①

LECCION 10: ALGORITMOS VORACES

EJEMPLO 2 - Problema de la Mochila

Tenemos n objetos cada uno con un peso p_i y un beneficio b_i. Además tenemos una mochila en la que podemos meter objetos. La mochila tiene capacidad de peso máximo M.

OBJETIVO → Llenar la mochila con objetos, de entre los n, que al sumar los beneficios de los escogidos este beneficio acumulado sea el mayor posible.

Supongamos que los objetos se pueden partir en trozos.

DATOS del PROBLEMA

n: numero de objetos disponibles
M: capacidad de la mochila
$p = (p_1, p_2, \dots, p_n)$ pesos de los objetos
$b = (b_1, b_2, \dots b_n)$ beneficio de los objetos

REPRESENTACIÓN de la SOLUCIÓN

Una solucion $S = (x_1, x_2, \dots x_n)$ con $0 \le x_i \le 1$ donde cada x_i representa el trozo escogido del objeto i.
Si $x_i = 1$ se coge todo el objeto
Si $x_i = 0$ no se coge el objeto

FORMULACION MATEMÁTICA

Maximizar $\sum_{i=1}^{n} x_i b_i$ sujeto a la restriccion $\sum_{i=1}^{n} x_i \cdot p_i \le M$ y $0 \le x_i \le 1$

EJEMPLO

$n=3$ $M=20$ $p=(18,15,10)$ $b=(25,24,15)$

SOLUCION 1 $S = \left(1, \frac{2}{15}, 0\right)$ Beneficio total $\sum_{i=1}^{3} x_i \cdot b_i = 1 \cdot 25 + \frac{2}{15} \cdot 24 + 0 \cdot 15$
$= 28.2$

Escoger por

$\sum_{i=1}^{3} x_i p_i = 1 \cdot 18 + \frac{2}{15} \cdot 15 = 20$

SOLUCION 2 $S = \left(0, \frac{2}{3}, 1\right)$

Beneficio total ⇒ $0 \cdot 25 + \frac{2}{3} \cdot 24 + 1 \cdot 15 = \boxed{31}$

$\sum_{i=1}^{3} x_i \cdot p_i = 0 \cdot 18 + \frac{2}{3} \cdot 15 + 1 \cdot 10 = 20.$

②

LECCIÓN 10: ALG. VORACES

Problema de la Mochila (continuación)

DISEÑO DE LA SOLUCIÓN

- Candidatos: todos los objetos de partida
- Función Selección.- Escoger el objeto mas prometedor
- Función Factible.- Como podemos añadir trozos será siempre cierta
- Añadir a la solución.- añadir el objeto si cabe o en otro caso la proporción del mismo que quede para complementar.
- Función objetivo - suma de los beneficios de cada candidato por la proporción seleccionada del mismo.

¿Cómo definirla? ⇓ IDEAS FELICES

- ALGORITMO.-

```
~~void~~ float MOCHILA(int M, vector<int> b, vector<int> p, vector<float> &X){
    //INICIALIZACION
    for (int i=0; i<n; i++)
        X[i]=0;

    int pesoAct=0; // peso acumulado hasta el momento
    float beneAct=0,0;
    while (pesoAct <M){
        int i= Seleccion(p,b); //seleccionamos el mejor objeto restante
        if (pesoAct +p[i] ≤ M)
        {
            X[i]=1
            pesoAct = pesoAct +p[i];   beneAct = beneAct +b[i];
        }
        else {
            X[i]=(M-pesoAct)/p[i]; //cojo la proporción de p[i]
                                   // para completar hasta M
            pesoAct=M    beneAct += X[i]* b[i];
        }
    }
}.
```

LECCION 10: ALGORITMOS VORACES.

PROBLEMA de la MOCHILA (continuación)

¿Cómo definir la función de selección?

