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 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 relacionadas 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.

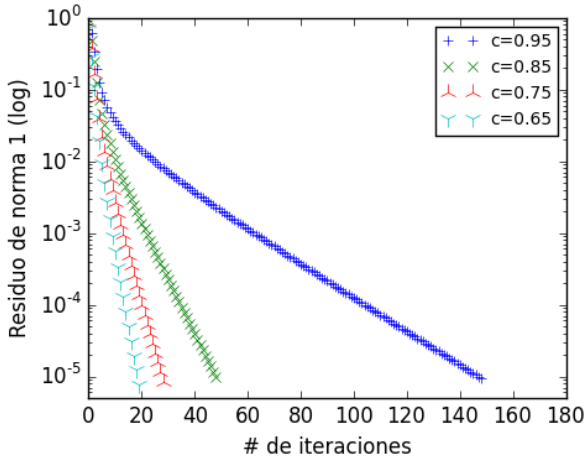

Figura 1: Comparación de la velocidad de convergencia del el método para el dataset web-Stanford.

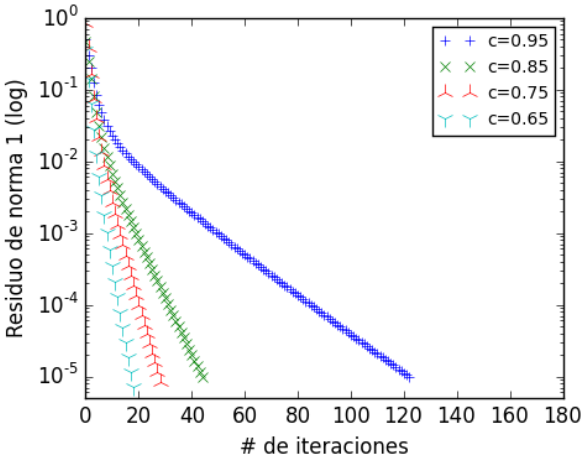

Figura 2: Comparación de la velocidad de convergencia del el método para el dataset web-NotreDame.

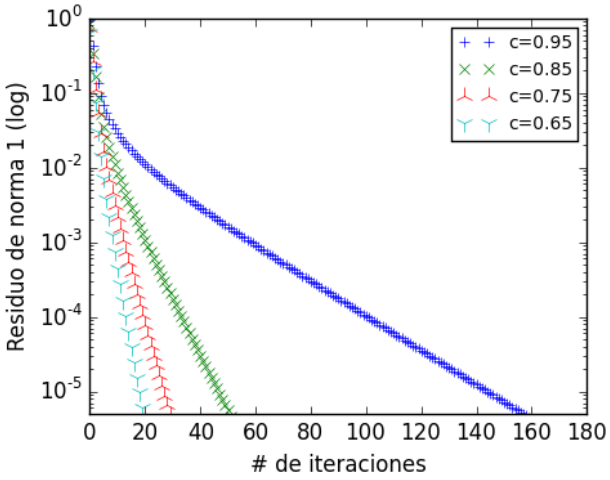

Figura 3: Comparación de la velocidad de convergencia del el método para el dataset web-Google.

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 $||-||_\infty$.

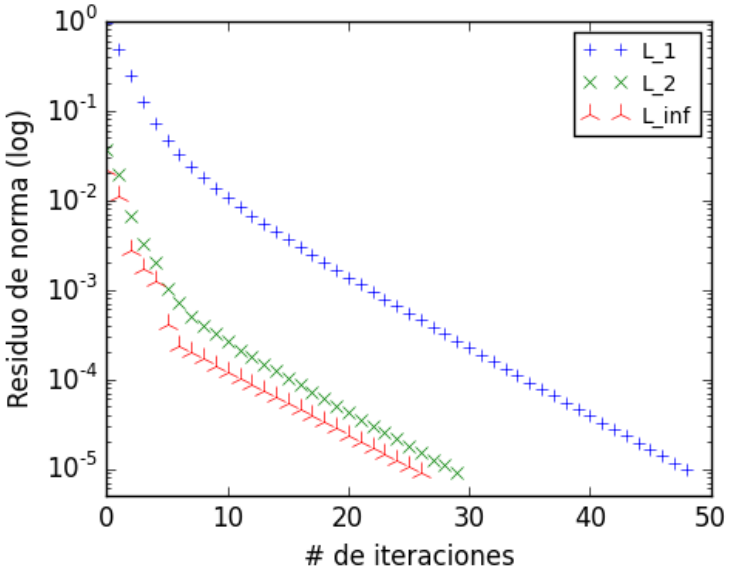

Figura 4: Comparación de la velocidad de convergencia del el método para el dataset web-Stanford, variando la norma del criterio de parada.

