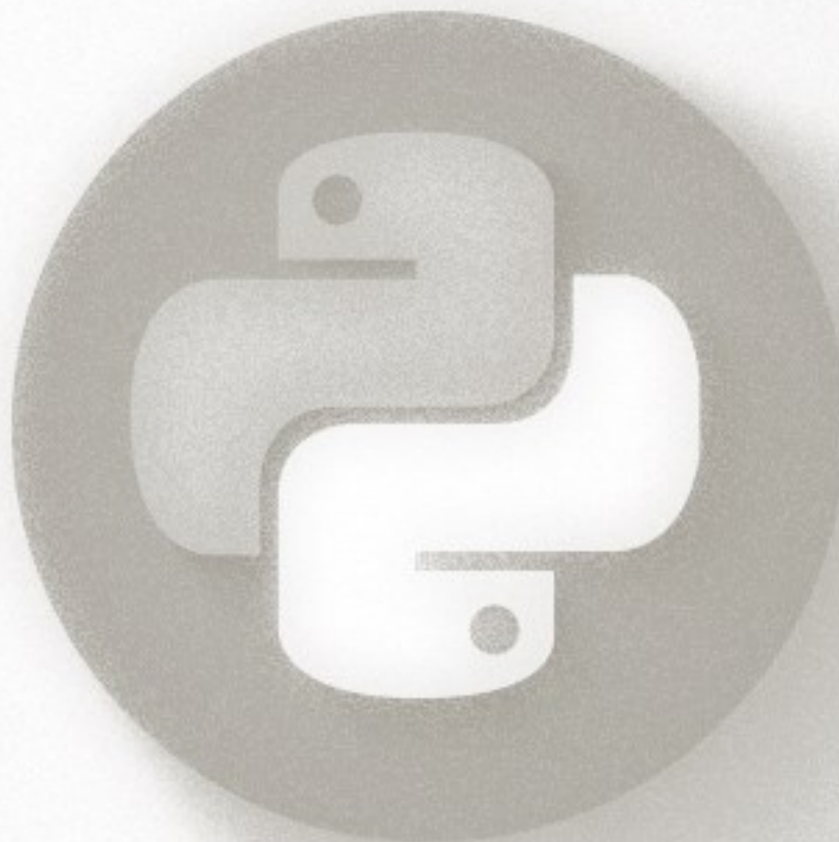


Programmazione lineare in Python

Dipartimento Economia, Statistica
e Finanza
«Giovanni Anania»
Corso di Laurea in Statistica per
l'Azienda



Relatore:
Prof. Alfredo Garro

Anno accademico 2021/2022

Candidato:
Nicola Lavecchia
Matricola: 212169

LA PROGRAMMAZIONE LINEARE

- È una tecnica di risoluzione di problemi reali che hanno una struttura lineare
- Un problema di PL ha come scopo l'allocazione di risorse soggette a vincoli, al fine di soddisfare una funzione obiettivo seguendo determinati step:



IL METODO DEL SIMPLESSO

- È stato il primo algoritmo pratico per la risoluzione di problemi di PL ed è tuttora il più usato e uno dei più efficienti, l'algoritmo segue alcune fasi:

1. Trasformazione del problema dalla forma standard alla forma canonica

2. Identificazione della soluzione di base accettabile

3. Applicazione della prova di ottimalità

4. In caso di ammissibilità ma non di ottimalità, si ricerca una base che garantisce una soluzione ottima



Il risolutore Excel

B	C	D	E	F	G	H	
		Drone 1	Drone 2	Drone 3			
	Profitto	€ 100	€ 400	€ 50	Usati	Disponibili	
	Costo	€ 320	€ 1.300	€ 120	0	€ 100.000	
	Spazio	0,5	1	0,4	0	256	
						Profitto	
	Unità	0	0	0		0	

**Trascrizione dei
dati su foglio di
calcolo e
immissione
formula nelle celle
dedicate**

Dati Revisione Parametri Risolutore

Imposta obiettivo:

A: ☒ Max ☐ Min ☐ Valore di:

Modificando le celle variabili:

Soggette ai vincoli:

Aggiungi
Cambia
Elimina
Reimposta tutto
Carica/Salva

☒ Rendi non negative le variabili senza vincoli

Selezionare un metodo di risoluzione: Opzioni

Metodo di risoluzione
Selezionare il motore GRG non lineare per i problemi lisci non lineari del Risolutore. Selezionare il motore Simplex LP per i problemi lineari e il motore evolutivo per i problemi non lisci.

Guida Risolvi Chiudi

Disponibili	
3000	€ 100.000
256	256
Profitto	
38380	

**Invocazione del
risolutore,
aggiunta dei
vincoli e scelta di
un metodo di
risoluzione**

	A	B	C	D	E	F	G	H	I	J
1	Microsoft Excel 16.0 Rapporto valori									
2	Foglio di lavoro: [Cartel1]Foglio1									
3	Data creazione rapporto: 01/12/2022 00:59:17									
4	Risultato: È stata trovata una soluzione. Tutti i vincoli e le condizioni di ottimalizzazione sono stati soddisfatti.									
5	Motore Risolutore									
9	Opzioni Risolutore									
12										
13										
14	Cella obiettivo (Max)									
15	Cella	Nome	Valore originale	Valore finale						
16	\$H\$7	Unità Profitto	0	38380						
17										
18										
19	Celle variabili									
20	Cella	Nome	Valore originale	Valore finale	Intere					
21	\$D\$7:\$F\$7									
25										
26										
27										
28	Vincoli									
29	Cella	Nome	Valore della cella	Formula	Stato	Tolleranza				
30	\$G\$4	Costo Usati	100000	\$G\$4<=\$H\$4	Vincolante	0				
31	\$G\$5	Spazio Usati	256	\$G\$5<=\$H\$5	Vincolante	0				
32										
33										
34										
35										
36										
37										
38										

Rapporto valori 1

Foglio1

+

Visualizzazione dei risultati

IL MODELLATORE PuLP

- La libreria PuLP è un modellatore PL scritto in Python, per creare e risolvere problemi di PL tramite questo modellatore si usano diverse funzioni date dalla classe già fornite:
- **LpProblem** : classe contenitore per un problema di PL
- **LpVariable**: variabili che vengono aggiunte ai vincoli nella PL
- **LpConstraint**: un vincolo dalla forma generale: $a_1x_1 + a_2x_2 \dots a_nx_n (<=, =, >=) b$
- **problem.solve()**: per risolvere il problema definito, è possibile utilizzare diversi risolutori da specificare tra parentesi.

PROBLEMA DI PRODUZIONE TRAMITE PuLP

Ingredienti	Proteina	Grasso	Fibra	Sale
Pollo	0,100	0,080	0,001	0,002
Manzo	0,200	0,100	0,005	0,005
Montone	0,150	0,110	0,003	0,007
Riso	0,000	0,010	0,100	0,002
Grano	0,040	0,010	0,150	0,008
Gel	0,000	0,000	0,000	0,000

**contributo al peso totale di
proteine, grassi, fibre e sale nel
prodotto finale**

```
from pulp import *
```

```
problema = LpProblem("Problema", LpMinimize)
```

```
problema += 0.013 * x1 + 0.008 * x2 + 0.010 * x3 + 0.002 * x4 + 0.005 * x5 + 0.001 * x6, "Costo totale ingredienti per porzione"
```

```
problema += x1 + x2 + x3 + x4 + x5 + x6 == 100, "Percentuale somma a 100"
```

```
problema += 0.100 * x1 + 0.200 * x2 + 0.150 * x3 + 0.000 * x4 + 0.040 * x5 + 0.000 * x6 >= 8.0, "Richiesta proteine"
```

```
problema += 0.080 * x1 + 0.100 * x2 + 0.110 * x3 + 0.010 * x4 + 0.010 * x5 + 0.000 * x6 >= 6.0, "Richiesta Grasso"
```

```
problema += 0.001 * x1 + 0.005 * x2 + 0.003 * x3 + 0.100 * x4 + 0.150 * x5 + 0.000 * x6 <= 2.0, "Richiesta fibre"
```

```
problema += 0.002 * x1 + 0.005 * x2 + 0.007 * x3 + 0.002 * x4 + 0.008 * x5 + 0.000 * x6 <= 0.4, "Richiesta sale"
```

**Definizione del problema e aggiunta di funzione
obiettivo e vincoli**


```
problema.solve()

print("Stato del problema: ", LpStatus[problema.status])
print()

for variabile in problema.variables():
    print(variabile.name, " = ", variabile.varValue)

print()

print("Costo totale degli ingredienti per porzione: ", value(problema.objective))
```

Result - Optimal solution found


Objective value: 0.52000000
Enumerated nodes: 0
Total iterations: 0
Time (CPU seconds): 0.02
Time (Wallclock seconds): 0.02

Option for printingOptions changed from normal to all
Total time (CPU seconds): 0.02 (Wallclock seconds): 0.02

Stato del problema: Optimal

Percentuale_Gel = 40.0
Percentuale_Grano = 0.0
Percentuale_Manzo = 60.0
Percentuale_Montone = 0.0
Percentuale_Pollo = 0.0
Percentuale_Riso = 0.0

Costo totale degli ingredienti per porzione: 0.52



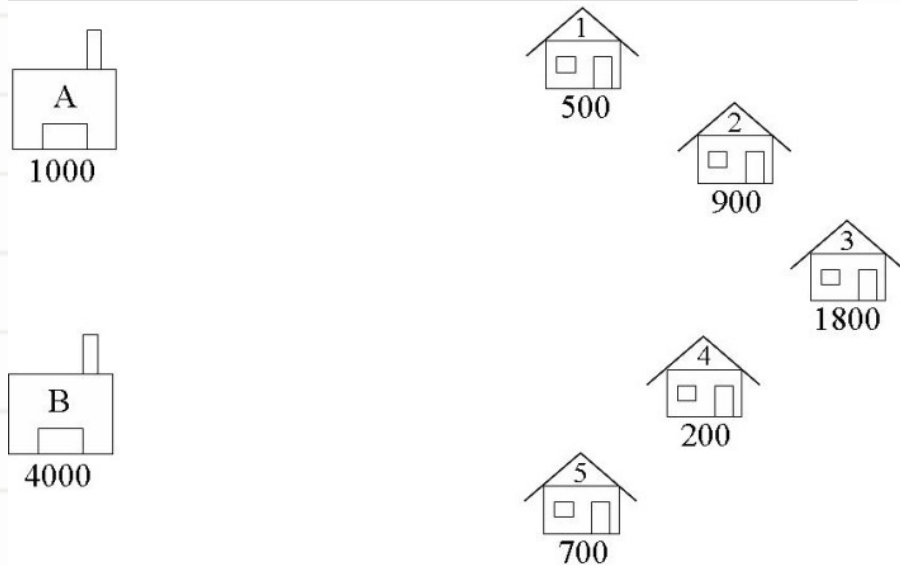
**Si lancia il comando di
risoluzione e si
stampano i risultati in
modo chiaro**



**Visualizzazione dei
risultati**

PROBLEMA DI TRASPORTO TRAMITE PuLP

Disponibilità dei magazzini e richiesta dei bar



$$\min \sum_{w \in W, b \in B} c_{(w,b)} x_{(w,b)}$$

**Funzione
obiettivo**

Variabili decisionali

A1 = numero di casse di birra da spedire dal Magazzino A al Bar 1

.....

.....

.....

A5 = numero di casse di birra da spedire dal Magazzino A al Bar 5

B1 = numero di casse di birra da spedire dal Magazzino B al Bar 1

.....

.....

.....

B5 = numero di casse di birra da spedire dal Magazzino B al Bar 5

Vincoli

$$A1 + A2 + A3 + A4 + A5 \leq 1000$$

$$B1 + B2 + B3 + B4 + B5 \leq 4000$$

$$A1 + B1 \geq 500$$

$$A2 + B2 \geq 900$$

$$A3 + B3 \geq 1800$$

$$A4 + B4 \geq 200$$

$$A5 + B5 \geq 700$$

```

from pulp import *

# Creazione lista magazzini
magazzini = ["A", "B"]

# Creazione di un dizionario che associa ad ogni magazzino la propria disponibilita' di casse
fornitura = {"A": 1000, "B": 4000}

# Creazione lista bar
bar = ["1", "2", "3", "4", "5"]

# Creazione di un dizionario che associa ad ogni bar la quantita' ordinata
ordini = {
    "1": 500,
    "2": 900,
    "3": 1800,
    "4": 200,
    "5": 700,
}

# Creazione di una matrice dei costi
costi = [
    # Bar
    # 1 2 3 4 5
    [2, 4, 5, 2, 1], # A   Magazzini
    [3, 1, 3, 2, 3], # B
]


# I dati dei costi vengono trasformati in un dizionario associato alle liste di magazzini e bar
costi = makeDict([magazzini, bar], costi, 0)

# Creazione della variabile problema
prob = LpProblem("Problema di trasporto della birra", LpMinimize)

# Creazione di una lista di tuple che contiene ogni possibile associazione di consegna magazzino-bar
consegne = [(m, b) for m in magazzini for b in bar]

# viene creato un dizionario per contenere le variabili decisionali riferite alle consegne
variabili = LpVariable.dicts("Consegna", (magazzini, bar), 0, None, LpInteger)

```



Creazione dati del problema

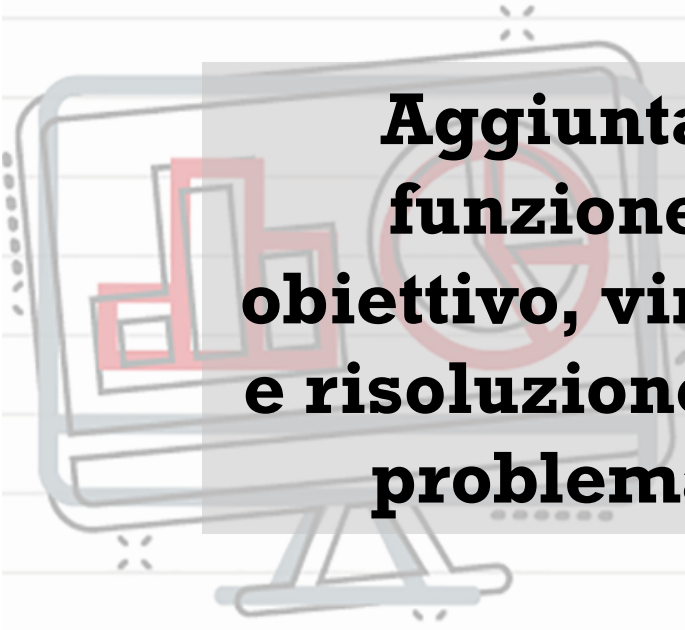
```
# Viene aggiunta la funzione obiettivo al problema
prob += (
    lpSum([variabili[m][b] * costi[m][b] for (m, b) in consegne]),
    "Somma totale dei costi di consegna",
)

# Vengono aggiunti al problema i vincoli di fornitura dei magazzini
for m in magazzini:
    prob += (
        lpSum([variabili[m][b] for b in bar]) <= fornitura[m],
        f"Somma dei prodotti inviati dal magazzino_{m}",
    )

# Vengono aggiunti al problema i vincoli di richiesta degli ordini da parte dei bar
for b in bar:
    prob += (
        lpSum([variabili[m][b] for m in magazzini]) >= ordini[b],
        f"Somma dei prodotti ricevuti dal bar{b}",
    )

# Il problema viene risolto utilizzando il risolutore scelto da PuLP
prob.solve()
```

```
Status: Optimal
Consegna_A_1 = 300
Consegna_A_2 = 0
Consegna_A_3 = 0
Consegna_A_4 = 0
Consegna_A_5 = 700
Consegna_B_1 = 200
Consegna_B_2 = 900
Consegna_B_3 = 1800
Consegna_B_4 = 200
Consegna_B_5 = 0
Costo totale del trasporto = 8600.0
```



**Aggiunta
funzione
obiettivo, vincoli
e risoluzione del
problema**



**Visualizzazione
dei
risultati**



UNIVERSITÀ DELLA
CALABRIA

DIPARTIMENTO DI **ECONOMIA,
STATISTICA E FINANZA**
"Giovanni Anania"

**GRAZIE PER
L'ATTENZIONE**