PD per il problema degli zaini multipli

Indice

Un algoritmo di PD
- Numero di passi
- La crescita con il numero di zaini

Errori tipici

In sintesi

Obiettivi

Al termine di questo capitolo saremo in grado di:

- scrivere la ricorrenza $v(j, k_1, \ldots, k_m)$ sugli oggetti e sulle capacità residue degli m zaini;
- eseguire a mano l'algoritmo di PD su un'istanza con $m = 2$ zaini, una matrice per ogni oggetto;
- ricostruire dalle tabelle s una soluzione ottima, con lo zaino di ogni oggetto selezionato;
- contare i $\Theta(m\, n \prod_{i=1}^{m}(c_i + 1))$ passi e spiegare perché il numero di stati cresce esponenzialmente con m.

Prerequisiti

- Da questo modulo, il capitolo «PD per il problema dello zaino 0/1»: la ricorrenza $v(j, k)$ del problema dello zaino 0/1 (KP) su oggetti e capacità, la matrice s delle decisioni e la ricostruzione di una soluzione.
- Dal modulo *Problemi di ottimizzazione combinatoria*, il capitolo «Problemi di zaino»: il problema degli zaini multipli, con m zaini di capacità $c_1, \ldots, c_m$.
- Dal modulo *Algoritmi*: le convenzioni dello pseudocodice e la dimensione dell'input.

Il problema di questo capitolo

Il problema di questo capitolo è il
problema degli zaini multipli (MKP):

la sua definizione e le sue ipotesi sono nella sua pagina.

- Scriviamo un algoritmo di PD che dà il valore ottimo di un'istanza dell'MKP
- Lo stesso algoritmo dà anche una soluzione ottima: lo zaino di ogni oggetto selezionato

Un algoritmo di PD

Introduzione: Un algoritmo di PD

- In questa sezione scriviamo un algoritmo di PD per l'MKP, in due passi: prima il valore ottimo, poi una soluzione ottima
- L'algoritmo segue l'algoritmo di PD del KP, ma la capacità diventa un vettore: una capacità residua per ogni zaino
- Lo eseguiamo a mano su un'istanza con $m = 2$ zaini e quattro oggetti, ne contiamo i passi, e vediamo come cresce il numero di stati con il numero di zaini

Zaini, soluzioni e vettori unitari

- Indichiamo con $i \in \mathscr{K} = \{1, 2, \ldots, m\}$ uno zaino generico, perché nei capitoli di PD la lettera k indica una capacità. Lo zaino i ha capacità c_i.

- Una soluzione assegna ogni oggetto j a uno zaino oppure lo esclude: si descrive con il vettore $(a_1, a_2, \ldots, a_n)$, dove
  - $a_j = i$ se l'oggetto j è inserito nello zaino i,
  - $a_j = 0$ se l'oggetto j non è selezionato.

- Il vettore $\boldsymbol{e}_i$ è l'i-esimo vettore unitario di lunghezza m: $\boldsymbol{k} - w_j \boldsymbol{e}_i$ è il vettore $\boldsymbol{k}$ con la componente k_i diminuita di w_j.

Esempio: istanza dell'MKP

Consideriamo $m = 2$ zaini con capacità $c_1 = 5$ e $c_2 = 4$, e un insieme $\mathscr{I} = \{1, 2, 3, 4\}$ di 4 oggetti ($n = 4$), con i seguenti profitti e pesi:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$
p_j	3	9	7	5
w_j	3	4	4	2

Usiamo questa istanza dell'MKP come esempio di tutto il capitolo.

