PROBLEMA DE LA MOCHILA ENTERA

ALBERTO VERDEJO

UNIVERSIDAD
COMPLUTENSE
MADRID

Problema de la mochila (versión entera)

- Hay n objetos, cada uno con un peso (entero) $p_i > 0$ y un valor (real) $v_i > 0$.
- La mochila soporta un peso total (entero) máximo $M > 0$.
- Y el problema consiste en maximizar

$$\sum_{i=1}^{n} x_i v_i$$

con la restricción

$$\sum_{i=1}^{n} x_i p_i \leq M,$$

donde $x_i \in \{0, 1\}$ indica si hemos cogido (1) o no (0) el objeto i.

Problema de la mochila (versión entera)

- En las soluciones no importa el orden en el que los objetos son introducidos en la mochila.
- Las soluciones dependen de los objetos que tengamos disponibles para introducir en la mochila y del peso que soporte esta.
- Definimos la siguiente función:

$$mochila(i,j) = \text{máximo valor que podemos poner en una mochila de peso máximo } j \text{ considerando los objetos del 1 al } i$$

- Se cumple el principio de optimalidad de Bellman.

Definición recursiva

- Casos recursivos:

$$mochila(i,j) = \begin{cases} mochila(i-1,j) & \text{si } p_i > j \\ \text{máx}(mochila(i-1,j), mochila(i-1,j-p_i)+v_i) & \text{si } p_i \leq j \end{cases}$$

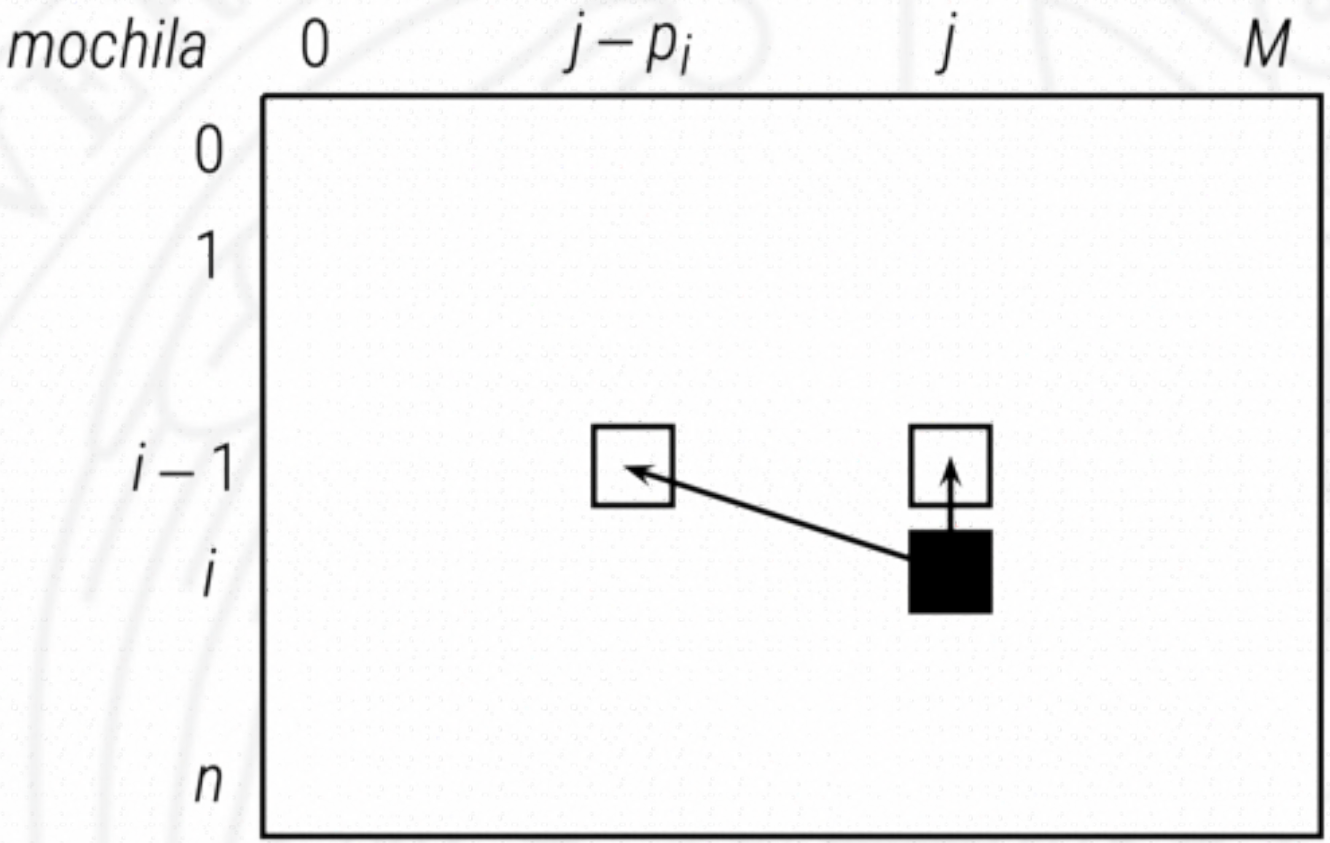
con $1 \leq i \leq n$ y $1 \leq j \leq M$

