Algoritmi per il problema dello zaino

Dip. Informatica ed Appl. *Prof. G. Persiano* *Università di Salerno*

Il problema dello zaino ha due varianti: lo zaino 0/1 e lo zaino frazionario. In questi appunti presentiamo un algoritmo *pseudopolinomiale* per il problema dello zaino 0/1.

1 Zaino 0/1

Un'istanza del problema dello zaino 0/1 è costituita da n oggetti e da una capacità massima B. L'i-esimo oggetto ha volume v_i e prezzo (valore) c_i. Lo scopo è di individuare il sottoinsieme di prezzo massimo tra quelli che hanno volume complessivo non maggiore di B.

Formalmente, una soluzione ammissibile per un'istanza $I = ((v_1, \cdots, v_n), (c_1, \cdots, c_n), W)$ è costituita da una sequenza $S = (b_1, \cdots, b_n) \in \{0, 1\}^n$ tale che

$$\sum_{i=1}^{n} b_i \cdot v_i \leq B.$$

Il nostro compito è quello di trovare, tra tutte le soluzioni ammissibili per un'istanza I, una soluzione $S^* = (b_1^*, \cdots, b_n^*)$ che massimizzi il profitto $C(S^*)$ di S^* definito come

$$C(S^*) = \sum_{i=1}^{n} b_i^* \cdot c_i.$$

Equazione di ricorrenza. Data un'istanza $I = ((v_1, \cdots, v_n), (c_1, \cdots, c_n), B)$ denotiamo con $C(k, w)$ il massimo prezzo di una soluzione ammissibile per l'istanza $I(k, w) = ((v_k, \cdots, v_n), (c_k, \cdots, c_n), w)$. Ovviamente, essendo $I = I(1, B)$, il prezzo massimo di una soluzione ammissibile per I è $C(1, B)$.

Abbiamo che $C(k, 0) = 0$ per tutti i valori di k. Inoltre, $C(n, w) = c_n$ se $w \geq v_n$ e $C(n, w) = 0$ altrimenti. Proviamo che

$$C(k, w) = \begin{cases} \max\{C(k+1, w), C(k+1, w - v_k) + c_k\}; & \text{se } v_k \leq w; \\ C(k+1, w); & \text{se } v_k > w; \end{cases}$$

per $k < n$.

L'insieme delle soluzioni ammissibili per $I(k, w)$ può essere partizionato in due insiemi: $S^+(k, w)$ (l'insieme delle soluzioni che includono l'oggetto k) e $S^-(k, w)$ (l'insieme delle soluzioni che non lo includono). Il prezzo massimo è chiaramente ottenuto calcolando il massimo tra il massimo profitto $C^+(k, w)$ ottenuto dalle migliori soluzioni in $S^+(k, w)$ e il massimo profitto $C^-(k, w)$ ottenuto dalle migliori soluzioni in $S^-(k, w)$. Ovviamente, se $v_k > w$ allora $S^+(k, w) = \emptyset$ e quindi $C(k, w) = C^-(k, w)$.

Calcoliamo ora un'espressione per $C^-(k, w)$ e $C^+(k, w)$. Per $C^-(k, w)$, osserviamo che ogni soluzione ammissibile per $I(k + 1, w)$ è una soluzione ammissibile per $I(k, w)$ che non include l'oggetto k e viceversa. Pertanto $C^-(k, w) = C(k + 1, w)$.

Per $C^+(k, w)$, osserviamo che una soluzione $S = (b_{k+1}, \cdots, b_n)$ è ammissibile per $I(k+1, w-v_i)$ (istanza in cui consideriamo solo gli oggetti $k+1, \cdots, n$ ed abbiamo una capacità massima $w - v_i$) se e solo se la soluzione $S' = (1, b_{k+1}, \cdots, b_n)$ (aggiungiamo l'oggetto k alla soluzione S) è ammissibile per $I(k, w)$. Inoltre il profitto di S' è uguale al profitto di S aumentato del profitto c_k derivante dall'inclusione dell'oggetto k. Pertanto $C^+(k, w) = C(k + 1, w - v_i) + c_k$.

Risolvere l'equazione di ricorrenza. L'equazione di ricorrenza presentata al paragrafo precedente può essere risolta calcolando i valori $C(k, w)$ partendo da $k = n$ fino a $k = 1$. Per ciascuno valore di k, i valore di $C(k, w)$ sono calcolati da $w = 0$ fino a $w = B$.

Algoritmo ZainoIntero($((v_1, \cdots, v_n), (c_1, \cdots, c_n), B)$)

for $k = 1$ **to** n
    $C(k, 0) = 0$;
for $w = 0$ **to** B
    if $w \geq v_n$ **then** $C(n, w) = c_n$
        else $C(n, w) = 0$;
endfor

for $k = n - 1$ **to** 1
    for $w = 0$ **to** B
        if $w \geq v_k$ **then**
            $C(k, w) = \max\{C(k + 1, w), C(k + 1, w - v_k) + c_k\}$;
            else $C(k, w) = C(k + 1, w)$;

return $C(1, B)$;

L'algoritmo impiega tempo proporzionale a $O(n \cdot B)$. Sottolineamo che pertanto l'algoritmo presentato *non* è polinomiale. Un algoritmo polinomiale deve compiere un numero di passi polinomiale nella lunghezza (in bit) dell'input. Un istanza $I = ((v_1, \cdots, v_n), (c_1, \cdots, c_n), B)$ è lunga $O(n \log M)$ ove M è il massimo tra gli interi che denotanto i volumi, i prezzi e la capienza dello zaino. L'algoritmo presentato invece compie invece un numero di passi $O(n \cdot 2^{\log M})$ ed è pertanto esponenziale. La teoria dell'NP-completezza ci dice che è molto inverosimile che esista un algoritmo polinomiale per lo zaino. Algoritmi che sono polinomiali nel massimo numero presente in input (e non nella lunghezza del massimo input) sono detti *pseudopolinomiali*.