I due passi dell'algoritmo di PD

- Passo 1: calcolo del valore ottimo
  - una formula ricorsiva dà il valore ottimo di ogni sottoistanza, con le sue capacità residue
- Passo 2: calcolo di una soluzione ottima
  - le tabelle s memorizzano, per ogni oggetto, lo zaino in cui è inserito, oppure 0

Passo 1: le sottoistanze

Per ogni $j \in \{0, 1, \ldots, n\}$ e ogni vettore $\boldsymbol{k} = (k_1, k_2, \ldots, k_m)$ con $k_i \in \{0, 1, \ldots, c_i\}$ per ogni $i \in \mathscr{K}$, sia

$$v(j, \boldsymbol{k}) = v(j, k_1, \ldots, k_m)$$

il valore ottimo della sottoistanza dell'MKP data da:

1. m zaini, lo zaino i con capacità k_i,
2. un insieme $\{1, 2, \ldots, j\}$ di oggetti disponibili,
3. per ogni oggetto $h \in \{1, 2, \ldots, j\}$, un profitto p_h e un peso w_h.

Passo 1: la formula ricorsiva

La formula ricorsiva per calcolare $v(j, \boldsymbol{k})$ è la seguente:

$$v(j,\boldsymbol{k}) = \begin{cases} 0 & \text{se } j = 0, \\ v(j-1,\boldsymbol{k}) & \text{se } j \geq 1 \text{ e} \\ & w_j > k_i \text{ per ogni } i, \\ \max\Big\{v(j-1,\boldsymbol{k}), & \text{se } j \geq 1 \text{ e } w_j \leq k_i \\ \quad p_j + \max\limits_{i:\, w_j \leq k_i} v(j-1,\boldsymbol{k} - w_j \boldsymbol{e}_i)\Big\} & \text{per qualche } i. \end{cases}$$

Sapienza Università di Roma

Come si legge la formula ricorsiva

- Le componenti di $\boldsymbol{k}$ sono le **capacità residue** degli zaini: nella visione top-down, le decisioni sugli oggetti $j+1, \ldots, n$ hanno già usato $c_i - k_i$ unità dello zaino i.
- Con $m = 1$ la formula ricorsiva è la ricorrenza del KP.
- I casi base sono $v(0, \boldsymbol{k}) = 0$ per ogni vettore $\boldsymbol{k}$, perché nessun oggetto è disponibile.

Sottostruttura ottima

- La formula genera al più $m + 1$ **sottoproblemi indipendenti**:
  - l'oggetto j viene escluso, oppure
  - viene inserito in uno degli zaini i con $w_j \leq k_i$, e allora diminuisce solo la capacità dello zaino i.

$\rightsquigarrow$ **Sottostruttura ottima verificata!**

Sottoproblemi sovrapposti e approccio bottom-up

- Assegnamenti diversi degli oggetti $j+1, \ldots, n$ possono lasciare le stesse capacità residue, e quindi lo stesso valore $v(j, \boldsymbol{k})$ può servire più volte.

  $\rightsquigarrow$ **Sottoproblemi sovrapposti verificati!**

- Il valore $v(j, \boldsymbol{k})$ dipende solo da valori della riga $j-1$.
- Quindi l'**approccio bottom-up** calcola le righe $j = 0, 1, \ldots, n$ in quest'ordine, e in ogni riga tutti i vettori $\boldsymbol{k}$ in un ordine qualsiasi.
- Con $m = 2$ la riga j è una matrice, con una riga per ogni k_1 e una colonna per ogni k_2.

Passo 2: calcolo di una soluzione ottima

- Per ogni oggetto j e vettore $\boldsymbol{k}$ memorizziamo:
  - $s[j][\boldsymbol{k}] = i$ se l'oggetto j è inserito nello zaino i nella soluzione ottima trovata per $(j, \boldsymbol{k})$,
  - $s[j][\boldsymbol{k}] = 0$ se l'oggetto j è escluso.
- L'insieme di tutti i vettori $\boldsymbol{k}$ è

$$\mathscr{C} = \{0, 1, \ldots, c_1\} \times \{0, 1, \ldots, c_2\} \times \cdots \times \{0, 1, \ldots, c_m\}.$$

Segue lo pseudocodice dell'**algoritmo di PD con l'approccio bottom-up**,
che memorizza le tabelle s insieme alle tabelle v.

