DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico Nº 2

"Ohhh solo tiran π-edras..."

Segundo cuatrimestre de 2015 — Métodos Numéricos

Integrante	LU	Correo electrónico
Frizzo, Franco	013/14	francofrizzo@gmail.com
Martínez, Manuela	160/14	martinez.manuela.22@gmail.com
Rabinowicz, Lucía	105/14	lu.rabinowicz@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Resumen

En este trabajo, se busca solucionar dos problemas sobre elaboración de *rankings*: la clasificación de páginas web para ordenar los resultados de un motor de búsqueda en Internet, y la de equipos según su desempeño en ligas deportivas. Para resolverlos, se presentarán los algoritmos *PageRank* y *GeM*, respectivamente. Ambos proporcionan una solución representando los datos en forma matricial y buscando luego el autovector principal de esta matriz. Se realizan pruebas experimentales sobre estos métodos, variando sus parámetros de entrada, y comparándolos con algoritmos alternativos en términos de eficiencia y efectividad.

Palabras clave: *PageRank*, motores de búsqueda, *rankings*, cálculo de autovectores

Índice

1. Introducción

En la actualidad, Internet ocupa un lugar central en la vida cotidiana de millones de personas, y una de las principales tareas que estas llevan a cabo en línea es la búsqueda de información. A eso se debe el rol central que ocupan los buscadores, y la gran importancia de contar con algoritmos que permitan indexar y clasificar la información disponible en la red y, muy especialmente, seleccionar de este enorme volumen de datos solo aquellos que sean más relevantes para un determinado contexto de búsqueda.

Por otro lado, las partidos de las grandes ligas deportivas movilizan a enormes masas de seguidores a lo largo y a lo ancho de todo el mundo, y además, importantes intereses económicos dependen de sus resultados. Por lo tanto, es de gran interés que estos resultados puedan definirse con criterios justos y objetivos, que no sea preseten a la especulación y sean robustos ante posibles intentos de manipulación.

Si bien a primera vista estos dos problemas pueden parecer poco relacionados, poseen características en común: en ambos casos, se busca dar un orden de importancia a una determinada colección de elementos, organizándolos en un *ranking* en base información estadística que puede obtenerse sobre las relaciones existentes entre ellos.

En este trabajo, se introducirá uno de los métodos que ha sido utilizado con gran éxito para resolver el primero de los problemas mencionados: el algoritmo de *PageRank*, presentado en 1998 por Sergey Brin y Larry Page[1], que constituye una parte esencial del motor del reconocido buscador Google. Tras una introducción teórica a los conceptos matemáticos que fundamentan este método, se presentará una implementación del mismo y se expondrán resultados experimentales con el fin de analizar su eficiencia y su capacidad para resolver el problema planteado.

Por otra parte, y de forma paralela, se procederá a abordar la situación planteada por el problema de la clasificación de equipos en ligas deportivas. Se mostrará que, tal y como se expone en Govan et al.[4], *PageRank* puede ser adaptado sin mayores dificultades para emplearse como criterio de clasificación en este contexto, y se analizará el caso particular de su utilización para elaborar la tabla de posiciones del torneo de Primera Divisón de la Asociación del Fútbol Argentino (AFA), realizando experimentos para evaluar su desempeño en comparación con el sistema de puntajes actual, y la forma en la que su funcionamiento ve afectado por variaciones en los parámetros del algoritmo.

1.1. Conceptos teóricos

En esta sección, enunciamos a modo introductorio los principales conceptos que funcionan como sustento teórico de los métodos numéricos que emplearemos. Nuestro objetivo no es profundizar en los mismos, dado que ya existen excelentes trabajos realizados que abordan cada uno de ellos más en detalle.

Consideremos un sistema que puede tomar diferentes estados, dentro de un conjunto finito de ellos. Un *proceso estocástico* es una sucesión de estos estados. Si consideramos un caso particular de proceso estocástico, en que la probabilidad de que el sistema se encuentre en un estado dado en el momento $k+1$ queda completamente determinada por su estado en el momento k, tenemos una *cadena de Márkov*.

Supongamos que el sistema tiene n estados posibles, y denominemos p_{ij} a la probabilidad de que el sistema pase del estado j al estado i. Entonces podemos acomodar estos valores en una matriz $P \in \mathbb{R}^{n \times n}$, que llamaremos *matriz de transición* de la cadena.

Notemos que todas las entradas de la matriz, dado que indican valores de una probabilidad, son no negativas, y además, para $j = 1, \cdots, n$, tenemos que $\sum_{i=1}^{n} p_{ij} = 1$. A una matriz que cumpla con estas condiciones la denominaremos *matriz estocástica por columnas*, y, como enuncian Bryan y Leise en [2], cumplen con la importante propiedad de tener a 1 como autovalor dominante; es decir, 1 es autovalor y además, para cualquier otro $\lambda \in \mathbb{R}$ autovalor de una de estas matrices, se cumple que $|\lambda| < 1$. Si, además, una matriz estocástica cumple que todas sus entradas son estrictamente positivas, entonces podemos afirmar que el espacio de autovectores asociados al autovalor 1 tiene dimensión 1.

En este trabajo, será de interés para nosotros poder calcular este autovector. Se cuentan con métodos muy diversos para realizarlo, pero muchos de ellos son excesivamente costosos, especialmente cuando las matrices son de dimensiones considerables. Afortunadamente, para matrices con las características que acabamos de mencionar, el autovector principal puede calcularse mediante un método iterativo, muy sencillo de formular, conocido como *método de la potencia*. Este método se basa en el siguiente resultado: si x es el autovector principal de la matriz A, y v es un vector inicial cualquiera, entonces $\lim_{k\to\infty} A^k v = x$. Es decir, partiendo de un vector inicial y multiplicando repetidas veces por la matriz A, el resultado eventualmente convergerá al autovector buscado. Para una demostración de este hecho, como así también una exposición más detallada del proceso, puede consultarse [5, Sección 3]. Como podrá verse en la sección siguiente, y comprobarse luego experimentalmente, este algoritmo resulta muy apropiado para resolver los problemas planteados en este trabajo, y es capaz de aprovechar características propias de las matrices con las que trataremos para mejorar su eficiencia.

2. Desarrollo

2.1. Métodos utilizados

2.1.1. *PageRank*: autovectores para elaborar un *ranking* de páginas web

Cualquier usuario de Internet que haya realizado alguna búsqueda en Google habrá notado que, en la inmensa mayoría de los casos, la información deseada no se encuentra demasiado lejos de los primeros resultados arrojados por el buscador. ¿Cómo es clasificada la ingente cantidad de datos disponibles en la Web para lograr presentar al usuario aquellos que tienen mayor relevancia? El punto clave reside en aprovechar la información que cada página web brinda acerca de las demás a través de los vínculos o *links* existentes entre ellas. La idea en que se basa el criterio adoptado es simple: cuantas más páginas tengan *links* que apunten hacia otra página dada, más alta será la probabilidad de que esta última contenga información relevante; y mucho más lo será si los enlaces provienen de orígenes que son, a su vez, considerados importantes.

La World Wide Web, como cualquier otra red de páginas unidas a través de enlaces, puede ser considerada como un grafo dirigido, con las primeras representadas por los nodos y los segundos, por las aristas. Como punto de partida para elaborar el *ranking*, recurriremos a una versión modificada de la matriz de adyacencia de este grafo, donde para cada página el peso de los enlaces salientes será inversamente proporcional a su cantidad. Denotemos n_j a la cantidad de *links* salientes que posee la página j, y L_i al conjunto de páginas que poseen un link que apunta hacia i. Entonces, si llamamos A a esta matriz, tenemos que $a_{ij} = \frac{1}{n_j}$ si $j \in L_i$, y $a_{ij} = 0$ en caso contrario.

