Memoria de divide y vencerás

Rubén Morales Pérez Francisco Javier Morales Piqueras
Bruno Santindrian Manzanedo Ignacio de Loyola Barragan Lozano
Francisco Leopoldo Gallego Salido

22 de mayo de 2016

Índice

1. División en dos equipos

1.1. Enunciado

Se desea dividir un conjunto de n personas para formar dos equipos que competirán entre sí. Cada persona tiene un cierto nivel de competición, que viene representado por una puntuaciónn (un valor numérico entero). Con el objeto de que los dos equipos tengan un nivel similar, se pretende construir los equipos de forma que la suma de las puntuaciones de sus miembros sea lo m ás similar posible. Diseña e implementa un algoritmo vuelta atrás para resolver este problema. Realizar un estudio empírico de la eficiencia de los algoritmos.

1.2. Explicación

Para simplificar el problema y darle mayor flexibilidad identificamos a cada jugador única y exlusivamente por su puntuación, aunque puede haber un jugador con la misma puntuación. Esto permite que si hay dos jugadores con nivel de competición x e y con $x = y$ y cada uno está en un equipo podrían intercambiar sus posiciones sin descompensar los equipos.
Tenemos n jugadores con puntuación respectiva x_i $\forall i \in I = [1, 1000] \cap \mathbb{N}$. Si inicialmente tuviésemos los niveles de los jugadores en otro rango $J = [a, b]$ $a, b \in \mathbb{N} : a < b$ podemos dejarlo así, ya que el rango de los niveles no afecta al problema, salvo que apliquemos una transformación que reduzca el rango, ya que podríamos perder información por el truncamiento de los números enteros.
En cualquier caso la transformación al dominio del problema del intervalo J es $\forall j \in J$ aplicaríamos una función $F : [a, b] \to [1, 1000] \cap \mathbb{N}$, teniendo en cuenta que $E(x)$ es la función parte entera:

$$F(x) = \begin{cases} E\left(\frac{x-a}{b-a} \cdot 1000\right) + 1 & \text{si } \left(\frac{x-a}{b-a} \cdot 1000\right) < E\left(\frac{x-a}{b-a} \cdot 1000\right) + 1/2 \\ E\left(\frac{x-a}{b-a} \cdot 1000\right) & \text{en otro caso} \end{cases}$$

1.3. Eficiencia

En cuanto a la eficiencia de este algoritmo, podemos representar los n jugadores como un vector de booleanos, donde si un jugador toma el valor *true* está en un equipo y si toma el valor *false* está en otro. De esta forma observamos que podemos representar la división por equipos en un único vector. Tomando el primer jugador, puede estar en un equipo o en otro, 2 combinaciones. Tomando el siguiente jugador puede estar en uno u otro, independientemente de donde estuviese el otro jugador, 2^2 combinaciones. El tercer jugador vuelve a tener 2 opciones, 2^3 combinaciones.
Al tratarse de un algoritmo de vuelta atrás (backtraking) tenemos que recorrer todas las opciones, por lo que el problema queda representado como un árbol binario donde cada nodo tiene dos opciones, estar en un equipo o en otro.

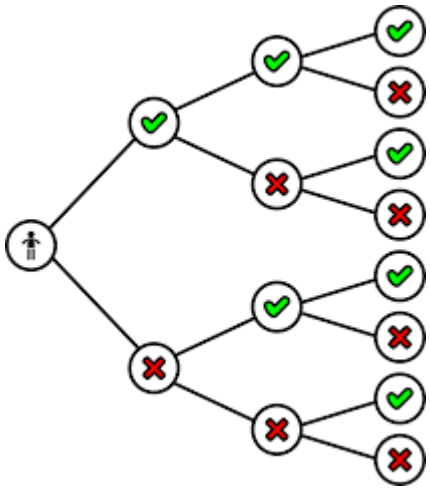

Figura 1: Distribución de equipos

De aquí deducimos la eficiencia del algoritmo con respecto al número total de jugadores $O(n) = 2^n$. Llegados a este punto nos podríamos plantear si podemos mejorar el algoritmo, curiosamente una forma de mejorarlo será ponerle una restricción, ya que nos ahorrará muchas comprobaciones. La restricción nombrada será forzar que el número de jugadores en los dos equipos sea parecido, la diferencia de jugadores entre ambos equipos no será mayor de 1. Esto hace que si hay un número de jugadores par ambos equipos siempre tengan el mismo número de jugadores. Ahorramos comprobaciones, pero el algoritmo sigue teniendo eficiencia $O(n) = 2^n$ ya que pasaremos por todas las opciones, pero cuando la combinación en la que estemos no cumpla esa restricción la ignoraremos. La ventaja de este planteamiento se verá más adelante.

1.4. Comparando ambos algoritmos

EL rango de jugadores usado ha sido $I = [3, 26] \cap \mathbb{N}$, vemos como el estudio empírico corrobora las deduciones previamente explicadas. Los ajustes en ambos casos han sido hechos a funciones $f(n) = a \cdot 2^n$ $a \in \mathbb{R}$ $\forall n \in I$. Sus coeficientes r^2 son 1 en ambos casos y el término a está especificado en las gráficas, aunque también se encuentra en el archivo *Equipos/Datos/ajustes.txt*.

