PROBLEMA DE LA MOCHILA REAL

UNIVERSIDAD COMPLUTENSE MADRID

ALBERTO VERDEJO

Problema de la mochila real

- Cuando Alí-Babá consiguió entrar en la Cueva de los Cuarenta Ladrones, encontró allí objetos muy valiosos, de los que conocía muy bien el peso y valor.

- Solo podía llevar consigo aquellas riquezas que cupieran en su pequeña mochila, que soportaba un peso máximo conocido.
- Suponiendo que los objetos fueran fraccionables, ¿qué objetos debería elegir Alí-Babá para maximizar el valor total de su mochila?

Problema de la mochila real

- Hay n objetos, cada uno con un peso $p_i > 0$ y un valor $v_i > 0$.
- La mochila soporta un peso total 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 es la *fracción* de objeto i tomada, $0 \leq x_i \leq 1$.

Ejemplo

$M = 100, n = 5$

	1	2	3	4	5
p_i	10	20	30	40	50
v_i	20	30	66	40	60

seleccionar	x_i					valor
máx. v_i	0	0	1	0,5	1	146

Ejemplo

$M = 100, n = 5$

	1	2	3	4	5
p_i	10	20	30	40	50
v_i	20	30	66	40	60

seleccionar	x_i					valor
máx. v_i	0	0	1	0,5	1	146
mín. p_i	1	1	1	1	0	156

Ejemplo

$M = 100, n = 5$

	1	2	3	4	5
p_i	10	20	30	40	50
v_i	20	30	66	40	60
$\frac{v_i}{p_i}$	2	1,5	2,2	1	1,2

seleccionar	x_i					valor
máx. v_i	0	0	1	0,5	1	146
mín. p_i	1	1	1	1	0	156

Ejemplo

$M = 100, n = 5$

	1	2	3	4	5
p_i	10	20	30	40	50
v_i	20	30	66	40	60
$\frac{v_i}{p_i}$	2	1,5	2,2	1	1,2

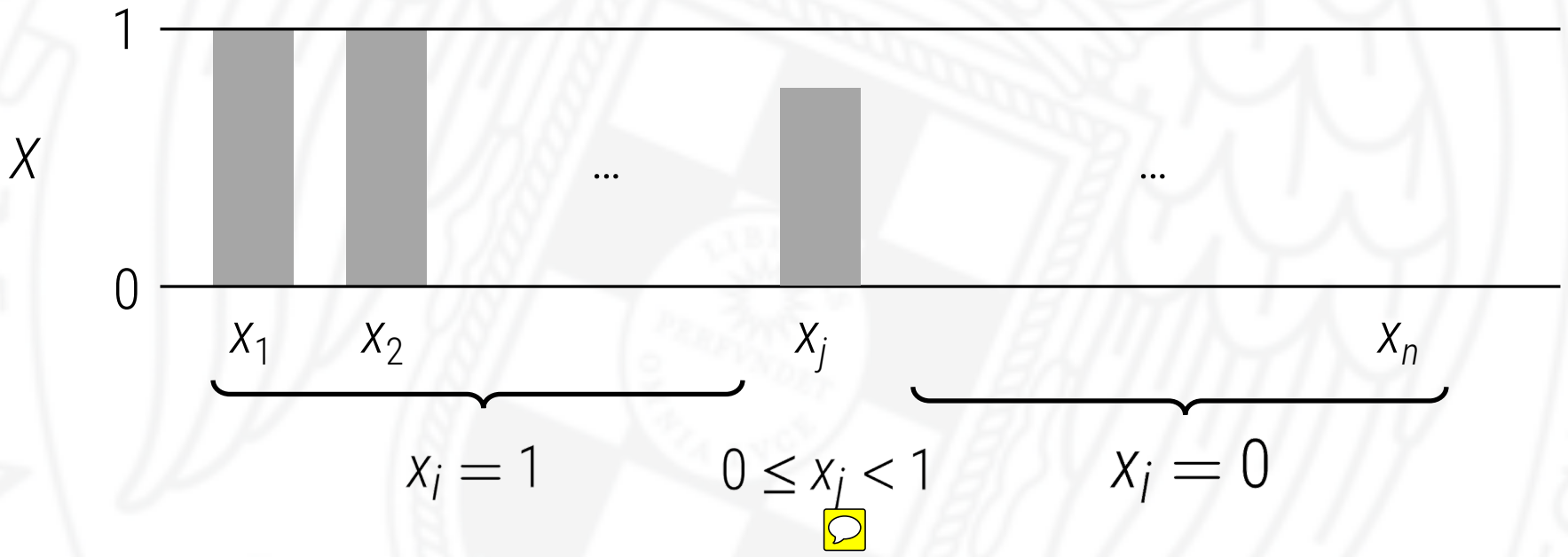
seleccionar	x_i					valor
máx. v_i	0	0	1	0,5	1	146
mín. p_i	1	1	1	1	0	156
máx. $\frac{v_i}{p_i}$	1	1	1	0	0,8	164