1) El objeto con más beneficio $\cancel{bi}\ i^* = \arg\max_{i=1..n} b_i$

2) El objeto menos pesado $i^* = \arg\min_{i=1..n} p_i$

3) El objeto con mejor proporción b_i/p_i (beneficio por unidad de peso).
$i^* = \arg\max_{i=1..n} b_i/p_i$

EJEMPLO 1

$n=4$ $M=10$ $p=(10,3,3,4)$ $b=(10,9,9,9)$ $\frac{b}{p}=(1,3,3,2{,}25)$

CRITERIO 1 (por mayor beneficio) $S=(1,0,0,0)$ $B_T=10$

CRITERIO 2 (por menor peso) $S=(0,1,1,1)$ $B_T=\cancel{18}\ 27$

CRITERIO 3 (por mayor b_i/p_i) $S=(0,1,1,1)$ $B_T=\cancel{18}\ 27$

EJEMPLO 2

$n=4$ $M=10$ $p=(10,3,3,4)$ $b=(10,1,1,1)$ $\frac{b}{p}=(1,\frac{1}{3},\frac{1}{3},\frac{1}{4})$

CRITERIO 1 $\Longrightarrow$ $S=(1,0,0,0)$ $B_T=10$

CRITERIO 2 $\Longrightarrow$ $S=(0,1,1,1)$ $B_T=3$

CRITERIO 3 $\Longrightarrow$ $S=(1,0,0,0)$ $B_T=10$

El criterio 3 garantiza siempre la solución óptima — Ejercicio Demostrarlo.

(4)

LECCION 10 - ALG VORACES

Problema de la MOCHILA

EFICIENCIA - Si el vector esta ordenado por la razón b/p entonces seleccionar un objeto es O(1). Por lo tanto la eficiencia de la MOCHILA_VORAZ sería en el peor de los casos O(n).

EJERCICIO - En el caso de que los objetos no se puedan partir la técnica greedy puede llegar a dar soluciones no optimas. Buscar un ejemplo de esto.

DOS MOCHILAS

- Suponed que tenemos n objetos y dos mochilas M_1 y M_2 para llenar con los objetos.

```
void DosMochilas (int M1, int M2, const vector<int> &b, const vector<int> &p;
                  vector <float> &X){
  1) int n= X.size(); //n: de objetos
  2) for (int i=0; i<n; i++)  X[i]=0;
  3) int pesoAct=0; float b_Act=0.0; int en_m1=0; int en_m2=0
  4) while (pesoAct < M1+M2){
       int i= seleccion(b,p);
       if (p[i] <= (M1-en_m1)) {
          X[i]=1; pesoAct +=p[i]; en_m1 +=p[i];
          b_Act+= b[i];
       }
       else
       if (p[i] <= (M2-en_m2)){
          X[i]=1; pesoAct +=p[i]; en_m2 += p[i]
          b_Act += b[i];
       }
       else
       if (pesoAct +p[i] <= M1+M2) { //entra entero
          X[i]=1; pesoAct +=p[i]; en_m1 += (M1-enm1)/p[i]    // a = (M1-enm1)
          en_m2 +=(p[i]-a)
       }
       else{ //no entra entero
          X[i] = ((M1-en m1) + (M2-enm2)) / p[i];
          pesoAct = M1+M2;
          b_Act += X[i]* b[i];
       }
  }
```

4a

Dem – Seleccionar por mayor razón beneficio/peso

Supongamos que tenemos una solución óptima
$$x = (x_1, x_2, \ldots, x_n)$$
que incluye un objeto i, pero no incluye otro objeto j de menor proporción (parte de j o todo j).

Supongamos entonces que $x_i > x_j$ (la proporción incluida de i es mayor que la de j) y además se cumple que

$\frac{b_i}{p_i} < \frac{b_j}{p_j}$ (hemos incluido más de i a pesar que la razón es menor)

Supongamos ahora que construimos otra solución quitando parte de i y sustituyéndola por j. Sea esta cantidad r

$0 \leq r \leq x_i p_i \wedge r \leq (1 - x_j) p_j$ (r menor que lo que nos queda por poner de j)

$$b_{nuevo} = b_{ANTIGUO} - r\frac{b_i}{p_i} + r\frac{b_j}{p_j}$$
$$= b_{ANTIGUO} + r\left(\frac{b_j}{p_j} - \frac{b_i}{p_i}\right) > b_{ANTIGUO}$$ entonces x no era optimal $\rightarrow\leftarrow$