Algorithm: MKP_DP (1/2)

Input : Un'istanza dell'MKP data da n, m, dai vettori $p[1..n]$, $w[1..n]$ e dal vettore delle capacità $\boldsymbol{c}$

Output : Il valore ottimo dell'istanza dell'MKP e le tabelle $s[j][\boldsymbol{k}]$

```
1 function MKP_DP(n, m, p, w, c)
2 |   v := new array[0..n][0..c_1] ··· [0..c_m];
  |     s := new array[0..n][0..c_1] ··· [0..c_m]
3 |   foreach k ∈ 𝒞 do
4 |   |   v[0][k] ← 0; s[0][k] ← 0
5 |   end
```

Algorithm: MKP_DP (2/2)

function *MKP_DP*$(n, m, p, w, \boldsymbol{c})$ (continua)

```
6     for j := 1 to n do
7         foreach k ∈ 𝒞 do
8             v[j][k] ← v[j − 1][k]; s[j][k] ← 0
9             for i := 1 to m do
10                if w[j] ≤ k_i e
                    p[j] + v[j − 1][k − w[j] e_i] > v[j][k] then
11                    v[j][k] ← p[j] + v[j − 1][k − w[j] e_i];
                        s[j][k] ← i
12                end
13            end
14        end
15    end
16    return v[n][c], s
17 end
```

Come funziona MKP_DP

- Per ogni vettore $\boldsymbol{k}$, l'algoritmo di PD prima esclude l'oggetto j, e poi prova a inserirlo in ogni zaino $i = 1, 2, \ldots, m$ in cui entra.

- In caso di pareggio si conserva la prima opzione incontrata:
  - escludere l'oggetto j,
  - poi lo zaino di indice minimo,

  perché il valore viene aggiornato solo quando è strettamente maggiore.

Esempio: esecuzione dell'algoritmo di PD (1/7)

Consideriamo l'istanza dell'MKP dell'esempio precedente, con capacità $c_1 = 5$ e $c_2 = 4$:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$
p_j	3	9	7	5
w_j	3	4	4	2

- La riga $j = 0$ è $v[0][\boldsymbol{k}] = 0$ per ogni $\boldsymbol{k}$.
- Per ogni oggetto j calcoliamo la matrice $v[j][k_1][k_2]$, con una riga per ogni $k_1 \in \{0, 1, \ldots, 5\}$ e una colonna per ogni $k_2 \in \{0, 1, \ldots, 4\}$.
- Prima i calcoli della cella $\boldsymbol{k} = \boldsymbol{c} = (5, 4)$ per ogni oggetto, e delle celle usate dalla ricostruzione.

Esempio: esecuzione dell'algoritmo di PD (2/7)

Oggetti $j = 1$ e $j = 2$:

$$\begin{aligned} v[1][(5,4)] &= \max\{v[0][(5,4)],\ p_1 + v[0][(2,4)],\ p_1 + v[0][(5,1)]\} \\ &= \max\{0,\ 3+0,\ 3+0\} = 3 \end{aligned}$$

$$\begin{aligned} v[1][(3,0)] &= \max\{v[0][(3,0)],\ p_1 + v[0][(0,0)]\} \\ &= \max\{0,\ 3+0\} = 3 \end{aligned}$$

$$\begin{aligned} v[2][(5,4)] &= \max\{v[1][(5,4)],\ p_2 + v[1][(1,4)],\ p_2 + v[1][(5,0)]\} \\ &= \max\{3,\ 9+3,\ 9+3\} = 12 \end{aligned}$$

$$\begin{aligned} v[2][(3,4)] &= \max\{v[1][(3,4)],\ p_2 + v[1][(3,0)]\} \\ &= \max\{3,\ 9+3\} = 12 \end{aligned}$$

Esempio: esecuzione dell'algoritmo di PD (3/7)

