UNIVERSITÀ
di **VERONA**
Dipartimento
di **INFORMATICA**

Ricerca Operativa

Modellizzare problemi di ottimizzazione

Parte 2

Laurea triennale in Matematica Applicata
Anno Accademico 2021/2022

Alice Raffaele – *alice.raffaele@univr.it*

Da modelli semplici a più generali

Assegnamento del posto in ufficio per la COVID – Problema 0 (un giorno, un ufficio)

Come stabilire l'assegnazione dei posti in un ufficio con un numero di scrivanie maggiore rispetto a quello di ricercatori e ricercatrici? A N persone è chiesto di esprimere la loro probabilità di essere in università (una priorità da 0 a 3, dove 0 significa "Non vengo"). Nell'autunno 2020, le normative in materia di salute e sicurezza relative alla COVID 19 imponevano che il numero massimo di persone autorizzate a condividere l'ufficio contemporaneamente fosse $C \leq N$.

Persona	Priorità
Alice	3
Andrea	1
Chiara	2
Elia	0
Fabio	1
Franco	0
Federico	2
Matteo	3
Michele	1
Rossana	1

$C = 6$

Come stabilire un assegnamento ottimale, tale da massimizzare le priorità soddisfatte?

Costruiamo il modello:

- **Variabili**: una variabile binaria x_i per ogni persona i, $i = 1, \ldots, N$, per indicare se a i venga assegnata una scrivania oppure no.
- **Vincoli**:
  - massimo numero di persone consentito in ufficio: $\sum_{i=1}^{N} x_i \leq C$.
- **Funzione obiettivo**: massimizzare la priorità totale

$$\max \quad \sum_{i=1}^{N} p_i x_i .$$

Assegnamento del posto in ufficio per la COVID – Problema 1a (un giorno, più uffici)

Si consideri la seguente estensione del Problema 0. Ci sono J uffici disponibili in cui assegnare le N persone. Ogni ufficio j ha una massima capienza di $C_j \leq N$.

Persona	Priorità
Alice	3
Andrea	1
Chiara	2
Elia	0
Fabio	1
Franco	0
Federico	2
Matteo	3
Michele	1
Rossana	1

$$J = 2, C_1 = 6, C_2 = 3.$$

Possiamo vedere il Problema 0 come un caso speciale del Problema 1a? E viceversa?

Costruiamo il modello:

- **Variabili**: una variabile binaria $x_{i,j}$ per ogni persona i e per ogni ufficio j, con $i = 1, \ldots, N$ e $j = 1, \ldots, J$, per indicare se a i venga assegnato un posto nell'ufficio j oppure no.
- **Vincoli**:
  - massimo numero di persone in ogni ufficio:
    $$\sum_{i=1}^{N} x_{i,j} \leq C_j, \quad \forall j = 1, \ldots, J;$$
  - ogni persona è al massimo assegnata in un solo ufficio:
    $$\sum_{j=1}^{J} x_{i,j} \leq 1, \quad \forall i = 1, \ldots, N;$$
- **Funzione obiettivo**: massimizzare la priorità totale

$$\max \quad \sum_{i=1}^{N} \sum_{j=1}^{J} p_i x_{i,j}$$

.

Assegnamento del posto in ufficio per la COVID – Problema 1b (più giorni, un ufficio

Si consideri un'altra possibile estensione del Problema 0. Si vuole decidere come assegnare i posti dell'ufficio disponibile ai ricercatori e alle ricercatrici non per un giorno solo, ma per D giorni (non per forza consecutivi). Le persone esprimono una loro priorità di avere un posto per ogni giorno.

Persona	Lun	Mar	Mer	Gio	Ven
Alice	3	3	3	3	3
Andrea	1	3	1	0	0
Chiara	2	1	3	0	0
Elia	0	0	2	0	0
Fabio	1	0	2	0	1
Franco	0	0	0	3	2
Federico	2	1	1	3	1
Matteo	3	3	3	3	3
Michele	1	0	3	2	0
Rossana	1	0	3	2	0

$C = 6, D = 5$

Possiamo vedere il Problema 0 come un caso speciale del Problema 1b? E viceversa? Il Problema 1a può essere considerato un caso speciale del Problema 1b? E viceversa?

Costruiamo il modello:

- **Variabili**: una variabile binaria $x_{i,d}$ per ogni persona i e per ogni giorno d, con $i = 1, \ldots, N$ e $d = 1, \ldots, D$, per indicare se a i è assegnato un posto in ufficio nel giorno d oppure no.
- **Vincoli**:
  - massimo numero di persone ogni giorno:
    $$\sum_{i=1}^{N} x_{i,d} \leq C, \quad \forall d = 1, \ldots, D.$$
- **Funzione obiettivo**: massimizzare la priorità totale in tutti i giorni
  $$\max \quad \sum_{i=1}^{N} \sum_{d=1}^{D} p_{i,d} x_{i,d}$$

.

Assegnamento del posto in ufficio per la COVID – Problema 2a (più giorni, più uffici)

Si considerino le estensioni Problema 1a e Problema 1b assieme.

Persona	Lun	Mar	Mer	Gio	Ven
Alice	3	3	3	3	3
Andrea	1	3	1	0	0
Chiara	2	1	3	0	0
Elia	0	0	2	0	0
Fabio	1	0	2	0	1
Franco	0	0	0	3	2
Federico	2	1	1	3	1
Matteo	3	3	3	3	3
Michele	1	0	3	2	0
Rossana	1	0	3	2	0

$$J = 2, C_1 = 6, C_2 = 3, D = 5$$

Possiamo vedere il Problema 0 come caso speciale del Problema 2a? E viceversa?

Costruiamo il modello:

- **Variabili**: una variabile binaria $x_{i,j,d}$ per ogni persona i, per ogni ufficio j, e per ogni giorno d, con $i = 1, \ldots, N$, $j = 1, \ldots, J$, e $d = 1, \ldots, D$, per indicare se a i è assegnato un posto nell'ufficio j nel giorno d oppure no.
- **Vincoli**:
  - massimo numero di persone in ogni ufficio ogni giorno: $\sum_{i=1}^{N} x_{i,j,d} \leq C, \quad \forall j = 1, \ldots, J; \forall d = 1, \ldots, D;$
  - ogni persona è al più assegnata con un posto in un ufficio ogni giorno: $\sum_{j=1}^{J} x_{i,j,d} \leq 1, \quad \forall i = 1, \ldots, N; \forall d = 1, \ldots, D;$
- **Funzione obiettivo**: massimizzare la priorità totale su tutti i giorni

$$\max \quad \sum_{i=1}^{N} \sum_{j=1}^{J} \sum_{d=1}^{D} p_{i,d} x_{i,j,d}$$

.