Nuestro objetivo es calcular un *ranking* $x = (x_1, \cdots, x_n)$, donde x_i es el puntaje asignado a la página i. Una manera de hacerlo siguiendo el criterio antes mencionado es planteando el sistema de ecuaciones

$$x_i = \sum_{j \in L_i} \frac{x_j}{n_j} \qquad i = 1, \cdots n$$

Notemos que este sistema corresponde a la representación matricial $Ax = x$. Es decir, el problema se reduce a encontrar un autovector de A asociado al autovalor 1. ¿Podemos garantizar la existencia de una solución a este sistema, y en tal caso, su unicidad?

Analizando la matriz A, podemos notar que, por un lado, $0 \leq a_{ij} \leq 1 (\forall\ i, j)$, y por otra parte, si $n_j > 0$, entonces $\sum_{i=1}^{n} a_{ij} = 1$, y si, por el contrario, $n_j = 0$, entonces $\sum_{i=1}^{n} a_{ij} = 0$. Conceptualmente, cuando una página tiene *links* salientes, los pesos de los mismos suman 1, mientras que las columnas cuyos valores suman 0 corresponden a las páginas que no tienen enlaces salientes (o *nodos colgantes*). Por lo tanto, si consideramos una red sin nodos colgantes, tenemos que A es una matriz estocástica por columnas; esto garantiza la existencia de un autovector asociado al autovalor 1 y, nos dice, además, que cualquier otro autovalor λ de A deberá cumplir $|\lambda| < 1$. El modelo construido representa un proceso de Márkov; como explican Brin y Page[1], este puede interpretarse como el itinerario de un *navegante aleatorio*, que posicionado en una página j cualquiera, se dirige a cualquiera de sus links con probabilidad $\frac{1}{n_j}$.

Para resolver el problema de los nodos colgantes, consideramos que cuando el navegante llega a uno de estos nodos, selecciona equiprobablemente una página cualquiera de la red para continuar su recorrido. Matematicamente esto puede representarse con una nueva matriz $P_1 = A + D$, donde $D = vd^t$, con

$$v_i = \frac{1}{n} \qquad \text{y} \qquad d_i = \begin{cases} 1 & \text{si } n_i = 0 \\ 0 & \text{si no} \end{cases}$$

P_1 sí es una matriz estocástica por columnas, lo cual nos asegura que encontraremos solución. Más aún, dado que 1 es el autovalor principal de P_1, sabemos que podremos hacerlo aplicando el método de la potencia. Ahora bien, es altamente deseable que la solución hallada sea única, para que las páginas queden clasificadas de manera única. Con el sistema tal y como está planteado, no es difícil encontrar ejemplos en los que existe más de una solución posible (ver [2]). Afortunadamente, esto también se resuelve de manera sencilla mediante el agregado al modelo de un *factor de amortiguación* o *teletransportación*[1] c, $0 < c < 1$, que permite definir

$$P_2 = cP_1 + (1-c)E$$

donde $E \in \mathbb{R}^{n \times n}$ tiene $\frac{1}{n}$ en todas sus posiciones. Notemos que, dado que P_1 y E son matrices estocásticas por columnas, P_2, que es un promedio ponderado de ambas, también lo es; y que, además, P_2 tiene todos sus coeficientes estrictamente positivos. Es decir, P_2 representa una cadena de Márkov ergódica, y por lo tanto, el sistema $P_2 x = x$ tiene solución única.

2.1.2. *GeM*: adaptando *PageRank* para clasificar equipos en ligas deportivas

El algoritmo presentado en la sección anterior no solo ha resultado sumamente exitoso para resolver el problema para el que fue ideado; también existen otros problemas que pueden resolverse con la misma idea básica. Como ya mencionamos anteriormente, uno de ellos es la elaboración de *rankings* de equipos para ligas deportivas a partir de los resultados obtenidos en los partidos.

En Govan et al.[4], los autores mencionan algunos métodos empleados para este fin, y los comparan frente a uno desarrollado por ellos mismos: *GeM*, que está fuertemente basado en el algoritmo de *PageRank*. La idea es la siguiente: una temporada de una liga deportiva también puede entenderse como un grafo dirigido. Los nodos representan a los equipos, y una arista entre los nodos i y j quiere decir que el equipo i perdió al menos una vez contra el equipo j. El peso de las aristas es proporcional a la diferencia absoluta en el marcador del partido al que representan (la suma de ellas, si se trata de más de un partido), normalizadas de forma tal que la suma de los pesos de todas las aristas salientes de cada nodo sea igual a 1.

A partir del grafo anteriormente mencionado, puede construirse la matriz A de adyacencia del grafo; y, de la misma manera que en el caso de la clasificación de páginas en buscadores, pueden derivarse de ella las matrices P_1 y P_2 para resolver los problemas de los nodos colgantes (que representan, en este caso, a los equipos invictos) y de la posible existencia de más de una solución. El sistema resultante es completamente análogo al obtenido para calcular *PageRank*, por lo que no detallaremos los pormenores matemáticos que permiten encontrar su solución. Sí deben tenerse en cuenta las inevitables diferencias implementativas que se derivarán de los distintos contextos de uso en los que se desempeñarán ambos algoritmos, que trataremos con más profundidad en la próxima sección.

[1] El nombre *factor de teletransportación* hace referencia aque, desde la interpretación del navegante aleatorio antes mencionada, la alteración que se efectúa al sistema puede interpretarse como agregarle la posibilidad de que en cada paso, con probabilidad c, el navegante se "teletransporte" a una página elegida al azar, independientemente de la existencia de un *link* que conecte esta con su posición actual.

2.2. Implementación

Ambos métodos fueron implementados en lenguaje C++, reutilizando el código en los casos en que fue posible, pero atendiendo a las diferencias en los contextos de aplicación esperados para cada uno, especialmente a la hora de decidir las estructuras de datos utilizadas.

En los dos casos, una vez leídos los datos de entrada, se crea la matriz correspondiente al sistema a resolver. A continuación, partiendo de un vector de probabilidad uniforme ($v \in \mathbb{R}^n$, con $v_i = \frac{1}{n}$ para $i = 1, \cdots, n$), se aplican sucesivas iteraciones del método de la potencia. Como criterio de detención, se estima la convergencia del método en cada iteración; para esto, se calcula la diferencia Manhattan entre el vector obtenido en cada repetición y en la inmediatamente anterior, y se la compara con un umbral de tolerancia, proporcionado como parámetro al momento de la ejecución. El algoritmo se detiene cuando la distancia obtenida está por debajo del umbral.

2.2.1. Implementación de *PageRank*

En el caso del algoritmo *PageRank*, se tuvieron especialmente en cuenta dos hechos: por un lado, el contexto de uso hace esperar que deban manejarse enormes cantidades de datos, por lo que es importante hacer hincapié en la eficiencia de la implementación; por otra parte, en una red con una estructura similar a la de la World Wide Web, cabe esperar que la cantidad de links presentes en cada página sea muy escasa en comparación con el número total de páginas. Esto permite optimizar la forma en la que se almacena la información: en particular, la matriz A correspondiente al sistema, que contiene los datos sobre los *links* de la red, es *esparsa*, es decir, gran parte de sus coeficientes son 0. Dado que las alteraciones posteriores de esta matriz (las efectuadas a fin de solucionar el problema de los nodos colgantes y el de las posibles múltiples soluciones), a partir de las cuales se obtiene la matriz P_2, pueden ser aplicadas *ad-hoc* al momento de operar, se decidió aprovechar la ventaja mencionada y limitarse a almacenar en memoria la matriz A.

