ROAR III

Ricerca Operativa Applicazioni Reali

Alessandro Gobbi Alice Raffaele Gabriella Colajanni Eugenia Taranto

IIS Antonietti, Iseo (BS)

14 novembre 2022

Introduzione

Chi siamo e i nostri contatti

Alessandro Gobbi (UniBS)
alessandro.gobbi@unibs.it

Alice Raffaele (UniVR)
alice.raffaele@univr.it

Gabriella Colajanni (UniCT)
gabriella.colajanni@unict.it

Eugenia Taranto (UniCT)
eugenia.taranto@unict.it

Correzione dei compiti – Trasporto

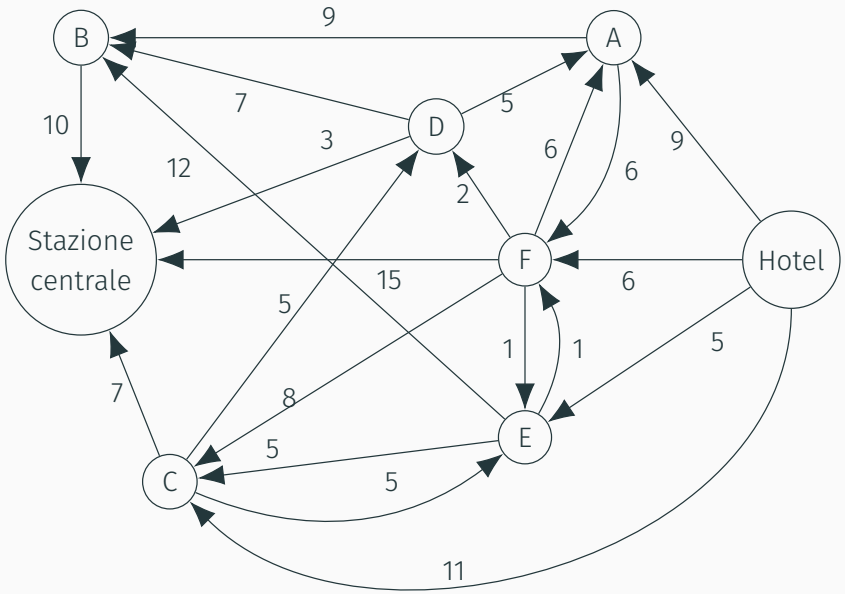
```python
from pulp import *

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

# Set e parametri
indici_impianti = [1,2,3,4]
indici_bar = [1,2,3]
capacita_prod = {1: 75, 2: 90, 3: 80, 4: 75}
domanda = {1: 60, 2: 75, 3: 80}
costi = {
    1: {1: 0.4, 2: 0.3, 3: 0.2},
    2: {1: 0.2, 2: 0.3, 3: 0.5},
    3: {1: 0.1, 2: 0.6, 3: 0.2},
    4: {1: 0.5, 2: 0.1, 3: 0.3},
}
num_max_impianti_attivati = 3
costo_attivazione_impianti = 1350

# Variabili
x = LpVariable.dicts("x", (indici_impianti, indici_bar), 0, None, LpContinuous)
y = LpVariable.dicts("y", indici_impianti, 0, 1, LpBinary)

# Funzione obiettivo
model += lpSum(x[i][j]*costi[i][j] for i in indici_impianti for j in indici_bar) + lpSum(y[i]*costo_attivazione_impianti for i in indici_impianti)

# Vincoli
for i in indici_impianti:
    model += lpSum(x[i][j] for j in indici_bar) <= capacita_prod[i] * y[i]

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

model += lpSum(y[i] for i in indici_impianti) <= num_max_impianti_attivati

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

```
# Per sapere se l'istanza del problema è stata risolta all'ottimo (status == 1)
# oppure se è infeasible (status == -1)
if model.status == 1:
    print("Soluzione ottima trovata\n")

    # 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)))

    # Per sapere quanto caffè è prodotto da ogni impianto
    for i in indici_impianti:
        produzione = 0;
        for j in indici_bar:
            produzione += x[i][j].varValue
        print("L'impianto " + str(i) + " produce " + str(produzione) + " Kg di caffè.")

    # Per sapere il numero di impianti aperti
    numero = 0;
    for i in indici_impianti:
        numero += y[i].varValue
    print("Numero impianti aperti: " + str(numero));
else:
    if model.status == -1:
        print("Infeasible")
```

Correzione dei compiti – Un treno da prendere

Un treno da prendere

Per Remo è arrivato il giorno di lasciare Grafopoli e di tornare a casa. Il treno di ritorno partirà dalla Stazione Centrale alle 11:03 ma Remo, come sempre in ritardo, riesce a lasciare l'Hotel solo alle 10:50. Considerando affidabili i tempi di percorrenza (in minuti) riportati nel grafo, rappresentante la rete cittadina, formulare il modello matematico del problema e implementarlo e risolverlo con Python+PuLP.



Dato un grafo diretto $G = (V, A)$ pesato (sia $c_{i,j}$ il peso dell'arco (i, j)) e dati due nodi s e $d \in V$, il modello matematico di PLI del problema di cammino minimo da s a d è il seguente:

$$\textbf{min} \sum_{(i,j)\in A} c_{i,j} \cdot x_{i,j} \qquad \text{funzione obiettivo}$$

$$\sum_{(s,j)\in A} x_{s,j} = 1 \qquad \text{nodo sorgente}$$

$$\sum_{(i,d)\in A} x_{i,d} = 1 \qquad \text{nodo destinazione}$$

$$\sum_{(i,j)\in A} x_{i,j} = \sum_{(j,k)\in A} x_{j,k} \qquad \text{per ogni nodo } j \in V \setminus \{s, d\}$$

$$x_{i,j} \in \{0, 1\} \qquad \text{per ogni arco } (i, j) \in A.$$

```
from pulp import *

# Inizializzazione del problema assegnando un nome e la direzione dell'ottimizzazione
modello = 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}

sorgente = "H"
destinazione = "S"

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

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

# Vincoli
modello += lpSum(x[i,j] for i,j in A.keys() if i == sorgente) == 1
modello += lpSum(x[i,j] for i,j in A.keys() if j == destinazione) == 1

