Il problema dello zaino
(knapsack problem)

Damiano Macedonio

mace@unive.it

Problema dello zaino 0-1

- Supponiamo di avere un insieme X composto da n oggetti etichettati con gli interi da 1 a n: {1, 2, ..., n}
  - L'oggetto i-esimo ha peso p[i] e valore v[i]
  - Esiste una unica istanza di ciascun oggetto
- Disponiamo un contenitore in grado di trasportare al massimo un peso pari a P
- Vogliamo determinare un sottoinsieme $Y \subseteq X$ di oggetti, tale che
  - Il peso complessivo degli oggetti in Y sia ≤ P
  - Il valore complessivo degli oggetti in Y sia il massimo possibile

Definizione formale

- Vogliamo determinare un sottoinsieme $Y \subseteq X$ di oggetti tale che:

$$\sum_{x \in Y} p(x) \leq P$$

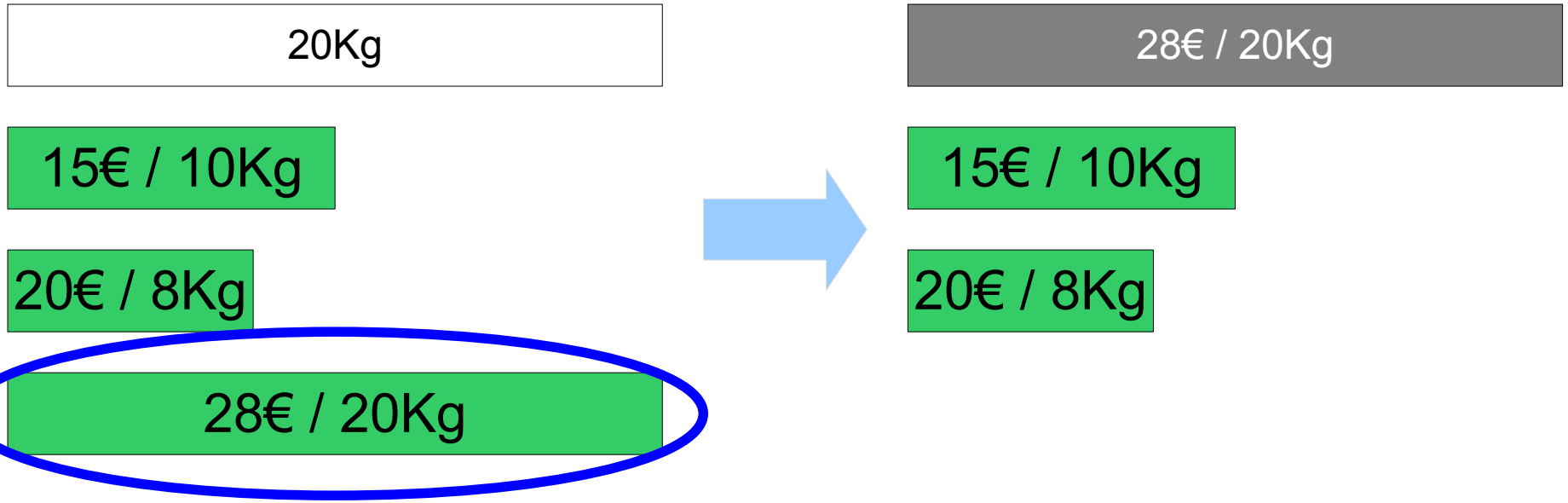
e tale da massimizzare il valore complessivo:

$$\sum_{x \in Y} v(x)$$

Approccio greedy #1

- Ad ogni passo, scelgo tra gli oggetti non ancora nello zaino quello di valore massimo, tra tutti quelli che hanno un peso minore o uguale alla capacità residua dello zaino

Esempio

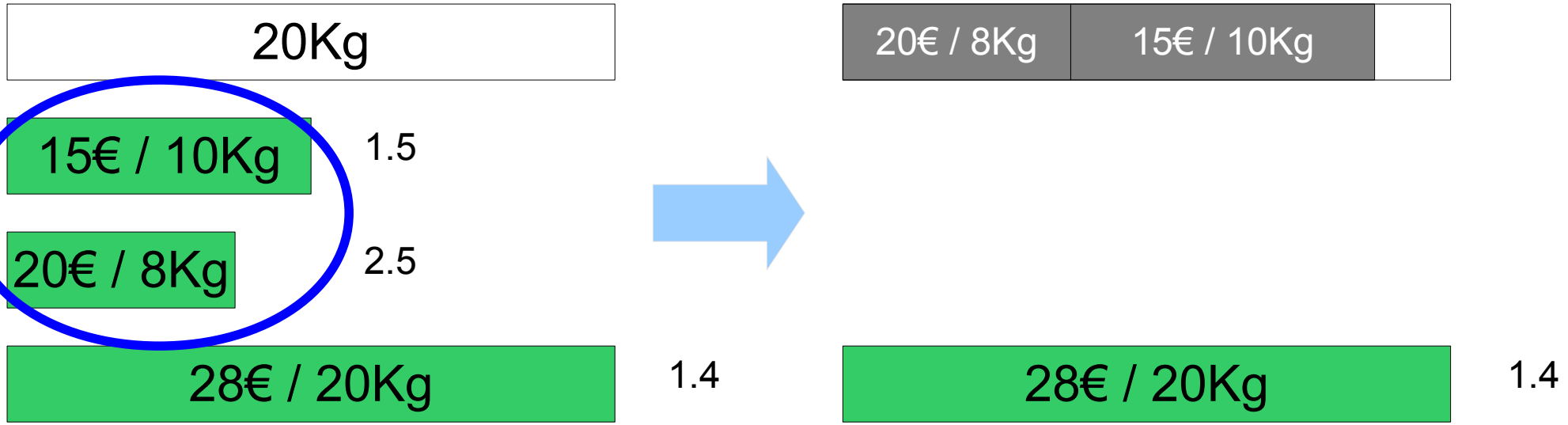


- La soluzione calcolata in questo esempio non è la soluzione ottima!
- Nota: questo algoritmo non fornisce sempre la soluzione ottima

Approccio greedy #2

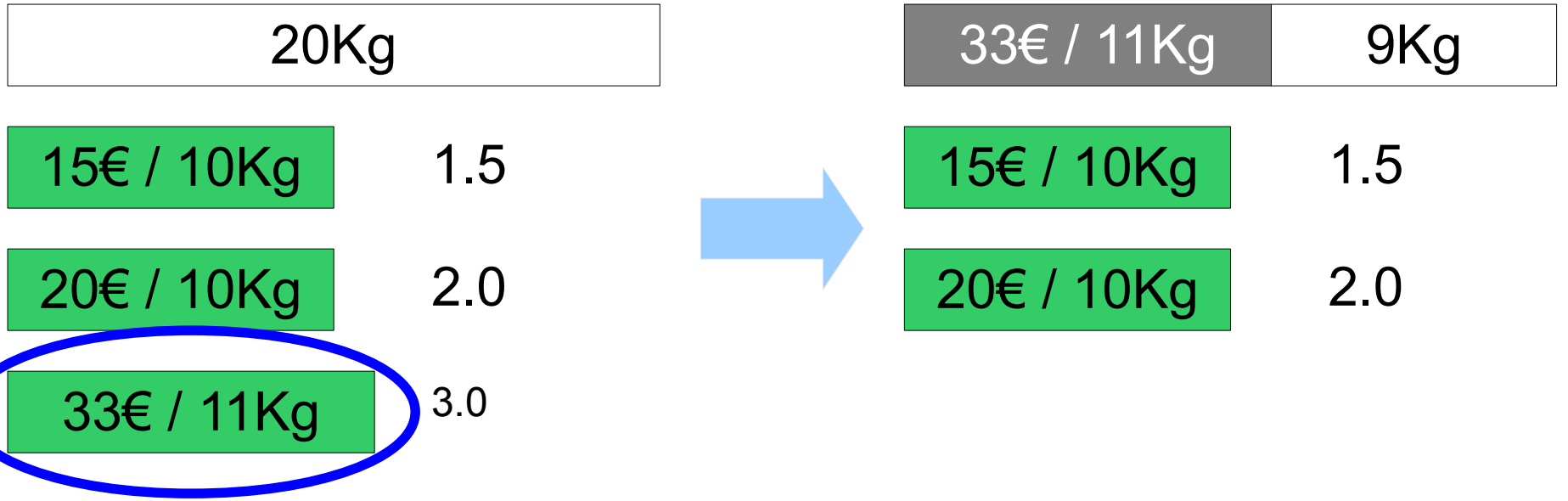
- Ad ogni passo, scelgo tra gli oggetti non ancora nello zaino quello di valore specifico massimo, tra tutti quelli che hanno un peso minore o uguale alla capacità residua dello zaino
  - Il valore specifico è definito come il valore di un oggetto diviso il suo peso

Esempio



- In questo esempio, l'algoritmo greedy #2 calcola la soluzione ottima, ma...

Esempio



- ...in questo altro esempio anche l'algoritmo greedy #2 non produce la soluzione ottima

Soluzione ottima basata sulla Programmazione Dinamica

- Questa soluzione funziona esclusivamente se i pesi sono numeri interi
- Sia V[i, j] il massimo valore ottenibile da un sottoinsieme degli oggetti {1, 2, ... i} in uno zaino che ha capacità massima j
  - i=1, 2, ...n
  - j=0, 1, ...P

Soluzione ottima basata sulla Programmazione Dinamica

- Sia V[i, j] il massimo valore ottenibile da un sottoinsieme degli oggetti {1, 2, ..., i} in uno zaino che ha capacità massima j
- V[i, 0] = 0 per ogni i=1..n
  - Se il peso massimo è zero, il valore è sempre zero
- V[1, j] = v[1] se j ≥ p[1]
  - C'è abbastanza spazio per l'oggetto numero 1
- V[1, j] = 0 se j < p[1]
  - Non c'è abbastanza spazio per l'oggetto numero 1

Soluzione ottima basata sulla Programmazione Dinamica

- Per calcolare V[i, j] distinguiamo due casi
  - Se $j<p[i]$ significa che l'i-esimo oggetto è troppo pesante per essere contenuto nello zaino. In tal caso V[i, j] = V[i-1, j]
  - Se $j \geq p[i]$ possiamo scegliere la migliore tra le seguenti possibilità
    - Inserire l'oggetto i-esimo nello zaino. Lo spazio residuo è j-p[i], il valore massimo ottenibile in questa ipotesi è V[i-1, j-p[i]]+v[i]
    - Non inserire l'oggetto i-esimo nello zaino. Il valore massimo ottenibile in questa ipotesi è V[i-1, j]
    - Scegliendo la migliore delle possibilià, possiamo dire V[i, j] = max{ V[i-1, j], V[i-1, j-p[i]]+v[i] }

Soluzione ottima basata sulla Programmazione Dinamica

- Riassumendo

$$V(i,j)=\begin{cases} V[i-1,j] & se\ j<p[i] \\ max\{V[i-1,j],V[i-1,j-p[i]]+v[i]\} & se\ j\geq p[i] \end{cases}$$

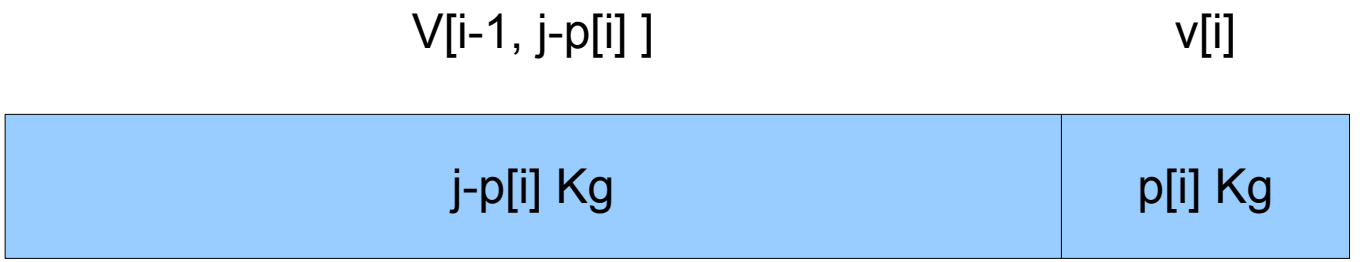


Tabella di programmazione dinamica / matrice V

p =[2, 7, 6, 4]
v =[12.7, 6.4, 1.7, 0.3]

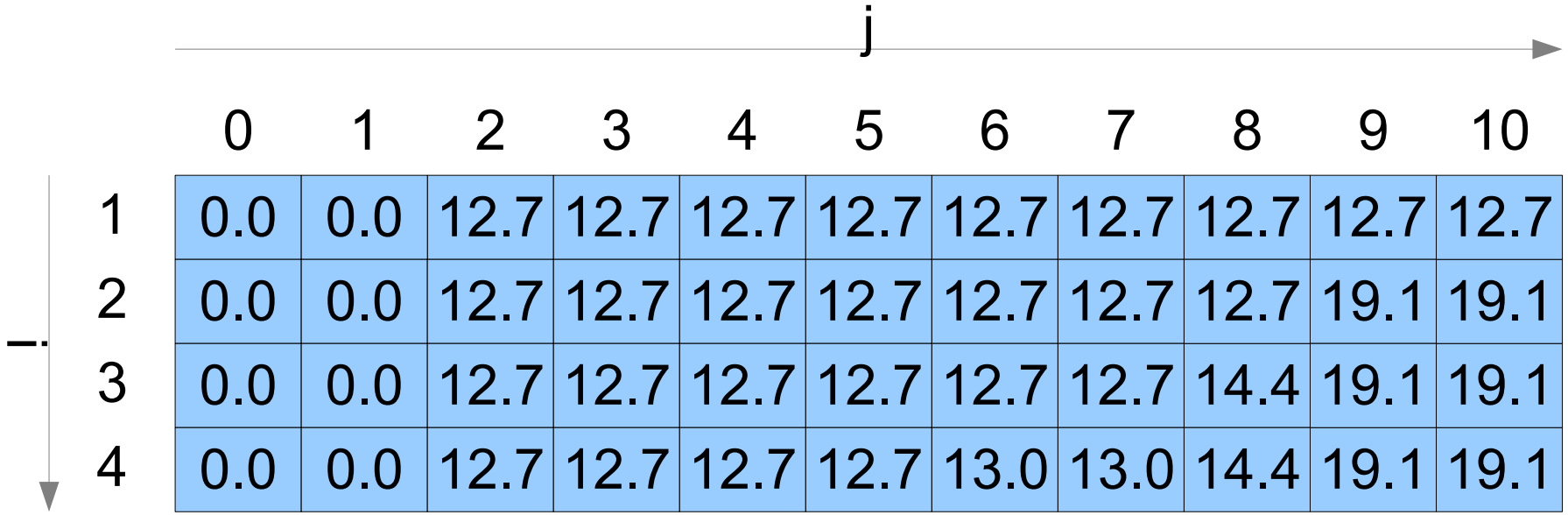
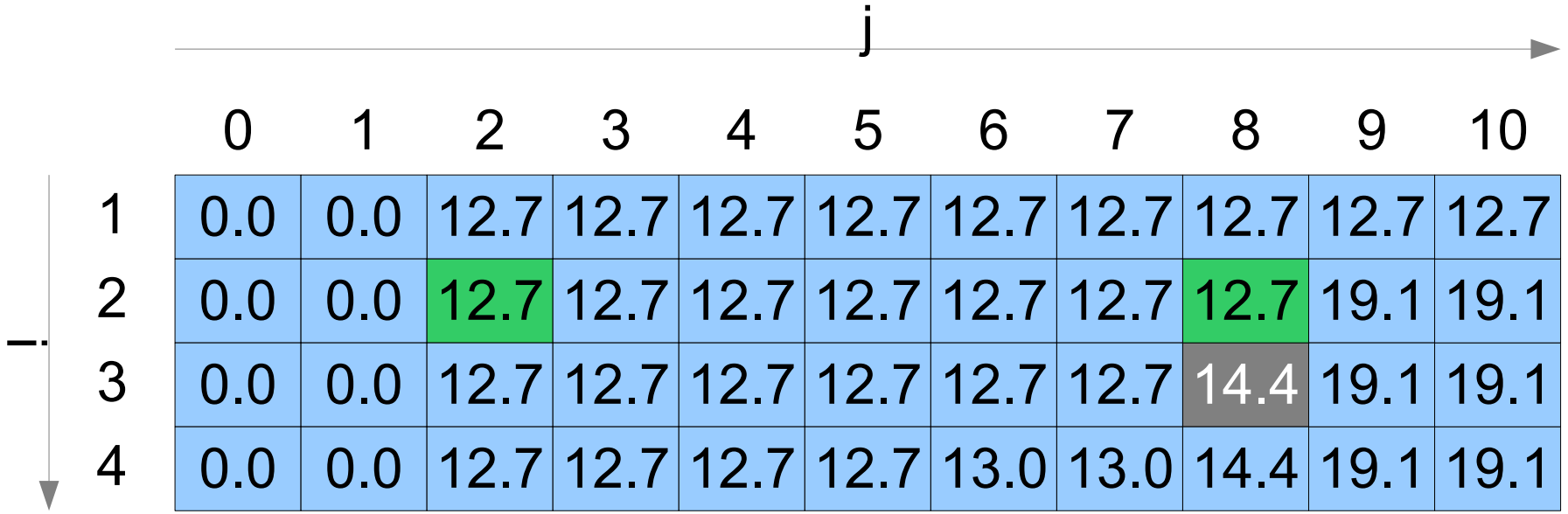


Tabella di programmazione dinamica / matrice V

p =[2, 7, 6, 4]
v =[12.7, 6.4, 1.7, 0.3]



$$V(3,8) = max\{V(2,8), V(2,8-p[3]) + v[3]\}$$
$$= max\{12.7, 14.4\}$$

Stampare la soluzione

- Il massimo valore che è possibile inserire nello zaino rispettando il vincolo di peso massimo è contenuto nella cella V[n, P] della tabella di programmazione dinamica
- Come facciamo a sapere quali oggetti fanno parte della soluzione ottima?
  - Usiamo una tabella ausiliaria booleana K[i, j] che ha le stesse dimensioni di V[i, j]
  - K[i, j] = true se e solo se l'oggetto i-esimo fa parte della soluzione ottima che ha valore complessivo V[i, j]

Tabella di programmazione dinamica / stampa soluzione ottima

```
d := P;
i := n;
while( i>0 ) do
    if ( K[i,d] == true ) then
        stampa "Seleziono oggetto " i
        d := d-p[i];
    endif
    i := i-1;
endwhile
```

p =[2, 7, 6, 4]
v =[12.7, 6.4, 1.7, 0.3]

	0	1	2	3	4	5	6	7	8	9	10
1	F	F	T	T	T	T	T	T	T	T	T
2	F	F	F	F	F	F	F	F	F	T	T
3	F	F	F	F	F	F	F	F	T	F	F
4	F	F	F	F	F	F	T	T	F	F	F