La representación elegida fue CSC (*Compressed Sparse Column*). Esta representación consiste en tres vectores. El primero de ellos (`vals`) almacena todos los valores no nulos de la matriz, recorriéndola por columnas; el segundo (`ind_filas`), de igual longitud que el primero, contiene el índice de la fila en que se encuentra el elemento respectivo en `vals`; el tercero (`ptr_cols`), tiene una entrada por columna, y señala en qué posición de `vals` se encuentra el primer valor correspondiente a cada una de ellas. Esta representación fue elegida por sobre DOK (*Dictionary of Keys*) por considerarla más eficiente, tanto en memoria como temporalmente, y por sobre CSR (*Compressed Sparse Row*) porque permite distinguir eficientemente las columnas que solo contienen ceros (que requieren en este caso un tratamiento especial): en la presente implementación, se las representa colocando el valor -1 en la posición correspondiente del vector `ptr_cols`.

El único inconveniente que se encontró con la representación elegida es que resulta muy impráctica a la hora de cargar los datos. Es por eso que como paso intermedio entre la lectura de datos y la creación de la matriz, se utilizó una estructura temporal que se asemeja más al formato de DOK: un vector correspondiente a las columnas, cuyas entradas son a su vez vectores conteniendo los índices de la columna respectiva donde deberá colocarse un valor no nulo.

La implementación del método de la potencia no presentó grandes complicaciones: consiste en un ciclo que inicia con un vector con todas sus componentes iguales a $\frac{1}{n}$ y lo multiplica, en cada iteración, por la matriz del sistema. Implementar este producto implicó algo más de

complejidad, ya que la matriz por la que se debe multiplicar es diferente a la almacenada en memoria. La multiplicación se realiza columna a columna, siguiendo el orden natural de la estructura elegida para la matriz. Se utiliza un vector acumulador para calcular el resultado, que se inicializa con todas sus posiciones en 0. Luego, para $i = 1, \cdots, n$, se utiliza la matriz A para calcular el producto entre la columna i de la matriz P_2 por el valor v_i del vector de entrada, y el resultado se suma al vector acumulador. Para cada columna pueden presentarse dos casos, que deben tratarse de forma distinta:

1. Si la columna i de A solo contiene valores nulos (es decir, `ptr_cols[i-1] = -1`), tendremos que, para todo $j = 1, \cdots, n$, $p_{2ij} = c(\frac{1}{n}) + (1-c)(\frac{1}{n}) = \frac{1}{n}$. Luego, el producto de esta columna por v_i puede calcularse directamente, y será un vector con todas sus componentes iguales a $\frac{v_i}{n}$.
2. Si, por el contrario, en la columna i de A hay valores no nulos, para $j = 1, \cdots, n$, tendremos que $p_{2ij} = c(a_{ij}) + (1-c)(\frac{1}{n})$. Es decir, el producto de esta columna por v_1 puede calcularse en dos pasos: primero, consideramos un vector con todas sus componentes iguales a $(1-c)(\frac{1}{n})$, y luego, para los índices j tales que $a_{ij} \neq 0$, le sumamos $c(a_{ij})$ a la componente correspondiente de dicho vector.

2.2.2. Implementación de *GeM*

La implementación del método *GeM* fue considerablemente más simple, debido a que, si bien ya no es válida la hipótesis de que la matriz obtenida será esparsa, sí puede asumirse que la instancias a tratar serán considerablemente menores. Dado que el algoritmo es básicamente el mismo, gran parte del código de *PageRank* pudo reutilizarse; las principiales modificaciones estuvieron en la forma de representación de la matriz y, por consiguiente, en la implementación del producto de esta con un vector.

Para el almacenamiento de la matriz, se decidió utilizar un vector de vectores (cada uno de estos representando una fila de la matriz), por considerar que brindaba un equilibrio adecuado entre eficiencia y facilidad de implementación. Dado que ahora se guardan en memoria todos los coeficientes, se decidió almacenar P_2 en lugar de A. De esta forma, las operaciones que transforman la segunda en la primera solo se hacen una vez, durante la carga de los datos, y se incrementa la simplicidad y la eficiencia de la función encargada de calcular el producto entre la matriz y el vector. Esta última función opera ahora de la forma tradicional, multiplicando cada fila de P_2 por el vector proporcionado y almacenando el resultado en la componente correspondiente del resultado.

3. Experimentación

3.1. *PageRank* y páginas web

A continuación, se presentan los experimentos que se realizaron. Con el código del trabajo práctico se incluye una serie de scripts de *bash* que permiten recrear los experimentos realizados, como así también los gráficos que se incluyen en este informe; esto puede hacerse ingresando al directorio `exp` dentro de la raíz, y ejecutando el comando `./expi.sh`, siendo `i` el número de experimento.

3.1.1. Experimento 1

En el primer experimento se quiere observar como varía el tiempo de ejecución a medida que se modifica el valor del parámetro c, para lograr que la norma Manhattan entre dos iteraciones consecutivas sea menor que la tolerancia indicada. Para ello se toma una determinada cantidad de páginas web que no se modificarán a lo largo del experimento y se ejecutará el programa para diferentes valores de c.

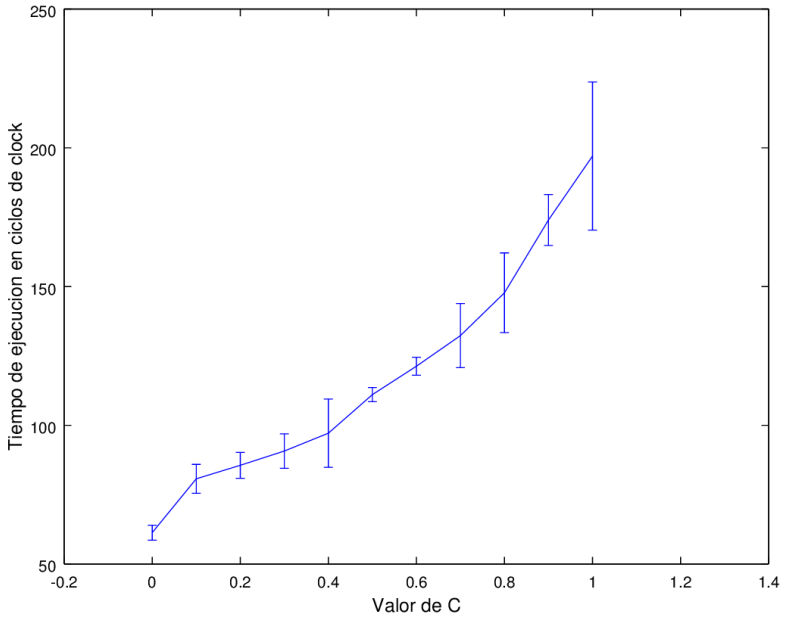
Hipótesis

Se conjetura que cuanta mayor sea la probabilidad de teletransportarse de una página a otra, menos tiempo va a tardar el algoritmo en llegar al estado en el que la tolerancia sea menor que la norma Manhattan. $(1 - c)$ indica cuál es la probabilidad de, estando en cualquier página, teletransportarse a otra. Por esto es que cuanto más chico sea el valor de c, menos tiempo de ejecución demorará el algoritmo.

