Práctica 2 - eSportsLS

Optimización combinatoria

Martí Ejarque Galindo (marti.ejarque)
Victor Xirau Guardans (victor.xirau)

Enero 2020

Contents

1 Explicación detallada de los algoritmos **3**
1.1 Justificación de métodos seleccionados para cada criterio . . . 3
1.2 Opcionals 6

2 Comparativa de algoritmos **7**
2.1 Comparativa 7
2.2 Método de pruebas usado 8

3 Análisi de los resultados **9**
3.1 Menú más equilibrado 9
3.2 Fase de grupos 10

4 Problemas observados **13**

5 Conclusiones **14**

6 Bibliografia **16**

1 Explicación detallada de los algoritmos

1.1 Justificación de métodos seleccionados para cada criterio

- Menú más equilibrado:

Se ha decidido emplear este algoritmo con este criterio ya que en esta parte de la práctica, se nos pide crear un menú con ciertas restricciones. Està claro que estas restricciones las pueden pasar más de una combinación de platos, por lo tanto necesitamos explorar todas las soluciones, ya que todas pueden ser solución. Una vez se han consultado todas las combinaciones que superan las restricciones, se mira cual es la más óptima. Por lo tanto, necesitamos backtracking para explorar todas las soluciones y greedy para podar todas aquellas soluciones parciales que sabemos que cumpliran con la restriccion, y por lo tanto podrán ser solución.

```
public static void foodBacktracking(Food[] f, int k, int[] config, Double[] max, int[] opt){
config[k] = 0;
boolean trobat = false;
while(config[k] < 2){
  if(sumaValors(config) > 4 || k == f.length - 1){
              if(isValid(f,config,k, fact: false,max, opt)){
                  trobat = true;
              }
  }else{
              if(isValid(f,config,k, fact: true,max, opt)){
                  foodBacktracking(f, k: k+1, config, max, opt);
              }
          }
  config[k]++;
  }
}
```

- Poda:

En este caso, la poda consiste en eliminar todos esos hijos los cuales la suma de las calorías con los hijos ya seleccionados juntamente con el actual o bien superan las 3000 kcal o superan el 35% de grasas. Cuando estams delante de una posible solución, miramos que las calorias del menú estan entre 1000 y 3000 kilocalorías, y que las grasas consumidas, en gramos, estan entre un 20% y un 35% de las kilocalorías. Si pasa esas dos restricciones, para quedarnos con el menú más óptimo, miraremos que la relación kcal/grasa sea lo más próximo a 5, ia que es el índice más óptimo que podemos construir para un menú.

- Configuración:

Para guardar-nos los mejores menús (los que cumplen con las restricciones), necesitamos un array de integers (de largada igual a la cantidad de platos), donde ponemos a 1 el plato seleccionado y a 0 el que no lo està, teniendo en cuenta que solo pueden haber 5 platos en el menú.

- Fase de grupos:

En este caso, Branch Bound será una buena opción a escoger, ya que nos permitirà eliminar muchas brancas del árbol de soluciones desde el inicio. Concretamente utilizaremos el LC-BranchBound, con el objetivo de seleccionar esos nodos del árbol de soluciones que tengan el mínimo counter, es decir, de cada etapa miraremos cual tiene el mínimo counter y seguidamente eliminaremos los otros nodos de dicha etapa.

Para la fase de grupos hemos creado una funcional adicional que nos ordena los equipos de tal manera que nos genera una matriz de tamaño (total equipos)%6 x 6. Esta matriz la llenamos del tal manera que los españoles quedan todos en la primera fila ordenados por winrate y las demas casillas a null. No solo eso sinó que en el array de equipos todas aquellas casillas que contenian españoles ahora estan a null. Esta ordenación por winrate permite que, si mediante nuestro metodo la repartición no es justa para los españoles, como mínimo será aquel equipo con mayor winrate el que tendrá un mayor numero de counters en su grupo permitiendo asi que no sea el equipo con menor winrate el que se enfrente al mayor numero de counters.

```
public static void grupsBnB(Equip[][] g, Equip[] e, int[] config, Champion[] champ, int k, int Esp, int indise) {

    int j = 0;
    while (config[j] < g.length && k < Esp) {

        int d = totalCounters(champ, e[j], g, k);
        if (d < millor) {
            if (config[j] == 0) {
                millor = d;
                millorX = j;
            }

        }

        if (j == e.length-1) {

            config[millorX] = total + 1;
            total++;
            j = 0;
            millor = Integer.MAX_VALUE;
            g[k][indise] = e[millorX];
            e[millorX] = null;
            grupsBnB(g, e, config, champ,  k: k + 1, Esp , indise);

        } else {
            j++;
        }
    }
}
```

