Formula:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} d_{i,j}x_{i,j} \\
\text{s.t.} \quad & \sum_{(i,j)\in A:j=h} x_{i,j} = 1 && \forall\, h \in C \\
& \sum_{(i,j)\in A:i=h} x_{i,j} = 1 && \forall\, h \in C \\
& x_{i,j} = 1 \Rightarrow u_i + v_j = u_j && \forall\, (i,j) \in A : i \neq 0,\ j \neq 0 \\
& x_{i,j} = 1 \Rightarrow w_i + p_j = w_j && \forall\, (i,j) \in A : i \neq 0,\ j \neq 0 \\
& v_i \leq u_i \leq V && \forall\, i \in C \\
& p_i \leq w_i \leq P && \forall\, i \in C \\
& x_{i,j} \in \{0,1\} && \forall\, (i,j) \in A
\end{aligned}
$$

In [1]:
```
import os
os.environ["SPARK_HOME"] = "D:\RISET\Spark\spark-3.1.1-bin-hadoop2.7"
from platform import python_version
print(python_version())
```

```
3.7.10
```

In [4]:
```
import numpy as np
import pandas as pd
import xlwings as xw
import folium
import docplex.mp.solution as mp_sol
from docplex.mp.model import Model
from geopy.distance import great_circle
import json

n = 15
customer = [i for i in range(1, n + 1)]
node = [0] + customer
arcos = [(i,j) for i in node for j in node if i != j]
f = open('pelanggan.json')
data = json.load(f)
# get customer demand and location data
df = pd.DataFrame(data)
# print(df)
# get distance data
distance = np.loadtxt('distance.txt')

vehicle_capacity = 250
```

In [5]:
```
mdl = Model('CVRP')
```

In [6]:
```
x = mdl.binary_var_dict(arcos, name='x')
u = mdl.continuous_var_dict(customer, name='u')
w = mdl.continuous_var_dict(customer, name='w')
```

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

In [7]:
```
mdl.minimize(mdl.sum(distance[(i, j)] * x[(i, j)] for i, j in arcos))
```

$$\sum_{(i,j)\in A:j=h} x_{i,j} = 1 \qquad \forall\, h \in C$$

In [8]:
```
for h in customer:
    mdl.add_constraint(mdl.sum(x[(i,j)] for i,j in arcos if i==h)==1, ctname='out_%d'%h)
```

$$\sum_{(i,j)\in A:i=h} x_{i,j} = 1 \qquad \forall\, h \in C$$

In [9]:
```
for h in customer:
    mdl.add_constraint(mdl.sum(x[(i,j)] for i,j in arcos if j==h)==1, ctname='in_%d'%h)
```

$$
\begin{aligned}
& x_{i,j} = 1 \Rightarrow u_i + v_j = u_j && \forall\, (i,j) \in A : i \neq 0,\ j \neq 0 \\
& x_{i,j} = 1 \Rightarrow w_i + p_j = w_j && \forall\, (i,j) \in A : i \neq 0,\ j \neq 0
\end{aligned}
$$

In [10]:
```
for i, j in arcos:
    if i!=0 and j!=0:
        mdl.add_indicator(x[(i,j)], u[i] + df.demand[j] == u[j])
```

$$
\begin{aligned}
& v_i \leq u_i \leq V && \forall\, i \in C \\
& p_i \leq w_i \leq P && \forall\, i \in C
\end{aligned}
$$

In [12]:
```
for i in customer:
    mdl.add_constraint(df.demand[i] <= u[i])
    mdl.add_constraint(u[i] <= vehicle_capacity)
```

In [13]:
```
mdl.parameters.timelimit = 60
mdl.parameters.parallel = 1
mdl.parameters.threads =  0
# mdl.parameters.benders.strategy = -1
# mdl.parameters.mip.limits.cutsfactor = 0
# mdl.parameters.mip.tolerances.mipgap = 0.1
mdl.parameters.mip.strategy.branch = 0
solucion = mdl.solve(log_output=True)
```

```
 347502 138121    15460.5800    19    20628.0000    15350.5532  2939235   25.58%
 368978 144960    19618.4107     7    20628.0000    15387.1268  3107347   25.41%
Starting limited solution polishing.
 384127 151901    15501.6540    21    20628.0000    15418.5357  3275236   25.25%
 397303 155258    18580.1708    16    20628.0000    15448.3926  3359383   25.11%

Clique cuts applied:  5
Cover cuts applied:  2
Implied bound cuts applied:  390
```

```
Mixed integer rounding cuts applied:  1
Gomory fractional cuts applied:  6

Root node processing (before b&c):
  Real time             =    0.36 sec. (15.81 ticks)
Parallel b&c, 8 threads:
  Real time             =   59.67 sec. (16495.24 ticks)
  Sync time (average)   =    4.02 sec.
  Wait time (average)   =    0.14 sec.
                          ------------
Total (root+branch&cut) =   60.03 sec. (16511.04 ticks)
```

In [14]:
```
mdl.get_solve_status()
```

Out[14]:
```
<JobSolveStatus.FEASIBLE_SOLUTION: 1>
```

In [15]:
```
# Encontrar las rutas por separado
route = []
for h in customer:
    if x[(0, h)].solution_value > 0.9:
        arcos_route = [(0, h)]
        j = h
        while j!=0:
            i = j
            for k in node:
                if i!=k and x[(i, k)].solution_value > 0.9:
                    j = k
                    arcos_route.append((i, j))
                    break
        route.append(arcos_route)
route
```

Out[15]:
```
[[(0, 9), (9, 3), (3, 8), (8, 4), (4, 1), (1, 2), (2, 0)],
 [(0, 10), (10, 0)],
 [(0, 12),
  (12, 5),
  (5, 15),
  (15, 14),
  (14, 7),
  (7, 13),
  (13, 6),
  (6, 11),
  (11, 0)]]
```

In [ ]:
```

```