Valores utilizados como parámetros

Se toman 13 páginas con 16 *links*. Además el parámetro c toma los valores `0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1`. La tolerancia es de `0.00001`.

Resultados



Discusión

Como se puede observar en el experimento, a medida que aumenta el valor de c, aumenta el tiempo de ejecución. Esto se debe a que la matriz inicial no es homogénea y la que formamos a partir del c si lo es. Así, cuando la segunda toma dicha importancia, el sistema tiende más rápido a ser homogéneo y requiere menos iteraciones del ciclo para lograr una norma menor a la tolerancia deseada.

3.1.2. Experimento 2

Con este experimento se pretende observar la diferencia en el tiempo de ejecución cuando se varía la cantidad de *links* en una determinada cantidad de páginas, manteniendo el valor de la tolerancia constante.

Para ello se utilizan listas de páginas diferentes como parámetro de entrada en cada una de las ejecuciones a comparar, pero manteniendo la cantidad de las mismas. Se toma el tiempo que se demora en ejecutar el algoritmo y se lo divide por la cantidad de iteraciones realizadas. Esto permite obtener un promedio del tiempo que demora por cada una de las iteraciones del ciclo.

Hipótesis

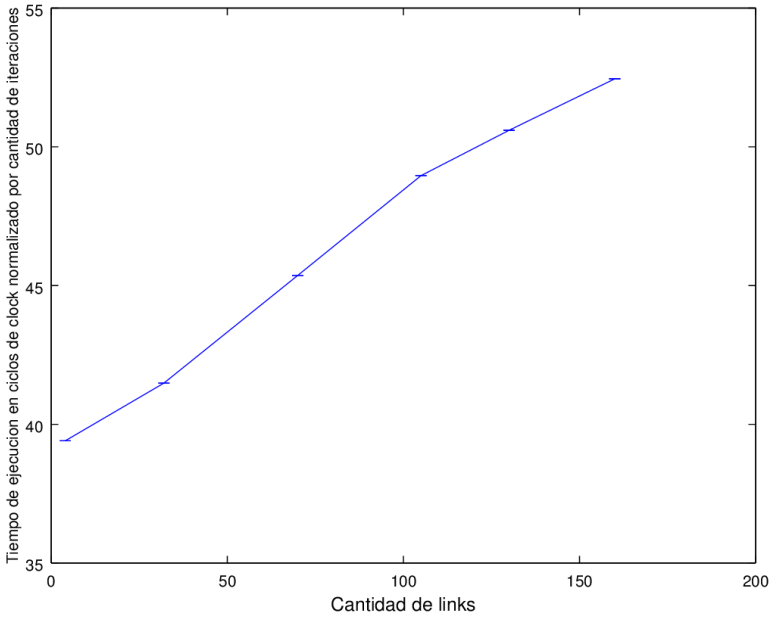
Suponemos que variar la cantidad de *links* altera más el tiempo de ejecución que variar la cantidad de páginas sin que estén relacionadas entre ellas. Esto se debe a que, dada la forma en la que está implementado el algoritmo, sacando provecho del hecho de que la matriz es esparsa, agregar entradas a la matriz de adyacencia tendrá un mayor impacto sobre la cantidad

de operaciones a realizar. Por lo tanto, se espera observar que el tiempo de ejecución aumente a medida que la cantidad de relaciones entre páginas sea mayor.

Valores utilizados como parámetros

Para este experimento se toman 50 páginas. La cantidad de *links* entre ellas varía, tomando los valores `4, 32, 70, 105, 130, 160`. Además el valor de c es `0.85` y el valor de la tolerancia es `0.00001`.

Resultados



Discusión

Como se puede observar en el gráfico, a medida que aumentan las relaciones entre las páginas el tiempo de ejecución es mayor.

3.1.3. Experimento 3

En este experimento también se observa la diferencia en el tiempo de ejecución, pero comparando igual cantidad de páginas y relaciones entre las mismas y variando el valor de la tolerancia.

Para ello se toma como parámetro de entrada en cada una de las ejecuciones una misma lista de páginas y se incrementa el valor de la tolerancia.

Hipótesis

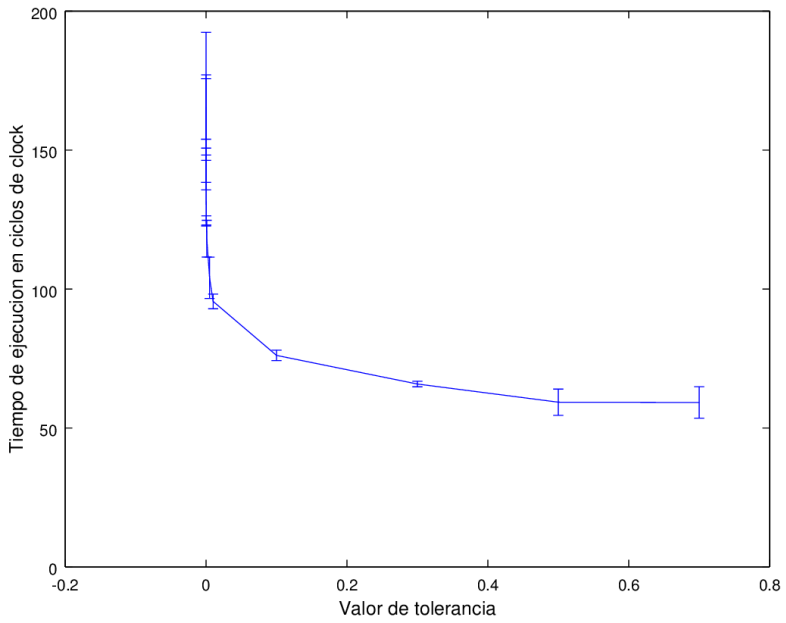
Creemos que cuanto mayor sea la tolerancia, menor será el tiempo de ejecución del algoritmo. Esto se debe a que el algoritmo termina cuando la diferencia Manhattan es menor que la

tolerancia. Entonces cuanto más grande sea el valor de la tolerancia, más rápido se cumplirá la condición para salir del ciclo y menos tardará en ejecutarse el algoritmo.

Valores utilizados como parámetros

Se toman 13 páginas con 16 *links*. Además la tolerancia toma los siguientes valores: `0.000001`, `0.000005`, `0.00001`, `0.00005`, `0.0001`, `0.0005`, `0.001`, `0.005`, `0.01`, `0.1`, `0.3`, `0.5`, `0.7`. El parámetro c es igual a `0.85`.

Resultados



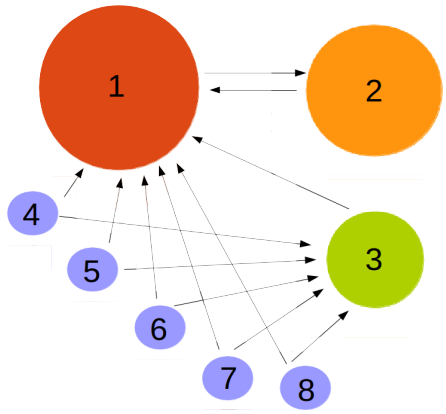
Conclusiones y observaciones

Como se puede ver en el gráfico, al aumentar la tolerancia disminuye el tiempo de ejecución. En este sentido, se pudo confirmar la hipótesis.

