DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico I

(No) Todo Pasa

Métodos Numéricos
Primer Cuatrimestre de 2016

Integrante	LU	Correo electrónico
Cortés, Lucas	302/13	lucascortes@me.com
Lamela, Emanuel	021/13	emanuel93_13@hotmail.com
Zimenspitz, Ezequiel	155/13	ezeqzim@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

El contexto de este trabajo es el de Sport Analytics, considerando deportes en los cuáles los *schedules* de los mismos son complejos y probablemente asimétricos para los participantes. El puntapié inicial del mismo es plantear técnicas de armados de *rankings* que determinen una valorización de cada participante. Nuestro objetivo será entonces elaborar métodos para confeccionar estos rankings y análisis comparativos que permitan determinar el que mejor se asimile a la realidad.

Palabras claves

◦ Sport Analytics
◦ Colley Matrix Method
◦ Eliminación Gaussiana
◦ Factorización de Cholesky

Índice

1. Introducción Teórica

"¿Qué equipo/quién crees que gana hoy?", una simple pregunta que es difícil de responder con certeza dada la cantidad de aspectos que ofrecen los deportes en general. Antes de analizar una manera de responder la misma con datos y fundamentos teóricos, abocamos en algunos casos en los cuáles esto sería útil:

- **Inversiones:** Convencer de que fortalecer financieramente a una entidad/un jugador es seguro.
- **Puntuación:** En determinados deportes o contextos, vencer a distintos oponentes no siempre genera la misma cantidad de puntos. Esto ayuda a determinar una valuación de los mismos.
- **Medición:** Podría aportar métricas internas para determinar cómo se encuentra la entidad/el jugador comparativamente con los demás.

Proveer metodologías que incorporen aspectos varios y relevantes de los encuentros es esencial para que incremente su precisión. Dicho esto, el método que analizaremos es el ***Colley Matrix Method*** **[1]**.

1.1. Colley Matrix Method

Este método busca categorizar en un *ranking* de *ratings* a los equipos participantes, teniendo en consideración el schedule que atravesó cada uno de ellos, sin importar la diferencia de la cantidad de partidos jugados por cada equipo y solo considerando si el equipo ganó o perdió y no la diferencia en puntajes obtenidos en los mismos. Es necesario notar que sólo aplica a modelos de competencias que no admiten empate como un resultado posible.

Definamos primero el modelo utilizado para el problema. Sea $\Gamma = \{1, 2, ..., T\}$ el conjunto de participantes de la competencia. Luego, para cada equipo $\boldsymbol{i \in \Gamma}$ denominamos $\boldsymbol{n_i}$ al número total de partidos jugados por el equipo i, $\boldsymbol{w_i}$ al número de partidos ganados por el equipo i y, análogamente, $\boldsymbol{l_i}$ a la cantidad de encuentros por el equipo i. Por último, dados $i, j \in \Gamma$, $i \neq j$, $\boldsymbol{n_{ij}}$ al número de enfrentamientos entre i y j. Una vez definido esto, a través de una serie de postulados y argumentos matemáticos, el paper **[1]** plantea que las probabilidades se obtienen como resultado de un sistema de ecuaciones lineales de la forma $\boldsymbol{Cr = b}$.

Donde:

- $C \in R^{T \times T}, C_{ij} = \begin{cases} -n_{ij} & si \quad i \neq j \\ 2 + n_i & si \quad i = j \end{cases}$
- $r \in R^T$, donde $r_i =$ probabilidad de que el equipo i gane su siguiente partido
- $b \in R^T$, donde $b_i = 1 + (w_i - l_i)/2$

Por lo tanto, lo que se busca despejar son los elementos del vector r.

C se denomina la **matriz de Colley** que particularmente, por lo demostrado en **[1]**, es **simétrica** ($A = A^t$) y **definida positiva** ($\forall x \neq 0, x^t Ax > 0$).

Para resolver este sistema, usaremos dos algoritmos distintos para obtener sistemas de fácil resolución: *Eliminación Gaussiana* y *Factorización de Cholesky*.

1.1.1. Eliminación Gaussiana

La eliminación gaussiana es un algoritmo que transforma un sistema de ecuaciones en un sistema equivalente, con la característica de que este nuevo sistema es triangular superior.

Esto se logra a través de operaciones que no alteran el conjunto solución de un sistema:

- Multiplicar una ecuación por un escalar

- Intercambiar ecuaciones
- Sumar a una ecuación con un múltiplo de otra

Se aplican estas operaciones de tal forma que uno obtenga "ceros" debajo de la diagonal de la matriz resultado. Profundización acerca de este método puede ser encontrada en **[2]**.

Luego se resuelve y obtenemos el resultado deseado. Sea $A \in R^{nxn}, n \in N$. El sistema $Ax = b$ se transforma en uno equivalente $Ux = b'$, con U una matriz triangular superior.

Los x_i se obtienen de la siguiente manera:

$x_i = (b'_i - \sum_{j=i+1}^{n} u_{ij}x_i)/u_{ii}$, que es la técnica conocida como *back-substitution*.

Se puede observar que si $\exists\, i \in \{1, ..., n\}/a_{ii} = 0$ entonces no se puede realizar este procedimiento. De todas formas, para las matrices *simétricas* y *definidas positivas* se puede probar que se puede realizar la eliminación sin pivoteos y obtener la U sin problemas.

La combinación de la Eliminación Gaussiana y el proceso de *back-substitution* nos deja una complejidad temporal de $O(n^3)$, ya que es $O(n^3)$ por parte de la Eliminación y $O(n^2)$ en el siguiente y último procedimiento. Hay que considerar también la cantidad de operaciones en punto flotante que se hacen debido a la aritmética que conlleva. Más sobre esto posteriormente.

1.1.2. Factorización de Cholesky

La factorización de Cholesky es un caso particular de una factorización LU, con L matriz triangular inferior y U matriz triangular superior. Siendo $C \in R^{n\times n}$, n natural, bajo la hipótesis de que A sea *simétrica* y *definida positiva*, podemos afirmar que existe una factorización de la forma LU de C, tal que $U = L^t$ y tal que vale que $\underset{1<i<n}{\forall} L_{ii} > 0$.

Donde, la matriz $A = LL^t$ se forma bajo las siguientes ecuaciones:

$$LL^t{}_{ij} = \begin{cases} \sqrt{C_{ii} - \sum_{k=1}^{i-1} {L_{ik}}^2} & si \quad i = j \\ \\ \frac{1}{L_{ii}}(C_{ij} - \sum_{k=1}^{i-1} L_{ik}L^t{}_{jk}) & si \quad i \neq j \end{cases}$$

Luego, el sistema equivalente será $LL^tx = b$, entonces podemos resolver $Ly = b$ por forward-substitution y luego $L^tx = y$ para obtener el resultado deseado por back-substitution.

Si bien la complejidad del mismo es $O(n^3)$, la cantidad de *flops* es mucho menor. Uno estaría tentado a argumentar que este algoritmo es pesado por el calculo utilizando raíces cuadradas. Vamos a comprobar más adelante que esto es falso y el algoritmo de Cholesky posee una constante *oculta* (oculta por la notación O) mucho menor que su contraparte Gaussiana.

Además de ello, se puede notar que la factorización *no* involucra al vector b, por ende uno podría reutilizar los resultados obtenidos, pagando una sola vez el costo cúbico, y luego realizando cálculos del orden $O(n^2)$, para resolver infinidades de sistemas en costo cuadrático.

2. Desarrollo

Dada la participación del método CMM en este trabajo, utilizamos los algoritmos y propiedades enunciadas en la sección anterior. Tener en cuenta que los deportes y *schedules* de los mismos analizados en este trabajo: no cuentan con empate ni obligan, necesariamente, a los equipos a enfrentarse una misma cantidad de veces entre sí. Ambos son requisitos fundamentales para este análisis.

2.1. Roles de algoritmos utilizados

Consideremos $\Gamma = \{1, 2, ..., T\}$ el conjunto de equipos/jugadores junto con su *schedule*, $C \in R^{T \times T}$ la matriz de Colley y b que se generan en consecuencia.

Como sabemos que C, por lo explicado en **[1]**, es *simétrica definida positiva*, podemos utilizar la Eliminación Gaussiana o la factorización de Cholesky para resolverlo sin problemas:

Ataquemos el primero. Notar que vamos a tener en cuenta la siguiente matriz $A \in R^{T \times T+1}$:

$$A = \Big(C \,|\, b \Big)$$

El procedimiento consiste en iterar por cada fila de la matriz buscando dejar en 0 todas las posiciones por debajo de la diagonal.

Sea A^k la matriz luego de realizados k pasos del procedimiento. En la primer iteración, realiza el siguiente cálculo:

$$Fila_i = Fila_i - \tfrac{a_{i1}}{a_{11}}.Fila_1$$

De esta manera logra poner en 0 a todas las posiciones A_{i1} con $i > 1$. Luego, en la iteración k:

$$Fila_i^k - \tfrac{a_{ik}^k}{a_{kk}^k}.Fila_k^k$$

Poniendo en 0 las posiciones a_{ik} con $i > k$. Luego, en la iteración $n - 1$ se obtiene una matriz U triangular superior.

Recordemos que este algoritmo tiene un caso borde que ocurre cuando algún elemento $a_{ii} = 0$, pero ese caso es salvado por el hecho de que la matriz es *simétrica definida positiva*. Aunque se menciono que sirve para casos *sin* pivoteo, se puede extender su complemento también: los elementos de la diagonal de A^1 son positivos, por ser *definida positiva*, y por ende a_{11} lo es, en consecuencia los valores que obtengo en la diagonal de A^2 son positivos puesto que es el resultado de división de positivos. Esto se extiende a todas las iteraciones de manera inductiva.

A causa de las divisiones, sumamos pivoteo parcial. En la iteración k, antes de realizar las operaciones anteriores, busca la fila que tenga el valor absoluto más grande a_{ik}, $\forall i = k, ..., T$, y la intercambia con la fila k. Esto se realiza con el fin de minimizar errores de redondeo que se puedan dar por trabajar con aritmética finita. Al dividir por un valor más alto, generamos un número más cercano al 0, dónde la densidad de valores, en punto flotante, es más elevada en la representación *IEEE*.

Vayamos ahora a la Factorización de Cholesky. Se puede utilizar en este contexto dada la naturaleza de la matriz de Colley de ser *simétrica definida positiva*, lo cual asegura la existencia de una factorización LU particular (explicado en la sección 1.1.2).

El cálculo de la factorización no es demasiado complejo y está explćito en la sección de introducción. Lo que vale destacar es que aprovechamos el hecho de que la infomación esencial de la descomposición se puede almacenar en la parte inferior de una matriz, en otras palabras la matriz triangular inferior L. Por ende, la parte superior puede rellenarse con los valores traspuestos de si misma y operar con los procedimientos de resolución de sistemas triangulares $in - place$. Esto implica una reducción importante almacenamiento que si bien sigue siendo $O(n^2)$, en la práctica implica un ahorro de una matriz entera en memoria.

Sea cual sea el método elegido, una vez realizado utilizamos *backward o forward substitution*, según correspondiese para obtener los r_i. *Forward-substitution* resuelve un sistema triangular inferior, mientras que *Backward-substitution*, uno triangular superior.

En el trabajo no se provee más que una explicación detallada de los algoritmos utilizados, pero pseudo-códigos y profundización acerca de ellos puede ser encontrada en **[2]**.

2.2. Detalles Implementativos

Toda la algoritmia y los experimentos fueron desarrollados en C++. La estructura que elegimos para modelar las matrices fue *std::vector<std::vector<double>>*. Los aspectos fundamentales que nos llevaron a tomar esta determinación fueron:

- Gran cantidad de los calculos involucran accesos no contiguos a elementos, y la estructura elegida cual permite accesos aleatorios en tiempo constante.
- Utilizar *precisión doble* asegura menos error de redondeo. La densidad de valores representables aumenta considerablemente en contraposición a su par de *precisión simple*.
- En el caso particular del método de Cholesky, permitió almacenar en un sólo contenedor toda la información relevante y poder solventar el costo en memoria.

En cuánto a la metodología llevada adelante en los experimentos: cada uno cuenta con su explicación escrita y, dependiendo del caso, diagramada.

3. Resultados y Discusión

3.1. Análisis Cuantitativo

En este apartado vamos a analizar los algoritmos desarrollados en términos cuantitativos. Es decir una comparación entre el rendimiento de los métodos implementados.

3.1.1. Eliminación Gaussiana vs Factorización de Cholesky (complejidad algorítimica)

Hipótesis: Tanto el algoritmo de Eliminación Gaussiana como el de Factorización de Cholesky tienen complejidad $O(n^3)$.

Para analizar la hipótesis generamos instancias de posibles partidos de distinta cantidad de jugadores (50, 100, 150... 500). Vamos a tomar números aleatorios para seleccionar los equipos y resultados de cada partido y mediremos los tiempos para el algoritmo de Cholesky y el de Eliminación Gaussiana para cada una de esas instancias de partidos. Para la medición de cada una de estas instancias ejecutamos 40 veces cada algoritmo y registramos el promedio.

El gráfico siguiente muestra la cantidad de ticks en promedio (siguiendo el procedimiento mencionado) mostrando una complejidad polinomial del algoritmo.

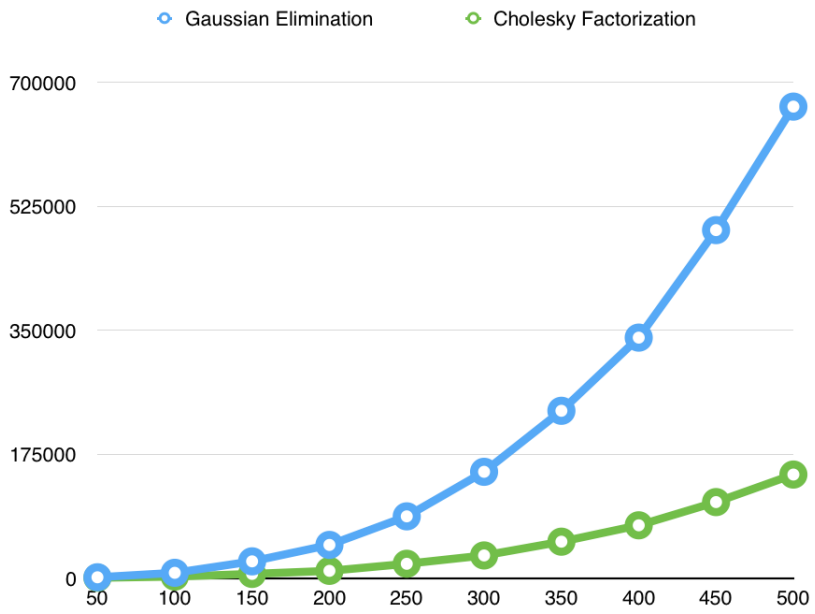

Figura 1: Eliminación gaussiana vs Factorización de Cholesky

Si a estas funciones las calculamos dividiéndolas por n^2 vemos que se obtienen funciones lineales como muestra el siguiente gráfico.