for v in V:
    if v not in [sorgente, destinazione]:
        modello += 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
status = modello.solve()
```

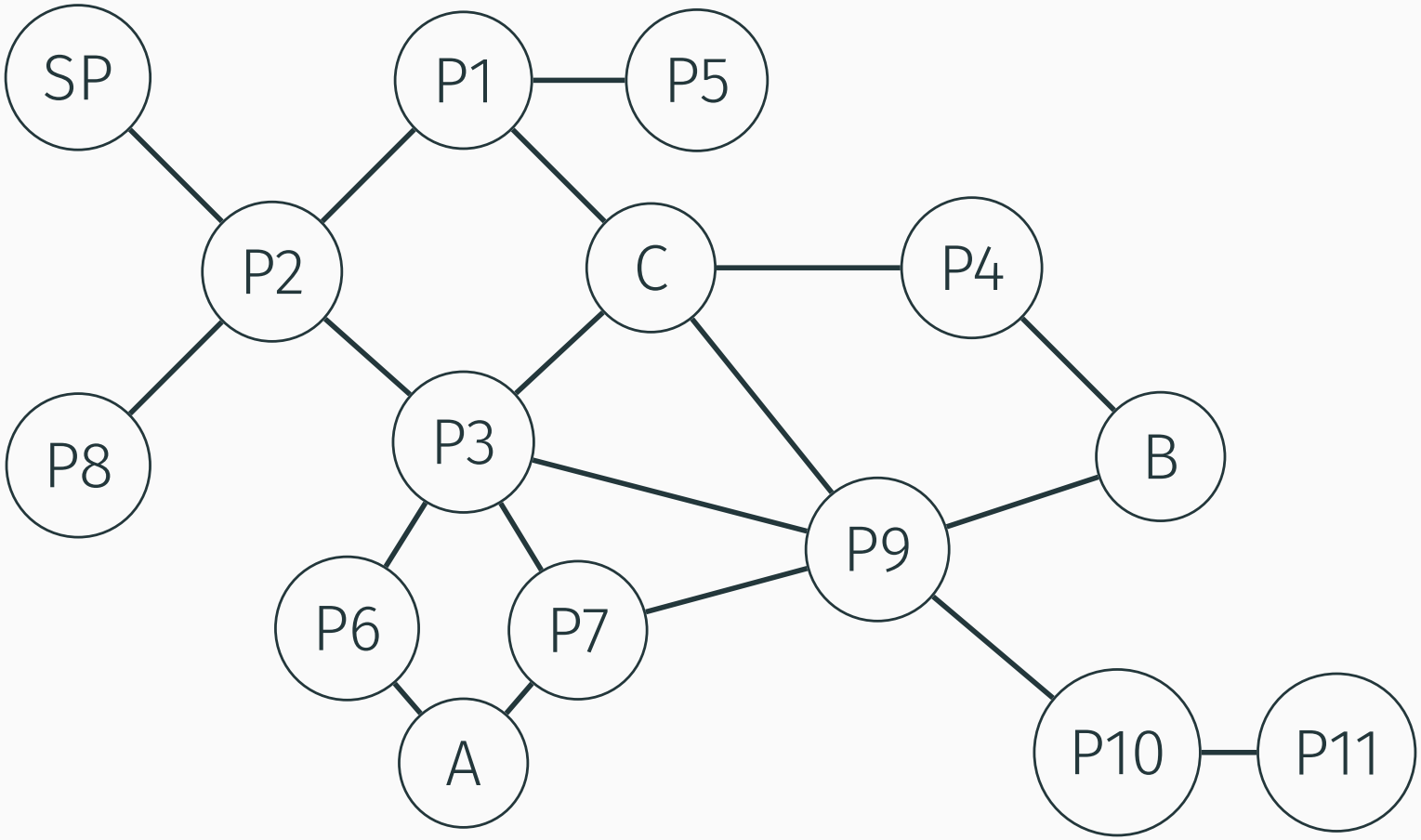
```
from pulp import *

# Inizializzazione del problema assegnando un nome e la direzione dell'ottimizzazione
# Stampa soluzione ottima se trovata
if status == 1:
    print("Soluzione ottima:")
    for v in modello.variables():
        print("{} = {}".format(v.name, v.varValue))

