Algoritmi e Strutture Dati

Algoritmi Golosi (Greedy)

Maria Rita Di Berardini, Emanuela Merelli[1]

[1]Dipartimento di Matematica e Informatica
Università di Camerino

Problema della selezione di attività

Problema: Sia $S = \{a_1, a_2, \cdots, a_n\}$ un insieme di n attività che devono usare in maniera mutuamente esclusiva una data risorsa

Ogni attività a_i ha un tempo di inizio s_i ed un tempo di fine f_i, con $0 \leq s_i < f_i < \infty$, e si svolge nell'intervallo aperto $[s_i, f_i)$

Due attività a_i e a_j si dicono compatibili se gli intervalli $[s_i, f_i)$ e $[s_j, f_j)$ non si sovrappongono (e quindi se $s_i \geq f_j$ o $s_j \geq f_i$)

Obiettivo: selezionare il più grande sottoinsieme di attività in S mutuamente compatibili

Problema della selezione di attività: un esempio

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	2	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

$\{a_3, a_9, a_{11}\}$ è formato da attività mutuamente compatibili

$\{a_1, a_4, a_8, a_{11}\}$ hanno cardinalità massima

- Il problema della selezione di attività ha una sottostruttura ottima. Può essere quindi risolto con un algoritmo di PD
- Soddisfa anche la **proprietà della scelta golosa**: può essere risolto con un algoritmo goloso

Sottostruttura ottima della selezione di attività

Spazio di sottoproblemi: tutti i sottoproblemi della forma

$$S_{ij} = \{a_k \in S \mid f_i \leq s_k < f_k \leq s_j\}$$

dove S_{ij} è il sottoinsieme di attività in S che iniziano dopo la fine di a_i ($f_i \leq s_k$) e finisco prima dell'inizio di a_j ($f_k \leq s_j$)

Per rappresentare tutto l'insieme S introduciamo due attività fittizie:

- a_0 con tempo di fine $f_0 = 0$
- a_{n+1} con tempo di inizio $s_{n+1} = \infty$

Allora $S = S_{0\,n+1}$

Assumiamo, inoltre, che le attività in S siano ordinate in modo crescente rispetto ai tempi di fine, ossia: $f_1 \leq f_2 \leq \ldots \leq f_n$

Sottostruttura ottima della selezione di attività

Assumiamo $S_{ij} \neq \emptyset$ (caso non banale) e sia A_{ij} una soluzione ottima S_{ij}. Assumiamo inoltre che $a_y \in A_{ij}$.

Possiamo utilizzare a_y per individuare due sottoproblemi:

- S_{iy} (iniziano dopo la fine di a_i e finiscono prima dell'inizio di a_y)
- S_{yj} (iniziano dopo la fine di a_y e finiscono prima dell'inizio di a_j)

$S_{ij} = S_{iy} \cup \{a_y\} \cup S_{yj}$ + una delle attività non compatibili con a_y; quindi

$$A_{ij} = A_{iy} \cup \{a_y\} \cup A_{yj}$$

dove A_{iy} e A_{yj} sono le soluzioni per S_{iy} e S_{yj} contenute all'interno della soluzione ottima A_{ij} per S_{ij}

Sottostruttura ottima della selezione di attività

Assumiamo, per assurdo, che A_{iy} non sia ottima

Esiste un insieme A'_{iy} di attività in S_{iy} mutuamente compatibili tale che $|A'_{iy}| > |A_{iy}|$

Allora $A'_{ij} = A'_{iy} \cup \{a_y\} \cup A_{yj}$ è una soluzione di S_{ij} migliore della soluzione ottima A_{ij}, il che è chiaramente una contraddizione

In maniera simile possiamo dimostrare che A_{yj} è ottima

Il problema della selezione di attività soddisfa la sottostruttura ottima e, quindi, può essere risolto mediante un algoritmo di PD (l' algoritmo è abbastanza simile a quello per il prodotto di una sequenza di matrici)

Problema della selezione di attività

Un algoritmo goloso (così come gli algoritmi di PD) costruisce la soluzione ottima effettuando una serie di scelte, ma in questo caso la scelta non dipende da soluzioni di sottoproblemi

Un algoritmo goloso costruisce una soluzione **globalmente ottima** effettuando una serie di scelte **golose**, o localmente ottime

Scelta golosa = scelta ottima in un dato momento

Gli algoritmi golosi sono, in generale più semplici ed efficienti degli algoritmi di PD

Non sempre la "tecnica golosa" ci consente di risolvere un dato problema di ottimizzazione (proprietà della scelta golosa)

Problema della selezione di attività

Per il problema della selezione di attività, effettuare la scelta golosa significa scegliere l'attività che, in un dato momento, "occupa la risorsa per il minor tempo possibile", ossia quella con minor tempo di fine

L'algoritmo goloso sceglierà:

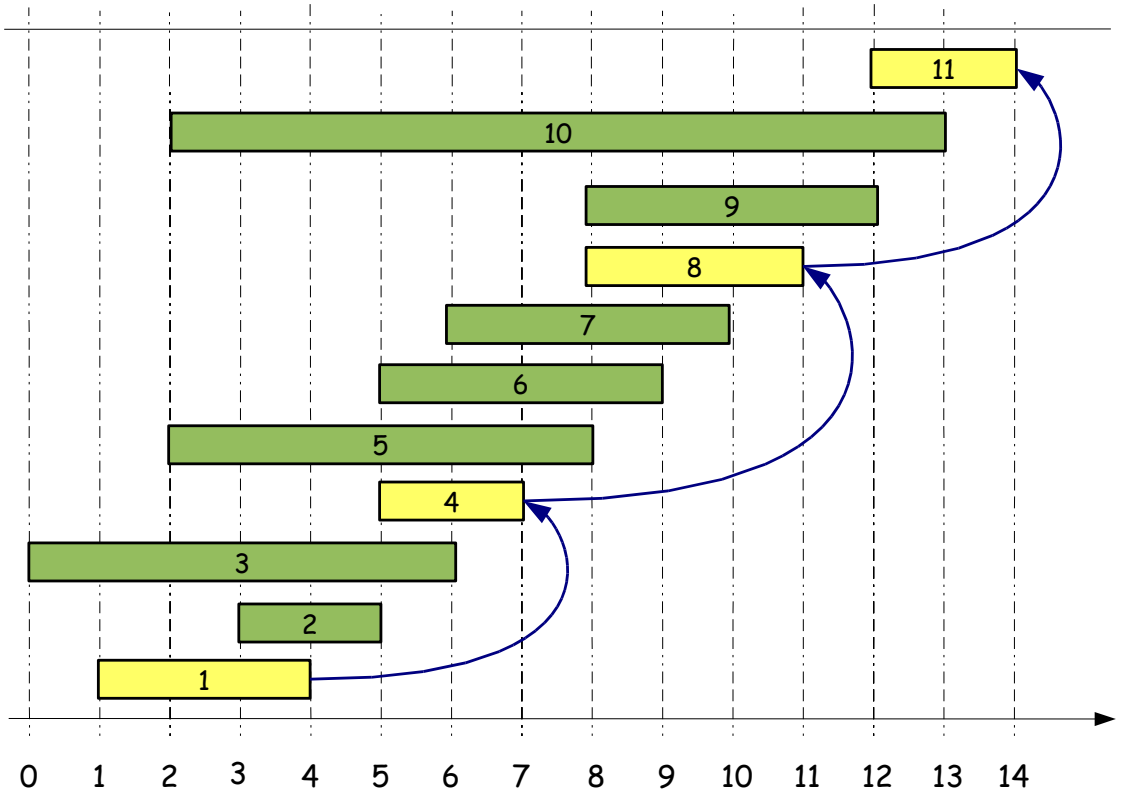
1. l'attività a_1, quella che minor tempo di fine,
2. la prima attività a_i $(i \geq 2)$ compatibile con a_1, ossia tale che

$$s_i \geq f_1$$

   (a causa dell'ordinamento dei tempi di fine, $s_1 < f_1 \leq f_i$, se $i \geq 2$)
3. la prima attività compatibile con la seconda attività scelta
4. e così via

Una soluzione ottima

Le attività sono ordinate in maniera crescente rispetto a f_i



Un algoritmo greedy per la selezione di attività

GREEDYACTIVITYSELECTOR(s, f)

```
1  n ← length[s]
2  A ← {a_1}
3  m ← 1 ▷ memorizza l'indice dell'ultima attività scelta
4  for i ← 2 to n
5  do if s_i ≥ f_m
6        then A ← A ∪ {a_i}
7             m ← i
8  return A
```

Ad ogni passo l'algoritmo aggiunge ad A la prima attività compatibile (quella localmente ottima) con a_m (l'ultima attività selezionata)

Questa serie di scelte localmente ottime porta ad una soluzione che è globalmente ottima. Perchè?

Correttezza dell'algoritmo goloso

Il problema della selezione di attività soddisfa la proprietà della scelta golosa:

per ogni sottoproblema non banale, esiste sempre una soluzione ottima che contiene la scelta golosa

Nel nostro caso:

Se $S_{ij} \neq \emptyset$, e a_m è l'attività in S_{ij} con minor tempo di fine (la scelta golosa), allora esiste sempre una soluzione ottima per S_{ij} che contiene a_m

Correttezza dell'algoritmo goloso

Sia $S_{ij} \neq \emptyset$ e $a_m \in S_{ij}$ con minor tempo di fine; sia, inoltre, A una soluzione ottima per S_{ij}

Correttezza dell'algoritmo goloso

Sia $S_{ij} \neq \emptyset$ e $a_m \in S_{ij}$ con minor tempo di fine; sia, inoltre, A una soluzione ottima per S_{ij}

Distinguiamo i seguenti casi:

- $a_m \in A$: non abbiamo nulla da dimostrare
- $a_m \notin A$: scegliamo

$$A' = (A - \{a_y\}) \cup \{a_m\}$$

dove a_y è l'attività in A com minor tempo di fine e dimostriamo che A' è ottima per S_{ij}

cerchiamo una nuova soluzione per S_{ij} tale che $a_m \in A'$

Correttezza dell'algoritmo goloso

Per dimostrare

$$A' = (A - \{a_y\}) \cup \{a_m\}$$

è una soluzione ottima dobbiamo provare che A' è :

- **ammissibile** (contiene attività mutuamente compatibili), e
- ha lo stesso valore della soluzione ottima A, ossia $|A'| = |A|$

Correttezza dell'algoritmo goloso

Per dimostrare

$$A' = (A - \{a_y\}) \cup \{a_m\}$$

è una soluzione ottima dobbiamo provare che A' è :

- **ammissibile** (contiene attività mutuamente compatibili), e
- ha lo stesso valore della soluzione ottima A, ossia $|A'| = |A|$

  questa cosa è in realtà ovvia perchè $|A'| = (|A| - 1) + 1$

Correttezza dell'algoritmo goloso

Per dimostrare

$$A' = (A - \{a_y\}) \cup \{a_m\}$$

è una soluzione ottima dobbiamo provare che A' è :

- **ammissibile** (contiene attività mutuamente compatibili), e
- ha lo stesso valore della soluzione ottima A, ossia $|A'| = |A|$

  questa cosa è in realtà ovvia perchè $|A'| = (|A| - 1) + 1$

Dimostriamo l'ammissibilità

Ammissibilità di $A' = (A - \{a_y\}) \cup \{a_m\}$

Le attività in $A - \{a_y\}$ sono mutuamente compatibili in A' perchè lo erano già in A

Ammissibilità di $A' = (A - \{a_y\}) \cup \{a_m\}$

Le attività in $A - \{a_y\}$ sono mutuamente compatibili in A' perchè lo erano già in A

Dimostriamo che a_m è compatibile con ogni $a_k \in A - \{a_y\}$

Ammissibilità di $A' = (A - \{a_y\}) \cup \{a_m\}$

Le attività in $A - \{a_y\}$ sono mutuamente compatibili in A' perchè lo erano già in A