Demostración de optimalidad

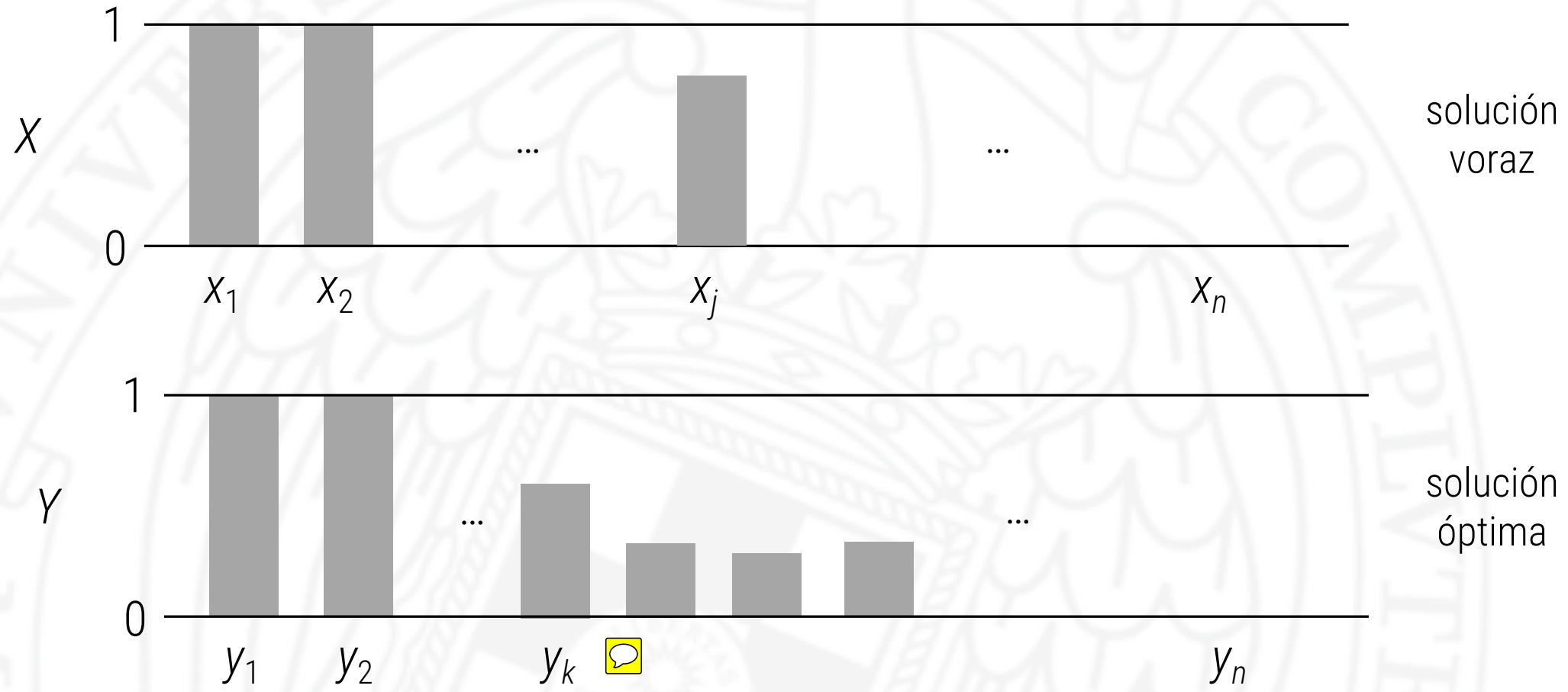
- Objetos ordenados de mayor a menor valor por unidad de peso:

$$\frac{v_1}{p_1} \geq \frac{v_2}{p_2} \geq \ldots \geq \frac{v_n}{p_n}$$

- Solución X construida por el algoritmo voraz:

