Tema 1

Introducción a la Programación Dinámica. El Problema de la Mochila

"La programación dinámica no es un algoritmo. Es más bien un principio general aplicable a diversos problemas de optimización que verifican una cierta propiedad denominada descomponibilidad".

1.1. El problema de la Mochila

Un excursionista debe decidir, entre n objetos, cuales de ellos va a llevarse en su mochila. Cada objeto supone para el excursionista un beneficio c_j y ocupa una capacidad de a_j. La mochila tiene una capacidad máxima b. La formulación del problema sería la siguiente:

$$(KP) \qquad \text{Maximizar } \sum_{j=1}^{n} c_j x_j$$

$$\text{s.a}$$

$$\sum_{j=1}^{n} a_j x_j \leq b$$

$$x_j \in \{0,1\} \qquad j = 1, ..., n$$

donde $a_j \in \mathbb{N} \quad \forall j = 1, ..., n$; $b \in \mathbb{N}$ y $c_j \in \mathbb{R} \quad \forall j = 1, ..., n$.

La idea principal de la programación dinámica consiste en intentar reducir el problema (KP) de n variables en una secuencia de problemas en una sola variable. Para ello es necesario llevar a cabo dos operaciones:

Encajar el problema dado en una familia de problemas de la misma naturaleza.

En el caso del problema (KP) se considera la siguiente familia de n(b+1) problemas:

$$[KP_i(E)] \qquad \text{Maximizar } \sum_{j=i}^{n} c_j x_j$$

$$\text{s.a}$$

$$\sum_{j=i}^{n} a_j x_j + E \leq b$$

$$x_j \in \{0,1\} \qquad j = i, ..., n$$

con i variando entre 1 y n, y E entero variando entre 0 y b.

La cantidad E se denomina **variable de estado** (o **vector de estado** cuando la catidad sea vectorial). El conjunto de los posibles valores de E se denomina **espacio de estados** y se representa por ξ. En este caso $\xi = \{0, 1, ..., b\}$.

Se puede observar que el problema original [KP] es un miembro de la familia (es de hecho $[KP_1(0)]$). Se denotará por $F_i^*(E)$ al valor óptimo de $[KP_i(E)]$. Cuando $E < 0$ o $E > b$ el problema es infactible y se acuerda que $F_i^*(E) = -\infty$. El valor óptimo del problema original es $F^* = F_1^*(0)$.

Buscar una relación recurrente que conecte los valores óptimos de los diferentes problemas de la familia

Para ello vamos a analizar la familia de problemas $[KP_i(E)]$. Supongamos que en la mochila, al decidir sobre los objetos j=1,...,i-1, se ha ocupado una capacidad E. Por tanto queda una capacidad b-E para decidir sobre los objetos j=i,...,n. El excursionista desea saber cuál es la mejor decisión que puede tomar.

Etapa n. En $[KP_n(E)]$ el excursionista debe decidir si introduce o no el objeto n. Si ha ocupado ya una capacidad $E > b - a_n$, no puede hacerlo, y por lo tanto $F_n^*(E) = 0$. En caso de que $E \leq b - a_n$, aún existe la posibilidad de introducir el objeto. Lo hará si su beneficio es positivo ($c_n \geq 0$) con lo cual $F_n^*(E) = c_n$, o lo desechará en caso contrario ($c_n < 0$), con lo cual $F_n^*(E) = 0$

Etapa n-1. En $[KP_{n-1}(E)]$ debe decidir si se introducen los objetos $n-1$ y n, pero con lo indicado en el apartado anterior se puede simplificar el problema a decidir solamente sobre el objeto $n-1$. Si se introduce dicho objeto, en la etapa n el problema a resolver será $[KP_n(E + a_{n-1})]$, por tanto se obtendría un beneficio de $c_{n-1} + F_n^*(E + a_{n-1})$, y si no lo introducimos, en la etapa n el problema a resolver será $[KP_n(E)]$ y obtendremos un beneficio de $F_n^*(E)$. Por tanto la decisión se basa simplemente en la siguiente comparación.

$$F_{n-1}^*(E) = Max\{c_{n-1} + F_n^*(E + a_{n-1}), F_n^*(E)\}$$

Etapa i. Con un razonamiento análogo al de la etapa $n-1$, en una etapa intermedia i, $1 \leq i \leq n$, tendremos la siguiente relación:

$$F_i^*(E) = Max\{c_i + F_{i+1}^*(E + a_i), F_{i+1}^*(E)\}$$

Algoritmo para la resolución del problema de la mochila por programación dinámica

Etapa n

Para todo E $(0 \leq E \leq b)$ calcular $F_n^*(E)$ - $F_n^*(E) = 0$ si $E > b - a_n$

- $F_n^*(E) = c_n$ si $E \leq b - a_n$ y $c_n \geq 0$

- $F_n^*(E) = 0$ si $E \leq b - a_n$ y $c_n < 0$

Etapa i

Para cada $i = n-1, ..., 2$ calcular:

$$F_i^*(E) = Max\{c_i + F_{i+1}^*(E + a_i), F_{i+1}^*(E)\}$$

para todo $0 \leq E \leq b$.