Dimostriamo che a_m è compatibile con ogni $a_k \in A - \{a_y\}$

Poichè $a_k, a_y \in A$, a_k e a_y sono compatibili, quindi $s_k \geq f_y$ oppure $s_y \geq f_k$

Ammissibilità di $A' = (A - \{a_y\}) \cup \{a_m\}$

Le attività in $A - \{a_y\}$ sono mutuamente compatibili in A' perchè lo erano già in A

Dimostriamo che a_m è compatibile con ogni $a_k \in A - \{a_y\}$

Poichè $a_k, a_y \in A$, a_k e a_y sono compatibili, quindi $s_k \geq f_y$ oppure $s_y \geq f_k$

se fosse $s_y \geq f_k$, allora $f_k \geq f_y$ ($a_y \in A$ con minor tempo di fine) implicherebbe quindi $s_y \geq f_k \geq f_y$; impossibile

Ammissibilità di $A' = (A - \{a_y\}) \cup \{a_m\}$

Le attività in $A - \{a_y\}$ sono mutuamente compatibili in A' perchè lo erano già in A

Dimostriamo che a_m è compatibile con ogni $a_k \in A - \{a_y\}$

Poichè $a_k, a_y \in A$, a_k e a_y sono compatibili, quindi $s_k \geq f_y$ oppure $s_y \geq f_k$

se fosse $s_y \geq f_k$, allora $f_k \geq f_y$ ($a_y \in A$ con minor tempo di fine) implicherebbe quindi $s_y \geq f_k \geq f_y$; impossibile

deve essere $s_k \geq f_y$

Infine, $s_k \geq f_y \geq f_m$ implica a_k compatibile con a_m

Sottoproblemi da risolvere

La proprietà della scelta golosa ci dice che è sempre possibile costruire una soluzione ottima per S_{ij} scegliendo come prossima attività a_m (l'elemento in S_{ij} con minor tempo di fine)

Inoltre, $S_{ij} = S_{im} \cup \{a_m\} \cup S_{mj}$ + delle attività non compatibili con a_m

Quindi, una volta scelta a_m non ci rimane che risolvere i problemi S_{im} e S_{mj}

In realtà $S_{im} = \emptyset$.

Infatti $S_{im} \subseteq S_{ij}$ tale che $a_k \in S_{im}$ implica $f_i \leq s_k < f_k \leq s_m$

Se fosse $a_k \in S_{im} \neq \emptyset$, allora $f_k \leq s_m < f_m$ (il che contraddice il fatto he $a_m \in S_{ij}$ con minor tempo di fine)

Ricapitolando:

Abbiamo dimostrato che, per risolvere un sottoproblema $S_{ij} \neq \emptyset$ (non banale)

- possiamo scegliere l'attività $a_m \in S_{im}$ con minor tempo di fine,
- questa scelta genera un solo sottoproblema non banale S_{mj} che può essere risolto utilizzando lo stesso approccio

Quindi l'algoritmo greedy proposto è corretto

Elementi della strategia golosa

Un algoritmo goloso costruisce una soluzione ottima di un dato problema effettuando una serie di scelte

Ogni decisione viene presa tenendo conto della scelta che, al momento, è la migliore

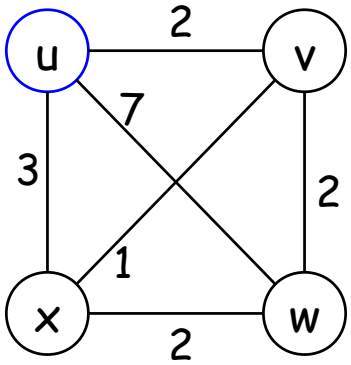
Questa strategia euristica non sempre produce una soluzione ottima

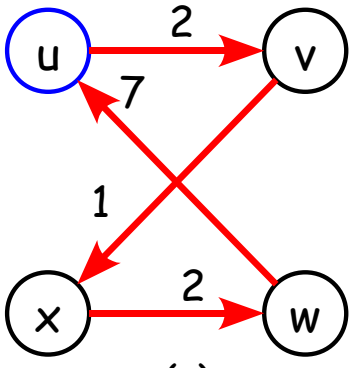
Ma quando lo fa, ci consente di scrivere algoritmi molto efficienti