- Poda:

Para la fase de grupos, la poda consistirá en eliminar todos aquellos grupos de nacionalidad no española que, en la misma etapa de consulta, tengan una cantidad de counters más elevada que el nodo con menos counter de dicha etapa, por lo que estamos eliminando, por cada etapa, todos los equipos que hacen más counter al equipo español consultado.

- Configuración:

La configuración serà un array de integers, de largada igual a la cantidad de equipos presentes en el dataset seleccionado, en donde indicaremos cual es el equipo que hace menor counter, y a que español, es decir, se indicarà cual es el que hace menor counter con el número de la posición correspondiente al español, el qual tiene esa relación de mínimo counter con el equipo seleccionado.

1.2 Opcionals

Al empezar la codificación de la practica la habiamos planteado teniendo en cuenta uno de los datasets que posteriormente se han marcado como inutil, o no necesario. Aun así, nosotros ya habíamos empezado a codificar la practica i hemos decidido considerar su implementación como una implementación opcional. Para ello seguimos utilizando los equipos que hemos colocado en la posición 0 de nuestra matriz, ordenados por winrate, y asi seguimos teniendo como mínimo favoritismo por un español. Para su comprobación también hemos creado datasets adicionales como el 48+ y el 48++ en los que hay algun español de mas o mas de 2 españoles mas por grupo.

2 Comparativa de algoritmos

En esta práctica, se ha trabajado con dos algoritmos: BackTracking y Branch Bound. Estos algoritmos nos permiten buscar soluciones a los problemas planteados de manera que se contemplen todas esas posibilidades que den solución al problema. Pero cada algoritmo emplea una lógica diferente, que veremos a continuación.

- BackTracking:

Este método de búsqueda de soluciones consiste en buscar todas las soluciones posibles a un problema, todo eso explorando un árbol de soluciones(fictício, que se va generando a medida que se explora). Por cada nodo actual en el que estamos, exploramos sus hijos (nodos siguientes), hasta llegar a una solución, sea válida o no. Al ser un algoritmo recursivo, éste explora el árbol de bajada hasta al fondo, y después sube hasta llegar a un nodo con otro hijo no explorado.

- Branch & Bound:

En este método, exploramos los diferentes nodos con sus respectivos hijos, pero no por profundidad, sino por anchura del árbol de soluciones. Hay diferentes métodos de implementar el BB. En nuestro caso se explora los diferentes nodos que están en la misma altura, y se selecciona ese que tenga el mínimo coste.

2.1 Comparativa

De primeras, se puede apreciar que BackTracking tiene un coste más elevado que BB, debido a que en Backtracking contemplamos todos las soluciones, y posteriormente seleccionamos las que creamos convenientes, mientras que BB nos retorna directamente la solución que se busca. Pero este coste se ve minimizado con la implementación del método Greedy.

- Greedy en BackTracking:

El método Greedy aplicado a BackTracking nos permite descartar posibles brancas del árbol de soluciones que sabemos (debido a una restricción) que ia no son factibles. Esto hace de la búsqueda de una solución (sea la más

óptima o no) una tarea con menos coste asintótico. Posteriormente necesitaremos una función que nos diga de todas las soluciones encontradas, cual es la más óptima.

Por lo tanto, el coste de BackTracking sin Greedy es de $O(m^n)$según el teorema Muster Theorem, siendo m las diferentes ramas dentro de un nodo y n la cantidad de "capas" dentro del árbol de decisiones. Peró, con el criterio greedy, este coste pasa a ser de $O(n*m^n)$ según el teorema Muster Theorem, ya que por cada nodo, comprobamos que sea válido (que pase las restricciones que se imponen en el problema).

En cambio, el algoritmo Branch Bound tiene un coste máximo igual al coste de un BackTracking, pero de coste medio obtenemos un coste asintótico mucho mejor, sobretodo para datasets más grandes. Esta situación es debido a que podamos muchas ramas que sabemos que ya no nos darán una solución óptima, y por lo tanto tenemos un coste computacional más bajo.

2.2 Método de pruebas usado

Para comprobar que los resultados son los aparentemente correctos, se ha printado por cada vuelta de la recursividad , la configuración anteriormente planteada, por lo que se he ido viendo cómo las distintas configuraciones apuntaban a la solución esperada. Para comprobar que la poda se hacía en algún caso, se ha printado dentro de la condición de poda una señal para saber que se ha realizado. Lo mismo se ha hecho cuando el programa ha considerado que había encontrado una solución, para nosotros posteriormente comprobar que era una solución válida.

3 Análisi de los resultados

3.1 Menú más equilibrado

Tal y como hemos mencionado con anterioridad, el BackTracking aumenta su coste para datasets de gran información. Como podemos observar a continuación, todo y teniendo en cuenta el greedy, la evolución del coste de ejecución de cada dataset sigue una evolución exponencial.

- Dataset50:

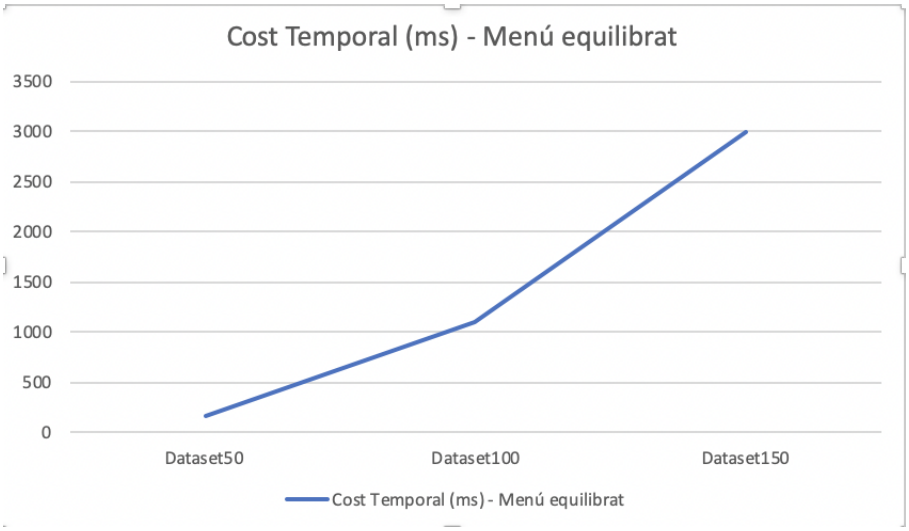
```
El algorisme ha trigat 160133132ns  161ms  0s  en executar-se.



La combinació de plats més òptima és:
    Plato 1: Naan con Pato con Guindilla con Cordero con Ciruela contiene 930kcal y 208.95g de grasas.
    Plato 2: Rabanitos con Salchicha con Guisantes con Guindilla con Pimiento contiene 534kcal y 123.004005g de grasas.
    Plato 3: Pepino con Membrillo con Cebolla con Guindilla con Escaramujo contiene 295kcal y 12.315g de grasas.
    Plato 4: Plátano con Membrillo con Coliflor con Rabanitos con Tortilla integral contiene 337kcal y 74.937004g de grasas.
Aquest menu te un total de: 2096.0kcal y 419.206009g de grasses.
Aquestes grasses suposen el: 20.000286688931297% del total de calories.
```

- Dataset100:

```
El algorisme ha trigat 1109130140ns  1109ms  1s  en executar-se.



La combinació de plats més òptima és:
    Plato 1: Kiwi con Escaramujo con Calabaza con Beicon con Berenjena contiene 901kcal y 262.221g de grasas.
    Plato 2: Cerezas con Berenjena con Pechuga de pavo con Carne grasa de cerdo con Albaricoque contiene 539kcal y 83.499g de grasas.
    Plato 3: Vino con Tallarines cocidos con Salami con Colinabo con Frambuesas contiene 1022kcal y 179.30199g de grasas.
    Plato 4: Arándano rojo con Pumpernickel con Lichi con Tortilla integral con Lucio contiene 513kcal y 69.983g de grasas.
Aquest menu te un total de: 2975.0kcal y 595.0049899999999g de grasses.
Aquestes grasses suposen el: 20.000167731092432% del total de calories.
```

