Programación Dinámica Determinística “Ejemplo de la Mochila”

[1] Gonzalez Lucy-Ingenieria Agricola, codigo273239
[2] Daira Garcia,Ingenieria Agricola, codigo 273351

Abstract. The mountaineer see how much of each food packages to bring in your backpack, in a way that maximizes its profit, without exceeding the capacity for its food, the solution is given by the deterministic dynamic programming method.

Keywords: Programación, Dinámica, Deterministica, mochila.

1 Introduction

Un montañista esta planeando una excursión muy especial. Evaluando la capacidad de su morral, la dificultad de la excursión, algunos implementos indispensables y sus fuerzas, cree que tiene en su morral una capacidad de C E Z kilos (u otra unidad de peso, o de manera mas precisa, de masa) disponibles para alimentos. De acuerdo con su experiencia, sus necesidades y sus gustos ha escogido ni tipos de alimentos A_1, A_2,….A_n, todos más o menos equilibrados. Estos alimentos vienen en paquetes indivisibles (por ejemplo en lata) y pi E Z indica el peso de cada paquete del alimento A_i. Teniendo en cuenta la composición de cada alimento, las calorías, las vitaminas, los minerales, el sabor, el contenido de agua, etc., el montañista asigno a cada paquete del alimento A_i un beneficio global b_i.

En este problema se supone que no es obligación llevar paquetes de cada uno de los alimentos. También su supone que no hay cotas inferiores ni superiores para el numero de paquetes de cada alimento.

Tal vez ningún montañista ha tratado de resolver este problema para organizar su morral, seguramente ni siquiera ha tratado de plantearlo. Lo que si es cierto es que hay muchos problemas, de gran tamaño y de mucha importancia, que tienen una estructura análoga. Hay libros y muchos artículos sobre este problema.

2 Planteamiento del Problema

Si x_j indica el número de paquetes del alimento A_j que el montañista debe llevar en su morral, entonces se debe maximizar el beneficio, bajo ciertas restricciones:

$$\max \sum_{j=1}^{n} b_j x_j$$

$$\sum_{j=1}^{n} p_j x_j \leq C$$

$$x_j \in Z,\ j = 1,\ldots n.$$

Este problema se puede resolver por la fuerza bruta construyendo todas las combinaciones, haciendo variar x_j entre 0 y $[C/p_j]$, verificando si cada combinación es factible.

$$\sum_{j=1}^{n} p_j x_j \leq C$$

La función objetivo (la función que hay que maximizar) es lineal, la restricción también el lineal, las variables deben ser enteras y se puede suponer que los coeficientes b_j y p_j también son enteros. Entonces este problema también se puede resolver por métodos de programación entera (programación lineal con variables enteras).

3 Método de la Programación Dinámica Deterministica

El nombre de programación dinámica se debe a que inicialmente el método se aplico a la optimización de algunos sistemas dinámicos, es decir, sistemas que evolucionan con el tiempo. Sin embargo, el tiempo no es indispensable, se requiere simplemente que los sistemas se pueden expresar por etapas o por fases.

Dicho de otra forma, la idea básica de la programación dinámica consiste en convertir un problema de n variables e una sucesión de problemas más simples, por ejemplo, de una variable, y para más sencillez, una variable discreta.

Desde el punto de vista recurrente un problema complejo se resuelve mediante el planteamiento de problemas más sencillos pero análogos al problema general, hasta encontrar la solución del problema general. Aunque es sencillo y aplicable a muchos problemas se aplica de manera específica a cada uno. Es decir no existe un algoritmo o (un programa de computador) único que se pueda aplicar a todos los problemas.

Por tanto para el caso general de la mochila o problemas análogos se presenta las siguientes funciones de recursión:

3.1 Recursion General

$$f_{n+1}(d) = 0$$

Para toda d

Para la etapa n la recursión es:

$$f_n(d) = \max(b_n * x_n)$$

Para las etapas n-1, n-2, ……1 la recursión es:

$$f_n(d) = \max(b_n * x_n) + f_{n+1}(d - P_n * x_n)$$

Donde:

- Pn=Peso del producto tipo n
- d= c, c-1, c-2,…….0
- Xn= 0, 1, 2,..,de tal forma que se cumpla: Pn(xn) <= d

4 Ejemplo

Se supone que se va a llenar una mochila de 10 lb, con los siguientes datos:

Tabla 1. Cantidad de productos, el peso de cada producto y su respectivo beneficio.

PRODUCTO	PESO (lb)	BENEFICIO
1	4	11
2	3	7
3	5	12

Para maximizar el beneficio total, con que se debe llenar la mochila.

