In [1]:
```
import numpy as np
import math
import sys
from docplex.cp.model import *
```

In [2]:
```
NumberCustomers = 10
Capacity = 200
Customers = [i for i in range(1, NumberCustomers + 1)]
CustomersDepot = [0] + Customers + [NumberCustomers + 1]  ##index 0 and 11 are depot
Vehicles = [i for i in range(1, 4)]  #NV
ReadyTime=[0,74,151,116,42,65,42,90,167,95,180,0]
DueTime = [240,104,181,146,72,95,72,120,197,125,210,240]
ser=[0,10,10,10,10,10,10,10,10,10,10,0]  # Service Time
PDemand=[0,10,40,10,20,23,16,23,3,25,26,0]   # Pick up
DDemand=[0,10,10,30,19,14,9,13,13,23,3,0]    # Delivery
```

In [3]:
```
def compute_euclidean_distance_matrix(locations):
    n_city = len(locations)
    distances = np.zeros((n_city,n_city))
    for from_counter, from_node in enumerate(locations):
        for to_counter, to_node in enumerate(locations):
            if from_counter != to_counter:
                distances[from_counter][to_counter] = (int(
                    math.hypot((from_node[0] - to_node[0]),
                               (from_node[1] - to_node[1]))))
    return distances.astype(int)

XCOORD=[40,88,42,72,10,65,27,12,49,57,31,40]
YCOORD=[50,30,5,35,20,55,43,24,42,48,67,50]
locations=[(40,50),(88,30),(42,5),(72,35),(10,20),(65,55),(27,43),(12,24),(49,42),(57,48),(31,67),(40,50

distance_matrix = compute_euclidean_distance_matrix(locations)
distance_matrix
```

Out[3]:
```
array([[ 0, 52, 45, 35, 42, 25, 14, 38, 12, 17, 19,  0],
       [52,  0, 52, 16, 78, 33, 62, 76, 40, 35, 67, 52],
       [45, 52,  0, 42, 35, 55, 40, 35, 37, 45, 62, 45],
       [35, 16, 42,  0, 63, 21, 45, 61, 24, 19, 52, 35],
       [42, 78, 35, 63,  0, 65, 28,  4, 44, 54, 51, 42],
       [25, 33, 55, 21, 65,  0, 39, 61, 20, 10, 36, 25],
       [14, 62, 40, 45, 28, 39,  0, 24, 22, 30, 24, 14],
       [38, 76, 35, 61,  4, 61, 24,  0, 41, 51, 47, 38],
       [12, 40, 37, 24, 44, 20, 22, 41,  0, 10, 30, 12],
       [17, 35, 45, 19, 54, 10, 30, 51, 10,  0, 32, 17],
       [19, 67, 62, 52, 51, 36, 24, 47, 30, 32,  0, 19],
       [ 0, 52, 45, 35, 42, 25, 14, 38, 12, 17, 19,  0]])
```

In [4]:
```
XCOORD=[40,88,42,72,10,65,27,12,49,57,31,40]
YCOORD=[50,30,5,35,20,55,43,24,42,48,67,50]
A = [(i, j) for i in CustomersDepot for j in CustomersDepot ]
#print(A)
TravelDistance = {(i, j):np.hypot(XCOORD[i]-XCOORD[j], YCOORD[i]-YCOORD[j]) for i, j in A}
#TravelDistance[1,2]

TravelDistance
```

Out[4]:
```
{(0, 0): 0.0,
 (0, 1): 52.0,
 (0, 2): 45.044422518220834,
 (0, 3): 35.34119409414458,
 (0, 4): 42.42640687119285,
 (0, 5): 25.495097567963924,
 (0, 6): 14.7648230602334,
 (0, 7): 38.2099463490856,
 (0, 8): 12.041594578792296,
 (0, 9): 17.11724276862369,
 (0, 10): 19.235384061671343,
 (0, 11): 0.0,
 (1, 0): 52.0,
 (1, 1): 0.0,
 (1, 2): 52.354560450833695,
 (1, 3): 16.76305461424021,
 (1, 4): 78.63841300535,
 (1, 5): 33.97057550292606,
 (1, 6): 62.369864518050704,
```