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

elif status == -1:
    print("Istanza non ammissibile")
```

Parte 1:
Lavoro di gruppo (45 minuti)

Il Maximum Independent Set

Cestini e pattumiere

L'assessore per le politiche ambientali del Comune di Iseo vuole posizionare dei nuovi cestini davanti agli edifici e presso gli incroci principali del paese. In particolare, in ogni punto può scegliere se ricorrere a un cestino "all in one" per la raccolta differenziata (con scomparti separati per carta, plastica, vetro e indifferenziato), o se mettere una classica pattumiera per l'indifferenziato, oppure se non posizionare nulla. Infatti, in ogni strada, vuole posizionare al più un cestino, all'inizio o alla fine della via. Un cestino "all in one" costa 190 €, mentre una pattumiera classica costa 20 €. L'assessore non può spendere più di 1000 € in totale. Inoltre, vuole mettere almeno tre cestini "all in one" tra l'IIS Antonietti, la Biblioteca, il Comune e la scuola primaria. Dove dovrebbe posizionare i cestini e le pattumiere per massimizzarne il numero totale rispettando tutte le condizioni?



A = IIS Antonietti; B = Biblioteca; C = Comune; SP = Scuola primaria.

Lavoro di gruppo

1. Formulare il modello matematico del problema utilizzando il paradigma della Programmazione Lineare Intera.
2. Implementare e risolvere il modello matematico del problema con Python+PuLP, sfruttando `dicts` e i cicli for dove possibile per la definizione di variabili e vincoli.
3. Stampare il valore della funzione obiettivo all'ottimo, quello delle sole variabili che all'ottimo non sono nulle, e quanto ha dovuto spendere l'assessore.
4. Come si potrebbe modificare il modello per dare maggiore priorità ai punti più "sporchi" del paese, dove sarebbe meglio posizionare cestini e pattumiere (sempre tenendo conto degli altri vincoli)?

Correzione – Formulazione del modello matematico

Sia $G = (V, E)$ il grafo indiretto rappresentante il comune di Iseo.

- **Variabili**: una variabile binaria x per ogni vertice i in V e per ogni tipologia di bidone: $xC_i, xP_i, \forall i \in V$ tale che $xC_i = 1$ o $xP_i = 1$ se in i c'è un cestino o una pattumiera, 0 altrimenti.
- **Vincoli**:
  - al più un cestino/pattumiera per ogni strada:

$$xC_i + xP_i + xC_j + xP_j \leq 1, \quad \forall (i, j) \in E$$

  - almeno tre cestini tra l'IIS Antonietti, la biblioteca, il comune e la scuola primaria: $xC_A + xC_B + xC_C + xC_{SP} \geq 3$
  - budget disponibile:

$$costo_{cest} * \sum_{i \in V} xC_i + costo_{pattum} * \sum_{i \in V} xP_i \leq budget$$

- **Funzione obiettivo**: massimizzare il numero di cestini e pattumiere:

$$\max \quad \sum_{i \in V} (xC_i + xP_i)$$

Correzione – Modello matematico

$$\max \quad \sum_{i \in V} (xC_i + xP_i)$$

$$xC_i + xP_i + xC_j + xP_j \leq 1, \quad \forall (i,j) \in E$$

$$xC_A + xC_B + xC_C + xC_{SP} \geq num\_min\_cestini\_spec$$

$$costo_{cest} * \sum_{i \in V} xC_i + costo_{pattum} * \sum_{i \in V} xP_i \leq budget$$

$$xC_i, xP_i \in \{0,1\}, \quad \forall i \in V$$

Correzione – Implementazione con Python+PuLP

```
from pulp import *

V = ['P1', 'P2', 'P3', 'P4', 'P5', 'P6', 'P7', 'P8', 'P9', 'P10', 'P11', 'A', 'B', 'C', 'SP']
E = [('P1', 'P2'), ('P1', 'P5'), ('P1', 'C'), ('P2', 'P3'), ('P2', 'P8'), ('P2', 'SP'),
    ('P3', 'P6'), ('P3', 'P7'), ('P3', 'P9'), ('P3', 'C'), ('P4', 'B'), ('P4', 'C'), ('P6', 'A'),
    ('P7', 'P9'), ('P7', 'A'), ('P9', 'P10'), ('P9', 'B'), ('P9', 'C'), ('P10', 'P11')]
costo_cestino = 190
costo_pattumiera = 20
budget = 1000

# Definizione del modello
modello = LpProblem('cestini-pattumiere', LpMaximize)

# Variabili
xC = LpVariable.dicts('xC', V, lowBound=0, upBound=1, cat=LpBinary)
xP = LpVariable.dicts('xP', V, lowBound=0, upBound=1, cat=LpBinary)

# Vincoli
# Al più un cestino/pattumiera in ogni strada
for (i,j) in E:
    modello += xC[i] + xP[i] + xC[j] + xP[j] <= 1
# Almeno tre cestini in A, B, C, SP
modello += xC['A'] + xC['B'] + xC['C'] + xC['SP'] >= 3
# Budget
modello += lpSum(xC[i]*costo_cestino + xP[i]*costo_pattumiera for i in V) <= budget

# Funzione obiettivo
modello += lpSum(xC[i] + xP[i] for i in V)

# Ottimizzazione
status = modello.solve(PULP_CBC_CMD(msg=0))
```

```
# Stampa della soluzione trovata
if status == 1:
    print("Numero massimo di cestini/pattumiere posizionabili:", round(value(modello.objective)))
    print("\nIn particolare:")
    spese = 0
    for i in V:
        if xC[i].varValue > 0:
            print(i + '- cestino all-in-one')
            spese += costo_cestino
        if xP[i].varValue > 0:
            print(i + ' - pattumiera')
            spese += costo_pattumiera
    print("\nBudget speso: " + str(spese) + " €")
```

Correzione – Soluzione ottima

```
Numero massimo di cestini/pattumiere posizionabili: 8

In particolare:
P5 – pattumiera
P6 – pattumiera
P7 – pattumiera
P8 – pattumiera
P10 – pattumiera
B– cestino all-in-one
C– cestino all-in-one
SP– cestino all-in-one

Budget speso: 670 €
```

Lettura dell'istanza da file

- E se avessimo i dati di un'altra città e dovessimo risolvere lo stesso identico problema, solo su altre strade e con altri punti speciali, e magari con altri costi o budget?
- Il modello sarebbe lo stesso, cambierebbero solo i dati!
- Possiamo:
  1. scrivere i dati dell'*istanza* specifica di questo problema in un file di testo a parte;
  2. definire una funzione in Python per *leggere i dati*;
  3. *separare*, nell'implementazione in Python, la logica del modello dai dati dell'istanza (in altre parole, niente dati specifici dell'istanza nel modello).

Lettura dell'istanza da file – 1) File di testo per l'istanza

```
PUNTI = P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 A B C SP
NUM_LATI = 19
P1 P2
P1 P5
P1 C
P2 P3
P2 P8
P2 SP
P3 P6
P3 P7
P3 P9
P3 C
P4 B
P4 C
P6 A
P7 P9
P7 A
P9 P10
P9 B
P9 C
P10 P11
COSTO_CESTINO = 190
COSTO_PATTUMIERA = 20
BUDGET = 1000
SPECIALI = A B C SP
NUMERO_MINIMO_CESTINI_SPEC = 3
```

Lettura dell'istanza da file – 2) Definizione di una funzione `leggi_input`

```
def leggi_input(nome_file):
    V = []
    E = []
    costo_cestino = 0
    costo_pattumiera = 0
    budget = 0
    V_speciali = []
    num_minimo_cestini_speciali = 0
    with open(nome_file, 'r') as file:
       while True:
           riga = file.readline()
           if not riga:
               break
           if "PUNTI" in riga:
               riga_pezzi = riga.split()
               V = riga_pezzi[2:] # dal terzo elemento in poi (scartiamo "PUNTI" e "=")
           if "NUM_LATI" in riga:
               riga_pezzi = riga.split()
               num_lati = int(riga_pezzi[2])
               for i in range(num_lati):
                   riga_pezzi = file.readline().split()
                   lato = (riga_pezzi[0], riga_pezzi[1])
                   E += [lato]
           if "COSTO_CESTINO" in riga:
               riga_pezzi = riga.split()
               costo_cestino = int(riga_pezzi[2])
           if "COSTO_PATTUMIERA" in riga:
               riga_pezzi = riga.split()
               costo_pattumiera = int(riga_pezzi[2])
           if "BUDGET" in riga:
               riga_pezzi = riga.split()
               budget = int(riga_pezzi[2])
           if "SPECIALI" in riga:
               riga_pezzi = riga.split()
               V_speciali = riga_pezzi[2:]
           if "NUMERO_MINIMO_CESTINI_SPEC" in riga:
               riga_pezzi = riga.split()
               num_minimo_cestini_speciali = int(riga_pezzi[2])
    file.close()
    return V, E, costo_cestino, costo_pattumiera, budget, V_speciali, num_minimo_cestini_speciali
```

Lettura dell'istanza da file – 3) Aggiornamento dell'implementazione del modello