Tenemos la function

$$f_n(d) = \max(b_n * x_n) + f_{n+1}(d - P_n * x_n)$$

4.1Grafo

Encontrar la solución optima equivale a determinar el camino mas largo desde el nodo (10,1) a alguno de la etapa 4. Para n=< 3 el nodo (d , n) representa la situación en la cual d libras de espacio se podrían asignar a productos tipo n, n+1, ….3 El nodo (d,4) representa d libras de espacio sin utilizar. Cada arco desde un nodo de la etapa n a un nodo de la etapa n+1 representa una decisión de cuantos productos del tipo t se colocan en la mochila. Por ejemplo el arco (10,1) a (6,2) representa colocar un producto tipo 1 en la mochila. Asi quedan (10-4=6 lb) para productos tipo 2 y 3. Este arca tiene una longitud de 11, lo que representa el beneficio conseguido al llevar un producto del tipo 1 en la mochila.

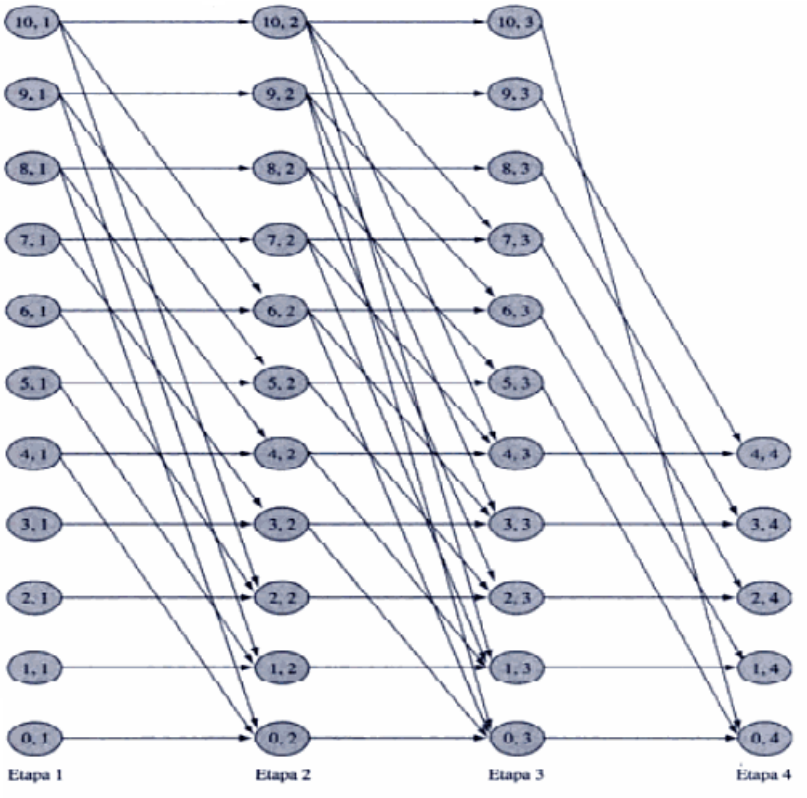

Fig. 1. Grafo de la mochila, Metodo Dinamico Deterministico.

Se define $f_n(d)$ como el máximo beneficio que se puede conseguir con una mochila de de libras que se llena con productos del tipo n, n+1,…3.

4.2 Etapa III

$$f_n(d) = \max(b_n * x_n) + f_{n+1}(d - P_n * x_n)$$

Beneficio = b = 12
Peso = 5lb

$$f_3(10) = \max(12*2) + f_4(10-5*2) = 24+0 = 24 \otimes$$
$$f_3(9) = \max(12*1) = 12$$
$$f_3(8) = \max(12*1) = 12$$
$$f_3(7) = \max(12*1) = 12$$
$$f_3(6) = \max(12*1) = 12$$
$$f_3(5) = \max(12*1) = 12$$
$$f_3(4) = \max(12*0) = 0$$
$$f_3(3) = \max(12*0) = 0$$
$$f_3(2) = \max(12*0) = 0$$
$$f_3(1) = \max(12*0) = 0$$
$$f_3(0) = \max(12*0) = 0$$

4.3 Etapa II

$$fn(d) = \max(b_i * x_i) + f_{n+1}(d - p * x_i)$$

Beneficio = 7
Peso = 3 lb

$$f_2(10) = \begin{Bmatrix} \max(7*0) + f_3(10-3*0) = \max(0) + f_3(10) = 0+24 = 24 \otimes \\ \max(7*1) + f_3(10-3*1) = \max(7) + f_3(7) = 7+12 = 19 \\ \max(7*2) + f_3(10-3*2) = \max(14) + f_3(4) = 14+0 = 14 \\ \max(7*3) + f_3(10-3*3) = \max(21) + f_3(1) = 21+0 = 21 \end{Bmatrix}$$

Donde:

$f_2(10) = 24$

$$f_2(9) = \begin{Bmatrix} \max(7*0) + f_3(9-3*0) = \max(0) + f_3(9) = 0+12 = 12 \\ \max(7*1) + f_3(9-3*1) = \max(7) + f_3(6) = 7+12 = 19 \\ \max(7*2) + f_3(9-3*2) = \max(14) + f_3(3) = 14+0 = 14 \\ \max(7*3) + f_3(9-3*3) = \max(21) + f_3(0) = 21+0 = 21 \otimes \end{Bmatrix}$$

