



UNIVERSITÀ  
di **VERONA**

Dipartimento  
di **INFORMATICA**

# Ricerca Operativa

## Modellizzare problemi di ottimizzazione

### Parte 1

---

*Laurea triennale in Matematica Applicata*  
*Anno Accademico 2021/2022*

Alice Raffaele – [alice.raffaele@univr.it](mailto:alice.raffaele@univr.it)

# Introduzione

---

## Esempio 1 – Insalata e pomodori

Un'azienda agricola deve determinare quanti ettari di terreno devono essere dedicati alla produzione di lattuga e pomodori. Si è stimato che, coltivando un ettaro di terreno, si possono produrre annualmente 20 quintali di lattuga e 30 quintali di pomodori. Per portare a termine le coltivazioni, l'azienda dovrà assegnare un suo bracciante ad ogni ettaro coltivato a lattuga e due braccianti ad ogni ettaro coltivato a pomodori. Per avere sufficiente manodopera per le altre coltivazioni, l'azienda non vuole utilizzare più di 100 lavoratori. Sapendo che l'azienda vende ogni chilogrammo di lattuga e pomodoro rispettivamente a 1 euro e a 1.5 euro, e vuole assicurarsi un profitto annuo di almeno 50000 euro dalla vendita di questi due prodotti, quanti ettari dovrà dedicare alla coltivazione di lattuga e quanti alla coltivazione di pomodori per minimizzare il numero complessivo di ettari coltivati?

## Esempio 2 – Mansioni domestiche

Anna e Zeno sono appena andati a convivere e vogliono suddividere tra loro i principali lavori domestici: fare la spesa, cucinare, lavare i piatti, fare il bucato, stirare e buttare la spazzatura. La loro bravura nei vari lavori è tuttavia diversa, e si riflette sui tempi necessari per svolgerli, riportati nella tabella seguente (in minuti):

	Spesa	Cucinare	Piatti	Bucato	Ferro da stiro	Rifiuti
<b>Anna</b>	25	30	10	8,5	42	6
<b>Zeno</b>	37	20	12	13	35	4

Come possono organizzarsi, in modo tale che ognuno di loro svolga esattamente tre mansioni e che il tempo totale impiegato sia minimo?

# Elementi principali di un problema di ottimizzazione

- Un problema,  $n$  istanze
- In ogni problema di ottimizzazione, ci sono tre elementi principali:



Variabili decisionali



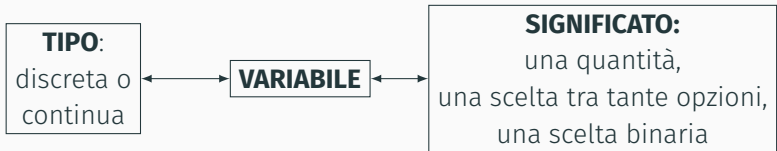
Vincoli



Funzione obiettivo

# Variabili decisionali

- Quali sono le **decisioni da prendere** nel problema che stiamo considerando?
- Quali sono le **incognite** che possono influenzare il valore della soluzione che si otterrebbe?
- **Nota:** occhio alla differenza con i parametri del problema, ovvero le informazioni che potrebbero sì variare da istanza a istanza, ma che sono comunque determinate.



## Esempio 1 – Variabili decisionali

Si introducono:

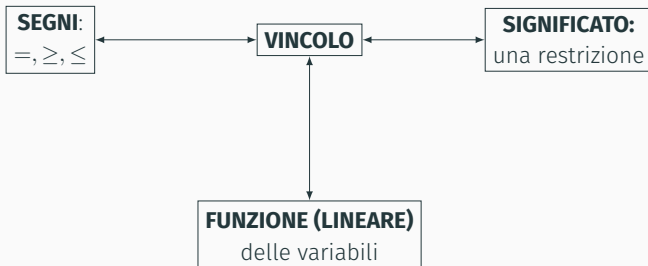
- $x_L$ , indicante il numero di ettari da dedicare alla coltivazione di lattuga;
- $x_P$ , indicante il numero di ettari da dedicare alla coltivazione di pomodori.

Sia  $x_L$  sia  $x_P$  sono quantità:

- non negative  $\rightarrow x_L \geq 0, x_P \geq 0$ ;
- continue  $\rightarrow x_L, x_P \in \mathbb{R}^+$ .

In forma compatta:  $x_i \in \mathbb{R}^+, \forall i \in \{L, P\}$ .

- Alcune variabili potrebbero non poter assumere tutti i valori possibili perché ci sono alcune **restrizioni** che limitano il loro dominio.
- Un vincolo è rappresentato da un'equazione o una disequazione formulata **usando le variabili** del problema.
- In molti casi, queste funzioni sono **lineari**.





## Esempio 1 – Vincoli

Si deve tenere in considerazione che:

- per ogni ettaro coltivato a lattuga si impegna un bracciante, per ogni ettaro coltivato a pomodori se ne impegnano due, e il massimo numero di braccianti è 100:

$$x_L + 2x_P \leq 100;$$

- ogni ettaro di terreno produce 20 quintali di lattuga o 30 quintali di pomodori all'anno; ogni chilogrammo di lattuga è venduto a 1 € e ogni chilogrammo di pomodori è venduto a 1,5 €; il profitto annuo dev'essere almeno di 50000 euro:

$$20 \cdot 100 \cdot 1 \cdot x_L + 30 \cdot 100 \cdot 1.5 \cdot x_P \geq 50000.$$

# Funzione obiettivo

- In base a quale *criterio* decidiamo che una soluzione sia ottima?
- Tale criterio è definito dalla **funzione obiettivo**, un'altra funzione (molto spesso lineare) delle variabili.



## Esempio 1 – Funzione obiettivo e modello risultante

Si vuole minimizzare il numero complessivo di ettari coltivati:

$$\min \quad x_L + x_P.$$

**Modello risultante:**

$$\min \quad x_L + x_P$$

$$x_L + 2x_P \leq 100$$

$$2000x_L + 4500x_P \geq 50000$$

$$x_i \in \mathbb{R}^+, \quad \forall i \in \{L, P\}.$$

## Esempio 2 – Modellizzazione

### Costruiamo il modello:

- Sia  $N := \{A, Z\}$  l'insieme delle persone (Anna e Zeno) e sia  $M := \{S, C, P, B, F, R\}$  l'insieme delle mansioni.
- **Variabili:** introduciamo una *variabile binaria*  $x_{i,j}$  per ogni persona  $i$  e per ogni mansione  $j$ , tale che

$$x_{i,j} = \begin{cases} 1, & \text{se } i \text{ compie la mansione } j; \\ 0, & \text{altrimenti.} \end{cases}$$

Per esempio,  $x_{A,S} = 1$  indica che Anna si occuperà di fare la spesa.

- **Vincoli:**

- ogni persona deve svolgere tre mansioni:

$$x_{A,S} + x_{A,C} + x_{A,P} + x_{A,B} + x_{A,F} + x_{A,R} = 3$$

$$x_{Z,S} + x_{Z,P} + x_{Z,B} + x_{Z,F} + x_{Z,R} = 3$$

in forma compatta:

$$\sum_{j \in M} x_{i,j} = 3, \quad \forall i \in N$$

- ogni mansione può essere svolta solo da una persona:

$$x_{A,S} + x_{Z,S} = 1$$

$$x_{A,C} + x_{Z,C} = 1$$

...

$$x_{A,R} + x_{Z,R} = 1$$

in forma compatta:

$$\sum_{i \in N} x_{i,j} = 1, \quad \forall j \in M$$

- **Funzione obiettivo:** minimizzare il tempo totale impiegato

$$\begin{aligned} \min \quad & 25x_{A,S} + 30x_{A,C} + 10x_{A,P} + 8,5x_{A,B} + 42x_{A,F} + 6x_{A,R} + \\ & + 37x_{Z,S} + 20x_{Z,C} + 12x_{Z,P} + 13x_{Z,B} + 35x_{Z,F} + 4x_{Z,R} \end{aligned}$$

in forma compatta:

$$\min \sum_{i \in N} \sum_{j \in M} t_{i,j} x_{i,j}.$$

### Modello risultante:

$$\min \sum_{i \in N} \sum_{j \in M} t_{i,j} x_{i,j}$$

$$\sum_{j \in M} x_{i,j} = 3, \quad \forall i \in N$$

$$\sum_{i \in N} x_{i,j} = 1, \quad \forall j \in M$$

$$x_{i,j} \in \{0, 1\}, \quad \forall i \in N, \forall j \in M.$$

- La Programmazione Matematica è lo sforzo condiviso della comunità scientifica per **definire e studiare modelli** di ottimizzazione.
- Ogni volta che consideriamo classi di variabili, vincoli e funzioni obiettivo diverse, otteniamo un modello diverso.
- Noi ci concentreremo su **alcuni paradigmi** lineari, ma in realtà è un'area molto più ampia che include anche la programmazione:
  - multi-obiettivo;
  - stocastica;
  - robusta;
  - e tante altre tipologie.



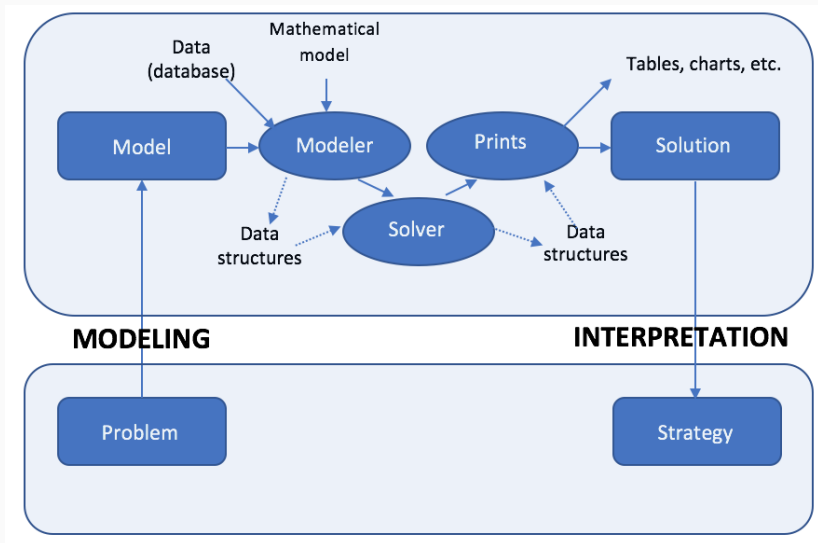
# Programmazione Lineare, Lineare Intera, e Mista

Paradigma	Caratteristica	Complessità
<b>Programmazione Lineare (PL)</b>	tutte le variabili sono continue	P
<b>Programmazione Lineare Intera (PLI)</b>	tutte le variabili sono intere	NP-HARD
<b>Programmazione Lineare Mista</b>	alcune variabili sono continue e altre sono intere	NP-HARD
<b>Programmazione Binaria (0-1 PL)</b>	tutte le variabili possono assumere solo 0 o 1 come valore	NP-HARD

## Dalla teoria della complessità computazionale:

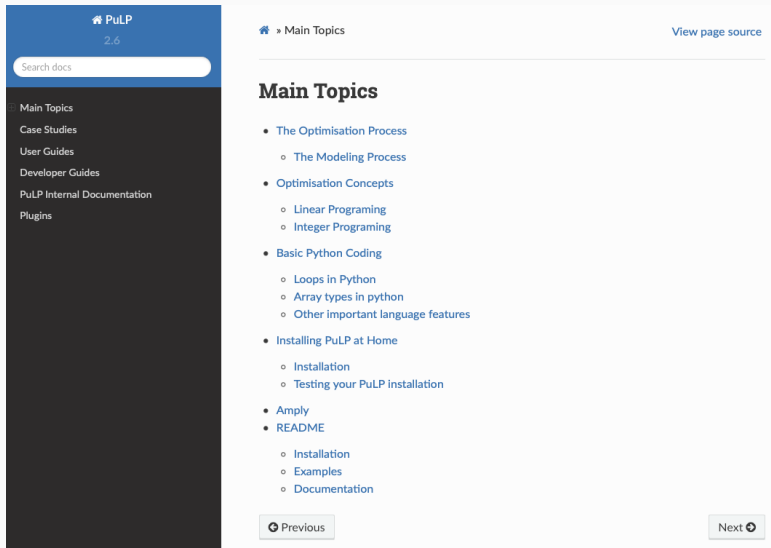
- P: classe di problemi per i quali esiste un algoritmo di tempo polinomiale per risolverli → Questi problemi sono FACILI;
- NP: classe di problemi le cui soluzioni possono essere verificate in tempo polinomiale ( $P \subset NP$ );
- NP-HARD: classe di problemi che sono “almeno tosti quanto i problemi più tosti in NP” → Questi problemi sono DIFFICILI.

# E dopo la formulazione del modello?





# La libreria PuLP per Python



PuLP  
2.6

Search docs

- Main Topics
- Case Studies
- User Guides
- Developer Guides
- PuLP Internal Documentation
- Plugins

» Main Topics [View page source](#)

## Main Topics

- [The Optimisation Process](#)
  - [The Modeling Process](#)
- [Optimisation Concepts](#)
  - [Linear Programing](#)
  - [Integer Programing](#)
- [Basic Python Coding](#)
  - [Loops in Python](#)
  - [Array types in python](#)
  - [Other important language features](#)
- [Installing PuLP at Home](#)
  - [Installation](#)
  - [Testing your PuLP installation](#)
- [Amplify](#)
- [README](#)
  - [Installation](#)
  - [Examples](#)
  - [Documentation](#)

[⬅ Previous](#) [Next ➡](#)

Link: <https://github.com/coin-or/pulp>

# Esempio 1 – Implementazione con PuLP

```
from pulp import *

# Inizializzazione del problema assegnando un nome e la direzione dell'ottimizzazione
model = LpProblem("Ettari", LpMinimize)

# Variabili
xL = LpVariable("Num_ettari_lattuga", 0, None, LpContinuous)
xP = LpVariable("Num_ettari_pomodori", 0, None, LpContinuous)

# Vincoli
model += 1*xL + 2*xP <= 100
model += 2000*xL + 4500*xP >= 50000

# Funzione obiettivo
model += xL + xP

# Chiamata al solver
model.solve()

# Stampa soluzione ottima trovata
for v in model.variables():
    print(v.name, "=", round(v.varValue, 2))

# Valore della funzione obiettivo
print("Numero minimo di ettari richiesti = {}".format(round(value(model.objective), 2)))
```

## Esempio 2 – Implementazione con PuLP

```
from pulp import *

# Inizializzazione del problema (nome + min/max)
model = LpProblem("MansioniDomestiche", LpMinimize)

# Variabili
xAS = LpVariable("Anna-Spesa", 0, 1, LpBinary)
xAC = LpVariable("Anna-Cucinare", 0, 1, LpBinary)
xAP = LpVariable("Anna-Piatti", 0, 1, LpBinary)
xAB = LpVariable("Anna-Bucato", 0, 1, LpBinary)
xAF = LpVariable("Anna-Ferro", 0, 1, LpBinary)
xAR = LpVariable("Anna-Rifiuti", 0, 1, LpBinary)
xZS = LpVariable("Zeno-Spesa", 0, 1, LpBinary)
xZC = LpVariable("Zeno-Cucinare", 0, 1, LpBinary)
xZP = LpVariable("Zeno-Piatti", 0, 1, LpBinary)
xZB = LpVariable("Zeno-Bucato", 0, 1, LpBinary)
xZF = LpVariable("Zeno-Ferro", 0, 1, LpBinary)
xZR = LpVariable("Zeno-Rifiuti", 0, 1, LpBinary)

# Funzione obiettivo
model += 25*xAS + 30*xAC + 10*xAP + 8.5*xAB + 42*xAF + 6*xAR +
        + 37*xZS + 20*xZC + 12*xZP + 13*xZB + 35*xZF + 4*xZR

# Vincoli
model += xAS + xAC + xAP + xAB + xAF + xAR == 3
model += xZS + xZC + xZP + xZB + xZF + xZR == 3
model += xAS + xZS == 1
model += xAC + xZC == 1
model += xAP + xZP == 1
model += xAB + xZB == 1
model += xAF + xZF == 1
model += xAR + xZR == 1

# Chiamata al solver
model.solve()
```

## Esempio 2 – Implementazione con PuLP (più compatta)

```
from pulp import *

# Inizializzazione del problema assegnando un nome e la direzione dell'ottimizzazione
model = LpProblem("MansioniDomestiche", LpMinimize)

# Set e parametri
N = ["Anna", "Zeno"]
M = ["Spesa", "Cucinare", "Piatti", "Bucato", "Ferro", "Rifiuti"]

tempi = {
    "Anna": {"Spesa": 25, "Cucinare": 30, "Piatti": 10, "Bucato": 8.5, "Ferro": 42, "Rifiuti": 6},
    "Zeno": {"Spesa": 37, "Cucinare": 20, "Piatti": 12, "Bucato": 13, "Ferro": 35, "Rifiuti": 4}
}
Assegnamenti = [(i,j) for i in N for j in M]

# Variabili
vars = LpVariable.dicts("x", (N, M), 0, 1, LpBinary)

# Funzione obiettivo
model += lpSum(vars[i][j] * tempi[i][j] for (i,j) in Assegnamenti)

# Vincoli
for i in N:
    model += lpSum(vars[i][j] for j in M) == len(M)/2

for j in M:
    model += lpSum(vars[i][j] for i in N) == 1

# Chiamata al solver
model.solve()

# Stampa soluzione ottima trovata
for v in model.variables():
    print(v.name, "=", v.varValue)

# Valore della funzione obiettivo
print("Tempo_totale_minimo_richiesto = {} minuti".format(round(value(model.objective),2)))
```

## **Problemi classici**

---



## Zaino – Una vecchia chiavetta USB

Volete realizzare una playlist musicale avendo a disposizione una raccolta di brani alta risoluzione e una vecchia chiavetta USB, parzialmente usata, in cui sono disponibili soltanto 140 MB. Volete che la vostra playlist consti almeno di 6 brani di cui al massimo 2 con titolo inglese. L'indice di gradimento (in una scala da 1 a 10) e la dimensione in MB di ogni file sono riportati nella tabella seguente:

	Canzone	Gradimento	Dimensione (MB)
1	Perfect	9,5	25
2	No roots	8	20
3	La musica non c'è	9	30
4	Musica leggerissima	7	20
5	Zitti e buoni	6,5	18
6	Resta qui	9	22
7	Famous blue raincoat	10	27
8	Rolling in the deep	8,5	19

Volete decidere quali canzoni inserire sulla chiavetta, in modo tale da massimizzare il gradimento complessivo senza eccedere la capacità disponibile.

## Costruiamo il modello:

- Sia  $I = \{1, 2, 3, 4, 5, 6, 7, 8\}$  l'insieme delle canzoni; ogni canzone  $i$  è associata a una preferenza  $p_i$  e a una dimensione  $d_i$ .
- **Variabili:** introduciamo una variabile binaria  $x_i$  per ogni  $i \in I$ , che indica se la canzone  $i$  viene caricata sulla chiavetta USB ( $x_i = 1$ ) oppure no ( $x_i = 0$ ).
- **Vincoli:**
  - capacità massima della chiavetta USB:  $\sum_{i \in I} d_i x_i \leq 140$ ;
  - almeno sei brani:  $\sum_{i \in I} x_i \geq 6$
  - al massimo due brani con titolo inglese:  $x_1 + x_2 + x_7 + x_8 \leq 2$ ;
- **Funzione obiettivo:** massimizzare il gradimento delle canzoni selezionate:  $\max \sum_{i \in I} p_i x_i$ .

## Modello risultante:

$$\max \sum_{i \in I} p_i x_i$$

$$\sum_{i \in I} d_i x_i \leq 140$$

$$\sum_{i \in I} x_i \geq 6$$

$$x_1 + x_2 + x_7 + x_8 \leq 2$$

$$x_i \in \{0, 1\}, \quad \forall i \in I$$

# Implementazione con PuLP:

```
from pulp import *

# Inizializzazione del problema assegnando un nome e la direzione dell'ottimizzazione
model = LpProblem("ChiavettaUSB", LpMaximize)

# Set e parametri
I = [1,2,3,4,5,6,7,8]
preferenze = {1: 9.5, 2: 8, 3: 9, 4: 7, 5: 6.5, 6: 9, 7: 10, 8: 8.5}
dimensione = {1: 25, 2: 20, 3: 30, 4: 20, 5: 18, 6: 22, 7: 27, 8: 19}
capacita_USB = 140

# Variabili
vars = LpVariable.dicts("x", I, 0, 1, LpBinary)

# Funzione obiettivo
model += lpSum(vars[i] * preferenze[i] for i in I)

# Vincoli
model += lpSum(vars[i]*dimensione[i] for i in I) <= 140
model += lpSum(vars[i] for i in I) >= 6
model += vars[1] + vars[2] + vars[7] + vars[8] <= 2

# Chiamata al solver
model.solve()

# Stampa soluzione ottima trovata
for v in model.variables():
    print(v.name, "=", v.varValue)

# Valore della funzione obiettivo
print("Gradimento totale canzoni =", format(round(value(model.objective),2)))
```

## Produzione – Su due ruote

L'azienda Passione2ruote di Catania produce tre prodotti: biciclette, ciclomotori, e tricicli per bambini. Per un periodo di produzione, sono disponibili i seguenti dati riguardanti il profitto, il costo di produzione (in euro) e lo spazio occupato in magazzino (storage, in  $m^3$ ) di ogni unità prodotta:

	<b>Biciclette</b>	<b>Ciclomotori</b>	<b>Tricicli</b>
<b>Profitto</b>	100	300	50
<b>Costo</b>	300	1200	120
<b>Storage</b>	0.5	1	0.5

L'azienda dispone di un capitale massimo di 93.000 euro e possiede uno storage disponibile di 101  $m^3$ . Quante unità di ciascun prodotto dovrebbero essere prodotte per massimizzare il profitto totale?

## Costruiamo il modello:

- Sia  $P := \{B, C, T\}$  l'insieme dei prodotti dell'azienda; ogni unità del prodotto  $i$  è associata a un profitto  $p_i$ , un costo  $c_i$  e uno spazio occupato  $s_i$ .
- **Variabili:** si introduce una variabile intera non negativa  $x_i \geq 0$  per ogni prodotto  $i \in P$ , indicante quante unità del prodotto  $i$  devono essere realizzate.
- **Vincoli:**
  - capitale massimo:  $\sum_{i \in P} c_i x_i \leq 93000$ ;
  - storage disponibile:  $\sum_{i \in P} s_i x_i \leq 101$ ;
- **Funzione obiettivo:** massimizzare il profitto:  $\max \sum_{i \in I} p_i x_i$ .

## Modello risultante:

$$\max \sum_{i \in I} p_i x_i$$

$$\sum_{i \in P} c_i x_i \leq 93000$$

$$\sum_{i \in P} s_i x_i \leq 101$$

$$x_i \geq 0, \quad \forall i \in P.$$

# Implementazione con PuLP:

```
from pulp import *

# Inizializzazione del problema assegnando un nome e la direzione dell'ottimizzazione
model = LpProblem("2Ruote", LpMaximize)

# Set e parametri
P = {"Biciclette", "Ciclomotori", "Tricicli"}
profitti = {"Biciclette": 100, "Ciclomotori": 300, "Tricicli": 50}
costi = {"Biciclette": 300, "Ciclomotori": 1200, "Tricicli": 50}
storage = {"Biciclette": 0.5, "Ciclomotori": 1, "Tricicli": 0.5}
capitale = 93000
storage_max = 101

# Variabili
vars = LpVariable.dicts("x", P, 0, None, LpContinuous)

# Funzione obiettivo
model += lpSum(vars[i] * profitti[i] for i in P)

# Vincoli
model += lpSum(vars[i]*costi[i] for i in P) <= capitale
model += lpSum(vars[i]*storage[i] for i in P) <= storage_max

# Chiamata al solver
model.solve()

# Stampa soluzione ottima trovata
for v in model.variables():
    print(v.name, "=", v.varValue)

# Valore della funzione obiettivo
print("Profitto massimo=", format(round(value(model.objective),2)))
```



## Dieta – Dal nutrizionista

Secondo un nutrizionista sostenitore della dieta mediterranea, le quantità minime di nutrienti che devono essere assunte ogni giorno sono 1700 chilocalorie, 200 g di carboidrati, 70 g di proteine, 60 g di grassi e 0,7 g di calcio. Generalmente, il nutrizionista è solito prescrivere una dieta composta da otto alimenti: pane, pasta, latte, uova, pollo, tonno, cioccolato e verdure. La seguente tabella mostra quante calorie (in Kcal), carboidrati, proteine, grassi (in grammi) e calcio (in mg) fornisce una porzione di ogni alimento:

	Pane	Pasta	Latte	Uova	Pollo	Tonno	Cioccolato	Verdure
<b>Calorie (kcal)</b>	150	390	70	70	150	150	112	45
<b>Carboidrati (g)</b>	30	75	5	0	2	0	7	8
<b>Proteine (g)</b>	5	11	5	6	36	25	2	3
<b>Grassi (g)</b>	2	3	3	6	5	15	10	2
<b>Calcio (mg)</b>	52	5	150	50	22	4	11	50

Il nutrizionista raccomanda anche almeno due porzioni di verdura al giorno, mentre il numero massimo di porzioni per ogni alimento è riportato nella tabella seguente:

	Pane	Pasta	Latte	Uova	Pollo	Tonno	Cioccolato	Verdure
<b>N° max di porzioni</b>	2	2	2	1	1	2	2	6

Il numero di porzioni di pane e pasta non può essere maggiore di 3, mentre quello delle porzioni di latte, pollo e tonno deve essere almeno 4. Il costo (in €) di una porzione di ogni alimento è il seguente:

	Pane	Pasta	Latte	Uova	Pollo	Tonno	Cioccolato	Verdure
<b>Costo per porzione</b>	0,50	3,50	1,00	1,50	4,50	2,00	1,50	4,00

Determinare quali alimenti dovranno essere mangiati in un giorno per rispettare tutte le prescrizioni della dieta indicata dal nutrizionista, in modo tale da minimizzare il costo totale.

## Costruiamo il modello:

- Siano  
 $C := \{Pane, Pasta, Latte, Uova, Pollo, Tonno, Cioccolato, Verdure\}$  e  
 $N := \{Kcal, Carboidrati, Proteine, Grassi, Calcio\}$  gli insiemi dei cibi e dei nutrienti. Ogni porzione di un cibo  $c \in C$  fornisce un certo valore  $v_{c,n}$  per ogni nutriente  $n \in N$  e costa  $s_c$ .
- Devono considerarsi un numero minimo  $n_{min_c}$  e massimo  $n_{max_c}$  di porzioni di un cibo  $c$ . Inoltre, anche un valore minimo  $q_n$  per ogni nutriente  $n \in N$  deve essere raggiunto.
- **Variabili:** si introduce una variabile continua non negativa  $x_c$  per ogni  $c \in C$ , indicante il numero di porzioni del cibo  $c$  da inserire nella dieta.
- **Vincoli:**
  - porzioni minime e massime di ogni cibo:

$$n_{min_c} \leq x_c \leq n_{max_c}, \quad \forall c \in C;$$

- quantità minima da raggiungere per ogni nutriente:

$$\sum_{c \in C} v_{c,n} x_c \geq q_n, \quad \forall n \in N;$$

- **Vincoli:**

- massimo numero di porzioni di pane e pasta:

$$x_{Pane} + x_{Pasta} \leq 3;$$

- minimo numero di porzioni di latte, uova, pollo e tonno:

$$x_{Latte} + x_{Uova} + x_{Pollo} + x_{Tonno} \geq 4;$$

- **Funzione obiettivo:** minimizzare il costo totale

$$\min \sum_{c \in C} s_c x_c.$$

## Modello risultante:

$$\min \sum_{c \in C} s_c x_c$$

$$\sum_{c \in C} v_{c,n} x_c \geq q_n, \quad \forall n \in N$$

$$x_{Pane} + x_{Pasta} \leq 3$$

$$x_{Latte} + x_{Uova} + x_{Pollo} + x_{Tonno} \geq 4$$

$$n_{min_c} \leq x_c \leq n_{max_c}, \quad \forall c \in C$$

$$x_c \geq 0, \quad \forall c \in C.$$

# Implementazione con PuLP:

```
from pulp import *

# Inizializzazione del problema assegnando un nome e la direzione dell'ottimizzazione
model = LpProblem("DietaMediterranea", LpMinimize)

# Set e parametri
C = ["Pane", "Pasta", "Latte", "Uova", "Pollo", "Tonno", "Cioccolato", "Verdure"]
N = ["Kcal", "Carboidrati", "Proteine", "Grassi", "Calcio"]
quantita_minime = {"Kcal": 1700, "Carboidrati": 200, "Proteine": 70, "Grassi": 60, "Calcio": 700}
valori = {
    "Pane": {"Kcal": 150, "Carboidrati": 30, "Proteine": 5, "Grassi": 2, "Calcio": 52},
    "Pasta": {"Kcal": 390, "Carboidrati": 75, "Proteine": 11, "Grassi": 3, "Calcio": 5},
    "Latte": {"Kcal": 70, "Carboidrati": 5, "Proteine": 5, "Grassi": 3, "Calcio": 150},
    "Uova": {"Kcal": 70, "Carboidrati": 0, "Proteine": 6, "Grassi": 6, "Calcio": 50},
    "Pollo": {"Kcal": 150, "Carboidrati": 2, "Proteine": 36, "Grassi": 5, "Calcio": 22},
    "Tonno": {"Kcal": 150, "Carboidrati": 0, "Proteine": 25, "Grassi": 15, "Calcio": 4},
    "Cioccolato": {"Kcal": 112, "Carboidrati": 7, "Proteine": 2, "Grassi": 10, "Calcio": 11},
    "Verdure": {"Kcal": 45, "Carboidrati": 8, "Proteine": 3, "Grassi": 2, "Calcio": 50},
}
CN = [(i, j) for i in C for j in N]

porzioni_minime = {"Pane": 0, "Pasta": 0, "Latte": 0, "Uova": 0, "Pollo": 0, "Tonno": 0,
                   "Cioccolato": 0, "Verdure": 2}
porzioni_massime = {"Pane": 2, "Pasta": 2, "Latte": 2, "Uova": 1, "Pollo": 1, "Tonno": 2,
                    "Cioccolato": 2, "Verdure": 6}
costo_porzione = {"Pane": 0.5, "Pasta": 3.5, "Latte": 1, "Uova": 1.5, "Pollo": 4.5, "Tonno": 2,
                  "Cioccolato": 1.5, "Verdure": 4}
max_porzioni_pane_pasta = 3
min_porzioni_latte_pollo_tonno = 4
```

```

# Variabili
vars = LpVariable.dicts("x", C, 0, None, LpContinuous)

# Funzione obiettivo
model += lpSum(vars[c] * costo_porzione[c] for c in C)

# Vincoli
for n in N:
    model += lpSum(vars[c]*valori[c][n] for c in C) >= quantita_minime[n]

for c in C:
    model += vars[c] >= porzioni_minime[c]
    model += vars[c] <= porzioni_massime[c]

model += vars["Pane"] + vars["Pasta"] <= max_porzioni_pane_pasta
model += vars["Latte"] + vars["Pollo"] + vars["Tonno"] >= min_porzioni_latte_pollo_tonno

# Chiamata al solver
model.solve()

# Stampa soluzione ottima trovata
for v in model.variables():
    print(v.name, " = ", round(v.varValue, 2))

# Valore della funzione obiettivo
print("Costo minimo dieta = ", format(round(value(model.objective), 2)))

```

## Trasporto – Buongiorno, kaffè?

Una catena di bar ha stipulato un contratto commerciale con un'industria di torrefazione per la fornitura esclusiva di caffè. L'industria ha a disposizione due impianti di torrefazione  $T_1$  e  $T_2$  con cui potrà rifornire i tre bar  $B_1$ ,  $B_2$  e  $B_3$  della catena. Vista la differente distanza tra gli impianti e i bar e i differenti mezzi di trasporto utilizzati, i costi di trasporto €/Kg di caffè da un impianto a un bar risultano differenti e sono riassunti nella seguente tabella:

	$B_1$	$B_2$	$B_3$
$T_1$	0,4	0,3	0,2
$T_2$	0,2	0,3	0,5

Sapendo che gli impianti di torrefazione  $T_1$  e  $T_2$  possono produrre giornalmente al massimo 54 e 44 kg di caffè e che i tre bar necessitano di 35, 30 e 33 kg di caffè, qual è la quantità da trasportare da ogni impianto a ogni bar per minimizzare i costi?



## Costruiamo il modello:

- Siano  $T := \{1, 2\}$  e  $B := \{1, 2, 3\}$  l'insieme degli impianti di torrefazione e dei bar. Sia  $c_{i,j}$  il costo di trasporto €/Kg dall'impianto  $i$  al bar  $j$ . Sia  $p_i$  la massima quantità in Kg di caffè che può produrre l'impianto  $i$  e sia  $d_j$  la domanda del bar  $j$ .
- **Variabili:** si introduce una variabile continua e non negativa  $x_{i,j}$  per ogni impianto  $i \in T$  e per ogni bar  $j \in B$ , che indica la quantità in Kg di caffè da trasportare dall'impianto  $i$  al bar  $j$ .
- **Vincoli:**
  - capacità di produzione di ogni impianto:

$$\sum_{j \in B} x_{i,j} \leq p_i, \quad \forall i \in T;$$

- domanda dei bar:

$$\sum_{i \in T} x_{i,j} \geq d_j, \quad \forall j \in B;$$

- **Funzione obiettivo:** minimizzare i costi di trasporto

$$\min \sum_{i \in T} \sum_{j \in B} c_{i,j} x_{i,j}.$$

### Modello risultante:

$$\min \sum_{i \in T} \sum_{j \in B} c_{i,j} x_{i,j}$$

$$\sum_{j \in B} x_{i,j} \leq p_i, \quad \forall i \in T$$

$$\sum_{i \in T} x_{i,j} \geq d_j, \quad \forall j \in B$$

$$x_{i,j} \geq 0, \quad \forall i \in T, \forall j \in B.$$

# Implementazione con PuLP:

```
from pulp import *

# Inizializzazione del problema assegnando un nome e la direzione dell'ottimizzazione
model = LpProblem("TrasportoCaffe", LpMinimize)

# Set e parametri
T = [1,2]
B = [1,2,3]
capacita_prod = {1: 54, 2: 44}
domanda = {1: 35, 2: 30, 3: 33}
costi = {
    1: {1: 0.4, 2: 0.3, 3: 0.2},
    2: {1: 0.2, 2: 0.3, 3: 0.5}}

# Variabili
x = LpVariable.dicts("x", (T, B), 0, None, LpContinuous)

# Funzione obiettivo
model += lpSum(x[i][j]*costi[i][j] for i in T for j in B)

# Vincoli
for i in T:
    model += lpSum(x[i][j] for j in B) <= capacita_prod[i]

for j in B:
    model += lpSum(x[i][j] for i in T) >= domanda[j]

# Chiamata al solver
model.solve()

# Stampa soluzione ottima trovata
for v in model.variables():
    print(v.name, "=", v.varValue)

# Valore della funzione obiettivo
print("Costo minimo per il trasporto =", format(round(value(model.objective),2)))
```

## Assegnamento – Incarichi e bonus

Una compagnia finanziaria deve decidere chi assumere fra i tre candidati C1, C2 e C3. In base ai loro differenti curriculum, l'azienda sa che in caso di assunzione dovrà assicurare loro uno stipendio mensile fisso rispettivamente di 1450, 1600 e 1300 euro. Inoltre, nel mese corrente, la compagnia ha necessità di portare a termine tre progetti (LAV1, LAV2, LAV3) che richiedono diverse abilità ed esperienza. Al progetto LAV1 dovranno essere assegnate almeno 2 persone, agli altri due progetti almeno 1 persona ciascuno. In base all'assegnazione dei lavori ai candidati, la compagnia finanziaria dovrà retribuire i dipendenti con uno o più bonus in busta paga. La stima di tale bonus (€), riferito a ciascun candidato se fosse assegnato a ciascuno dei tre lavori, è riportata nella tabella seguente:

	LAV1	LAV2	LAV3
C1	150	230	110
C2	100	90	150
C3	350	410	210

Quali candidati assume e assegna ai vari lavori la compagnia finanziaria, in modo da minimizzare i costi totali?

## Costruiamo il modello:

- Sia  $C := \{1, 2, 3\}$  il set di candidati e  $L := \{1, 2, 3\}$  il set di lavori da portare a termine. Sia  $s_i$  lo stipendio mensile del candidato  $i$  e sia  $b_{i,j}$  il bonus da dare al candidato  $i$  qualora venisse assegnato al lavoro  $j$ . Ogni lavoro  $j$  deve essere svolto da almeno  $n_j$  persone.
- **Variabili:**
  - una variabile binaria  $y_i$  per ogni candidato  $i \in C$ , per indicare se  $i$  viene assunto o no;
  - una variabile binaria  $x_{i,j}$  per ogni candidato  $i \in C$  e per ogni lavoro  $j \in L$ , per indicare se il candidato  $i$  è assegnato al lavoro  $j$ .

- **Vincoli:**

- ogni lavoro deve essere svolto da almeno  $n_j$  persone:

$$\sum_{i \in C} x_{i,j} \geq n_j, \quad \forall j \in L;$$

- il candidato  $i$  può svolgere un lavoro  $j$  solo se viene assunto:

$$x_{i,j} \leq y_i, \quad \forall i \in C, \forall j \in L$$

- **Funzione obiettivo:** minimizzare i costi totali

$$\min \sum_{i \in C} s_i y_i + \sum_{i \in C} \sum_{j \in L} b_{i,j} x_{i,j}.$$

### Modello risultante:

$$\min \sum_{i \in C} s_i y_i + \sum_{i \in C} \sum_{j \in L} b_{i,j} x_{i,j}$$

$$\sum_{i \in C} x_{i,j} \geq n_j, \quad \forall j \in L$$

$$x_{i,j} \leq y_i, \quad \forall i \in C, \forall j \in L$$

$$x_{i,j}, y_i \in \{0, 1\}, \quad \forall i \in C, \forall j \in L.$$

# Implementazione con PuLP:

```
from pulp import *

# Inizializzazione del problema assegnando un nome e la direzione dell'ottimizzazione
model = LpProblem("AssegnazioneIncarichi", LpMinimize)

# Set e parametri
C = [1,2,3]
L = [1,2,3]
num_min_candidati_lavori = {1: 2, 2: 1, 3: 1}
stipendi = {1: 1450, 2: 1600, 3: 1300}
bonus = {
    1: {1: 150, 2: 230, 3: 110},
    2: {1: 100, 2: 90, 3: 150},
    3: {1: 350, 2: 410, 3: 210}}
Assegnazioni = [(i,j) for i in C for j in L]

# Variabili
vars_y = LpVariable.dicts("y", C, 0, 1, LpBinary)
vars_x = LpVariable.dicts("x", (C, L), 0, 1, LpBinary)

# Funzione obiettivo
model += lpSum(vars_y[i] * stipendi[i] for i in C) + lpSum(vars_x[i][j]*bonus[i][j] for i in C for j in L)

# Vincoli
for i in C:
    for j in L:
        model += vars_x[i][j] <= vars_y[i]
for j in L:
    model += lpSum(vars_x[i][j] for i in C) >= num_min_candidati_lavori[j]

# Chiamata al solver
model.solve()

# Stampa soluzione ottima trovata
for v in model.variables():
    print(v.name, "=", v.varValue)

# Valore della funzione obiettivo
print("Costo_minimo_per_la_compagnia_finanziaria_{}".format(round(value(model.objective),2)))
```