3.1.4. Experimento 4

Otra de las pruebas consiste en comparar los *rankings* formados al ejecutar los algoritmos de *PageRank* y el de In-deg. Primero se analiza una lista de páginas de entrada en la que una de ellas sea apuntada por el resto (Web 1).

Luego se realiza la comparación con una lista en la que hayan dos páginas (a las que llamaremos página 1 y página 3) que sean apuntadas por varias paginas. Además, la página 3 tendrá un link a la 1. También se cuenta con otra (a la que llamaremos página 2) que apunta y es apuntada por la página 1, tal como se muestra en el siguiente esquema (Web 2):



Hipótesis

En el primero, suponemos que los *rankings* obtenidos en ambos casos serán iguales ya que existe una sola página principal y el resto tiene igual cantidad de relaciones. Por otro lado, en el segundo notaremos diferencias. Llamaremos enlace fuerte a aquel que va de una página que es apuntada por muchas otras hacia otra página. El método *PageRank* tiene en cuenta cuál es el peso de los *links* que apuntan a las distintas páginas. En el caso de IN-DEG, solo se utiliza como información la cantidad de enlaces que llegan a cada una de las distintas páginas.

En el experimento, si consideramos el método *PageRank*, la página 1 quedará en primer lugar, ya que es la que recibe más *links* a la misma, pero en segundo lugar quedará la página 2. Esto se debe a que el enlace de la 1 a la 2 es un enlace muy fuerte. En el caso de IN-DEG, esto no sucede ya que este método solo toma en cuenta que existe un único link hacia la página 2. Por lo tanto en *PageRank* las posiciones serán 1-2-3 y luego el resto. En IN-DEG tendremos primero a la página 1 seguida de la 3, luego la 2 y finalmente el resto.

Valores utilizados como parámetros

Para la Web 1, se tomaron 9 páginas con 10 *links*. Para la Web 2, se tomaron 8 páginas con 13 *links*.

El valor de c es `0.85` y el valor de tolerancia es `0.00001`.

Resultados

Ranking Web 1 - *PageRank*		
Posición	Nodo	Puntaje
1	1	0.473848
2	8	0.078821
3	9	0.078821
4	2	0.061418
5	3	0.061418
6	4	0.061418
7	5	0.061418
8	6	0.061418
9	7	0.061418

Ranking Web 1 - In-deg		
Posición	Nodo	Puntaje
1	1	0.8
2	8	0.1
3	9	0.1
4	2	0
5	3	0
6	4	0
7	5	0
8	6	0
9	7	0

Ranking Web 2 - *PageRank*		
Posición	Nodo	Puntaje
1	1	0.448059
2	2	0.448059
3	3	0.058594
4	4	0.058594
5	5	0.018750
6	6	0.018750
7	7	0.018750
8	8	0.018750

Ranking Web 2 - In-deg		
Posición	Nodo	Puntaje
1	1	0.538462
2	3	0.384615
3	2	0.076923
4	4	0
5	5	0
6	6	0
7	7	0
8	8	0

Conclusiones y observaciones

Como podemos ver en los *rankings* del primer análisis en ambos casos quedan iguales. En cambio, en el segundo observamos variaciones, ya que en la lista de entrada el peso de todos los enlaces no es el mismo, es decir, hay enlaces más fuertes que otros. A diferencia de *PageRank*, el método IN-DEG no tiene en cuenta esta propiedad. Por lo tanto, si en las páginas de entrada todos los *links* tienen igual fuerza, el resultado obtenido por ambos métodos no difiere.

3.2. *GeM* y ligas deportivas

3.2.1. Experimento 1

En este experimento se verá la diferencia entre los métodos de *GeM* y el de la AFA para determinar el *ranking* de una liga deportiva. Para ello se tomará como parámetro de entrada una tabla con los partidos, sus respectivas fechas, y sus resultados. En la misma, deberá existir un equipo que ocupe una de las primaras posiciones en la tabla de *rankings* que haya perdido contra otro que ocupa una de las últimas. Observaremos la diferencia entre el *ranking* generado por cada uno de los métodos.

Hipótesis

Cuando se juega un partido entre dos equipos, en el método de *GeM* se tendrá en cuanta que tan fuerte son los equipos involucrados y cual fue la diferencia de goles en el partido. En cambio, en el método de la AFA el ganador recibirá 3 puntos sin importar los resultados ni las posiciones que los mismos tenían hasta el momento. Esto puede generar diferencias en el *ranking* ya que cada método considera distinta información para determinar las nuevas posiciones.

Por otro lado, si un equipo que se encuentra en las últimas posiciones de la tabla juega contra uno que esta en las primeras y gana el peor, para el método de la AFA es exactamente igual que haya jugado con cualquier otro. Para *GeM*, al tener en cuenta que tan fuertes son los equipos, esto generará que el nuevo vencedor quede en una mejor posición en la tabla. Así en este método entre dos fechas se pueden generar saltos, es decir, un equipo puede pasar de tener una muy mala posición en la tabla a estar dentro de los mejores solo por haber ganado un partido contra un equipo fuerte. En el método de la AFA no pasa ya que sólo se le sumarán 3 puntos al equipo triunfador. Si éste se encontraba en las ultimas posiciones de la tabla no podrá ascender más de 3 posiciones.

Valores utilizados como parámetros

Resultados del Torneo de Primera División del Fútbol Argentino hasta la Fecha 23. El valor de c es `0.85` y el valor de tolerancia es `0.00001`.

Resultados

Ranking Liga Deportiva - *GeM*			*Ranking* Liga Deportiva - AFA		
Posición	Nodo	Puntaje	Posición	Nodo	Puntaje
1	7	0.080913	1	24	0.054289
2	1	0.069879	2	7	0.053203
3	22	0.069065	3	21	0.049946
4	24	0.058309	4	23	0.048860
5	25	0.051788	5	22	0.047774
6	21	0.051674	6	15	0.041260
7	23	0.046326	7	6	0.041260
8	20	0.045206	8	11	0.041260
9	17	0.043581	9	28	0.040174
10	30	0.039828	10	5	0.040174
11	12	0.035738	11	16	0.039088
12	11	0.034629	12	12	0.038002
13	6	0.033908	13	20	0.034745
14	5	0.032996	14	25	0.034745
15	29	0.028015	15	29	0.033659
16	10	0.027041	16	27	0.030402
17	16	0.026036	17	2	0.028230
18	15	0.023666	18	17	0.028230
19	28	0.023078	19	1	0.028230
20	26	0.022266	20	30	0.027144
21	19	0.021956	21	10	0.026059
22	9	0.021713	22	26	0.026059
23	3	0.018678	23	19	0.024973
24	2	0.017951	24	8	0.024973
25	27	0.017670	25	13	0.023887
26	14	0.013740	26	14	0.022801
27	13	0.013541	27	4	0.021716
28	4	0.011738	28	3	0.018458
29	8	0.011141	29	18	0.015201
30	18	0.007928	30	9	0.015201

Conclusiones y observaciones

Como podemos ver en las tablas de *rankings*, los ordenes no son exactamente iguales. Por ejemplo, mirando el equipo 1 (Aldosivi), en el *ranking* de la AFA se encuentra en la posición 20 y cuando usamos el método de *GeM* éste esta en la posición 2. Esto se debe a que Aldosivi le ganó tanto al equipo 24 (San Lorenzo) como al 7 (Boca Juniors), que se encontraban primero y segundo en la tabla de posiciones de la AFA, respectivamente. Por esto, en *GeM* sube muchas posiciones mientras que con la AFA solo se le otorgan 3 puntos por cada uno de los dos partidos.