Etapa 1

Calcular $F^* = Max\{F_2^*(0), c_1 + F_2^*(a_1)\}$

Comentario. La etapa 1 se puede extender calculando, para cada $0 \leq E \leq b$

$$F_1^*(E) = Max\{F_2^*(E), c_1 + F_2^*(E + a_1)\}$$

De este modo, $F^* = F_1^*(0)$ y además, para cada $0 \leq E \leq b$, $F_1^*(E)$ representaría la utilidad máxima posible en una mochila de carga máxima $b - E$. ■

Obtención de la solución óptima

Mediante el algoritmo anterior se obtiene el valor óptimo de (KP) pero no la solución óptima $\mathbf{x}^*$ explícitamente. Para ello se procede del siguiente modo.
Para $i = 1, ..., n-1$

$$x_i = \begin{cases} 0 & \text{si } F_i^*(E) = F_{i+1}^*(E) \\ 1 & \text{si } F_i^*(E) = c_i + F_{i+1}^*(E + a_i) \end{cases}$$

Siendo inicialmente E=0 y actualizando en cada paso el valor de E a $E + a_i x_i^*$. Finalmente,

$$x_n = \begin{cases} 0 & \text{si } F_n^*(E) = 0 \\ 1 & \text{si } F_n^*(E) = c_n \end{cases}$$

1.2. Ejercicios

1. Un navegante solitario dispone en su barco de 5 metros cúbicos para almacenar cuatro objetos. El objeto A tiene un volumen de 2 m^3 y reporta al navegante 3 unidades de beneficio (ub). Los objetos B,C, y D ocupan respectivamente 4,3 y 2 m^3 y el beneficio respectivo es de 5,1 y 1 ub.

   a) Determinar mediante un algoritmo de programación dinámica cuáles son los objetos que debe llevar el navegante.

   b) ¿Qué ocurriría si la capacidad del barco fuese respectivamente de 4, 3, 2 ó 1 metros cúbicos?

2. Adaptación del algoritmo de programación dinámica para la resolución del proble-

ma de la mochila 0-1 unidimensional al caso en que la restricción sea de igualdad:

$$\begin{aligned}
&\text{Max } 3x_1 + 5x_2 + x_3 + x_4 - x_5 \\
&\quad \text{s.a} \\
&\qquad 2x_1 + 4x_2 + 3x_3 + 2x_4 + x_5 = 5 \\
&\qquad x_i \in \{0,1\} \quad i = 1,2,3,4,5
\end{aligned}$$

3. Adaptación del algoritmo de programación dinámica para la resolución del problema de la mochila unidimensional al caso en que las variables son enteras y acotadas de la forma $0 \leq x_j \leq M_j$:

$$\begin{aligned}
&\text{Max } 5x_1 + 3x_2 + x_3 + x_4 \\
&\quad \text{s.a} \\
&\qquad 4x_1 + 2x_2 + 3x_3 + 2x_4 \leq 5 \\
&\qquad x_i \in \{0,1,2\} \quad i = 1,2,3,4
\end{aligned}$$

4. Adaptación del algoritmo de programación dinámica para la resolución del problema de la mochila unidimensional al caso en que las variables son enteras y no acotadas:

$$\begin{aligned}
&\text{Maximizar} \quad 5x_1 + 4x_2 + 2x_3 \\
&\quad \text{s.a} \\
&\qquad 4x_1 + 3x_2 + 2x_3 \leq 8 \\
&\qquad x_j \in \mathbb{Z}_+ \qquad j = 1,2,3
\end{aligned}$$

5. Adaptación del algoritmo de programación dinámica para la resolución del problema de la mochila 0-1 bidimensional:

$$\text{Max } 4x_1 + x_2 + 5x_3$$

$$\text{s.a}$$

$$2x_1 + x_2 + 3x_3 \leq 4$$

$$x_1 + 2x_2 + 2x_3 \leq 3$$

$$x_i \in \{0, 1\} \quad i = 1, 2, 3$$

6. Un camión puede transportar un total de 10 toneladas de productos. Hay tres clases de productos para transportar, cuyo peso y valor se muestran en la siguiente tabla. Suponiendo que por lo menos se debe transportar un artículo de cada clase, determinar el cargamento que maximiza el valor total.

Clase	Valor (miles de euros)	Peso (tn)
A	2	1
B	5	2
C	6	2