Assegnamento del posto in ufficio per la COVID – Problema 2b (più giorni, più uffici, stesso ufficio per ogni persona)

Si consideri la seguente variante del Problema 2a: tutte le persone mantengono lo stesso posto in un determinato ufficio tutti i giorni in cui viene loro assegnato un posto.

Persona	Lun	Mar	Mer	Gio	Ven
Alice	3	3	3	3	3
Andrea	1	3	1	0	0
Chiara	2	1	3	0	0
Elia	0	0	2	0	0
Fabio	1	0	2	0	1
Franco	0	0	0	3	2
Federico	2	1	1	3	1
Matteo	3	3	3	3	3
Michele	1	0	3	2	0
Rossana	1	0	3	2	0

$$J = 2, C_1 = 6, C_2 = 3, M = 5$$

Costruiamo il modello:

- **Variabili**:
  - una variabile binaria $x_{i,j,d}$ per ogni persona i, per ogni ufficio j, e per ogni giorno d, con $i = 1, \ldots, N$, $j = 1, \ldots, J$, e $d = 1, \ldots, D$, per indicare se a i venga assegnato un posto nell'ufficio j nel giorno d oppure no;
  - una variabile binaria $y_{i,j}$ per ogni persona i e per ogni ufficio j, per indicare se a i venga assegnato un posto nell'ufficio j oppure no.

- **Vincoli**:
  - massimo numero di persone in ogni ufficio ogni giorno:

$$\sum_{i=1}^{N} x_{i,j,d} \leq C_j, \quad \forall j = 1, \ldots, J; \forall d = 1, \ldots, D$$

  - legame tra le due diverse famiglie di variabili:

$$x_{i,j,d} \leq y_{i,j}, \quad \forall i = 1, \ldots, N; \forall j = 1, \ldots, J$$

  - al massimo assegnamento in un solo ufficio:

$$\sum_{j=1}^{J} y_{i,j} \leq 1, \quad \forall i = 1, \ldots, N$$

- **Funzione obiettivo**: massimizzare la priorità totale su tutti i giorni:

$$\max \quad \sum_{i=1}^{N} \sum_{j=1}^{J} \sum_{d=1}^{D} p_{i,d} x_{i,j,d}$$

.

Giochi

Sudoku

8								
		3	6					
	7			9		2		
	5				7			
				4	5	7		
			1				3	
		1					6	8
		8	5				1	
	9					4		

Come possiamo modellare questo gioco come un problema di Programmazione Lineare Intera?

Costruiamo il modello:

- **Variabili**: introduciamo una variabile binaria $x_{i,j,k} = 1$, per ogni cella (i, j) e per ogni possibile valore k, con $i = 1, \ldots, 9$, $j = 1, \ldots, 9$, e $k = 1, \ldots, 9$, per indicare se nella cella (i, j) c'è la cifra k oppure no.
- **Funzione obiettivo**: o $\rightarrow$ È un problema di *ammissibilità*, non ci interessa minimizzare o massimizzare alcunché.
- **Vincoli**:
  - Ogni cella deve essere riempita con una cifra:

$$\sum_{k=1}^{9} x_{i,j,k} = 1, \forall i \in \{1, \ldots, 9\}, \forall j \in \{1, \ldots, 9\}$$

  - Ogni cifra da 1 a 9 deve apparire in ogni riga:

$$\sum_{j=1}^{9} x_{i,j,k} = 1, \forall i \in \{1, \ldots, 9\}, \forall k \in \{1, \ldots, 9\}$$

  - Ogni cifra da 1 a 9 deve apparire in ogni colonna:

$$\sum_{i=1}^{9} x_{i,j,k} = 1, \forall j \in \{1, \ldots, 9\}, \forall k \in \{1, \ldots, 9\}$$

- **Vincoli**:
  - Ogni cifra da 1 a 9 deve apparire in ogni quadrato 3x3:

$$\sum_{j=(3p-2)}^{3p} \sum_{i=(3q-2)}^{3q} x_{i,j,k} = 1, \forall p \in \{1,2,3\}, q \in \{1,2,3\}, k \in \{1, \cdots, n\}$$

  - Numeri già presenti (parametri):

$$x_{i,j,k} = 1, \forall i,j,k \in DATA$$

Modello risultante:

$$\min \quad 0$$

$$\sum_{k=1}^{9} x_{i,j,k} = 1, \forall i \in \{1,\dots,9\}, \forall j \in \{1,\dots,9\}$$

$$\sum_{j=1}^{9} x_{i,j,k} = 1, \forall i \in \{1,\dots,9\}, \forall k \in \{1,\dots,9\}$$

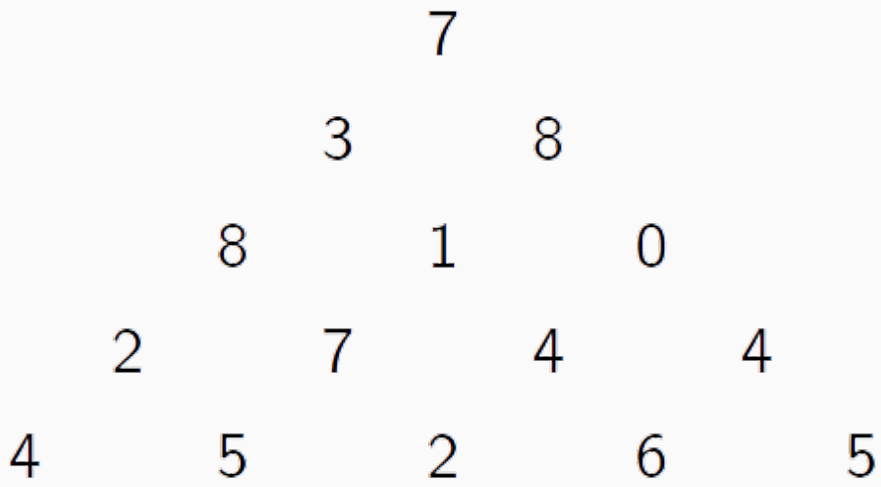
$$\sum_{i=1}^{9} x_{i,j,k} = 1, \forall j \in \{1,\dots,9\}, \forall k \in \{1,\dots,9\}$$

$$\sum_{j=(3p-2)}^{3p} \sum_{i=(3q-2)}^{3q} x_{i,j,k} = 1, \forall p \in \{1,2,3\}, q \in \{1,2,3\}, k \in \{1,\cdots,n\}$$

$$x_{i,j,k} = 1, \forall i,j,k \in DATA$$

$$x_{i,j,k} \in \{0,1\}, \quad \forall i \in \{1,\dots,9\}, \forall j \in \{1,\dots,9\}, \forall k \in \{1,\dots,9\}$$

Triangolo



Come modellare questo problema come un problema di Programmazione Lineare Intera, in modo tale da calcolare la somma più grande che si possa ottenere partendo dal vertice e raggiungendo la base? In ogni passo per andare alla riga sottostante, si può andare a sinistra oppure a destra, scegliendo un solo numero.

Costruiamo il modello:

- Possiamo vedere il triangolo come una matrice triangolare inferiore con N righe e colonne. Sia $v_{i,j}$ il valore presente nella cella (i, j).
- **Variabili**: introduciamo una variabile binaria $x_{i,j}$ per ogni cella (i, j), con $i = 1, \ldots, N$ e $j = 1, \ldots, N$, per indicare se aggiungiamo alla somma il numero in (i, j) oppure no.

- **Vincoli**:
  - solo un numero per ogni riga:

$$\sum_{j=1}^{N} x_{i,j} = 1, \quad \forall i = 1, \ldots, N$$

  - data una generica cella (i, j) nella riga i, possiamo essere arrivati dalla riga $i-1$ dalla cella direttamente sopra $(i-1, j)$ oppure da quella a sinistra di 1 $(i-1, j-1)$:

$$x_{i,j} \leq x_{i-1,j-1} + x_{i-1,j}, \quad \forall i = 2, \ldots, N, \forall j = 2, \ldots, N$$

  - nella prima colonna:

$$x_{i,1} \leq x_{i-1,1}, \quad \forall i = 2, \ldots, N$$

- **Funzione obiettivo**: massimizzare la somma dei numeri presi

$$\max \quad \sum_{i=1}^{N} \sum_{j=1}^{N} v_{i,j} x_{i,j}.$$

Modello risultante:

$$\max \quad \sum_{i=1}^{N}\sum_{j=1}^{N} v_{i,j}x_{i,j}$$

$$\sum_{j=1}^{N} x_{i,j} = 1, \quad \forall i = 1,\dots,N$$

$$x_{i,j} \leq x_{i-1,j-1} + x_{i-1,j}, \quad \forall i = 2,\dots,N, \forall j = 2,\dots,N$$

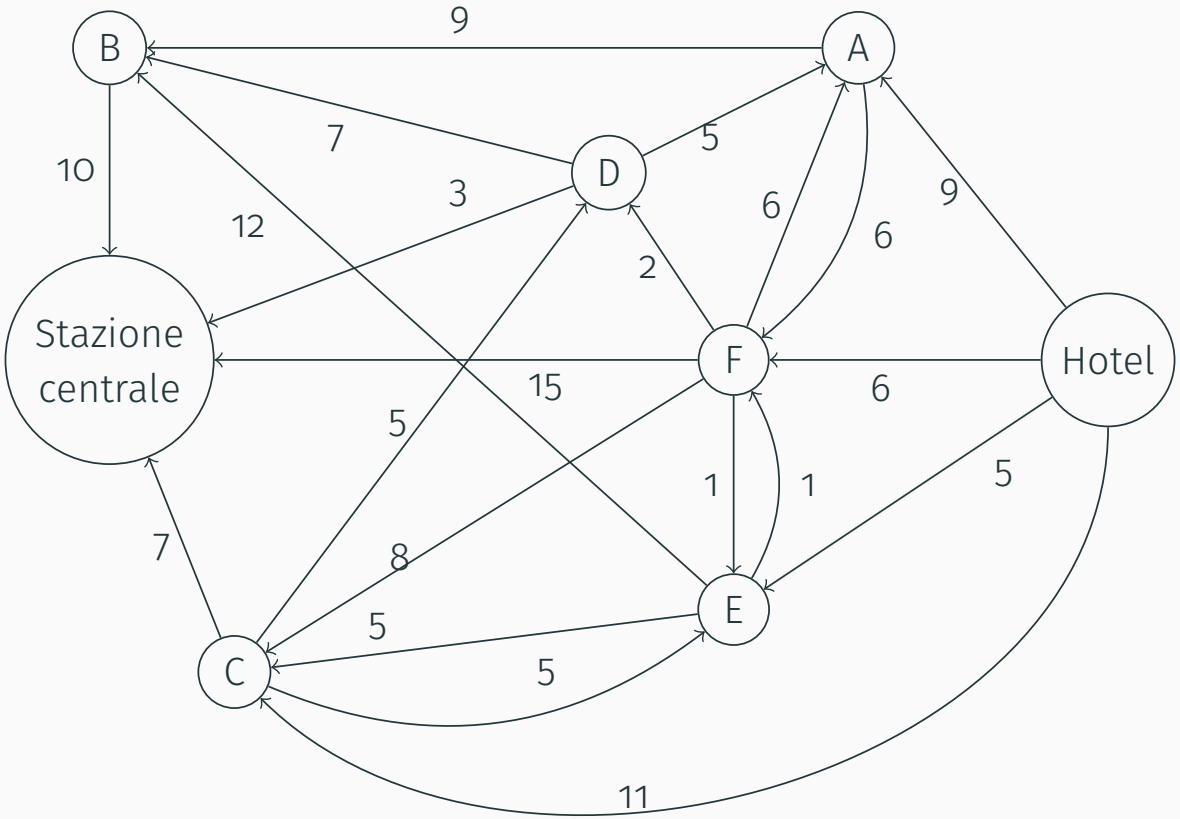
$$x_{i,1} \leq x_{i-1,1}, \quad \forall i = 2,\dots,N$$

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

Problemi su grafi

Cammino minimo – Per non perdere il treno

Avete trascorso qualche giorno di vacanza a Roma in hotel e ora è giunto il momento di tornare a casa. Data la seguente mappa stradale, dove i vertici rappresentano gli incroci, gli archi le strade, e i numeri sopra gli archi indicano i tempi di cammino (in minuti), quale sarebbe il percorso che vi permetterebbe di risparmiare più tempo possibile per raggiungere la stazione centrale partendo dall'hotel?



Costruiamo il modello:

- Sia $G = (V, A)$ il grafo diretto pesato rappresentante la mappa stradale, dove a ogni arco $(i, j) \in A$ è associato un tempo di cammino $t_{i,j}$.
- **Variabili**: si introduce una variabile binaria $x_{i,j}$ per ogni arco $(i, j) \in A$ per indicare se tale arco sia percorso o no.
- **Vincoli**:
  - vertice di partenza:

$$x_{H,A} + x_{H,C} + x_{H,E} + x_{H,F} = 1$$

in forma compatta:

$$\sum_{(H,j)\in A} x_{H,j} = 1$$

- **Vincoli**:
  - vertice di arrivo:

$$x_{B,S} + x_{C,S} + x_{D,S} + x_{F,S} = 1$$

    in forma compatta:

$$\sum_{(i,S)\in A} x_{i,S} = 1$$

  - vertici intermedi:

$$\sum_{(i,j)\in A} x_{i,j} = \sum_{(j,k)\in A} x_{j,k}, \quad \forall j \in V \setminus \{H, S\}$$

- **Funzione obiettivo**: minimizzare il tempo totale di percorrenza

$$\min \quad \sum_{(i,j)\in A} t_{i,j} x_{i,j}.$$

Modello risultante:

$$\min \quad \sum_{(i,j)\in A} t_{i,j} x_{i,j}$$

$$\sum_{(H,j)\in A} x_{H,j} = 1$$

$$\sum_{(i,j)\in A} x_{i,j} = \sum_{(j,k)\in A} x_{j,k}, \quad \forall j \in V \setminus \{H, S\}$$

$$\sum_{(i,S)\in A} x_{i,S} = 1$$

$$x_{i,j} \in \{0, 1\}, \quad \forall (i,j) \in A.$$

Implementazione con PuLP:

```
from pulp import *

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

# Set e parametri
V = ["A","B","C","D","E","F","H","S"]
A = {
     ("A","B"): 9, ("A","F"): 6, ("B","S"): 10, ("C","D"): 5, ("C","E"): 5, ("C","S"): 7, ("D","A"): 5,
     ("D","B"): 7, ("D","S"): 3, ("E","B"): 12, ("E","C"): 5, ("E","F"): 1, ("F","A"): 6, ("F","C"): 8,
     ("F","D"): 2, ("F","E"): 1, ("F","S"): 15, ("H","A"): 9, ("H","C"): 11, ("H","E"): 5, ("H","F"): 6}

source = "H"
destination = "S"

# Variabili
x = LpVariable.dicts("x", A.keys(), 0, 1, LpBinary)

# Funzione obiettivo
model += lpSum(x[i,j]*A[i,j] for i,j in A.keys())

# Vincoli
model += lpSum(x[i,j] for i,j in A.keys() if i == source) == 1
model += lpSum(x[i,j] for i,j in A.keys() if j == destination) == 1

for v in V:
    if v not in [source, destination]:
        model += lpSum(x[i,j] for i,j in A.keys() if j == v) == lpSum(x[i,j] for i,j in A.keys() if i == v)

# Chiamata al solver
model.solve()


# Stampa soluzione ottima se trovata
if LpStatus[model.status] == "Optimal":
    print("Soluzione ottima:")
    for v in model.variables():
        print("{} = {})".format(v.name, v.varValue))

    # Valore della funzione obiettivo
    print("\nCammino minimo = {}".format(round(value(model.objective),2)))

elif LpStatus[model.status] == "Infeasible":
    print("Istanza non ammissibile")
```

Massimo flusso su una rete

Una compagnia aerea deve quantificare i voli notturni da offrire per spostarsi da New York (NY) a San Francisco (SF). Questi voli devono per forza fare scalo in alcune città (identificate dalle lettere A, B, C e D). A causa degli spazi di parcheggi limitati nelle piste degli aeroporti, non possono viaggiare più di un certo numero di voli tra ogni coppia di città, come specificato nella tabella a lato. Nella tabella sono anche mostrate le rotte disponibili: per esempio, le prime due righe indicano come i voli in partenza da NY possano fare scalo in A o in B, mentre invece non è possibile raggiungere direttamente C da NY. Formulare un modello per determinare il numero massimo di voli che potrebbero viaggiare da NY a SF.

Rotta	Max numero di voli
NY – A	50
NY – B	40
A – C	30
A – D	25
B – C	20
B – D	15
C – SF	40
D – SF	35

Costruiamo il modello:

- Sia $G = (V, A)$ il grafo diretto associato all'istanza considerata, dove $V := \{NY, A, B, C, D, SF\}$ e ogni arco $(i, j) \in A$ è dotato di un peso corrispondente al massimo numero di voli tra la città i e la città j.
- **Variabili**:
  - si introduce una variabile $x_{i,j}$ intera e non negativa per indicare il numero di voli che verranno fatti tra la città i e la città j;
  - si introduce una variabile ϕ intera e non negativa per indicare il numero di voli tra NY e San Francisco.

- **Vincoli**:
  - massimo numero di voli per tratta:

$$x_{i,j} \leq c_{i,j}, \quad \forall (i,j) \in A$$

  - tutti i voli che partono da NY devono arrivare a San Francisco:

$$\sum_{(NY,j)\in A} x_{NY,j} = \phi$$

$$\sum_{(i,SF)\in A} x_{i,SF} = \phi$$

  - in ogni città intermedia, tutti i voli che arrivano devono anche ripartire:

$$\sum_{(i,j)\in A} x_{i,j} = \sum_{(j,k)\in A} x_{j,k}, \quad \forall j \in V \setminus \{NY, SF\}$$

- **Funzione obiettivo**: massimizzare il numero di voli da NY a San Francisco

$$\max \quad \phi.$$

Modello risultante:

$$
\begin{aligned}
\max \quad & \phi \\
& \sum_{(NY,j)\in A} x_{NY,j} = \phi \\
& \sum_{(i,SF)\in A} x_{i,SF} = \phi \\
& \sum_{(i,j)\in A} x_{i,j} = \sum_{(j,k)\in A} x_{j,k}, \quad \forall j \in V \setminus \{NY, SF\} \\
& x_{i,j} \leq c_{i,j}, \quad \forall (i,j) \in A \\
& x_{i,j} \in \mathbb{N}, \quad \forall (i,j) \in A \\
& \phi \in \mathbb{N}.
\end{aligned}
$$

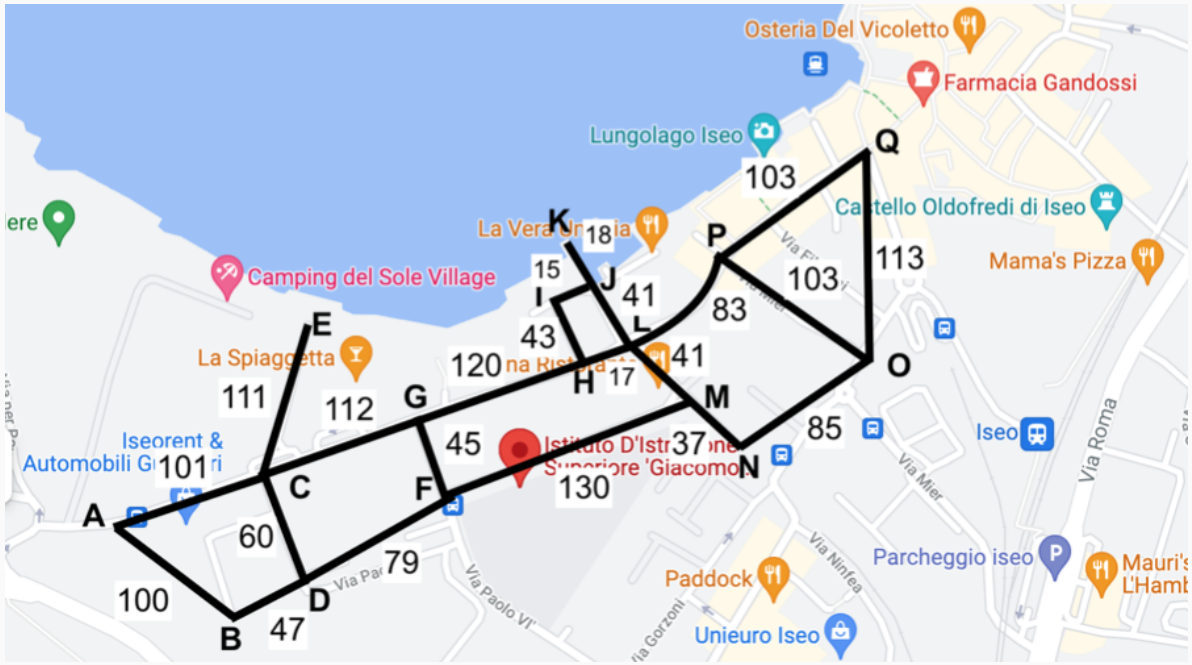
Implementazione con PuLP:

```
from pulp import *

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

# Set e parametri
V = ["NY","A","B","C","D","SF"]
A = {
     ("NY","A"): 50, ("NY","B"): 40, ("A","C"): 30, ("A","D"): 25,
     ("B","C"): 20, ("B","D"): 15, ("C","SF"): 40, ("D","SF"): 35}

# Variabili
x = LpVariable.dicts("x", A.keys(), 0, None, LpInteger)
phi = LpVariable("phi", 0, None, LpInteger)

# Funzione obiettivo
model += phi
```

```
# Vincoli
for i,j in A.keys():
    model += x[i,j] <= A[i,j]

model += lpSum(x[i,j] for i,j in A.keys() if i == "NY") == phi
model += lpSum(x[i,j] for i,j in A.keys() if j == "SF") == phi

for v in V:
    if v not in ["NY","SF"]:
        model += lpSum(x[i,j] for i,j in A.keys() if j == v) == lpSum(x[i,j] for i,j in A.keys() if i == v

# Chiamata al solver
model.solve()

# Stampa soluzione ottima se trovata
if LpStatus[model.status] == "Optimal":
    print("Soluzione ottima:")
    for v in model.variables():
        print(v.name, " = ", v.varValue)

    # Valore della funzione obiettivo
    print("\nMassimo flusso = {}".format(round(value(model.objective),2)))

elif LpStatus[model.status] == "Infeasible":
    print("Istanza non ammissibile")
```

Minimo albero di supporto – Un problema di... connessione

A Iseo, un comune in provincia di Brescia, buona parte delle abitazioni e degli edifici sono raggiunti con la fibra ottica. Tuttavia, nella zona rappresentata nella mappa, ci sono alcuni edifici pubblici del Comune (indicati con le lettere dalla A alla Q) che non sono ancora stati coperti. Sapendo che la ditta incaricata dal Comune di Iseo chiede, per ogni metro di fibra ottica, 5 euro, in quali strade andrà posata la fibra ottica in modo da minimizzare la spesa totale sostenuta dal Comune?



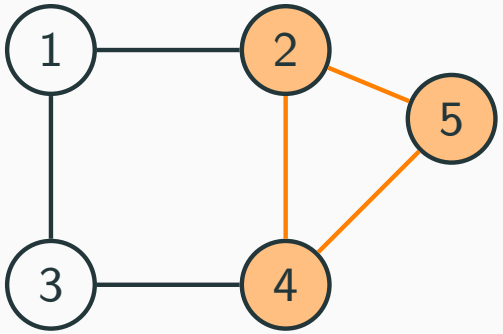
Costruiamo il modello:

- Sia $G = (V, E)$ un grafo indiretto pesato, ossia ogni lato $e \in E$ è associato a un costo $c_e \in \mathbb{R}^+$. Sia $n := |V|$.
- **Variabili**: si introduce una variabile binaria x_e per ogni lato $e \in E$, che indica se il lato e viene considerato o no per formare l'albero di supporto di costo minimo.

- **Vincoli**:
  - un albero di supporto di G ha $n-1$ lati:

$$\sum_{e \in E} x_e = n - 1;$$

  - l'albero è aciclico $\rightarrow$ per ogni $S \subset V$, sia $E(S) := \{(i,j) \in E \mid i,j \in S\}$.



$S = \{2,4,5\}$, $E(S) = \{(2,4),(2,5),(4,5)\}$.

$\downarrow$

Per evitare che si selezionino lati tali da formare un ciclo:

$$\sum_{e \in E(S)} x_e \leq |S| - 1, \quad \forall S \subset V, S \neq \emptyset;$$

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

$$\min \quad \sum_{e \in E} c_e x_e.$$

Modello risultante:

$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e \\
& \sum_{e \in E} x_e = n - 1 \\
& \sum_{e \in E(S)} x_e \leq |S| - 1, \quad \forall S \subset V, S \neq \emptyset \\
& x_e \in \{0, 1\}, \quad \forall e \in E.
\end{aligned}
$$

Implementazione con PuLP:

```
from pulp import *
import itertools

def get_combinations(xs):
    returner = []
    for L in range(2, len(xs)+1):
        for subset in itertools.combinations(xs, L):
            returner.append(subset)
    return returner

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

# Set e parametri
V = ["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q"]
E = {
     ("A","B"): 100, ("A","C"): 101, ("B","D"): 47, ("C","D"): 60, ("C","E"): 111,
     ("C","G"): 112, ("D","F"): 79, ("F","G"): 45, ("F","M"): 130, ("G","H"): 120,
     ("H","I"): 43, ("H","L"): 17, ("I","J"): 15, ("J","K"): 18, ("J","L"): 41,
     ("L","M"): 41, ("L","P"): 41, ("M","N"): 37, ("N","O"): 85, ("O","P"): 103,
     ("O","Q"): 113, ("P","Q"): 103}

# Variabili
x = LpVariable.dicts("x", E.keys(), 0, 1, LpBinary)
```

```
# Funzione obiettivo
model += lpSum(x[i]*E[i] for i in E.keys())

# Vincoli
model += lpSum(x[i] for i in E.keys()) == len(V)-1

combinations = get_combinations(V)
for comb in combinations:
    lati_S = []
    for v in comb:
        lati_v = [i for i in E.keys() if i[0] == v]
        for i in lati_v:
            if i[0] in comb and i[1] in comb:
                lati_S.append(x[i])
    model += lpSum(lati_S) <= len(comb) - 1

# Chiamata al solver
model.solve()

# Stampa soluzione ottima se trovata
if LpStatus[model.status] == "Optimal":
    for v in model.variables():
        print(v.name, " = ", v.varValue)

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

elif LpStatus[model.status] == "Infeasible":
    print("Istanza non ammissibile")
```

Esempi di applicazioni reali

Il prestito interbibliotecario della Provincia di Brescia

Quale collegamento può esserci tra delle biblioteche e la ricerca operativa?

Uno dei servizi principali offerti dalla Rete Bibliotecaria Bresciana e Cremonese è il **prestito interbibliotecario**, implementato tra le più di trecento biblioteche delle province di Brescia e Cremona. Gussago e Bazzoli si sono concentrati solo sulle biblioteche della provincia di Brescia, riducendo così il numero a circa duecentotrenta.

Biblioteche nella Rete Bibliotecaria Bresciana e Cremonese
© OpenStreetMap contributors.

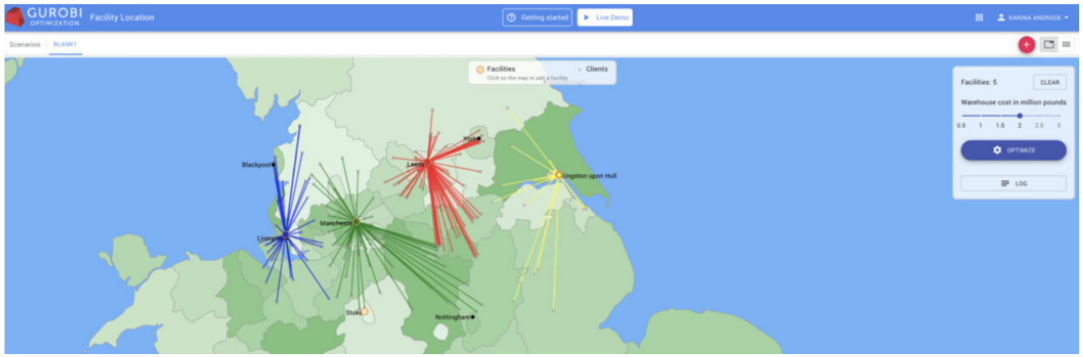
Quando un utente ha bisogno di una risorsa non posseduta dalla biblioteca locale a cui è iscritto, la sua richiesta può essere soddisfatta da una delle altre biblioteche presenti nella rete che invece l'hanno disponibile. La richiesta viene inserita nel sistema informativo della RBBC e viene quindi soddisfatta da una di queste altre biblioteche. Il libro o il dvd (o un'altra tipologia di *esemplari* disponibili nella rete) viene preparato dalla biblioteca prestante e viene ritirato da un corriere, che porta l'esemplare in un centro di raccolta, situato nella periferia di Brescia. Appena possibile, nei giorni successivi, la risorsa verrà consegnata da un altro corriere alla biblioteca di destinazione, quella dove l'utente andrà a ritirarla.

`https://maddmaths.simai.eu/divulgazione/roar-in-azione-prestito-interbibliotecario-problema-routing/`

Amazon Logistics e i problemi di Facility Location

I Facility Location Problem

Supponiamo di avere a disposizione un insieme di strutture da usare come depositi di beni da consegnare. Dobbiamo decidere quali di queste strutture convenga effettivamente *attivare*, ossia quali di queste avviare e poi mantenere il funzionamento. A ogni struttura si vuole assegnare un insieme di clienti che verranno poi da essa serviti.



Esempio di Facility Location Problem (fonte: Gurobi)

Il posizionamento di queste strutture e le distanze dai singoli clienti diventano uno degli aspetti chiave da considerare. Infatti, attivare più o meno strutture avrà un impatto sui costi che sarà necessario sostenere per soddisfare le richieste dei clienti.

