UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA GESTIONALE

TESINA PER L'ESAME DI:
RICERCA OPERATIVA II – PROBLEM SOLVING PER IL MANAGEMENT

A cura di:

Dario Fiore
M62002987

Traccia 30

Si vuole realizzare una libreria, composta da diversi scaffali, allo scopo di contenere (allineandoli) alcuni volumi di larghezze e altezze diverse e note a priori (valori riportati in tabella). La larghezza degli scaffali è fissata (100 cm) ed uguale per tutti gli scaffali. Quindi, la libreria ha una larghezza di 100 cm. L'altezza di ogni scaffale invece non lo è, ma deve essere scelta in modo che sia sufficiente a contenere tutti i libri che si vogliono mettere sullo stesso scaffale. L'altezza di ogni scaffale determina ovviamente l'altezza della libreria. Più la libreria è altra, più costa. L'obiettivo è quindi realizzare la libreria con il minor costo di costruzione possibile, al fine di contenere tutti i libri, nel rispetto della larghezza fissata.

Id_Libro	Larghezza	Altezza
1	20	25
2	15	20
3	10	17
4	10	18
5	7	15
6	5	15
7	14	19
8	15	17
9	10	17
10	7	20
11	5	18
12	20	10
13	7	12
14	5	14
15	10	14
16	5	16
17	15	16
18	10	18
19	8	15
20	9	15

Nel caso in cui per ogni Id di un libro, fossero presenti un numero di unità pari all'ID del libro, come deve essere modificato il modello?

1. Modello matematico del problema in forma implicita

Il presente studio si propone di progettare e realizzare una libreria destinata ad ospitare una serie di libri caratterizzati da dimensioni di larghezza e altezza diverse, predefinite e riportate in tabella. La larghezza degli scaffali è standardizzata a 100 cm, definendo pertanto la larghezza complessiva della libreria.
L'altezza di ciascuno scaffale, invece, è variabile e dipende dalla necessità di accogliere tutti i libri assegnati allo stesso. È importante osservare che l'altezza di ogni scaffale influisce direttamente sull'altezza totale della libreria, la quale incide sui costi di costruzione.
L'obiettivo consiste dunque nell'ottimizzare il processo di realizzazione della libreria, minimizzando i costi di costruzione, pur garantendo l'inclusione di tutti i libri e rispettando la larghezza prefissata.

Nella fase successiva del problema, ciascun ID libro sarà associato all'unità corrispondente indicata dal medesimo ID (ad esempio, ID 1=1 libro, ID 2=2 libri, ID 3=3 libri etc...).

Si è pensato di vedere il problema come un particolare problema di localizzazione puntuale, in quanto in questo contesto, gli "impianti" corrispondono agli scaffali della libreria che devono essere costruiti, e l'obiettivo è minimizzare i costi complessivi di costruzione.

Gli insiemi da tenere in considerazione sono:

- L'insieme I dei libri di cardinalità n.
- L'insieme J degli scaffali che in realtà non è dato dalla traccia però se per assurdo mettessimo un libro su ogni scaffale l'insieme avrà cardinalità massima m = n.

$$I = \{1....n\} \qquad \textit{insieme degli id dei libri } (20)$$
$$J = \{1....m\} \qquad \textit{insieme massimo degli scaffali } (20)$$

- Dati del problema :

- l_i = larghezza del libro i
- a_i = altezza del libro i
- b_i = id del libro i
- D = larghezza di ogni scaffale pari a 100cm, essendo uguale per tutti gli scaffali non utilizziamo il pedice j.

- Variabili decisionali:

- Una variabile binaria x_{ij} che assume valore 1 se il libro i è presente sullo scaffale j, 0 altrimenti.

- Una variabile continua y_{ij} che rappresenta l'altezza dello scaffale j, è una variabile ovviamente positiva perché legata a una dimensione.

$$x_{ij} -> \begin{cases} 1\ se\ il\ libro\ i\ \text{è}\ situato\ sullo\ scaffale\ j \\ 0\ altrimenti \end{cases}$$
$$y_j \quad -> \ altezza\ dello\ scaffale\ j$$

  - Formulazione

- ***Funzione obiettivo***

$$min \sum_{j \in J} y_j$$

La funzione obiettivo è quella di minimizzare i costi totali di costruzione della libreria, che dipendono direttamente dall'altezza degli scaffali. Quindi, l'obiettivo è trovare la configurazione degli scaffali che minimizzi la somma dei costi di costruzione degli scaffali utilizzati.

- ***Vincoli sulla larghezza degli scaffali***

$$\sum_{i \in I} l_i x_{ij} \leq 100 \qquad \forall\, j \in J$$

Questi sono **m** vincoli, tanti quanti saranno gli scaffali utilizzati e impongono che la somma delle larghezze dei libri posizionati su uno stesso scaffale non superi la larghezza massima dello scaffale stesso che è fissata a 100cm.

- ***Vincoli sulle afferenze dei libri***

$$\sum_{j \in J} x_{ij} = 1 \qquad \forall\, i \in I$$

Questi sono **n** vincoli, tanti quanti sono i libri, e impongono che ogni libro debba essere posizionato su uno e un solo scaffale.

Vincolo sulle altezze degli scaffali:

$$y_j \geq a_i x_{ij} \qquad \forall\, j \in J,\ i \in I$$

Questi sono **n*m** vincoli, ossia fissato il libro i e lo scaffale j, se il libro i è presente sullo scaffale j, allora $x_{ij} = 1$ e quindi l'altezza dello scaffale dovrà essere maggiore o uguale all'altezza di quel libro. Questo per tutti i libri e per tutti gli scaffali.