4. Conclusiones

Durante la realización de este trabajo, se pudo observar un ejemplo más de cómo las herramientas teóricas que brinda el álgebra lineal pueden tener aplicaciones sumamente prácticas. El éxito del método de *PageRank* es indiscutible, ya que su gran precisión a la hora de seleccionar resultados de relevancia fue crucial para posicionar al buscador Google por sobre sus competidores y convertirlo en el más utilizado a nivel mundial.

En cuanto a su adaptación para la clasificación en ligas deportivas, *GeM*, se encontraron múltiples razones por las que puede resultar muy efectivo. A diferencia del método utilizado por la AFA, toma en cuenta las posiciones en las que se encuentran los equipos a enfrentarse. Esto genera que si un equipo que se encuentra en una posición del *ranking* muy baja gana un partido con uno que esta entre los primeros, el vencedor subirá más en la tabla que lo que podría subir si le dan solamente una cantidad fija de puntos. Por otro lado en *GeM*, importa cual es la diferencia de goles, en cambio, con el método de la AFA es indistinta.

Una de las características que se observa al utilizar el método *GeM* es que, comparando las posiciones entre dos fechas consecutivas, puede pasar que se genere un salto con algún equipo. Es decir, un equipo puede pasar de estar en las últimas posiciones de la tabla a una de las primeras, lo cual podría generar sorpresas, en ocasiones desagradables.

El método de *GeM* tiene la fortaleza de ser robusto ante intentos de especular con los resultados, ya que toma en cuenta no solo si ganan o pierden los equipos si no que se basa en más información para determinar el *ranking*. No obstante, su complejidad también es una de las desventajas que puede atribuirsele: claramente, el método utilizado por la AFA es mas sencillo, y, como las ligas deportivas son vistas por muchas personas, puede generar problemas que el método utilizado para calcular posiciones en la tabla sea extremadamente difícil de comprender.

Apéndices

A. Enunciado del trabajo práctico

Contexto y motivación

A partir de la evolución de Internet durante la década de 1990, el desarrollo de motores de búsqueda se ha convertido en uno de los aspectos centrales para su efectiva utilización. Hoy en día, sitios como Yahoo, Google y Bing ofrecen distintas alternativas para realizar búsquedas complejas dentro de un red que contiene miles de millones de páginas web.

En sus comienzos, una de las características que distinguió a Google respecto de los motores de búsqueda de la época fue la calidad de los resultados obtenidos, mostrando al usuario páginas relevantes a la búsqueda realizada. El esquema general de los orígenes de este motor de búsqueda es brevemente explicado en Brin y Page [1], donde se mencionan aspectos técnicos que van desde la etapa de obtención deinformación de las páginas disponibles en la red, su almacenamiento e indexado y su posterior procesamiento, buscando ordenar cada página de acuerdo a su importancia relativa dentro de la red. El algoritmo utilizado para esta última etapa es denominado PageRank y es uno (no el único) de los criterios utilizados para ponderar la importancia de los resultados de una búsqueda. En este trabajo nos concentraremos en el estudio y desarrollo del algoritmo PageRank.

Por otro lado, las competencias deportivas, en todas sus variantes y disciplinas, requieren casi inevitablemente la comparación entre competidores mediante la confección de *Tablas de Posiciones* y *Rankings* en base a resultados obtenidos en un período de tiempo determinado. Estos ordenamientos de equipos están generalmente (aunque no siempre) basados en reglas relativamente claras y simples, como proporción de victorias sobre partidos jugados o el clásico sistema de puntajes por partidos ganados, empatados y perdidos. Sin embargo, estos métodossimples y conocidos por todos muchas veces no logran capturar la complejidad de la competencia y la comparación. Esto es particularmenteevidente en ligas donde, por ejemplo, todos los equipos no juegan la misma cantidad de veces entre sí.

A modo de ejemplo, la NBA y NFL representan dos ligas con fixtures de temporadas regulares con estas características. Recientemente, el Torneo de Primera División de AFA se suma a este tipo de competencias, ya que la incorporación de la *Fecha de Clásicos* parece ser una interesante idea comercial, pero no tanto desde el punto de vista deportivo ya que cada equipo juega contra su *clásico* más veces que el resto. Como contraparte, éstos rankings son utilizados muchas veces como criterio de decisión, como por ejemplo para determinar la participación en alguna competencia de nivel internacional, con lo cual la confección de los mismos constituye un elemento sensible, afectando intereses deportivos y económicos de gran relevancia.

El problema, Parte I: PageRank y páginas web

El algoritmo PageRank se basa en la construcción del siguiente modelo. Supongamos que tenemos una red con n páginas web $Web = \{1, \ldots, n\}$ donde el objetivo es asignar a cada una de ellas un puntaje que determine la importancia relativa de la misma respecto de las demás. Para modelar las relaciones entre ellas, definimos la *matriz de conectividad* $W \in \{0,1\}^{n \times n}$ de forma tal que $w_{ij} = 1$ si la página j tiene un link a la página i, y $w_{ij} = 0$ en caso contrario. Además, ignoramos los *autolinks*, es decir, links de una página a sí misma, definiendo $w_{ii} = 0$. Tomando

esta matriz, definimos el grado de la página j, n_j, como la cantidad de links salientes hacia otras páginas de la red, donde $n_j = \sum_{i=1}^{n} w_{ij}$. Además, notamos con x_j al puntaje asignado a la página $j \in Web$, que es lo que buscamos calcular.

La importancia de una página puede ser modelada de diferentes formas. Un link de la página $u \in Web$ a la página $v \in Web$ puede ser visto como que v es una página importante. Sin embargo, no queremos que una página obtenga mayor importancia simplemente porque es apuntada desde muchas páginas. Una forma de limitar esto es ponderar los links utilizando la importancia de la página de origen. En otras palabras, pocos links de páginas importantes pueden valer más que muchos links de páginas poco importantes. En particular, consideramos que la importancia de la página v obtenida mediante el link de la página u es proporcional a la importancia de la página u e inversamente proporcional al grado de u. Si la página u contiene n_u links, uno de los cuales apunta a la página v, entonces el aporte de ese link a la página v será x_u/n_u. Luego, sea $L_k \subseteq Web$ el conjunto de páginas que tienen un link a la página k. Para cada página pedimos que

$$x_k = \sum_{j \in L_k} \frac{x_j}{n_j}, \qquad k = 1, \ldots, n. \tag{1}$$

Definimos $P \in \mathbb{R}^{n \times n}$ tal que $p_{ij} = 1/n_j$ si $w_{ij} = 1$, y $p_{ij} = 0$ en caso contrario. Luego, el modelo planteado en (1) es equivalente a encontrar un $x \in \mathbb{R}^n$ tal que $Px = x$, es decir, encontrar (suponiendo que existe) un autovector asociado al autovalor 1 de una matriz cuadrada, tal que $x_i \geq 0$ y $\sum_{i=1}^{n} x_i = 1$. En Bryan y Leise [2] y Kamvar et al. [5, Sección 1] se analizan ciertas condiciones que debe cumplir la red de páginas para garantizar la existencia de este autovector.