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

El conjunto de datos que consideramos para esta sección es el correspondiente al actual torneo de primera división del fútbol argentino. Dicha liga cuenta con las siguientes características relevantes al problema que queremos tratar:

1. Hay un total de 30 equipos que juegan todos una vez entre sí. Además, cada equipo juega un partido adicional con su clásico rival.

2. Los empates son un resultado posible, e incluso muy frecuente. En nuestra implementación de GeM, la posición que tomamos frente a esta situación (que no está contemplada en [GMA08]) es la de no hacer nada: es decir, si los equipos i y j empataron (o más generalmente, la diferencia absoluta de goles entre todos los partidos que jugaron es 0), entonces en las posiciones (i, j) y (j, i) de nuestra matriz de adyacencia ponemos 0. Consideramos que esta medida es más que razonable, pues en definitiva lo que busca el algoritmo GeM es establecer una *relación de fuerza* entre los equipos a partir de la diferencia de goles que se sacan, la cual, en el caso concreto del empate, es nula.

3. El sistema de puntuación estándar es: 3 puntos por partido ganado, 1 por empate y 0 por derrota.

4. Al momento de realizarse el presente trabajo, sólo se ha jugado hasta la fecha 26 del torneo. Por lo tanto sólo analizaremos los datos hasta la misma.

A diferencia de lo que sucedía en la sección (3.1), aquí no nos interesa realizar un análisis sobre la performance o la convergencia del método, sino más bien un análisis sobre la *calidad* (en algún sentido) de los rankings generados por el mismo, contrastándolos con los reales. Por lo tanto las hipótesis irán en esa dirección. A continuación las enlistamos.

- No todos los goles tendrán el mismo peso. Esto es lógico si consideramos la naturaleza del método: si el equipo A le gana al equipo B, que está invicto hasta el momento, por X goles, esto impactará mucho más en el rankeo de A que si le hubiera ganado por X goles a C, un equipo que pierde usualmente. Esto ya plantea una diferencia importante con el método de puntuación estándar, donde en cualquiera de los dos casos A mejoraría su puntaje de la misma manera.

- Algo interesante para analizar es la influencia de los empates en los dos tipos de rankeos. Como ya se dijo, un empate en nuestra implementación de GeM no altera la relación de fuerzas entre los equipos. De hecho, a los fines prácticos, un empate es equivalente a que ambos equipos nunca hayan jugado entre sí (en nuestro modelo), por lo que es claro que la matriz de GeM antes o después de un empate es la misma. Sin embargo, con el método estándar los empates sí que modifican el ranking. No alteran la posición relativa de los equipos que empataron, pero claramente puede mejorar la posición de los dos equipos frente al resto (ya que ambos sumaron un punto). Ante esta situación, podemos suponer que los empates van a tener un efecto no despreciable en los resultados reales, siendo una potencial fuente de diferencias con la versión GeM.

- Para valores de c [5] muy chicos los puntajes de los equipos deberían tender a homogeneizarse, generando rankeos poco deseables considerando la *performance* de los equipos. Como contrapartida, para valores de c muy altos, el rankeo debería ser más justo.

3.2.1. Análisis cualitativo de los resultados

	Método GeM		Método Estándar	
Posición	Equipo	Puntaje	Equipo	Puntaje
1	Boca Juniors	0.0859387	Boca Juniors	58
2	Aldosivi	0.0653494	San Lorenzo	54
3	River Plate	0.063251	Rosario Central	52
4	San Lorenzo	0.0624152	Racing Club	49
5	Rosario Central	0.048369	River Plate	48
6	Racing Club	0.0480862	Independiente	45
7	San Martín (SJ)	0.0439307	Banfield	43
8	Quilmes	0.0421091	Belgrano	43
9	Newell's Old Boys	0.038184	Tigre	42
10	Vélez Sarsfield	0.0374806	Estudiantes (LP)	42
11	Independiente	0.0363642	Lanús	41
12	Belgrano	0.0362427	Quilmes	39
13	Gimnasia y Esgrima (LP)	0.0334854	Unión	38
14	Banfield	0.0308403	Gimnasia y Esgrima (LP)	37
15	Estudiantes (LP)	0.029259	Newell's Old Boys	33
16	Unión	0.0287605	San Martín (SJ)	32
17	Tigre	0.0272169	Sarmiento	30
18	Sarmiento	0.0258276	Aldosivi	30
19	Lanús	0.0251797	Temperley	29
20	Huracán	0.0247069	Argentinos Juniors	29
21	Defensa y Justicia	0.0239225	Olimpo	29
22	Olimpo	0.0224303	Defensa y Justicia	27
23	Arsenal	0.0208047	Huracán	26
24	Godoy Cruz	0.0174517	Vélez Sarsfield	26
25	Temperley	0.0158447	Godoy Cruz	25
26	Crucero del Norte	0.0156778	Colón	24
27	Argentinos Juniors	0.0153485	Arsenal	23
28	Nueva Chicago	0.0142273	Atlético de Rafaela	22
29	Atlético de Rafaela	0.0112011	Nueva Chicago	17
30	Colón	0.010094	Crucero del Norte	14

Tabla 1 Ranking correspondiente a la fecha 26 usando el método GeM, para un valor de $c = 0{,}85$, y el método estándar.

Al observar la tabla 1, notamos como primera gran sorpresa que Aldosivi (8 ganados-12 perdidos) aparece segundo en nuestro ranking GeM, siendo que en la realidad ocupa la

[5] Recordar que 1-c era la probabilidad de teletransportación, o en este caso más concretamente, la probabilidad de que un equipo cualquiera pueda ganar contra cualquier otro.

posición 18. Este resultado, *a priori* extraño, parece cobrar sentido si consideramos lo sucedido en la fecha 13 del torneo: Aldosivi goleó por 3-0 a Boca, el actual puntero según nuestro ranking (y también según el ranking oficial). En efecto, si vemos la tabla 2 podemos apreciar como la contundente derrota de Boca (que hasta ese momento estaba primero) permitió catapultar a Aldosivi del cuarto al primer puesto, con una diferencia respecto del segundo particularmente amplia.

Vale destacar, que en la realidad, este resultado tuvo un impacto significativamente menor: Boca siguió estando primero, con la pequeña diferencia de que fue alcanzado por San Lorenzo, mientras que Aldosivi se mantuvo lejos de la punta, en el puesto 8.

En los sucesivos partidos, Aldosivi sufrió una larga lista de derrotas, sin embargo, como vimos en la tabla 1 esto apenas si perjudicó su posición. Podemos decir que esto es producto del *batacazo* que dió frente a Boca, que mantuvo durante casi la totalidad del torneo su hegemonía por sobre los otros equipos. Dicho de otra forma, la suerte de Aldosivi quedó en algún punto supeditada a la de Boca, pues entre mejor le fuera a este último, más significativos serían los 3 goles que le había hecho para su propio puntaje. Esto sucede a tal punto que, a la fecha siguiente, Boca pierde 2-0 con Velez pero en lugar de perjudicarlo genera que, posiblemente en combinación con el hecho de que Aldosivi también perdiera por una diferencia de un gol, y de que River (quien había perdido el superclásico) ganara por 2, vuelva a estar puntero, desplazando a Aldosivi al tercer puesto. Es decir que lo importante para rescatar de este úlimo hecho, es que que al finalizar la fecha 14 los 3 goles de Aldosivi ante Boca perdían un poco de peso, pues ahora, en total, este tenía 2 goles más en su contra.

Luego, por todo lo dicho hasta aquí, creemos que nuestra primer hipótesis, concerniente a la distinta importancia que se le asigna a los goles según la performance del equipo que los recibe, queda verificada.

	Fecha 12		Fecha 13	
Posición	Equipo	Puntaje	Equipo	Puntaje
1	Boca Juniors	0.108944	Aldosivi	0.117217
2	River Plate	0.0815019	Boca Juniors	0.0892615
3	Rosario Central	0.0735106	River Plate	0.0657072
4	Aldosivi	0.0546487	Rosario Central	0.0608375
5	San Lorenzo	0.0511854	Banfield	0.0546665
6	Racing Club	0.046461	San Lorenzo	0.0497559
7	Belgrano	0.0464054	Gimnasia y Esgrima (LP)	0.0478387
8	Banfield	0.0440866	Vélez Sarsfield	0.0426006
9	San Martín (SJ)	0.0433587	Racing Club	0.0418281
10	Newell's Old Boys	0.0353829	Newell's Old Boys	0.036451
11	Lanús	0.0320765	Belgrano	0.0356502
12	Gimnasia y Esgrima (LP)	0.029684	San Martín (SJ)	0.0350081
13	Sarmiento	0.0263866	Lanús	0.0346346
14	Arsenal	0.0252073	Estudiantes (LP)	0.025968
15	Huracán	0.0241481	Argentinos Juniors	0.0241652
16	Estudiantes (LP)	0.024007	Huracán	0.0215508
17	Tigre	0.023322	Godoy Cruz	0.0211769
18	Vélez Sarsfield	0.0229058	Sarmiento	0.0210192
19	Independiente	0.0224838	Arsenal	0.0200022
20	Argentinos Juniors	0.0221471	Unión	0.0199389
21	Temperley	0.0212702	Defensa y Justicia	0.019309
22	Unión	0.0212572	Tigre	0.0179675
23	Defensa y Justicia	0.0206203	Independiente	0.0170322
24	Quilmes	0.0168138	Temperley	0.0158759
25	Godoy Cruz	0.0166558	Quilmes	0.0158634
26	Colón	0.0159257	Crucero del Norte	0.0127548
27	Olimpo	0.0135905	Colón	0.0106429
28	Atlético de Rafaela	0.0130908	Olimpo	0.00977677
29	Crucero del Norte	0.0127524	Atlético de Rafaela	0.00877535
30	Nueva Chicago	0.0101696	Nueva Chicago	0.00672374

Tabla 2 Posiciones de las fechas 12 y 13 calculadas con GeM y $c = 0{,}85$.

A continuación veamos, con un experimento muy simple pero ilustrativo, como el tomar un modelo que ignora los empates frente a otro que les da cierta importancia puede afectar sensiblemente la tabla de posiciones. Para eso veamos que sucede al considerar los resultados de la primera fecha que se muestran en la tabla 3. Rápidamente, podemos notar que en ambos casos los equipos que comparten el primer lugar son los mismos: los que ganaron su respectivo partido. Sin embargo, en la versión GeM tanto los equipos que empataron como los que perdieron tienen todos la última posición, cosa que no ocurre en el modelo estándar, donde los equipos que empataron están segundos mientras que los que perdieron están últimos. Esto en principio parece un rankeo un poco injusto para los que empataron, que son puestos al mismo nivel de los perdedores.

Vale decir que esta es una situación muy particular: al ser la primer fecha, la matriz

de adyacencias no solo será muy esparsa, sino que al menos la mitad de sus filas serán 0 (correspondientes a los equipos que ganaron o empataron)mientras que de la otra mitad a lo sumo tendrán un coeficiente no nulo (correspondientes a los equipos que perdieron). Debido a esto, es que, por ejemplo, todos los equipos que ganaron tienen el mismo puntaje a pesar de que hayan ganado por distintas diferencias (donde la más grande fue de River por 3 goles). Por lo tanto, este debe considerarse como un ejemplo particular donde el dar puntos a los empates genera una situación un poco más justa, pero no debe extrapolarse una conclusión apresurada y afirmar que esto valga para cualquier caso. La experimentación necesaria, así como el análisis que se requiere, para determinar esas conclusiones son de una dificultad no trivial (especialmente considerando que en el caso de GeM el puntaje de un equipo no depende sólo de sus resultados sino también de los de sus adversarios), por lo que no lo abordaremos en más profundidad en este trabajo.

	Método GeM		Método Estándar	
Posición	Equipo	Puntaje	Equipo	Puntaje
1	River Plate	0.0450669	Lanús	3
2	Lanús	0.0450669	Vélez Sarsfield	3
3	Vélez Sarsfield	0.0450669	Unión	3
4	Temperley	0.0450669	Temperley	3
5	Rosario Central	0.0450669	San Lorenzo	3
6	Independiente	0.0450669	Rosario Central	3
7	Estudiantes (LP)	0.0450669	River Plate	3
8	Argentinos Juniors	0.0450669	Independiente	3
9	Boca Juniors	0.0450669	Estudiantes (LP)	3
10	Belgrano	0.0450669	Defensa y Justicia	3
11	Defensa y Justicia	0.0450669	Argentinos Juniors	3
12	Unión	0.0450669	Belgrano	3
13	San Lorenzo	0.0450669	Boca Juniors	3
14	Olimpo	0.0243606	Crucero del Norte	1
15	Tigre	0.0243606	Tigre	1
16	Arsenal	0.0243606	San Martín (SJ)	1
17	Sarmiento	0.0243606	Godoy Cruz	1
18	San Martín (SJ)	0.0243606	Newell's Old Boys	0
19	Atlético de Rafaela	0.0243606	Nueva Chicago	0
20	Banfield	0.0243606	Olimpo	0
21	Racing Club	0.0243606	Quilmes	0
22	Quilmes	0.0243606	Racing Club	0
23	Colón	0.0243606	Aldosivi	0
24	Nueva Chicago	0.0243606	Colón	0
25	Newell's Old Boys	0.0243606	Banfield	0
26	Aldosivi	0.0243606	Huracán	0
27	Huracán	0.0243606	Sarmiento	0
28	Godoy Cruz	0.0243606	Atlético de Rafaela	0
29	Gimnasia y Esgrima (LP)	0.0243606	Arsenal	0
30	Crucero del Norte	0.0243606	Gimnasia y Esgrima (LP)	0

Tabla 3 Ranking correspondiente a la fecha 1 usando el método GeM, para un valor de $c = 0{,}85$, y el método estándar. Los sombreados verde, amarillo y rojo señalan la posición del equipo que puede ser 1º, 2º o 3º respectivamente.

Finalmente, analicemos el efecto de variar el valor del c. Para esto consideremos los resultados expuestos en la tabla 4, que nos muestra el rankeo correspondiente a la fecha 26 usando GeM con dos valores extremos de c: 0,1 y 0,99. Recordar que el primero es el caso en que la probabilidad de que un equipo pierda contra otro depende mucho menos de los resultados previos y tiende a ser la misma para todos, mientras que el segundo es el opuesto: los resultados previos determinan casi totalmente la probabilidad de que un equipo pierda con otro.

Curiosamente vemos que no hay diferencias grandes entre ambos rankings. De hecho en muchos casos, la posición del equipo no cambió. Aunque en efecto, como habíamos supuesto, el c pequeño hizo que las puntuaciones se homogeneizaran, vemos que en la práctica la ventaja que le da a un equipo ser más goleador que otros y ganarle a equipos importantes sigue siendo

lo suficientemente significativa como para que el ranking no presente resultados absurdos (como que un equipo que rankeaba de la mitad para abajo de la tabla con un c alto, pase a estar en los primeros lugares con uno bajo). De hecho, más bien pareciera que pasa lo contrario.

Si volvemos a considerar el particular caso de Aldosivi, vemos que al tomar un c más cercano a 0 su posición en la tabla empeoró en cuatro posiciones (respecto de las versiones con $c = 0{,}85$ $c = 0{,}99$). Esto es muy razonable si tenemos en cuenta que el buen rankeo de Aldosivi estaba fuertemente basado en un resultado aislado (un *outlier* si se quiere) como fue su amplia victoria sobre Boca, el equipo que mejor *performance* tuvo en el campeonato. No sólo eso, sino que también le gano por 1-0 a San Lorenzo, otro de los principales equipos. Al reducir la incidencia de los resultados de todos los partidos sobre el cálculo de los puntajes, Aldosivi recibe muchos menos "votos" de Boca o San Lorenzo, pues los mismos ahora se reparten más equitativamente entre todos los equipos, y como Aldosivi tuvo pocas victorias en general se ve perjudicado con este cambio.

De la misma forma, pasa algo muy parecido con Vélez (el equipo que más posiciones bajó al considerar el c más chico) que también le ganó a Boca (2-0) y al mismo Aldosivi (2-0), pero en términos generales apenas pudo ganar 6 partidos y perdió 13.

De hecho, la tendencia general es que la subtabla correspondiente a $c = 0{,}1$ tiende a asemejarse mucho más al ranking oficial, que puede verse en la tabla 1, de que lo hace la subtabla correspondiente a $c = 0{,}99$. Recordar que el método estándar no considera las diferencias de goles, y sí considera la cantidad de victorias, derrotas y empates. Podemos pensar entonces, basándonos en lo que venimos viendo, que un valor más chico de c genera que la cantidad de victorias pese más y al mismo tiempo reduce la importancia de goleadas muy abultadas, lo cual puede ser útil para reducir el impacto de resultados aislados como el de Aldosivi o el de Velez.

En resumen, llegamos a la conclusión de que, contra lo que habíamos supuesto en las hipótesis, la utilización de un c bajo puede dar resultados más sensatos que la de uno muy alto, pues evita que equipos que muestran un bajo rendimiento logren un excelente rankeo sólo por *batacazos* aislados. Por el contrario, el c demasiado alto acrecenta estas diferencias. Particularmente, encontramos que, al menos para este caso, un valor de $c = 0{,}35$ resultado bastante apropiado, basándonos en que, en sentido decreciente, es el primer valor para el cual Aldosivi pasa a estar por debajo de Racing (14 ganados-5 perdidos), que mostró un mejor nivel a lo largo del torneo. No ponemos el ranking obtenido para este valor dado que no aporta nada nuevo a la discución.

	$c = 0{,}1$		$c = 0{,}99$	
Posición	Equipo	Puntaje	Equipo	Puntaje
1	Boca Juniors	0.0391469	Boca Juniors	0.0945905
2	San Lorenzo	0.0370791	Aldosivi	0.0744865
3	River Plate	0.0368742	River Plate	0.0675502
4	Racing Club	0.0357606	San Lorenzo	0.0656075
5	Rosario Central	0.0353159	Rosario Central	0.0505593
6	Aldosivi	0.0347251	Racing Club	0.049052
7	Quilmes	0.0344085	San Martín (SJ)	0.0468815
8	Independiente	0.0341835	Quilmes	0.0433725
9	Belgrano	0.034173	Vélez Sarsfield	0.0410949
10	San Martín (SJ)	0.033846	Newell's Old Boys	0.039274
11	Newell's Old Boys	0.0337048	Independiente	0.0355528
12	Gimnasia y Esgrima (LP)	0.033465	Belgrano	0.0354938
13	Lanús	0.0334376	Gimnasia y Esgrima (LP)	0.0337113
14	Banfield	0.0332383	Banfield	0.0304023
15	Tigre	0.0331063	Estudiantes (LP)	0.0290798
16	Estudiantes (LP)	0.0326988	Unión	0.0283743
17	Unión	0.0325988	Tigre	0.0260657
18	Sarmiento	0.0325042	Sarmiento	0.0245763
19	Vélez Sarsfield	0.0324059	Defensa y Justicia	0.023483
20	Arsenal	0.0322979	Huracán	0.0229007
21	Huracán	0.0322357	Lanús	0.0220692
22	Defensa y Justicia	0.0318777	Olimpo	0.0215218
23	Olimpo	0.0318753	Arsenal	0.0179464
24	Argentinos Juniors	0.0317518	Godoy Cruz	0.0142913
25	Godoy Cruz	0.0317468	Crucero del Norte	0.012915
26	Temperley	0.0315835	Temperley	0.0124947
27	Crucero del Norte	0.0313297	Argentinos Juniors	0.0115357
28	Nueva Chicago	0.0310144	Nueva Chicago	0.0114375
29	Atlético de Rafaela	0.0308366	Atlético de Rafaela	0.00759916
30	Colón	0.0307777	Colón	0.00608033

Tabla 4 Ranking correspondiente a la fecha 26 usando el método GeM para dos valores de c distintos: 0,1 y 0,99

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.