

Problema *KNAPSACK*

Nel problema *KNAPSACK* è dato uno zaino con capacità un intero positivo b e n oggetti.

A ciascun oggetto i sono associati due interi positivi:

- un peso p_i ;
- un valore v_i .

Il problema consiste nello stabilire quali oggetti mettere nello zaino, senza superarne la capacità e massimizzando la somma dei valori degli oggetti inseriti.

Formulazione del problema con variabili

Introduciamo delle variabili booleane

$$x_i = \begin{cases} 0 & \text{oggetto } i \text{ non inserito} \\ 1 & \text{oggetto } i \text{ inserito} \end{cases}$$

In tal caso il problema è formulabile in questo modo:

$$\begin{aligned} \max \quad & \sum_{i=1}^n v_i x_i && \text{(valore oggetti inseriti)} \\ & \sum_{i=1}^n p_i x_i \leq b && \text{(vincolo capacità zaino)} \\ & x_i \in \{0, 1\} && \forall i \in \{1, \dots, n\} \end{aligned}$$

Branch-and-bound per *KNAPSACK*

Rispetto allo schema generale visto in precedenza dobbiamo specificare:

- come si calcola un upper bound su un sottinsieme;
- come si effettua il branching;
- come si individuano soluzioni ammissibili con cui, eventualmente, aggiornare il valore del lower bound LB .

Upper bound $U(S)$

Cominciamo con il calcolare l'upper bound $U(S)$ su *tutta* la regione ammissibile S .

L'upper bound si calcola risolvendo un rilassamento, detto rilassamento lineare, ottenuto sostituendo i vincoli $x_i \in \{0, 1\}$ con i vincoli $x_i \in [0, 1]$.

Risoluzione del rilassamento lineare

- Riordinare, eventualmente, gli oggetti in modo non crescente rispetto ai rapporti valore/peso $\frac{v_i}{p_i}$, cioè si abbia che:

$$\frac{v_1}{p_1} \geq \frac{v_2}{p_2} \geq \dots \geq \frac{v_n}{p_n}.$$

- Si calcolino i valori

$$b - p_1, \quad b - p_1 - p_2, \quad b - p_1 - p_2 - p_3, \quad \dots$$

fino ad arrivare al primo valore negativo

$$b - \sum_{j=1}^{r+1} p_j$$

se vi si arriva (se non vi si arriva vuol dire semplicemente che tutti gli oggetti possono essere messi nello zaino e la soluzione ottima del problema è proprio quella di mettere tutti gli oggetti nello zaino).

Continua

La soluzione ottima del rilassamento lineare è la seguente

$$x_1 = x_2 = \dots = x_r = 1, \quad x_{r+2} = x_{r+3} = \dots = x_n = 0$$
$$x_{r+1} = \frac{b - \sum_{j=1}^r p_j}{p_{r+1}}$$

e ha il valore ottimo

$$\sum_{j=1}^r v_j + v_{r+1} \frac{b - \sum_{j=1}^r p_j}{p_{r+1}}.$$

Notiamo anche che la soluzione

$$x_1 = x_2 = \dots = x_r = 1, \quad x_{r+1} = x_{r+2} = x_{r+3} = \dots = x_n = 0$$

ottenuta approssimando per difetto il valore dell'unica variabile (la x_{r+1}) che può avere coordinate non intere nella soluzione del rilassamento lineare, è appartenente a S (il peso dei primi r oggetti non supera la capacità dello zaino). Quindi tale soluzione può essere utilizzata per il calcolo del lower bound LB .

Sottinsiemi di S di forma particolare

Consideriamo sottinsiemi di S con questa forma:

$$S(I_0, I_1) = \{N \in S : \begin{array}{l} \text{l'oggetto } i \notin N \ \forall i \in I_0, \\ \text{l'oggetto } i \in N \ \forall i \in I_1 \end{array}\}$$

dove $I_0, I_1 \subseteq \{1, \dots, n\}$ e $I_0 \cap I_1 = \emptyset$.

In altre parole $S(I_0, I_1)$ contiene tutti gli elementi di S che non contengono gli oggetti in I_0 e contengono gli oggetti in I_1 . Possono invece indifferentemente contenere o non contenere gli oggetti nell'insieme

$$I_f = \{i_1, \dots, i_k\} = \{1, \dots, n\} \setminus (I_0 \cup I_1),$$

con $i_1 < \dots < i_k$.

Upper bound su $S(I_0, I_1)$

Il nostro originario problema *KNAPSACK* ristretto al sottinsieme $S(I_0, I_1)$ si presenta nella seguente forma:

$$\begin{aligned} \max \quad & \sum_{i \in I_1} v_i + \sum_{i \in I_f} v_i x_i \\ & \sum_{i \in I_f} p_i x_i \leq b - \sum_{i \in I_1} p_i \\ & x_i \in \{0, 1\} \quad \forall i \in I_f \end{aligned}$$

Possiamo notare che si tratta ancora di un problema di tipo *KNAPSACK* dove è presente una quantità costante nell'obiettivo ($\sum_{i \in I_1} v_i$), dove lo zaino ha ora capacità $b - \sum_{i \in I_1} p_i$ e dove l'insieme di oggetti in esame è ora ristretto ai soli oggetti in I_f .

Continua


Trattandosi ancora di un problema dello zaino, possiamo applicare a esso la stessa procedura che abbiamo adottato per trovare l'upper bound $U(S)$. Tale procedura darà in output oltre all'upper bound $U(S(I_0, I_1))$, anche una soluzione appartenente a $S(I_0, I_1)$ (e quindi a S) utilizzabile per il calcolo del lower bound LB .

Procedura di calcolo

- **Passo 1** Se $b - \sum_{i \in I_1} p_i < 0$, il nodo non contiene soluzioni ammissibili (gli oggetti in I_1 hanno già un peso superiore alla capacità b dello zaino). In tal caso ci si arresta e si pone

$$U(S(I_0, I_1)) = -\infty$$

Continua

 **Passo 2** Altrimenti, si sottraggano successivamente a $b - \sum_{i \in I_1} p_i$ i pesi degli oggetti in I_f

$$b - \sum_{i \in I_1} p_i - p_{i_1}$$

$$b - \sum_{i \in I_1} p_i - p_{i_1} - p_{i_2}$$

\vdots

arrestandoci se

Caso A si arriva ad un valore negativo, ovvero esiste $r \in \{1, \dots, k-1\}$ tale che

$$b - \sum_{i \in I_1} p_i - p_{i_1} - \dots - p_{i_r} \geq 0, \quad b - \sum_{i \in I_1} p_i - p_{i_1} - \dots - p_{i_r} - p_{i_{r+1}} < 0.$$

Caso B Si sono sottratti i pesi di tutti gli oggetti in I_f senza mai arrivare ad un valore negativo.

Continua

 **Passo 3** Se ci si trova nel caso A si restituiscano come output:

Upper bound

$$U(S(I_0, I_1)) = \sum_{i \in I_1} v_i + \sum_{h=1}^r v_{i_h} + v_{i_{r+1}} \times \frac{b - \sum_{i \in I_1} p_i - \sum_{h=1}^r p_{i_h}}{p_{i_{r+1}}}.$$

Soluzione l'elemento $N = I_1 \cup \{i_1, \dots, i_r\} \in S$, con il relativo valore

$$f(N) = \sum_{i \in I_1} v_i + \sum_{h=1}^r v_{i_h}$$

Nel caso B l'output sarà:

Upper bound

$$U(S(I_0, I_1)) = \sum_{i \in I_1} v_i + \sum_{h=1}^k v_{i_h}.$$

Soluzione l'elemento $N = I_1 \cup I_f \in S$ con il relativo valore

$$f(N) = U(S(I_0, I_1)) = \sum_{i \in I_1} v_i + \sum_{h=1}^k v_{i_h}$$

Branching

Vediamo ora di descrivere l'operazione di branching. Dapprima la descriviamo per l'insieme S e poi l'estendiamo agli altri sottinsiemi generati dall'algoritmo.

Supponiamo di trovarci, al termine dell'esecuzione della procedura per il calcolo di $U(S)$, nel caso A (il caso B è un caso banale in cui tutti gli oggetti possono essere inseriti nello zaino). Avremo quindi un indice $r + 1$ che è il primo oggetto per cui la sottrazione successiva dei pesi assume valore negativo.

La regola di branching prescrive di suddividere S nei due sottinsiemi

$$S(\{r + 1\}, \emptyset) \quad \text{e} \quad S(\emptyset, \{r + 1\}),$$

ovvero in un sottinsieme della partizione si aggiunge l'oggetto $r + 1$ all'insieme I_0 , nell'altro lo si aggiunge all'insieme I_1 .

Estensione

Quanto visto per l'insieme S può essere esteso a tutti i sottinsiemi di forma $S(I_0, I_1)$: dato un tale sottinsieme, l'oggetto i_{r+1} che appare nel calcolo dell'upper bound nel caso A viene aggiunto in I_0 in un sottinsieme della partizione di $S(I_0, I_1)$ e in I_1 nell'altro sottinsieme, ovvero la partizione di $S(I_0, I_1)$ sarà data dai seguenti sottinsiemi

$$S(I_0 \cup \{i_{r+1}\}, I_1) \quad \text{e} \quad S(I_0, I_1 \cup \{i_{r+1}\}).$$

Si noti che con questa regola di branching tutti i sottinsiemi che appariranno nell'insieme \mathcal{C} saranno del tipo $S(I_0, I_1)$ e quindi un upper bound per essi potrà sempre essere calcolato tramite la procedura vista.

NB Nel caso B il sottinsieme $S(I_0, I_1)$ è già cancellato in quanto si ha certamente $U(S(I_0, I_1)) \leq LB$ e quindi non si deve definire l'operazione di branching.

Complessità

Nel caso peggiore non si riesce a cancellare alcun nodo $S(I_0, I_1)$ con $I_0 \cup I_1 \subset \{1, \dots, n\}$.

In tal caso il numero di nodi generati è dell'ordine di 2^n .

Quindi, *nel caso peggiore* il comportamento *non* è migliore di quello di un algoritmo di enumerazione completa ma in media funziona molto meglio.

Questo è vero in generale per gli algoritmi branch-and-bound.