Una interpretación equivalente para el problema es considerar al *navegante aleatorio*. Éste empieza en una página cualquiera del conjunto, y luego en cada página j que visita sigue navegando a través de sus links, eligiendo el mismo con probabilidad $1/n_j$. Una situación particular se da cuando la página no tiene links salientes. En ese caso, consideramos que el navegante aleatorio pasa a cualquiera de las página de la red con probabilidad $1/n$. Para representar esta situación, definimos $v \in \mathbb{R}^{n \times n}$, con $v_i = 1/n$ y $d \in \{0, 1\}^n$ donde $d_i = 1$ si $n_i = 0$, y $d_i = 0$ en caso contrario. La nueva matriz de transición es

$$\begin{aligned} D &= v d^t \\ P_1 &= P + D. \end{aligned}$$

Además, consideraremos el caso de que el navegante aleatorio, dado que se encuentra en la página j, decida visitar una página cualquiera del conjunto, independientemente de si esta se encuentra o no referenciada por j (fenómeno conocido como *teletransportación*). Para ello, consideramos que esta decisión se toma con una probabilidad $c \geq 0$, y podemos incluirlo al modelo de la siguiente forma:

$$\begin{aligned} E &= v \bar{1}^t \\ P_2 &= c P_1 + (1 - c) E, \end{aligned}$$

donde $\bar{1} \in \mathbb{R}^n$ es un vector tal que todas sus componentes valen 1. La matriz resultante P_2 corresponde a un enriquecimiento del modelo formulado en (1). Probabilísticamente, la componente

x_j del vector solución (normalizado) del sistema $P_2x = x$ representa la proporción del tiempo que, en el largo plazo, el navegante aleatorio pasa en la página $j \in Web$. Denotaremos con π al vector solución de la ecuación $P_2x = x$, que es comúnmente denominado *estado estacionario*.

En particular, P_2 corresponde a una matriz *estocástica por columnas* que cumple las hipótesis planteadas en Bryan y Leise [2] y Kamvar et al. [5], tal que P_2 tiene un autovector asociado al autovalor 1, los demás autovalores de la matriz cumplen $1 = \lambda_1 > |\lambda_2| \geq \cdots \geq |\lambda_n|$ y, además, la dimensión del autoespacio asociado al autovalor λ_1 es 1. Luego, π puede ser calculada de forma estándar utilizando el método de la potencia.

Una vez calculado el ranking, se retorna al usuario las t páginas con mayor puntaje.

El problema, Parte II: PageRank y ligas deportivas

Existen en la literatura distintos enfoques para abordar el problema de determinar el *ranking* de equipos de una competencia en base a los resultados de un conjunto de partidos. En Govan et al. [4] se hace una breve reseña de dos ellos, y los autores proponen un nuevo método basado en el algoritmo PageRank que denominan GeM[2]. Conceptualmente, el método GeM representa la temporada como un red (grafo) donde las páginas web representan a los equipos, y existe un link (que tiene un valor, llamado peso, asociado) entre dos equipos que los relaciona modelando los resultados de los posibles enfrentamientos entre ellos. En base a este modelo, Govan et al. [4] proponen calcular el ranking de la misma forma que en el caso de las páginas web.

En su versión básica, que es la que consideraremos en el presente trabajo, el método GeM (ver, e.g., [4, Sección GeM Ranking Method]) es el siguiente[3]:

1. La temporada se representa mediante un grafo donde cada equipo representa un nodo y existe un link de i a j si el equipo i perdió al menos una vez con el equipo j.

2. Se define la matriz $A^t \in \mathbb{R}^{n\times n}$

$$A^t_{ji} = \begin{cases} w_{ji} & \text{si el equipo } i \text{ perdió con el equipo } j, \\ 0 & \text{en caso contrario,} \end{cases}$$

   donde w_{ji} es la diferencia absoluta en el marcador. En caso de que i pierda más de una vez con j, w_{ji} representa la suma acumulada de diferencias. Notar que A^t es una generalización de la matriz de conectividad W definida en la sección anterior.

3. Definir la matriz $H^t_{ji} \in \mathbb{R}^{n\times n}$ como

$$H^t_{ji} = \begin{cases} A^t_{ji} / \sum_{k=1}^{n} A^t_{ki} & \text{si hay un link } i \text{ a } j, \\ 0 & \text{en caso contrario.} \end{cases}$$

4. Tomar $P = H^t$, y aplicar el método PageRank como fue definido previamente, siendo π la solución a la ecuación $P_2x = x$. Notar que los páginas sin links salientes, en este contexto se corresponden con aquellos equipos que se encuentran invictos.

5. Utilizar los puntajes obtenidos en π para ordenar los equipos.

[2] Aunque no se especifica, asumimos que el nombre se debe a las iniciales de los autores.

[3] Notar que en artículo, Govan et al. [4] lo definen sobre la traspuesta. La definición y las cuentas son equivalentes, simplemente se modifica para mantener la consistencia a lo largo del enunciado.

En función del contexto planteado previamente, el método GeM define una estructura que relaciona equipos dependiendo de los resultados parciales y obtener un ranking utilizando solamente esta información.

Enunciado

El objetivo del trabajo es experimentar en el contexto planteado utilizando el algoritmo PageRank con las variantes propuestas. A su vez, se busca comparar los resultados obtenidos cualitativa y cuantitativamente con los algoritmos tradicionales utilizados en cada uno de los contextos planteados. Los métodos a implementar (como mínimo) en ambos contexto planteados por el trabajo son los siguientes:

1. *Búsqueda de páginas web:* PageRank e In-deg, éste último consiste en definir el ranking de las páginas utilizando solamente la cantidad de ejes entrantes a cada una de ellas, ordenándolos en forma decreciente.

2. *Rankings en competencias deportivas:* GeM y al menos un método estándar propuesto por el grupo (ordenar por victorias/derrotas, puntaje por ganado/empatado/perdido, etc.) en función del deporte(s) considerado(s).

El contexto considerado en 1., en la búsqueda de páginas web, representa un desafío no sólo desde el modelado, si no también desde el punto de vista computacional considerando la dimensión de la información y los datos a procesar. Luego, dentro de nuestras posibilidades, consideramos un entorno que simule el contexto real de aplicación donde se abordan instancias de gran escala (es decir, n, el número total de páginas, es grande). Para el desarrollo de PageRank, se pide entonces considerar el trabajo de Bryan y Leise [2] donde se explica la intución y algunos detalles técnicos respecto a PageRank. Además, en Kamvar et al. [5] se propone una mejora del mismo. Si bien esta mejora queda fuera de los alcances del trabajo, en la Sección 1 se presenta una buena formulación del algoritmo. En base a su definición, P_2 no es una matriz esparsa. Sin embargo, en Kamvar et al. [5, Algoritmo 1] se propone una forma alternativa para computar $x^{(k+1)} = P_2 x^{(k)}$. Este resultado debe ser utilizado para mejorar el almacenamiento de los datos.

En la práctica, el grafo que representa la red de páginas suele ser esparso, es decir, una página posee relativamente pocos links de salida comparada con el número total de páginas. A su vez, dado que n tiende a ser un número muy grande, es importante tener en cuenta este hecho a la hora de definir las estructuras de datos a utilizar. Luego, desde el punto de vista de implementación se pide utilizar alguna de las siguientes estructuras de datos para la representación de las matrices esparsas: *Dictionary of Keys* (dok), *Compressed Sparse Row* (CSR) o *Compressed Sparse Column* (CSC). Se deberá incluir una justificación respecto a la elección que consdiere el contexto de aplicación. Además, para PageRank se debe implementar el método de la potencia para calcular el autovector principal. Esta implementación debe ser realizada íntegramente en C++.

