O problema da mochila (knapsack) tem importantes aplicações como o carregamento ótimo de containers. GJ [MP9]

Mochila (m, n, v, w): Dados um conjunto $U = \{1, 2, \cdots, n\}$ com respectivos valores v_i e pesos w_i, para $1 \leq i \leq n$, encontrar um subconjunto $S \subseteq U$ que maximize $\sum_{i \in S} v_i$ com a restrição $\sum_{i \in S} w_i \leq m$.

O número m é a capacidade da mochila e queremos selecionar o subconjunto mais valioso que cabe na mochila.

Primeiro descreveremos um algoritmo exato que usa programação dinâmica mas não é polinomial, e depois um algoritmo aproximativo polinomial.

$$W_{ij} = \min \{w(S) : S \subseteq \{1, \cdots, i\} \text{ e } v(S) \geq j\}$$

peso mínimo de um subconjunto de $\{1, \cdots, i\}$ cujo valor é pelo menos j.

Algoritmo Mochila-Exato (m, n, v, w)

1. para i de 0 a n faça $W_{i0} \leftarrow 0$
2. $j \leftarrow 0$
3. repita
4. $\quad j \leftarrow j+1$
5. $\quad W_{0j} \leftarrow \infty$
6. $\quad$ para i de 1 a n faça
7. $\quad\quad$ se $v_i \geq j$
8. $\quad\quad\quad$ então $W_{ij} \leftarrow \min\{W_{i-1,j}, w_i + W_{i-1,j-v_i}\}$
9. $\quad\quad\quad$ senão $W_{ij} \leftarrow W_{i-1,j}$
10. até que $W_{nj} > m$
11. seja S um subconjunto de $\{1, \ldots, n\}$
12. $\quad$ com $w(S) = W_{n,j-1}$ e $v(S) \geq j-1$
13. retorne S

Exemplo: $m = 7$, $n = 5$, $v = (2,1,3,4,1)$ $w = (4,2,1,2,2)$

W	0	1	2	3	4	5	6	7	8	9	10
0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
1	0	4	4	∞	∞	∞	∞	∞	∞	∞	∞
2	0	2	4	6	∞	∞	∞	∞	∞	∞	∞
3	0	1	1	1	3	5	7	∞	∞	∞	∞
4	0	1	1	1	2	3	3	3	5	7	9
5	0	1	1	1	2	3	3	3	5	7	9

$opt(I) = 9$

$S_1^* = \{1, 3, 4\}$

$S_2^* = \{2, 3, 4, 5\}$

$W_{22} = \min\{W_{12}, 2 + W_{1,(2-1)}\} = \min\{4, 6\} = 4$

$W_{52} = \min\{W_{42}, 2 + W_{4,(2-1)}\} = \min\{\infty, 2+\infty\} = \infty$

$W_{5,9} = \min\{W_{49}, 2 + W_{4,(9-1)}\} = \min\{7, 7\} = 7$

$W_{ij} = \min\{w(S) : S \subset \{1,\cdots,i\} \text{ e } v(S) \geq j\}$

peso mínimo de um subconjunto de $\{1,\cdots,i\}$ cujo valor é pelo menos j.

Se $i \notin S$, então $S \subseteq \{1,\cdots,i-1\}$: $W_{(i-1)j}$

Se $i \in S$, então $S - \{i\} \subseteq \{1,\cdots,i-1\}$: $W_{(i-1)(j-v_i)}$

Se $v_i \geq j$, então $W_{ij} \leftarrow \min\{W_{(i-1)j}, w_i\}$

Observe que se $v_i \geq j$, então comparamos os pesos de $S \subseteq \{1,\cdots,i-1\}$ e $S = \{i\}$.

Se $v_i < j$ então $W_{ij} \leftarrow \min\{W_{(i-1)j}, w_i + W_{(i-1)(j-v_i)}\}$

Observe que se $v_i < j$, então comparamos os pesos de $S \subseteq \{1,\cdots,i-1\}$ e $\{i\} \cup S'$.

Para a condição de parada na linha 10, observe que $W_{nj} \leq W_{nk}$ para $j \leq k$, isto é, os pesos nas linhas não diminuem, todo subconjunto $S \subseteq \{1, \ldots, n\}$ que satisfaz $v(S) \geq k$, também satisfaz $v(S) \geq j$, lembre que a última linha:

$$W_{nj} = \min \{ w(S) : S \subseteq \{1, \ldots, i\} \text{ e } v(S) \geq j \}$$

Portanto, se $W_{nj} > m$, então $W_{nk} > m$ para todo $k > j$ e podemos parar.

O algoritmo consome tempo proporcional do tamanho da tabela W, que é limitado por $(n+1)(\tau_v + 1)$, onde

$$\tau_v = \sum_{i \text{ tq } w_i \leq m} v_i .$$

Algoritmo Mochila-$IK_\varepsilon(m, n, v, w)$

1. se $w_i > m$ para todo i
2. então devolva $\emptyset$
3. senão $\vartheta \leftarrow \max_{i \text{ tq } w_i \leq m} v_i$
4. $\lambda \leftarrow \frac{\varepsilon \vartheta}{n}$
5. para i de 1 a n faça $u_i \leftarrow \left\lfloor \frac{v_i}{\lambda} \right\rfloor$
6. $S \leftarrow$ Mochila-Exato(m, n, u, w)
7. retorne S

Descreveremos um esquema de aproximação. Seja ε um racional no intervalo $(0, 1)$. O algoritmo faz uma mudança de escala nos valores da instância (m, n, v, w) obtendo uma outra instância (m, n, u, w) onde o algoritmo Mochila-Exato consome tempo polinomial.

Teorema: O algoritmo Mochila-IK_ε é uma $(1-\varepsilon)$-aproximação polinomial.

prova: Como S é uma solução ótima do problema Mochila (m,n,u,w), temos que $\sum_{i\in S} w_i \leq m$, e portanto S é uma solução viável do problema Mochila (m,n,v,w).

Seja S^* uma solução ótima de Mochila(m,n,v,w), então

$$\sum_{i\in S} v_i \geqslant \lambda \sum_{i\in S} u_i \geqslant \lambda \sum_{i\in S^*} u_i \geqslant \lambda \sum_{i\in S^*} \left(\frac{v_i}{\lambda} - 1\right)$$

S é solução ótima de Mochila(m,n,u,w)

$u_i > \frac{v_i}{\lambda} - 1$, para todo $i \in S^*$

na verdade $=$

$$\geqslant v(S^*) - \lambda |S^*| \geqslant v(S^*) - \lambda n = v(S^*) - \varepsilon\vartheta$$

$$\geqslant (1-\varepsilon)\, opt\,(m,n,v,w)$$

$\uparrow$

o valor do objeto mais valioso cujo peso não excede a capacidade da mochila, isto é $opt(m,n,v,w) \geqslant \vartheta$ é um limite inferior para o ótimo. ∎

Para a análise de tempo, a linha 6 consome $\Theta(n(\tau_u+1))$, onde $\tau_u = \sum_{i \text{ t.q. } w_i \leq m} u_i$.

Mas $u_i \leq \frac{v_i}{\lambda} \leq \frac{n}{\varepsilon}$, para todo i tal que $w_i \leq m$.

Portanto $\tau_u \leq \frac{n^2}{\varepsilon}$ e o algoritmo consome tempo $\Theta(\frac{n^3}{\varepsilon})$ polinomial. ▨

Obtemos um esquema de aproximação; para cada ε no intervalo $(0,1)$, uma $(1-\varepsilon)$-aproximação que consome tempo polinomial $\Theta(\frac{n^3}{\varepsilon})$.