Oggetti $j = 3$ e $j = 4$:

$$\begin{aligned} v[3][(5,4)] &= \max\{v[2][(5,4)],\ p_3 + v[2][(1,4)],\ p_3 + v[2][(5,0)]\} \\ &= \max\{12,\ 7+9,\ 7+9\} = 16 \end{aligned}$$

$$\begin{aligned} v[3][(3,4)] &= \max\{v[2][(3,4)],\ p_3 + v[2][(3,0)]\} \\ &= \max\{12,\ 7+3\} = 12 \end{aligned}$$

$$\begin{aligned} v[4][(5,4)] &= \max\{v[3][(5,4)],\ p_4 + v[3][(3,4)],\ p_4 + v[3][(5,2)]\} \\ &= \max\{16,\ 5+12,\ 5+9\} = 17 \end{aligned}$$

Esempio: esecuzione dell'algoritmo di PD (4/7)

Oggetto $j = 1$, con $p_1 = 3$ e $w_1 = 3$:

$v[1][k_1][k_2]$	$k_2 = 0$	$k_2 = 1$	$k_2 = 2$	$k_2 = 3$	$k_2 = 4$
$k_1 = 0$	0	0	0	3	3
$k_1 = 1$	0	0	0	3	3
$k_1 = 2$	0	0	0	3	3
$k_1 = 3$	3	3	3	3	3
$k_1 = 4$	3	3	3	3	3
$k_1 = 5$	3	3	3	3	3

Esempio: esecuzione dell'algoritmo di PD (5/7)

Oggetto $j = 2$, con $p_2 = 9$ e $w_2 = 4$:

$v[2][k_1][k_2]$	$k_2 = 0$	$k_2 = 1$	$k_2 = 2$	$k_2 = 3$	$k_2 = 4$
$k_1 = 0$	0	0	0	3	9
$k_1 = 1$	0	0	0	3	9
$k_1 = 2$	0	0	0	3	9
$k_1 = 3$	3	3	3	3	12
$k_1 = 4$	9	9	9	12	12
$k_1 = 5$	9	9	9	12	12

Esempio: esecuzione dell'algoritmo di PD (6/7)

Oggetto $j = 3$, con $p_3 = 7$ e $w_3 = 4$:

$v[3][k_1][k_2]$	$k_2 = 0$	$k_2 = 1$	$k_2 = 2$	$k_2 = 3$	$k_2 = 4$
$k_1 = 0$	0	0	0	3	9
$k_1 = 1$	0	0	0	3	9
$k_1 = 2$	0	0	0	3	9
$k_1 = 3$	3	3	3	3	12
$k_1 = 4$	9	9	9	12	16
$k_1 = 5$	9	9	9	12	16

Esempio: esecuzione dell'algoritmo di PD (7/7)

Oggetto $j = 4$, con $p_4 = 5$ e $w_4 = 2$:

$v[4][k_1][k_2]$	$k_2 = 0$	$k_2 = 1$	$k_2 = 2$	$k_2 = 3$	$k_2 = 4$
$k_1 = 0$	0	0	5	5	9
$k_1 = 1$	0	0	5	5	9
$k_1 = 2$	5	5	5	8	14
$k_1 = 3$	5	5	8	8	14
$k_1 = 4$	9	9	14	14	16
$k_1 = 5$	9	9	14	14	17

Esempio: le tabelle s (1/4)

Per l'istanza dell'esempio precedente, queste sono le matrici $s[j][k_1][k_2]$ calcolate dall'algoritmo di PD: 0 indica che l'oggetto j è escluso, 1 e 2 lo zaino in cui è inserito.

Oggetto $j = 1$, con $p_1 = 3$ e $w_1 = 3$:

$s[1][k_1][k_2]$	$k_2 = 0$	$k_2 = 1$	$k_2 = 2$	$k_2 = 3$	$k_2 = 4$
$k_1 = 0$	0	0	0	2	2
$k_1 = 1$	0	0	0	2	2
$k_1 = 2$	0	0	0	2	2
$k_1 = 3$	1	1	1	1	1
$k_1 = 4$	1	1	1	1	1
$k_1 = 5$	1	1	1	1	1

Esempio: le tabelle s (2/4)

Oggetto $j = 2$, con $p_2 = 9$ e $w_2 = 4$:

$s[2][k_1][k_2]$	$k_2 = 0$	$k_2 = 1$	$k_2 = 2$	$k_2 = 3$	$k_2 = 4$
$k_1 = 0$	0	0	0	0	2
$k_1 = 1$	0	0	0	0	2
$k_1 = 2$	0	0	0	0	2
$k_1 = 3$	0	0	0	0	2
$k_1 = 4$	1	1	1	1	1
$k_1 = 5$	1	1	1	1	1

Esempio: le tabelle s (3/4)

Oggetto $j = 3$, con $p_3 = 7$ e $w_3 = 4$:

$s[3][k_1][k_2]$	$k_2 = 0$	$k_2 = 1$	$k_2 = 2$	$k_2 = 3$	$k_2 = 4$
$k_1 = 0$	0	0	0	0	0
$k_1 = 1$	0	0	0	0	0
$k_1 = 2$	0	0	0	0	0
$k_1 = 3$	0	0	0	0	0
$k_1 = 4$	0	0	0	0	1
$k_1 = 5$	0	0	0	0	1

Esempio: le tabelle s (4/4)

Oggetto $j = 4$, con $p_4 = 5$ e $w_4 = 2$:

$s[4][k_1][k_2]$	$k_2 = 0$	$k_2 = 1$	$k_2 = 2$	$k_2 = 3$	$k_2 = 4$
$k_1 = 0$	0	0	2	2	0
$k_1 = 1$	0	0	2	2	0
$k_1 = 2$	1	1	1	1	1
$k_1 = 3$	1	1	2	1	1
$k_1 = 4$	0	0	2	2	0
$k_1 = 5$	0	0	2	2	1

Ricostruzione di una soluzione ottima

- Calcolate le tabelle s, si ricostruisce una soluzione ottima:
  - scorrendo gli oggetti da $j = n$ fino a $j = 1$,
  - partendo da $\boldsymbol{k} = \boldsymbol{c}$ e da $a_j = 0$ per ogni oggetto j:

$$a_j \leftarrow s[j][\boldsymbol{k}], \qquad \text{e, se } a_j = i \geq 1, \quad \boldsymbol{k} \leftarrow \boldsymbol{k} - w_j \boldsymbol{e}_i.$$

Algorithm: MKP_Reconstruct

Input : Il numero n di oggetti, il vettore delle capacità $\boldsymbol{c}$, il vettore $w[1..n]$ e le tabelle s

Output : Una soluzione ottima dell'istanza dell'MKP, come vettore degli zaini

```
1  function MKP_Reconstruct(n, c, w, s)
2      a := new array[1..n]
3      k := c
4      for j := n downto 1 do
5          a[j] ← s[j][k]
6          if a[j] ≥ 1 then
7              k ← k − w[j] e_{a[j]}
8          end
9      end
10     return a
11 end
```

Esempio: calcolo di una soluzione ottima (1/2)

Dopo l'esecuzione dell'algoritmo di PD sull'istanza dell'esempio precedente, partiamo da $j = 4$ e $\boldsymbol{k} = (5, 4)$:

Iterazione	j	$\boldsymbol{k}$	$s[j][\boldsymbol{k}]$	$a[1]$	$a[2]$	$a[3]$	$a[4]$	Azione
–	–	–	–	0	0	0	0	$\boldsymbol{k} = (5, 4)$
1	4	$(5, 4)$	1	0	0	0	1	$a[4] \leftarrow 1$, $\boldsymbol{k} \leftarrow (3, 4)$
2	3	$(3, 4)$	0	0	0	0	1	$a[3] \leftarrow 0$, $\boldsymbol{k} = (3, 4)$
3	2	$(3, 4)$	2	0	2	0	1	$a[2] \leftarrow 2$, $\boldsymbol{k} \leftarrow (3, 0)$
4	1	$(3, 0)$	1	1	2	0	1	$a[1] \leftarrow 1$, $\boldsymbol{k} \leftarrow (0, 0)$

Esempio: calcolo di una soluzione ottima (2/2)

- Il vettore $a = (1, 2, 0, 1)$ è una soluzione ottima dell'istanza dell'MKP dell'esempio:
  - lo zaino 1 contiene gli oggetti $\{1, 4\}$, di peso totale $3 + 2 = 5 \leq c_1$,
  - lo zaino 2 contiene gli oggetti $\{2\}$, di peso totale $4 \leq c_2$.

  Il valore ottimo è $v(4, 5, 4) = 17$.

- Riempire gli zaini uno dopo l'altro non basta:
  - il miglior riempimento del solo zaino 1 è $\{2\}$, di profitto 9,
  - il miglior riempimento dello zaino 2 con gli oggetti rimasti è $\{3\}$, di profitto 7,

  per un totale di $16 < 17$.

Quanti passi esegue l'algoritmo di PD?

Contiamo quante volte viene eseguita
ogni riga dello pseudocodice di MKP_DP.

Numero di passi (1/3)

- Sia

$$N = |\mathscr{C}| = \prod_{i=1}^{m} (c_i + 1)$$

  il numero di vettori $\boldsymbol{k}$.

- Indichiamo con $S(n, m, \boldsymbol{c})$ il numero di passi eseguiti da `MKP_DP` nel caso peggiore.

- Inserire l'oggetto j nello zaino i cambia solo la componente k_i, quindi le righe 10 e 11 contano un passo ciascuna.

Numero di passi (2/3)

- Le righe 2 e 16 vengono eseguite 1 volta ciascuna.
- Le righe 3 e 4 vengono eseguite N volte ciascuna.
- La riga 6 viene eseguita n volte.
- Le righe 7 e 8 vengono eseguite nN volte ciascuna.
- Le righe 9, 10 e 11 vengono eseguite al più $m\,nN$ volte ciascuna.

Numero di passi (3/3)

Quindi il numero totale di passi nel caso peggiore è

$$S(n, m, \boldsymbol{c}) = 3m\,nN + 2nN + 2N + n + 2,$$

e, poiché

$$3m\,nN \leq S(n, m, \boldsymbol{c}) \leq 10m\,nN,$$

si ha

$$S(n, m, \boldsymbol{c}) = \Theta\left(m\,n \prod_{i=1}^{m} (c_i + 1)\right).$$

Il numero di passi dell'algoritmo di PD

L'algoritmo di PD per l'MKP esegue

$$\Theta(m\, n \prod_{i=1}^{m}(c_i + 1))$$

passi nel caso peggiore.

- Per un numero m fissato di zaini questo è $\Theta(n \prod_{i=1}^{m} c_i)$, un numero **pseudo-polinomiale** di passi.
- La ricostruzione esegue $\Theta(n)$ passi.

La crescita con il numero di zaini (1/2)

- Ogni nuovo zaino moltiplica il numero di stati per $c_i + 1$.
- Se tutte le capacità sono uguali a c, la tabella ha

$$(n+1)(c+1)^m$$

  celle.
- Il numero di stati cresce **esponenzialmente** con m, mentre la dimensione dell'input cresce di un solo numero, la capacità c_i.

La crescita con il numero di zaini (2/2)

- Con $n = 50$ oggetti e capacità $c_i = 100$, la tabella ha:

m	numero di celle
1	$51 \cdot 101 = 5151$
2	$51 \cdot 101^2 = 520\,251$
5	$51 \cdot 101^5 \approx 5.4 \cdot 10^{11}$

- Zaini con la stessa capacità danno stati simmetrici: se $c_1 = c_2$, i vettori $(k_1, k_2, \dots)$ e $(k_2, k_1, \dots)$ hanno lo stesso valore ottimo, e l'algoritmo di PD li calcola entrambi.

Riepilogo (1/2)

Passo 1: il valore ottimo:

- lo stato è dato dall'oggetto j e dal vettore $\boldsymbol{k}$ delle capacità residue
- l'oggetto j viene escluso o inserito in uno zaino i con $w_j \leq k_i$: solo k_i diminuisce di w_j
- casi base $v(0, \boldsymbol{k}) = 0$; con $m = 1$ la ricorrenza del KP
- sottostruttura ottima e sottoproblemi sovrapposti: bottom-up, la riga j dopo la riga $j - 1$

Passo 2: una soluzione ottima:

- $s[j][\boldsymbol{k}]$ è lo zaino dell'oggetto j, oppure 0 se l'oggetto j è escluso
- sull'esempio: $a = (1, 2, 0, 1)$, di valore $v(4, 5, 4) = 17$

Riepilogo (2/2)

Numero di passi:

- $S(n, m, \boldsymbol{c}) = \Theta(m\, n \prod_{i=1}^{m}(c_i + 1))$ per MKP_DP, $\Theta(n)$ per la ricostruzione
- pseudo-polinomiale per un numero m fissato di zaini

La crescita con m:

- ogni nuovo zaino moltiplica il numero di stati per $c_i + 1$
- il numero di stati cresce esponenzialmente con m

Errori tipici

Introduzione: Errori tipici

- Raccogliamo gli errori che si fanno più spesso con l'algoritmo di PD per l'MKP
- Due riguardano le capacità, due le scelte della ricorrenza
- Controllateli prima di eseguire a mano l'algoritmo di PD

Errori tipici (1/2)

Osservazione 1: Diminuire tutte le capacità

L'oggetto j va in un solo zaino: $\boldsymbol{k} - w_j \boldsymbol{e}_i$ diminuisce la componente k_i, non tutte come nel problema dello zaino multidimensionale.

Osservazione 2: Usare un'unica capacità $c_1 + \cdots + c_m$

Il KP con la capacità totale dà solo un upper bound: la sua soluzione può non essere divisibile fra gli zaini.

Errori tipici (2/2)

Osservazione 3: Riempire gli zaini uno dopo l'altro

Un riempimento ottimo dello zaino 1*, seguito da un riempimento ottimo dello zaino* 2 *con gli oggetti rimasti, può essere peggiore dell'ottimo.*

Osservazione 4: Dimenticare l'opzione «oggetto escluso»

Il termine $v(j-1, \boldsymbol{k})$ *è sempre un candidato, anche quando l'oggetto* j *entra in qualche zaino.*

Riepilogo

Le capacità:

- l'oggetto j diminuisce solo la componente k_i dello zaino in cui è inserito
- un'unica capacità totale $c_1 + \cdots + c_m$ dà solo un upper bound

Le scelte:

- riempire gli zaini uno dopo l'altro può mancare l'ottimo
- l'opzione «oggetto escluso», $v(j-1, \boldsymbol{k})$, è sempre un candidato

In sintesi

Introduzione: In sintesi

- Raccogliamo le idee chiave del capitolo sull'algoritmo di PD per l'MKP
- Il quadro riassuntivo raccoglie il risultato sull'esempio del capitolo

Riepilogo (1/2)

- Lo stato dell'algoritmo di PD è dato dall'oggetto j e dal vettore $\boldsymbol{k}$ delle capacità residue degli m zaini.
- L'oggetto j viene escluso o inserito in uno zaino i in cui entra: $m + 1$ opzioni, e diminuisce solo k_i.
- L'algoritmo di PD esegue $\Theta(m\, n \prod_{i=1}^{m}(c_i + 1))$ passi: il numero di stati cresce esponenzialmente con m.
- Le tabelle s memorizzano lo zaino di ogni oggetto, e una soluzione ottima si legge all'indietro a partire da $(n, \boldsymbol{c})$.

Riepilogo (2/2)

Quadro riassuntivo dell'esempio dell'MKP ($\boldsymbol{c} = (5, 4)$, quattro oggetti):

Metodo	Lower bound	Upper bound	Origine
Algoritmo di PD	17	17	valore ottimo, soluzione $a = (1, 2, 0, 1)$