```
V, E, costo_cestino, costo_pattumiera, budget, V_speciali, num_minimo_cestini_speciali = leggi_input('cestini_pattumiere_v2.txt')

# Definizione del modello
modello = LpProblem('cestini-pattumiere', LpMaximize)

# Variabili
xC = LpVariable.dicts('xC', V, lowBound=0, upBound=1, cat=LpBinary)
xP = LpVariable.dicts('xP', V, lowBound=0, upBound=1, cat=LpBinary)

# Vincoli

# Al più un cestino/pattumiera in ogni strada
for (i,j) in E:
    modello += xC[i] + xP[i] + xC[j] + xP[j] <= 1

# Almeno num_minimo_cestini_speciali tra i punti speciali
modello += lpSum(xC[i] for i in V_speciali) >= num_minimo_cestini_speciali

# Budget
modello += lpSum(xC[i]*costo_cestino + xP[i]*costo_pattumiera for i in V) <= budget

# Funzione obiettivo
modello += lpSum(xC[i] + xP[i] for i in V)
```

Parte 2:
Lavoro di gruppo (1 ora)

Riprendiamo la challenge dell'anno scorso...

VRP Challenge

La catena di supermercati *SuperAmazingMarket* ha deciso di attivare un nuovo servizio: la consegna della spesa a domicilio.
A ogni cliente, dislocato nella provincia di Brescia, che richiede il servizio, dovrà essere consegnato un determinato numero di borse della spesa contenenti la merce richiesta.
Ogni supermercato aderente al servizio, anch'esso collocato all'interno nella provincia, avrà a disposizione per la consegna alcuni veicoli di piccola, media e grande capacità: il veicolo di piccola capacità potrà trasportare al più 30 borse della spesa, il veicolo medio potrà trasportarne al più 50, mentre quello grande al più 70. Ogni giorno, ciascun veicolo caricherà la merce dal supermercato in cui si trova e partirà per effettuare le consegne.

VRP Challenge (cont.)

Ogni veicolo potrà stare in viaggio al più 3 ore, dalla partenza a quando rientrerà nel supermercato da cui è partito. Durante il percorso, potrà transitare (anche più di una volta) accanto ai clienti, senza obbligatoriamente servirli, e accanto ai punti vendita, compreso il proprio (senza però rifornirsi di altre borse).
Una volta rientrato al proprio supermercato, il veicolo non potrà più ripartire. Si prevede che ogni corriere (che guiderà il veicolo e consegnerà la spesa) costerà all'azienda 50€/giorno, e che impiegherà circa 1 minuto per portare una coppia di borse dal veicolo all'abitazione del cliente. Durante la fase effettiva di consegna, il veicolo sarà spento. Si stima infine che la benzina avrà un costo di 0.07 €/minuto. L'azienda chiede il vostro aiuto per cercare di spendere il meno possibile, stabilendo:

- quali veicoli far partire da ogni supermercato;
- quale percorso deve seguire ogni veicolo;
- quali clienti deve servire ogni veicolo.

VRP Challenge (cont.)

In particolare, l'azienda è interessata alle soluzioni del problema su tre ipotetici diversi scenari, caratterizzati da:

- numero di clienti da servire;
- numero di supermercati;
- numero di veicoli a disposizione e loro assegnazione ai vari supermercati.

In ogni scenario, si ipotizza inoltre quante borse della spesa ogni cliente debba ricevere e quali tratte stradali i veicoli possano percorrere.
Tutte queste informazioni sono riassunte in tre diversi file di testo, uno per ogni scenario.

Implementazione con Python+PuLP

Nel file *vrp_challenge.py* trovate implementato il modello matematico che rappresenta il problema VRP Challenge. In particolare:

- la funzione *leggi_input* per leggere un'istanza da file di testo;
- le variabili decisionali del problema;
- **alcuni** vincoli.

Purtroppo, infatti, sono stati cancellati per sbaglio la funzione obiettivo e alcuni vincoli del problema. Per fortuna, sono però rimasti i commenti che descrivevano cosa imponessero quegli stessi vincoli.

Lavoro di gruppo

1. Leggere bene lo script *`vrp_challenge.py`* e comprendere perché sono state definite certe variabili e perché sono stati imposti certi vincoli.
2. Completare lo script implementando in Python+PuLP le istruzioni mancanti, seguendo i suggerimenti nei commenti.

Correzione – Funzione obiettivo

Obiettivo: minimizzare i costi totali (costi fissi di utilizzo veicoli + costi variabili di trasporto)

```
modello += costo_fisso_veicolo*lpSum(u[v] for v in V)
        + costo_benzina_minuto*lpSum(x[i][j][v]*tempi[i][j]
        for i in S+C for j in S+C for v in V)
```

Correzione – Vincoli a)

Ogni cliente *c* è assegnato a esattamente un veicolo *v*:

```
for c in C:
    modello += lpSum(y[c][v] for v in V) == 1
```

Correzione – Vincoli b)

Ogni cliente *c* può essere servito da un veicolo *v* solo se il veicolo *v* è effettivamente usato:

```
for c in C:
    for v in V:
        modello += y[c][v] <= u[v]
```

Correzione – Vincoli c)

Per ogni veicolo *v*, la somma del numero di borse dei clienti assegnati a *v* non può superare la capacità massima di *v*:

```
for v in V:
    modello += lpSum(y[c][v] * clienti[c]['num_borse']
                     for c in C) <= veicoli[v]['capacità']
```

Conclusione

Compito per sabato 26 novembre 2022

Secret Santa

Per Natale, il gruppo di lettura della Biblioteca di Chiari organizza un *Secret Santa* per scambiarsi dei doni. In particolare, ogni partecipante che aderisce all'iniziativa è assegnato a un altro a cui dovrà regalare un libro e, a sua volta, riceverà un libro in dono da un'altra persona ancora.

L'elenco delle persone aderenti è il seguente: Alessandra, Bruna, Carlo, Daniela, Elisa, Fabio, Germana, Katia, Luca, Mariangela, Nicola, Roberta, Simone, Vilma e William. Avendo proposto l'iniziativa anche l'anno precedente, la coordinatrice del gruppo di lettura:

- preferirebbe non riassegnare gli stessi destinatari alle stesse persone;
- vuole evitare che due persone siano assegnate l'una all'altra;
- vuole che sia Fabio a regalare un libro ad Alessandra.

Gli assegnamenti già fatti in passato sono i seguenti: Alessandra ha già regalato un libro a Elisa, Bruna a Nicola, Carlo a Simone, Daniela a Katia, Elisa a Fabio, Fabio a Germana, Katia ad Alessandra, Luca a William.

Quesiti

1. A quale problema classico della Ricerca Operativa che abbiamo già incontrato può essere ridotto questo problema?
2. Implementare e risolvere il problema con Python+PuLP in due modi:
   - 2.1 prima, scrivendo tutti i dati dell'istanza direttamente nello script (come avete sempre fatto finora);
   - 2.2 poi, provando a leggere i dati dell'istanza da un file di testo (che vi forniamo) e adattando di conseguenza lo script.

Consegnare alla Prof.ssa Picchi entrambe le versioni dello script di Python sviluppato **entro mercoledì 24 novembre**.

Sondaggio finale

www.menti.com – Codice: **6607 6880**