

# Algoritmi di approssimazione per lo zaino (1/2)

# Indice

Perché gli algoritmi di approssimazione per il KP?

Notazione

Oggetto critico e capacità residua

L'algoritmo greedy

L'algoritmo Ext-Greedy

L'algoritmo  $H^2$

L'algoritmo  $H^\ell$

L'algoritmo Basic-FPTAS

# Obiettivi

Alla fine di questo capitolo saremo in grado di:

- ▶ eseguire a mano gli algoritmi greedy, Ext-Greedy,  $H^2$ ,  $H^\ell$  e Basic-FPTAS su un'istanza del KP;
- ▶ spiegare perché l'algoritmo greedy non ha una garanzia costante e perché Ext-Greedy ha garanzia 2;
- ▶ dimostrare le garanzie  $\frac{4}{3}$  di  $H^2$ ,  $\frac{\ell+2}{\ell+1}$  di  $H^\ell$  e  $\frac{1}{1-\varepsilon}$  dell'algoritmo Basic-FPTAS;
- ▶ confrontare i tempi di calcolo di  $H^\ell$  e dell'algoritmo Basic-FPTAS, e dire perché solo il secondo è un FPTAS;
- ▶ spiegare perché una garanzia di prestazione assoluta per il KP richiede un algoritmo esatto.

# Prerequisiti

- ▶ Da questo modulo, il capitolo «Principi degli algoritmi di approssimazione»: rapporto di prestazione, garanzie di prestazione relative e assolute, tightness, schemi di approssimazione, PTAS e FPTAS.
- ▶ Da questo modulo, i capitoli «Problemi di zaino» e «Formulazioni dei problemi di zaino»: il KP, il rilassamento lineare  $LP_{KP}$  e il bound di Dantzig, che usa l'oggetto critico.
- ▶ Da questo modulo, i capitoli «Euristiche costruttive» e «PD per il problema dello zaino 0/1»: l'algoritmo greedy e la programmazione dinamica sui profitti, usati dall'algoritmo Basic-FPTAS.

# Il problema di questo capitolo

Il problema di questo capitolo è il  
problema dello zaino 0/1 (KP):

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

- ▶ Studiamo algoritmi euristici per il KP che hanno una **garanzia di prestazione**
- ▶ Dall'algoritmo greedy, che non ne ha nessuna, fino a uno schema che si avvicina all'ottimo quanto vogliamo

# Perché gli algoritmi di approssimazione per il KP?

# Introduzione: Perché gli algoritmi di approssimazione per il KP?

- ▶ Partiamo da una sfida su una piccola istanza: la regola più naturale, quella greedy, perde una soluzione molto semplice
- ▶ Ci chiediamo quanto la regola greedy possa essere peggiore dell'ottimo
- ▶ Il capitolo risponde: peggiore quanto vogliamo; e mostra un rimedio economico e algoritmi sempre migliori

# Una sfida

Uno zaino di capacità  $c = 10$  e quattro oggetti con profitti e pesi

|       | $i = 1$ | $i = 2$ | $i = 3$ | $i = 4$ |
|-------|---------|---------|---------|---------|
| $p_i$ | 6       | 8       | 19      | 4       |
| $w_i$ | 2       | 3       | 9       | 5       |

- ▶ La regola greedy prende gli oggetti per efficienza non crescente, quando entrano: raccoglie il profitto 18.
- ▶ L'oggetto 3 da solo ha profitto 19.

# Quanto può sbagliare la regola greedy?

- ▶ La regola greedy può essere peggiore dell'ottimo quanto vogliamo?
- ▶ Provate a costruire un'istanza in cui raccoglie un profitto di 2 e il valore ottimo è grande a piacere.
- ▶ La risposta è sì, e il rimedio è economico: **confrontare con il miglior oggetto singolo.**
- ▶ Questo capitolo dimostra la garanzia di questa correzione, e di tre algoritmi migliori, fino a uno schema che si avvicina all'ottimo quanto vogliamo a un prezzo polinomiale.

## La sfida:

- ▶ Su una piccola istanza la regola greedy raccoglie 18, mentre un solo oggetto ha profitto 19

## Le domande del capitolo:

- ▶ Quanto può essere lontano dall'ottimo un algoritmo euristico, nel caso peggiore?
- ▶ Quali algoritmi hanno una garanzia, e a quale prezzo in tempo di calcolo?

# Notazione

# Introduzione: Notazione

- ▶ Fissiamo la notazione usata in tutto il capitolo: oggetti, profitti, pesi e capacità
- ▶ Confrontiamo il valore di una soluzione euristica con il valore ottimo
- ▶ Supponiamo una volta per tutte che gli oggetti siano ordinati per efficienza

# Oggetti, profitti e pesi

- ▶ Come nella pagina del problema,  $c$  è la capacità dello zaino e  $\mathcal{I} = \{1, 2, \dots, n\}$  l'insieme degli oggetti.
- ▶ In questo capitolo l'oggetto  $i \in \mathcal{I}$  ha profitto  $p_i$  e peso  $w_i$ .

## Valore ottimo e valore euristico

- ▶ Indichiamo con  $z(I)$  il **valore ottimo** del KP sull'istanza  $I$ .
- ▶ Indichiamo con  $\mathcal{A}(I)$  il profitto della soluzione ammissibile restituita da un algoritmo euristico  $\mathcal{A}$ .
- ▶ Poiché ogni algoritmo euristico di questo capitolo restituisce una soluzione ammissibile,

$$\mathcal{A}(I) \leq z(I) \quad \text{e} \quad \frac{z(I)}{\mathcal{A}(I)} \geq 1$$

per ogni istanza  $I$  del KP.

# Oggetti ordinati per efficienza

- Supponiamo sempre che gli oggetti siano ordinati per efficienza non crescente

$$\frac{p_i}{w_i} \geq \frac{p_{i+1}}{w_{i+1}} \quad \text{per ogni} \quad i \in \{1, 2, \dots, n-1\}.$$

- L'ordinamento richiede tempo  $O(n \log n)$ , una funzione polinomiale del numero  $n$  di oggetti.

# Riepilogo

## Notazione:

- ▶  $c$  capacità,  $\mathcal{I} = \{1, 2, \dots, n\}$  oggetti,  $p_i$  profitti,  $w_i$  pesi
- ▶  $z(I)$  valore ottimo,  $\mathcal{A}(I) \leq z(I)$  valore di un algoritmo euristico  $\mathcal{A}$

## Ipotesi:

- ▶ gli oggetti sono ordinati per efficienza  $\frac{p_i}{w_i}$  non crescente, in tempo  $O(n \log n)$

# Oggetto critico e capacità residua

# Introduzione: Oggetto critico e capacità residua

- ▶ L'oggetto critico è il primo oggetto che non entra quando prendiamo gli oggetti in ordine di efficienza
- ▶ Compare in tutti gli algoritmi e in tutte le dimostrazioni di questo capitolo
- ▶ Lo definiamo, insieme alla capacità residua, e li calcoliamo entrambi su un esempio

## Definizione 1: oggetto critico e capacità residua

Dato un insieme  $\mathcal{I} = \{1, 2, \dots, n\}$  di  $n$  oggetti ordinati per efficienza non crescente, l'**oggetto critico** è l'oggetto

$$s \in \mathcal{I} \quad \text{tale che} \quad \sum_{i=1}^{s-1} w_i \leq c \quad \text{e} \quad \sum_{i=1}^s w_i > c$$

e la **capacità residua** è

$$rc = c - \sum_{i=1}^{s-1} w_i$$

# Oggetto critico e capacità residua: osservazioni

- ▶ L'oggetto critico è il primo che non entra per intero nello zaino se tutti i precedenti sono selezionati.
- ▶ La capacità residua è la capacità dello zaino che rimane quando tutti gli oggetti che precedono l'oggetto critico sono selezionati.
- ▶ Ordinare gli oggetti per efficienza non crescente richiede tempo  $O(n \log n)$ , e l'oggetto critico si trova poi scorrendoli.
- ▶ L'oggetto critico si può trovare anche in tempo  $O(n)$ , senza ordinare, con una ricerca della mediana pesata: l'ordinamento è comodo, non necessario.

## Esempio (1/2)

Consideriamo uno zaino di capacità  $c = 9$  e un insieme  $\mathcal{I} = \{1, 2, 3, 4, 5, 6, 7\}$  di 7 oggetti ( $n = 7$ ), con i seguenti profitti e pesi:

|       | $i = 1$ | $i = 2$ | $i = 3$ | $i = 4$ | $i = 5$ | $i = 6$ | $i = 7$ |
|-------|---------|---------|---------|---------|---------|---------|---------|
| $p_i$ | 6       | 5       | 8       | 9       | 6       | 7       | 3       |
| $w_i$ | 2       | 3       | 6       | 7       | 5       | 9       | 4       |

Le efficienze degli oggetti sono:

$$\frac{p_1}{w_1} = \frac{6}{2}, \quad \frac{p_2}{w_2} = \frac{5}{3}, \quad \frac{p_3}{w_3} = \frac{8}{6}, \quad \frac{p_4}{w_4} = \frac{9}{7},$$

$$\frac{p_5}{w_5} = \frac{6}{5}, \quad \frac{p_6}{w_6} = \frac{7}{9}, \quad \frac{p_7}{w_7} = \frac{3}{4}.$$

## Esempio (2/2)

- ▶ Gli oggetti sono ordinati per efficienza non crescente:  
(1, 2, 3, 4, 5, 6, 7).
- ▶ L'oggetto critico è  $s = 3$  poiché:

$$\sum_{i=1}^2 w_i = 2 + 3 = 5 \leq 9 \quad \text{e} \quad \sum_{i=1}^3 w_i = 2 + 3 + 6 = 11 > 9$$

- ▶ La capacità residua è:

$$rc = c - \sum_{i=1}^2 w_i = 9 - 2 - 3 = 4$$

Questa istanza torna più volte nel capitolo.

## Oggetto critico:

- ▶ il primo oggetto  $s$  che non entra quando tutti i precedenti sono selezionati
- ▶ si trova in tempo  $O(n \log n)$  ordinando, oppure in tempo  $O(n)$  con una ricerca della mediana pesata

## Capacità residua:

- ▶  $rc = c - \sum_{i=1}^{s-1} w_i$ , la capacità lasciata dagli oggetti  $1, 2, \dots, s-1$
- ▶ nell'esempio:  $s = 3$  e  $rc = 4$

# L'algoritmo greedy

# Introduzione: L'algoritmo greedy

- ▶ L'algoritmo greedy è l'euristica costruttiva più naturale per il KP
- ▶ È veloce e semplice, ed è il mattone di tutti gli altri algoritmi del capitolo
- ▶ Scriviamo il suo pseudocodice e lo eseguiamo su quattro istanze: sulle ultime è lontano dall'ottimo

## L'algoritmo greedy *GR*

- ▶ Supponiamo che gli oggetti siano ordinati per efficienza non crescente

$$\frac{p_i}{w_i} \geq \frac{p_{i+1}}{w_{i+1}} \quad \text{per ogni } i \in \{1, 2, \dots, n-1\}$$

- ▶ L'algoritmo *greedy* (indicato con *GR*) è un'euristica costruttiva per il KP.
- ▶ Costruisce una soluzione ammissibile esaminando gli oggetti in ordine di *efficienza* non crescente, dove l'efficienza è il rapporto fra profitto e peso.
- ▶ Partendo da un sottoinsieme vuoto di oggetti, ogni oggetto viene aggiunto alla soluzione se e solo se il vincolo di capacità non viene violato.

---

### Algorithm: Greedy (1/2)

---

**Input** : istanza  $I$  del KP con  $n$  oggetti, profitti  $p[1..n]$ , pesi  $w[1..n]$ , capacità  $c$

**Output** : soluzione ammissibile  $x[1..n]$  del KP

```
1 function Greedy( $n, p, w, c$ )
2    $x :=$  nuovo vettore[1.. $n$ ]           // oggetti selezionati
3   for  $i := 1$  to  $n$  do
4      $x[i] \leftarrow 0$                    // all'inizio l'oggetto  $i$  non è
      selezionato
5   end
6    $rc := c$                              // capacità residua
```

---

---

## Algorithm: Greedy (2/2)

---

**function** *Greedy*( $n, p, w, c$ ) (continua)

```
7   |   for  $i := 1$  to  $n$  do
8   |       if  $w[i] \leq rc$  then
9   |            $x[i] \leftarrow 1$            // selezionare l'oggetto  $i$ 
10  |            $rc \leftarrow rc - w[i]$       // aggiornare la capacità
      |           residua
11  |       end
12  |   end
13  |   return  $x$ 
14 end
```

---

## GR e l'oggetto critico

- Dato l'oggetto critico  $s$ ,  $GR$  seleziona tutti gli oggetti che precedono l'oggetto critico, cioè

$$\{1, 2, \dots, s-1\} \subseteq S \quad \text{e} \quad GR(I) \geq \sum_{i=1}^{s-1} p_i.$$

- Inoltre, l'oggetto  $s$  non viene selezionato da  $GR$ .
- $GR$  può selezionare anche alcuni oggetti che seguono l'oggetto critico.

Vediamo ora quattro esecuzioni di  $GR$ , su quattro istanze.

## Esempio: esecuzione di $GR$ (1/3)

Per l'istanza del KP dell'esempio dell'oggetto critico ( $c = 9$ ,  $n = 7$ ), gli oggetti sono ordinati per efficienza non crescente:  $(1, 2, 3, 4, 5, 6, 7)$ .

|       | $i = 1$ | $i = 2$ | $i = 3$ | $i = 4$ | $i = 5$ | $i = 6$ | $i = 7$ |
|-------|---------|---------|---------|---------|---------|---------|---------|
| $p_i$ | 6       | 5       | 8       | 9       | 6       | 7       | 3       |
| $w_i$ | 2       | 3       | 6       | 7       | 5       | 9       | 4       |

L'esecuzione di  $GR$  procede come segue:

1. L'oggetto 1 (peso  $w_1 = 2$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $9 - 2 = 7$ .
2. L'oggetto 2 (peso  $w_2 = 3$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $7 - 3 = 4$ .

## Esempio: esecuzione di *GR* (2/3)

3. L'oggetto 3 (peso  $w_3 = 6$ ) non viene selezionato perché non entra nella capacità disponibile  $rc = 4$ .
4. L'oggetto 4 (peso  $w_4 = 7$ ) non viene selezionato perché non entra nella capacità disponibile  $rc = 4$ .
5. L'oggetto 5 (peso  $w_5 = 5$ ) non viene selezionato perché non entra nella capacità disponibile  $rc = 4$ .
6. L'oggetto 6 (peso  $w_6 = 9$ ) non viene selezionato perché non entra nella capacità disponibile  $rc = 4$ .
7. L'oggetto 7 (peso  $w_7 = 4$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $4 - 4 = 0$ .

## Esempio: esecuzione di $GR$ (3/3)

Quindi  $GR$  restituisce la soluzione ammissibile  $S = \{1, 2, 7\}$ , di profitto totale

$$GR(I) = \sum_{i \in S} p_i = p_1 + p_2 + p_7 = 6 + 5 + 3 = 14.$$

Pertanto, 14 è un lower bound valido sul valore ottimo  $z(I)$ :

$$14 \leq z(I).$$

- ▶ In questa istanza l'oggetto critico è  $s = 3$ .
- ▶  $GR$  seleziona tutti gli oggetti che precedono l'oggetto critico, cioè gli oggetti 1 e 2, non seleziona l'oggetto 3 (l'oggetto critico), e poi seleziona anche l'oggetto 7, che segue l'oggetto critico.

## Esempio: esecuzione di $GR$ (1/4)

Consideriamo uno zaino di capacità  $c = 10$  e un insieme  $\mathcal{I} = \{1, 2, 3, 4\}$  di 4 oggetti ( $n = 4$ ), con i seguenti profitti e pesi:

|       | $i = 1$ | $i = 2$ | $i = 3$ | $i = 4$ |
|-------|---------|---------|---------|---------|
| $p_i$ | 6       | 8       | 19      | 4       |
| $w_i$ | 2       | 3       | 9       | 5       |

Le efficienze degli oggetti sono:

$$\frac{p_1}{w_1} = \frac{6}{2}, \quad \frac{p_2}{w_2} = \frac{8}{3}, \quad \frac{p_3}{w_3} = \frac{19}{9}, \quad \frac{p_4}{w_4} = \frac{4}{5}.$$

Quindi gli oggetti sono ordinati per efficienza non crescente: (1, 2, 3, 4).

## Esempio: esecuzione di $GR$ (2/4)

L'oggetto critico è  $s = 3$  poiché:

$$\sum_{i=1}^2 w_i = 2 + 3 = 5 \leq 10 \quad \text{e} \quad \sum_{i=1}^3 w_i = 2 + 3 + 9 = 14 > 10$$

e la capacità residua è:

$$rc = c - \sum_{i=1}^2 w_i = 10 - 2 - 3 = 5.$$

## Esempio: esecuzione di $GR$ (3/4)

L'esecuzione di  $GR$  procede come segue:

1. L'oggetto 1 (peso  $w_1 = 2$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $10 - 2 = 8$ .
2. L'oggetto 2 (peso  $w_2 = 3$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $8 - 3 = 5$ .
3. L'oggetto 3 (peso  $w_3 = 9$ ) non viene selezionato perché non entra nella capacità disponibile  $rc = 5$ .
4. L'oggetto 4 (peso  $w_4 = 5$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $5 - 5 = 0$ .

## Esempio: esecuzione di $GR$ (4/4)

Quindi  $GR$  restituisce la soluzione ammissibile  $S = \{1, 2, 4\}$ , di profitto totale

$$GR(I) = \sum_{i \in S} p_i = p_1 + p_2 + p_4 = 6 + 8 + 4 = 18.$$

Pertanto, 18 è un lower bound valido sul valore ottimo  $z(I)$ :  
 $18 \leq z(I)$ .

- ▶ In questa istanza l'oggetto critico è  $s = 3$ :  $GR$  seleziona gli oggetti 1 e 2, che lo precedono, non seleziona l'oggetto 3 (l'oggetto critico), e poi seleziona l'oggetto 4, che lo segue.
- ▶ L'oggetto 3 ha il profitto massimo ( $p_3 = 19$ ) e selezionarlo da solo è ottimo, ma  $GR$  non lo seleziona.

## Esempio: esecuzione di GR (1/4)

Consideriamo uno zaino di capacità  $c = 10$  e un insieme  $\mathcal{I} = \{1, 2, 3, 4, 5\}$  di 5 oggetti ( $n = 5$ ), con i seguenti profitti e pesi:

|       | $i = 1$ | $i = 2$ | $i = 3$ | $i = 4$ | $i = 5$ |
|-------|---------|---------|---------|---------|---------|
| $p_i$ | 4       | 6       | 5       | 12      | 3       |
| $w_i$ | 1       | 2       | 2       | 6       | 2       |

Le efficienze degli oggetti sono:

$$\frac{p_1}{w_1} = \frac{4}{1} = 4, \quad \frac{p_2}{w_2} = \frac{6}{2} = 3, \quad \frac{p_3}{w_3} = \frac{5}{2} = 2.5,$$

$$\frac{p_4}{w_4} = \frac{12}{6} = 2, \quad \frac{p_5}{w_5} = \frac{3}{2} = 1.5.$$

Quindi gli oggetti sono ordinati per efficienza non crescente:  
(1, 2, 3, 4, 5).

## Esempio: esecuzione di $GR$ (2/4)

L'oggetto critico è  $s = 4$  poiché:

$$\sum_{i=1}^3 w_i = 1 + 2 + 2 = 5 \leq 10 \quad \text{e} \quad \sum_{i=1}^4 w_i = 1 + 2 + 2 + 6 = 11 > 10$$

e la capacità residua è:

$$rc = c - \sum_{i=1}^3 w_i = 10 - 5 = 5.$$

## Esempio: esecuzione di $GR$ (3/4)

L'esecuzione di  $GR$  procede come segue:

1. L'oggetto 1 (peso  $w_1 = 1$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $10 - 1 = 9$ .
2. L'oggetto 2 (peso  $w_2 = 2$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $9 - 2 = 7$ .
3. L'oggetto 3 (peso  $w_3 = 2$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $7 - 2 = 5$ .
4. L'oggetto 4 (peso  $w_4 = 6$ ) non viene selezionato perché non entra nella capacità disponibile  $rc = 5$ .
5. L'oggetto 5 (peso  $w_5 = 2$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $5 - 2 = 3$ .

## Esempio: esecuzione di $GR$ (4/4)

Quindi  $GR$  restituisce la soluzione ammissibile  $S = \{1, 2, 3, 5\}$ , di profitto totale

$$GR(I) = \sum_{i \in S} p_i = p_1 + p_2 + p_3 + p_5 = 4 + 6 + 5 + 3 = 18.$$

Pertanto, 18 è un lower bound valido sul valore ottimo  $z(I)$ :  
 $18 \leq z(I)$ .

- ▶ In questa istanza l'oggetto critico è  $s = 4$ :  $GR$  seleziona gli oggetti 1, 2 e 3 (che precedono l'oggetto critico), salta l'oggetto 4 (l'oggetto critico) e poi seleziona l'oggetto 5.
- ▶ La soluzione ottima è  $\tilde{S} = \{2, 3, 4\}$  di valore  $z(I) = 6 + 5 + 12 = 23$ , migliore della soluzione trovata da  $GR$ .

## Esempio: esecuzione di GR (1/5)

Consideriamo uno zaino di capacità  $c = 21$  e un insieme  $\mathcal{I} = \{1, 2, 3, 4, 5, 6\}$  di 6 oggetti ( $n = 6$ ), con i seguenti profitti e pesi:

|       | $i = 1$ | $i = 2$ | $i = 3$ | $i = 4$ | $i = 5$ | $i = 6$ |
|-------|---------|---------|---------|---------|---------|---------|
| $p_i$ | 4       | 6       | 4       | 3       | 2       | 20      |
| $w_i$ | 1       | 2       | 2       | 2       | 2       | 21      |

Le efficienze degli oggetti sono:

$$\frac{p_1}{w_1} = \frac{4}{1} = 4, \quad \frac{p_2}{w_2} = \frac{6}{2} = 3, \quad \frac{p_3}{w_3} = \frac{4}{2} = 2,$$

$$\frac{p_4}{w_4} = \frac{3}{2} = 1.5, \quad \frac{p_5}{w_5} = \frac{2}{2} = 1, \quad \frac{p_6}{w_6} = \frac{20}{21}.$$

Quindi gli oggetti sono ordinati per efficienza non crescente:  
(1, 2, 3, 4, 5, 6).

## Esempio: esecuzione di $GR$ (2/5)

L'oggetto critico è  $s = 6$  poiché:

$$\sum_{i=1}^5 w_i = 1 + 2 + 2 + 2 + 2 = 9 \leq 21$$

$$\text{e } \sum_{i=1}^6 w_i = 1 + 2 + 2 + 2 + 2 + 21 = 30 > 21$$

e la capacità residua è:

$$rc = c - \sum_{i=1}^5 w_i = 21 - 9 = 12.$$

## Esempio: esecuzione di $GR$ (3/5)

L'esecuzione di  $GR$  procede come segue:

1. L'oggetto 1 (peso  $w_1 = 1$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $21 - 1 = 20$ .
2. L'oggetto 2 (peso  $w_2 = 2$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $20 - 2 = 18$ .
3. L'oggetto 3 (peso  $w_3 = 2$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $18 - 2 = 16$ .

## Esempio: esecuzione di $GR$ (4/5)

4. L'oggetto 4 (peso  $w_4 = 2$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $16 - 2 = 14$ .
5. L'oggetto 5 (peso  $w_5 = 2$ ) viene selezionato. La capacità disponibile  $rc$  diventa  $14 - 2 = 12$ .
6. L'oggetto 6 (peso  $w_6 = 21$ ) non viene selezionato perché non entra nella capacità disponibile  $rc = 12$ .

Quindi  $GR$  restituisce la soluzione ammissibile  $S = \{1, 2, 3, 4, 5\}$ , di profitto totale

$$GR(I) = \sum_{i \in S} p_i = p_1 + p_2 + p_3 + p_4 + p_5 = 4 + 6 + 4 + 3 + 2 = 19.$$

Pertanto, 19 è un lower bound valido sul valore ottimo  $z(I)$ :  
 $19 \leq z(I)$ .

## Esempio: esecuzione di *GR* (5/5)

- ▶ L'oggetto di profitto massimo è l'oggetto 6, con  $p_6 = 20$ , che compare per ultimo nell'ordine per efficienza.
- ▶ *GR* seleziona tutti gli oggetti da 1 a 5 (che precedono l'oggetto critico), ma non seleziona l'oggetto 6 (l'oggetto critico stesso) perché il suo peso  $w_6 = 21$  supera la capacità residua  $rc = 12$ .
- ▶ Da solo l'oggetto 6 entrerebbe nello zaino (poiché  $w_6 = 21 \leq c = 21$ ) e darebbe un profitto di 20, migliore della soluzione greedy.
- ▶ La soluzione ottima è  $\tilde{S} = \{6\}$  di valore  $z(I) = 20$ .

## L'algoritmo greedy $GR$ :

- ▶ esamina gli oggetti per efficienza non crescente e aggiunge ogni oggetto che entra
- ▶ seleziona tutti gli oggetti  $1, 2, \dots, s-1$  che precedono l'oggetto critico, quindi  $GR(I) \geq \sum_{i=1}^{s-1} p_i$
- ▶ salta l'oggetto critico  $s$ , ma continua a scorrere gli oggetti e può selezionare oggetti che lo seguono

## Il suo difetto:

- ▶ può perdere l'oggetto di profitto massimo, anche quando quel solo oggetto è ottimo

# L'algoritmo Ext-Greedy

# Introduzione: L'algoritmo Ext-Greedy

- ▶ Gli esempi mostrano il difetto di *GR*: può perdere l'oggetto di profitto massimo
- ▶ Ext-Greedy lo corregge quasi senza costo, confrontando la soluzione greedy con il miglior oggetto singolo
- ▶ Questa piccola modifica dà una garanzia di prestazione, come dimostriamo più avanti nel capitolo

## L'algoritmo Ext-Greedy $GR^e$

- ▶ L'algoritmo *Ext-Greedy* (indicato con  $GR^e$ ) è un'euristica costruttiva per il KP.
- ▶ Prima calcola una soluzione ammissibile eseguendo l'algoritmo greedy  $GR$ .
- ▶ Poi confronta questa soluzione con la migliore soluzione ammissibile formata da un solo oggetto.
- ▶ Infine restituisce la migliore delle due soluzioni.

---

### Algorithm: Extended Greedy (1/3)

---

**Input** : istanza  $I$  del KP con  $n$  oggetti, profitti  $p[1..n]$ , pesi  $w[1..n]$ , capacità  $c$

**Output** : soluzione ammissibile  $x[1..n]$  del KP

```
1 function ExtendedGreedy( $n, p, w, c$ )
2    $x := \text{Greedy}(n, p, w, c)$            // soluzione greedy
3    $z := 0$                              // valore della soluzione greedy
4   for  $i := 1$  to  $n$  do
5     if  $x[i] = 1$  then
6        $z \leftarrow z + p[i]$  // sommare il profitto di un
        oggetto selezionato
7     end
8   end
```

---

---

## Algorithm: Extended Greedy (2/3)

---

**function** *ExtendedGreedy*( $n, p, w, c$ ) (continua)

```
9   |  $i_m := 1$                                 // miglior oggetto singolo
10  |  $p_{\max} := p[1]$ 
11  | for  $i := 2$  to  $n$  do
12  |   | if  $p[i] > p_{\max}$  then
13  |   |   |  $i_m \leftarrow i$  // aggiornare il miglior oggetto
14  |   |   |   | singolo
15  |   |   |   |  $p_{\max} \leftarrow p[i]$  // aggiornare il suo profitto
16  |   | end
    | end
```

---

---

## Algorithm: Extended Greedy (3/3)

---

**function** *ExtendedGreedy*( $n, p, w, c$ ) (continua)

```
17    $\bar{x} :=$  nuovo vettore[1.. $n$ ]  // migliore soluzione con
    un solo oggetto
18   for  $i := 1$  to  $n$  do
19      $\bar{x}[i] \leftarrow 0$ 
20   end
21    $\bar{x}[i_m] \leftarrow 1$           // selezionare solo l'oggetto  $i_m$ 
22   if  $z < p_{\max}$  then
23     for  $i := 1$  to  $n$  do
24        $x[i] \leftarrow \bar{x}[i]$       // copiare componente per
       componente
25     end
26   end
27   return  $x$ 
28 end
```

---

# Profitto massimo e miglior oggetto singolo

- Sia  $p_{\max}$  il **profitto massimo**

$$p_{\max} = \max \{ p_i : i \in \mathcal{I} \}.$$

- Sia  $\bar{S}$  la **migliore soluzione ammissibile con un solo oggetto**

$$\bar{S} = \{ i_m \} \text{ e } i_m \in \arg \max \{ p_i : i \in \mathcal{I} \}$$

## Esempio: esecuzione di $GR^e$

Partiamo dalla soluzione ammissibile calcolata da  $GR$  sull'istanza del KP dell'esempio dell'oggetto critico ( $c = 9$ ,  $n = 7$ ):

$$S = \{1, 2, 7\}, \quad GR(I) = \sum_{i \in S} p_i = p_1 + p_2 + p_7 = 6 + 5 + 3 = 14.$$

Calcoliamo ora il profitto massimo di un singolo oggetto:

$$p_{\max} = \max \{ p_i : i \in \mathcal{I} \} = p_4 = 9, \quad \bar{S} = \{4\}.$$

Poiché  $14 \geq 9$ ,  $GR^e$  mantiene la soluzione greedy e restituisce

$$S = \{1, 2, 7\}.$$

## Esempio: esecuzione di $GR^e$

Partiamo dalla soluzione ammissibile calcolata da  $GR$  sull'istanza del KP con  $c = 10$  e 4 oggetti, il cui oggetto 3 ha profitto 19:

$$S = \{1, 2, 4\}, \quad GR(I) = \sum_{i \in S} p_i = p_1 + p_2 + p_4 = 6 + 8 + 4 = 18.$$

Calcoliamo ora il profitto massimo di un singolo oggetto:

$$p_{\max} = \max \{ p_i : i \in \mathcal{I} \} = p_3 = 19, \quad \bar{S} = \{3\}.$$

Poiché  $18 < 19$ ,  $GR^e$  sostituisce la soluzione greedy con la migliore soluzione ammissibile con un solo oggetto e restituisce

$$S = \{3\}.$$

## Due bound su $GR^e(I)$

- ▶ Dato l'oggetto critico  $s$ , abbiamo

$$GR^e(I) \geq GR(I) \geq \sum_{i=1}^{s-1} p_i.$$

- ▶ Inoltre, per definizione di  $GR^e$ , abbiamo

$$GR^e(I) \geq p_{\max}.$$

- ▶ Per ogni istanza  $I$  del KP, i valori obiettivo soddisfano

$$GR(I) \leq GR^e(I) \leq z(I).$$

- ▶ Questo vale perché  $GR^e$  restituisce la migliore fra la soluzione greedy e la migliore soluzione ammissibile con un solo oggetto, quindi non può essere peggiore di  $GR$ , e il valore di ogni soluzione ammissibile è al più il valore ottimo  $z(I)$ .

## L'algoritmo Ext-Greedy $GR^e$ :

- ▶ esegue  $GR$ , poi confronta la sua soluzione con il miglior oggetto singolo  $\bar{S} = \{i_m\}$ , di profitto  $p_{\max}$
- ▶ restituisce la migliore delle due soluzioni

## Due bound usati più avanti:

- ▶  $GR^e(I) \geq \sum_{i=1}^{s-1} p_i$  e  $GR^e(I) \geq p_{\max}$
- ▶  $GR(I) \leq GR^e(I) \leq z(I)$

# L'algoritmo $H^2$

# Introduzione: L'algoritmo $H^2$

- ▶  $GR^e$  può ancora perdere molto, quando non riesce a inserire insieme gli oggetti di profitto maggiore
- ▶ L'algoritmo  $H^2$  “scommette” su due oggetti di una soluzione ottima, provando tutte le coppie ammissibili, e completa ogni scommessa con  $GR^e$
- ▶ Paga più tempo di calcolo, e vedremo che guadagna una garanzia migliore

## L'algoritmo $H^2$

- ▶ L'algoritmo  $H^2$  è un'euristica costruttiva per il KP; è l'algoritmo chiamato  $G^{3/4}$  in letteratura.
- ▶ Enumera tutte le coppie ammissibili di oggetti  $(i, k)$  e “scommette” che i due oggetti appartengano a una soluzione ottima.
- ▶ Per ogni coppia, fissa  $i$  e  $k$  nello zaino, costruisce un'istanza residua con la capacità rimanente e tenendo solo gli oggetti di profitto al più  $\min\{p_i, p_k\}$ , e poi completa la soluzione parziale eseguendo l'algoritmo Ext-Greedy  $GR^e$  sull'istanza residua.
- ▶ Infine restituisce la migliore soluzione trovata su tutte le coppie, confrontandola anche con la migliore soluzione ammissibile con un solo oggetto.

---

### Algorithm: $H^2$ (1/2)

---

**Input** : istanza  $I$  del KP con insieme di oggetti  
 $\mathcal{I} = \{1, 2, \dots, n\}$ , profitti  $p[1..n]$ , pesi  $w[1..n]$ ,  
capacità  $c$

**Output** : valore ammissibile  $H^2(I)$  del KP

```
1 function  $H2(\mathcal{I}, p, w, c)$   
2    $p_{\max} := \max\{p[j] : j \in \mathcal{I}\}$   
3    $lb := p_{\max}$  // miglior valore trovato  
4    $\mathcal{P} := \{(i, k) \in \mathcal{I} \times \mathcal{I} : i < k, w[i] + w[k] \leq c\}$ 
```

---

---

## Algorithm: $H^2$ (2/2)

---

**function**  $H2(\mathcal{I}, p, w, c)$  (continua)

```
5   foreach  $(i, k)$  in  $\mathcal{P}$  do
6        $rc := c - w[i] - w[k]$            // capacità residua
7        $\mathcal{J}(i, k) := \{j \in \mathcal{I} \setminus \{i, k\} : p[j] \leq$ 
            $\min\{p[i], p[k]\}, w[j] \leq rc\}$ 
8        $l_{ik} :=$ 
           istanza del KP ristretta a  $\mathcal{J}(i, k)$  con capacità  $rc$ 
9        $z_{res} := GR^e(l_{ik})$  // valore di  $GR^e$  sull'istanza
           residua
10       $lb_{ik} := p[i] + p[k] + z_{res}$ 
11      if  $lb_{ik} > lb$  then
12           $lb \leftarrow lb_{ik}$ 
13      end
14  end
15  return  $lb$ 
16 end
```

---

## Definizione 2: coppie ordinate ammissibili di oggetti

*Data un'istanza del KP con insieme di oggetti  $\mathcal{I} = \{1, 2, \dots, n\}$  e capacità dello zaino  $c$ , l'insieme di tutte le coppie ordinate ammissibili di oggetti è*

$$\mathcal{P} = \left\{ (i, k) \in \mathcal{I} \times \mathcal{I} : i < k, \quad w_i + w_k \leq c \right\}.$$

- ▶ L'algoritmo  $H^2$  si basa sulla seguente idea: una soluzione ottima o è formata da un solo oggetto, quello di profitto massimo, o contiene (almeno) due oggetti di profitto “grande”.
- ▶ L'algoritmo quindi *scommette* sui due oggetti di profitto maggiore di una soluzione ottima, enumerando tutte le coppie ammissibili in  $\mathcal{P}$ , e completa ogni scommessa su un'istanza residua.

## Le tre operazioni di $H^2$ (1/2)

L'algoritmo  $H^2$  considera le coppie di oggetti  $(i, k) \in \mathcal{P}$ , una alla volta, ed esegue tre operazioni:

1. La **prima operazione** consiste nell'inserire gli oggetti  $i$  e  $k$  nello zaino. Si ottiene così una **soluzione parziale** di profitto  $p_i + p_k$  e una **capacità residua**  $c - w_i - w_k$ .
2. La **seconda operazione** è la costruzione di un'istanza **residua** (indicata con  $I(i, k)$ ) che contiene solo gli oggetti di profitto minore o uguale ai profitti dei due oggetti inseriti. L'**insieme residuo di oggetti** è:

$$\mathcal{I}(i, k) = \left\{ j \in \mathcal{J} \setminus \{i, k\} : \begin{array}{l} p_j \leq \min \{p_i, p_k\}, \\ w_j \leq c - w_i - w_k \end{array} \right\}$$

Vengono scartati anche gli oggetti di peso maggiore della capacità residua.

## Le tre operazioni di $H^2$ (2/2)

3. La **terza operazione** consiste nel completare la soluzione parziale eseguendo l'algoritmo  $GR^e$  sull'istanza residua  $I(i, k)$ .

- Per ogni coppia di oggetti  $(i, k) \in \mathcal{P}$ , abbiamo il lower bound:

$$lb(i, k) = p_i + p_k + GR^e(I(i, k))$$

- Il lower bound globale prodotto dall'algoritmo  $H^2$  è:

$$H^2(I) = \max \left\{ \max_{(i,k) \in \mathcal{P}} \{ lb(i, k) \}, p_{\max} \right\}$$

# Una scommessa

Questo algoritmo euristico si può vedere come una  
“scommessa”  
sui due oggetti di profitto maggiore di una soluzione ottima  
(o sull'oggetto di profitto maggiore).

## Esempio: algoritmo $H^2$ (1/4)

Consideriamo un insieme  $\mathcal{J} = \{1, 2, 3, 4, 5, 6, 7\}$  di 7 oggetti ( $n = 7$ ) con i seguenti profitti e pesi:

| oggetto | $p_j$ | $w_j$ | $\frac{p_j}{w_j}$ | efficienza     |
|---------|-------|-------|-------------------|----------------|
| $j = 1$ | 6     | 2     | $\frac{6}{2}$     | $= 3$          |
| $j = 2$ | 5     | 3     | $\frac{5}{3}$     | $\approx 1.66$ |
| $j = 3$ | 8     | 6     | $\frac{8}{6}$     | $\approx 1.33$ |
| $j = 4$ | 9     | 7     | $\frac{9}{7}$     | $\approx 1.28$ |
| $j = 5$ | 6     | 5     | $\frac{6}{5}$     | $= 1.2$        |
| $j = 6$ | 7     | 9     | $\frac{7}{9}$     | $\approx 0.77$ |
| $j = 7$ | 3     | 4     | $\frac{3}{4}$     | $= 0.75$       |

e una capacità dello zaino  $c = 12$ , per cui  $p_{\max} = 9$ .

## Esempio: algoritmo $H^2$ (2/4)

- ▶ La soluzione ottima è  $\tilde{S} = \{1, 2, 4\}$ , di peso  $2 + 3 + 7 = 12$  e valore  $z(I) = 6 + 5 + 9 = 20$ .
- ▶ L'oggetto critico è  $s = 4$ , e  $GR^e$  restituisce la soluzione  $\{1, 2, 3\}$  con  $GR^e(I) = 6 + 5 + 8 = 19$ .

Vediamo ora l'esecuzione di  $H^2$ , con inizializzazione  $lb = p_{\max} = 9$ .

## Esempio: algoritmo $H^2$ (3/4)

**Esecuzione di  $H^2$**  (inizializzazione  $lb = p_{\max} = 9$ ):

- ▶ Coppia (1, 2): la capacità residua è  $rc = 12 - 2 - 3 = 7$ ,  $\mathcal{I}(1, 2) = \{7\}$  e  $GR^e(I(1, 2)) = 3$ , quindi  $lb(1, 2) = 6 + 5 + 3 = 14 > lb$  e  $lb$  viene aggiornato a 14.
- ▶ Coppia (1, 3): la capacità residua è  $rc = 12 - 2 - 6 = 4$ ,  $\mathcal{I}(1, 3) = \{2, 7\}$  e  $GR^e(I(1, 3)) = 5$ , quindi  $lb(1, 3) = 6 + 8 + 5 = 19 > lb$  e  $lb$  viene aggiornato a 19.
- ▶ Coppia (1, 4): la capacità residua è  $rc = 12 - 2 - 7 = 3$ ,  $\mathcal{I}(1, 4) = \{2\}$  e  $GR^e(I(1, 4)) = 5$ , quindi  $lb(1, 4) = 6 + 9 + 5 = 20 > lb$  e  $lb$  viene aggiornato a 20.

## Esempio: algoritmo $H^2$ (4/4)

- ▶ Coppia (1, 5): la capacità residua è  $rc = 12 - 2 - 5 = 5$ ,  $\mathcal{I}(1, 5) = \{2, 7\}$  e  $GR^e(I(1, 5)) = 5$ , quindi  $lb(1, 5) = 6 + 6 + 5 = 17 < lb$ .
- ▶ Coppia (1, 6): la capacità residua è  $rc = 12 - 2 - 9 = 1$ ,  $\mathcal{I}(1, 6) = \emptyset$ , quindi  $lb(1, 6) = 6 + 7 = 13 < lb$ .
- ▶ Coppia (1, 7): la capacità residua è  $rc = 12 - 2 - 4 = 6$ ,  $\mathcal{I}(1, 7) = \emptyset$ , quindi  $lb(1, 7) = 6 + 3 = 9 < lb$ .
- ▶ Le coppie  $(i, k)$  con  $i \geq 2$  si trattano allo stesso modo, e nessuna dà un valore maggiore di 20.

Quindi  $H^2(I) = 20 = z(I)$ : su questa istanza  $H^2$  trova la soluzione ottima, che  $GR^e$  non trova.

# Il tempo di calcolo di $H^2$

## Proposizione 1

*L'algoritmo  $H^2$  si può calcolare in tempo  $O(n^3)$ , e in tempo  $O(n^2)$  con un'implementazione più accurata.*

- ▶ Non diamo una dimostrazione formale, ma solo le idee principali.
- ▶ Il tempo di calcolo dell'algoritmo  $H^2$  è determinato essenzialmente dall'esecuzione dell'algoritmo  $GR^e$ , che richiede tempo  $O(n)$ , per ogni coppia di oggetti.
- ▶ Il numero di esecuzioni di  $GR^e$  è  $\binom{n}{2}$ , che è in  $O(n^2)$ , quindi il tempo di calcolo totale è  $O(n^3)$ .
- ▶ Con un'implementazione più accurata si può ridurre a  $O(n^2)$ .

## L'algoritmo $H^2$ :

- ▶ scommette su due oggetti di profitto maggiore di una soluzione ottima, provando ogni coppia  $(i, k) \in \mathcal{P}$
- ▶ per ogni coppia: soluzione parziale, istanza residua  $I(i, k)$  con gli oggetti di profitto al più  $\min\{p_i, p_k\}$ , completamento con  $GR^e$
- ▶  $H^2(I) = \max\{\max_{(i,k) \in \mathcal{P}} lb(i, k), p_{\max}\}$

## Tempo di calcolo:

- ▶  $O(n^2)$  coppie,  $O(n)$  per ogni  $GR^e$ :  $O(n^3)$ , oppure  $O(n^2)$  con un'implementazione accurata

# L'algoritmo $H^\ell$

# Introduzione: L'algoritmo $H^\ell$

- ▶ Se scommettere su due oggetti aiuta, scommettere su più oggetti può aiutare di più
- ▶ L'algoritmo  $H^\ell$  scommette sugli  $\ell$  oggetti di profitto maggiore di una soluzione ottima, per un intero  $\ell$  fissato
- ▶ La garanzia migliora con  $\ell$ , ma il tempo di calcolo cresce molto rapidamente con  $\ell$

# L'algoritmo $H^\ell$

- ▶ L'algoritmo  $H^\ell$  è un'euristica costruttiva per il KP.
- ▶ Prima calcola la migliore soluzione ammissibile formata da al più  $\ell - 1$  oggetti.
- ▶ Poi enumera tutti gli insiemi ammissibili  $L$  di esattamente  $\ell$  oggetti e “scommette” che siano gli  $\ell$  oggetti di profitto maggiore di una soluzione ottima.
- ▶ Per ogni insieme, inserisce gli oggetti di  $L$ , costruisce un'istanza residua con la capacità rimanente e gli oggetti di profitto al più  $\min\{p_i : i \in L\}$ , e completa la soluzione parziale eseguendo l'algoritmo Ext-Greedy  $GR^e$  sull'istanza residua.
- ▶ Infine restituisce la migliore soluzione trovata.

---

**Algorithm:**  $H^\ell$  (1/2)

---

**Input** : istanza  $I$  del KP con insieme di oggetti  
 $\mathcal{I} = \{1, 2, \dots, n\}$ , profitti  $p[1..n]$ , pesi  $w[1..n]$ ,  
capacità  $c$ , intero  $\ell$  con  $2 \leq \ell \leq n$

**Output** : valore ammissibile  $H^\ell(I)$  del KP

```
1 function  $H(I, p, w, c, \ell)$   
2   |  $lb := \max \{ \sum_{j \in S} p[j] : S \in \hat{T}(\ell) \}$            // insiemi  
   | ammissibili di al più  $\ell - 1$  oggetti
```

---

---

## Algorithm: $H^\ell$ (2/2)

---

**function**  $H(\mathcal{I}, p, w, c, \ell)$  (continua)

```
3   foreach  $L$  in  $T(\ell)$            // insiemi ammissibili di  
    esattamente  $\ell$  oggetti  
4   do  
5        $rc := c - \sum_{i \in L} w[i]$            // capacità residua  
6        $\mathcal{J}(L) := \{j \in \mathcal{I} \setminus L : p[j] \leq \min_{i \in L} p[i], w[j] \leq rc\}$   
7        $lb_L := \sum_{i \in L} p[i] + GR^e(I(L))$  // completare la  
        scommessa  
8       if  $lb_L > lb$  then  
9          $lb \leftarrow lb_L$   
10      end  
11  end  
12  return  $lb$   
13 end
```

---

Da  $H^2$  a  $H^\ell$

- ▶ L'algoritmo  $H^2$  scommette sui *due* oggetti di profitto maggiore di una soluzione ottima.
- ▶ L'algoritmo  $H^\ell$  generalizza questa idea: scommette sugli  $\ell$  oggetti di profitto maggiore, per un intero fissato  $\ell$  con  $2 \leq \ell \leq n$ .
- ▶ Per  $\ell = 2$  è esattamente l'algoritmo  $H^2$ .

# Insiemi ammissibili di oggetti

Definiamo le famiglie di tutti gli insiemi ammissibili di esattamente  $\ell$  oggetti e di al più  $\ell - 1$  oggetti:

$$T(\ell) = \left\{ L \subseteq \mathcal{I} : \sum_{j \in L} w_j \leq c, \quad |L| = \ell \right\}$$

e

$$\hat{T}(\ell) = \left\{ L \subseteq \mathcal{I} : \sum_{j \in L} w_j \leq c, \quad |L| \leq \ell - 1 \right\}$$

## Le tre operazioni di $H^\ell$

Per ogni insieme  $L \in T(\ell)$  l'algoritmo esegue tre operazioni:

1. inserisce nello zaino tutti gli oggetti  $i \in L$ , ottenendo una **soluzione parziale** di profitto  $\sum_{i \in L} p_i$  e una **capacità residua**  $c - \sum_{i \in L} w_i$ ;
2. costruisce l'**istanza residua**  $I(L)$ , il cui **insieme residuo di oggetti** è

$$\mathcal{I}(L) = \left\{ j \in \mathcal{I} \setminus L : p_j \leq \min \{ p_i : i \in L \}, w_j \leq c - \sum_{i \in L} w_i \right\};$$

3. completa la soluzione parziale eseguendo  $GR^e$  su  $I(L)$ , ottenendo il valore

$$lb(L) = \sum_{i \in L} p_i + GR^e(I(L)).$$

## Il valore restituito da $H^\ell$

Il valore restituito dall'algoritmo è

$$H^\ell(I) = \max \left\{ \max_{L \in T(\ell)} lb(L), \max_{S \in \hat{T}(\ell)} \sum_{j \in S} p_j \right\}.$$

- ▶ Il primo termine viene dalle scommesse su esattamente  $\ell$  oggetti, ciascuna completata da  $GR^e$ .
- ▶ Il secondo termine è la migliore soluzione ammissibile formata da al più  $\ell - 1$  oggetti.

## Esempio: esecuzione di $H^\ell$ (1/3)

Consideriamo uno zaino di capacità  $c = 12$  e un insieme  $\mathcal{I} = \{1, 2, 3, 4, 5\}$  di 5 oggetti ( $n = 5$ ), con i seguenti profitti e pesi:

|       | $i = 1$ | $i = 2$ | $i = 3$ | $i = 4$ | $i = 5$ |
|-------|---------|---------|---------|---------|---------|
| $p_i$ | 2       | 3       | 3       | 3       | 3       |
| $w_i$ | 1       | 3       | 3       | 3       | 3       |

- ▶ Gli oggetti sono ordinati per efficienza non crescente, poiché  $\frac{2}{1} = 2 > \frac{3}{3} = 1$ .
- ▶ La soluzione ottima inserisce i quattro oggetti di peso 3, con  $z(I) = 12$ .

## Esempio: esecuzione di $H^\ell$ (2/3)

**Con**  $\ell = 2$ , cioè l'algoritmo  $H^2$  della sezione precedente.

- ▶ Una coppia di due oggetti di profitto 3, per esempio  $L = \{2, 3\}$ , lascia capacità residua 6 e l'insieme residuo  $\mathcal{I}(L) = \{1, 4, 5\}$ .
- ▶  $GR^e$  inserisce l'oggetto 1 e l'oggetto 4, di profitto  $2 + 3 = 5$ , quindi  $lb(L) = 3 + 3 + 5 = 11$ .
- ▶ Una coppia che contiene l'oggetto 1 lascia un insieme residuo vuoto, poiché nessun altro oggetto ha profitto al più 2.
- ▶ Quindi  $H^2(I) = 11 < 12$ .

## Esempio: esecuzione di $H^\ell$ (3/3)

Con  $\ell = 3$ .

- ▶ Un insieme di tre oggetti di profitto 3, per esempio  $L = \{2, 3, 4\}$ , lascia capacità residua 3 e l'insieme residuo  $\mathcal{I}(L) = \{1, 5\}$ .
- ▶  $GR^e$  restituisce la migliore fra l'oggetto 1 da solo (profitto 2, dopo il quale l'oggetto 5 non entra più) e l'oggetto 5 da solo (profitto 3), quindi  $lb(L) = 9 + 3 = 12$ .
- ▶ Quindi  $H^3(I) = 12 = z(I)$ .

Scommettere su un oggetto in più basta per trovare la soluzione ottima su questa istanza, sulla quale  $H^2$  non la trova.

# Il tempo di calcolo di $H^\ell$

## Osservazione 1

*L'algoritmo  $H^\ell$  si può calcolare in tempo  $O(n^{\ell+1})$ .*

- ▶ Il tempo di calcolo è determinato dalle esecuzioni di  $GR^e$ , che richiede tempo  $O(n)$ , una per ogni insieme in  $T(\ell)$ .
- ▶ Questi insiemi sono al più  $\binom{n}{\ell}$ , che è in  $O(n^\ell)$ , quindi il tempo di calcolo totale è  $O(n^{\ell+1})$ .
- ▶ Con un'implementazione più accurata si può ridurre a  $O(n^\ell)$ .
- ▶ Oltre alla memorizzazione dei dati, non si usa memoria aggiuntiva significativa, e lo spazio totale richiesto è solo  $O(n)$ .

## L'algoritmo $H^\ell$ :

- ▶ migliore insieme ammissibile di al più  $\ell - 1$  oggetti, poi ogni insieme ammissibile  $L \in T(\ell)$  di esattamente  $\ell$  oggetti, completato da  $GR^e$  sull'istanza residua  $I(L)$
- ▶ per  $\ell = 2$  è l'algoritmo  $H^2$
- ▶ nell'esempio,  $H^2(I) = 11$  e  $H^3(I) = 12 = z(I)$

## Tempo di calcolo e spazio:

- ▶ tempo  $O(n^{\ell+1})$ , oppure  $O(n^\ell)$  con un'implementazione accurata; spazio  $O(n)$

# L'algoritmo Basic-FPTAS

# Introduzione: L'algoritmo Basic-FPTAS

- ▶ L'algoritmo  $H^\ell$  enumera  $O(n^\ell)$  insiemi di oggetti, un numero che cresce molto rapidamente con  $\ell$
- ▶ L'algoritmo Basic-FPTAS segue una strada del tutto diversa: invece di scommettere sugli oggetti, arrotonda i profitti e risolve esattamente l'istanza arrotondata
- ▶ Lo descriviamo, spieghiamo perché si arrotondano solo i profitti, e lo eseguiamo su una piccola istanza

# L'algoritmo Basic-FPTAS $FP^\epsilon$

- ▶ L'algoritmo *Basic-FPTAS* (indicato con  $FP^\epsilon$ ) sostituisce i profitti degli oggetti con profitti scalati, che assumono pochi valori distinti.
- ▶ Poi risolve l'istanza scalata *all'ottimo* con la programmazione dinamica sui profitti.
- ▶ La soluzione ottima dell'istanza scalata viene restituita come soluzione ammissibile dell'istanza originale, in generale non ottima.

---

## Algorithm: Basic-FPTAS

---

**Input** : istanza  $I$  del KP con  $n$  oggetti, profitti  $p[1..n]$ , pesi  $w[1..n]$ , capacità  $c$ , accuratezza  $\varepsilon \in (0, 1)$

**Output** : soluzione ammissibile  $x[1..n]$  del KP

```
1 function BasicFPTAS( $n, p, w, c, \varepsilon$ )
2    $lb := GR^e(I)$  // valore della soluzione
   Ext-Greedy, un lower bound su  $z(I)$ 
3    $K := \varepsilon \cdot lb / n$  // fattore di scala
4   for  $j := 1$  to  $n$  do
5      $p'[j] \leftarrow \lfloor p[j] / K \rfloor$  // profitto scalato
6   end
7    $Q := \lfloor 2n / \varepsilon \rfloor$  // upper bound sul valore ottimo
   dell'istanza scalata
8    $x := DPProfits(n, p', w, c, Q)$  // istanza scalata
   risolta all'ottimo, tabella troncata a  $Q$ 
9   return  $x$ 
10 end
```

---

# Fattore di scala e profitti scalati

- Il **fattore di scala** è

$$K = \frac{\varepsilon GR^e(I)}{n}.$$

- I **profitti scalati** si ottengono arrotondando per difetto i profitti a un multiplo di  $K$ :

$$p'_j = \left\lfloor \frac{p_j}{K} \right\rfloor \quad \text{per ogni} \quad j \in \mathcal{I}.$$

- L'istanza scalata  $I'$  ha gli stessi pesi e la stessa capacità di  $I$ , quindi  $I$  e  $I'$  hanno esattamente le stesse soluzioni ammissibili.

# Si scalano i profitti, non i pesi

- ▶ Arrotondare i pesi per difetto farebbe sembrare ammissibili soluzioni inammissibili.
- ▶ Arrotondarli per eccesso scarterebbe soluzioni ammissibili.
- ▶ In entrambi i casi l'istanza scalata risponderebbe a una domanda diversa.
- ▶ Arrotondare i profitti per difetto, invece, lascia intatto l'insieme delle soluzioni ammissibili e altera solo la funzione obiettivo.

## L'istanza scalata si risolve con la programmazione dinamica sui profitti

- ▶ L'istanza scalata si risolve con il **secondo algoritmo di PD** del capitolo «PD per il problema dello zaino 0/1».
- ▶  $u(j, q)$  è il peso totale minimo di un sottoinsieme di  $\{1, 2, \dots, j\}$  di profitto totale esattamente  $q$ , e il valore ottimo è il massimo  $q$  con  $u(n, q) \leq c$ .
- ▶ La tabella è troncata a  $Q = \lfloor 2n/\varepsilon \rfloor$ . Questo valore non è minore del valore ottimo dell'istanza scalata, che è al più

$$\frac{z(I)}{K} \leq \frac{2 GR^e(I)}{K} = \frac{2n}{\varepsilon}$$

per la garanzia  $z(I) \leq 2 GR^e(I)$  di  $GR^e$ .

- ▶ La tabella ha dimensione  $n \times Q$ , e il tempo di calcolo è  $O(nQ) = O(n^2/\varepsilon)$ .

## Esempio: esecuzione di $FP^\varepsilon$ ( $1/4$ )

Consideriamo uno zaino di capacità  $c = 8$  e un insieme  $\mathcal{I} = \{1, 2, 3, 4\}$  di 4 oggetti ( $n = 4$ ), con i seguenti profitti e pesi:

|       | $i = 1$ | $i = 2$ | $i = 3$ | $i = 4$ |
|-------|---------|---------|---------|---------|
| $p_i$ | 19      | 19      | 14      | 14      |
| $w_i$ | 2       | 4       | 3       | 3       |

- ▶ Gli oggetti sono ordinati per efficienza non crescente, poiché  $\frac{19}{2} = 9.5 > \frac{19}{4} = 4.75 > \frac{14}{3} \approx 4.67$ .
- ▶ Eseguiamo  $FP^\varepsilon$  con  $\varepsilon = \frac{1}{2}$ .

## Esempio: esecuzione di $FP^\varepsilon$ (2/4)

### Lower bound.

- ▶  $GR$  inserisce l'oggetto 1 (capacità residua  $8 - 2 = 6$ ) e l'oggetto 2 (capacità residua  $6 - 4 = 2$ ), mentre gli oggetti 3 e 4 non entrano, quindi  $GR(I) = 19 + 19 = 38$ .
- ▶ Il miglior oggetto singolo ha profitto  $19 \leq 38$ , quindi  $GR^e(I) = 38$ .

**Scalatura.** Il fattore di scala e i profitti scalati sono

$$K = \frac{\varepsilon GR^e(I)}{n} = \frac{\frac{1}{2} \cdot 38}{4} = 4.75,$$

$$p'_1 = p'_2 = \left\lfloor \frac{19}{4.75} \right\rfloor = 4, \quad p'_3 = p'_4 = \left\lfloor \frac{14}{4.75} \right\rfloor = 2.$$

Il profitto scalato totale è  $P' = 4 + 4 + 2 + 2 = 12$ , non maggiore di  $Q = \lfloor 2n/\varepsilon \rfloor = 16$ , quindi qui il troncamento non interviene.

## Esempio: esecuzione di $FP^\epsilon$ (3/4)

**Programmazione dinamica sui profitti.** Sull'istanza scalata, i pesi minimi  $u(4, q)$  sono

| $q$       | 0 | 2 | 4 | 6 | 8 | 10 | 12 |
|-----------|---|---|---|---|---|----|----|
| $u(4, q)$ | 0 | 3 | 2 | 5 | 6 | 9  | 12 |

Il massimo  $q$  con  $u(4, q) \leq c = 8$  è  $q = 8$ , ottenuto con gli oggetti  $\{1, 2\}$  di peso totale 6.

## Esempio: esecuzione di $FP^\varepsilon$ (4/4)

**Soluzione restituita.**  $FP^\varepsilon$  restituisce  $S = \{1, 2\}$ , di profitto effettivo

$$FP^\varepsilon(I) = 19 + 19 = 38.$$

- ▶ La soluzione ottima è  $\tilde{S} = \{1, 3, 4\}$ , di peso  $2 + 3 + 3 = 8$  e valore  $z(I) = 19 + 14 + 14 = 47$ .
- ▶ La scalatura ha reso gli oggetti 1 e 2 buoni quanto gli oggetti 1, 3 e 4 insieme.

Il rapporto di prestazione è

$$\frac{z(I)}{FP^\varepsilon(I)} = \frac{47}{38} \approx 1.24.$$

## L'algoritmo Basic-FPTAS $FP^\varepsilon$ :

- ▶ calcola  $GR^e(I)$  e il fattore di scala  $K = \varepsilon GR^e(I)/n$
- ▶ arrotonda per difetto i profitti,  $p'_j = \lfloor p_j/K \rfloor$ ; i pesi e la capacità restano invariati
- ▶ risolve esattamente l'istanza scalata con la programmazione dinamica sui profitti, con la tabella troncata a  $Q = \lfloor 2n/\varepsilon \rfloor$

## Tempo di calcolo:

- ▶  $O(nQ) = O(n^2/\varepsilon)$

## Riepilogo di questa parte (1/2)

### Oggetto critico e algoritmi greedy:

- ▶ l'oggetto critico  $s$  è il primo oggetto che non entra, prendendo gli oggetti per efficienza non crescente;  $rc$  è la capacità residua
- ▶  $GR$  aggiunge ogni oggetto che entra;  $GR^e$  restituisce la migliore fra la soluzione greedy e il miglior oggetto singolo

### Scommettere sugli oggetti:

- ▶  $H^2$  scommette su due oggetti di una soluzione ottima,  $H^\ell$  su  $\ell$  oggetti, ed entrambi completano ogni scommessa con  $GR^e$  su un'istanza residua
- ▶ tempi di calcolo  $O(n^3)$  e  $O(n^{\ell+1})$

## Riepilogo di questa parte (2/2)

### Arrotondare i profitti:

- ▶ l'algoritmo Basic-FPTAS  $FP^\varepsilon$  scala i profitti con  $K = \varepsilon GR^e(I)/n$  e risolve esattamente l'istanza scalata con la programmazione dinamica sui profitti, in tempo  $O(n^2/\varepsilon)$

### Prossimamente:

- ▶ il rapporto di prestazione nel caso peggiore di ogni algoritmo, gli schemi di approssimazione e la garanzia di prestazione assoluta