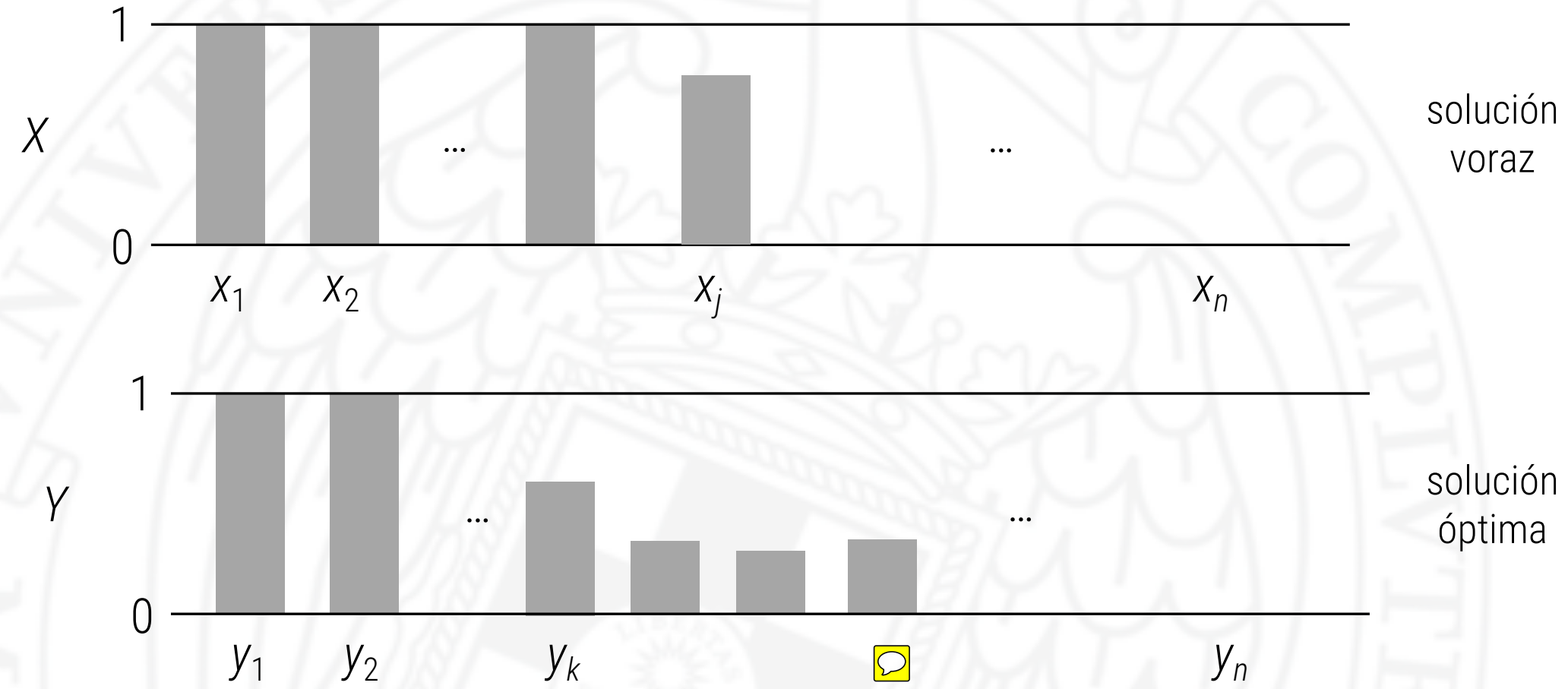


Demostración de optimalidad