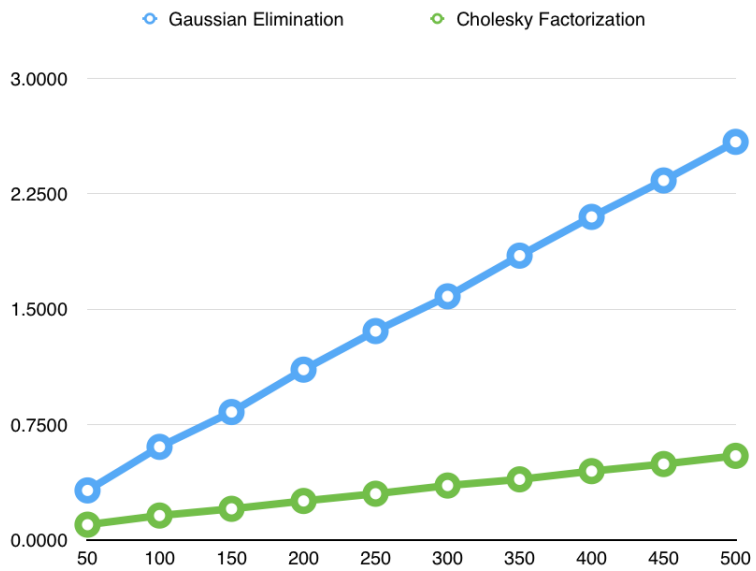

Figura 2: Eliminación gaussiana vs Factorización de Cholesky (base cúbica)

Para concluir que el orden es realmente cúbico, tomamos estos mismos datos pero los representamos dividiéndolos por n^3. Las funciones que obtenemos vemos que son casi constantes y se estabilizan a medida que n crece. Las instancias de n pequeños tienen una cantidad de ticks levemente mayor pero esto es atribuible a motivos de caché del procesador y procesos paralelos.

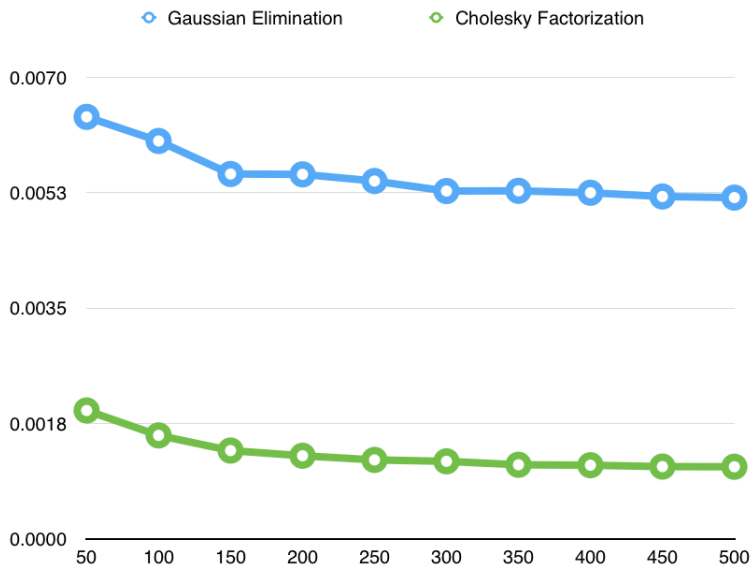

Figura 3: Eliminación gaussiana vs Factorización de Cholesky (base cúbica)

3.1.2. Eliminación Gaussiana vs Factorización de Cholesky (modificación del término independiente)

Hipótesis: El algoritmo de Eliminación Gaussiana tiene complejidad $O(n^3)$ mientras que el algoritmo de Factorización de Cholesky tiene complejidad $O(n^2)$ para recalcular las soluciones de los sistemas por modificaciones en el término independiente.

Dado que Gauss modifica el término independiente en cada iteración para alcanzar el sistema equivalente, modificarlo posteriormente implica volver a cargar los datos iniciales y a aplicar la Eliminación para resolverlo con el nuevo b.

En el caso de Cholesky, no se hace uso del término independiente, por lo tanto basta calcular una única vez la factorización y resolver el sistema para cada b que se quiera.

Para verificar esto, nuestra implementación verifica que las soluciones de cada algoritmo sean las mismas para cada b. Esta igualdad se realiza con un determinado grado de tolerancia debido a la aritmética de punto flotante.

Una vez que determinada la igualdad de las soluciones, procedemos a analizar el costo de cada método. Corrimos el algoritmo con instancias de 50 a 500 equipos con intervalos de 50 y trabajamos con 20 modificaciones de término independiente, sin modificar la matriz de Colley inicial.

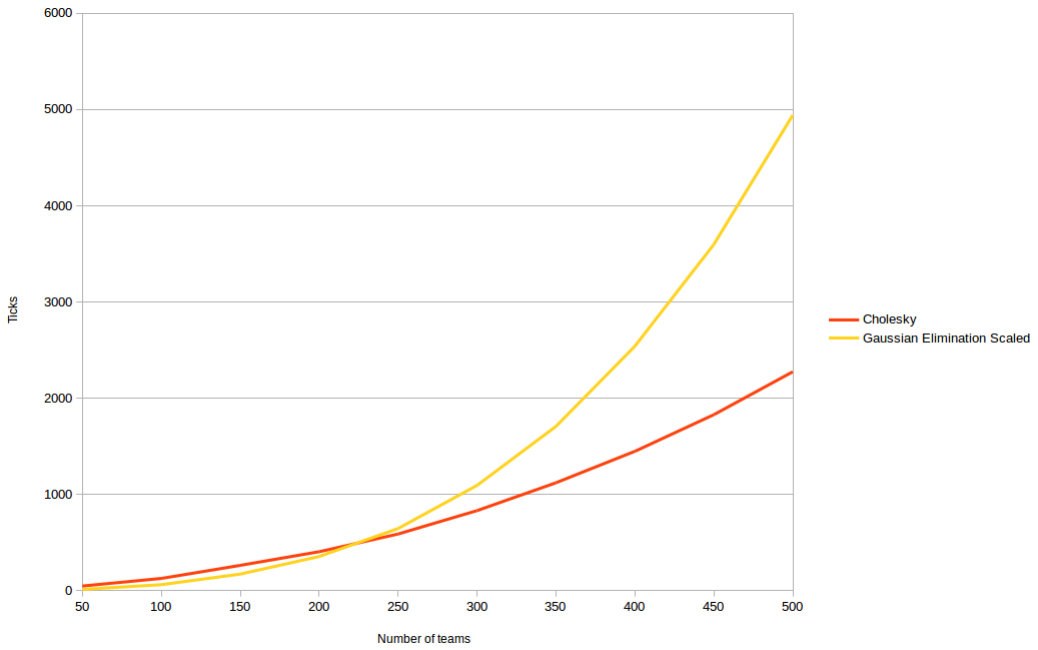

Figura 4: Comparación de complejidad temporal modificando el b en cada iteración

Este gráfico tiene la curva de la Eliminación Gaussiana escalada para poder visualizarlo en conjunto con la otra.

Puede apreciarse que el método de Gauss tiene una curva mucho más pronunciada que la factorización y se parecen a las de las complejidaded esperadas. No obstante no podemos discernir si son la misma pero multiplicadas por una constante, de modo que el próximo paso es suavizarlas.

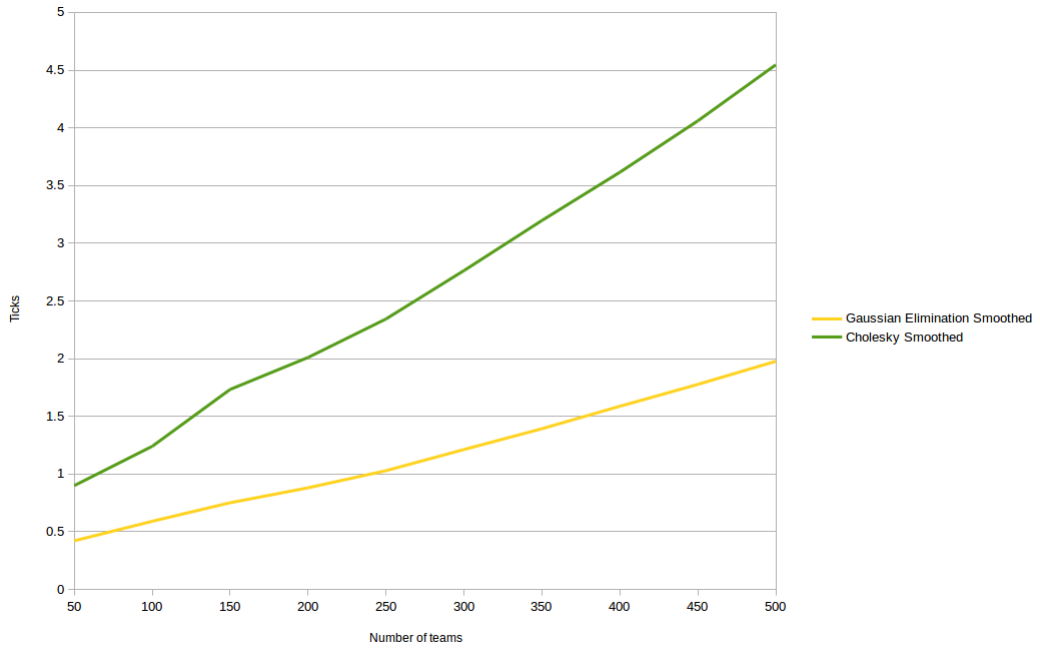

Figura 5: Comparación de complejidad temporal modificando el b en cada iteración suavizado

La curva de Gauss fue dividida por el cuadrado del tamaño de la entrada, mientras que Cholesky fue dividivo por el tamaño de la entrada. Ambas se ven lineales, de modo que podríamos pensar que la hipótesis es correcta. No obstante, vamos a dividir nuevamente a cada curva por el tamaño de la entrada y ver si la resultante se parece a una función constante.

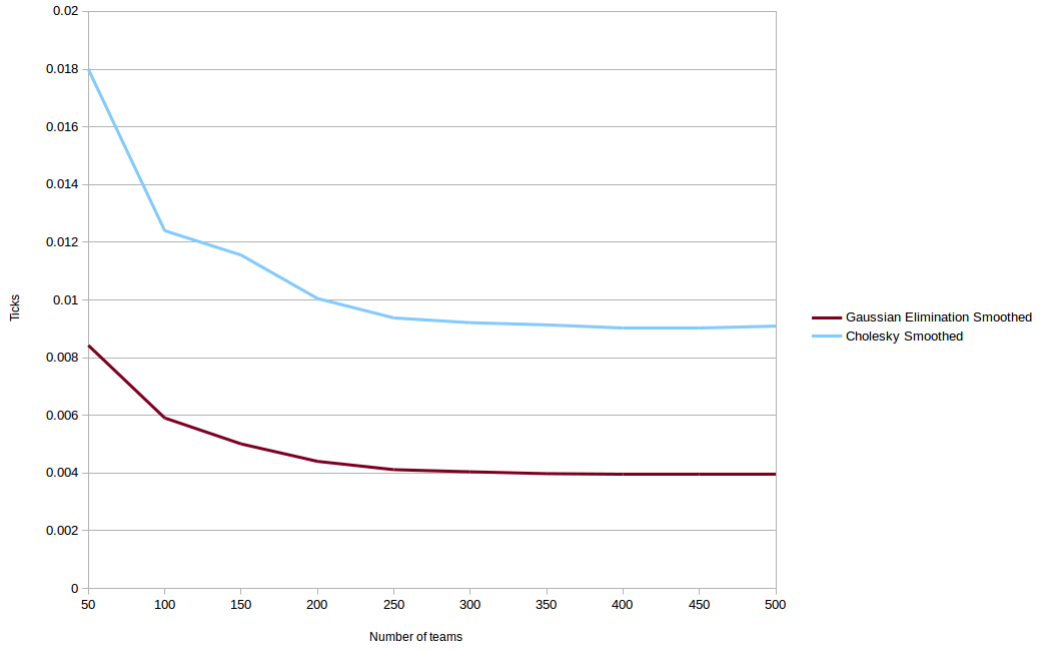

Figura 6: Comparación de complejidad temporal modificando el b en cada iteración suavizado

Efectivamente, a partir de muestras de tamaño 250 se aprecia que ambos algoritmos tienden a una constante. Las instancias de n pequeños tienen una cantidad de ticks levemente mayor pero, esto es atribuible a motivos de caché del procesador y procesos paralelos.

3.2. Análisis Cualitativo

3.2.1. Características distintivas de los métodos

Teniendo en una mano el *CMM* y en la otra el *Winning-Percentage*, uno estaría tentado a atinar una respuesta a la pregunta "¿Cómo se comparan?". En esta sección vamos a presentar resultados a partir de los cuáles se puedan observar diferencias o similitudes.