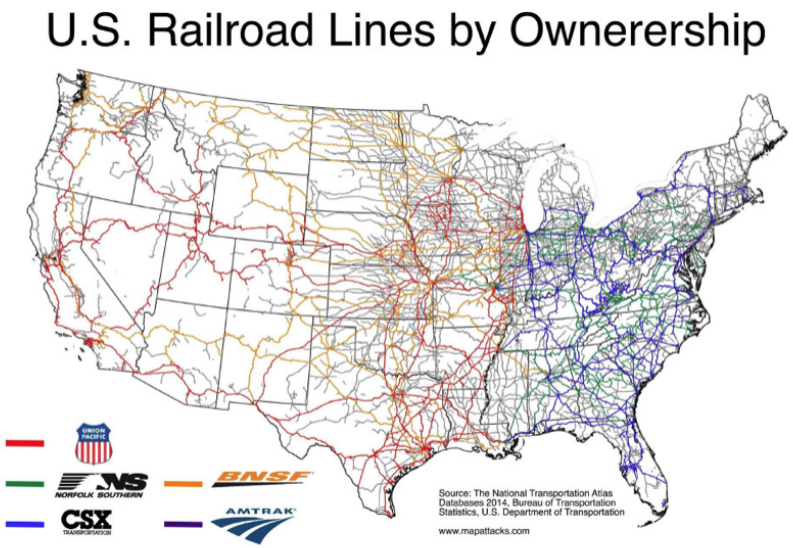
Questo problema si applica non solo alle consegne degli ordini fatti attraverso una piattaforma online, bensì è molto rilevante anche quando, per esempio, bisogna decidere quante e quali scuole, ospedali, stazioni di polizia o fermate dell'autobus sia opportuno avere.

`https://maddmaths.simai.eu/divulgazione/roar-in-azione-amazon-logistics-facility-location-problem/`

Come la matematica aiuta a far funzionare meglio i treni

Esempio di pianificazione: il Block Planning Problem

Consideriamo la rete ferroviaria degli Stati Uniti, mostrata nella figura sottostante.



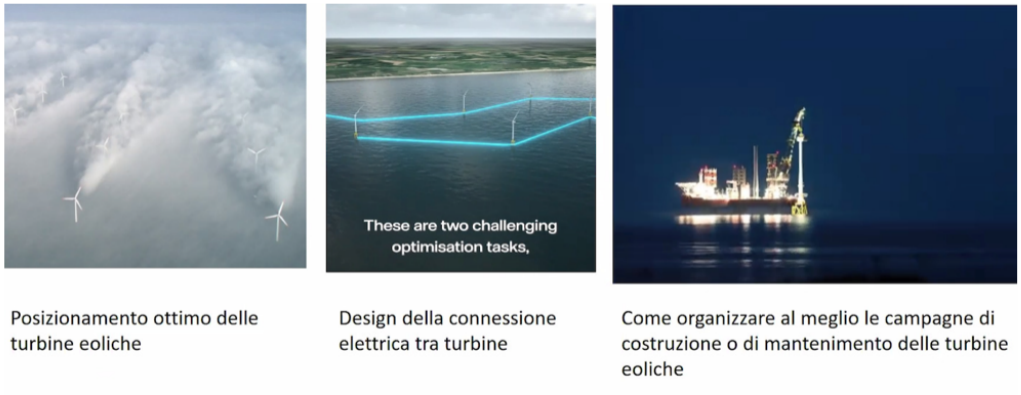
Rete ferroviaria degli Stati Uniti (fonte: Map Attacks).

A differenza dell'Italia o di altri Stati europei, le ferrovie statunitensi sono prevalentemente usate per il trasporto delle merci. Date le grandi distanze e l'abbondanza di terre sconfinate, le persone per spostarsi preferiscono infatti ricorrere agli aerei.
La rete ferroviaria è divisa tra più compagnie che non solo operano i treni, ma sono anche proprietarie di quella parte di rete. Le stazioni in media sono molto piccole ma con molte piattaforme, le zone estese della rete sono a singolo binario, e i treni possono arrivare a essere lunghi anche tre chilometri.

https://maddmaths.simai.eu/divulgazione/roar-in-azione-matematica-fa-funzionare-meglio-treni/

La matematica per un futuro più sostenibile

Dopo il conseguimento del titolo di dottorato nel 2018, ha lavorato per Vattenfall come ingegnere leader, gestendo e supervisionando un gruppo di ricercatori operativi e sviluppando in prima persona modelli e codici. Tra i progetti su cui ha lavorato, ci sono il posizionamento ottimale delle turbine eoliche (ce ne aveva parlato anche qui), il design delle connessioni elettriche tra le turbine, e le campagne di costruzione e mantenimento delle stesse.



Alcuni progetti su cui ha lavorato Martina Fischetti durante il periodo in Vattenfall.

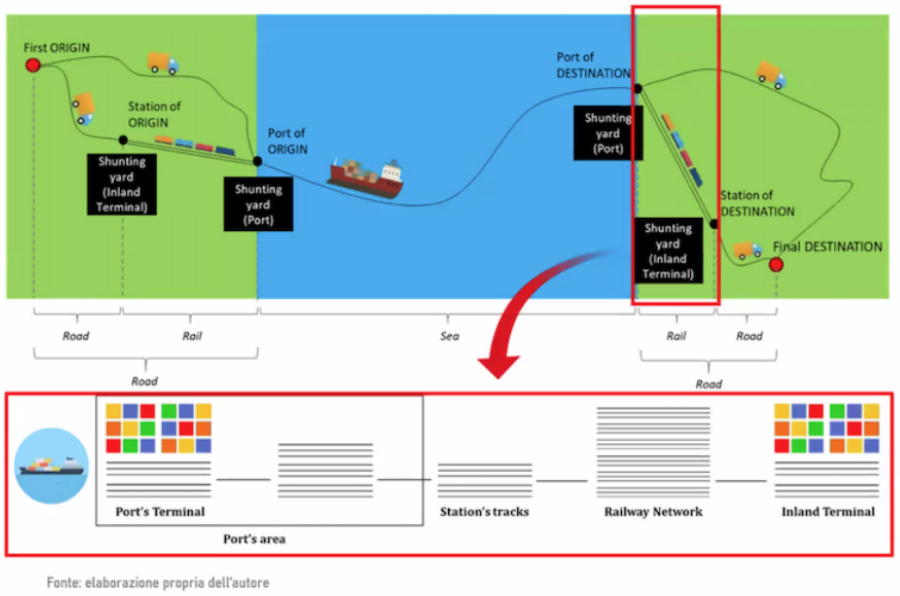
Il suo lavoro ha ottenuto numerosi premi internazionali, tra cui l'*EURO Doctoral Dissertation Award* nel 2019 e il titolo di *Best Industrial Ph.D.* da Innovation Fund Denmark, o il posto da finalista per il premio Edelman, sempre nel 2019. Nel 2021, è stata nominata una delle dodici *YoungWomen4OR* da parte di WISDOM, il forum di EURO per supportare e incoraggiare tutti i generi nel settore della ricerca operativa.