- ***<u>m</u> vincoli di interezza e <u>n*m</u> di binarietà rispettivamente per la variabile y_j e x_{ij} :***

$$y_j \geq 0 \qquad \forall\, j \in J$$
$$x_{ij} \in \{0, 1\} \qquad \forall\, j \in J,\ i \in I$$

- ***Vincoli aggiuntivi:***

$$y_{j+1} \leq y_j \qquad \forall\, j \in J$$

Questi sono un gruppo di **m** vincoli che non sono indispensabili, in assenza di questi la soluzione ottima calcolata sarebbe la stessa però potrei avere soluzioni in cui gli scaffali utilizzati sono elencati in maniera disordinata. Quindi questi vincoli mi consentono di costruire scaffali in maniera ordinata, in particolare ordina le altezze degli scaffali in modo tale da far diventare ultimi gli scaffali con altezza nulla.

2. Codice python e implementazione del modello su Gurobi

Per la risoluzione del problema si è deciso di utilizzare il software Gurobi, che prevede la scrittura del modello in linguaggio Python.

Il codice utilizzato si articola nei seguenti punti:

- Lettura dei dati di input da un file di testo esterno;
- Inizializzazione del modello e definizione delle variabili;
- Definizione della funzione obiettivo;
- Aggiunta dei vincoli;
- Risoluzione del problema mediante libreria gurobi;
- Stampa a schermo della soluzione;

2.1 Lettura dei dati di input

I dati del problema sono riportati in un file di testo chiamato “Esame.rtf”, pertanto, tramite la funzione “open” di python è stato possibile aprire il file in modalità di lettura e lavorarci sopra.

Una volta aperto il file, il codice per la lettura dei dati è il seguente:

LETTURA DEI PARAMETRI DA FILE

```
file = open('esame.rtf','r')
filelines = file.readlines()

num_line = 1
line = filelines[num_line].split()
numlib = range(int(line[0]))
numscaf = range(20)

num_line = num_line + 2
SIZE = []
for i in numlib:
    sizeline = []
    line = filelines[num_line].split()
    for j in range(1,3):
        sizeline.append(int(line[j]))
    SIZE.append(sizeline)
    num_line = num_line +1
```

Creando la lista *filelines* e utilizzando l’istruzione *file.readlines* python leggerà tutte le righe del file ‘esame.rtf’ e le memorizzerà nella lista *filelines*.

Successivamente ho creato un indice di riga *num_line* che rappresenta il numero della riga in cui mi trovo (gli indici delle liste in python partono da 0).

Nel secondo blocco, ponendo num_line = 1, mi sono posizionato sulla seconda riga della lista *filelines* e ho aggiunto la stringa alla lista '*line*'. Infine, siccome ho bisogno di un range di **interi**, ho creato una lista *numlib* al cui interno ho messo il valore intero della lista *line* e tramite la funzione *range* ho ottenuto una lista formata da una sequenza di numeri da 0 a 19.

Nel terzo blocco ho aggiornato l'indice di riga a 3 e mi sono posizionato sulla quarta riga del file di testo corrispondente alla prima riga della matrice delle dimensioni dei libri.

A questo punto, siccome la matrice è una lista di liste, ho preliminarmente inizializzato una lista vuota *SIZE* in cui alla fine dovrò aggiungere tante liste (ognuna delle quali contiene le dimensioni dell'i-esimo libro) quanti sono i libri. Successivamente, tramite due cicli for uno dentro l'altro, ho:

- creato una lista vuota *sizeline* per l'i-esima riga del file e aggiunto alla lista *line* la lista di stringhe corrispondente all'i-esima riga;
- Con il secondo ciclo for ho aggiunto alla lista *sizeline* gli interi corrispondenti alle stringhe contenute nella lista *line*;
- Uscito dal secondo ciclo for torno nel primo ciclo in cui aggiungo alla lista *SIZE* la lista *sizeline*, incremento l'indice di 1 e torno al passo iniziale;

2.2 Inizializzazione del modello e definizione delle variabili

RICHIAMA LA LIBRERIA GUROBI E INIZIALIZZA IL MODELLO

```
import gurobipy as gp
from gurobipy import GRB

# Inizializza il modelo
mod = gp.Model('Esame')
```

DEFINIZIONE VARIABILI

```
x = mod.addVars(numlib,numscaf, vtype = GRB.BINARY, name = 'x')
y = mod.addVars(numscaf, vtype = GRB.CONTINUOUS, name = 'y')
```

2.3 Funzione obiettivo

```
obj = mod.setObjective(gp.quicksum(y[j] for j in numscaf), GRB.MINIMIZE)
```

2.4 Vincoli

2.4.1 Vincoli di afferenza

```
mod.addConstrs(gp.quicksum(x[i,j] for j in numscaf) == 1 for i in numlib)
```

2.4.2 Vincoli sulle larghezze degli scaffali

```
mod.addConstrs(gp.quicksum(SIZE[i][0]*x[i,j] for i in numlib) <= 100 for j in numscaf)
```

2.4.3 Vincoli sulle altezze degli scaffali

```
mod.addConstrs((y[j])>= SIZE[i][1]*x[i,j] for j in numscaf for i in numlib)
```

2.4.4 Vincoli aggiuntivi sull'ordine degli scaffali

```
mod.addConstrs((y[j+1])<= y[j] for j in range(19))
```

2.5 Risoluzione del problema mediante libreria gurobi

```
mod.optimize()
if mod.status == GRB.OPTIMAL:
    print (mod.objval)
    mod.write('esame.sol')
mod.write('esame.lp')
```

Tramite l'istruzione *mod.optimize()* chiedo a gurobi di eseguire l'ottimizzazione del modello che ho definito *mod*.

L' istruzione *if* controlla lo stato dell'ottimizzazione, se lo stato è ottimale viene eseguito quanto richiesto nel codice sottostante:

- Stampa il valore ottimale della funzione obiettivo;
- Scrivi la soluzione ottimale del modello in un file chiamato *Esame.sol;*

Fuori dall'istruzione *if*, con il comando *mod.write('esame.lp')*, si scrive la formulazione per esteso del modello nel file denominato *esame.lp*.

2.6 Stampa la soluzione

```
if mod.status == GRB.OPTIMAL:
    print("L'altezza della libreria è: ", mod.objval, "cm")
    print("Gli scaffali usati e le relative altezze sono:")
    for j in numscaf:
        if y[j].x > 0.5:
            print("Scaffale", j+1, "=", y[j].x, "cm")
    print ()
    print("Afferenze libri:")
    for i in numlib:
        for j in numscaf:
            if x[i,j].x > 0.5 :
                print("Il libro", i+1, "afferisce allo scaffale",j+1)
```

Questo codice, tramite l' istruzione *if* controlla lo stato dell'ottimizzazione, se lo stato è ottimale viene eseguito quanto richiesto nel codice sottostante:

- Stampa il valore ottimo della funzione obiettivo, ovvero l'altezza della libreria;
- Analizzando tramite un ciclo for sull'insieme degli scaffali, se la variabile *y[j]* è diversa da 0, allora stampa a schermo il numero dello scaffale e la relativa altezza (valore di y[j]).
- Analizzando tramite due cicli for, uno sull'insieme dei libri, uno sull'insieme degli scaffali, se la variabile *x[i,j]* è diversa da 0, allora stampa a schermo il numero dell'id del libro (i+1) e il relativo scaffale a cui afferisce (j+1).

3. Risultati con i dati forniti dal testo

```
L'altezza della libreria è:  52.0 cm
Gli scaffali usati e le relative altezze sono:
Scaffale 1 = 25.0 cm
Scaffale 2 = 17.0 cm
Scaffale 3 = 10.0 cm

Afferenze libri:
Il libro 1 afferisce allo scaffale 1
Il libro 2 afferisce allo scaffale 1
Il libro 3 afferisce allo scaffale 2
Il libro 4 afferisce allo scaffale 1
Il libro 5 afferisce allo scaffale 2
Il libro 6 afferisce allo scaffale 2
Il libro 7 afferisce allo scaffale 1
Il libro 8 afferisce allo scaffale 1
Il libro 9 afferisce allo scaffale 2
Il libro 10 afferisce allo scaffale 1
Il libro 11 afferisce allo scaffale 1
Il libro 12 afferisce allo scaffale 3
Il libro 13 afferisce allo scaffale 2
Il libro 14 afferisce allo scaffale 2
Il libro 15 afferisce allo scaffale 2
Il libro 16 afferisce allo scaffale 2
Il libro 17 afferisce allo scaffale 2
Il libro 18 afferisce allo scaffale 1
Il libro 19 afferisce allo scaffale 2
Il libro 20 afferisce allo scaffale 2
```

4. Variazione del modello in base alla seconda richiesta

Nel caso in cui per ogni id di un libro fossero presenti un numero di unità pari proprio all'id del libro, allora potrei prevedere due strade.

La prima, apparentemente più semplice, in cui il modello resta esattamente quello di prima solo che l'insieme dei libri I passa dall'avere 20 elementi (libri) all' averne 210 (1*1 + 2*2 + 3*3 + 4*4 + …+ 20*20), e anche l'insieme degli scaffali J sarà composto da 210 elementi (numero massimo di scaffali). E' un modello sicuramente corretto ma del tutto inefficiente:

- numero di variabili x_{ij} = 210*210 = 44100
- numero di variabili y_j = 210
- numero di vincoli di afferenza = 210
- numero di vincoli sulla larghezza degli scaffali = 210
- numero di vincoli sull'altezza degli scaffali = 44100
- numero di vincoli di binarietà della variabile x_{ij} = 44100
- numero di vincoli di positività della variabile y_j = 210

Per un totale di 44310 variabili e 88830 vincoli.

E' stato fatto un solo tentativo di implementazione del modello su gurobi per verificare l'inefficienza del modello. Di seguito riportati i risultati:

```
5191954 738520     cutoff  126       329.00000  324.00000  1.52%   601 124147s
5195289 738433  324.01468  117  277  329.00000  324.00000  1.52%   601 124162s
5198552 738098     cutoff  148       329.00000  324.00000  1.52%   601 124195s
5205678 737838  325.47197  142  270  329.00000  324.00000  1.52%   601 124218s
5207640 737643  324.00000  135  338  329.00000  324.00000  1.52%   601 124234s
5211223 737370  326.00000  120  240  329.00000  324.00000  1.52%   600 124250s
5214782 737081  324.00000  136  275  329.00000  324.00000  1.52%   600 124267s
5218325 736935  324.00000  124  261  329.00000  324.00000  1.52%   600 124283s
5221890 736843 infeasible  148       329.00000  324.00000  1.52%   600 124299s
5225429 736753  326.00000  130  261  329.00000  324.00000  1.52%   600 124315s
5228840 736513  325.00000  144  178  329.00000  324.00000  1.52%   600 124330s
5231098 736420  324.00000  135  346  329.00000  324.00000  1.52%   599 124347s
5233342 736326     cutoff  127       329.00000  324.00000  1.52%   599 124364s
5235709 736106  325.00000  113  262  329.00000  324.00000  1.52%   599 124383s
5238007 735893  324.00000  129  368  329.00000  324.00000  1.52%   599 124404s
5240669 735611 infeasible  124       329.00000  324.00000  1.52%   599 124425s
5243508 735280  325.00000  129  267  329.00000  324.00000  1.52%   599 124447s
5246338 735072  324.00000  119  354  329.00000  324.00000  1.52%   599 124470s
5249229 734834 infeasible  106       329.00000  324.00000  1.52%   599 124494s
5252004 734538 infeasible  128       329.00000  324.00000  1.52%   599 124518s
```

Dopo un tempo di 124518 secondi (35 ore) l'algoritmo ha trovato un incumbent di 329, ossia la migliore soluzione ammissibile trovata fino a quel momento e un best bound di 324 ossia il miglior lower bound calcolato fino a quel momento con un gap del 1,52%.

La seconda strada prevede di applicare una modifica al modello precedente che mi permette di diminuire il numero di variabili e di vincoli e quindi permette a gurobi di risolvere il problema in un tempo sensibilmente ridotto.

Innanzitutto, oltre all'insieme I dei libri che rimane di n = 20 elementi e all'insieme J degli scaffali che in questo caso sarà composto da m = 210 elementi (numero massimo di scaffali ottenibile posizionando un libro per ogni scaffale) sarà necessario prevedere un vettore b delle disponibilità di ogni id libro:

$$I = \{1....n\} \qquad \textit{insieme degli id dei libri } (20)$$
$$J = \{1....m\} \qquad \textit{insieme massimo degli scaffali } (210) \ (\textit{disponibilità del libro } i = id(i))$$
$$b = \{b_1, b_2, \dots, b_n\} \qquad (b_1 = 1, b_2 = 2, \dots, b_n = n)$$

Inoltre, in questo caso non siamo più interessati a sapere solo se Il libro i è presente sullo scaffale j, ma abbiamo bisogno di un'informazione aggiuntiva, ossia quante unità di libro i sono presenti sullo scaffale j.

Per avere questa informazione aggiuntiva è necessario cambiare la tipologia di variabile x_{ij} da binaria a intera. Così facendo, se x_{ij} è maggiore di 0, allora saprò per certo che il libro i è sullo scaffale j e saprò anche quante unità ne sono presenti.

Le modifiche che comporta sui vincoli questo cambiamento di variabile sono fondamentalmente due:

- I vincoli di afferenza saranno del tipo:

$$\sum_{j \in J} x_{ij} = b_i \qquad \forall\, i \in I$$

  Dove b_i è la disponibilità del libro i. Cioè fissato il libro di tipologia i, che avrà una certa disponibilità che indico con b_i, allora la somma delle quantità del libro *i* presenti sui diversi scaffali della libreria deve essere pari proprio alla disponibilità di *i*.

- I vincoli sull'altezza di ogni scaffale scritti nel primo modello non valgono più, perché prima avevo una variabile x_{ij} binaria, ora la variabile è intera e quindi è come se andassi a sommare l'altezza del libro di tipologia *i* tante volte quanti sono i libri di tipologia i sullo *j*-esimo scaffale, il che non ha affatto senso.

Si è pensato di risolvere questo problema introducendo un nuovo insieme di variabili binarie u_{ij} che valgono 1 se la variabile x_{ij} è maggiore di 0, valgono invece 0 se x_{ij} è pari anch'essa a 0.

Questo comporta l'introduzione anche di una serie di vincoli che legano le due variabili:

$$x_{ij}u_{ij} \geq x_{ij} \qquad \forall\, j \in J,\ i \in I$$

Questi sono **m*n** vincoli che impongono che se x_{ij} è maggiore di 0, allora u_{ij} è obbligata a valere 1. Ma se x_{ij} vale 0, solo con questo vincolo non posso dire niente su u_{ij}, quindi necessito di una ulteriore famiglia di vincoli, che è la seguente.

$$u_{ij} \leq x_{ij} \qquad \forall\, j \in J,\ i \in I$$

Anche questi sono **m*n** vincoli che, insieme alla famiglia di vincoli di cui sopra, impongono che se x_{ij} vale 0 allora anche u_{ij} è vincolata a valere 0 (dato che è una variabile binaria).

A questo punto siamo liberi di riscrivere il vincolo sulle altezze degli scaffali allo stesso modo di come è stato scritto nel primo modello, con l'unica differenza che cambia il nome della variabile:

$$y_j \geq a_i u_{ij} \qquad \forall\, j \in J,\ i \in I$$

Gli ultimi vincoli che rimangono da aggiungere sono i vincoli di interezza sulle variabili x_{ij} e i vincoli di binarietà sulle variabili u_{ij}.

$$u_{ij} \in \{0,1\} \qquad \forall\, j \in J,\ i \in I$$
$$x_{ij} \geq 0 \qquad \forall\, j \in J,\ i \in I$$

Le variabili e i vincoli in gioco in questo caso saranno in numero pari a:

- 20*210 (4200) variabili x_{ij} ;
- 210 variabili y_j ;
- 4200 variabili $u_{ij;}$
- 20 vincoli di afferenza;
- 210 vincoli sulla larghezza degli scaffali;
- 4200 vincoli sull'altezza degli scaffali;
- 4200 vincoli di binarietà della variabile u_{ij} ;
- 4200 vincoli di interezza della variabile x_{ij};
- 8400 vincoli di variable upper bound tra la variabile x_{ij} e la variabile u_{ij};
- 210 vincoli di positività della variabile y_j ;

Per un totale di 8610 variabili e 21440 vincoli in gioco. Tramite questo modello abbiamo diminuito dell'80% il numero di variabili e il numero di vincoli in gioco. Infatti implementando questo modello su gurobi, ci è stata fornita la soluzione ottima in 5 minuti.

Formulazione completa del modello:

$$\min \sum_{j \in J} y_j$$

$$s.t.$$

$$\sum_{j \in J} x_{ij} = b_i \qquad \forall\, i \in I$$

$$\sum_{i \in I} l_i x_{ij} \leq 100 \qquad \forall\, j \in J$$

$$x_{ij} u_{ij} \leq x_{ij} \qquad \forall\, j \in J,\ i \in I$$

$$u_{ij} \leq x_{ij} \qquad \forall\, j \in J,\ i \in I$$

$$y_j \geq a_i u_{ij} \qquad \forall\, j \in J,\ i \in I$$

$$y_{j+1} \leq y_j \qquad \forall\, j \in J$$

$$y_j \geq 0 \qquad \forall\, j \in J$$

$$u_{ij} \in \{0,1\} \qquad \forall\, j \in J,\ i \in I$$

$$x_{ij} \geq 0 \qquad \forall\, j \in J,\ i \in I$$

4.1 Codice python e risoluzione del problema su gurobi

4.1.1 Lettura dei dati da file

```
file = open('esame.rtf','r')
filelines = file.readlines()

num_line = 1
line = filelines[num_line].split()
numlib = range(int(line[0]))
numscaf = range(210)

num_line = num_line + 2
SIZE = []
DISP = []
for i in numlib:
    SIZE_line = []
    line = filelines[num_line].split()
    DISP.append(int(line[0]))
    for j in range(1,3):
        SIZE_line.append(int(line[j]))
    SIZE.append(SIZE_line)
    num_line = num_line +1
```

Il codice di lettura è rimasto praticamente invariato se non per il terzo blocco, in cui ho aggiornato l'indice di riga a 3 e mi sono posizionato sulla quarta riga del file di testo corrispondente alla prima riga della matrice delle dimensioni dei libri.

A questo punto, siccome la matrice è una lista di liste, ho preliminarmente inizializzato una lista vuota S*IZE* in cui alla fine dovrò aggiungere tante liste (ognuna delle quali

contiene le dimensioni dell'i-esimo libro) quanti sono i libri, inoltre ho creato una lista vuota *DISP* in cui alla fine andrò a mettere le disponibilità dei libri. Successivamente, tramite due cicli for uno dentro l'altro, ho:

- creato una lista vuota *sizeline* per l'i-esima riga del file, aggiunto alla lista line la lista di stringhe corrispondente all'i-esima riga e aggiunto alla lista *DISP* il valore intero corrispondente alla stringa in posizione 0 della lista line;
- Con il secondo ciclo for ho aggiunto alla lista *sizeline* gli interi corrispondenti alle stringhe contenute nella lista line;
- Uscito dal secondo ciclo for torno nel primo ciclo in cui aggiungo alla lista *SIZE* la lista sizeline, incremento l'indice di 1 e torno al passo iniziale;

Alla fine mi troverò una lista *DISP* corrispondente al vettore delle disponibilità che nel modello ho chiamato b_i e una lista di liste *SIZE* che è la matrice delle dimensioni già vista nel primo modello.

4.1.2 Inizializzazione modello e dichiarazione delle variabili

```
import gurobipy as gp
from gurobipy import GRB

# Inizializza il modelo
mod = gp.Model('Richiesta2')
```

```
x = mod.addVars(numlib,numscaf, vtype = GRB.INTEGER, name = 'x')
y = mod.addVars(numscaf, vtype = GRB.CONTINUOUS, name = 'y')
u = mod.addVars(numlib, numscaf, vtype = GRB.BINARY, name = 'u')
```

4.1.3 Funzione obiettivo e vincoli

Funzione obiettivo:

```
obj = mod.setObjective(gp.quicksum(y[j] for j in numscaf), GRB.MINIMIZE)
```

Vincoli sulla disponibilità dei libri:

```
mod.addConstrs(gp.quicksum(x[i,j] for j in numscaf) == DISP[i] for i in numlib)
```

Vincoli sulla larghezza degli scaffali:

```
mod.addConstrs(gp.quicksum(SIZE[i][0]*x[i,j] for i in numlib) <= 100 for j in numscaf)
```

Vincoli sull'altezza degli scaffali:

```
mod.addConstrs(y[j] >= SIZE[i][1]*u[i,j] for i in numlib for j in numscaf)
```

Vincoli che legano la variabile x_{ij} alla variabile u_{ij}:

```
mod.addConstrs(x[i,j]*u[i,j] >= x[i,j] for i in numlib for j in numscaf)
```

```
mod.addConstrs(u[i,j]<= x[i,j] for i in numlib for j in numscaf)
```

Vincoli aggiuntivi sull'ordine degli scaffali:

```
mod.addConstrs((y[j+1])<= y[j] for j in range(209))
```

4.1.4 Risoluzione del problema libreria gurobi

```
mod.optimize()
if mod.status == GRB.OPTIMAL:
    print (mod.objval)
    mod.write('richiesta2.sol')
mod.write('richiesta2.lp')
```

4.1.5 Stampa la soluzione

```
if mod.status == GRB.OPTIMAL:
    print("L'altezza della libreria è: ", mod.objval, "cm")
    print("Gli scaffali usati e le relative altezze sono:")
    for j in numscaf:
        if y[j].x > 0.5:
            print("Scaffale", j+1, "=", y[j].x, "cm")
    print ()
    print("Appartenenza libri-scaffali:")
    for i in numlib:
        for j in numscaf:
            if x[i,j].x > 0.5 :
                print("Ci sono",x[i,j].x,"libri di ID:",i+1,"sullo scaffale",j+1)
```

Questo codice, tramite l' istruzione *if* controlla lo stato dell'ottimizzazione, se lo stato è ottimale viene eseguito quanto richiesto nel codice sottostante:

- Stampa il valore ottimo della funzione obiettivo, nonché altezza della libreria
- Analizzando tramite un ciclo for sull'insieme degli scaffali, se la variabile y[j] è diversa da 0, allora stampa a schermo il numero dello scaffale e la relativa altezza (valore di y[j]).
- Analizzando tramite due cicli for, uno sull'insieme dei libri, uno sull'insieme degli scaffali, se la variabile x[i,j] è diversa da 0, allora stampa a schermo il numero di libri di libri (x[i,j]) di tipologia i+1 e il relativo scaffale a cui afferisce (j+1).

4.1.6 Risultati con i dati forniti dal testo

```
L'altezza della libreria è:  329.0 cm
Gli scaffali usati e le relative altezze sono:
Scaffale 1 = 25.0 cm
Scaffale 2 = 20.0 cm
Scaffale 3 = 19.0 cm
Scaffale 4 = 18.0 cm
Scaffale 5 = 18.0 cm
Scaffale 6 = 17.0 cm
Scaffale 7 = 17.0 cm
Scaffale 8 = 17.0 cm
Scaffale 9 = 16.0 cm
Scaffale 10 = 16.0 cm
Scaffale 11 = 16.0 cm
Scaffale 12 = 15.0 cm
Scaffale 13 = 15.0 cm
Scaffale 14 = 15.0 cm
Scaffale 15 = 15.0 cm
Scaffale 16 = 14.0 cm
Scaffale 17 = 14.0 cm
Scaffale 18 = 12.0 cm
Scaffale 19 = 10.0 cm
Scaffale 20 = 10.0 cm
Scaffale 21 = 10.0 cm
```

```
Appartenenza libri-scaffali:
Ci sono 1.0 libri di ID: 1 sullo scaffale 1
Ci sono 2.0 libri di ID: 2 sullo scaffale 1
Ci sono 3.0 libri di ID: 3 sullo scaffale 7
Ci sono 4.0 libri di ID: 4 sullo scaffale 4
Ci sono 1.0 libri di ID: 5 sullo scaffale 12
Ci sono 4.0 libri di ID: 5 sullo scaffale 13
Ci sono 2.0 libri di ID: 6 sullo scaffale 9
Ci sono 3.0 libri di ID: 6 sullo scaffale 10
Ci sono 1.0 libri di ID: 6 sullo scaffale 12
Ci sono 7.0 libri di ID: 7 sullo scaffale 3
Ci sono 2.0 libri di ID: 8 sullo scaffale 6
Ci sono 6.0 libri di ID: 8 sullo scaffale 8
Ci sono 4.0 libri di ID: 9 sullo scaffale 6
Ci sono 4.0 libri di ID: 9 sullo scaffale 7
Ci sono 1.0 libri di ID: 9 sullo scaffale 8
Ci sono 10.0 libri di ID: 10 sullo scaffale 2
Ci sono 6.0 libri di ID: 11 sullo scaffale 1
Ci sono 5.0 libri di ID: 11 sullo scaffale 2
Ci sono 2.0 libri di ID: 12 sullo scaffale 19
Ci sono 5.0 libri di ID: 12 sullo scaffale 20
Ci sono 5.0 libri di ID: 12 sullo scaffale 21
Ci sono 13.0 libri di ID: 13 sullo scaffale 18
Ci sono 1.0 libri di ID: 14 sullo scaffale 2
Ci sono 2.0 libri di ID: 14 sullo scaffale 12
Ci sono 11.0 libri di ID: 14 sullo scaffale 17
Ci sono 1.0 libri di ID: 15 sullo scaffale 13
Ci sono 10.0 libri di ID: 15 sullo scaffale 16
Ci sono 4.0 libri di ID: 15 sullo scaffale 17
Ci sono 3.0 libri di ID: 16 sullo scaffale 9
Ci sono 8.0 libri di ID: 16 sullo scaffale 10
```

```
Ci sono 5.0 libri di ID: 17 sullo scaffale 11
Ci sono 2.0 libri di ID: 18 sullo scaffale 1
Ci sono 6.0 libri di ID: 18 sullo scaffale 4
Ci sono 10.0 libri di ID: 18 sullo scaffale 5
Ci sono 3.0 libri di ID: 19 sullo scaffale 12
Ci sono 7.0 libri di ID: 19 sullo scaffale 13
Ci sono 8.0 libri di ID: 19 sullo scaffale 14
Ci sono 1.0 libri di ID: 19 sullo scaffale 15
Ci sono 6.0 libri di ID: 20 sullo scaffale 12
Ci sono 4.0 libri di ID: 20 sullo scaffale 14
Ci sono 10.0 libri di ID: 20 sullo scaffale 15
```

5. Procedure per ottenere Upper e Lower Bound

Si desidera trovare due procedure per ottenere un Upper Bound e un Lower Bound per il problema di riferimento.

5.1 Lower Bound

In un problema a minimizzare una delle tecniche che si possono utilizzare per ottenere un lower bound, ossia una stima inferiore del numero minimo di risorse necessarie per risolvere un problema, è il cosiddetto rilassamento dei vincoli.

Nel caso in esame si potrebbe pensare di rilassare i vincoli di binarietà posti sulle variabili x_{ij} consentendo loro di assumere valori frazionari nel range [0,1]. Nella pratica questo significa che sarà possibile allocare frazioni di libri all'interno degli scaffali (ex: se $x_{ij} = 0,5$ allora metà libro del libro i andrà posizionato nello scaffale j).

Questo metodo sicuramente consentirà di ottenere una stima inferiore dell'altezza minima della libreria per poter ospitare tutti i libri, in quanto una soluzione calcolata con variabili intere binarie non varrà mai meno di questa.

Queste valutazioni sono state confermate e avvalorate dall'implementazione del modello "rilassato" su gurobi:

Dichiarazione variabili con riassamento:

```
x = mod.addVars(numlib,numscaf, vtype = GRB.CONTINUOUS, name = 'x')
y = mod.addVars(numscaf, vtype = GRB.CONTINUOUS, name = 'y')
```

Aggiunta del vincolo di continuità nell'intervallo [0,1]:

```
for i in numlib:
    for j in numscaf:
        mod.addConstr(x[i,j] <=1)
        mod.addConstr(x[i,j] >=0)
```

Risultato:

```
L'altezza della libreria è  25.0 cm
```

5.2 Upper Bound

In un problema a minimizzare un upper bound è un valore di funzione obiettivo maggiore o uguale del valore ottimo che si ottiene risolvendo il problema di ottimizzazione.

Il valore che fornisce l'upper bound è una soluzione ammissibile del problema ma di qualità inferiore alla soluzione ottima. Tuttavia, talvolta, per mancanza di tempo e per istanze di dimensioni elevate, si possono usare tecniche meno dispendiose che ci danno una soluzione che è sicuramente ammissibile ma che non è detto che sia la soluzione ottima. Queste tecniche sono chiamate "euristiche".

Tra le varie euristiche per il calcolo dell'upper bound, nel problema in questione, sarebbe particolarmente utile utilizzare l'euristica greedy.

Preliminarmente ci conviene dimostrare, anche se è banale, che questo problema è un problema di ottimizzazione combinatoria, quindi sarà possibile esprimerlo in termini di ground set system, subset system, funzione obiettivo w(S):

- Come ground-set system prendiamo l'insieme dei libri B = {1,2,3,..,20};

- Come sub-set system prendiamo tutti sottoinsiemi del ground-set system che siano soluzioni ammissibili, ossia tutte le combinazioni di come sistemare 20 libri in una libreria di larghezza 100cm, tenendo conto delle dimensioni dei libri. Quindi scriveremo $\Sigma = \{S_1,\ldots,S_m\}$.

- La funzione obiettivo a minimizzare l'abbiamo già definita come sommatoria su J di y_j , dove quest'ultima è l'altezza dello scaffale j, funzione della disposizione dei libri da me scelta.

L'algoritmo greedy per il problema di riferimento è:

Step 1
Poni $T_0 = \emptyset$ e $i = 0$.

Step 2
Scegli un elemento e dal ground-set tale che la soluzione parziale rimanga tale e il costo della soluzione parziale $T_{i-1} \cup \{e\}$ sia minimo.

Step 3
Poni $_i = T_{i-1} \cup \{e\}$.

Step 4
Se T_i è una soluzione ammissibile, allora STOP.

Step 5
Se T_i non è ammissibile. Poni $i = i+1$e torna allo Step 2.

Lo step 4, ovvero la valutazione dell'ammissibilità della soluzione parziale T_i, si può effettuare valutando se tutti i libri sono stati allocati in uno scaffale, tenendo conto del vincolo sulle larghezze degli scaffali. L'algoritmo si fermerà dunque quando non saranno più rimasti libri da allocare.

Questo algoritmo svolge 21 iterazioni per trovare un upper bound per il problema di riferimento:

Iterazione 1:

T_0 = {Ø}

Libri da allocare= {1,2,3,…,20}

Costo di T_0 = 0

Spazio occupato nel primo scaffale = 0

Iterazione 2 :

T_1 ={12}

Libri da allocare= {1,2,3,4,5,6,7,8,9,10,11,13,14,15,16,17,18,19,20}

Costo di T_1 = 10

Spazio occupato nel primo scaffale = 20cm

Iterazione 3:

T_2 ={12,13}

Libri da allocare= {1,2,3,4,5,6,7,8,9,10,11,14,15,16,17,18,19,20}

Costo di T_2 = 12

Spazio occupato nel primo scaffale = 27 cm

Iterazione 4:

T_3 ={12,13,14}

Libri da allocare= {1,2,3,4,5,6,7,8,9,10,11,15,16,17,18,19,20}

Costo di T_3 = 14

Spazio occupato nel primo scaffale = 32 cm

Iterazione 5:

T_4 ={12,13,14,15}

Libri da allocare= {1,2,3,4,5,6,7,8,9,10,11,16,17,18,19,20}

Costo di T_4 = 14

Spazio occupato nel primo scaffale = 42 cm

 Iterazione 6:

T_5 ={12,13,14,15,6}

Libri da allocare= {1,2,3,4,5,7,8,9,10,11,16,17,18,19,20}

Costo di T_5 = 15

Spazio occupato nel primo scaffale = 47 cm

 Iterazione 7:

T_6 ={12,13,14,15,6,5}

Libri da allocare= {1,2,3,4,7,8,9,10,11,16,17,18,19,20}

Costo di T_6 = 15

Spazio occupato nel primo scaffale = 54 cm

 Iterazione 8:

T_7 ={12,13,14,15,5,6,19}

Libri da allocare= {1,2,3,4,7,8,9,10,11,16,17,18,20}

Costo di T_7 = 15

Spazio occupato nel primo scaffale = 62 cm

 Iterazione 9:

T_8 ={12,13,14,15,6,5,19,20}

Libri da allocare= {1,2,3,4,7,8,9,10,11,16,17,18}

Costo di T_8 = 15

Spazio occupato nel primo scaffale = 71 cm

 Iterazione 10:

T_9 ={12,13,14,15,6,5,19,20,16}

Libri da allocare= {1,2,3,4,7,8,9,10,11,17,18}

Costo di T_9 = 16

Spazio occupato nel primo scaffale = 76 cm

 Iterazione 11:

T_{10} ={12,13,14,15,6,5,19,20,16,17}

Libri da allocare = {1,2,3,4,7,8,9,10,11,18}

Costo di T_{10} = 16

Spazio occupato nel primo scaffale = 91cm (il prossimo libro occupa 10cm quindi cambio scaffale)

 Iterazione 12:

T_{11} ={12,13,14,15,6,5,19,20,16,17;9}

Libri da allocare= {1,2,3,4,7,8,10,11,18}

Costo di T_{11} = 33

Spazio occupato nel secondo scaffale = 10 cm

 Iterazione 13:

T_{12} ={12,13,14,15,6,5,19,20,16,17,9,3}

Libri da allocare= {1,2,4,7,8,10,11,18}

Costo di T_{12} = 33

Spazio occupato nel secondo scaffale = 20 cm

 Iterazione 14:

T_{13} ={12,13,14,15,6,5,19,20,16,17,9,3,8}

Libri da allocare= {1,2,4,7,10,11,18}

Costo di T_{13} = 33

Spazio occupato nel secondo scaffale = 35 cm

 Iterazione 15:

T_{14} ={12,13,14,15,6,5,19,20,16,17,9,3,8,11}

Libri da allocare= {1,2,4,7,10,18}

Costo di T_{14} = 34

Spazio occupato nel secondo scaffale = 40 cm

 Iterazione 16:

T_{15} ={12,13,14,15,6,5,19,20,16,17,9,3,8,11,18}

Libri da allocare= {1,2,4,7,10}

Costo di T_{15} = 34

Spazio occupato nel secondo scaffale = 50 cm

 Iterazione 17:

T_{16} ={12,13,14,15,6,5,19,20,16,17,9,3,8,11,18,4}

Libri da allocare= {1,2,7,10}

Costo di T_{16} = 34

Spazio occupato nel secondo scaffale = 60 cm

 Iterazione 18:

T_{17} ={12,13,14,15,6,5,19,20,16,17,9,3,8,11,18,4,7}

Libri da allocare= {1,2,10}

Costo di T_{17} = 35

Spazio occupato nel secondo scaffale = 74 cm

 Iterazione 19:

T_{18} ={12,13,14,15,6,5,19,20,16,17,9,3,8,11,18,4,7,10}

Libri da allocare= {1,2}

Costo di T_{18} = 36

Spazio occupato nel secondo scaffale = 81 cm

 Iterazione 20:

T_{19} ={12,13,14,15,6,5,19,20,16,17,9,3,8,11,18,4,7,10,2}

Libri da allocare= {1}

Costo di T_{19} = 36

Spazio occupato nel secondo scaffale = 96 cm (il prossimo libro occupa 20 cm quindi cambio scaffale)

 Iterazione 21:

T_{20} ={12,13,14,15,6,5,19,20,16,17;9,3,8,11,18,4,7,10,2;1}

Libri da allocare= { Ø }

Costo di T_{20} = 61

Spazio occupato nel terzo scaffale = 20 cm

La soluzione trovata dall'algoritmo greedy è 61cm e tre scaffali, mentre quella trovata dal solutore è 52 cm e tre scaffali.

SOLUZIONE EURISTICA		
SCAFFALI	3	1
	2	2,10,7,4,18,11,8,3,9
	1	17,16,20,19,5,6,15,14,13,12

SOLUZIONE OTTIMA		
SCAFFALI	3	12
	2	20,19,17,16,15,14,13,9,6,5,3
	1	18,11,10,8,7,4,2,1

Considerando queste due soluzioni possiamo calcolare il gap:

$$gap = \frac{|\mathrm{OPT(I)} - \mathrm{EUR(I)}|}{|\mathrm{OPT(I)}|}$$

Nell' istanza in esame abbiamo un $gap = \frac{|52-61|}{|52|}$ = 17,3%

Quindi l'euristica greedy produce un errore del 17,3%, ossia fornisce una soluzione che si discosta del 17,3% dalla soluzione ottima.

Se abbiamo tempo potremmo provare a migliorare il valore della soluzione euristica utilizzando un apposito algoritmo di ricerca locale, ossia data la soluzione S_0 fornita dall'algoritmo greedy, dobbiamo definire una mossa, ovvero un'operazione che a partire di S_0 mi genere altre soluzioni ammissibili in un intorno di S_0.

Questa mossa potrebbe essere un'operazione di 2-scambio, cioè prendiamo due libri appartenenti a scaffali diversi e li scambiamo tra di loro, ove possibile.

Oppure la mossa potrebbe essere quella di rimuovere un libro da uno scaffale e tentare, ove possibile, di aggiungerlo in un altro.

Esempio:

- Parto dalla soluzione ammissibile S_0 fornita dall'euristica greedy e pongo i = 1
- Sia S_i la soluzione più conveniente ottenibile da S_{i-1} per 2-scambio
- Se $w(S_i) >= w(S_{i-1})$ allora STOP. S_{i-1} è la miglior soluzione trovata fino a questo punto
- Altrimenti poni i = i+1 e vai al secondo punto

Per migliorare la qualità delle soluzioni di questa euristica di ricerca locale è possibile procedere per due strade:

- La strada della diversificazione, seguendo un approccio multi-start che consiste nell' applicare più volte l'algoritmo a partire da diverse soluzioni iniziali sparse su tutto lo spazio delle soluzioni
- La strada dell'intensificazione, ovvero sono arrivato a un valore S_m tramite l'euristica di ricerca locale e da lì non mi muovo più perché quello è il minimo locale nell'intorno che sto visualizzando. Allora posso pensare di aumentare le dimensioni dell'intorno passando da una ricerca locale 2-opt a una 3-opt, ossia scambiando 3 libri appartenenti a 3 diversi scaffali tra di loro. Questo significa che in ogni intorno dovrò calcolare e valutare in funzione obiettivo un numero di soluzioni molto più grande, perdendo più tempo, quindi nel trade-off tra accuratezza e tempi di calcolo risulta importante scegliere correttamente la dimensione dell'intorno.

Ecco un esempio di iterazione di ricerca locale 2-opt:

- Soluzione S_0 fornita dall'algoritmo greedy:

  scaffale 1 = 17,16,20,19,5,6,15,14,13,12

scaffale 2 = 2,10,7,4,18,11,8,3,9

scaffale 3 = 1

- Prendo una soluzione S_1 ottenuta per 2-scambio, scambiando il libro 1 sullo scaffale 3 con il libro 12 sullo scaffale 1. La soluzione S_1 è una soluzione ammissibile perché i due libri hanno la stessa larghezza, quindi è rispettato il vincolo sulla larghezza degli scaffali.
- Calcolo il valore della funzione obiettivo in S_1: $\sum_J y(j) = y_1+y_2+y_3 = 25 + 20+ 10 =$ 55 cm (ora sullo scaffale 1 il libro più alto è proprio il libro 1 di altezza 25cm, quindi lo scaffale 1 dovrà essere alto 25 cm, mentre lo scaffale 3 ora contiene solo il libro 12 di altezza 10cm quindi lo scaffale 3 sarà alto 10 cm).
- $z(S_1) < z(S_0)$ (55<61) quindi S_1 è la migliore soluzione al momento, pongo i = i+1 e torno al passo 2.