Veamos qué ocurre cuando ponemos a prueba ambos métodos en la NBA con la condicion de que todos los equipos disputaron *aproximadamente* 41 encuentros, que es la mitad de una temporada regular. Los resultados que obtuvimos fueron los siguientes:

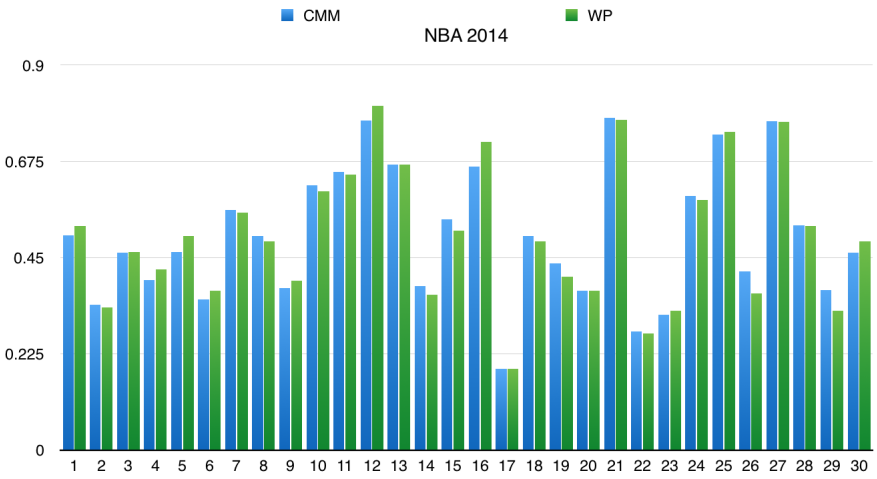

Figura 7: Season 2014

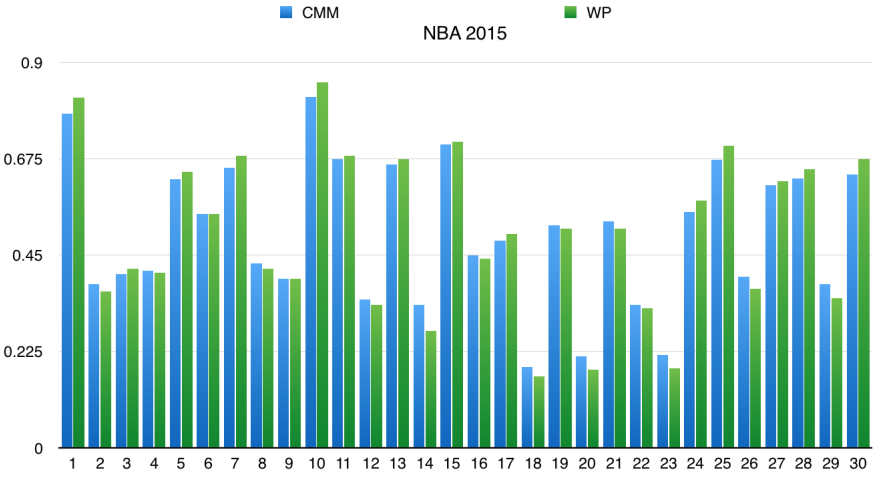

Figura 8: Season 2015

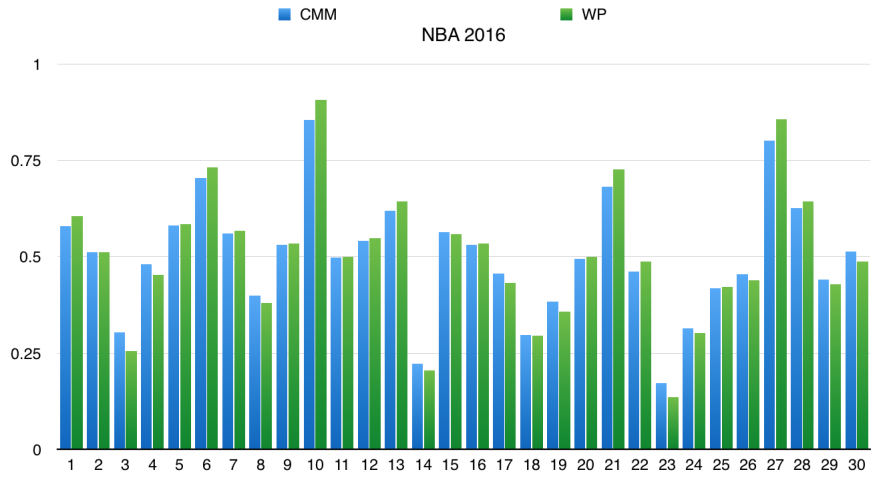

Figura 9: Season 2016

Se puede observar claramente que no hay diferencias sustanciales y esto tiene que ver con que observamos un caso *"promedio"*. Una explicación más matemática yace en lo planteado en **[1]**, en el estimador de r_i, obtenido a partir de la **Regla de sucesión de Laplace [2]**. Ocurre que $r_i = \frac{1+w_i}{2+n_i} \approx \frac{w_i}{n_i} \approx \frac{w_i}{k}$ donde este último termino es el valor exacto que se obtiene de plantear el *Winning-Percentage* sobre el equipo i, para este escenario.

Al ser un estimador, y tener similitudes con el mismo, tiene sentido que, siendo un caso aleatorio, a valores más grandes tiendan a parecerse.

Pero esto ocurre en un caso real, y la realidad tiende a comportarse parecido al promedio. Contrastemos esto con casos particulares.

Hipotesis: partidos ganados / partidos jugados, no tiene en cuenta la infomación sobre contra a quién les tocó jugar.

Para este experimento, lo plantedo fue un conjunto de participantes $\Gamma = \{1, 2, ..., 5\}$, donde se pone al equipo 1 en la siguiente situación:

El equipo 1 en 4 ocasiones vence al equipo 2, 3 al equipo 3, 2 al equipo 4 y, finalmente, 1 al contricante 5 y los resultados fueron:

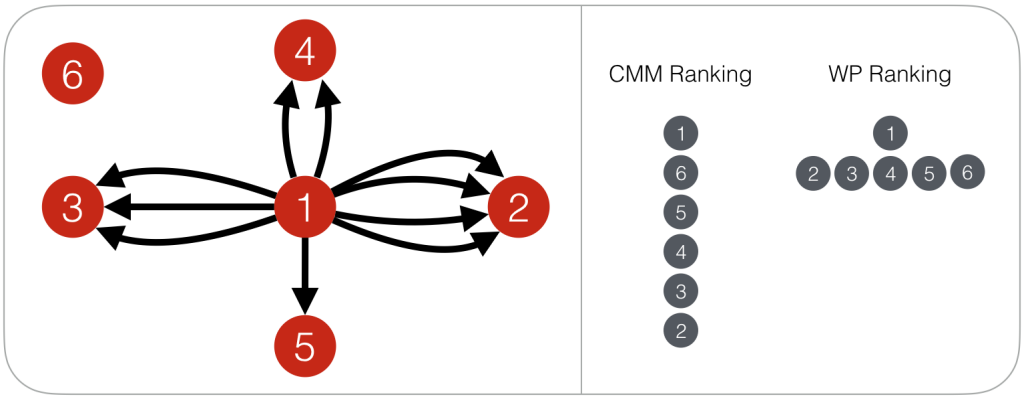

Figura 10: Experimento

Para ambos casos, el algoritmo WP asigna $r_1 = 1$ y $r_i = 0, i = 2, ..., 6$. [1] En cuanto a ratings respecta,

[1]Esto refuerza el hecho de haber usado **[2]** en el CMM: El resto de los equipos tiene el mismo rating habiendo perdido (además

el equipo 2 es efectivamente el último y por ende ocurre lo observado.

Hipotesis: El método CMM previene "sobre-calificar" a un equipo venciendo reiteradamente a un equipo con bajo ranking, en contraposición al WP. En otras palabras, esto quiere decir que no debería implicar el mismo aumento de rating vencer a un participante con rating elevado que a uno con rating bajo.

La configuración del schedule que buscamos para ejemplificar nuestra hipotesis fue uno tal que tenemos $\Gamma = \{1, 2, ..., 4\}$ en el cual los equipos 1, ..., 3 juegan entre sí y, por ende, podemos armar un rating *parcial* entre ellos. Luego, hacemos jugar al parcialmente "peor" participante, entre los mismos, contra el equipo 4 de tal forma que ambos algoritmos generan casos distintos. Esta fue la configuración:

Y el resultado fue:

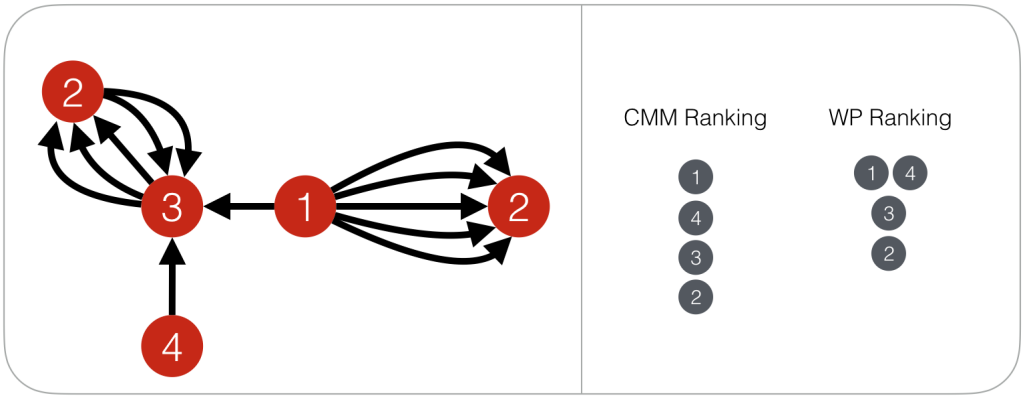

Figura 11: Experimento

La "lógica" indica que ganarle al peor de un grupo no debería implicar un ascenso importante en los ratings, mucho menos ser mejor o igual que el primero, y en este caso comprobamos que el WP no siempre sigue esa lógica.

Conclusión: Pudimos observar que en casos menos particulares, los criterios nos otorgan valores similares y que por otra parte situaciones más raras muestran diferencias sustanciales. En general, la diferencia tiende al hecho de que el CMM dá una identidad a los equipos, que tiene sentido puesto que la matriz de Colley está formulada en base a los datos particulares de cada participante. Por ende, los valores r_i en definitiva quedan en función de ellos. *Winning-Percentage* es un índice de efectividad a fin de cuentas. La utilidad fundamental del CMM es la de ponderar el "peso" de cada contricante en relación a sus victorias o derrotas, y, sumado a eso, previene *outliers* más efectivamente. Es por eso que el CMM resalta en *schedules* difusos y poco lineales.

3.2.2. El método es justo?

Hipótesis: El algoritmo genera el raiting de cada equipo en base a los raitings de los equipos contra los que jugó. Si un equipo no jugó contra nadie su rating es 1/2. Cuando el equipo A gana un juego, todos los demás equipos a los que le ganó van a tener un mayor rating ya que A ahora ganó más partidos y es más valioso. Si un equipo B no jugó nunca contra ningún equipo que haya jugado con A, el resultado de un nuevo partido de A no debe afectar a B.

Para graficar y analizar la situación vamos a graficar a los equipos como nodos de un grafo, y cada partido ganado es una arista dirigida con sentido al equipo que perdió.

de jugado) distinta cantidad de encuentros.

Además, vamos a dividir a la hipótesis en partes. Primero analizamos que pasa cuando tenemos varios equipos (por ejemplo 5 como en la figura) corriendo la instancia correspondiente obtenemos que los 5 equipos tendrán rating igual a 1/2, como queríamos ver.

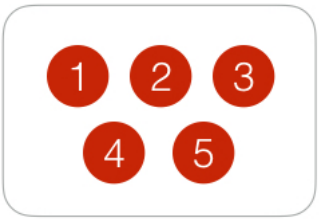

Figura 12: Equipos que no juegan nunca entre sí

Luego tomamos un grafo no conexo como el de la figura siguiente. Usando el CMM vemos que como dice la hipótesis, el rating de los equipos 1, 2, 3 y 4 no cambia al variar el resultado de los partidos entre el grafo conexo de 5, 6 y 7.

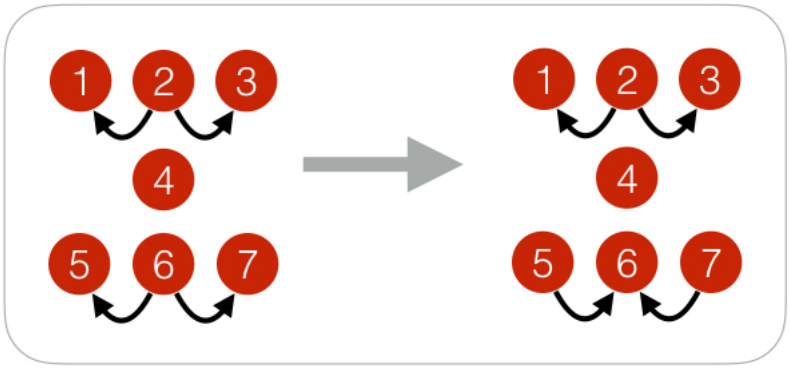

Figura 13: Modificación de resultado de partidos en subconjunto no conexo

Ahora que vimos que el resultado de un partido solo puede afectar a los nodos tales que existe un camino entre ellos, veamos el caso de un grafo conexo. Analizamos cómo afecta el resultado de un partido donde uno de los jugadores tanto ganó como perdió contra otros equipos y queremos ver cómo afectó al rating de los otros equipos. Antes de jugar el equipo 4 contra el 2 nos encontramos en una situación donde 2 tiene rating 1/2 y luego 1 es el que mayor rating tiene ya que le gano a 3 y a 4, luego 3 y 4 tienen mayor rating que 5, 6, 7 y 8. En el caso de que 4 le gane a 2, podemos observar utilizando el CMM que el equipo 1 aumenta su rating, como enunciamos en la hipótesis. Esto se da ya que 4 tiene más partidos ganados y es más valioso ganarle a 4 ahora que antes. Además el equipo 2 disminuye su rating ya que antes no había perdido y ahora sí. Pero lo extraño que observamos es que todos los equipos excepto el 2 aumentan su rating, incluso los que están muy poco relacionados, como el equipo 5. Esto se da ya que el rating de cada equipo está en relación al rating de los equipos contra los que jugó. Y al aumentar el rating de 4, aumenta el rating de 1 ya que 1 jugó contra 3 y contra 4. Por eso mismo aumenta el de 3, luego 5 y 6, etc. Para el caso de que 4 pierde contra 2 observamos la misma situación solo que 2 ahora aumenta su rating y todo el grafo excluyendo al 2 lo disminuye.

También analizando los números más en detalle pudimos observar que la jugada entre 2 y 4 afecta en proporción mucho mayor a los ratings de 2 y de 4 que al resto. De hecho cuantas más aristas de distancia más disminuye el impacto en el rating al punto de que el cambio en 5 y 6 es muy poco significativo.

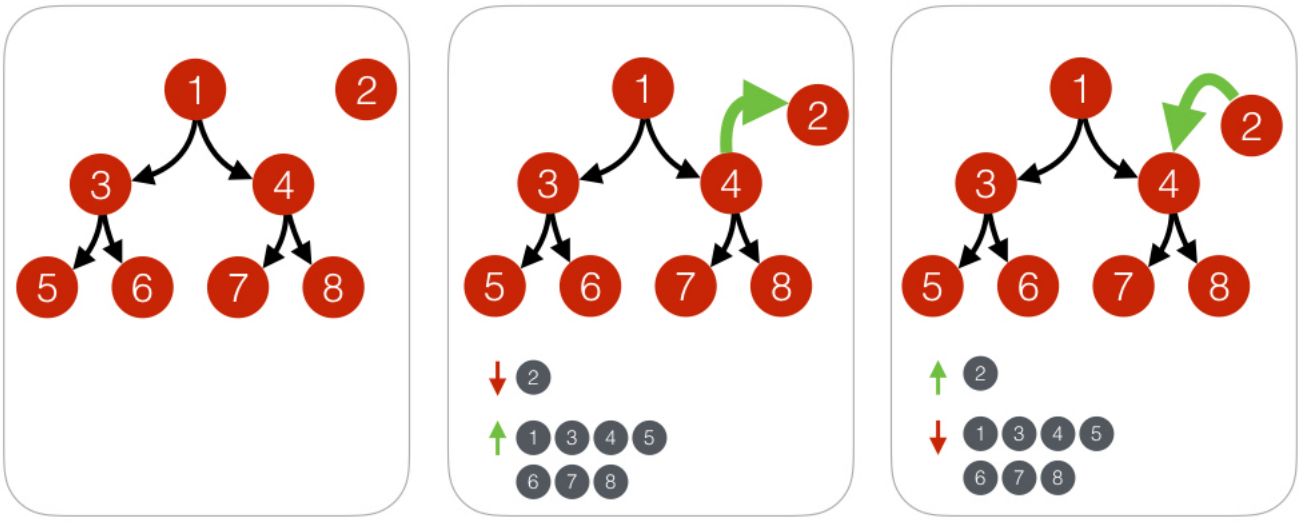

Figura 14: Modificación de raitings en base al resultado de un partido

Por último analizamos un caso particular donde el resultado de un partido indiréctamente puede alterar el ranking de otros equipos que no participaron de ese partido. En el ejemplo de la figura siguiente, el equipo 1 tiene muchos partidos ganados mientras que el 2 no tanto. Sin embargo si el equipo 2 le gana un partido al 1, los equipos 4 y 6 intercambian su posición en el ranking. Esto es un comportamiento muy poco esperado o poco intuitivo ya que por ejemplo podrían darse casos donde un equipo tiene la capacidad de decidir perder para afectar a otro equipo.

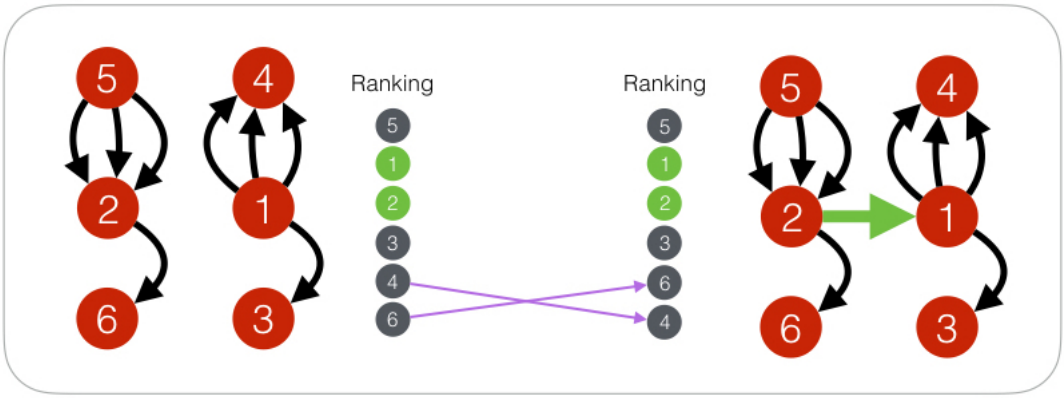

Figura 15: Modificación de rankings en base al resultado de un partido

Conclusiones: Aunque creemos que el CMM aporta mucho valor al lograr que ganarle a un equipo que gana mucho aumente más el rating que ganarle a un equipo que gana poco, notamos comportamientos que no son quizás tan intuitivos para establecer los resultados del juego. Algunos ejemplos de esto son los mencionados anteriormente como que no queda tan claro por qué a veces suben los ratings de unos equipos al ganar otros, o el hecho de que el resultado de un partido pueda alterar el ranking de otros que no participaron en ese partido. De todos modos creemos que es una muy buena aproximación para un manejo más real de los resultados de una competencia.

3.2.3. Mejor estrategia

Este apartado surge de la idea de analizar si es posible que un equipo alcance la cima del ranking, si los resultados hubieran sido distintos. Es decir, contamos con un schedule fijo, seleccionamos un equipo participante y tenemos la posibilidad de modificar los partidos perdidos del mismo a ganados. Interesa

minimizar la cantidad de modificaciones de este estilo tal que el equipo seleccionado quede en primera posición.

La estrategia que planteamos es la siguiente:

Sea EQUIPO* el equipo que interesa analizar

Calculamos los rankings con la entrada original.

Si ranking(EQUIPO*) es el máximo retornamos la cantidad de iteraciones.

Si no, buscamos el equipo con máximo ranking tal que EQUIPO* tenga algún partido perdido que pueda ser modificado.

Se invierte el resultado y recalculamos los rankings con este cambio.

En caso de que sea imposible modificar un partido (ya se modificaron todos los posibles), retornamos la posicion final.

Hipótesis: Los equipos con más derrotas deberán modificar más resultados que los que tienen menos.

Estos fueron los standings finales de la temporada 2014-2015 de la NBA. A la derecha figura la cantidad de partidos que cada equipo debería ganar en vez de perder para quedar en la primera posición según nuestro método.

Posición	Equipo	Record	Mínimo
1	Golden State	67-15	0
2	Atlanta	60-22	8
3	Houston	56-26	7
4	LA Clippers	56-26	8
5	Memphis	55-27	10
6	San Antonio	55-27	11
7	Cleveland	53-29	14
8	Portland	51-31	13
9	Chicago	50-32	18
10	Dallas	50-32	13
11	Toronto	49-33	18
12	Washington	46-36	21
13	New Orleans	45-37	19
14	Oklahoma City	45-37	19
15	Milwaukee	41-41	26
16	Boston	40-42	27
17	Phoenix	39-43	25
18	Brooklyn	38-44	29
19	Indiana	38-44	30
20	Utah	38-44	26
21	Miami	37-45	29
22	Charlotte	33-49	33
23	Detroit	32-50	35
24	Denver	30-52	35
25	Sacramento	29-53	34
26	Orlando	25-57	41
27	LA Lakers	21-61	44
28	Philadelphia	18-64	48
29	New York	17-65	50
30	Minnesota	16-66	47

Golden State quedó en primera posición y por lo tanto requiere 0 partidos para tener el mejor ranking. No es algo sorprendente, pues además ganaron los Play-Off.