En función de la experimentación, se deberá realizar un estudio particular para cada algoritmo (tanto en términos de comportamiento del mismo, como una evaluación de los resultados obtenidos) y luego se procederá a comparar cualitativamente los rankings generados. La experimentación deberá incluir como mínimo los siguientes experimentos:

1. Estudiar la convergencia de PageRank, analizando la evolución de la norma Manhattan

(norma L_1) entre dos iteraciones sucesivas. Comparar los resultados obtenidos para al menos dos instancias de tamaño mediano-grande, variando el valor de c.

2. Estudiar el tiempo de cómputo requerido por PageRank.

3. Para cada algoritmo, proponer ejemplos de tamaño pequeño que ilustren el comportamiento esperado (puede ser utilizando las herramientas provistas por la cátedra o bien generadas por el grupo).

Puntos opcionales:

1. Demostrar que los pasos del Algoritmo 1 propuesto en Kamvar et al. [5] son correctos y computan P_2x.

2. Establecer una relación con la proporción entre $\lambda_1 = 1$ y $|\lambda_2|$ para la convergencia de PageRank.

El segundo contexto de aplicación no presenta mayores desafíos desde la perspectiva computacional, ya que en el peor de los casos una liga no suele tener mas que unas pocas decenas de equipos. Más aún, es de esperar que en general la matriz que se obtiene no sea esparsa, ya que probablemente un equipo juegue contra un número significativo de contrincantes. Sin embargo, la popularidad y sensibilidad del problema planteado requieren de un estudio detallado y pormenorizado de la calidad de los resultados obtenidos. El objetivo en este segundo caso de estudio es puramente experimental.

En función de la implementación, aún cuando no represente la mejor opción, es posible reutilizar y adaptar el desarrollo realizado para páginas web. También es posible realizar una nueva implementación desde cero, simplificando la operatoria y las estructuras, en C++, MATLAB o PYTHON.

La experimentación debe ser realizada con cuidado, analizando (y, eventualmente, modificando) el modelo de GeM:

1. Considerar al menos un conjunto de datos reales, con los resultados de cada fecha para alguna liga de alguún deporte.

2. Notar que el método GeM asume que no se producen empates entre los equipos (o que si se producen, son poco frecuentes). En caso de considerar un deporte donde el empate se da con cierta frecuencia no despreciable (por ejemplo, fútbol), es fundamental aclarar como se refleja esto en el modelo y analizar su eventual impacto.

3. Realizar experimentos variando el parámetro c, indicando como impacta en los resultados. Analizar la evolución del ranking de los equipos a través del tiempo, evaluando también la evolución de los rankings e identificar características/hechos particulares que puedan ser determinantes para el modelo, si es que existe alguno.

4. Comparar los resultados obtenidos con los reales de la liga utilizando el sistema estándar para la misma.

Puntos opcionales:

1. Proponer (al menos) dos formas alternativas de modelar el empate entre equipos en GeM.

Parámetros y formato de archivos

El programa deberá tomar por línea de comandos dos parámetros. El primero de ellos contendrá la información del experimento, incluyendo el método a ejecutar (`alg`, 0 para PageRank, 1 para el método alternativo), la probabilidad de teletransportación c, el tipo de instancia (0 páginas web, 1 deportes), el *path* al archivo/directorio conteniendo la definición de la red (que debe ser relativa al ejecutable, o el path absoluto al archivo) y el valor de tolerancia utilizado en el criterio de parada del método de la potencia.

El siguiente ejemplo muestra un caso donde se pide ejecutar PageRank, con una probabilidad de teletransportación de 0.85, sobre la red descripta en `test1.txt` (que se encuentra en el directorio `tests/`), correspondiente a una instancia de ranking aplicado a deportes y con una tolerancia de corte de 0,0001.

```
0 0.85 1 tests/red-1.txt 0.0001
```

Para la definición del grafo que representa la red, se consideran dos bases de datos de instancias con sus correspondientes formatos. La primera de ellas es el conjunto provisto en SNAP [6] (el tipo de instancia es 0), con redes de tamaño grande obtenidos a partir de datos reales. Además, se consideran las instancias que se forman a partir de resultados de partidos entre equipos, para algún deporte elegido por el grupo.

En el caso de la base de SNAP, los archivos contiene primero cuatro líneas con información sobre la instancia (entre ellas, n y la cantidad total de links, m) y luego m líneas con los pares i, j indicando que i apunta a j. A modo de ejemplo, a continuación se muestra el archivo de entrada correspondiente a la red propuesta en Bryan y Leise [2, Figura 1]:

```
# Directed graph (each unordered pair of nodes is saved once):
# Example shown in Bryan and Leise.
# Nodes: 4 Edges: 8
# FromNodeId    ToNodeId
1   2
1   3
1   4
2   3
2   4
3   1
4   1
4   3
```

Para el caso de rankings en ligas deportivas, el archivo contiene primero una línea con información sobre la cantidad de equipos (n), y la cantidad de partidos totales a considerar (k). Luego, siguen k lńeas donde cada una de ellas representa un partido y contiene la siguiente información: número de fecha (es un dato opcional al problema, pero que puede ayudar a la hora de experimentar), equipo i, goles equipo i, equipo j, goles equipo j. A continuación se muestra el archivo de entrada con la información del ejemplo utilizado en Govan et al. [4]:

```
6 10
```

```
1 1 16 4 13
1 2 38 5 17
1 2 28 6 23
1 3 34 1 21
1 3 23 4 10
1 4 31 1 6
1 5 33 6 25
1 5 38 4 23
1 6 27 2 6
1 6 20 5 12
```

Es importante destacar que, en este último caso, los equipos son identificados mediante un número. Opcionalmente podrá considerarse un archivo que contenga, para cada equipo, cuál es el código con el que se lo identifica.

Una vez ejecutado el algoritmo, el programa deberá generar un archivo de salida que contenga una línea por cada página (n líneas en total), acompañada del puntaje obtenido por el algoritmo PageRank/In-deg/método alternativo.

Para generar instancias de páginas web, es posible utilizar el código Python provisto por la cátedra. La utilización del mismo se encuentra descripta en el archivo README. Es importante mencionar que, para que el mismo funcione, es necesario tener acceso a Internet. En caso de encontrar un bug en el mismo, por favor contactar a los docentes de la materia a través de la lista. Desde ya, el código puede ser modificado por los respectivos grupos agregando todas aquellas funcionalidades que consideren necesarias.

Para instancias correspondientes a resultados entre equipos, la cátedra provee un conjunto de archivos con los resultados del Torneo de Primera División del Fútbol Argentino hasta la Fecha 23. Es importante aclarar que los dos partidos suspendidos, River - Defensa y Justicia y Racing - Godoy Cruz han sido arbitrariamente completados con un resultado inventado, para simplificar la instancia. En función de datos reales, una alternativa es considerar el repositorio DataHub [3], que contiene información estadística y resultados para distintas ligas y deportes de todo el mundo.

Fechas de entrega

- *Formato Electrónico:* Martes 6 de Octubre de 2015, hasta las 23:59 hs, enviando el trabajo (informe + código) a la dirección `metnum.lab@gmail.com`. El subject del email debe comenzar con el texto `[TP2]` seguido de la lista de apellidos de los integrantes del grupo.
- *Formato físico:* Miércoles 7 de Octubre de 2015, a las 18 hs. en la clase práctica.

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

Referencias

[1] 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.

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

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

[4] 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.

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

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