1.4.1. Tiempos

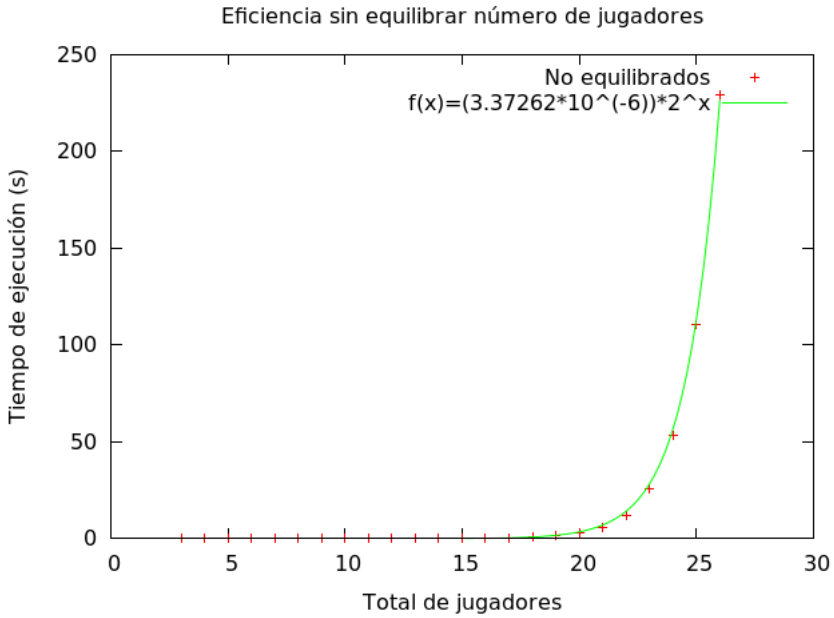

Figura 2: Distribución de equipos sin equilibrar

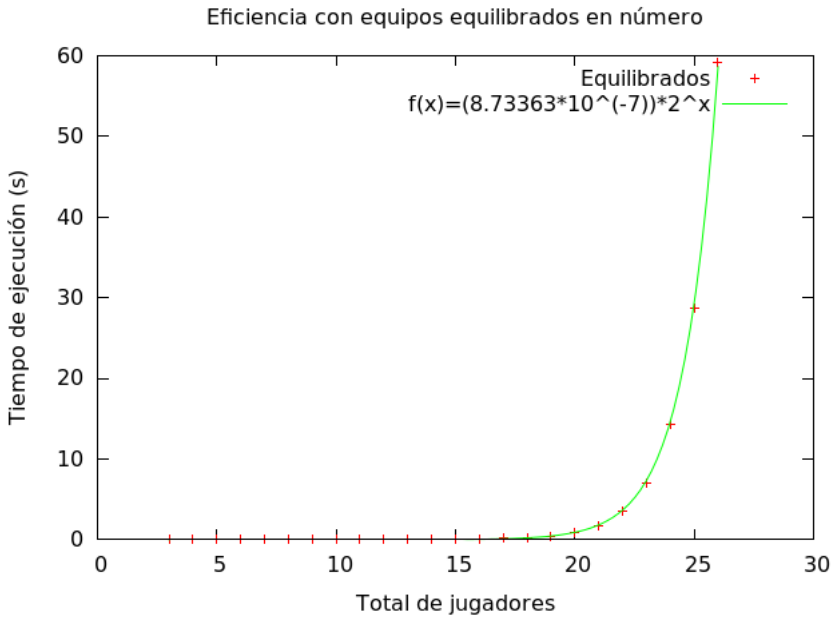

Figura 3: Distribución de equipos equilibrados en número

Comparando ambos tiempos:

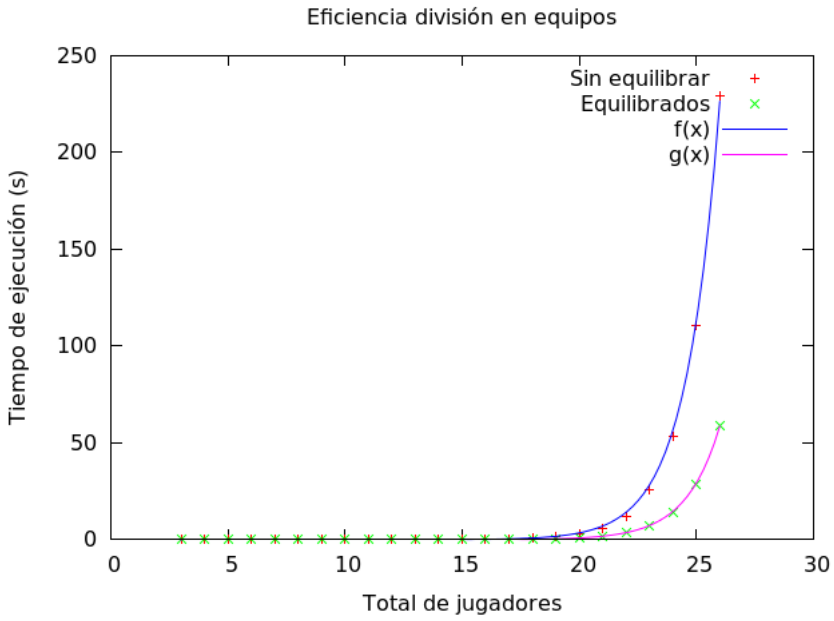

Figura 4: Comparativa de tiempo

1.4.2. Diferencia de nivel

Recordemos que un jugador tiene un nivel entre 1 y 1000, es lógico pensar que con pocos jugadores el margen de optimización del algoritmo es menor debido al gran peso que toma la aleatoriedad con la que se determinan los niveles.

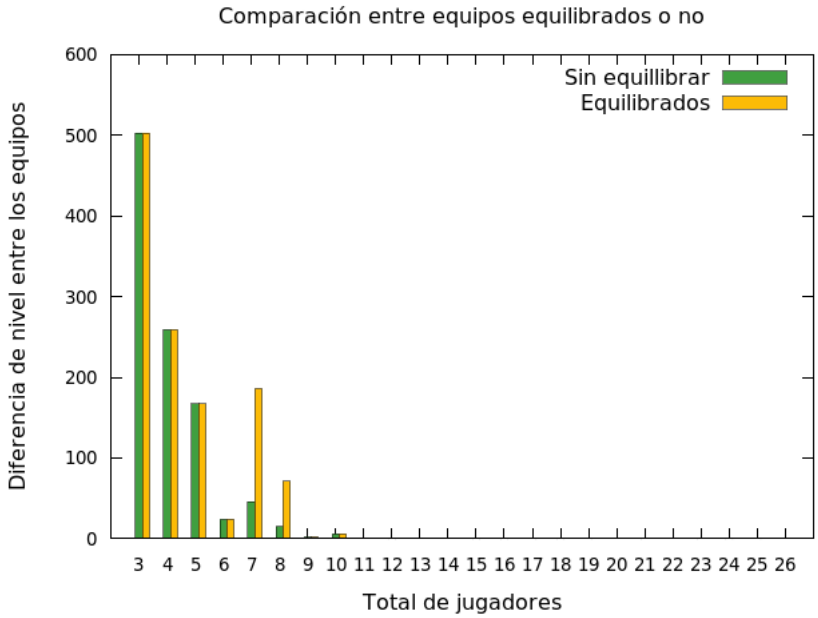

Figura 5: Diferencia de nivel en los equipos

1.4.3. Diferencia de jugadores

Al ajustar la diferencia con el segundo algoritmo vemos que el primero tiende a desajustar el número de jugadores, llegando a tener los equipos una diferencia de 4 jugadores cuando siendo 24 en total.

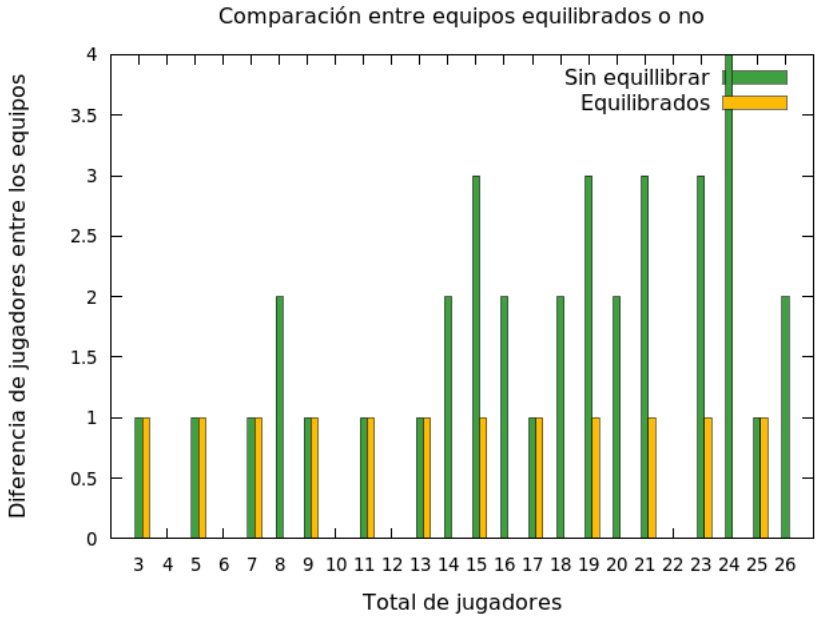

Figura 6: Diferencia en número de jugadores

1.5. Conclusión

Al ver la dificultad exponencial del problema, si tenemos valores mayores de 30 podríamos llegar a plantearnos algoritmos más rápidos. Para instancias del problema con un número pequeño de elementos el algoritmo planteado nos sirve, pero si aumentamos el número puede que la fuerza bruta no sea tan buena idea.
Una forma de disminuir el tiempo del algoritmo es evitar comprobaciones. Esto lo podemos hacer con algún tipo de corte del árbol de soluciones, lo que denominaremos “Branch and Bound", podríamos conformarnos con que la diferencia de nivel fuese menor que un $n \in \mathbb{N}$, de forma que no seguiríamos recorriendo el árbol una vez encontrada una solución satistactoria. Otra opción para disminuir el tiempo es paralelizar el problema, podemos hacer que cada procesador compruebe una rama diviendo el tiempo total.
Como hemos comprobado, el segundo algoritmo iguala al primero en eficacia, pero lo supera en eficiencia. Esto nos hace pensar que puede no ser necesario siempre recorrer el árbol de soluciones para asegurarnos la optimalidad.

2. Bibliografia