Ricerca Operativa: Il problema dello zaino

Simone Parisotto

Un problema di esempio

Il "problema dello zaino", detto anche **Knapsack problem**, è un problema di ottimizzazione combinatoria posto nel modo seguente:

Sia dato uno zaino ed n oggetti, ognuno dei quali caratterizzato da un peso e da un valore di utilità. Il problema si propone di scegliere quali di questi oggetti mettere nello zaino per ottenere il minimo peso tale che il valore di utilità risulti maggiore o uguale ad un valore dato.

Analizziamo un esempio. Questo ci servirà per spiegare come opera l'algoritmo proposto. Sia

$$\begin{aligned} \min \quad & 4x_1 + x_2 + 3x_3 + x_4 \\ & 5x_1 + 3x_2 + 3x_3 + x_4 \geq 4 \\ & x_i = \{0, 1\} \quad i = 1, \ldots, 4 \end{aligned}$$

da risolvere mediante un algortimo di tipo Branch&Bound. Osserviamo innanzi tutto che:

- $c = [4, 1, 3, 1]$ è il vettore dei pesi;
- $a = [5, 3, 3, 1]$ è il vettore delle utilità;
- $b = 4$ è il minimo valore di utilità complessiva che si vuole.

Vogliamo che ad ogni oggetto venga assegnato un valore ("0" se non viene inserito nello zaino, "1" se viene inserito) tale da minimizzare il peso complessivo dello zaino.

Calcoliamo il valore dei rapporti c_i/a_i e ordiniamo i valori in maniera crescente

Indice:	1	2	3	4
Posizione:	2	1	3	4
Rapporto:	$\frac{4}{5}$	$\frac{1}{3}$	1	1

Possiamo quindi riordinare il problema ottenendo:

$$\begin{aligned} \min \quad & x_2 + 4x_1 + 3x_3 + x_4 \\ & 3x_2 + 5x_1 + 3x_3 + x_4 \geq 4 \\ & x_i = \{0, 1\} \quad i = 1, \ldots, 4. \end{aligned}$$

Facciamo un cambio di variabile: $x_1 = y_2$, $x_2 = y_1$, $x_3 = y_3$ e $x_4 = y_4$. Il problema diventa quindi:

$$\begin{aligned} \min \quad & y_1 + 4y_2 + 3y_3 + y_4 \\ & 3y_1 + 5y_2 + 3y_3 + y_4 \geq 4 \\ & y_i = \{0, 1\} \quad i = 1, \ldots, 4. \end{aligned}$$

Ora al problema riordinato applichiamo l'algoritmo *Branch&Bound*:

- $Q = \varnothing$.

- $u = +\infty$ ma possiamo prendere come upper-bound $u = 9$ usando la soluzione $y = [1, 1, 1, 1]$.
- dobbiamo, attraverso la funzione "valuta", calcolare un lower-bound per il problema e lo facciamo tramite un rilasciamento continuo dello stesso, chiamato B:

$$\begin{aligned} B: \qquad \min \quad & y_1 + 4y_2 + 3y_3 + y_4 \\ & 3y_1 + 5y_2 + 3y_3 + y_4 \geq 4 \\ & 0 \leq y_i \leq 1 \quad i = 1, \ldots, 4. \end{aligned}$$

  Abbiamo $k = \max\left\{h : \sum_{i<h} a_i < b\right\} = 2$ che individua la partizione

$$\mathcal{B} = \{2\},\ \mathcal{L} = \{3, 4\},\ \mathcal{U} = \{1\}.$$

  Ricordiamo che la regola per costruire la partizione prevede:

$$\mathcal{B} = \{k\},\ \mathcal{L} = \{k+1, \ldots, n\} \text{ e } \mathcal{U} = \{1, \ldots, k-1\}.$$

  Calcoliamo l'ottimo di tale rilasciamento:

$$\bar{y_i} = \begin{cases} 1 & i \in \mathcal{U} \\ \left(b_i - \sum_{i \in \mathcal{U}} a_i\right)/a_k & i \in \mathcal{B} \\ 0 & i \in \mathcal{L} \end{cases}$$

  che quindi diventa $\bar{y} = [1, \frac{1}{5}, 0, 0]^T$. Ora $\bar{y}$ non è una soluzione intera per cui la funzione "valuta" restituisce $v(B') = c\bar{y}^T = \frac{9}{5}$.
- Poichè $v(B) < u$ allora $Q = \{B, v(B)\}$.
- Seleziono B e $v(B)$ da Q e li rimuovo ottenendo $Q = \varnothing$.
- Separo:

$$\begin{aligned} B_1': \quad y(2) = 0 \qquad \min \quad & y_1 + 0 + 3y_3 + y_4 \\ & 3y_1 + 0 + 3y_3 + y_4 \geq 4 \\ & 0 \leq y_i \leq 1 \quad i = 1, \ldots, 4 \end{aligned}$$

$$\begin{aligned} B_2': \quad y(2) = 1 \qquad \min \quad & y_1 + 4 + 3y_3 + y_4 \\ & 3y_1 + 5 + 3y_3 + y_4 \geq 4 \\ & 0 \leq y_i \leq 1 \quad i = 1, \ldots, 4. \end{aligned}$$

- Applico la funzione valuta a B_1'. Abbiamo $k = 3$ quindi, dai calcoli, si ottiene $\bar{y} = [1, 0, \frac{1}{3}, 0]$. Ora $\bar{y}$ non è una soluzione intera per cui la funzione "valuta" restituisce $v(B_1') = c\bar{y}^T = 2$.
- Poichè $v(B_1') < u$ allora $Q = \{B_1', v(B_1')\}$.
- Applico la funzione valuta a B_2'. Abbiamo $k = 0$ quindi, dai calcoli, si ottiene $\bar{y} = [0, 1, 0, 0]$. È una soluzione intera per cui calcolo $u' = c\bar{y}^T = 4$. Ora $u' < u$ per cui posso aggiornare i vettori soluzione:

$$\begin{aligned} & y = [0, 1, 0, 0]; \\ & u = 4. \end{aligned}$$

- Notiamo come, in entrambe le valutazioni dei problemi separati, l'algoritmo abbia tenuto conto del valore del passo precedente ($y(2)$ in questo caso).

- Seleziono B_1' e $v(B_1')$ da Q e li rimuovo ottenendo $Q = \varnothing$.
- Separo:

$$B_1'': \quad \begin{matrix} y(2) = 0 \\ y(3) = 0 \end{matrix} \qquad \begin{matrix} \min & y_1 + 0 + 0 + y_4 \\ & 3y_1 + 0 + 0 + y_4 \geq 4 \\ & 0 \leq y_i \leq 1 \quad i = 1, \ldots, 4. \end{matrix}$$

$$B_2'': \quad \begin{matrix} y(2) = 0 \\ y(3) = 1 \end{matrix} \qquad \begin{matrix} \min & y_1 + 0 + 3 + y_4 \\ & 3y_1 + 0 + 3 + y_4 \geq 4 \\ & 0 \leq y_i \leq 1 \quad i = 1, \ldots, 4 \end{matrix}$$

- Applico la funzione valuta a B_1''. Abbiamo $k = 5$ quindi, dai calcoli, si ottiene $\bar{y} = [1, 0, 0, 1]$. È una soluzione intera per cui calcolo $u' = c\bar{y}^T = 2$. Ora $u' < u$ per cui posso aggiornare i vettori soluzione:

$$\begin{aligned} y &= [1, 0, 0, 1]; \\ u &= 2. \end{aligned}$$

- Applico la funzione valuta a B_2''. Abbiamo $k = 0$ quindi, dai calcoli, si ottiene $\bar{y} = [0, 0, 1, 0]$. È una soluzione intera per cui calcolo $u' = c\bar{y}^T = 3$. Ma $u' \not< u$ per cui non modifico le soluzioni y e u.
- $Q = \varnothing$ per cui la procedura termina.
- Riordiniamo i risultati ritornando alla x di partenza:

$$\begin{aligned} x &= [0, 1, 0, 1]; \\ u &= 2. \end{aligned}$$

Un possibile codice in Matlab

Vediamo ora un possibile algoritmo, chiamato *knapsack.m* per risolvere il "problema dello zaino" in Matlab:

```
%_____________________________________________________________________
%
% PROBLEMA DELLO ZAINO
%
% Il nostro problema da risolvere e':
%
%                    min c1*x1 + ... + cn*xn
%                        a1*x1 + ... + an*xn >= b
%                           xi = {0 ,1}; i = 1,...,n
%
% Vogliamo che ad ogni oggetto [x1 ,..., xn] venga assegnato un valore :
%
%             0 se non viene inserito nello zaino
%             1 se viene inserito nello zaino
%
% tale da minimizzare il peso complessivo dello zaino.
%
% N.B . I vettori pesi e utilita' devono essere passati come vettori riga.
%
% Chiamata della funzione dal terminale Matlab:
%
%                     [x,min,iter] = zaino (c,a,b)
%
% OUTPUTS:
% x         = vettore 0-1 di dimensioni n x 1
% min       = valore del minimo della funzione
% iter      = numero delle iterazioni eseguite
%
% INPUTS :
% c         = vettore dei pesi
% a         = vettore delle utilita'
% b         = minimo valore di ultilita' complessivamente che si vuole
%
%_____________________________________________________________________
%
```

```
% Autore: Simone Parisotto
% Data: 19/03/2010
%_____________________________________________________________________

function [x,min,iter] = knapsack(c,a,b)

%% CONTROLLI PER LA RISOLUZIONE
if length(c) ~= length(a)
    error('Le lunghezze dei due vettori non coincidono')
end

[rows cols] = size(a);
if rows ~= 1
    error('Dobbiamo lavorare in una dimensione')
end

if ( ~isposintmat(c)) || (~isposintmat(a) )
    error('Uno dei due vettori contiene costanti negative')
end

%% TABELLA PER IL RIORDINO IN ORDINE CRESCENTE DEL RAPPORTO c / a

A = [c;
     a;
     c./a];

% Riordiniamo (oc = posizione indici per ottenere l'ordine crescente)
[s,oc] = sort(c./a);
A = A ( : , oc );

% Ricalcoliamo i nuovi vettori c ed a riordinati
c = A(1,:);
a = A(2,:);

%% INIZIALIZZAZIONI E UPPER BOUND DEL PROBLEMA
x = ones (1,cols);
u = c*x';

% Di seguito inizializzeremo vari insiemi a NaN: questo per evitare di
% lavorare con matrici vuote. Considereremo, quindi, la prima riga
% inizializzata a NaN come la rappresentazione dell'insieme vuoto.

% Insieme dei problemi da risolvere
P  = NaN(1,cols);
% Insieme ausiliario nella costruzione di P (serve per tenere traccia
% degli indici da escludere di volta in volta)
PB = NaN(1,cols);
% Insieme delle valutazioni dei rilasciamenti
VP = NaN ;
% Insieme degli indici da modificare di volta in volta
K  = NaN ;

%% FUNZIONE VALUTA

% Trovo il valore critico k
k = find( cumsum(a)-b <= 0, 1 ,'last' ) + 1;

% Costruiamo t , vettore ausiliario su cui svolgeremo tutte le operazioni
if isempty(k)
    if b/a (1) > 0
        t = [ 1 zeros(1,cols-1) ];
    else
        t = zeros(1 ,cols );
    end
else
    if k > cols
        t = ones(1,cols);
    else
        % Trovo la partizione di base
        U = 1:k-1;      % li metto a 1
        B = k;          % valore critico
        L = k+1: cols; % li metto a 0

        t(U) = 1 ;
        t(B) = ( b-sum(a(U)) )./( a(k) );
        t(L) = 0 ;
    end

end

% Guardo se t e' una soluzione intera
test = mod(t,1) ;
test_index = find(test > 0, 1);

% se e' una soluzione intera aggiorno il vettore soluzione e il minimo
```

```
    if isempty(test_index)

        xnew = t ;
        unew = c*t';

        if unew < u
            u = unew;
            x = xnew;
        end

        % altrimenti calcolo un rilasciamento continuo
    else

        v = c*t';
        nn = PB(1,:);
        % Se la valutazione del problema e' minore del minimo allora
        % aggiorno aggiungendo in coda i problemi aperti e tenedo
        % memoria dell'indice k

        if v < u
            P  = [P;nn];
            VP = [VP; v];
            K  = [K; k];
        end

    end

    %% FUNZIONE SEPARA
    [rows_P cols_P] = size(P);

    % Il ciclo si ferma solo quando P contiene solo la prima riga (cio'
    % equivale a dire che l'insieme e' vuoto e che non ci sono piu'
    % problemi in coda da risolvere)

    iter = 0;
    while rows_P > 1
        % Calcoliamo il numero di iterazioni
        iter = iter + 1;
        % Se la valutazione fatta al passo precedente e' minore di u
        if VP(end) < u

            % Selezioniamo l'ultimo problema della coda
            nn = P(end, : ) ;
            % e lo rimuoviamo dall'insieme (assieme alla sua valutazione)
            P  = P(1:end-1,:);
            VP = VP(1:end-1,:);
            % kk e' l'indice della posizione in cui si trova il numero
            % non intero che servira' per la separazione dei
            % sottoproblemi
            kk = K(end);

            % SOTTO PROBLEMA 1 : t(kk) = 1
            nn(kk) = 1;
            % e lo aggiungiamo all'insieme ausiliario
            PB = [PB;nn];

            % SOTTO PROBLEMA 2 : t(kk) = 0
            nn(kk) = 0 ;
            % e lo aggiungo in coda all'insieme ausiliario
            PB = [PB;nn];
            % l'indice non ci serve piu' e lo rimuoviamo
            K = K(1:end-1,:);
        end

        %% APPLICO LA FUNZIONE VALUTA A PB FINCHE' PB NON RISULTA VUOTO
        [rows_PB cols_PB] = size(PB);
        % Il ciclo si ferma solo quando PB contiene solo la prima riga (cio'
        % equivale a dire che l'insieme e' vuoto e che non ci sono piu'
        % problemi in coda da risolvere)

        while rows_PB > 1
            % Copio i vettori originali in vettori ausiliari per poterli
            % modificare senza perdere i dati originali
            cc = c ;
            aa = a ;
            bb = b ;

            % seleziono l'ultimo vettore di PB che mi tiene conto dei passi
            % gia' effettuati dai cicli in precedenza
            tPB = PB(rows_PB,:);

            %   tPB puo' contentere 1 o 0 a seconda se l'algoritmo
            %   definisca, rispettivamente, se prendere o non prendere un
            %   oggetto. Questo influisce sul termine noto bb che andremo a
            %   modificare opportunamente:
```

```
        for i = 1:cols
            if tPB(i) == 0
                cc(i) = 0 ;
                aa(i) = 0 ;
            else
                if tPB(i) == 1
                    bb = bb-aa(i);
                end
            end
        end

        % Trovo il valore critico k
        k = find( cumsum(aa)-bb <= 0, 1,'last') + 1 ;

        % Costruiamo t, vettore su cui svolgeremo tutte le operazioni
        if isempty(k)
            if bb/aa(1) > 0
                k = 1;
                t = [1 zeros(1,cols-1)];
            else
                t = zeros(1,cols);
            end
        else

            if k > cols
                t = ones(1,cols) ;
            else
                % trovo la partizione di base
                U = 1:k-1;      % li metto a 1
                B = k;          % valore critico
                L = k+1 : cols; % li metto a 0

                t(U) = 1 ;
                t(B) = ( bb-sum(aa(U)) )./( aa(k) ) ;
                t(L) = 0 ;
            end
        end
        % Teniamo conto dei passaggi gia' svolti dall'algoritmo

        for i = 1:cols
            if tPB(i) == 0
                t(i) = 0;
            else
                if tPB(i) == 1
                    t(i) = 1;
                end
            end
        end

        % Guardo se t e' una soluzione
        test = mod(t,1);
        test_index = find(test > 0, 1);

        % se e' una soluzione intera aggiorno il vettore soluzione e il minimo
        if isempty(test_index)

            xnew = t ;
            unew = c*t';

            if unew < u
                u = unew;
                x = xnew;
            end

        else

            v = c*t';
            nn = PB(rows_PB,:);

            % Se la valutazione del problema e' minore del minimo allora
            % aggiorno aggiungendo in coda i problemi aperti e tenedo
            % memoria dell'indice k
            if v < u
                P  = [P;nn];
                VP = [VP; v];
                K  = [K; k];
            end

        end

        rows_PB = rows_PB-1;
        PB = PB(1:rows_PB,:);

    end

```

```
    [rows_P cols_P] = size(P);

end

%% TORNIAMO AL PROBLEMA DI PARTENZA NON RIORDINATO
[s od] = sort(oc);

%% SOLUZIONI
x = x(od);
min = u;

function status = isposintmat(x)
%ISPOSINTMAT returns True if input argument is a matrix of positive integers.

% Copyright 2005-2008 The MathWorks, Inc.
% $Revision: 1.1.8.3 $ $Date: 2008/10/02 18:51:41 $

% Author(s): Qinghua Zhang

status = isnonnegintmat(x) && all(all(x>0));
return
```

Alcune simulazioni dell'algoritmo

Di seguito mostriamo i risultati ottenuti applicando l'algoritmo *knapsack.m* ad alcuni problemi:

```
-----------------------------------------------------------------

Esempio 1

c    =  4     1     3     1
a    =  5     3     3     1
b    =  4

x    =  0     1     0     1
min  =  2

iter =  2


-----------------------------------------------------------------

Esempio 2

c    =  5     7     2     4     7     1     3
a    =  10    10    10    10    10    10    10
b    =  30

x    =  0     0     1     0     0     1     1
min  =  6

iter =  0


-----------------------------------------------------------------

Esempio 3

c    =  4     1     3     1     2     3     4
a    =  5     3     3     1     7     1     4
b    =  10

x    =  0     1     0     0     1     0     0
min  =  3
```

```
iter =  0

-----------------------------------------------------------------

Esempio 4

c    =  7     3     5    11    17     2
a    =  4     3     6     5     3     2
b    =  8

x    =  0     0     1     0     0     1
min  =  7

iter =  1

-----------------------------------------------------------------

Esempio 5

c    =  3     5     2     6     8    10     2     7     1
a    =  2     3     2     3     3     2     2     2     3
b    =  7

x    =  0     0     1     0     0     0     1     0     1
min  =  5

iter =  0

-----------------------------------------------------------------

Esempio 6

c    =  2     2     2     2     1     2     2     2     1
a    =  7     4     5     8     2     1     4     3    10
b    =  12

x    =  0     0     0     0     1     0     0     0     1

min  =  2
iter =  4

-----------------------------------------------------------------
```