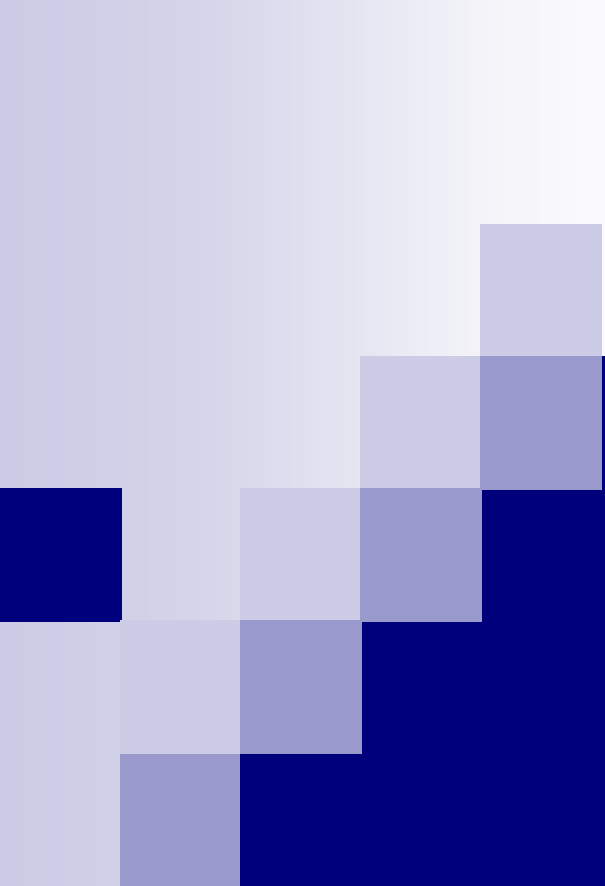




Ricerca Operativa

Laboratorio: utilizzo di solver
per programmazione
matematica

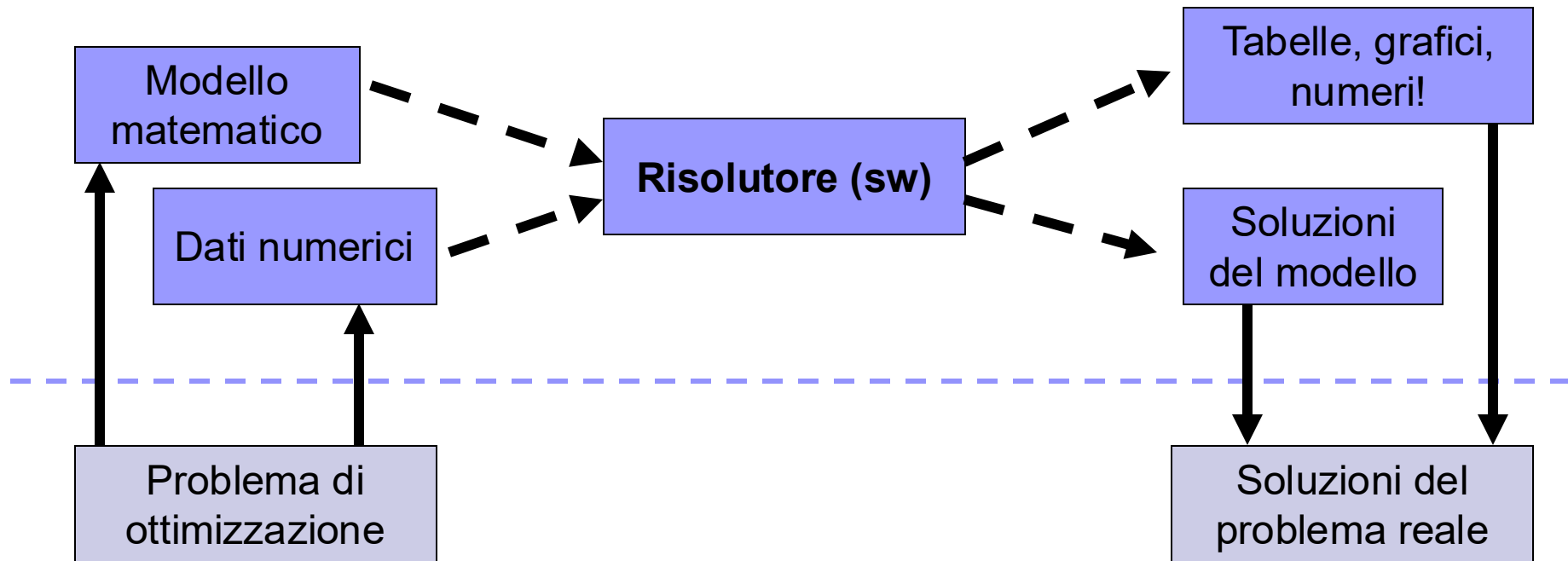
Elementi di un modello di Programmazione Matematica

- **Insiemi:** elementi del sistema;
- **Parametri:** dati del problema;
- **Variabili decisionali o di controllo:** grandezze sulle quali possiamo agire;
- **Vincoli:** relazioni matematiche che descrivono le condizioni di ammissibilità delle soluzioni.
- **Funzione obiettivo:** e la quantità da massimizzare o minimizzare.

Un modello **dichiara** le caratteristiche della soluzione ottima in linguaggio matematico

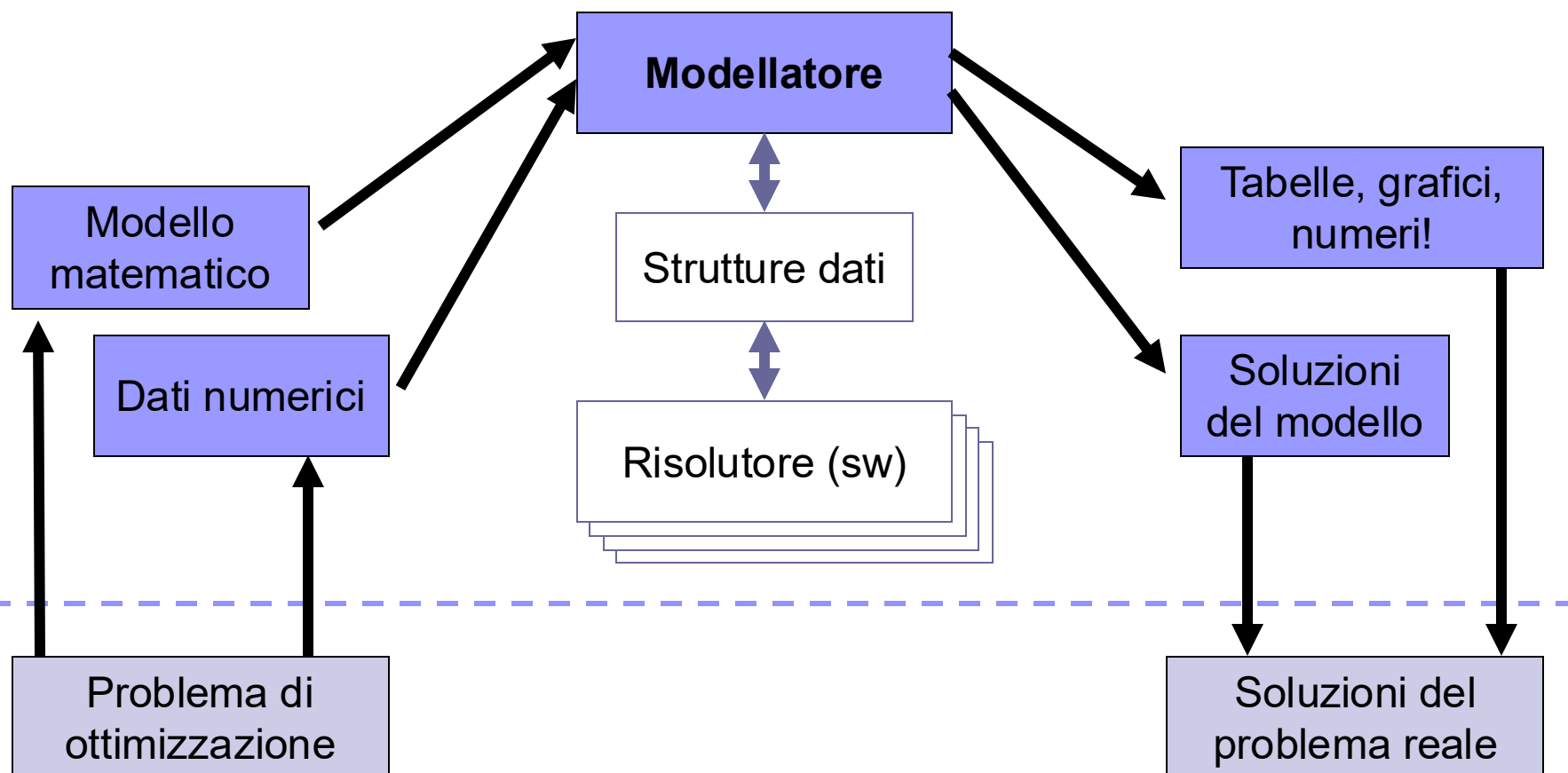
Utilizzo di solver

Un **solver** (o risolutore) è un software che riceve in **input** una descrizione di un problema di ottimizzazione e produce in **output** la soluzione ottima del modello e informazioni ad essa collegate.



Ruolo dei modellatori

Un **modellatore** fornisce un'interfaccia verso un risolutore.



Obiettivi dei modellatori

- Disporre di un **linguaggio** semplice:
 - ad alto livello;
 - simile a quello di modellazione (linguaggio matematico);
 - formalmente strutturato;
 - possibilità di commenti.
- Consentire la **separazione** tra **implementazione** del **modello** e implementazione delle tecniche di **soluzione**
- Dialogare con **diversi solver** (strutture di I/O standard).
- Mantenere la **separazione tra modello e dati** del problema: per cambiare l'istanza basta cambiare i dati, non il modello.
- Linguaggio per **script**.

Possibili configurazioni (alcune)

Modellatori:

☐ Foglio elettronico

☐ AMPL

☐ ZIMPL

☐ Lingo

☐ OPL

☐ Mosel

☐ **Python**, C, Java etc.

☐ Google OR Tools

☐ ...

Solver:

☐ Risolutore per fogli elettronici

☐ Cplex

☐ **Gurobi**

☐ Soplex + SCIP

☐ Xpress

☐ Minos (Progr. Non Lineare)

☐ Lindo

☐ GPLK (librerie)

☐ Baron

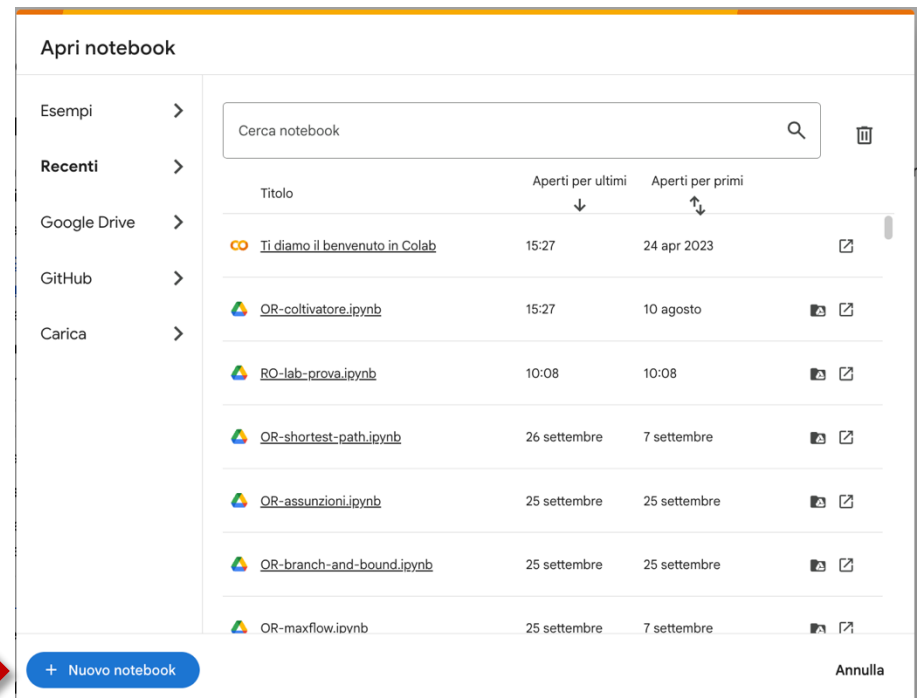
☐ Lp_solve

☐ Google GLOP

☐ ...

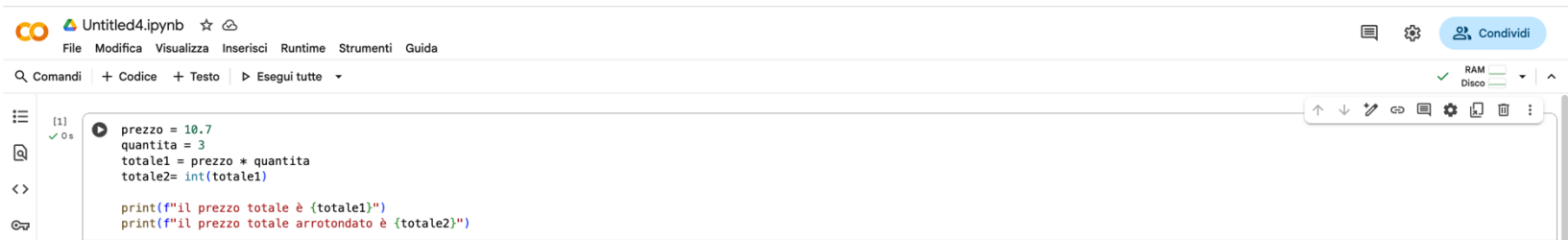
Introduzione a Colab

1. **Accedere al proprio account Google** (e.g. l'account accademico @studenti.unipd.it) [\[LINK\]](#)
preferibilmente utilizzare browser Chromium-based (e.g. Google Chrome, Chromium, etc)
2. **Aprire Google Colab** [\[LINK\]](#)
3. **Cliccare sul pulsante «+ Nuovo notebook»**



Introduzione a Colab I/III

1. **Copiare** il codice relativo che trovate sulla slide Introduzione a Python I/III ed **incollare** su Colab



The screenshot shows the Google Colab interface. At the top, there's a menu bar with 'File', 'Modifica', 'Visualizza', 'Inserisci', 'Runtime', 'Strumenti', and 'Guida'. Below the menu bar, there's a toolbar with 'Comandi', '+ Codice', '+ Testo', and 'Esegui tutte'. The main area contains a code cell with the following Python code:

```
[1] prezzo = 10.7
quantita = 3
totale1 = prezzo * quantita
totale2 = int(totale1)

print(f"il prezzo totale è {totale1}")
print(f"il prezzo totale arrotondato è {totale2}")
```

2. **Eeguire il codice** cliccando su «Esegui tutte» o sul pulsante in alto a sinistra della cella di codice



Introduzione a Colab II/III

L'**output** di una cella viene visualizzato al di sotto della cella stessa.

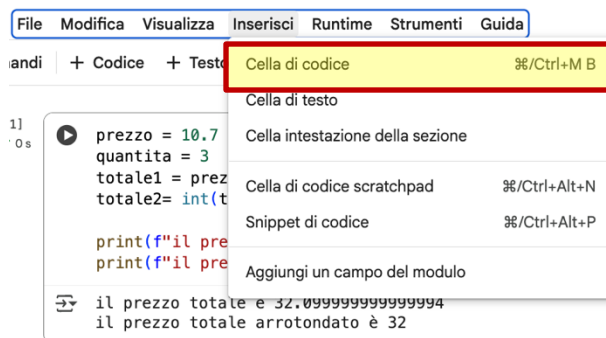
```
prezzo = 10.7
quantita = 3
totale1 = prezzo * quantita
totale2= int(totale1)

print(f"il prezzo totale è {totale1}")
print(f"il prezzo totale arrotondato è {totale2}")
```

```
il prezzo totale è 32.099999999999994
il prezzo totale arrotondato è 32
```

+ Codice

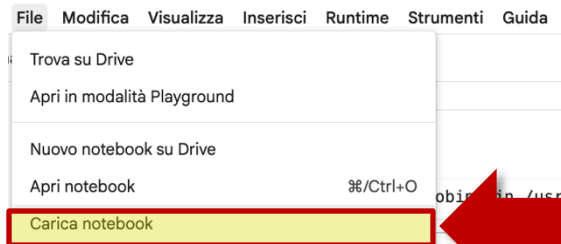
Per aggiungere una cella di codice è necessario premere il pulsante «+ Codice» o andare sul menù **Inserisci > Cella di codice**



Introduzione a Colab III/III

Per caricare un file su Colab:

1. Scaricare il file RO_lab_prova.ipynb [[LINK](#)]
2. File > Carica notebook



3. Trascinare o Caricare il file scaricato cliccando sul tasto **Sfoglia**



Introduzione a Python I/III

```
prezzo = 10.7
quantita = 3
totale1 = prezzo * quantita
totale2 = int(totale1)

print(f"il prezzo totale è {totale1}")
print(f"il prezzo totale arrotondato è {totale2}")
```

il prezzo totale è 32.099999999999994

il prezzo totale arrotondato è 32

Introduzione a Python II/III

```
lista = [41,27,1,12,7]

for i in range(len(lista)):
    print(f"Elemento di indice {i}: {lista[i]}")
    if(i>2):
        print(f"\telemento >= 2")
    else:
        print(f"\telemento < 2")
```

```
Valore elemento della lista di indice 0: 41
    elemento < 2
Valore elemento della lista di indice 1: 27
    elemento < 2
Valore elemento della lista di indice 2: 1
    elemento < 2
Valore elemento della lista di indice 3: 12
    elemento >= 2
Valore elemento della lista di indice 4: 7
    elemento >= 2
```

Introduzione a Python III/III

```
dizionario_eta = {  
    "Mario": 25,  
    "Luigi": 35,  
    "Cinzia": 20  
}  
  
for chiave, valore in dizionario_eta.items():  
    print(f"{chiave}: {valore}")  
  
print("---")  
  
for chiave in dizionario_eta.keys():  
    print(f"{chiave}: {dizionario_eta[chiave]}")
```

```
Mario: 25  
Luigi: 35  
Cinzia: 20  
---  
Mario: 25  
Luigi: 35  
Cinzia: 20
```

Introduzione a pandas I/IV

mix-ottimo-telecomandi.csv

```
i,telecomando_A,telecomando_B,quantita
display,1,2,10
logica,2,2,18
trasmettitori,1,1,12
tastierini,2,3,21
navigazione,1,0,9
led,1,1,10
profitto,3,8,nan
```

```
df = pd.read_csv("telecomandi.csv")
df = df.set_index('i')
print(df.head())
```

```
<bound method NDFrame.head of
      telecomando_A      telecomando_B      quantita
i
display           1           2          10.0
logica            2           2          18.0
trasmettitori     1           1          12.0
tastierini         2           3          21.0
navigazione        1           0           9.0
led                1           1          10.0
profitto           3           8           NaN
```

Introduzione a pandas II/IV

```
# seleziono riga indicizzata da "display"
display = df.loc["display"]

# seleziono colonna indicizzata da "telecomando_A"
telecomando_A = df["telecomando_A"]
```

```
<bound method NDFrame.head of
telecomando_A telecomando_B quantita
i
display 1 2 10.0
logica 2 2 18.0
trasmettitori 1 1 12.0
tastierini 2 3 21.0
navigazione 1 0 9.0
led 1 1 10.0
profitto 3 8 NaN>
```

Introduzione a pandas III/IV

```
# seleziono riga indicizzata da "display"
display = df.loc["display"].to_dict()
print(display)

{
    "telecomando_A": 1,
    "telecomando_B": 2
}
```

```
# seleziono colonna indicizzata da "telecomando_A"
telecomando_A = df["telecomando_A"].to_dict()
print(telecomando_A)

{
    "display": 1,
    "logica": 2,
    "trasmettitori": 1,
    "tastierini": 2,
    "navigazione": 1,
    "led": 1,
    "profitto": 3,
}
```

Introduzione a pandas IV/IV

```
# seleziono valore indicizzato da "display", "telecomando_A"
```

```
A = df.drop(index="profitto").drop(columns="quantita")
```

```
valore = A.loc["display", "telecomando_A"]
```

```
print(valore)
```

1

<bound method NDFrame.head of

	telecomando_A	telecomando_B
i		
display	1	2
logica	2	2
trasmettitori	1	1
tastierini	2	3
navigazione	1	0
led	1	1

Gurobi

- Gurobi è un tool di *decision intelligence*
- Tra i diversi moduli che offre, Gurobi Optimizer fornisce:
 - un **solver**, un programma che implementa algoritmi per la risoluzione di problemi di ottimizzazione e.g. linear programming (LP), mixed-integer linear programming (MILP), quadratic programming (QP) e mixed-integer quadratically constrained programming (MIQCP)
 - delle **API**, disponibili per l'interazione con il solver e la costruzione di problemi di ottimizzazione attraverso delle primitive per esprimere la notazione matematica normalmente utilizzata per problemi di ottimizzazione. Le API sono disponibili per diversi linguaggi (C++, Java, Python, etc.).

Gurobi: licenze e documentazione

- **Gurobi 12.x** è disponibile su www.gurobi.com
 - Software commerciale a pagamento o **gratis per scopi accademici**
<https://www.gurobi.com/academia/academic-program-and-licenses/>
 - **free restricted license** Modelli Lineari: max 2000 variabili e 2000 vincoli;
Modelli Quadratici: max 200 variabili e 200 vincoli.
- **Documentazione**
 - Manuale ufficiale
<https://docs.gurobi.com/projects/optimizer/en/current/>
 - Esempi ufficiali
<https://docs.gurobi.com/projects/examples/en/current/overview/examplelist.html>
 - Gurobot
<https://www.gurobi.com/solutions/gurobot/>

Gurobi Optimizer e Python

- **API** disponibili per Python 3.x attraverso il package **gurobipy**
 - Installazione su Colab attraverso il comando
`! python -m pip install gurobipy`
 - Guida ufficiale installazione
<https://support.gurobi.com/hc/en-us/articles/360044290292-How-do-I-install-Gurobi-for-Python>
 - Documentazione ufficiale
<https://docs.gurobi.com/projects/optimizer/en/current/reference/python.html>
 - Repository ufficiale esempi di modelli
<https://github.com/Gurobi/modeling-examples>

Esempio

Un coltivatore ha a disposizione 11 ettari di terreno da coltivare a lattuga o a patate. Le risorse a sua disposizione, oltre al terreno, sono: 70 kg di semi di lattuga, 18 t di tuberi, 145 t di concime. Supponendo che il mercato sia in grado di assorbire tutta la produzione e che i prezzi siano stabili, la resa stimata per la coltivazione di lattuga è di 3000 €/ettaro e quella delle patate è di 5000 €/ettaro. L'assorbimento delle risorse per ogni tipo di coltivazione è di 7 kg di semi e 10 t di concime per ettaro di lattuga, e 3 t di tuberi e 20 di concime per le patate. Stabilire quanto terreno destinare a lattuga e quanto a patate in modo da massimizzare la resa economica e sfruttando al meglio le risorse disponibili.

[RISORSA] [ro_contadino_1.ipynb](#)

Modello matematico

■ Variabili decisionali:

x_L : quantità in ettari da destinare a lattuga

x_P : quantità in ettari da destinare a patate

■ Funzione obiettivo:

$$\max 3000 x_L + 5000 x_P$$

■ Sistema dei vincoli:

$$x_L + x_P \leq 11 \quad (\text{ettari disponibili})$$

$$7 x_L \leq 70 \quad (\text{semi disponibili})$$

$$3 x_P \leq 18 \quad (\text{tuberi disponibili})$$

$$10 x_L + 20 x_P \leq 145 \quad (\text{concime disponibile})$$

$$x_L \geq 0, x_P \geq 0 \quad (\text{dominio})$$

Esempio base «contadino» I/

Codice per la creazione del modello e per l'aggiunta delle variabili

```
# CREAZIONE MODELLO [1]
```

```
m = Model("contadino_base")
```

```
# AGGIUNTA VARIABILI [2]
```

```
xL = m.addVar(name="xL", vtype=GRB.CONTINUOUS, lb=0.0,  
             ub=GRB.INFINITY) # ettari_lattuga
```

```
xP = m.addVar(name="xP", vtype=GRB.CONTINUOUS, lb=0.0,  
             ub=GRB.INFINITY) # ettari_patate
```

Parametri (tutti opzionali) della funzione **addVar**

- **lb** – Lower bound della nuova variabile
- **ub** – Upper bound della nuova variabile
- **obj** – Coefficiente della nuova variabile
- **vtype** – Tipo della nuova variabile, i tipi disponibili sono:
GRB.CONTINUOUS, GRB.BINARY, GRB.INTEGER, GRB.SEMICONT,
GRB.SEMIINT
- **name** – nome della nuova variabile. Solo caratteri ASCII, spazi bianchi sconsigliati

Esempio base «contadino» II/

Codice per l'aggiunta dei vincoli

AGGIUNTA VINCOLI [3]

```
m.addConstr(xL + xP <= 11, name="limite_ettari")
m.addConstr(7 * xL <= 70, name="limite_semi")
m.addConstr(3 * xP <= 18, name="limite_tuberi")
m.addConstr(10 * xL + 20 * xP <= 145, name="limite_concime")
```

Parametri della funzione **addConstr**

- **constr** – nuovo vincolo aggiunto (oggetto di tipo TempConstr [4])
NB: Un vincolo può avere solo un singolo operatore di confronto, infatti e.g.
 $1 \leq xL \leq 2$ è un vincolo non valido
- **name** (opzionale) – nome della nuovo vincolo. Solo calatteri ASCII, spazi bianchi sconsigliati

[1] <https://docs.gurobi.com/projects/optimizer/en/current/reference/python/model.html>

[2] <https://docs.gurobi.com/projects/optimizer/en/current/reference/python/model.html#Model.addVar>

[3] <https://docs.gurobi.com/projects/optimizer/en/current/reference/python/model.html#Model.addConstr>

[4] <https://docs.gurobi.com/projects/optimizer/en/current/reference/python/tempconstr.html#TempConstr>

Esempio base «contadino» II/

Codice per la definizione della funzione obiettivo e per l'esecuzione dell'ottimizzazione

DEFINIZIONE FUNZIONE OBIETTIVO [5] E OTTIMIZZAZIONE

```
m.setObjective(3000 * xL + 5000 * xP, GRB.MAXIMIZE)
m.optimize()
```

Parametri della funzione `setObjective`

- **expr** – Espressione matematica, può essere un oggetto di due tipi: **LinExpr** [6] o **QuadExpr** [7].
- **sense** (opzionale) – direzione dell'ottimizzazione, **GRB.MINIMIZE** (parametro di default) per minimizzare, **GRB.MAXIMIZE** per massimizzare

DEFINIZIONE FUNZIONE OBIETTIVO (ALTERNATIVA ED EQUIVALENTE)

```
expr = LinExpr(3000 * xL + 5000 * xP)
m.setObjective(expr, GRB.MAXIMIZE)
```

[5] <https://docs.gurobi.com/projects/optimizer/en/current/reference/python/model.html#Model.setObjective>

[6] <https://docs.gurobi.com/projects/optimizer/en/current/reference/python/linexpr.html#LinExpr>

[7] <https://docs.gurobi.com/projects/optimizer/en/current/reference/python/quadexpr.html#QuadExpr>

Esempio base «Contadino» [risorse]

Esercizio

- Rifare l'esercizio utilizzando variabili Python al posto dei valori utilizzati come coefficienti delle variabili decisionali e come bound nei vincoli

```
m.addConstr(7 * xL <= 70, name="limite_semi")
```

[RISORSA] ro_contadino_2.ipynb

Esercizio

Per l'assemblaggio di telecomandi, si hanno a disposizione 10 moduli display, 18 moduli di logica di controllo, 12 moduli di trasmissione, 21 tastierini, 9 moduli di navigazione e 10 moduli led. I telecomandi sono di due tipi. Il tipo A richiede un display, un modulo di navigazione, 2 tastierini, 2 moduli di logica, un modulo di trasmissione e un led. Il tipo B richiede 2 display, 3 tastierini, 2 moduli di logica e 3 moduli di trasmissione. Considerando che il tipo A permette un guadagno netto di 3 euro e il tipo B di 6 euro, determinare la produzione che massimizza il guadagno.

Modello PLI

Siano x_A e x_B le quantità di telecomandi di tipo A e B

$$\max 3 x_A + 6 x_B \quad (\text{guadagno complessivo})$$

s.t.

$$\begin{array}{rcll} x_A + 2 x_B & \leq & 10 & (\text{display}) \\ x_A & \leq & 9 & (\text{navigazione}) \\ 2 x_A + 3 x_B & \leq & 21 & (\text{tastierini}) \\ 2 x_A + 2 x_B & \leq & 18 & (\text{logica}) \\ x_A + 3 x_B & \leq & 12 & (\text{trasmissione}) \\ x_A & \leq & 10 & (\text{led}) \end{array}$$

$$x_A, x_B \in \mathbb{Z}_+$$

Esercizio.

Risolvere il problema dei telecomandi con utilizzando Python e Gurobi.

Per l'assemblaggio di telecomandi, si hanno a disposizione 10 moduli display, 18 moduli di logica di controllo, 12 trasmettitori, 21 tastierini, 9 moduli di navigazione e 10 led. I telecomandi sono di due tipi. Il tipo A richiede un display, un modulo di navigazione, 2 tastierini, 2 moduli di logica, un trasmettitore e un led. Il tipo B richiede 2 display, 3 tastierini, 2 moduli di logica e 3 trasmettitori. Considerando che il tipo A permette un guadagno netto di 3 euro e il tipo B di 8 euro, determinare la produzione che massimizza il guadagno.

[Luigi De Giovanni, Laura Brentegani, Modelli di Programmazione Lineare]

[Risorsa] [ro_telecomandi.ipynb](#)

[Risorsa] [ro_telecomandi_2.ipynb](#)

Implementazione di modelli «generali»

- I modelli precedenti includono i «dati» del problema:
 - Se cambiano i dati (coefficienti delle variabili, valori numerici utilizzati nei vincoli, etc.)
 - ⇒ Poca leggibilità
 - ⇒ Difficile riportare modifiche del modello
- Separare modello e dati
 - **File di codice** con il modello «generale» e la dichiarazione dei parametri del problema
 - **File (e.g. csv, xml, json, etc)** con i dati attribuiti ai parametri
- Per uno stesso file di modello possiamo utilizzare diversi file dati (ad esempio «contadino» e «telecomandi»)

Modello generale: esempio (mix ottimo di produzione)

I insieme dei beni che possono essere prodotti;

J insieme delle risorse disponibili;

P_i profitto (unitario) per il bene $i \in I$;

Q_j quantità disponibile della risorsa $j \in J$;

A_{ij} quantità di risorsa j necessaria per la produzione di un'unità del bene i .

$$\begin{aligned} \max \quad & \sum_{i \in I} P_i x_i \\ \text{s.t.} \quad & \sum_{i \in I} A_{ij} x_i \leq Q_j \quad \forall j \in J \\ & x_i \in \mathbb{R}_+ [\mathbb{Z}_+ \mid \{0, 1\}] \quad \forall i \in I \end{aligned}$$

Esercizio.

Risolvere il problema dei telecomandi con Python e Gurobi, utilizzando il modello generale.

Per l'assemblaggio di telecomandi, si hanno a disposizione 10 moduli display, 18 moduli di logica di controllo, 12 trasmettitori, 21 tastierini, 9 moduli di navigazione e 10 led. I telecomandi sono di due tipi. Il tipo A richiede un display, un modulo di navigazione, 2 tastierini, 2 moduli di logica, un trasmettitore e un led. Il tipo B richiede 2 display, 3 tastierini, 2 moduli di logica e 3 trasmettitori. Considerando che il tipo A permette un guadagno netto di 3 euro e il tipo B di 8 euro, determinare la produzione che massimizza il guadagno.

[Luigi De Giovanni, Laura Brentegani, Modelli di Programmazione Lineare]

[Risorsa] [ro_telecomandi_modello.ipynb](#)

[Risorsa] [telecomandi.csv](#)

Modello generale: dieta

I insieme delle risorse disponibili;

J insieme delle domande da coprire;

C_i costo (unitario) per l'utilizzo della risorsa $i \in I$;

D_j ammontare della domanda di $j \in J$;

A_{ij} capacità (unitaria) della risorsa i di soddisfare la domanda j .

$$\begin{array}{ll} \min & \sum_{i \in I} C_i x_i \\ \text{s.t.} & \\ & \sum_{i \in I} A_{ij} x_i \geq D_j \quad \forall j \in J \\ & x_i \in \mathbb{R}_+ [\mathbb{Z}_+ \mid \{0, 1\}] \quad \forall i \in I \end{array}$$

Esercizio

Un dietologo deve preparare una dieta che garantisca un apporto giornaliero di proteine, ferro e calcio di almeno 20 mg, 30 mg e 10 mg, rispettivamente. Il dietologo è orientato su cibi a base di verdura (5 mg/kg di proteine, 6 mg/Kg di ferro e 5 mg/Kg di calcio, al costo di 4 €/Kg), carne (15 mg/kg di proteine, 10 mg/Kg di ferro e 3 mg/Kg di calcio, al costo di 10 €/Kg) e frutta (4 mg/kg di proteine, 5 mg/Kg di ferro e 12 mg/Kg di calcio, al costo di 7 €/Kg). Determinare la dieta di costo minimo.

Siano x_1 , x_2 e x_3 le quantità di cibi a base di verdura, carne e frutta, rispettivamente

$$\min \quad 4x_1 + 10x_2 + 7x_3 \quad (\text{costo giornaliero dieta})$$

s.t.

$$5x_1 + 15x_2 + 4x_3 \geq 20 \quad (\text{proteine})$$

$$6x_1 + 10x_2 + 5x_3 \geq 30 \quad (\text{ferro})$$

$$5x_1 + 3x_2 + 12x_3 \geq 10 \quad (\text{calcio})$$

$$x_i \in \mathbb{R}_+, \quad \forall i \in \{1, 2, 3\}$$

Esercizi

- Formulare il problema con Python e Gurobi definendo i dati e gli insiemi necessari direttamente nel file
[Risorsa] ro_dieta_1.ipynb
- Risolvere il problema con Python e Gurobi leggendo i dati dal file csv
[Risorsa] ro_dieta_1_modello.ipynb
[Risorsa] dieta.csv

Esercizio

Il dietologo vuole inserire almeno 3 kg di alimenti a base di pesce azzurro (10 mg/kg di proteine, 15 mg/kg di ferro e 2 mg/kg di calcio, al costo di 3 euro/kg) nella dieta.

1. Modificare il file csv **dieta.csv** per aggiungere i nuovi dati del problema. Salvare il nuovo file come **dieta2.csv**.
2. Partendo dal file **ro_dieta_1_modello.ipynb** incorporare nel modello le nuove caratteristiche del problema. Risolvere il problema con Python e Gurobi leggendo i dati dal file **dieta2.csv**.

[Risorsa] ro_dieta_2_modello.ipynb

[Risorsa] dieta2.csv

Esercizio.

Un'azienda pubblicitaria deve svolgere un'indagine di mercato per lanciare un nuovo prodotto. Si deve contattare telefonicamente un campione significativo di persone: almeno 150 donne sposate, almeno 110 donne non sposate, almeno 120 uomini sposati e almeno 100 uomini non sposati. Le telefonate possono essere effettuate al mattino (al costo operativo di 1.1 euro) o alla sera (al costo di 1.6 euro). Le percentuali di persone mediamente raggiunte sono riportate in tabella.

	Mattino	Sera
Donne sposate	30%	30%
Donne non sposate	10%	20%
Uomini sposati	10%	30%
Uomini non sposati	10%	15%
Nessuno	40%	5%

Si noti come le telefonate serali sono più costose, ma permettono di raggiungere un maggior numero di persone: solo il 5% va a vuoto. Si vuole minimizzare il costo complessivo delle telefonate da effettuare (mattina/sera) in modo da raggiungere un campione significativo di persone

Modello PLI

- Siano x_1 e x_2 il numero di telefonate da fare al mattino e alla sera, rispettivamente

$\min$ $1.1 x_1 + 1.6 x_2$ (costo totale telefonate)

$s.t.$

$0.3x_1 + 0.3x_2 \geq 150$ (donne sposate)

$0.1x_1 + 0.2x_2 \geq 110$ (donne non sposate)

$0.1x_1 + 0.3x_2 \geq 120$ (uomini sposati)

$0.1x_1 + 0.15x_2 \geq 100$ (uomini non sposati)

$x_i \in \mathbb{Z}_+, \forall i \in \{1, 2\}$

ESERCIZI - Risolvere il problema con Python Gurobi.

A) implementare il modello, definire i dati direttamente nel file ipynb.

[Risorsa] ro_indagine.ipynb

B) creare il file csv relativo ai dati del problema, adattare il modello generale di dieta al nuovo problema e risolverlo leggendo i dati del file csv creato.

[Risorsa] ro_indagine_modello.ipynb

[Risorsa] indagine.csv

Esempio: Localizzazione di servizi

Una città è divisa in sei quartieri, dove si vogliono attivare dei centri unificati di prenotazione (CUP) per servizi sanitari. In ciascun quartiere è stata individuata una possibile località di apertura. Le distanze medie in minuti da ciascun quartiere a ciascuna delle possibili località è indicata in tabella. Si desidera che nessun utente abbia un tempo medio di spostamento **superiore a 15 minuti** per arrivare al CUP più vicino e si vuole minimizzare il numero di CUP attivati.

	Loc. 1	Loc. 2	Loc 3	Loc. 4	Loc. 5	Loc. 6
Q.re 1	5	10	20	30	30	20
Q.re 2	10	5	25	35	20	10
Q.re 3	20	25	5	15	30	20
Q.re 4	30	35	15	5	15	25
Q.re 5	30	20	30	15	5	14
Q.re 6	20	10	20	25	14	5

Modello PLI

Sia $x_i = 1$, se viene aperto il CUP nel quartiere i , 0 altrimenti

$$\min \quad x_1 + x_2 + x_3 + x_4 + x_5 + x_6$$

s.t.

$$x_1 + x_2 \geq 1 \text{ (esigenze q.re 1)}$$

$$x_1 + x_2 + x_6 \geq 1 \text{ (esigenze q.re 2)}$$

$$x_3 + x_4 \geq 1 \text{ (esigenze q.re 3)}$$

$$x_3 + x_4 + x_5 \geq 1 \text{ (esigenze q.re 4)}$$

$$x_4 + x_5 + x_6 \geq 1 \text{ (esigenze q.re 5)}$$

$$x_2 + x_5 + x_6 \geq 1 \text{ (esigenze q.re 6)}$$

$$x_i \in \{0,1\}$$

[Risorsa] ro_cup.ipynb

Esempio: Localizzazione di servizi

Una città è divisa in sei quartieri, dove si vogliono attivare dei centri unificati di prenotazione (CUP) per servizi sanitari. In ciascun quartiere è stata individuata una possibile località di apertura. Le distanze medie in minuti da ciascun quartiere a ciascuna delle possibili località è indicata in tabella. Si desidera che nessun utente abbia un tempo medio di spostamento **superiore a 15 minuti** per arrivare al CUP più vicino e si vuole minimizzare il numero di CUP attivati.

- **Variante 1:** alcuni quartieri centrali popolosi (quartiere 1 e quartiere 2) devono essere coperti da almeno due CUP.

[Risorsa] [ro_cup_quartieri_centrali.ipynb](#)

- **Variante 2:** data la prossimità delle località 4 e 5, al massimo un CUP può essere aperto nelle due località.

[Risorsa] [ro_cup_localita_adiacenti.ipynb](#)

Modello PLI generale

- Per poter utilizzare il modello generale («dieta») dobbiamo eseguire del **data preprocessing**.
- Ovvero, estrarre dai dati forniti (tabella) dei dati utilizzabili dal modello

	Loc. 1	Loc. 2	Loc 3	Loc. 4	Loc. 5	Loc. 6
Q.re 1	5	10	20	30	30	20
Q.re 2	10	5	25	35	20	10
Q.re 3	20	25	5	15	30	20
Q.re 4	30	35	15	5	15	25
Q.re 5	30	20	30	15	5	14
Q.re 6	20	10	20	25	14	5



Modello PLI generale

- Per poter utilizzare il modello generale («dieta») dobbiamo eseguire del **data preprocessing**.
- Ovvero, estrarre dai dati forniti (tabella) dei dati utilizzabili dal modello

	Loc. 1	Loc. 2	Loc 3	Loc. 4	Loc. 5	Loc. 6
Q.re 1	5	10	20	30	30	20
Q.re 2	10	5	25	35	20	10
Q.re 3	20	25	5	15	30	20
Q.re 4	30	35	15	5	15	25
Q.re 5	30	20	30	15	5	14
Q.re 6	20	10	20	25	14	5

«...Si desidera che nessun utente abbia
un tempo medio di spostamento
superiore a 15 minuti per arrivare al
CUP più vicino...»



Modello PLI generale

- Per poter utilizzare il modello generale («dieta») dobbiamo eseguire del **data preprocessing**.
- Ovvero, estrarre dai dati forniti (tabella) dei dati utilizzabili dal modello

	Loc. 1	Loc. 2	Loc 3	Loc. 4	Loc. 5	Loc. 6
Q.re 1	5	10	20	30	30	20
Q.re 2	10	5	25	35	20	10
Q.re 3	20	25	5	15	30	20
Q.re 4	30	35	15	5	15	25
Q.re 5	30	20	30	15	5	14
Q.re 6	20	10	20	25	14	5

«...Si desidera che nessun utente
abbia un tempo medio di
spostamento **superiore a 15 minuti**
per arrivare al CUP più vicino...»

	Loc. 1	Loc. 2	Loc 3	Loc. 4	Loc. 5	Loc. 6
Q.re 1	5	10	20	30	30	20
Q.re 2	10	5	25	35	20	10
Q.re 3	20	25	5	15	30	20
Q.re 4	30	35	15	5	15	25
Q.re 5	30	20	30	15	5	14
Q.re 6	20	10	20	25	14	5

Modello PLI generale

- Per poter utilizzare il modello generale («dieta») dobbiamo eseguire del **data preprocessing**.
- Ovvero, estrarre dai dati forniti (tabella) dei dati utilizzabili dal modello

	Loc. 1	Loc. 2	Loc 3	Loc. 4	Loc. 5	Loc. 6
Q.re 1	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
Q.re 2	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE
Q.re 3	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
Q.re 4	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE
Q.re 5	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE
Q.re 6	FALSE	TRUE	FALSE	FALSE	TRUE	TRUE

Esercizio

Risolvere il problema della localizzazione di servizi con Gurobi utilizzando il modello dieta generale ed il file csv **cup.csv** .

[Risorsa] [ro_cup_modello.ipynb](#)

NB: ricordarsi di installare **gurobipy**

```
! python -m pip install gurobipy
```

Modello generale: trasporti

I insieme dei centri di offerta; O_i ammontare dell'offerta in $i \in I$;

J insieme dei centri di domanda; D_j ammontare della domanda in $j \in J$.

C_{ij} costo (unitario) per il trasporto da $i \in I$ a $j \in J$;

$$\min \sum_{i \in I} \sum_{j \in J} C_{ij} x_{ij}$$

s.t.

$$\sum_{j \in J} x_{ij} \leq O_i \quad \forall i \in I$$

$$\sum_{i \in I} x_{ij} \geq D_j \quad \forall j \in J$$

$$x_{ij} \in \mathbb{R}_+ [\mathbb{Z}_+ \mid \{0, 1\}] \quad \forall i \in I, j \in J$$

Modello PLI

- Sia x_{ij} il numero di frigoriferi prodotti nello stabilimento i e smistati nel magazzino j

$$\begin{aligned} \min \quad & 6x_{A1} + 8x_{A2} + 3x_{A3} + 4x_{A4} + \\ & + 2x_{B1} + 3x_{B2} + 1x_{B3} + 3x_{B4} + \\ & + 2x_{C1} + 4x_{C2} + 6x_{C3} + 5x_{C4} \end{aligned}$$

s.t.

$$\begin{aligned} x_{A1} + x_{A2} + x_{A3} + x_{A4} &\leq 50 && \text{(capacità produttiva stabilimento A)} \\ x_{B1} + x_{B2} + x_{B3} + x_{B4} &\leq 70 && \text{(capacità produttiva stabilimento B)} \\ x_{C1} + x_{C2} + x_{C3} + x_{C4} &\leq 20 && \text{(capacità produttiva stabilimento C)} \end{aligned}$$

$$\begin{aligned} x_{A1} + x_{B1} + x_{C1} &\geq 10 && \text{(domanda magazzino 1)} \\ x_{A2} + x_{B2} + x_{C2} &\geq 60 && \text{(domanda magazzino 2)} \\ x_{A3} + x_{B3} + x_{C3} &\geq 30 && \text{(domanda magazzino 3)} \\ x_{A4} + x_{B4} + x_{C4} &\geq 40 && \text{(domanda magazzino 4)} \end{aligned}$$

$$x_{ij} \in \mathbb{Z}_+ \quad \forall i \in \{A, B, C\}, j \in \{1, 2, 3, 4\}$$

Esercizio. Trasporto di frigoriferi

Una ditta di produzione di elettrodomestici produce dei frigoriferi in tre stabilimenti e li smista in quattro magazzini intermedi di vendita. La produzione settimanale nei tre stabilimenti A, B e C è rispettivamente di 50, 70 e 20 unità. La quantità richiesta dai 4 magazzini è rispettivamente di 10, 60, 30 e 40 unità. I costi per il trasporto di un frigorifero tra gli stabilimenti e i magazzini 1, 2, 3 e 4 sono i seguenti:

- dallo stabilimento A: 6, 8, 3, 4 euro;
- dallo stabilimento B: 2, 3, 1, 3 euro;
- dallo stabilimento C: 2, 4, 6, 5 euro.

[Risorsa] [ro_trasporto_frighi_modello.ipynb](#)

[Risorsa] [frighi.csv](#)

Gestione degli stati in Gurobi

- L'ottimizzazione diversi risultati: una soluzione ottima, una soluzione sub-ottimale; inoltre il modello può essere illimitato (unbounded) o infeasible.
- Questi casi sono definiti da Gurobi e possono essere gestiti, soprattutto in fase di analisi dei risultati.
- Una lista esaustiva dei possibili stati restituiti da Gurobi è consultabile sulla documentazione ufficiale [8]
- Un esempio della gestione degli stati in Gurobi è presente nel file **ro_modello_trasporto_frighi.ipynb**

[8] docs.gurobi.com/projects/optimizer/en/current/reference/numericcodes/statuscodes.html

Esercizio: distribuzione PC

Un'azienda assembla dei PC in tre diversi stabilimenti con diverso costo unitario di produzione. I PC sono venduti a cinque clienti bancari e si sopportano dei costi di trasporto (inclusi gli oneri di importazione) per spedire un PC da ciascuno stabilimento a ciascun cliente. Sono definite le richieste di PC di ogni cliente e la produzione di ciascuno stabilimento è limitata. Non sono ammessi eccessi di produzione. I dati sono riassunti nella tabella seguente.

Scrivere in AMPL un modello del problema e fornire la soluzione, in termini di costo complessivo di trasporto e di quantità trasportate tra stabilimenti e sedi bancarie.