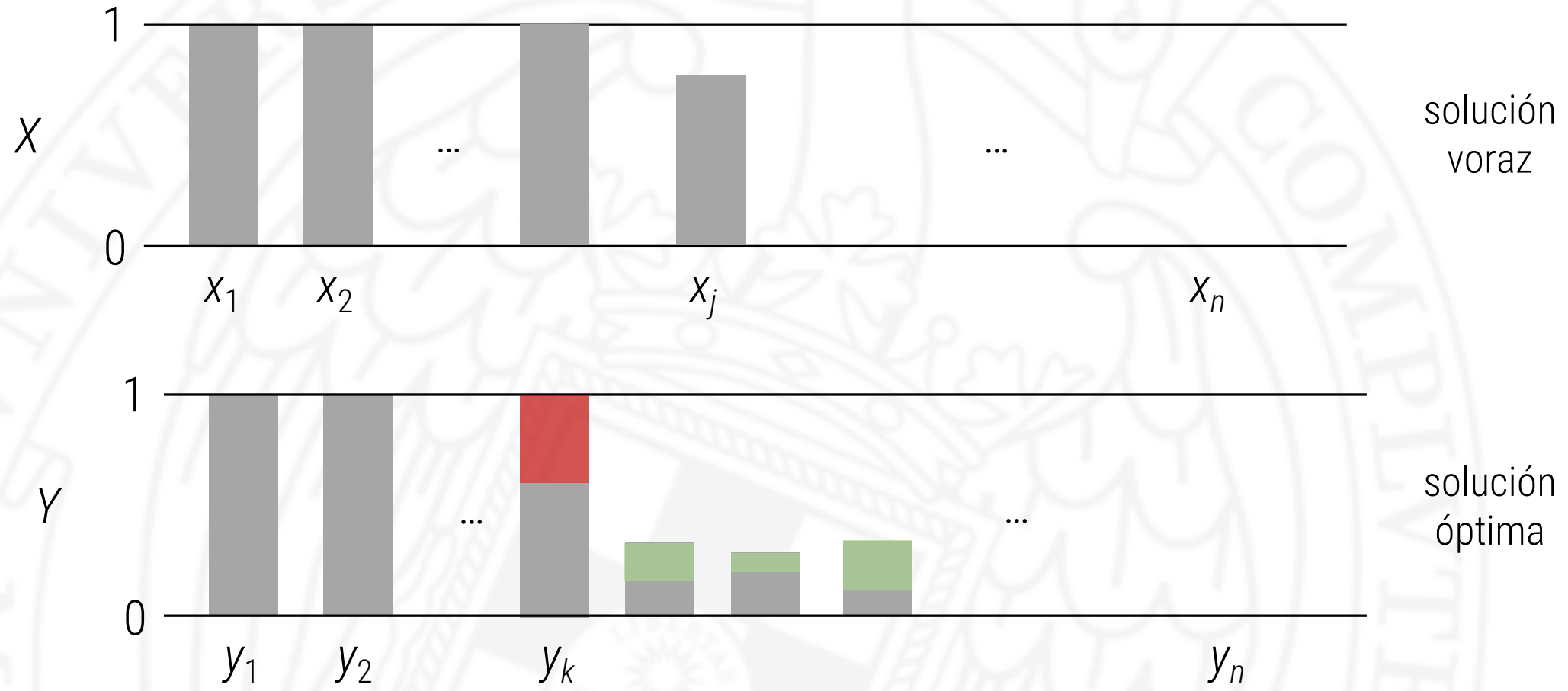
- Podemos asumir que $\sum_{i=1}^{n} y_i p_i = M$.