Un problema non risolvibile mediante algoritmi golosi

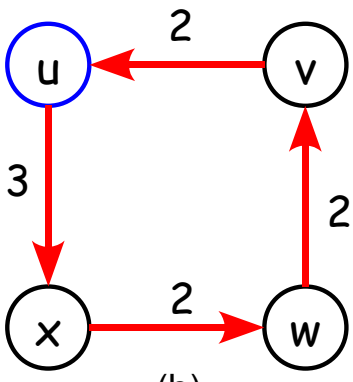
Il problema del commesso viaggiatore: dato un insieme di n città trovare la strada più breve per visitarle tutte una ed una sola volta e tornare alla città di partenza

Assumendo di rappresentare le città e le relative distanze come nodi ed archi di un grafo:





(a)
soluzione golosa
2 + 1 + 2 + 7 = 12



(b)
soluzione ottima
3 + 2 + 2 + 2 = 9

Proprietà della scelta golosa

Stabilisce che una soluzione globalmente ottima può essere ottenuta mediante una serie di scelte localmente ottime

Ad ogni passo, facciamo sempre la scelta che sembra migliore per il sottoproblema corrente, senza considerare le soluzioni dei sottoproblemi

La scelta fatta da un algoritmo goloso può dipendere dalle scelte fatte in precedenza (che determinano il problema corrente) ma non dalle scelte future (soluzioni dei sottoproblemi)

A differenza della PD (che, in generale, procede in maniera bottom-up), una strategia golosa procede di solito in **maniera top-down** (ossia, dall'alto verso il basso): facendo una scelta golosa dopo l'altra, riducendo ogni istanza di un determinato problema in un problema più piccolo

Il problema dello zaino: due versioni

Il problema dello zaino 0-1

Un ladro entra in un magazzino e trova n oggetti $\{a_1, ..., a_n\}$ di peso $w_1, ..., w_n$ e valore $v_1, ..., v_n$ (per ogni $i = 1, \cdots, n$, v_i e w_i sono dei numeri interi)

Il ladro vorrebbe realizzare il furto di maggior valore compatibile con la capacità W del suo zaino

Ciascun oggetto può essere preso o lasciato; il ladro non può prendere frazioni di oggetti né più volte lo stesso oggetto (di qui il nome "zaino 0-1")

Quali oggetti dovrà prendere??

Il problema dello zaino: due versioni

Il problema dello zaino frazionario

Un ladro entra in un magazzino e trova n oggetti $\{a_1, ..., a_n\}$ di peso $w_1, ..., w_n$ e valore $v_1, ..., v_n$ (per ogni $i = 1, \cdots, n$, v_i e w_i sono dei numeri interi)

Il ladro vorrebbe realizzare il furto di maggior valore compatibile con la capacità W del suo zaino

In questo caso il ladro **può prendere frazioni di oggetti**, anzichè fare una scelta binaria (0-1)

Quali oggetti dovrà prendere??

Il problema dello zaino: due versioni

Il problemi sono molto simili ...

ed entrambi verificano la sottostruttura ottima,

ma **solo** il problema dello zaino frazionario può essere risolto con un algoritmo goloso

Sottostruttura ottima

Problema 0-1:

- consideriamo il carico più prezioso che pesa al più W chili
- togliendo da questo carico un qualsiasi oggetto a_j, la soluzione che rimane deve contenere il carico più prezioso (il cui peso non supera $W - w_j$) che possiamo realizzare con gli $n - 1$ oggetti rimanenti (tutti tranne a_j)

Problema frazionario:

- consideriamo il carico più prezioso che pesa al più W chili
- assumiamo che il carico ottimo contenga una frazione x_i dell'oggetto a_j il cui peso è pari a w
- se togliamo dal carico ottimo questa parte dell'oggetto a_j, ciò che resta è il carico più prezioso (il cui peso non supera $W - w$) che possiamo realizzare con gli $n - 1$ oggetti originali più $w_j - w$ chili dell'oggetto a_j

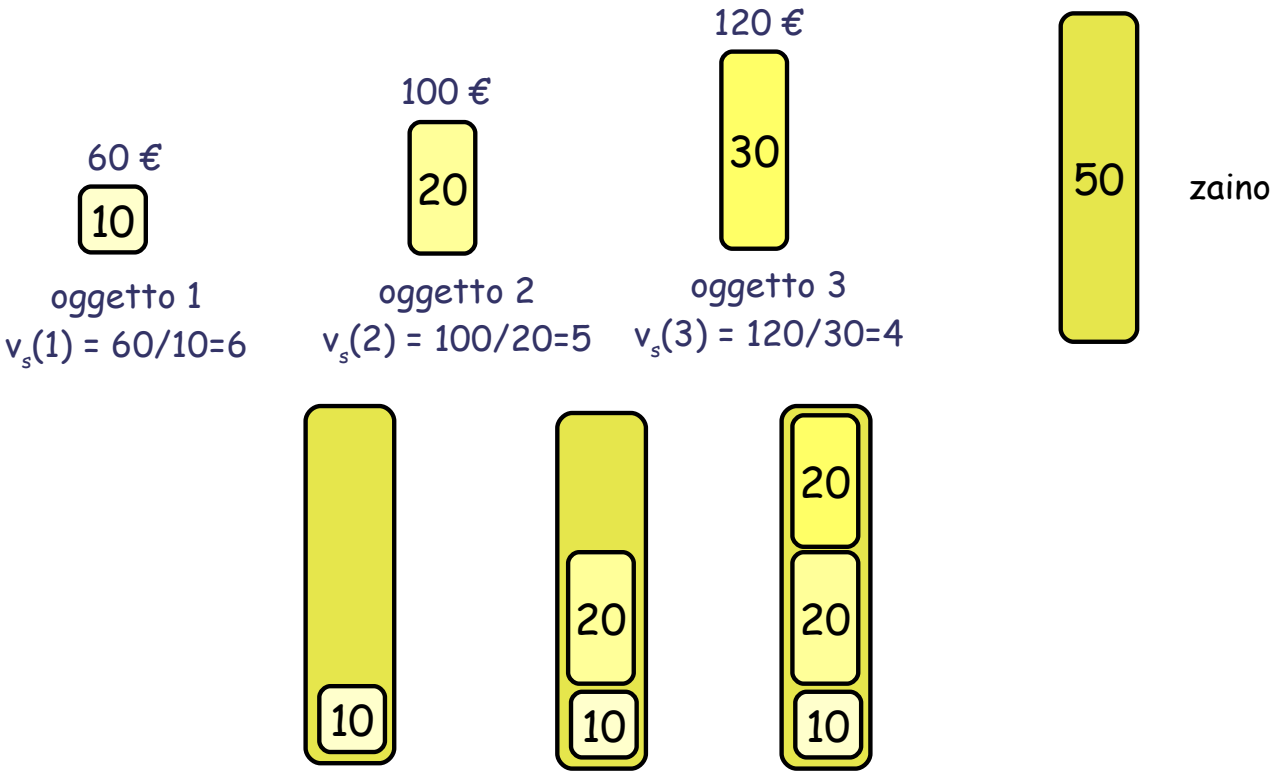
Un algoritmo goloso per lo zaino frazionario

Strategia golosa per il problema dello zaino frazionario:

1. calcoliamo innanzitutto il valore per unità peso, o **valore specifico**, di ciascun oggetto; tale peso specifico è definito come v_i / w_i
2. Ad ogni passo, il ladro prende la maggiore quantità possibile dell'oggetto con peso specifico maggiore

   (la strategia golosa per il problema dello zaino 0-1, consiste nel prendere l'oggetto con peso specifico maggiore)
3. esaurita la scorta di questo oggetto, e se nello zaino c'è ancora posto possiamo ripetere l'operazione

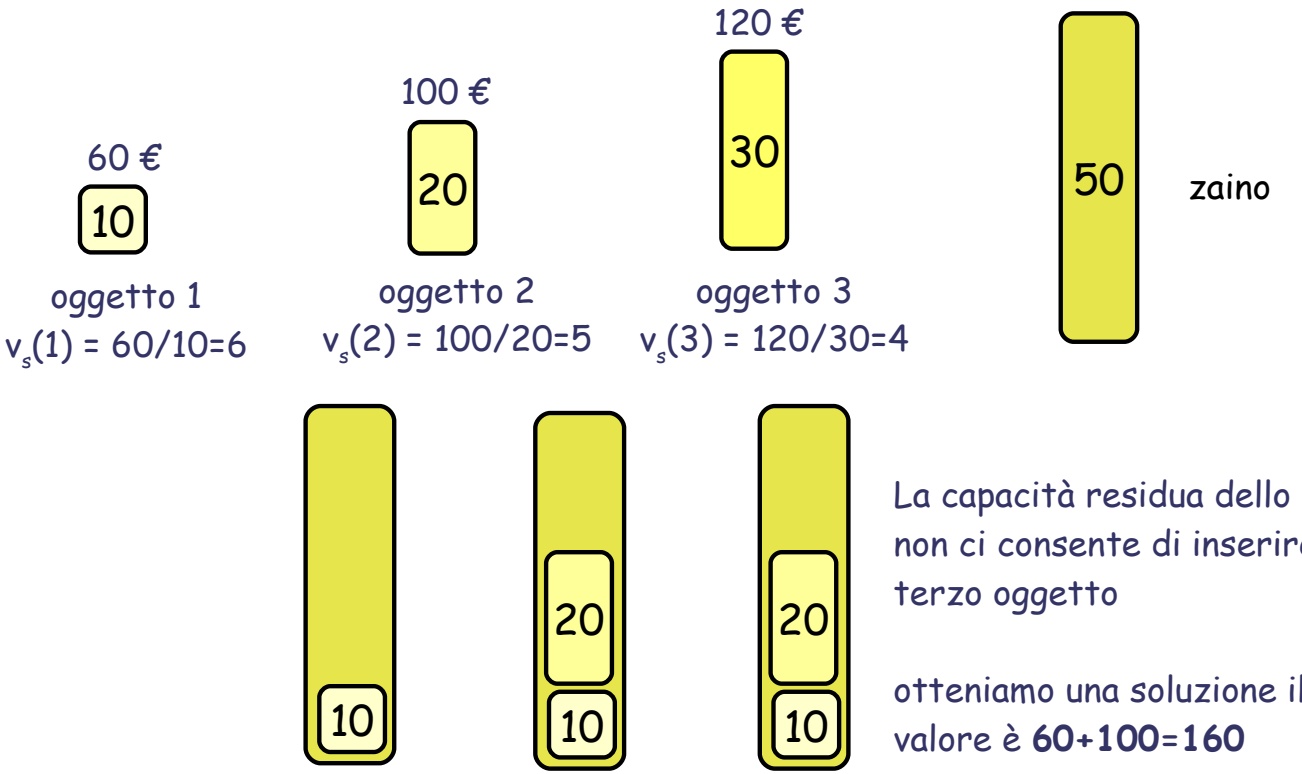
Un algoritmo goloso per lo zaino frazionario



Otteniamo una soluzione composta dali oggetti 1 e 2 (interi), più una parte (2/3) dell'oggetto 3 il cui peso è pari a 50 Kg e con valore

60 + 100 + (2/3) 120 = 60 + 100 + 80 = 240 € (soluzione ottima)

Un algoritmo goloso per lo zaino 0-1



La **soluzione ottima** include gli oggetti 2 e 3 ed ha un peso di 50Kg ed un valore pari a **100+120= 220 €**

Proprietà della scelta golosa

Sia $S = \{a_1, ..., a_n\}$ un insieme di n oggetti ciascuno con valore v_i, peso w_i e, quindi, con peso specifico v_i/w_i

Assumiamo che gli oggetti siano ordinati in maniera decrescente rispetto ai valori v_i/w_i

A questo punto, a_1 è l'oggetto con maggior peso specifico e denotiamo con q la maggiore quantità di tale oggetto che è possibile inserire nello zaino vuoto

Dobbiamo dimostrare che esiste sempre una soluzione ottima dello zaino frazionario contenente una quantità q dell'oggetto a_1

Proprietà della scelta golosa

Ad ogni soluzione A del problema può essere associata una tupla

$$t_A = \langle q_1, ..., q_n \rangle$$

Per ogni $i = 1, ..., n$, $0 \leq q_i \leq w_i$ e rappresenta la porzione dell'oggetto i selezionata ($q_i = 0$ se o_i non contenuto nello zaino)

- l'insieme degli oggetti residui è $R(t_A) = \{i \mid q_i = 0\}$
- gli oggetti nello zaino $Z(t_A) = \{i \in S \mid q_i > 0\}$
- il peso di t come $W(t_A) = \sum_{i=1}^{n} q_i w_i \leq W$
- il valore di t come $V(t_A) = \sum_{i=1}^{n} q_i v_i$

Proprietà della scelta golosa

Teorema: è sempre possibile costruire una soluzione ottima dello zaino frazionario contenente la maggiore quantità q dell'oggetto o_1 (quello con maggior peso specifico)

Proof: sia A una soluzione ottima del problema ed $t_A = \langle q_1, ..., q_n \rangle$ la tupla ad essa associata

Caso 1: $q_1 = q$, non abbiamo nulla da dimostrare

Proprietà della scelta golosa

Caso 2: $0 \leq q_1 < q$

Consideriamo la soluzione A' la cui tupla associata $t_{A'} = \langle q'_1, ..., q'_n \rangle$ è tale che:

$$q'_i = q_i \qquad \text{per } i > 2$$

$$q'_1 = q = q_1 + (q - q_1)$$

$$q'_2 = q_2 - \frac{v_1}{v_2}(q - q_1)$$

Dimostriamo che A' è ottima, ossia

1. è ammissibile, ossia che $W(t_{A'}) \leq W$
2. e tale che $V(t_{A'}) = V(t_A)$

Ammissibilità

Consideriamo

$$q'_1 w_1 + q'_2 w_2 =$$

$$q w_1 + (q_2 - \frac{v_1}{v_2}(q - q_1)) w_2 =$$

$$q w_1 + q_2 w_2 - v_1 (q - q_1) \frac{w_2}{v_2}$$

Ora

$$\frac{v_2}{w_2} \leq \frac{v_1}{w_1} \text{ implica } \frac{w_2}{v_2} \geq \frac{w_1}{v_1}$$

Quindi:

$$q w_1 + q_2 w_2 - v_1 (q - q_1) \frac{w_2}{v_2} \leq q w_1 + q_2 w_2 - v_1 (q - q_1) \frac{w_1}{v_1} =$$

$$q w_1 + q_2 w_2 - w_1 (q - q_1) = q_1 w_1 + q_2 w_2$$

Ammissibilità

Allora

$$q'_1 w_1 + q'_2 w_2 \leq w_1 = q_1 w_1 + q_2 w_2$$

Questo ci permette di concludere che

$$\begin{aligned} W(t_{A'}) &= \sum_{i=1}^{n} q'_i w_i \\ &= q'_1 w_1 + q'_2 w_2 + (\sum_{i=2}^{n} q'_i w_i) \\ &\leq q_1 w_1 + q_2 w_2 + (\sum_{i=2}^{n} q_i w_i) \\ &= \sum_{i=1}^{n} q_i w_i = W(t_A) \leq W \end{aligned}$$

Valore ottimo

In maniera simile,

$$q'_1 v_1 + q'_2 v_2 =$$

$$q v_1 + (q_2 - \frac{v_1}{v_2}(q - q_1)) v_2 =$$

$$q v_1 + q_2 v_2 - v_1(q - q_1) = q_1 v_1 + q_2 v_2$$

e

$$\begin{aligned} V(t_{A'}) &= \sum_{i=1}^{n} q'_i v_i \\ &= q'_1 v_1 + q'_2 v_2 + (\sum_{i=2}^{n} q'_i v_i) \\ &= q_1 v_1 + q_2 v_2 + (\sum_{i=2}^{n} q_i v_i) = \\ &= \sum_{i=1}^{n} q_i v_i = V(t_A) \end{aligned}$$