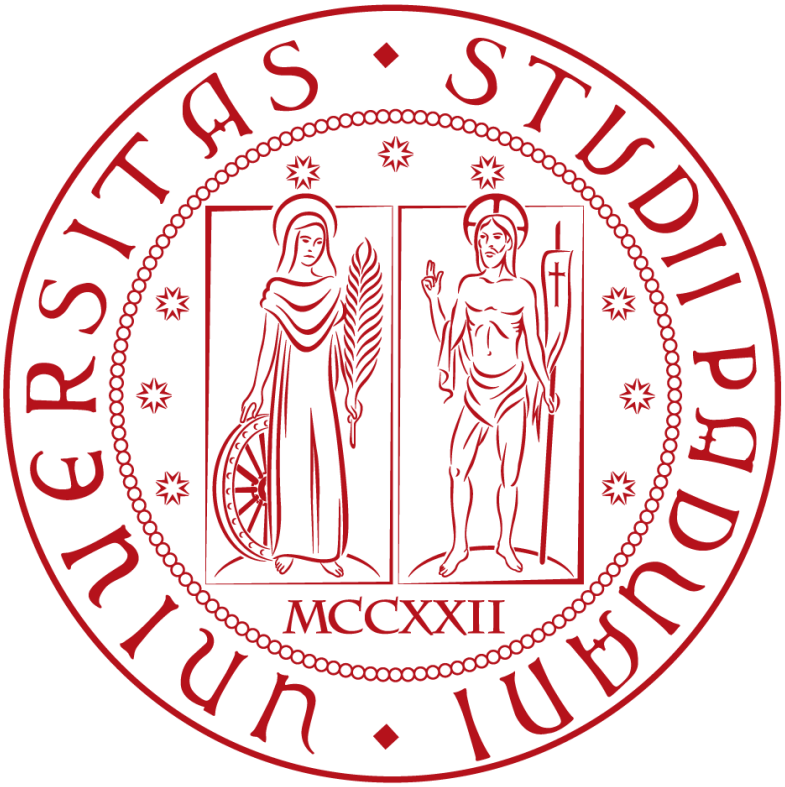


Relazione Progetto di Ricerca Operativa a.a. 2016/2017

Studente	**Matricola**
Alessia Bragagnolo	1100817

Indice

Abstract

Il problema preso in esame è simile al tipico problema dello zaino binario, dove gli oggetti che si hanno a disposizione possono avere diversa dimensione e apportare diverso profitto.
Diversamente dal classico problema però, lo zaino preso in considerazione ha tre dimensioni: larghezza(x), altezza(y) e profondità(z); bisognerà quindi prendere in considerazione tutti i vincoli sul posizionamento degli
oggetti e effettuare opportune scelte anche per quanto riguarda la loro rotazione all'interno dello zaino.
Prima verrà presentato il problema in forma testuale con un esempio e poi verrà implementato il modello descritto tramite il linguaggio AMPL e ne verrà data la soluzione ottima.

Descrizione del problema

Dato uno zaino di dimensioni fissate:

Dimensione zaino	x	y	z
	612	492	455

e un insieme di oggetti parallelepipedi numerati, delle seguenti dimensioni:

Oggetto	**x**	**y**	**z**	**profitto apportato dall' inserimento dell' oggetto**
1	345	283	65	24
2	370	354	111	5
3	292	167	109	8
4	101	93	52	9
5	256	98	103	20
6	500	430	60	22

Si vuole conoscere l'inserimento di quali oggetti nello zaino massimizza il profitto. Per questo è necessario valutare l'inserimento di un oggetto in ogni sua possibile rotazione; trattandosi di un insieme di parallelepipedi le rotazioni possibili sono solamente sei.

Inoltre si vogliono conoscere le coordinate in cui ogni oggetto andrebbe posizionato, prendendo come riferimento l'angolo posteriore inferiore sinistro, tali coordinate dovranno essere compatibili con le dimensioni dello zaino e la rotazione in cui l'oggetto viene inserito al suo interno.

Modello matematico

Variabili decisionali

- t_i: $i \in J\ \{1, 2, 3, 4, 5, 6\}$ variabile binaria che vale 1 se l'oggetto i viene inserito nello zaino, 0 altrimenti;
- $b_{i,j,d}$: $i \in J\ \{1, 2, 3, 4, 5, 6\}$, $j \in J\ \{1, 2, 3, 4, 5, 6\}$, $d \in D\ \{x, y, z\}$ variabile binaria che vale 1 se l'oggetto i precede nella dimensione d l'oggetto j, 0 altrimenti;
- $p_{i,k}$: $i \in J\ \{1, 2, 3, 4, 5, 6\}$, $k \in R\{uno, due, tre, quattro, cinque, sei\}$ variabile binaria che vale 1 se l'oggetto i inserito nello zaino usando la rotazione k, 0 altrimenti;
- $X_{i,d}$: $i \in J\ \{1, 2, 3, 4, 5, 6\}$, $d \in D\ \{x, y, z\}$ variabile intera che descrive le coordinate dello spigolo dell'oggetto i, nella dimensione d, se questo viene inserito nello zaino, altrimenti posta a zero.

Parametri

- $s_{i,d,r}$: $i \in J\ \{1, 2, 3, 4, 5, 6\}$, $d \in D\ \{x, y, z\}$, $r \in R\ \{uno, due, tre, quattro, cinque, sei\}$, descrive la dimensione d, dell'oggetto i, nella rotazione r;
- po_i: $i \in J\ \{1, 2, 3, 4, 5,6\}$ descrive il profitto ricavabile dall'oggetto i;
- D_d: $d \in D\ \{x, y, z\}$ descrive il valore della dimensione d dello zaino;
- BigM, intero arbitrariamente grande;

Vincoli e funzione obiettivo

F.o. max:

$$\sum_{j=1}^{6} po[j] * t[j]$$

Ogni profitto viene moltiplicato per t, che rappresenta la presenza dell'oggetto j nello zaino.

Se ogni oggetto fosse inserito nello zaino allora:

$$\sum_{j=1}^{6} po[j] * t[j] = \sum_{j=1}^{6} po_{[j]}$$

Vincoli:

- Il vincolo denominato capienza verifica che il totale del volume degli oggetti inseriti nello zaino non superi il totale del volume dello zaino e viene così implementato:

$$\sum_{j=1}^{6} s[i, first(D), first(R)] * s[i, member(2, D), first(R)] * s[i, last(D), first(R)]$$
$$\leq S[first(D)] * S[last(D)] * S[member(2, D)]$$

  Essendo un vincolo sul volume dell'oggetto non ha importanza la rotazione in cui viene inserito nel contenitore.

- Il vincolo denominato knapsack_limit serve a controllare che il posizionamento di un oggetto, in una certa rotazione nelle coordinate in esame non sfori la dimensione dello zaino

$$X[i, d] + \sum_{r=uno}^{sei} s[i, d, r] * p[i, r] \ \leq S[d]\ \forall\ i = 1 \dots 6, \forall\ d = x, y, z$$

- Il vincolo denominato coordinate_control un vincolo logico legato alla variabile logica t: impone che se un oggetto non viene preso (quindi t assume valore 0) le coordinate valgano 0 e viene così implementato:

$$X[i,d] \leq BigM * t[i] \; \forall i = 1 \dots 6 \, , \forall \, d = x, y, z$$

- Il vincolo limit_position_i impone che se un oggetto i precede un oggetto j nella dimensione d la sua lunghezza in tale dimensione a partire dalla coordinata del vertice e determinata in base alla rotazione assunta attualmente non superi la coordinata del vertice dell'oggetto j, cioè che gli oggetti i e j non si sovrappongano ed è cosi implementato:

$$X[i,d] + \sum_{r=uno}^{sei} s[i,d,r] * p[i,r] \leq X[j,d] + BigM * (1 - b[i,j,d])$$

$$i < j \;\; \forall i = 1 \dots 6 \, , \forall j = 1 \dots 6 \, , \forall \, d = x, y, z$$

- Il vincolo denominato limit_position_j ha lo stesso significato del vincolo precedente, viene però preso in esame l'oggetto *j,* il vincolo viene così implementato:

$$X[j,d] + \sum_{r=uno}^{sei} s[j,d,r] * p[j,r] \leq X[i,d] + BigM * (1 - b[j,i,d])$$

$$i < j \;\; \forall i = 1 \dots 6 \, , \forall j = 1 \dots 6 \, , \forall \, d = x, y, z$$

- Il vincolo controllo_adiacente_i un è vincolo logico che lega la presenza dell'oggetto nello zaino con l'ordine in cui viene inserito, ovviamente un oggetto non inserito non può precedere o seguire un altro oggetto, il vincolo viene così implementato:

$$b[i,j,d] \leq t[i]$$
$$\forall i = 1 \dots 6 \, , \forall j = 1 \dots 6 \, , \forall \, d = x, y, z$$

- Vale anche il viceversa:

$$b[j,i,d] \leq t[j]$$
$$\forall i = 1 \dots 6 \, , \forall j = 1 \dots 6 \, , \forall \, d = x, y, z$$

- Il vincolo denominato limit_rotation vincola l'inserimento di un oggetto in una sola possibile rotazione nello zaino ed è così implementato:

$$\sum_{r=uno}^{sei} p[i,r] \leq 1 * t[i]$$
$$\forall i = 1 \dots 6$$

- Il vincolo denominato min_rotation forza l'inserimento in almeno una rotazione, nel caso in cui l'oggetto venga scelto ed è così implementato:

$$\sum_{r=uno}^{sei} p[i,r] \geq t[i]$$
$$\forall i = 1 \dots 6$$

- Il vincolo di overlap serve a controllare che se un oggetto precede l'altro, almeno uno dei due sia inserito:

$$\sum_{d=x}^{z} (b[i,j,r] + b[j,i,r]) \geq t[i] + t[j] - 1$$
$$\forall i = 1 \dots 6 \,, \forall j = 1 \dots 6$$

- Il vincolo limit_spigolo si assicura che le coordinate dello spigolo di un oggetto siano positive:

$$X[j,d] \geq 0$$
$$\forall j = 1 \dots 6 \,, \forall\, d = x, y, z$$

Implementazione in AMPL

File.mod

```
set J; #insieme dei prodotti
set D ordered; #insieme delle dimensioni
set R ordered; # insieme delle rotazioni

param s{J,D,R} ; #la dimensione dell'oggetto J, nella dimensione D nella rotazione R
param po{J} ; #profitto ricavato dalla scelta dell'oggetto J
param S{D} ; #dimensione dello zaino nella dimensione D
param BigM;


var t{i in J} binary; #binaria che assume valore 1 se  si prende l'oggetto j
var b{i in J, j in J, d in D} binary;#binaria che assume valore 1 se l'oggetto i
precede l'oggetto j nella dimensione d
var p{i in J, k in R} binary; #binaria che assume valore 1 se l'oggetto j e' usato
nella rotazione R
var X{j in J, d in D} integer;


maximize profitto: sum{j in J} po[j]*t[j];


subject to capienza: sum {j in J}
s[j,first(D),first(R)]*s[j,member(2,D),first(R)]*s[j,last(D),first(R)]*t[j]<=
S[first(D)]*S[member(2,D)]*S[last(D)];


subject to knapsack_limit{d in D ,i in J}: X[i,d]+ sum {r in R} s[i,d,r]*p[i,r] <=
S[d];

subject to coordinate_control {d in D ,i in J}: X[i,d]<= BigM*t[i];

subject to limit_position_i{d in D ,i in J, j in J: i<j}: X[i,d]+ sum {r in
R}(s[i,d,r]*p[i,r]) <= X[j,d] + BigM *(1-b[i,j,d]);

subject to limit_position_j{d in D ,i in J, j in J: i<j}: X[j,d]+ sum {r in
R}(s[j,d,r]*p[j,r]) <= X[i,d] + BigM *(1-b[j,i,d]);

subject to controllo_adiacente_i{d in D ,i in J, j in J}: b[i,j,d]<=t[i];

subject to controllo_adiacente_j{d in D ,i in J, j in J}: b[j,i,d]<=t[j];

subject to overlap {j in J ,i in J}: sum {d in D} (b[i,j,d] + b[j,i,d]) >= t[i]+t[j]-1;

subject to limit_rotation{i in J}: sum {r in R} p[i,r] <= 1;

subject to min_rotation{i in J}: sum {r in R} p[i,r] >=t[i];

subject to limit_spigolo{i in J, d in D}:X[i,d] >= 0;
```

File.run

```
reset; # cancella dati memorizzati

model problema.mod; # carica il modello
data problema.dat; # carica i dati

option solver cplex; # scelta del motore di ottimizzazione
solve; # risolve il modello

printf "il profitto ricavato e'";
display profitto;
printf "\n";

printf "i seguenti oggetti venogono inseriti nello zaino";
printf "\n";

for {i in J} {
   if( t[i]>0) then {
      printf "oggetto ";
      print i;
      printf "inserito nelle coordinate ";
      printf "\n";
      printf "x= "; print X[i, first(D)];
      printf "y= "; print X[i, member(2,D)];
      printf "z= "; print X[i, last(D)];
      printf "\n";
      printf "inserito nella rotazione ";
      for{r in R}{
            if(p[i,r]>0) then{
                  print r;
                  printf "l'oggetto e'quindi largo ";
                  print s[i,first(D),r];
                  printf "alto ";
                  print s[i,member(2,D),r];
                  printf "profondo ";
                  print s[i,last(D),r];
            }
      }
   }
   printf "\n";

}
```

L'esecuzione del programma avviene dando il comando `include problema.run`.

Risultato del problema

Il modello implementato fornisce il seguente risultato:

Oggetti inseriti	1	3	5	6
Larghezza	65	292	276	500
Altezza	283	167	307	430
Profondita'	345	109	126	255
Coordinata x	547	0	271	47
Coordinata y	0	325	0	62
Coordinata z	0	275	258	0

Osservazioni

Con il modello precedente non vi erano vincoli sulla posizione in cui vengono collati gli oggetti. Una possibile raffinazione del modello consiste nel cercare di rendere più compatto l'inserimento degli oggetti nel contenitore.

Per far ciò viene così cambiata la funzione obiettivo:

$$max: S[first(D)] * S[last(D)] * S[member(2, D)] * \sum_{j=1}^{6} po[j] * t[j] - \sum_{i=1}^{6} \sum_{d=x}^{y} X[i, d]$$

Per testare che il modello compatti effettivamente gli oggetti nella soluzione i profitti degli oggetti stessi sono stati posti come unitari e si è invece deciso di massimizzare il volume occupato nello zaino.

Per far ciò sono state introdotte i seguenti nuovi parametri:

- pu_i: $i \in J\ \{1, 2, 3, 4, 5, 6\}$ parametro associato a dei pesi unitari;
- pv_i: $i \in J\ \{1, 2, 3, 4, 5, 6\}$ parametro che descrive il volume dell'oggetto i;
- min_po: parametro che assume il valore del volume dell'oggetto di volume minimo;
- new_po_i: $i \in J\ \{1, 2, 3, 4, 5, 6\}$ peso assegnato all'oggetto i, ottenuto come rapporto tra il volume dell'oggetto stesso e min_po.

Si sono provate quindi le seguenti tre modifiche alla funzione obiettivo:

$$\boldsymbol{A}: max: S[first(D)] * S[last(D)] * S[member(2, D)] * \sum_{j=1}^{6} pu[j] * t[j] - \sum_{i=1}^{6} \sum_{d=x}^{y} X[i, d]$$

$$\boldsymbol{B}: max: S[first(D)] * S[last(D)] * S[member(2, D)] * \sum_{j=1}^{6} pv * t[j] - \sum_{i=1}^{6} \sum_{d=x}^{y} X[i, d]$$

$$\boldsymbol{C}: max: S[first(D)] * S[last(D)] * S[member(2, D)] * \sum_{j=1}^{6} new\_po * t[j] - \sum_{i=1}^{6} \sum_{d=x}^{y} X[i, d]$$

Nuovo modello

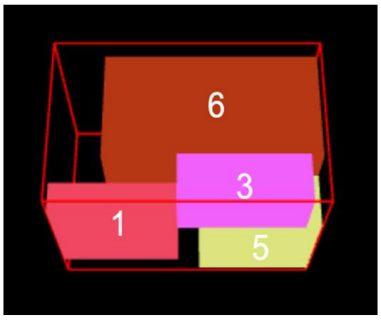
```
set J; #insieme dei prodotti
set D ordered; #insieme delle dimensioni
set R ordered; # insieme delle rotazioni

param s{J,D,R} ; #la dimensione dell'oggetto J, nella dimensione D nella rotazione R
param po{J} ; #profitto ricavato dalla scelta dell'oggetto J
param S{D} ; #dimensione dello zaino nella dimensione D
param BigM;
param pu{J} ;
param pv{j in
J}=s[j,first(D),first(R)]*s[j,member(2,D),first(R)]*s[j,last(D),first(R)]; #volume di
ogni oggetto
param min_po= min {i in J} pv[i];#oggetto piu'piccolo
param new_po{j in J}=pv[j]/min_po;
var t{i in J} binary; #binaria che assume valore 1 se  si prende l'oggetto j
var b{i in J, j in J, d in D} binary;#binaria che assume valore 1 se l'oggetto i
precede l'oggetto j nella dimensione d
var p{i in J, k in R} binary; #binaria che assume valore 1 se l'oggetto j e' usato
nella rotazione R
var X{j in J, d in D} integer;


#maximize profitto_compact:(S[first(D)]*S[member(2,D)]*S[last(D)])*(sum{j in J}
(pu[j]*t[j])) - sum{i in J} sum{d in D}X[i,d];

maximize profitto_compact:(S[first(D)]*S[member(2,D)]*S[last(D)])*(sum{j in J}
(pv[j]*t[j])) - sum{i in J} sum{d in D}X[i,d];

#maximize profitto_compact:(S[first(D)]*S[member(2,D)]*S[last(D)]) *( sum{j in J}
(new_po[j]*t[j])) - sum{i in J}sum{d in D}X[i,d] ;

subject to capienza: sum {j in J}
s[j,first(D),first(R)]*s[j,member(2,D),first(R)]*s[j,last(D),first(R)]*t[j]<=
S[first(D)]*S[member(2,D)]*S[last(D)];

subject to knapsack_limit{d in D ,i in J}: X[i,d]+ sum {r in R} s[i,d,r]*p[i,r] <=
S[d];
subject to coordinate_control {d in D ,i in J}: X[i,d]<= BigM*t[i];
subject to limit_position_i{d in D ,i in J, j in J: i<j}: X[i,d]+ sum {r in R}
s[i,d,r]*p[i,r] <= X[j,d] + BigM *(1-b[i,j,d]);
subject to limit_position_j{d in D ,i in J, j in J: i<j}: X[j,d]+ sum {r in R}
s[j,d,r]*p[j,r] <= X[i,d] + BigM *(1-b[j,i,d]);
subject to controllo_adiacente_i{d in D ,i in J, j in J}: b[i,j,d]<=t[i];
subject to controllo_adiacente_j{d in D ,i in J, j in J}: b[j,i,d]<=t[j];
subject to overlap {j in J ,i in J}: sum {d in D} (b[i,j,d] + b[j,i,d]) >= t[i]+t[j]-1;
subject to limit_rotation{i in J}: sum {r in R} p[i,r] <= 1*t[i];
subject to min_rotation{i in J}: sum{r in R} p[i,r] >=t[i];
subject to limit_spigolo{i in J, d in D}:X[i,d] >= 0;
```

Esempi

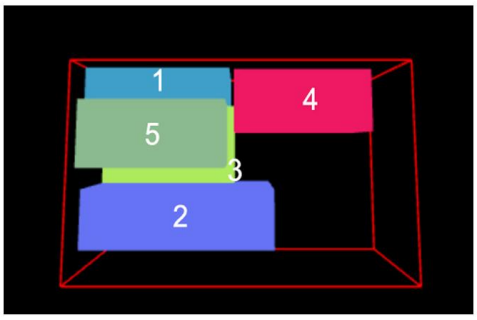
Di seguito alcuni esempi su cui è stato provato il modello.

Esempio 1:

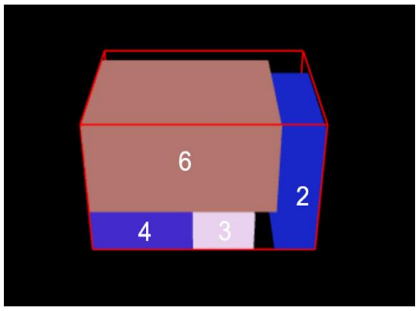
	Oggetti	1	2	3	4	5	6
	Larghezza	345	370	292	315	307	500
	Altezza	283	354	167	276	276	430
	Profondità	65	111	109	123	126	255
Iniziale	x	0	-	283	-	336	112
	y	58	-	0	-	167	62
	z	0	-	0	-	0	129
A	x	0	0	0	292	0	-
	y	0	0	0	0	109	-
	z	0	232	65	0	65	-
B	x	-	500	276	0	-	0
	y	-	0	0	0	-	0
	z	-	0	0	0	-	123
C	x	-	0	111	111	-	111
	y	-	0	0	167	-	0
	z	-	0	0	0	-	123



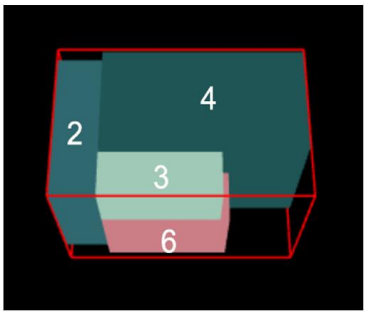
f.o. iniziale



A



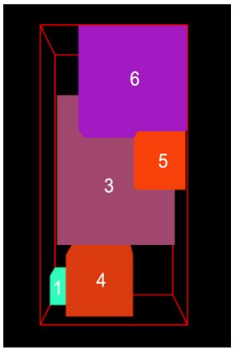
B



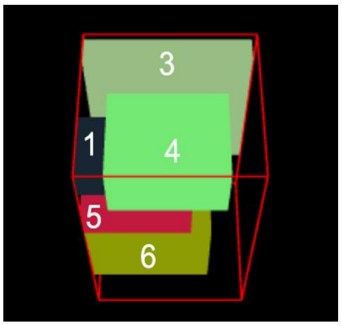
C

Esempio 2:

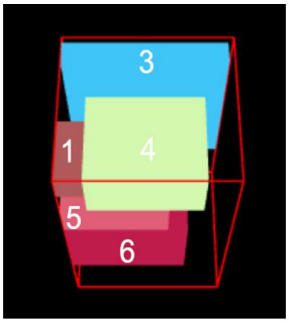
	Oggetti	1	2	3	4	5	6
	Larghezza	113	425	470	323	302	349
	Altezza	165	498	594	215	345	345
	Profondità	75	243	60	235	360	360
Iniziale	x	0	-	16	75	323	126
	y	0	-	215	0	443	641
	z	0	-	13	0	73	73
A	x	0	-	0	75	0	0
	y	0	-	0	0	215	378
	z	0	-	345	0	0	0
B	x	0	-	0	75	0	0
	y	0	-	0	0	215	378
	z	0	-	345	0	0	0
C	x	0	0	0	-	0	75
	y	0	345	0	-	594	0
	z	60	0	0	-	0	60



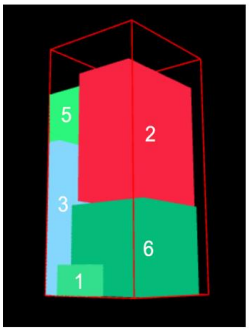
f.o. iniziale



A



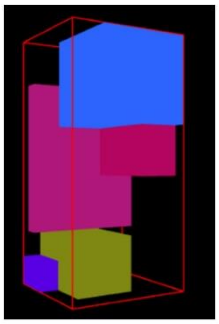
B



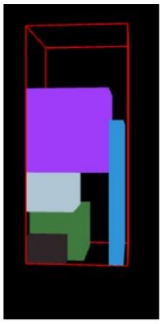
C

Esempio 3:

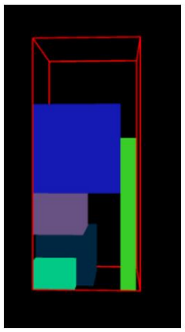
	Oggetti	1	2	3	4	5	6
	Larghezza	276	285	335	500	295	327
	Altezza	172	62	288	430	290	77
	Profondità	50	181	82	255	57	67
Iniziale	x	0	0	247	0	500	535
	y	0	0	0	0	0	425
	z	334	272	302	0	0	0
A	x	181	0	181	0	516	500
	y	0	0	172	0	0	0
	z	255	255	255	0	67	0
B	x	327	327	0	57	0	0
	y	62	0	67	62	0	0
	z	0	0	0	82	82	0
C	x	327	327	0	57	0	0
	y	62	0	67	62	0	0
	z	0	0	0	0	82	0



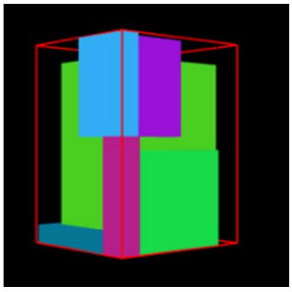
f.o. iniziale



A



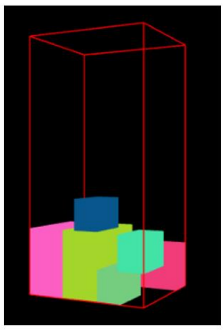
B



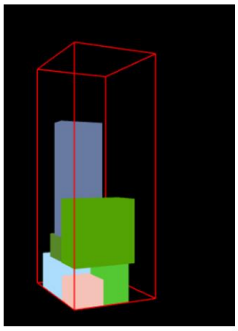
C

Esempio 4:

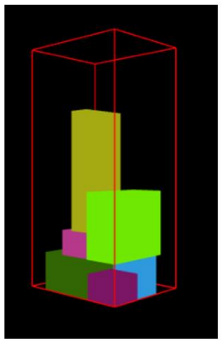
	Oggetti	1	2	3	4	5	6
	Larghezza	162	499	156	156	279	279
	Altezza	141	191	126	126	273	273
	Profondità	115	95	106	126	164	164
Iniziale	x	0	0	208	102	208	372
	y	126	0	273	0	0	0
	z	0	277	0	0	0	0
A	x	0	164	164	0	164	0
	y	0	270	164	0	0	162
	z	126	0	0	0	0	0
B	x	0	164	164	0	164	0
	y	0	270	164	0	0	162
	z	126	0	0	0	0	0
C	x	0	164	164	0	164	0
	y	0	270	164	0	0	162
	z	126	0	0	0	0	0



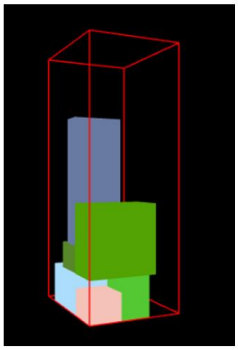
f.o. iniziale



A



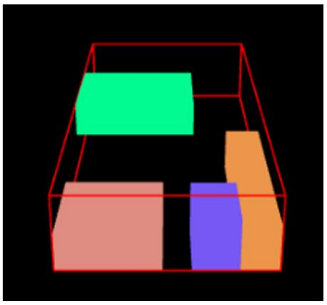
B



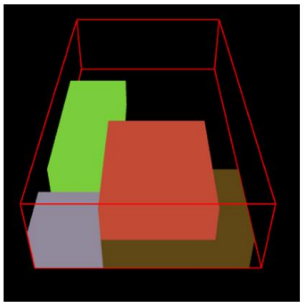
C

Esempio 5:

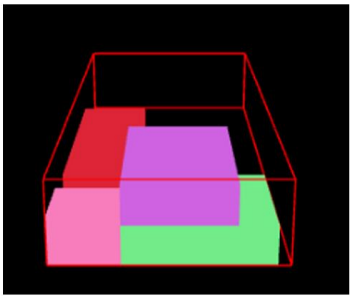
	Oggetti	1	2	3	4	5	6
	Larghezza	345	345	485	485	499	162
	Altezza	257	257	417	417	191	141
	Profondità	95	95	318	318	95	115
Iniziale	x	0	0	-	-	441	326
	y	499	0	-	-	0	0
	z	107	0	-	-	0	0
A	x	162	162	-	-	0	0
	y	0	0	-	-	257	0
	z	0	95	-	-	0	0
B	x	162	162	-	-	0	0
	y	0	0	-	-	257	0
	z	0	95	-	-	0	0
C	x	162	162	-	-	0	0
	y	0	0	-	-	257	0
	z	0	95	-	-	0	0



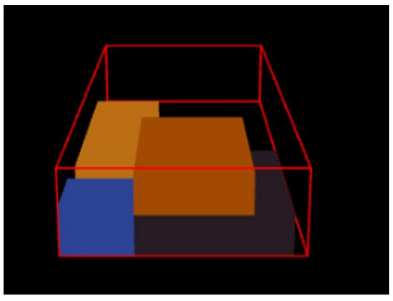
f.o. iniziale



A



B



C