Demostración de optimalidad

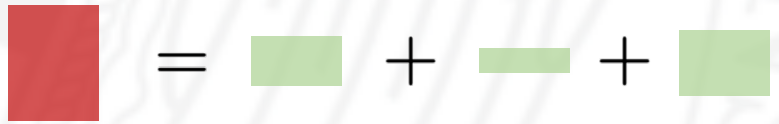


- Se debe cumplir que $k \leq j$ y por tanto $y_k < x_k$

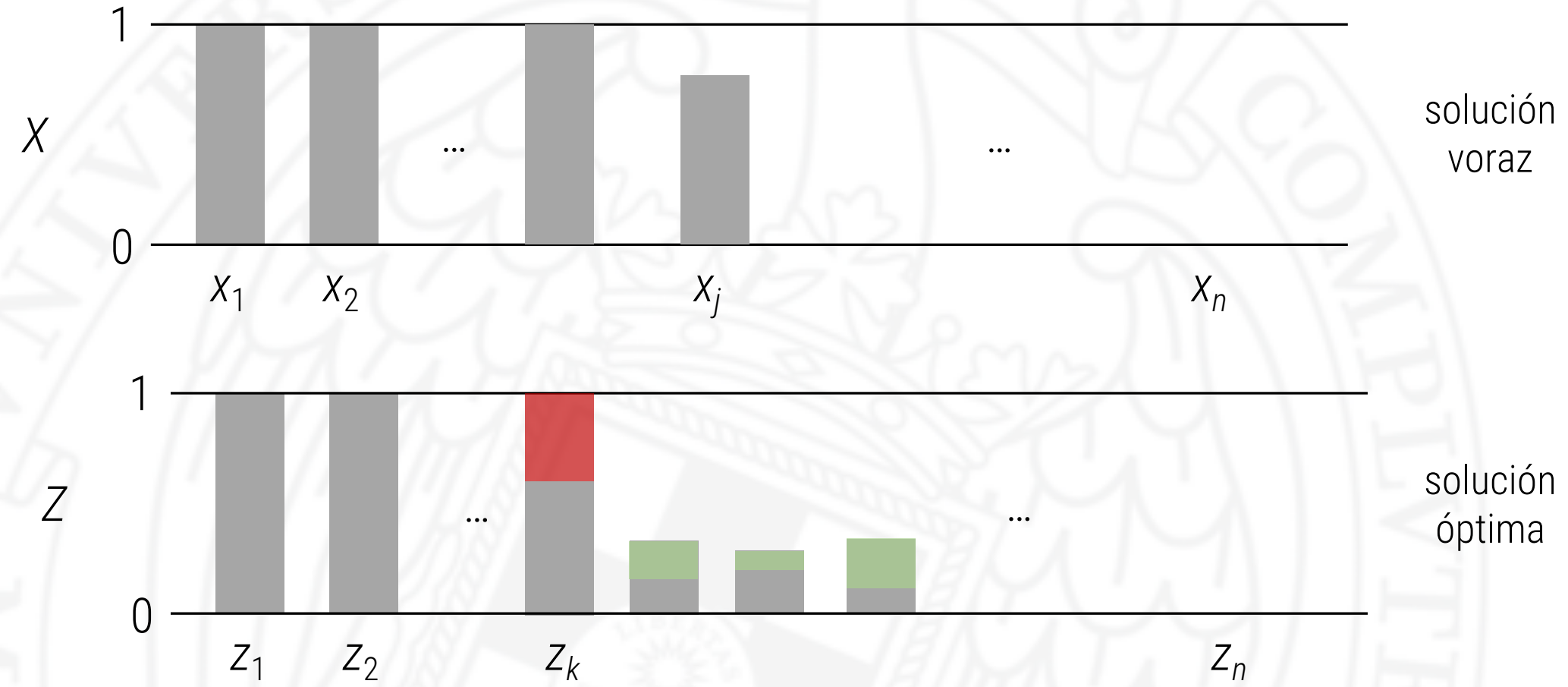
Demostración de optimalidad



- Aumentar y_k para hacerlo igual a x_k



Demostración de optimalidad



- La nueva solución Z cumple $\sum_{i=k+1}^{n} p_i(y_i - z_i) = p_k(z_k - y_k)$

Demostración de optimalidad

- Z sigue siendo óptima:

$$
\begin{aligned}
\sum_{i=1}^{n} v_i z_i &= \sum_{i=1}^{n} v_i y_i + v_k(z_k - y_k) - \sum_{i=k+1}^{n} v_i(y_i - z_i) \\
&= \sum_{i=1}^{n} v_i y_i + \frac{v_k}{p_k} p_k(z_k - y_k) - \sum_{i=k+1}^{n} \frac{v_i}{p_i} p_i(y_i - z_i) \\
&\geq \sum_{i=1}^{n} v_i y_i + \Bigg(p_k(z_k - y_k) - \sum_{i=k+1}^{n} p_i(y_i - z_i) \Bigg) \frac{v_k}{p_k} \\
&= \sum_{i=1}^{n} v_i y_i
\end{aligned}
$$

($\underbrace{\quad}_{0}$)

Implementación

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

struct Densidad {
    double densidad; // valor / peso
    int id;  // identificador del objeto
};

bool operator>(Densidad const& a, Densidad const& b) {
    return a.densidad > b.densidad;
}
```

Implementación

```
double mochila(vector<Objeto> const& objetos, double M,
               vector<double> & sol) {  // O(N log N)
   int N = objetos.size();
   vector<Densidad> D(N);
   for (int i = 0; i < N; ++i) {
      D[i].densidad = objetos[i].valor / objetos[i].peso;
      D[i].id = i;  // para saber a qué objeto corresponde
   }
   sort(D.begin(), D.end(), greater<Densidad>());
```

Implementación

```
    double peso = 0, valor = 0;
    int i;
    for (i = 0; i < N && peso + objetos[D[i].id].peso <= M; ++i) {
        sol[D[i].id] = 1;  // el objeto D[i].id cabe completo
        peso += objetos[D[i].id].peso;
        valor += objetos[D[i].id].valor;
    }
    if (i < N) { // partir el objeto D[i].id
        sol[D[i].id] = (M - peso) / objetos[D[i].id].peso;
        valor += sol[D[i].id] * objetos[D[i].id].valor;
    }
    return valor;
}
```

$$peso + x_i \cdot p_i = M \Rightarrow x_i = \frac{M - peso}{p_i}$$

Versión 0/1

- Los objetos no se pueden partir.
- La estrategia voraz anterior no siempre calcula una solución óptima.
- Contraejemplo: