure of Broad Topics on the Web". En: *Proceedings of the 11th International Conference on World Wide Web*. WWW '02. Honolulu, Hawaii, USA: ACM, 2002, págs. 251-262. ISBN: 1-58113-449-5. DOI: 10.1145/511446.511480. URL: http://doi.acm.org/10.1145/511446.511480.

[Data] http://www.cs.toronto.edu/~tsap/experiments/datasets/.

[Datb] *DataHub*. http://datahub.io.

[DB74] G. Dahlquist y A. Bjork. *Numerical Methods*. Prentice-Hall, 1974.

[GMA08] Angela Y. Govan, Carl D. Meyer y Rusell Albright. "Generalizing Google's PageRank to Rank National Football League Teams". En: *Proceedings of SAS Global Forum 2008*. 2008.

[GVL96] G. Golub y C. Van Loan. *Matrix Computations*. The John Hopkins University Press, 1996.

[Kam+03] Sepandar D. Kamvar y col. "Extrapolation methods for accelerating PageRank computations". En: *Proceedings of the 12th international conference on World Wide Web*. WWW '03. ACM, 2003, págs. 261-270. ISBN: 1-58113-680-3. DOI: 10.1145/775152.775190. URL: http://doi.acm.org/10.1145/775152.775190.

[Kle99] Jon M. Kleinberg. "Authoritative Sources in a Hyperlinked Environment". En: *J. ACM* 46.5 (sep. de 1999), págs. 604-632. ISSN: 0004-5411. DOI: 10.1145/324133.324140. URL: http://doi.acm.org/10.1145/324133.324140.

[Sna] *Stanford Large Network Dataset Collection*. http://snap.stanford.edu/data/#web.