Para los demás equipos, la estrategia indica que deben cambiar de perdidos a ganados, una cantidad de partidos tales que consigan una score final similar al que quedó en primera posición.

Hay un caso interesante entre Portland, Chicago y Dallas, donde Chicago tiene el mismo record que Dallas, pero necesita 5 partidos más que los otros dos equipos. Esto se debe al schedule de cada uno.

Seguramente Chicago jugó pocos partidos frente a los equipos por encima de él, y probablemente haya ganado algunos. Mientras que Portland y Dallas deben haber jugado más veces contras los mejor rankeados y perdido en la mayoría de las oportunidades, con lo que una modificación en esos partidos, les significa un salto mayor en la escalada por el ranking.

Dada la observación sobre alcanzar el score del que finalizó primero, probamos también qué sucede en el caso en que por ejemplo Golden State, ganara absolutamente todos los partidos. ¿Cuántos encuentros deberán ganar los demás equipos para tener un ranking mejor?

Corrimos el algoritmo con este nuevo schedule y observamos que efectivamente los resultados refuerzan las observación, pidiéndole a cada equipo que ganara cerca de 80 partidos sobre 82 totales (ganar en vez de perder con Golden State para disminuir su ranking y ganarle a casi todos los demás pues Golden State lo había hecho).

Con respecto a la hipótesis planteada, podemos decir que a peor score, mayor cantidad de partidos necesarios que intercambiar, aunque está intimamente ligado al schedule de cada equipo, especificamente a la cantidad de partidos jugados frente a los equipos mejor posicionados. Esto se ve porque la progresión de scores es lineal (16), pero la de ratings zigzaguea, tendiendo a una parecerse a una función lineal con bajas repentinas (17)

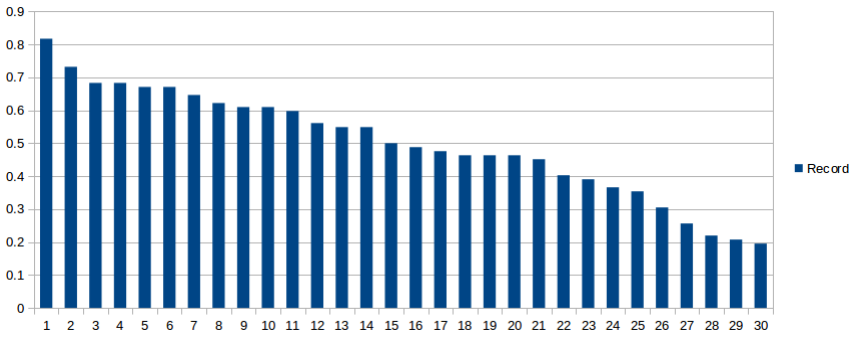

Figura 16: Scores finales

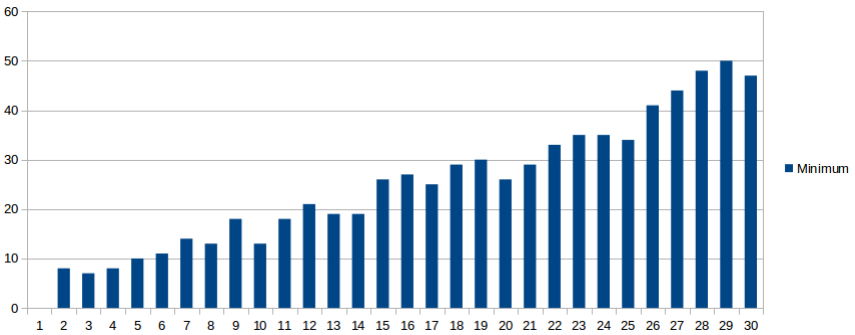

Figura 17: Número mínimo de partidos a modificar para alcanzar el primer puesto

4. Conclusiones

En el trabajo realizamos múltiples experimentos que analizaron tanto las implementaciones del CMM como las aplicaciones y pudimos comparar sus características con las de otros métodos como el WP en situaciones de Sports Analytics.

Por un lado implementamos el CMM con dos algoritmos diferentes: la eliminación gaussiana y la factorización de Cholesky. Pudimos ver que mientras que ambos tienen una complejidad temporal de $O(n^3)$, en Cholesky la factorización de la matriz no altera el término independiente. Por lo tanto el costo de $O(n^3)$ se realizaba una única vez y luego para cada vez que se corre el algoritmo con un término independiente diferente la complejidad era únicamente $O(n^2)$. Para todo esto realizamos experimentos que reflejaron nuestras hipótesis, obteniendo una demostración empírica.

Luego analizamos las características particulares del CMM.

Por un lado hicimos comparaciones con el método de Winning Percentage. Pudimos ver que el CMM aportaba identidad a los equipos ya que no se trata únicamente de ganar o perder sino que tiene en cuenta contra quién se juega. Ganarle a un equipo que tiene mayor cantidad de partidos ganados tiene mayor peso que ganarle a uno que siempre perdió. El WP termina generando un ranking que no representa idealmente los resultados de un torneo donde no hubo suficientes partidos.

Posteriormente indagamos si el CMM producía un ranking justo. Para esto analizamos distintas schedules con características particulares. Observamos por ejemplo que no siempre los ratings generados por el algoritmo eran tan intuitivos, ya que como relaciona los ratings entre los equipos, un partido entre dos afecta el rating de muchos otros que no participaron de ese partido. Por otro lado detectamos un caso más grave que se genera cuando el resultado de un encuentro entre dos equipos altera el ranking de otros dos que no participaron del mismo. De ese modo podrían, por ejemplo, existir equipos que decidan perder para alterar el ranking de otros.

Por último, creamos un algoritmo greedy que debía lograr obtener el mejor ranking posible de un equipo realizando una cierta cantidad de modificaciones de los resultados de sus partidos. El algoritmo que implementamos tiene en cuenta la característica de CMM de que ganarle a un equipo con alto rating aporta una mayor cantidad de rating que ganarle a uno con menor. De esta forma obtenemos un resultado minimal de la cantidad de partidos que deberían haber sido modificados.

De este modo creemos que la adopción de un algoritmo como el CMM tiene muchas ventajas frente a otros más simples como el WP. Da identidad a los equipos y genera un ranking más justo (aunque con sus defectos), especialmente para casos donde hubo pocos partidos entre los equipos.

5. Referencias

[1] Wesley N. Colley. Colley's Bias Free College Football Ranking Method: The Colley Matrix Explained. Ph.D., Princeton University

[2] Análisis Numérico - Burden & Faires

[3] `https://es.wikipedia.org/wiki/Factorizaci%C3%B3n_LU`

[4] `https://es.wikipedia.org/wiki/Regla_de_sucesi%C3%B3n`