Projecte 10: Optimització de rutes per a vehicles autònoms de repartiment

Lucia Garrido, Albert Guillaumet,
Adrià Muro i David Morillo

ÍNDEX

01

Introducció

INTRODUCCIÓ

- Objectiu: Implementar un model matemàtic per a trobar una solució òptima pel problema Vehicle Routing Problem (VRP)

- Metodologia: Hem fet ús d'optimització matemàtica i programació lineal per modelar rutes i assignacions amb una funció objectiu,que minimitzarà el cost, i estarà subjecte a certes restriccions

- Impacte: Millora de l'eficiència operativa en empreses de repartiment mitjançant tècniques avançades d'optimització.

02

Variables

CONSTANTS I VARIABLES DE DECISIÓ

- **N** : Nombre de clients.
- **K** : Nombre de vehicles.
- **Q** : Capacitat de cada vehicle.
- **d***i* : Demanda del client *i*.
- **c***ij* : Cost de viatjar del client *i* al client *j*.
- **x***ij* : Variable binària que indica si hi ha una ruta del client *i* al client *j*.
- **y***i* : Variable binària que indica si el client *i* és atès.

03

Funció objectiu

FUNCIÓ OBJECTIU

- Minimitzar el cost total de les rutes:

$$\text{Minimitzar} \sum_{i=0}^{N} \sum_{j=0}^{N} c_{ij} \cdot x_{ij}$$

04
Restriccions

RESTRICCIONS

- Cada client ha de ser atès exactament una vegada:

$$\sum_{j=1, j\neq i}^{N} x_{ij} = y_i$$

$$\sum_{i=1, i\neq j}^{N} x_{ij} = y_j$$

RESTRICCIONS

- La suma de les demandes dels clients en una ruta no ha de superar la capacitat del vehicle.

$$\sum_{i=1}^{N} d_i \cdot y_i \leq Q$$

RESTRICCIONS

- Cada vehicle ha de començar i acabar al dipòsit.

$$\sum_{j=1}^{N} x_{0j} = K$$

$$\sum_{i=1}^{N} x_{i0} = K$$

05
Dataset utilitzat

DATASET DE MAGATZEMS D'AMAZON

Order_ID	Agent_Age	Agent_Rat...	Store_Lati...	Store_Lon...	Drop_Latit...	Drop_Lon...
ialx566343618	37	4.9	22.745049	75.892471	22.765049	75.912471
akqg208421122	34	4.5	12.913041	77.683237	13.043041	77.813237
njpu434582536	23	4.4	12.914264	77.6784	12.924264	77.6884
rjto796129700	38	4.7	11.003669	76.976494	11.053669	77.026494
zguw716275638	32	4.6	12.972793	80.249982	13.012793	80.289982
fxuu788413734	22	4.8	17.431668	78.408321	17.461668	78.438321
njmo150975311	33	4.7	23.369746	85.33982	23.479746	85.44982
jvjc772545076	35	4.6	12.352058	76.60665	12.482058	76.73665
uaeb808891380	22	4.8	17.433809	78.386744	17.563809	78.516744
bgvc052754213	36	4.2	30.327968	78.046106	30.397968	78.116106
vmau710398846	21	4.7	10.003064	76.307589	10.043064	76.347589
lcwn330553507	23	4.7	18.56245	73.916619	18.65245	74.006619
wcjs752046999	34	4.3	30.899584	75.809346	30.919584	75.829346
blhl288691670	24	4.7	26.463504	80.372929	26.593504	80.502929

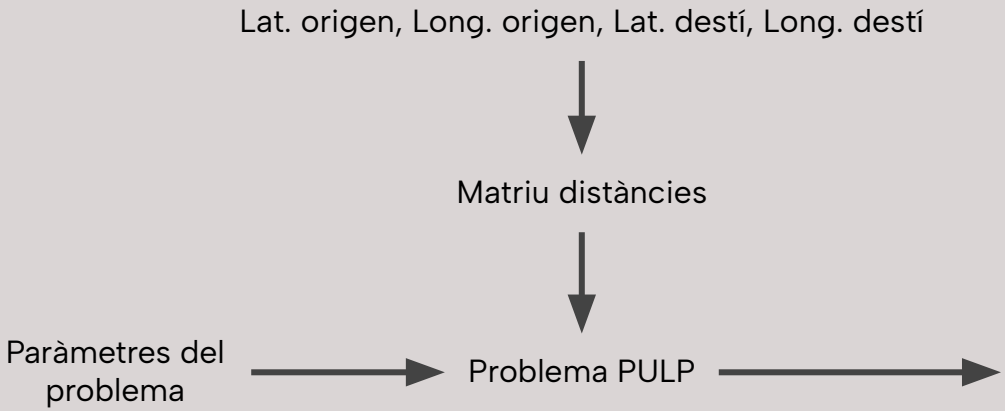
https://www.kaggle.com/datasets/sujalsuthar/amazon-delivery-dataset

06
Implementació

CODI FENT SERVIR PuLP

```
# Variables de decisió
x = pulp.LpVariable.dicts(
    "x",
    ((i, j, k) for i in range(N + 1) for j in range(N + 1) for k in range(K)),
    cat='Binary'
)
y = pulp.LpVariable.dicts(
    "y",
    ((i, k) for i in range(N + 1) for k in range(K)),
    cat='Binary'
)

# Funció objectiu: minimitzar el cost total de viatge
vrp += pulp.lpSum(costos[i][j] * x[i, j, k] for i in range(N + 1) for j in range(N + 1) for k in range(K))

# Restriccions

# Cada client ha de ser atès exactament una vegada
for i in range(1, N + 1):
    vrp += pulp.lpSum(y[i, k] for k in range(K)) == 1, f"Atendre_client_{i}"

# Capacitat dels vehicles
for k in range(K):
    vrp += pulp.lpSum(demanda[i] * y[i, k] for i in range(1, N + 1)) <= Q, f"Capacitat_vehicle_{k}"

# Cada vehicle ha de sortir i tornar al dipòsit exactament una vegada
for k in range(K):
    vrp += pulp.lpSum(x[0, j, k] for j in range(1, N + 1)) == 1, f"Sortida_dipòsit_{k}"
    vrp += pulp.lpSum(x[i, 0, k] for i in range(1, N + 1)) == 1, f"Tornada_dipòsit_{k}"

# Flux d'entrada i sortida de cada client
for k in range(K):
    for i in range(1, N + 1):
        vrp += pulp.lpSum(x[i, j, k] for j in range(N + 1) if j != i) == y[i, k], f"Flux_sortida_{i}_vehicle_{k}"
        vrp += pulp.lpSum(x[j, i, k] for j in range(N + 1) if j != i) == y[i, k], f"Flux_entrada_{i}_vehicle_{k}"

# Prohibir bucles (un client no pot visitar-se a si mateix)
for i in range(N + 1):
    for k in range(K):
        vrp += x[i, i, k] == 0, f"Prohibir_bucle_{i}_vehicle_{k}"

# Resoldre el problema
vrp.solve()
```

IMPLEMENTACIÓ I RESULTATS




```
Estat de la solució: Optimal
Cost total: 9.215

Matriu de rutes:
[0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1]
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Ruta del vehicle 0:
[4, 5, 8, 10]
Ruta del vehicle 1:
[1, 2, 3]
Ruta del vehicle 2:
[6, 7, 9]
```