Donde:

$f_2(9) = 21$

$$f_2(8) = \begin{Bmatrix} \max(7*0) + f_3(8-3*0) = \max(0) + f_3(8) = 0+12 = 12 \\ \max(7*1) + f_3(8-3*1) = \max(7) + f_3(5) = 7+12 = 19 \otimes \\ \max(7*2) + f_3(8-3*2) = \max(14) + f_3(2) = 14+0 = 14 \end{Bmatrix}$$

Donde:

$f_2(8) = 19$

$$f_2(7) = \begin{Bmatrix} \max(7*0) + f_3(7-3*0) = \max(0) + f_3(7) = 0+12 = 12 \\ \max(7*1) + f_3(7-3*1) = \max(7) + f_3(4) = 7+0 = 7 \\ \max(7*2) + f_3(7-3*2) = \max(14) + f_3(1) = 14+0 = 14 \otimes \end{Bmatrix}$$

Donde:

$f_2(7) = 14$

$$f_2(6) = \begin{Bmatrix} \max(7*0) + f_3(6-3*0) = \max(0) + f_3(6) = 0+12 = 12 \\ \max(7*1) + f_3(6-3*1) = \max(7) + f_3(3) = 7+0 = 7 \\ \max(7*2) + f_3(6-3*2) = \max(14) + f_3(0) = 14+0 = 14 \otimes \end{Bmatrix}$$

Donde:

$f_2(6) = 14$

$$f_2(5) = \begin{Bmatrix} \max(7*0) + f_3(5-3*0) = \max(0) + f_3(5) = 0+12 = 12 \otimes \\ \max(7*1) + f_3(5-3*1) = \max(7) + f_3(2) = 7+0 = 7 \end{Bmatrix}$$

Donde:

$f_2(5) = 12$

$$f_2(4) = \begin{Bmatrix} \max(7*0) + f_3(4-3*0) = \max(0) + f_3(4) = 0+0 = 0 \\ \max(7*1) + f_3(4-3*1) = \max(7) + f_3(1) = 7+0 = 7 \otimes \end{Bmatrix}$$

Donde:

$f_2(4) = 7$

$$f_2(3) = \begin{Bmatrix} \max(7*0) + f_3(3-3*0) = \max(0) + f_3(3) = 0+0 = 0 \\ \max(7*1) + f_3(3-3*1) = \max(7) + f_3(1) = 7+0 = 7 \otimes \end{Bmatrix}$$

Donde:

$f_2(3) = 7$

$$f_2(2) = \begin{Bmatrix} \max(7*0) + f_3(2-3*0) = \max(0) + f_3(2) = 0+0 = 0 \end{Bmatrix}$$

Donde:

$f_2(2) = 0$

$$f_2(1) = \begin{Bmatrix} \max(7*0) + f_3(1-3*0) = \max(0) + f_3(1) = 0+0 = 0 \end{Bmatrix}$$

Donde:

$f_2(1) = 0$

$$f_2(0) = \max(7*0) + f_3(0-3*0) = \max(0) + f_3(1) = 0+0 = 0$$

Donde:

$$f_2(1) = 0$$

4.4 Etapa I

Beneficio = 11
Peso = 4

$$f_1(10) = \begin{Bmatrix} max(11*0) + f_2(10-4*0) = max(0) + f_2(10) = 0+24 = 24 \\ max(11*1) + f_2(10-4*1) = max(11) + f_2(6) = 11+14 = 25 \\ max(11*2) + f_2(10-4*2) = max(22) + f_2(2) = 22+0 = 22 \end{Bmatrix}$$

Donde:

$$f_1(10) = 25$$

Tenemos $f_1(10) = 25$ y $X_1(10)=1$ Por lo tanto, podríamos incluir un producto tipo 1 en la mochila. Entonces quedan 10-4=6 libras para los productos de los tipos 2 y 3 de modo que incluimos $x_2(6)=2$ productos tipo 2. Por último hay 6-2(3) =0 libras para los productos tipo 3, por lo que llevaríamos $x_3(0)=0$ productos del tipo 3. En resumen, el beneficio máximo que podemos lograr con una mochila de 10 libras es $f_3(10)=25$. Para obtener un beneficio de 25 debemos llevar un producto del tipo 1 y dos de l tipo 2. La solución optima de este tipo de problemas no siempre usa todo el peso disponible

Referencias

1. Mora Escobar Héctor Manuel. "Optimización no lineal y dinámica". Universidad Nacional de Colombia. Bogota.1996 Pág. 313 - 314, 289.

2. Winston Wayne L. "Investigación de operaciones aplicaciones y algoritmos". Editorial Thomson. Cuarta edición. Pág. 979- 982