In [5]:
```
#d=TravelDistance[1,2]  #d=TravelDistance[i,j]
#print(d)
```

In [6]:
```
mdl = CpoModel()
```

In [7]:
```
x_i_k = {}
for t in CustomersDepot:
    for v in Vehicles:
        iv=mdl.interval_var(start=(ReadyTime[t], INTERVAL_MAX), end=(0, DueTime[t]), size=ser[t])
        iv.set_optional()
        x_i_k[(t,v)]=iv
        print(x_i_k[(t,v)])
#x_i_k
```

```
intervalVar(optional, end=0..240, size=0)
intervalVar(optional, end=0..240, size=0)
intervalVar(optional, end=0..240, size=0)
intervalVar(optional, start=74..intervalmax, end=0..104, size=10)
intervalVar(optional, start=74..intervalmax, end=0..104, size=10)
intervalVar(optional, start=74..intervalmax, end=0..104, size=10)
intervalVar(optional, start=151..intervalmax, end=0..181, size=10)
intervalVar(optional, start=151..intervalmax, end=0..181, size=10)
intervalVar(optional, start=151..intervalmax, end=0..181, size=10)
intervalVar(optional, start=116..intervalmax, end=0..146, size=10)
intervalVar(optional, start=116..intervalmax, end=0..146, size=10)
intervalVar(optional, start=42..intervalmax, end=0..72, size=10)
intervalVar(optional, start=42..intervalmax, end=0..72, size=10)
intervalVar(optional, start=42..intervalmax, end=0..72, size=10)
intervalVar(optional, start=65..intervalmax, end=0..95, size=10)
intervalVar(optional, start=65..intervalmax, end=0..95, size=10)
intervalVar(optional, start=65..intervalmax, end=0..95, size=10)
intervalVar(optional, start=42..intervalmax, end=0..72, size=10)
intervalVar(optional, start=42..intervalmax, end=0..72, size=10)
intervalVar(optional, start=42..intervalmax, end=0..72, size=10)
intervalVar(optional, start=90..intervalmax, end=0..120, size=10)
intervalVar(optional, start=90..intervalmax, end=0..120, size=10)
intervalVar(optional, start=90..intervalmax, end=0..120, size=10)
intervalVar(optional, start=167..intervalmax, end=0..197, size=10)
intervalVar(optional, start=167..intervalmax, end=0..197, size=10)
intervalVar(optional, start=167..intervalmax, end=0..197, size=10)
intervalVar(optional, start=95..intervalmax, end=0..125, size=10)
intervalVar(optional, start=95..intervalmax, end=0..125, size=10)
intervalVar(optional, start=95..intervalmax, end=0..125, size=10)
intervalVar(optional, start=180..intervalmax, end=0..210, size=10)
intervalVar(optional, start=180..intervalmax, end=0..210, size=10)
intervalVar(optional, start=180..intervalmax, end=0..210, size=10)
intervalVar(optional, end=0..240, size=0)
intervalVar(optional, end=0..240, size=0)
intervalVar(optional, end=0..240, size=0)
```

In [8]:
```
y_i = {}
for t in CustomersDepot:
    y_i[t]=mdl.interval_var(start=(ReadyTime[t], INTERVAL_MAX), end=(0, DueTime[t]), size=ser[t])

    print(y_i[t])
#y_i
```

```
intervalVar(end=0..240, size=0)
intervalVar(start=74..intervalmax, end=0..104, size=10)
intervalVar(start=151..intervalmax, end=0..181, size=10)
intervalVar(start=116..intervalmax, end=0..146, size=10)
intervalVar(start=42..intervalmax, end=0..72, size=10)
intervalVar(start=65..intervalmax, end=0..95, size=10)
intervalVar(start=42..intervalmax, end=0..72, size=10)
intervalVar(start=90..intervalmax, end=0..120, size=10)
intervalVar(start=167..intervalmax, end=0..197, size=10)
intervalVar(start=95..intervalmax, end=0..125, size=10)
intervalVar(start=180..intervalmax, end=0..210, size=10)
intervalVar(end=0..240, size=0)
```