Ora che si trova in uno dei centri di ricerca della Commissione Europea, Martina Fischetti continua ancora a occuparsi di applicazioni di ricerca operativa, ma con un focus particolare sulla *decarbonizzazione* dei trasporti, in particolare del trasporto pubblico.

https://maddmaths.simai.eu/divulgazione/roar-in-azione-matematica-per-futuro-piu-sostenibile/

Ottimizzazione di operazioni ferroviarie in area portuale

Quando si vogliono trasferire delle merci da un determinato punto di partenza a una certa destinazione, può capitare che le modalità di trasporto adoperate siano più di una. Spesso, infatti, non è sufficiente usare dei mezzi "su gomma" (per esempio, dei camion), ma è necessario pianificare dei tratti "su rotaia" (per esempio, con dei treni) o via mare.



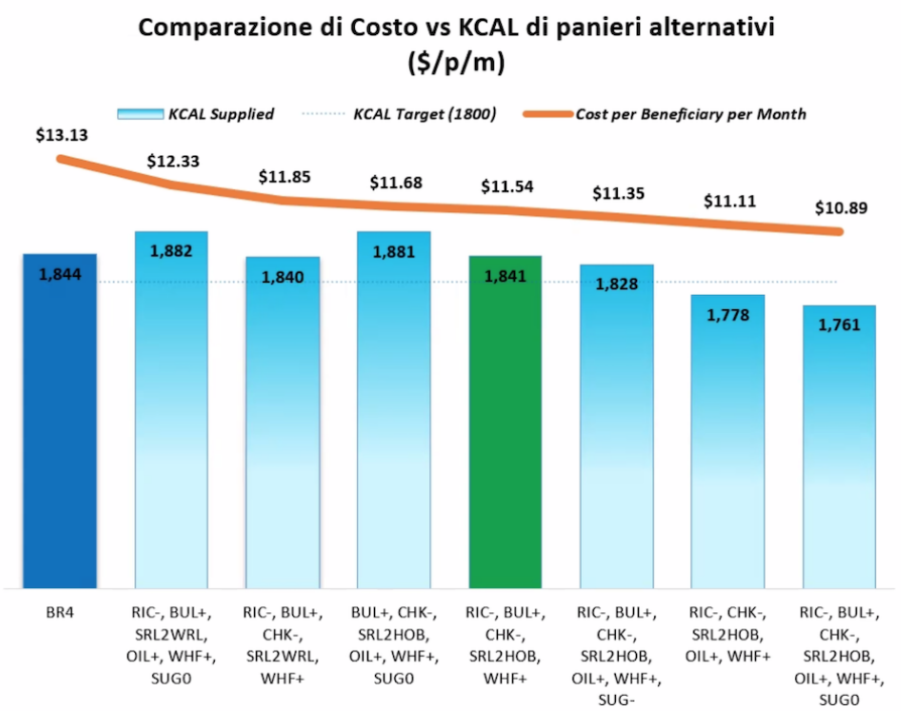
Fonte: elaborazione propria dell'autore

Trasferimento di merci da una data origine a una data destinazione, attraverso diverse modalità di trasporto.

https://maddmaths.simai.eu/divulgazione/roar-in-azione-ottimizzazione-operazioni-ferroviarie-area-portua

OPTIMUS: combattere la fame nel mondo con l'ottimizzazione

OPTIMUS permette perciò di valutare diverse alternative allo stesso tempo, come è stato fatto per esempio in Iraq, nel 2016-2017.



Analisi multiscenario effettuata in Iraq nel 2016-2017. Rispetto alla soluzione di partenza (colonna blu), nella soluzione alternativa considerata (colonna verde) il paniere ottimo era composto di meno riso e meno ceci, più bulgur e farina di grano fortificata.

Conclusioni

Verso i metodi della Programmazione Lineare e Lineare Intera

Un problema di *Programmazione Lineare* ha la seguente forma:

$$\begin{aligned} \min \text{ or } \max \quad & c^T x \\ Ax &\geq b \\ x &\geq 0, \end{aligned}$$

dove $x \in \mathbb{R}^n$ è il vettore delle *variabili decisionali*, $b \in \mathbb{R}^m$ è il vettore dei *termini noti*, $A \in \mathbb{R}^{m \times n}$, e $c \in \mathbb{R}^n$ è il vettore dei *coefficienti* nella funzione obiettivo.

Come abbiamo visto, se tutte le variabili sono intere, allora siamo nel contesto della *Programmazione Lineare Intera*.

In caso il problema abbia solo due variabili, si può pensare di ricorrere al **metodo grafico**. In generale, per risolvere problemi lineari di ottimizzazione con n variabili e m vincoli, si ricorre al **metodo del simplesso**.

Corsi di approfondimento nelle lauree magistrali UniVR

Mathematics for decisions (2021/2022)

Codice insegnamento
4S008838

Docenti
Romeo Rizzi, Alice Raffaele

Coordinatore
Romeo Rizzi

Crediti
6

Settore Scientifico Disciplinare (SSD)
MAT/09 - RICERCA OPERATIVA

Lingua di erogazione
Inglese

Periodo
Primo semestre dal 4-ott-2021 al 28-gen-2022.

Master's degree in Mathematics

Discrete optimization and decision making (2021/2022)

Teaching code
4S009081

Academic staff
Roberto Zanotti, Alice Raffaele

Coordinatore
Roberto Zanotti

Credits
6

Scientific Disciplinary Sector (SSD)
MAT/09 - OPERATIONS RESEARCH

Language
English

Period
Secondo semestre dal Mar 7, 2022 al Jun 10, 2022.

Master's degree in Data Science

Riferimenti

Bibliografia utile:

- Fischetti, Matteo. *Introduction to Mathematical Optimization*, Kindle Direct Publishing, 2019.
- Williams, H. Paul. *Model building in mathematical programming*. John Wiley & Sons, 2013.
- Vanderbei, Robert J.. *Linear Programming: Foundations and Extensions*, Springer Nature, 4th edition, 2013.
- Dantzig, George B., and Mukund N. Thapa. *Linear programming 1: introduction*. Springer Science & Business Media, 2006.

Sitografia:

- *Optimization with PuLP*, `https://coin-or.github.io/pulp/`.
- *ROAR* (Ricerca Operativa Applicazioni Reali), `https://github.com/aliceraffaele/ROAR`.