- Casos básicos:

$$\begin{aligned} mochila(0,j) &= 0 \qquad 0 \leq j \leq M \\ mochila(i,0) &= 0 \qquad 0 \leq i \leq n \end{aligned}$$

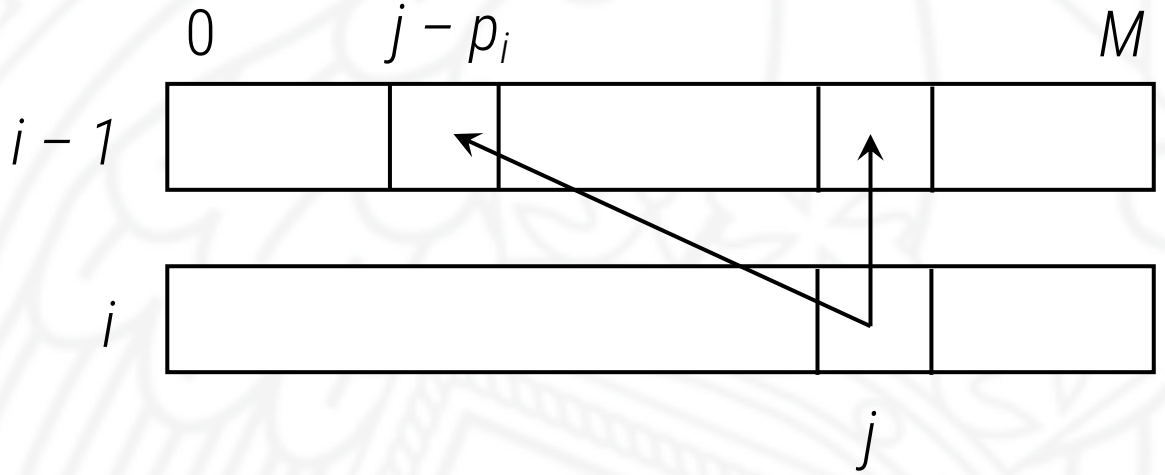
- Llamada inicial: $mochila(n, M)$

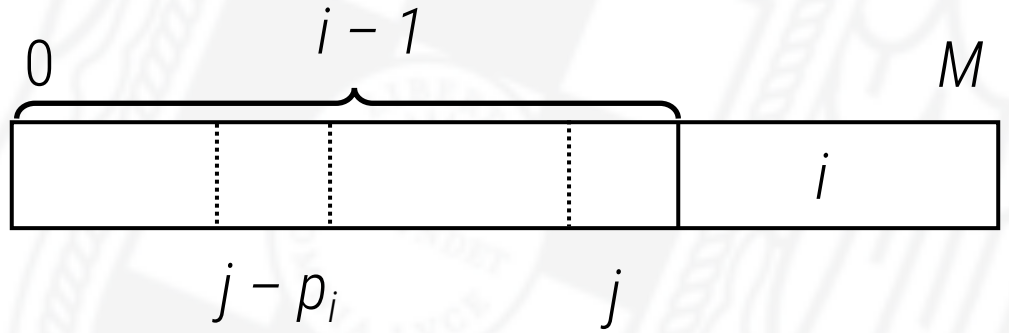
Tabla



Mejorar espacio adicional

- ¿Podemos reducir la memoria necesaria?

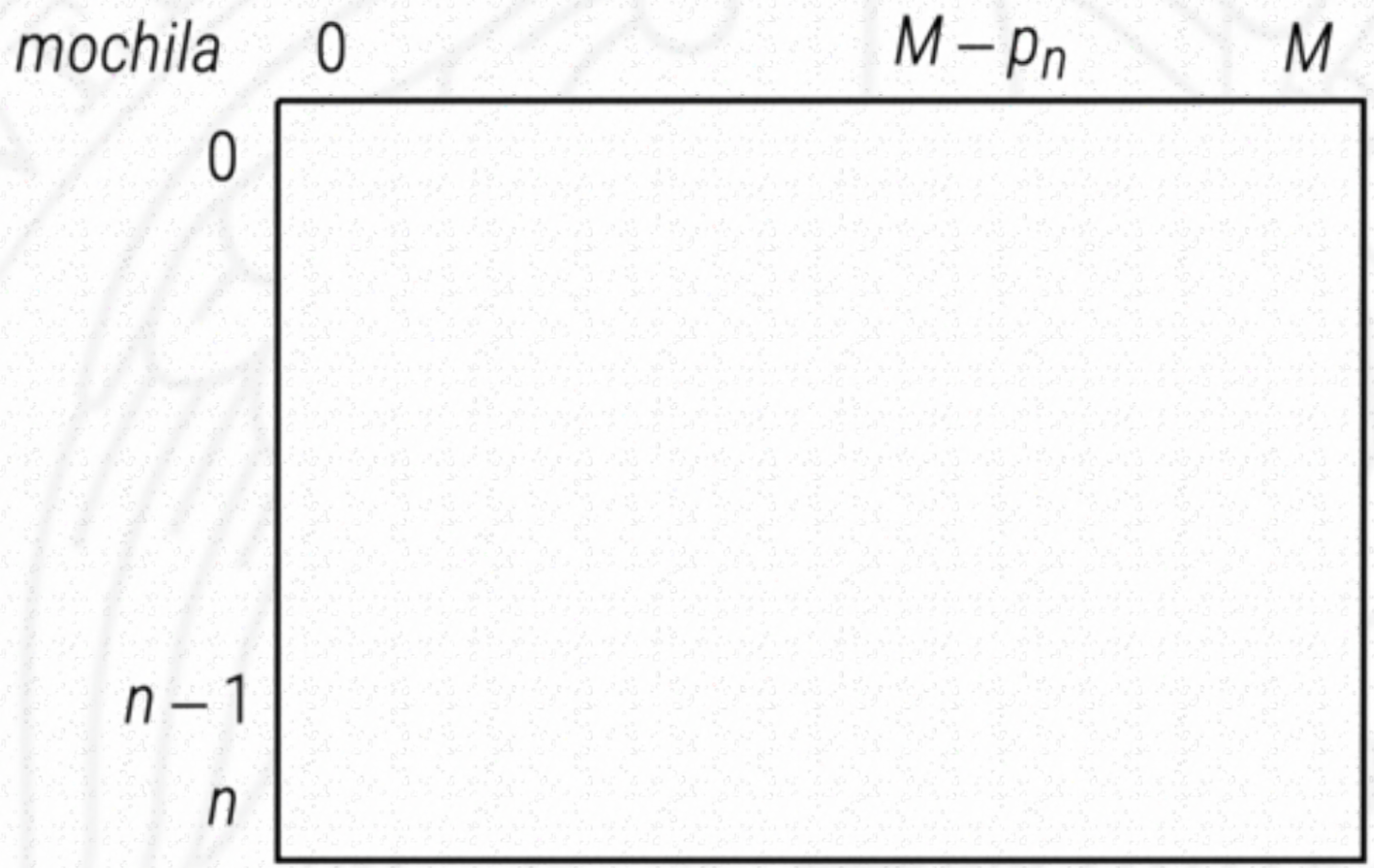




Reconstrucción de la solución óptima

$$mochila(i,j) = \text{máx}(\underbrace{mochila(i-1,j)}_{\text{no cogemos el objeto } i}, \underbrace{mochila(i-1,j-p_i)+v_i}_{\text{sí cogemos el objeto } i})$$

Reconstrucción de la solución óptima



Implementación

```
struct Objeto { int peso; double valor; };

double mochila_rec(vector<Objeto> const& obj, int i, int j,
                   Matriz<double> & mochila) {
  if (mochila[i][j] != -1) // subproblema ya resuelto
     return mochila[i][j];
  if (i == 0 || j == 0) mochila[i][j] = 0;
  else if (obj[i-1].peso > j)
     mochila[i][j] = mochila_rec(obj, i-1, j, mochila);
  else
     mochila[i][j] = max(mochila_rec(obj, i-1, j, mochila),
                         mochila_rec(obj, i-1, j - obj[i-1].peso, mochila)
                         + obj[i-1].valor);
  return mochila[i][j];
}
```

Implementación

```
double mochila(vector<Objeto> const& objetos, int M, vector<bool> & sol) {
   int n = objetos.size();
   Matriz<double> mochila(n+1, M+1, -1);
   double valor = mochila_rec(objetos, n, M, mochila);
   // cálculo de los objetos
   int i = n, j = M;
   sol = vector<bool>(n, false);
   while (i > 0 && j > 0) {
      if (mochila[i][j] != mochila[i-1][j]) {
         sol[i-1] = true; j = j - objetos[i-1].peso;
      }
      --i;
   }
   return valor;
}
```