Esercizio: distribuzione PC (dati)

Produzione			Costi di trasporto				
Unità	costo unit.	Capacità	Banca Intesa	Uni Credit	Anton Veneta	Credit Suisse	Banca Cina
Italia	220	10 000	5,5	7,5	6,9	8,0	10,3
Cina	180	20 000	15,0	14,3	13,0	16,4	5,0
Francia	200	10 000	6,0	7,8	6,3	6,8	11,0
Domanda			7 100	3 400	9 700	5 200	3 050

Esercizio: distribuzione PC (modello base)

Insiemi: S (stabilimenti) e B (banche)

Parametri: w_i (costi prod.), c_{ij} (costi trasp.), a_i (capacità prod.), b_j (richieste)

$$\min \sum_{i \in S, j \in B} (w_i + c_{ij}) x_{ij}$$

$$\text{s. t. } \sum_{j \in B} x_{ij} \leq a_i, \quad \forall i \in S$$

$$\sum_{i \in S} x_{ij} = b_j, \quad \forall j \in B \quad (\text{uguaglianza} \Rightarrow \text{no eccessi di produzione})$$

$$x_{ij} \in \mathbb{Z}_+ \quad \forall i \in S, j \in B$$

[Risorsa] [ro_pc.ipynb](#)
[Risorsa] [pc_trasporto.csv](#)
[Risorsa] [pc_produzione.csv](#)

Esercizio: distribuzione PC (modello esteso)

Insiemi: S (stabilimenti) e B (banche)

Parametri: w_i (costi prod.), c_{ij} (costi trasp.), a_i (capacità prod.), b_j (richieste),
 α (bilanciamento generale), β (bilanciamento singolo)

$$\min \sum_{i \in S, j \in B} (w_i + c_{ij}) x_{ij}$$

$$\text{s. t. } \sum_{j \in B} x_{ij} \leq a_i \quad \forall i \in S$$

$$\sum_{i \in S} x_{ij} = b_j \quad \forall j \in B \quad (\text{uguaglianza} \Rightarrow \text{no eccessi di produzione})$$

$$\sum_{j \in B} x_{\text{Italia } j} \geq \alpha \sum_{i \in S, j \in B} x_{ij}$$

$$\sum_{j \in B} x_{\text{Italia } j} \geq \beta \sum_{j \in B} x_{ij} \quad \forall i \in S \setminus \{\text{Italia}\}$$

$$x_{ij} \in \mathbb{Z}_+ \quad \forall i \in S, j \in B$$

[Risorsa] [ro_pc_esteso.ipynb](#)

[Risorsa] [pc_trasporto.csv](#)

[Risorsa] [pc_produzione.csv](#)

Esercizio: distribuzione PC (scenari)

Per bilanciare la produzione, si richiede che nello stabilimento italiano:

1. si assemblino almeno il 25% di tutti i PC;
2. considerare che l'azienda richiede che nello stabilimento italiano si assemblino **almeno il 30% dei PC (ipotesi 1)** [o il 40% dei PC (**ipotesi 2**)] prodotti in ciascuno degli altri stabilimenti;
3. produrre un elenco che permetta di individuare i casi in cui una banca riceve forniture da un solo paese;
4. visualizzare l'utilizzo delle capacità produttive per paese.
5. capire se conviene, nell'ipotesi 2, potenziare di 5000 unità la produzione in Cina, al costo di 4.000 euro?
6. tornare alla situazione senza bilanciamenti e studiare gli effetti della diminuzione (a intervalli di 6 euro) del costo di produzione in Italia (diminuzione massima di 40 euro), indicando in quali casi in Italia la produzione complessiva supera quella della Francia.

Dualità in Gurobi: un esempio

Un'industria produce due tipi di creme: fondente e gianduia. Per avere un kg di ciascuna crema sono necessari, tra gli altri, due ingredienti grezzi (zucchero e cacao) e la lavorazione su una macchina, come riportato in tabella:

	Fondente	Gianduia
Zucchero (kg)	3	2
Cacao (kg)	4	1
Lavorazione (ore)	2	1

Settimanalmente, si hanno a disposizione al più 1200 Kg di zucchero e al più 1000 Kg di gianduia, mentre la disponibilità massima settimanale di ore lavorative della macchina è pari a 700. Un kg di fondente è venduto a 24 Euro e un kg di gianduia è venduto a 14 Euro. Si vuole pianificare la produzione settimanale in modo da massimizzare il ricavo complessivo.

[Risorsa] [ro_creme_dualita.ipynb](#)

```
# RECUPERO VALORE VARIABILE DUALE ASSOCIATA AD UN VINCOLO PRIMALE [9]  
vincolo[j].Pi
```

[9] <https://docs.gurobi.com/projects/optimizer/en/current/reference/attributes/constraintlinear.html#pi>

Esercizio

PROBLEMA PRIMALE

$$\begin{aligned} \max \quad & 24x_1 + 14x_2 \\ & 3x_1 + 2x_2 \leq 1200 \\ & 4x_1 + x_2 \leq 1000 \\ & 2x_1 + x_2 \leq 700 \\ & x_1 \geq 0, x_2 \geq 0 \end{aligned}$$

PROBLEMA DUALE

$$\begin{aligned} \min \quad & 1200u_1 + 1000u_2 + 700u_3 \\ & 3u_1 + 4u_2 + 2u_3 \geq 24 \\ & 2u_1 + u_2 + u_3 \geq 14 \\ & u_1 \geq 0, u_2 \geq 0, u_3 \geq 0 \end{aligned}$$

Implementare il problema primale e il problema duale dell'esempio e:

1. verificare il valore ottimo delle variabili primali e duali
2. verificare il teorema della dualità forte
3. verificare le condizioni di complementarità primale-duale
4. vedere come cambiano i valori ottimi delle funzioni obiettivo primale e duale variando i termini noti dei vincoli primali, uno per volta
5. dire se esiste una relazione tra il valore ottimo delle variabili duali e le variazioni osservate nel valore ottimo della funzione obiettivo

[Risorsa] [ro_dualita_esercizio.ipynb](#).

Dualità in Gurobi: un esempio (analisi)

Per i punti 4 e 5, osserviamo che

- prima della variazione $z^*=c^T x^*$, $w^*=b^T u^*$, $z^*=w^*$
- **se la variazione Δb di b non è troppo elevata**, le variabili duali u^* sono ancora ottime, quindi $w_{\text{new}}^* = (b+\Delta b)^T u^* = w^* + \Delta b^T u^*$
- per la dualità forte, $z_{\text{new}}^* = w_{\text{new}}^* = w^* + \Delta b^T u^* = z^* + \Delta b^T u^*$
- pertanto u_i^* è la *variazione della funzione obiettivo primale se b_i varia di un'unità*
- u_i^* indica di quanto aumenta il ricavo se aumentiamo di un'unità la risorsa i , quindi *quanto siamo disposti a spendere per ottenere un'unità aggiuntiva di risorsa i* : u_i^* è il **prezzo ombra della risorsa i**

Esercizio: ovile (testo)

L'azienda Ovile produce due tipi di cibo per animali: granulare e in polvere. Le materie prime utilizzate per la produzione sono: avena, mais e melassa. Tali materie, ad eccezione della melassa, devono essere macinate prima della lavorazione. In seguito si mescolano le varie sostanze e si processa il composto (granulazione o polverizzazione) al fine di ottenere i due diversi tipi di prodotto. La percentuale di proteine, grassi e fibre contenute nelle materie prime e i requisiti nutrizionali (in %) che i prodotti devono soddisfare sono riportati nella seguente tabella.

Materie prime	Proteine	Grassi	Fibre
Avena	13.6	7.1	7
Mais	4.1	2.4	3.7
Melassa	5	0.3	25
Requisiti	≥ 9.5	≥ 2	≤ 6

Di seguito sono riportati la disponibilità delle materie prime e i costi unitari per il loro acquisto.

Materie prime	Disponibilità (kg)	Costo (Euro/kg)
Avena	11900	0.13
Mais	23500	0.17
Melassa	750	0.12

Infine, i costi di produzione (per un kg di materie prime) sono riportati nella seguente tabella.

Macina	Mescola	Granulazione	Polverizzazione
0.25	0.05	0.42	0.17

Tenendo conto che la domanda giornaliera (esatta) è di 9 tonnellate per il prodotto granulare e di 12 tonnellate per quello in polvere, determinare il piano produttivo che minimizza il costo totale.

Esercizio: ovile (*todo*)

Insiemi: I (MATERIE); J (CIBI); K (SOSTANZE); R (LAVORAZIONI).

Parametri: ...; $P_{rij} = 1$ se è richiesta la lavorazione r sulla materia i per il cibo j , 0 altrimenti; ...

Modello PL:

$$\begin{aligned} \min \quad & \sum_{i \in I} C_i \sum_{j \in J} x_{ij} + \sum_{r \in R} F_r \sum_{i \in I} \sum_{j \in J} P_{rij} x_{ij} \\ \text{s.t.} \quad & \sum_{i \in I} A_{ik} x_{ij} \geq B_k \sum_{i \in I} x_{ij} & \forall j \in J, k \in K : B_k > 0 \\ & \sum_{i \in I} A_{ik} x_{ij} \leq U_k \sum_{i \in I} x_{ij} & \forall j \in J, k \in K : U_k < 1 \\ & \sum_{j \in J} x_{ij} \leq Q_i & \forall i \in I, \\ & \sum_{i \in I} x_{ij} = D_j & \forall j \in J, \\ & x_{ij} \in \mathbb{R}_+ & \forall i \in I, j \in J \end{aligned}$$

[Risorsa] `ro_ovile_modello.ipynb`

[Risorsa] `ovile.csv`

Esercizio: ovile (modello specifico)

Variabili: x_{ij} è la quantità (in kg) di materia prima i (1=avena, 2=mais, 3=melassa) destinata al tipo di prodotto j (1=granulare, 2=polvere)

$$\min 0.13(x_{11} + x_{12}) + 0.17(x_{21} + x_{22}) + 0.12(x_{31} + x_{32}) \quad \text{costi avena, mais e melassa}$$

$+ 0.25(x_{11} + x_{12} + x_{21} + x_{22}) \quad \text{kg macinati}$

$$+ 0.05(x_{11} + x_{12} + x_{21} + x_{22} + x_{31} + x_{32}) \text{ kg mescolati}$$

$+ 0.42(x_{11} + x_{21} + x_{31}) \quad \text{kg in granuli}$

$+ 0.17(x_{12} + x_{22} + x_{32})$ kg in polvere

$$0.136x_{11} + 0.041x_{21} + 0.05x_{31} \geq 0.095(x_{11} + x_{21} + x_{31}) \quad \text{min 9.5\% proteine per granulare}$$

$$0.071x_{11} + 0.024x_{21} + 0.003x_{31} \geq 0.02(x_{11} + x_{21} + x_{31}) \quad \text{min 2\% grassi per granulare}$$

$$0.07x_{11} + 0.037x_{21} + 0.25x_{31} \leq 0.06(x_{11} + x_{21} + x_{31}) \quad \text{max 6\% fibre per granulare}$$

$$0.136x_{12} + 0.041x_{22} + 0.05x_{32} \geq 0.095(x_{12} + x_{22} + x_{32}) \quad \text{min 9.5\% proteine per polvere}$$

$$0.071x_{12} + 0.024x_{22} + 0.003x_{32} \geq 0.02(x_{12} + x_{22} + x_{32}) \quad \text{min 2\% grassi per polvere}$$

$$0.07x_{12} + 0.037x_{22} + 0.25x_{32} \leq 0.06(x_{12} + x_{22} + x_{32}) \quad \text{max 6\% fibre per polvere}$$

$$x_{11} + x_{12} \leq 11900 \quad \text{disponibilità avena}$$

$$x_{21} + x_{22} \leq 23500 \quad \text{disponibilità mais}$$

$$x_{31} + x_{32} \leq 750 \quad \text{disponibilità melassa}$$

$$x_{11} + x_{21} + x_{31} = 9000 \quad \text{domanda granulare}$$

$$x_{12} + x_{22} + x_{32} = 12000 \quad \text{domanda polvere}$$

$$x_{ij} \geq 0 \quad i = 1, \dots, 3 \quad j = 1, 2$$

Esercizio: fonderia (testo)

Un'acciaieria acquista rottame di quattro tipi differenti (T1, T2, T3, T4) per ottenere due leghe (L1, L2) con caratteristiche chimiche differenti. I quattro tipi di rottame hanno i seguenti contenuti in percentuale di Piombo, Zinco e Stagno, e il seguente prezzo unitario di acquisto (in migliaia di € a tonnellata).

	T1	T2	T3	T4
Piombo	40%	30%	25%	38%
Zinco	35%	40%	35%	32%
Stagno	25%	30%	40%	30%
prezzo	2.5	1.8	2	2.2

La lega L1 deve avere un contenuto non superiore al 30% di piombo, al 60% di zinco e al 42% di stagno.

La lega L2 deve avere un contenuto non superiore al 46% di piombo, al 38% di zinco e al 56% di stagno.

Definire le quantità di ciascun tipo di rottame da utilizzare in ciascuna delle leghe in modo da minimizzare il costo complessivo e soddisfare esattamente un ordine di 1500 tonnellate di lega L1 e 2000 tonnellate di lega L2.

Esercizio: fonderia (modello specifico)

$$\begin{aligned} \min & 2.5(x_{11} + x_{12}) + 1.8(x_{21} + x_{22}) + 2(x_{31} + x_{32}) + 2.2(x_{41} + x_{42}) \\ \text{s.t.} & x_{11} + x_{21} + x_{31} + x_{41} = 1500 \\ & x_{12} + x_{22} + x_{32} + x_{42} = 2000 \\ & 0.4x_{11} + 0.3x_{21} + 0.25x_{31} + 0.38x_{41} \leq 0.3(x_{11} + x_{21} + x_{31} + x_{41}) \\ & 0.35x_{11} + 0.4x_{21} + 0.35x_{31} + 0.32x_{41} \leq 0.6(x_{11} + x_{21} + x_{31} + x_{41}) \\ & 0.25x_{11} + 0.3x_{21} + 0.40x_{31} + 0.3x_{41} \leq 0.42(x_{11} + x_{21} + x_{31} + x_{41}) \\ & 0.4x_{12} + 0.3x_{22} + 0.25x_{32} + 0.38x_{42} \leq 0.46(x_{12} + x_{22} + x_{32} + x_{42}) \\ & 0.35x_{12} + 0.4x_{22} + 0.35x_{32} + 0.32x_{42} \leq 0.38(x_{12} + x_{22} + x_{32} + x_{42}) \\ & 0.25x_{12} + 0.3x_{22} + 0.40x_{32} + 0.3x_{42} \leq 0.56(x_{12} + x_{22} + x_{32} + x_{42}) \\ & x_{ij} \geq 0 \quad i = 1, \dots, 4 \quad j = 1, 2 \end{aligned}$$

Esercizio: fonderia (*todo*)

■ Modello generale (...)

Insiemi: I (ROTTAMI); J (LEGHE); K (METALLI).

Parametri: C_i (Prezzo rottame $i \in I$); R_j (Ordine lega $j \in J$); $A_{k,i}$ (contenuto metallo $k \in K$ in rottame $i \in I$); $U_{k,j}$ (contenuto massimo metallo $k \in K$ nella lega $j \in J$).

Variabili: x_{ij} (acquisti di rottame $i \in I$ usati per la lega $j \in J$)

Modello PL:

$$\begin{aligned} \min \quad & \sum_{i \in I} C_i \sum_{j \in J} x_{ij} \\ \text{s.t.} \quad & \sum_{i \in I} A_{ki} x_{ij} \leq U_{kj} \sum_{i \in I} x_{ij} \quad \forall j \in J, k \in K \\ & \sum_{i \in I} x_{ij} = R_j \quad \forall j \in J, \\ & x_{ij} \in \mathbb{R}_+ \quad \forall i \in I, j \in J \end{aligned}$$

[Risorsa] [ro_fonderia_modello.ipynb](#)

[Risorsa] [fonderia.csv](#)