- Dataset150:

```
El algorisme ha trigat 2988413698ns  2989ms  2s  en executar-se.



La combinació de plats més òptima és:
    Plato 1: Espinaca con Naan con Ternera con Galletas con chocolate con Gin-tonic contiene 1296kcal y 179.056g de grasas.
    Plato 2: Col rizada con Filete de perca con Fresa con Boniato con Limón contiene 303kcal y 12.103001g de grasas.
    Plato 3: Queso emmental con Tallarines cocidos con Arándano con Pera con Fideos cocidos contiene 773kcal y 339.903g de grasas.
    Plato 4: Trucha con Ciruela con Filete de perca con Maracuyá con Arenque contiene 451kcal y 33.541g de grasas.
Aquest menu te un total de: 2823.0kcal y 564.603001g de grasses.
Aquestes grasses suposen el: 20.000106305348915% del total de calories.
```



3.2 Fase de grupos

Para la fase de grupos (implementado en BranchBound), podemos apreciar como el coste de los distintos datasets se mantiene estable, en comparación con el coste de los datasets del menú. Podemos apreciar que entre el dataset de 12 i el dataset de 48 sólo hay una diferència de 13 ms, cosa que en el anterior criterio, la diferencia entre los más grandes es de 2828 ms.

- DatasetP12:

```
Per 12 equips s'han fet 2 grups.
Hi ha un total de 2 espanyols.
La distribució per grups és la següent:

    team 6[España] - 38.87          team 1[España] - 45.57
    team 3[EEUU] - 11.32 - 0        team 4[Francia] - 54.58 - 0
    team 5[EEUU] - 94.69 - 0        team 8[Korea] - 61.61 - 0
    team 7[EEUU] - 35.61 - 0        team 9[EEUU] - 79.52 - 0
    team 10[Francia] - 54.44 - 0    team 12[Korea] - 3.34 - 0
    team 11[Korea] - 11.43 - 0      team 2[Korea] - 91.36 - 1


El algorisme ha trigat 5947162ns  6ms  0s  en executar-se.


El format es el següent: <nom_equip>[<nacionalitat>] - <winrate> - <total_counters>
```

- DatasetP12+:

```
Per 12 equips s'han fet 2 grups.
Hi ha un total de 3 espanyols.
La distribució per grups és la següent:

    team 6[España] - 38.87          team 1[España] - 45.57
    team 12[España] - 3.34 - 0      team 3[EEUU] - 11.32 - 0
    team 4[Francia] - 54.58 - 0     team 5[EEUU] - 94.69 - 0
    team 7[EEUU] - 35.61 - 0        team 8[Korea] - 61.61 - 0
    team 9[EEUU] - 79.52 - 0        team 10[Francia] - 54.44 - 0
    team 11[Korea] - 11.43 - 0      team 2[Korea] - 91.36 - 1


El algorisme ha trigat 4508685ns  5ms  0s  en executar-se.


El format es el següent: <nom_equip>[<nacionalitat>] - <winrate> - <total_counters>
```

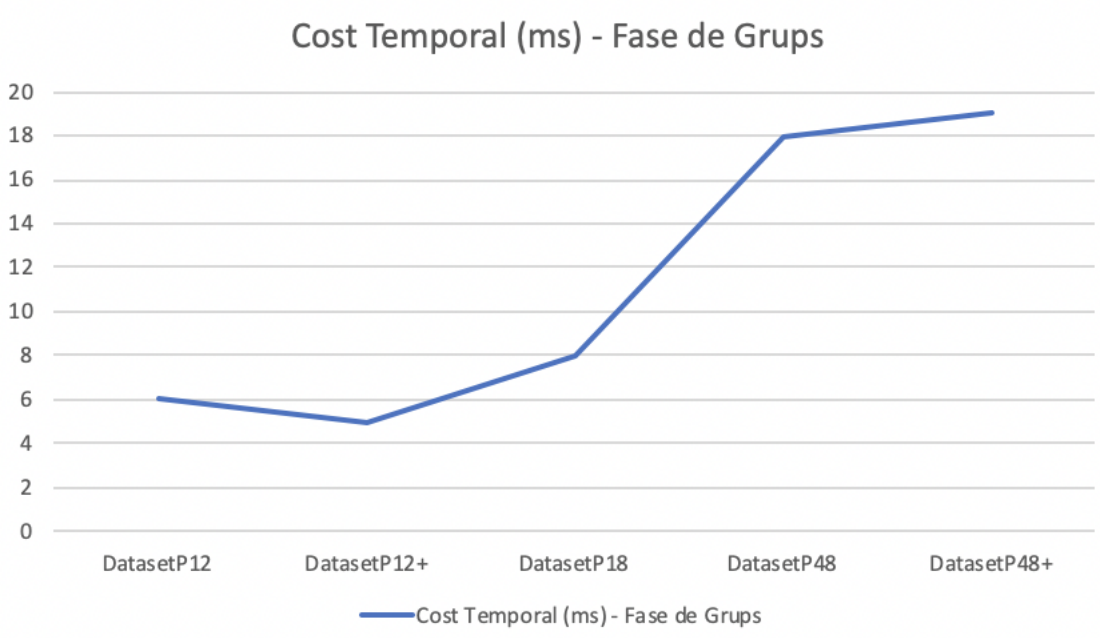
- DatasetP18:

```
Per 18 equips s'han fet 3 grups.
Hi ha un total de 3 espanyols.
La distribució per grups és la següent:

    team 12[España] - 3.34          team 6[España] - 38.87          team 1[España] - 45.57
    team 2[EEUU] - 91.36 - 0        team 3[Korea] - 11.32 - 0       team 4[Canada] - 54.58 - 0
    team 5[Korea] - 94.69 - 0       team 7[Canada] - 35.61 - 0      team 8[Korea] - 61.61 - 0
    team 9[EEUU] - 79.52 - 0        team 10[EEUU] - 54.44 - 0       team 13[Korea] - 39.72 - 0
    team 11[Canada] - 11.43 - 0     team 14[Korea] - 39.7 - 0       team 15[Francia] - 6.66 - 0
    team 16[EEUU] - 56.89 - 0       team 18[Korea] - 95.48 - 0      team 17[EEUU] - 51.26 - 0


El algorisme ha trigat 7649827ns  8ms  0s  en executar-se.


El format es el següent: <nom_equip>[<nacionalitat>] - <winrate> - <total_counters>
```

- DatasetP48:

```
El algorisme ha trigat 18390972ns  18ms  0s  en executar-se.
```

- Coste temporal:



4 Problemas observados

En cuanto a los problemas observados ha habido bastantes. Hemos afrontado varios problemas a lo largo de la codificación de la práctica pero no todos son dignos de mencionar en nuestra memoria. Un problema común en las dos implementaciones, tanto en backtracking como en branch and bound, sobretodo en branch and bound, ha sido la ideación e implementación de la configuración. En el caso de backtracking nos hemos acabado remitiendo a un array de integers con valores entre 0 y 1 indicando si aquel elemento era parte de la solución o no. Esto funciona para el primer algoritmo porque había que señalar que combinación de platos daban el total de calorías y grasas deseado pero no nos servía para el segundo algoritmo en el que necesitábamos más información de la configuración. Para la segunda hemos acabado optando por una configuración que iba cambiando para cada fila de la matriz que hemos explicado anteriormente. A cada iteración se inicializa a 0, cada casilla corresponde a un equipo y se llena con los valores numéricos que corresponden, cada vez, a los elementos con menos counters con el equipo español actual, en orden.

Pero empezar por la configuración como primer problema sería obviar todos los problemas que hemos tenido y que hemos tenido que resolver buscando en internet e informándonos para saber cómo implementar los diferentes algoritmos nosotros mismos, tanto backtracking como branch and bound. Esencialmente teníamos que entender cómo funcionaban para poder replicarlos y adaptarlos a nuestras necesidades y con el reto añadido de hacer alguno de ellos con greedy. Para ello tuvimos que realizar nuestra propia investigación y repasar varios de los apuntes tomados en clase llegando a crear programas enteros solo para probar diferentes ejemplos que encontramos y poder practicar antes de ponernos a intentar programar los retos planteados en la práctica. Ha sido gracias a esta investigación que hemos podido realizar el resto de la práctica con cierta facilidad ya que solo hemos necesitado aplicar los conocimientos adquiridos.

Pero es gracias a estos problemas que hemos ido afrontando que hemos aprendido más sobre los algoritmos que hemos implementado y que nos han permitido entender mejor la complejidad y utilidad de los mismos.

5 Conclusiones

Esta segunda práctica ha sido la más interesante hasta ahora. Es la primera vez que trabajamos con recursividad de esta manera y junto con la complicación que venía implícita hemos podido ver diferentes aproximaciones a un problema. No solo diferentes metodologías como lo son el "backtracking" y el "branch and bound" sinó también variaciones de estas como aplicar greedy como criterio o no.

El reto planteado, ordenar en función de las kcal y el porcentaje de grasas y ordenar en función de la cantidad de counters eran muy diferentes entre sí ofreciendo aplicaciones bastante dispares. Por un lado tenemos una aproximación matemática en la que hemos tenido que acotar los valores realizando una poda, aquellos que no cumplieran los prerrequisitos no podían ni optar a ser solución, y después de realizar esta poda había que optimizar el resultado. La segunda aproximación ha sido la de la ordenación de los equipos por la cantidad de counters que tenían con equipos españoles. Para ello hemos realizado los pasos explicados previamente en el algoritmo específico pero que nos ha ofrecido una vista diferente a un problema parecido. Sobretodo en cuanto a la creación de nuestra configuración, que ha supuesto uno de los retos más complicados de este segundo algoritmo.

Hemos aprendido mucho tanto a nivel académico como a nivel personal. El aprendizaje académico, después de mencionar las áreas de trabajo en las que más hemos aprendido y en las que más nos hemos centrado, queda bastante claro pero en lo personal también hemos podido crecer. Esta práctica nos ha permitido trabajar más como un equipo y ver lo que programar un programa así entre dos personas nos puede ofrecer. Es interesante ver como cambia de la programación individual a la grupal, en la que vemos cómo hemos compartido diferentes ideas y nos hemos ayudado mutuamente a superar problemas que hemos ido encontrando y sobretodo para plantear diferentes aproximaciones a problemas a las que muy probablemente no hubiéramos llegado sin nuestro compañero.

Más allá de esto, hemos mejorado en el uso de las herramientas que hemos utilizado para el desarrollo de este proyecto, tanto los IDE como aquellas herramientas que hemos utilizado para la programación compartida, a tiempo real. Esta mejora es interesante de ver ya que no solo nos servirá para futuras prácticas sino para poder aplicarlas el día de mañana en el mundo laboral.

Es por todo esto que esta práctica ha sido la más interesante hasta ahora. En la primera práctica tuvimos la introducción a la implementación de algoritmos recursivos con diferentes métodos de ordenación, en esta segunda práctica nos hemos adentrado más en agujero y hemos seguido viendo estos diferentes algoritmos implementados con recursividad.

6 Bibliografia

References

[1] Branch & Bound Algorithm – GeeksforGeeks
https://www.geeksforgeeks.org/branch-and-bound-algorithm/

[2] Branch & Bound Algorithm – towardsdatascience
https://towardsdatascience.com/the-branch-and-bound-algorithm-a7ae4d227a69

[3] Computer Science: Branch & Bound Algorithm – sciencedirect
https://www.sciencedirect.com/topics/computer-science/branch-and-bound-algorithm

[4] Pdf-BackTracking – Core
https://core.ac.uk/download/pdf/36721847.pdf

[5] Job Assignment Problem (B&B) – GeeksforGeeks
https://www.geeksforgeeks.org/job-assignment-problem-using-branch-and-bound/

[6] Journal Code – scitation
https://aip.scitation.org/doi/abs/10.1063/1.4965329?journalCode=apc

[7] Class – stanford
https://web.stanford.edu/class/ee392o/bb.pdf

[8] Branch & Bound Algorithm – Towards Data Science
https://towardsdatascience.com/the-branch-and-bound-algorithm-a7ae4d227a69

[9] Branch & Bound Algorithm – GrowingWthTheWeb
https://www2.seas.gwu.edu/bell/csci212/BranchandBound.pdf

[10] Greedy – WikiBooks
https://en.wikibooks.org/wiki/Algorithms/GreedyAlgorithms

[11] BackTracking – Universidad de Alcalà
ftp://www.cc.uah.es/pub/Alumnos/GInfInformatica/AlgoritmiayComplejidad/anteriores