In [9]:
```
Q={}
for v in Vehicles:
    Q[v]=mdl.sequence_var([x_i_k[(t, v)] for t in CustomersDepot], types=([t for t in CustomersDepot]))
    print(Q[v])
```

```
sequenceVar([Anonymous, Anonymous, Anonymous, Anonymous, Anonymous, Anonymous, Anonymous, Anonymous, An
onymous, Anonymous, Anonymous, Anonymous], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])
sequenceVar([Anonymous, Anonymous, Anonymous, Anonymous, Anonymous, Anonymous, Anonymous, Anonymous, An
onymous, Anonymous, Anonymous, Anonymous], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])
sequenceVar([Anonymous, Anonymous, Anonymous, Anonymous, Anonymous, Anonymous, Anonymous, Anonymous, An
onymous, Anonymous, Anonymous, Anonymous], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])
```

In [10]:
```
ff={}
for v in Vehicles:
    ff[v]=sum(mdl.step_at_start(x_i_k[t,v], PDemand[t]) for t in Customers )
    - sum(mdl.step_at_start(x_i_k[t,v], DDemand[t]) for t in Customers)
    print(ff[v])
```

```
sum([stepAtStart(intervalVar(optional, start=74..intervalmax, end=0..104, size=10), 10), stepAtStart(in
tervalVar(optional, start=151..intervalmax, end=0..181, size=10), 40), stepAtStart(intervalVar(optional
, start=116..intervalmax, end=0..146, size=10), 10), stepAtStart(intervalVar(optional, start=42..interv
almax, end=0..72, size=10), 20), stepAtStart(intervalVar(optional, start=65..intervalmax, end=0..95, si
ze=10), 23), stepAtStart(intervalVar(optional, start=42..intervalmax, end=0..72, size=10), 16), stepAtS
tart(intervalVar(optional, start=90..intervalmax, end=0..120, size=10), 23), stepAtStart(intervalVar(op
tional, start=167..intervalmax, end=0..197, size=10), 3), stepAtStart(intervalVar(optional, start=95..i
ntervalmax, end=0..125, size=10), 25), stepAtStart(intervalVar(optional, start=180..intervalmax, end=0.
.210, size=10), 26)])
sum([stepAtStart(intervalVar(optional, start=74..intervalmax, end=0..104, size=10), 10), stepAtStart(in
tervalVar(optional, start=151..intervalmax, end=0..181, size=10), 40), stepAtStart(intervalVar(optional
, start=116..intervalmax, end=0..146, size=10), 10), stepAtStart(intervalVar(optional, start=42..interv
almax, end=0..72, size=10), 20), stepAtStart(intervalVar(optional, start=65..intervalmax, end=0..95, si
ze=10), 23), stepAtStart(intervalVar(optional, start=42..intervalmax, end=0..72, size=10), 16), stepAtS
tart(intervalVar(optional, start=90..intervalmax, end=0..120, size=10), 23), stepAtStart(intervalVar(op
tional, start=167..intervalmax, end=0..197, size=10), 3), stepAtStart(intervalVar(optional, start=95..i
ntervalmax, end=0..125, size=10), 25), stepAtStart(intervalVar(optional, start=180..intervalmax, end=0.
.210, size=10), 26)])
sum([stepAtStart(intervalVar(optional, start=74..intervalmax, end=0..104, size=10), 10), stepAtStart(in
tervalVar(optional, start=151..intervalmax, end=0..181, size=10), 40), stepAtStart(intervalVar(optional
, start=116..intervalmax, end=0..146, size=10), 10), stepAtStart(intervalVar(optional, start=42..interv
almax, end=0..72, size=10), 20), stepAtStart(intervalVar(optional, start=65..intervalmax, end=0..95, si
ze=10), 23), stepAtStart(intervalVar(optional, start=42..intervalmax, end=0..72, size=10), 16), stepAtS
tart(intervalVar(optional, start=90..intervalmax, end=0..120, size=10), 23), stepAtStart(intervalVar(op
tional, start=167..intervalmax, end=0..197, size=10), 3), stepAtStart(intervalVar(optional, start=95..i
ntervalmax, end=0..125, size=10), 25), stepAtStart(intervalVar(optional, start=180..intervalmax, end=0.
.210, size=10), 26)])
```

In [11]:
```
for v in Vehicles:
    mdl.add(ff[v] <= Capacity)
```

In [12]:
```
for t in Customers:
    mdl.add(mdl.alternative(y_i[t], [x_i_k[(t,v)] for v in Vehicles]))
```

In [13]:
```
for v in Vehicles:
    mdl.no_overlap(Q[v],distance_matrix)
```

In [14]:
```
for v in Vehicles:
    mdl.presence_of(x_i_k[(0,v)])==1
    mdl.presence_of(x_i_k[(11,v)])==1
    mdl.first(Q[v],x_i_k[(0,v)])
    mdl.last(Q[v],x_i_k[(11,v)])
```

In [15]:
```
obj=sum((TravelDistance[t,t] * mdl.type_of_next(Q[v],x_i_k[(t,v)],t,t)) for v in Vehicles for t in Custo
```

In [16]:
```
mdl.add(minimize(obj))
```

In [17]:
```
## Solve the model

mdl.solve(TimeLimit=100,execfile='/Applications/CPLEX_Studio_Beta211/cpoptimizer/bin/x86-64_osx/cpoptimi
```

```
/var/folders/1n/gk5f1z157bq6zfqr0h_rb3840000gn/T/ipykernel_73538/2027698670.py:1(stream:54:10): Warning
: Sequence variable without noOverlap constraint.

_SEQ_1 = sequenceVar([_ITV_41, _ITV_1, _ITV_2, _ITV_3, _ITV_4, _ITV_5, _ITV_6, _ITV_7, _ITV_8, _ITV_9,
_ITV_10, _ITV_42], [0..11])
/var/folders/1n/gk5f1z157bq6zfqr0h_rb3840000gn/T/ipykernel_73538/2027698670.py:1(stream:57:10): Warning
: Sequence variable without noOverlap constraint.

_SEQ_2 = sequenceVar([_ITV_43, _ITV_11, _ITV_12, _ITV_13, _ITV_14, _ITV_15, _ITV_16, _ITV_17, _ITV_18,
_ITV_19, _ITV_20, _ITV_44], [0..11])
/var/folders/1n/gk5f1z157bq6zfqr0h_rb3840000gn/T/ipykernel_73538/2027698670.py:1(stream:60:10): Warning
: Sequence variable without noOverlap constraint.

_SEQ_3 = sequenceVar([_ITV_45, _ITV_21, _ITV_22, _ITV_23, _ITV_24, _ITV_25, _ITV_26, _ITV_27, _ITV_28,
_ITV_29, _ITV_30, _ITV_46], [0..11])
 ! ------------------------------------------------- CP Optimizer 20.1.0.0 --
 ! Minimization problem - 49 variables, 13 constraints
 ! TimeLimit            = 100
 ! Initial process time : 0.17s (0.17s extraction + 0.00s propagation)
 !  . Log search space  : 80.4 (before), 80.4 (after)
 !  . Memory usage      : 576.7 kB (before), 576.7 kB (after)
 ! Using parallel search with 4 workers.
 ! ----------------------------------------------------------------------------
 !          Best Branches  Non-fixed    W       Branch decision
                       0         49                 -
 + New bound is 0
 *             0       37  0.23s        1      (gap is 0.00%)
 ! ----------------------------------------------------------------------------
 ! Search completed, 1 solution found.
 ! Best objective         : 0 (optimal - effective tol. is 0)
 ! Best bound             : 0
 ! ----------------------------------------------------------------------------
 ! Number of branches     : 117
 ! Number of fails        : 36
 ! Total memory usage     : 2.9 MB (2.8 MB CP Optimizer + 0.1 MB Concert)
 ! Time spent in solve    : 0.23s (0.06s engine + 0.17s extraction)
 ! Search speed (br. / s) : 2340.0
 ! ----------------------------------------------------------------------------
```

Out[17]: <docplex.cp.solution.CpoSolveResult at 0x7f98f5037b90>

In [ ]:

In [ ]:

In [ ]: