CSCI 5654 - Linear Programming - Project

Team Members

1. Ketan Ramesh

2. Shreyas Gopalakrishna

Vehicle Routing Problem with Simultaneous Pickup and Delivery

In [1]:

```
!pip3 install pulp
```

```
Requirement already satisfied: pulp in /usr/local/lib/
python3.6/dist-packages (2.1)
Requirement already satisfied: pyparsing>=2.0.1 in /us
r/local/lib/python3.6/dist-packages (from pulp) (2.4.7
)
```

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import distance
import random
import pulp
import gurobipy as gp
from gurobipy import GRB
from scipy.spatial import distance
```

Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSDP)

Paper Link - [Vehicle Routing Problem with Deliveries and Pickups: Modelling Issues and Meta-heuristics Solution Approaches (https://core.ac.uk/download/pdf/19477982.pdf)](https://core.ac.uk/download/pdf/19477982.pdf)

The code provided below creates a `class` to frame and solve the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSDP) using the two-index flow formulation as mentioned in the paper. The formulation of the problem as a Mixed ILP is as follows:

$$min \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} x_{ij}$$

$$min \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} x_{ij}$$

$$s.t. \sum_{i=0}^{n} x_{ij} = 1, j \in \{1, \ldots, n\}, -(3.1)$$

$$s.t. \sum_{i=0}^{n} x_{ij} = 1, j \in \{1, \ldots, n\}, -(3.1)$$

$$\sum_{i=0}^{n} x_{ji} = 1, j \in \{1, \ldots, n\}, -(3.2)$$

$$\sum_{i=0}^{n} x_{ji} = 1, j \in \{1, \ldots, n\}, -(3.2)$$

$$\sum_{i=0}^{n} R_{ij} - q_j = \sum_{i=0}^{n} R_{ji}, j \in \{1, \ldots, n\}, -(3.3)$$

$$\sum_{i=0}^{n} R_{ij} - q_j = \sum_{i=0}^{n} R_{ji}, j \in \{1, \ldots, n\}, -(3.3)$$

$$\sum_{i=0}^{n} P_{ij} + b_j = \sum_{i=0}^{n} P_{ji}, j \in \{1, \ldots, n\}, -(3.4)$$

$$\sum_{i=0}^{n} P_{ij} + b_j = \sum_{i=0}^{n} P_{ji}, j \in \{1, \ldots, n\}, \ -(3.4)$$

$$\sum_{i=1}^{n} R_{i0} = 0, -(3.5)$$

$$\sum_{i=1}^{n} R_{i0} = 0, \ -(3.5)$$

$$\sum_{i=1}^{n} P_{0i} = 0, -(3.6)$$

$$\sum_{i=1}^{n} P_{0i} = 0, \ -(3.6)$$

$$R_{ij} + P_{ij} \leq Cx_{ij}, i, j \in \{0, \ldots, n\}, -(3.7)$$

$$R_{ij} + P_{ij} \leq Cx_{ij}, i, j \in \{0, \ldots, n\}, \ -(3.7)$$

$$x_{ij} = \{0, 1\}, i, j \in \{1, \ldots n\}$$

$$x_{ij} = \{0, 1\}, i, j \in \{1, \ldots n\}$$

$$R_{ij}, P_{ij} \geq 0, i, j \in \{1, \ldots, n\}$$

$$R_{ij}, P_{ij} \geq 0, i, j \in \{1, \ldots, n\}$$

where,

d_{ij} = Cost to travel from node i to j.

q_i = The delivery demand of node i.

b_i = The pickup demand of node i.

C = Vehicle capacity.

Decision Variables:

1. x_{ij} = {**1**, if edge (i, j) present in any route. **0**, otherwise}.
2. R_{ij} = The amount of delivery goods on board on arc ij.
3. P_{ij} = The amount of pickup goods on board on arc ij.

The objective is to minimize the sum of the distances of all routes that satisfy the given constraints. The constraints (3.1) and (3.2) ensure that every node is visited exactly once (every node leads to only one other node, every node must be visited by only one other node). The constraints (3.3) and (3.4) ensure that the flow conservation is met (at a node *j*, the amount of delivery load after servicing delivery of node *j*, must be equal to pickup load at the same node. The pickup load condition is similar in nature). The constraints (3.5) and (3.6) ensure that the depot has zero pickup and delivery load. The constraint (3.7) ensures that the pickup and demand loads do not exceed the capacity of the vehicle at all nodes in the routes.

In [84]:

```
class VRPSDP_GUROBI:
    def __init__(self, costMatrix, demand, pickup, numberOfVehicles
        self.costMatrix = costMatrix
        self.n = len(costMatrix)
        self.demand = demand
        self.pickup = pickup
        self.numberOfVehicles = numberOfVehicles
        self.capacityOfVehicle = capacityOfVehicle
        self.initialzeLP()

    def initialzeLP(self):
        self.cvrpLP = gp.Model('VRP-SDP')
        x, R, P = [], [], []

        # Create decision variables
        for i in range(self.n):
            xRow, RRow, PRow = [], [], []
            for j in range(self.n):
                xRow.append(self.cvrpLP.addVar(name='x('+str(i)+","+
                RRow.append(self.cvrpLP.addVar(name='R('+str(i)+","+
                PRow.append(self.cvrpLP.addVar(name='P('+str(i)+","+
            x.append(xRow)
            R.append(RRow)
```

```
        P.append(PRow)

    # Create objective
    objective = None
    for i in range(self.n):
        for j in range(self.n):
            objective += self.costMatrix[i][j] * x[i][j]
    self.cvrpLP.setObjective(objective,GRB.MINIMIZE)

    # constraint 1
    for j in range(1, self.n):
        const1 = None
        for i in range(self.n):
            if(const1 == None):
                const1 = x[i][j]
            else:
                const1 = const1 + x[i][j]
        self.cvrpLP.addConstr(const1 == 1)

    # constraint 2
    for j in range(1, self.n):
        const2 = None
        for i in range(self.n):
            if(const2 == None):
                const2 = x[j][i]
            else:
                const2 = const2 + x[j][i]
        self.cvrpLP.addConstr(const2 == 1)

    # constraint 3
    for j in range(1, self.n):
        const3a, const3b = None, None
        for i in range(self.n):
            if(const3a == None):
                const3a = R[i][j]
            else:
                const3a = const3a + R[i][j]
            if(const3b == None):
                const3b = R[j][i]
            else:
                const3b = const3b + R[j][i]
        self.cvrpLP.addConstr(const3a - self.demand[j] == const

    # constraint 4
    for j in range(1, self.n):
```

```
            const4a, const4b = None, None
            for i in range(self.n):
                if(const4a == None):
                    const4a = P[i][j]
                else:
                    const4a = const4a + P[i][j]
                if(const4b == None):
                    const4b = P[j][i]
                else:
                    const4b = const4b + P[j][i]
            self.cvrpLP.addConstr(const4a + self.pickup[j] == const4

        # constraint 5
        const5 = None
        for i in range(1, self.n):
            if(const5 == None):
                const5 = P[0][i]
            else:
                const5 = const5 + P[0][i]
        self.cvrpLP.addConstr(const5 == 0)

        # constraint 6
        const6 = None
        for i in range(1, self.n):
            if(const6 == None):
                const6 = R[i][0]
            else:
                const6 = const6 + R[i][0]
        self.cvrpLP.addConstr(const6 == 0)

        # constraint 7
        for i in range(self.n):
            for j in range(self.n):
                self.cvrpLP.addConstr(R[i][j] + P[i][j] <= self.capa

        # constraint 8
        for i in range(1, self.n):
            self.cvrpLP.addConstr(x[0][i] <= self.numberOfVehicles)

    def solve(self):
        status = self.cvrpLP.optimize()
        print(status)

    def getResult(self):
        print("Objective value: ", self.cvrpLP.ObjVal)
```

```
        for v in self.cvrpLP.getVars():
            print(v.varName, " = ", v.x)
        return self.cvrpLP
```

E-N13-K4.vrp - Christofides and Eilon

In [109]:

```
costMatrix = [[0,9,14,23,32,50,21,49,30,27,35,28,18],
[9,0,21,22,36,52,24,51,36,37,41,30,20],
[14,21,0,25,38,5,31,7,36,43,29,7,6],
[23,22,25,0,42,12,35,17,44,31,31,11,6],
[32,36,38,42,0,22,37,16,46,37,29,13,14],
[50,52,5,12,22,0,41,23,10,39,9,17,16],
[21,24,31,35,37,41,0,26,21,19,10,25,12],
[49,51,7,17,16,23,26,0,30,28,16,27,12],
[30,36,36,44,46,10,21,30,0,25,22,10,20],
[27,37,43,31,37,39,19,28,25,0,20,16,8],
[35,41,29,31,29,9,10,16,22,20,0,10,10],
[28,30,7,11,13,17,25,27,10,16,10,0,10],
[18,20, 6, 6,14,16,12,12,20,8, 10,10,0]]

len(costMatrix)
```

Out[109]:

```
13
```

In [110]:

```
demand = [0, 1200, 1700, 1500, 1400, 1700, 1400, 1200, 1900, 1800, 1
pickup = [1100, 0, 1200, 1700, 1500, 1400, 1700, 1400, 1200, 1900, 1
capacityOfVehicle = 6000
numberOfVehicles = 4
```

In [111]:

```
lp2 = VRPSDP_GUROBI(costMatrix, demand, pickup, numberOfVehicles, ca
lp2.solve()
result = lp2.getResult()
```

```
Gurobi Optimizer version 9.0.2 build v9.0.2rc0 (mac6
4)
Optimize a model with 231 rows, 507 columns and 1431
nonzeros
Model fingerprint: 0xb8164ab9
Variable types: 0 continuous, 507 integer (169 binar
y)
Coefficient statistics:
  Matrix range     [1e+00, 6e+03]
  Objective range  [5e+00, 5e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 2e+03]
Presolve removed 51 rows and 87 columns
Presolve time: 0.01s
Presolved: 180 rows, 420 columns, 1236 nonzeros
Variable types: 0 continuous, 420 integer (132 binar
y)
Found heuristic solution: objective 662.0000000

Root relaxation: objective 3.185667e+02, 254 iterati
```

In [112]:

```
# Use output to get path
# gets variables as (x,y) coordinates
variables = []
for v in result.getVars():
    if('x' in v.varName and v.x == 1):
        # print(v.name, " = ", v.varValue)
        temp = (v.varName.split('(')[1].split(')')[0].split(','))
        variables.append((int(temp[0]),int(temp[1])))
print(variables)

variablesR = {}
for v in result.getVars():
    if('R' in v.varName and v.x > 0):
        temp = (v.varName.split('(')[1].split(')')[0].split(','))
#         variablesR[(xCoordinates[int(temp[0])],yCoordinates[int(
        variablesR[(xCoordinates[int(temp[1])],yCoordinates[int(temp
print("variablesR", variablesR)
```

```
print("variablesR", variablesR)

variablesP = {}
for v in result.getVars():
    if('P' in v.varName and v.x > 0):
        # print(v.name, " = ", v.varValue)
        temp = (v.varName.split('(')[1].split(')')[0].split(','))
#         variablesP[(xCoordinates[int(temp[0])],yCoordinates[int(
        variablesP[(xCoordinates[int(temp[1])],yCoordinates[int(temp
print("variablesP",variablesP)

#recursive calls for getting the path
def recursiveList(start, L, X, c):
    if(start in c):
        return X
    for item in L:
        if(item[0] == start):
            X.append(item)
            c.append(start)
            return recursiveList(item[1], L, X, c)
    return X

pathList = []
setList = []
start = 0
for v in variables:
    if(v[0] == start):
        path = recursiveList(v[1], variables, [v], [start])
        print(path)
        pathList.append(path)
        set1 = []
        for i in path:
            set1.append(i[0])
            set1.append(i[1])
        a = list(set(set1))
        if(len(a)>1):
            setList.append(sorted(a))

print(setList)
```

```
[(0, 0), (0, 1), (0, 2), (0, 6), (0, 8), (1, 3), (2, 7
), (3, 12), (4, 11), (5, 10), (6, 0), (7, 4), (8, 5),
(9, 0), (10, 0), (11, 0), (12, 9)]
variablesR {(151, 164): 5600.0, (159, 261): 6000.0, (1
```

```
46, 246): 1400.0, (142, 239): 5200.0, (130, 254): 4400
.0, (161, 242): 4300.0, (156, 217): 2900.0, (128, 231)
: 1700.0, (148, 232): 1600.0, (128, 252): 3100.0, (163
, 247): 3300.0, (163, 236): 1800.0}
variablesP {(161, 242): 1200.0, (156, 217): 1700.0, (1
28, 231): 4100.0, (148, 232): 2600.0, (145, 215): 5700
.0, (128, 252): 2600.0, (163, 247): 1200.0, (163, 236)
: 3400.0}
[(0, 0)]
[(0, 1), (1, 3), (3, 12), (12, 9), (9, 0)]
[(0, 2), (2, 7), (7, 4), (4, 11), (11, 0)]
[(0, 6), (6, 0)]
[(0, 8), (8, 5), (5, 10), (10, 0)]
[[0, 1, 3, 9, 12], [0, 2, 4, 7, 11], [0, 6], [0, 5, 8,
10]]
```

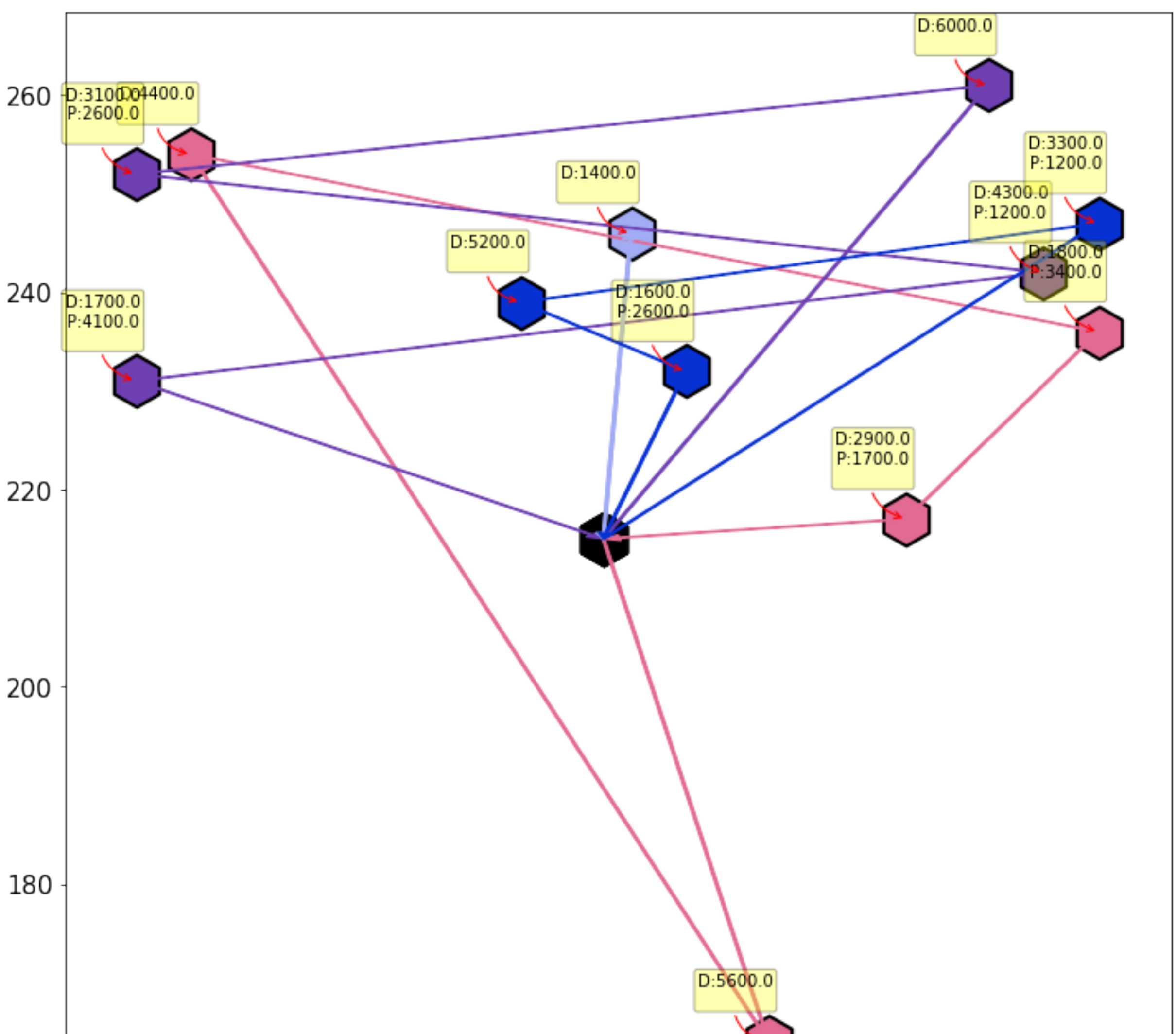
In [113]:

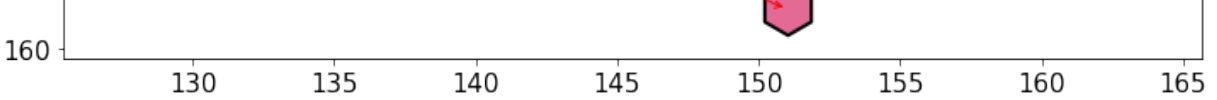
```
# Visualization
plt.figure(figsize=(12,12))
plt.rc('xtick',labelsize=15)
plt.rc('ytick',labelsize=15)

coordinateList = []
for s in setList:
    coordinate = []
    for j in s:
        coordinate.append([float(xCoordinates[j]), float(yCoordinate
    coordinateList.append(coordinate)
# print(coordinateList)

def addToPlot(L):
    x_val = [x[0] for x in L]
    y_val = [x[1] for x in L]

    r = random.random()
    b = random.random()
    g = random.random()
    newColor = (r, g, b)

    plt.scatter(x_val,y_val, c=newColor, edgecolor='black', linewid
    ax = plt.axes()

    length = len(L)-1
```

```
    for i in range(length):
        ax.arrow(L[i][0],  #x1
                 L[i][1],  # y1
                 L[i+1][0]-L[i][0], # x2 - x1
                 L[i+1][1]-L[i][1], # y2 - y1
                 width=0.1, head_width=0.6, head_length=0.6, colo
                 )
        aS = ""
        if((int(L[i+1][0]),int(L[i+1][1]))) in variablesR:
            aS += "D:" + str(variablesR[(int(L[i+1][0]),int(L[i+1][
        if((int(L[i+1][0]),int(L[i+1][1]))) in variablesP:
            aS += "P:" + str(variablesP[(int(L[i+1][0]),int(L[i+1][

        ax.annotate(aS, xy=(L[i+1][0], L[i+1][1]), xytext=(-20,20),
            textcoords='offset points', ha='center', va='bottom',
            bbox=dict(boxstyle='round,pad=0.2', fc='yellow', alpha=
            arrowprops=dict(arrowstyle='->', connectionstyle='arc3,
                            color='red'))

    ax.arrow(L[-1][0],  #x1
                 L[-1][1],  # y1
                 L[0][0]-L[-1][0], # x2 - x1
                 L[0][1]-L[-1][1], # y2 - y1
                 width=0.1, head_width=0.6, head_length=0.6, colo
                 )
for i in range(len(coordinateList)):
#     print(coordinateList[i])
    addToPlot(coordinateList[i])
plt.scatter(coordinateList[0][0][0],coordinateList[0][0][1], c='bla
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA s
equence, which should be avoided as value-mapping will
have precedence in case its length matches with 'x' &
'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for
all points.
/Users/shreyas/opt/anaconda3/lib/python3.7/site-packag
es/ipykernel_launcher.py:24: MatplotlibDeprecationWarn
ing: Adding an axes using the same arguments as a prev
ious axes currently reuses the earlier instance.  In a
future version, a new instance will always be created
and returned.  Meanwhile, this warning can be suppress
```

```
ed, and the future behavior ensured, by passing a uniq
ue label to each axes instance.
'c' argument looks like a single numeric RGB or RGBA s
equence, which should be avoided as value-mapping will
have precedence in case its length matches with 'x' &
'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for
all points.
'c' argument looks like a single numeric RGB or RGBA s
equence, which should be avoided as value-mapping will
have precedence in case its length matches with 'x' &
'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for
all points.
'c' argument looks like a single numeric RGB or RGBA s
equence, which should be avoided as value-mapping will
have precedence in case its length matches with 'x' &
'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for
all points.
```





E-N22-K4.vrp - Christofides and Eilon

In [94]:

```
# E-n22-k4.vrp = CE22P

xCoordinates = [145, 151, 159, 130, 128, 163, 146, 161, 142, 163, 14
yCoordinates = [215, 164, 261, 254, 252, 247, 246, 242, 239, 236, 2

costMatrix = np.ndarray(shape=(len(xCoordinates), len(yCoordinates)
for i in range(len(xCoordinates)):
    for j in range(len(yCoordinates)):
        costMatrix[i][j] = float(distance.euclidean([xCoordinates[i

demand = [0, 1100, 700, 800, 1400, 2100, 400, 800, 100, 500, 600, 1
pickup = [700, 0, 1100, 700, 800, 1400, 2100, 400, 800, 100, 500, 6
capacityOfVehicle = 6000
numberOfVehicles = 7
```

In [95]:

```
lp = VRPSDP_GUROBI(costMatrix, demand, pickup, numberOfVehicles, ca
```

In [96]:

```
lp.solve()
result = lp.getResult()
```

```
Gurobi Optimizer version 9.0.2 build v9.0.2rc0 (mac6
4)
Optimize a model with 591 rows, 1452 columns and 420
3 nonzeros
Model fingerprint: 0x75260d28
Variable types: 0 continuous, 1452 integer (484 bina
ry)
Coefficient statistics:
  Matrix range     [1e+00, 6e+03]
  Objective range  [3e+00, 1e+02]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 2e+03]
Presolve removed 87 rows and 150 columns
Presolve time: 0.09s
Presolved: 504 rows, 1302 columns, 3864 nonzeros
Variable types: 0 continuous, 1302 integer (420 bina
ry)
Found heuristic solution: objective 1106.2205425

Root relaxation: objective 2.343201e+02, 1078 iterat
```

In [102]:

```
# Use output to get path
# gets variables as (x,y) coordinates
variables = []
for v in result.getVars():
    if('x' in v.varName and v.x == 1):
        # print(v.name, " = ", v.varValue)
        temp = (v.varName.split('(')[1].split(')')[0].split(','))
        variables.append((int(temp[0]),int(temp[1])))
print(variables)

variablesR = {}
for v in result.getVars():
    if('R' in v.varName and v.x > 0):
        temp = (v.varName.split('(')[1].split(')')[0].split(','))
#         variablesR[(xCoordinates[int(temp[0])],yCoordinates[int(
        variablesR[(xCoordinates[int(temp[1])],yCoordinates[int(temp
print("variablesR", variablesR)
```

```
variablesP = {}
for v in result.getVars():
    if('P' in v.varName and v.x > 0):
        # print(v.name, " = ", v.varValue)
        temp = (v.varName.split('(')[1].split(')')[0].split(','))
#         variablesP[(xCoordinates[int(temp[0])],yCoordinates[int(
        variablesP[(xCoordinates[int(temp[1])],yCoordinates[int(temp
print("variablesP",variablesP)

#recursive calls for getting the path
def recursiveList(start, L, X, c):
    if(start in c):
        return X
    for item in L:
        if(item[0] == start):
            X.append(item)
            c.append(start)
            return recursiveList(item[1], L, X, c)
    return X

pathList = []
setList = []
start = 0
for v in variables:
    if(v[0] == start):
        path = recursiveList(v[1], variables, [v], [start])
        print(path)
        pathList.append(path)
        set1 = []
        for i in path:
            set1.append(i[0])
            set1.append(i[1])
        a = list(set(set1))
        if(len(a)>1):
            setList.append(sorted(a))

print(setList)
```

```
[(0, 0), (0, 9), (0, 12), (0, 13), (0, 19), (1, 20), (
2, 6), (3, 8), (4, 3), (5, 2), (6, 10), (7, 5), (8, 0)
, (9, 7), (10, 0), (11, 4), (12, 15), (13, 11), (14, 0
), (15, 17), (16, 14), (17, 16), (18, 0), (19, 21), (2
0, 18), (21, 1)]
```

```
variablesR {(163, 236): 5100.0, (156, 217): 5600.0, (1
29, 214): 4800.0, (129, 189): 6000.0, (155, 185): 1900
.0, (146, 246): 1000.0, (142, 239): 100.0, (130, 254):
900.0, (159, 261): 1700.0, (148, 232): 600.0, (163, 24
7): 3800.0, (161, 242): 4600.0, (128, 252): 2300.0, (1
64, 208): 4300.0, (128, 231): 3500.0, (147, 193): 3400
.0, (146, 208): 300.0, (141, 206): 2400.0, (139, 182):
3900.0, (164, 193): 900.0, (151, 164): 3000.0}
variablesP {(155, 185): 1900.0, (146, 246): 3000.0, (1
42, 239): 3400.0, (130, 254): 2700.0, (159, 261): 1900
.0, (148, 232): 5100.0, (163, 247): 500.0, (145, 215):
5000.0, (161, 242): 100.0, (128, 252): 1900.0, (164, 2
08): 1200.0, (128, 231): 1300.0, (147, 193): 1500.0, (
146, 208): 4500.0, (141, 206): 3600.0, (139, 182): 900
.0, (164, 193): 4000.0, (151, 164): 1900.0}
[(0, 0)]
[(0, 9), (9, 7), (7, 5), (5, 2), (2, 6), (6, 10), (10,
0)]
[(0, 12), (12, 15), (15, 17), (17, 16), (16, 14), (14,
0)]
[(0, 13), (13, 11), (11, 4), (4, 3), (3, 8), (8, 0)]
[(0, 19), (19, 21), (21, 1), (1, 20), (20, 18), (18, 0
)]
[[0, 2, 5, 6, 7, 9, 10], [0, 12, 14, 15, 16, 17], [0,
3, 4, 8, 11, 13], [0, 1, 18, 19, 20, 21]]
```

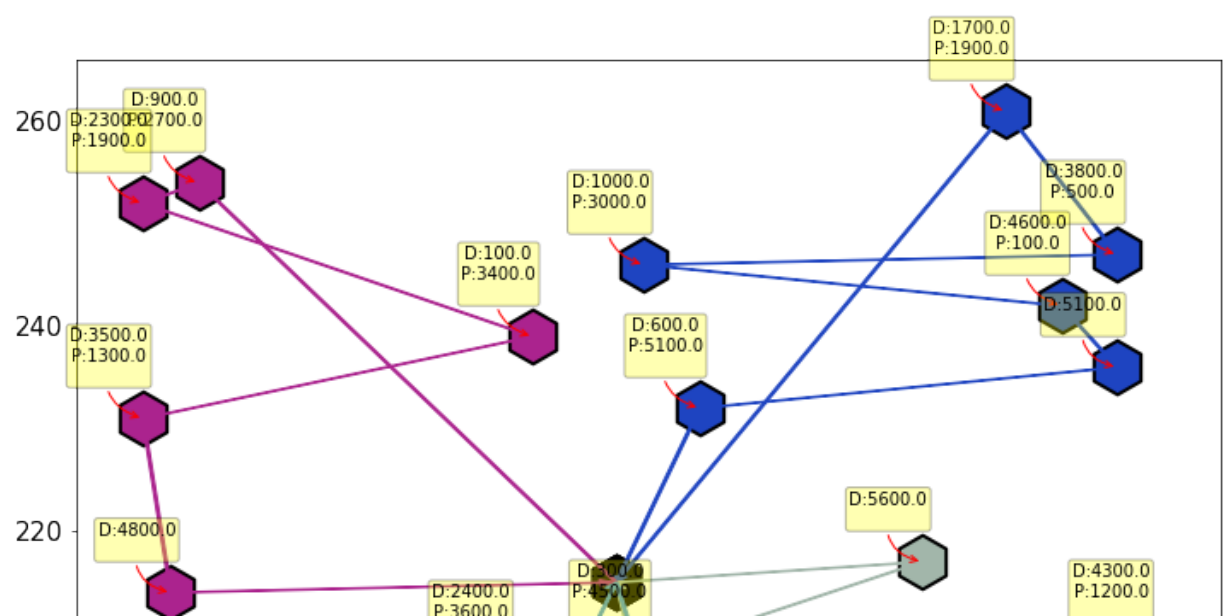
In [107]:

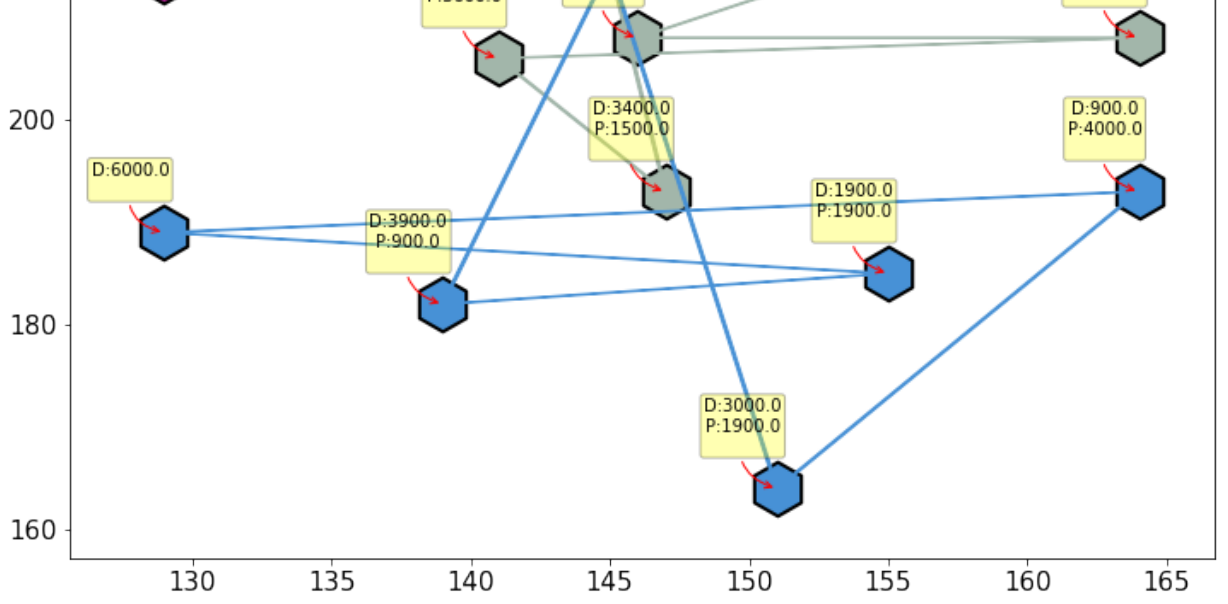
```
# Visualization
plt.figure(figsize=(12,12))
plt.rc('xtick',labelsize=15)
plt.rc('ytick',labelsize=15)

coordinateList = []
for s in setList:
    coordinate = []
    for j in s:
        coordinate.append([float(xCoordinates[j]), float(yCoordinate
    coordinateList.append(coordinate)
# print(coordinateList)

def addToPlot(L):
    x_val = [x[0] for x in L]
    y_val = [x[1] for x in L]
```

```
    r = random.random()
    b = random.random()
    g = random.random()
    newColor = (r, g, b)

    plt.scatter(x_val,y_val, c=newColor, edgecolor='black', linewid
    ax = plt.axes()

    length = len(L)-1

    for i in range(length):
        ax.arrow(L[i][0],  #x1
                 L[i][1],  # y1
                 L[i+1][0]-L[i][0], # x2 - x1
                 L[i+1][1]-L[i][1], # y2 - y1
                 width=0.1, head_width=0.6, head_length=0.6, col
                 )
        aS = ""
        if((int(L[i+1][0]),int(L[i+1][1]))) in variablesR:
            aS += "D:" + str(variablesR[(int(L[i+1][0]),int(L[i+1][
        if((int(L[i+1][0]),int(L[i+1][1]))) in variablesP:
            aS += "P:" + str(variablesP[(int(L[i+1][0]),int(L[i+1][

        ax.annotate(aS, xy=(L[i+1][0], L[i+1][1]), xytext=(-20,20),
            textcoords='offset points', ha='center', va='bottom',
            bbox=dict(boxstyle='round,pad=0.2', fc='yellow', alpha=
            arrowprops=dict(arrowstyle='->', connectionstyle='arc3,
                            color='red'))

    ax.arrow(L[-1][0],  #x1
                 L[-1][1],  # y1
                 L[0][0]-L[-1][0], # x2 - x1
                 L[0][1]-L[-1][1], # y2 - y1
                 width=0.1, head_width=0.6, head_length=0.6, col
                 )
for i in range(len(coordinateList)):
#     print(coordinateList[i])
    addToPlot(coordinateList[i])
plt.scatter(coordinateList[0][0][0],coordinateList[0][0][1], c='bla
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA s
equence, which should be avoided as value-mapping will
```

```
have precedence in case its length matches with 'x' &
'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for
all points.
/Users/shreyas/opt/anaconda3/lib/python3.7/site-packag
es/ipykernel_launcher.py:24: MatplotlibDeprecationWarn
ing: Adding an axes using the same arguments as a prev
ious axes currently reuses the earlier instance.  In a
future version, a new instance will always be created
and returned.  Meanwhile, this warning can be suppress
ed, and the future behavior ensured, by passing a uniq
ue label to each axes instance.
'c' argument looks like a single numeric RGB or RGBA s
equence, which should be avoided as value-mapping will
have precedence in case its length matches with 'x' &
'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for
all points.
'c' argument looks like a single numeric RGB or RGBA s
equence, which should be avoided as value-mapping will
have precedence in case its length matches with 'x' &
'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for
all points.
'c' argument looks like a single numeric RGB or RGBA s
equence, which should be avoided as value-mapping will
have precedence in case its length matches with 'x' &
'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for
all points.
```





E-N30-K4.vrp - Christofides and Eilon

In [118]:

```
xCoordinates = [162, 218, 218, 201, 214, 224, 210, 104, 126, 119, 1
yCoordinates = [354, 382, 358, 370, 371, 370, 382, 354, 338, 340, 3

costMatrix = np.ndarray(shape=(len(xCoordinates), len(yCoordinates)
for i in range(len(xCoordinates)):
    for j in range(len(yCoordinates)):
        costMatrix[i][j] = float(distance.euclidean([xCoordinates[i

demand = [0, 300, 3100, 125, 100, 200, 150, 150, 450, 300, 100, 950
pickup = [1000, 0, 300, 3100, 125, 100, 200, 150, 150, 450, 300, 10
capacityOfVehicle = 4500
numberOfVehicles = 4
```

In [119]:

```
lp = VRPSDP_GUROBI(costMatrix, demand, pickup, numberOfVehicles, ca
lp.solve()
result = lp.getResult()
```

```
Gurobi Optimizer version 9.0.2 build v9.0.2rc0 (mac6
4)
Optimize a model with 1047 rows, 2700 columns and 78
91 nonzeros
Model fingerprint: 0xf3fe0944
Variable types: 0 continuous, 2700 integer (900 bina
ry)
Coefficient statistics:
  Matrix range     [1e+00, 4e+03]
  Objective range  [1e+00, 1e+02]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+03]
Presolve removed 122 rows and 215 columns
Presolve time: 0.04s
Presolved: 925 rows, 2485 columns, 7397 nonzeros
Variable types: 0 continuous, 2485 integer (809 bina
ry)
Found heuristic solution: objective 2536.8168000

Root relaxation: objective 4.248668e+02, 2861 iterat
```

In [120]:

```
# Use output to get path
# gets variables as (x,y) coordinates
variables = []
for v in result.getVars():
    if('x' in v.varName and v.x == 1):
        # print(v.name, " = ", v.varValue)
        temp = (v.varName.split('(')[1].split(')')[0].split(','))
        variables.append((int(temp[0]),int(temp[1])))
print(variables)

variablesR = {}
for v in result.getVars():
    if('R' in v.varName and v.x > 0):
        temp = (v.varName.split('(')[1].split(')')[0].split(','))
#         variablesR[(xCoordinates[int(temp[0])],yCoordinates[int(
        variablesR[(xCoordinates[int(temp[1])],yCoordinates[int(temp
print("variablesR", variablesR)
```

```
print("variablesR", variablesR)

variablesP = {}
for v in result.getVars():
    if('P' in v.varName and v.x > 0):
        # print(v.name, " = ", v.varValue)
        temp = (v.varName.split('(')[1].split(')')[0].split(','))
#         variablesP[(xCoordinates[int(temp[0])],yCoordinates[int(
        variablesP[(xCoordinates[int(temp[1])],yCoordinates[int(temp
print("variablesP",variablesP)

#recursive calls for getting the path
def recursiveList(start, L, X, c):
    if(start in c):
        return X
    for item in L:
        if(item[0] == start):
            X.append(item)
            c.append(start)
            return recursiveList(item[1], L, X, c)
    return X

pathList = []
setList = []
start = 0
for v in variables:
    if(v[0] == start):
        path = recursiveList(v[1], variables, [v], [start])
        print(path)
        pathList.append(path)
        set1 = []
        for i in path:
            set1.append(i[0])
            set1.append(i[1])
        a = list(set(set1))
        if(len(a)>1):
            setList.append(sorted(a))

print(setList)
```

```
[(0, 0), (0, 15), (0, 20), (0, 21), (0, 26), (1, 6), (
2, 5), (3, 19), (4, 3), (5, 4), (6, 22), (7, 17), (8,
12), (9, 14), (10, 23), (11, 10), (12, 11), (13, 7), (
14, 8), (15, 16), (16, 13), (17, 9), (18, 0), (19, 0),
```

```
(20, 2), (21, 0), (22, 0), (23, 18), (24, 1), (25, 24)
, (26, 28), (27, 29), (28, 27), (29, 25)]
variablesR {(125, 355): 3625.0, (180, 360): 4225.0, (1
59, 331): 1500.0, (188, 393): 3400.0, (210, 382): 250.
0, (224, 370): 825.0, (175, 363): 400.0, (201, 370): 5
25.0, (214, 371): 625.0, (188, 357): 100.0, (115, 341)
: 2625.0, (125, 346): 1625.0, (126, 335): 2225.0, (152
, 349): 450.0, (129, 349): 550.0, (126, 347): 1500.0,
(104, 354): 2775.0, (126, 338): 2075.0, (119, 357): 30
75.0, (116, 355): 2925.0, (119, 340): 2525.0, (218, 35
8): 3925.0, (153, 351): 150.0, (218, 382): 550.0, (215
, 389): 1050.0, (184, 410): 3100.0, (207, 392): 2850.0
, (207, 406): 2950.0, (212, 394): 1850.0}
variablesP {(210, 382): 2150.0, (224, 370): 700.0, (17
5, 363): 4025.0, (201, 370): 925.0, (214, 371): 800.0,
(188, 357): 2350.0, (115, 341): 975.0, (125, 346): 187
5.0, (126, 335): 1575.0, (152, 349): 3225.0, (129, 349
): 2925.0, (126, 347): 2825.0, (104, 354): 825.0, (126
, 338): 1725.0, (119, 357): 150.0, (116, 355): 700.0,
(119, 340): 1125.0, (162, 354): 3850.0, (218, 358): 40
0.0, (153, 351): 3325.0, (218, 382): 2150.0, (215, 389
): 1850.0, (184, 410): 800.0, (207, 392): 1200.0, (207
, 406): 900.0, (212, 394): 1350.0}
[(0, 0)]
[(0, 15), (15, 16), (16, 13), (13, 7), (7, 17), (17, 9
), (9, 14), (14, 8), (8, 12), (12, 11), (11, 10), (10,
23), (23, 18), (18, 0)]
[(0, 20), (20, 2), (2, 5), (5, 4), (4, 3), (3, 19), (1
9, 0)]
[(0, 21), (21, 0)]
[(0, 26), (26, 28), (28, 27), (27, 29), (29, 25), (25,
24), (24, 1), (1, 6), (6, 22), (22, 0)]
[[0, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 23],
[0, 2, 3, 4, 5, 19, 20], [0, 21], [0, 1, 6, 22, 24, 25
, 26, 27, 28, 29]]
```

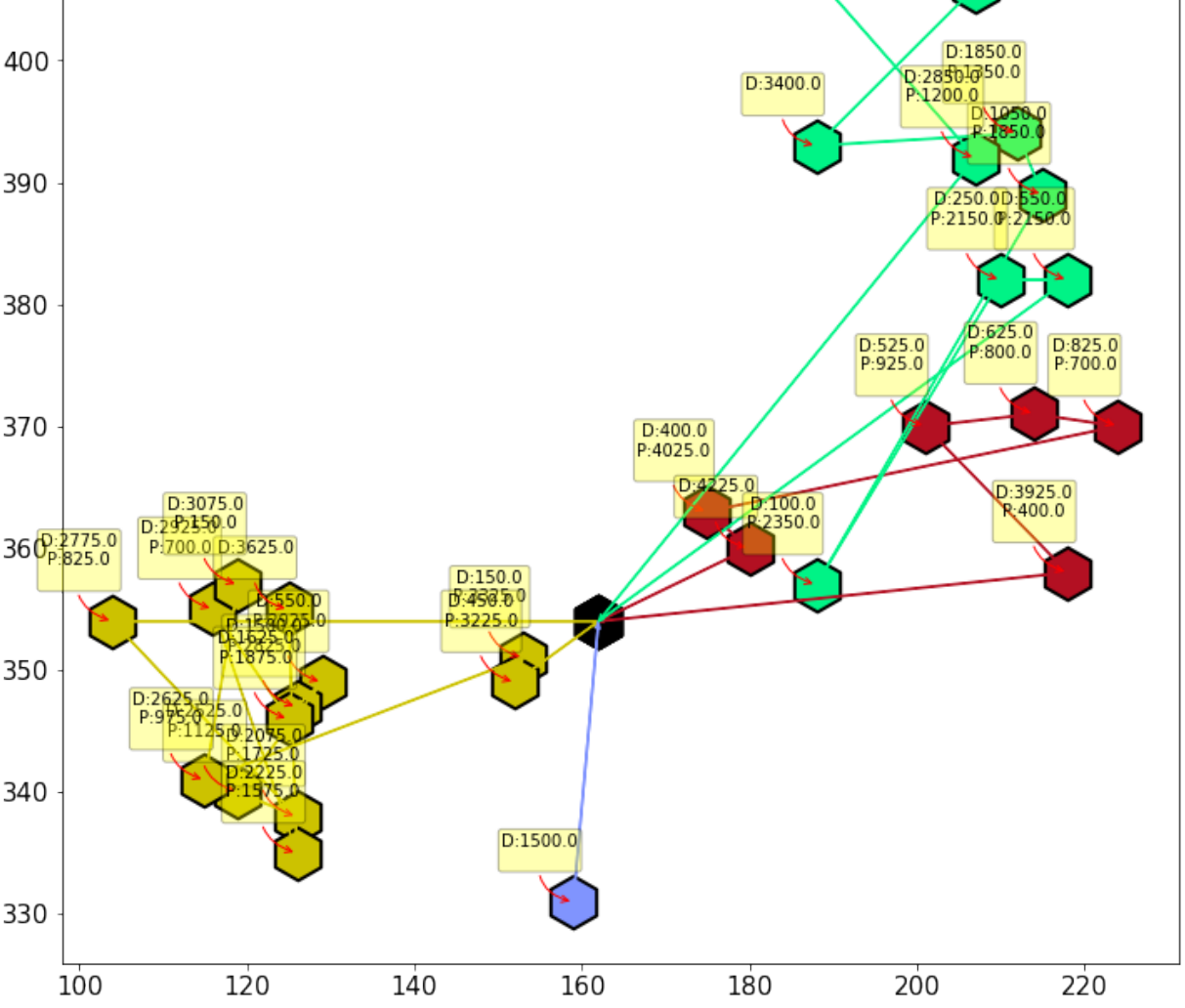
In [124]:

```
# Visualization
plt.figure(figsize=(12,12))
plt.rc('xtick',labelsize=15)
plt.rc('ytick',labelsize=15)

coordinateList = []
```

```
for s in setList:
    coordinate = []
    for j in s:
        coordinate.append([float(xCoordinates[j]), float(yCoordinate
    coordinateList.append(coordinate)
# print(coordinateList)

def addToPlot(L):
    x_val = [x[0] for x in L]
    y_val = [x[1] for x in L]

    r = random.random()
    b = random.random()
    g = random.random()
    newColor = (r, g, b)

    plt.scatter(x_val,y_val, c=newColor, edgecolor='black', linewid
    ax = plt.axes()

    length = len(L)-1

    for i in range(length):
        ax.arrow(L[i][0],  #x1
                 L[i][1],  # y1
                 L[i+1][0]-L[i][0], # x2 - x1
                 L[i+1][1]-L[i][1], # y2 - y1
                 width=0.1, head_width=0.6, head_length=0.6, col
                 )
        aS = ""
        if((int(L[i+1][0]),int(L[i+1][1]))) in variablesR:
            aS += "D:" + str(variablesR[(int(L[i+1][0]),int(L[i+1][
        if((int(L[i+1][0]),int(L[i+1][1]))) in variablesP:
            aS += "P:" + str(variablesP[(int(L[i+1][0]),int(L[i+1][

        ax.annotate(aS, xy=(L[i+1][0], L[i+1][1]), xytext=(-20,20),
            textcoords='offset points', ha='center', va='bottom',
            bbox=dict(boxstyle='round,pad=0.2', fc='yellow', alpha=
            arrowprops=dict(arrowstyle='->', connectionstyle='arc3,
                            color='red'))

    ax.arrow(L[-1][0],  #x1
                 L[-1][1],  # y1
                 L[0][0]-L[-1][0], # x2 - x1
                 L[0][1]-L[-1][1], # y2 - y1
                 width=0.1, head_width=0.6, head_length=0.6, col
```

```
                    )
for i in range(len(coordinateList)):
#     print(coordinateList[i])
    addToPlot(coordinateList[i])
plt.scatter(coordinateList[0][0][0],coordinateList[0][0][1], c='bla
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA s
equence, which should be avoided as value-mapping will
have precedence in case its length matches with 'x' &
'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for
all points.
/Users/shreyas/opt/anaconda3/lib/python3.7/site-packag
es/ipykernel_launcher.py:24: MatplotlibDeprecationWarn
ing: Adding an axes using the same arguments as a prev
ious axes currently reuses the earlier instance.  In a
future version, a new instance will always be created
and returned.  Meanwhile, this warning can be suppress
ed, and the future behavior ensured, by passing a uniq
ue label to each axes instance.
'c' argument looks like a single numeric RGB or RGBA s
equence, which should be avoided as value-mapping will
have precedence in case its length matches with 'x' &
'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for
all points.
'c' argument looks like a single numeric RGB or RGBA s
equence, which should be avoided as value-mapping will
have precedence in case its length matches with 'x' &
'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for
all points.
'c' argument looks like a single numeric RGB or RGBA s
equence, which should be avoided as value-mapping will
have precedence in case its length matches with 'x' &
'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for
all points.
```


D:3400.0
D:1850.0
D:2850.0
P:1200.0
D:250.0
P:2150.0
D:550.0
P:2150.0
D:525.0
P:925.0
D:625.0
P:800.0
D:825.0
P:700.0
D:400.0
P:4025.0
D:4225.0
D:100.0
P:2350.0
D:3925.0
P:400.0
D:3075.0
D:2775.0
P:825.0
D:3625.0
D:150.0
P:3225.0
D:2625.0
D:2075.0
P:1725.0
D:2225.0
P:1575.0
D:1500.0
400
390
380
370
360
350
340
330
100
120
140
160
180
200
220

In [ ]:

CSCI 5654 - Linear Programming - Project

Team Members

1. Ketan Ramesh

2. Shreyas Gopalakrishna

Vehicle Routing Problem with Simultaneous Pickup and Delivery

In [1]:

```
Requirement already satisfied: pulp in /usr/local/lib/
python3.6/dist-packages (2.1)
Requirement already satisfied: pyparsing>=2.0.1 in /us
r/local/lib/python3.6/dist-packages (from pulp) (2.4.7
)
```

In [1]:

Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSDP)

Paper Link - Vehicle Routing Problem with Deliveries and Pickups: Modelling Issues and Meta-heuristics Solution Approaches (https://core.ac.uk/download/pdf/19477982.pdf)

The code provided below creates a `class` to frame and solve the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSDP) using the two-index flow formulation as mentioned in the paper. The formulation of the problem as a Mixed ILP is as follows:

$$min \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} x_{ij}$$

$$s.t. \sum_{i=0}^{n} x_{ij} = 1, j \in \{1, \ldots, n\}, \ -(3.1)$$

$$\sum_{i=0}^{n} x_{ji} = 1, j \in \{1, \ldots, n\}, \ -(3.2)$$

$$\sum_{i=0}^{n} R_{ij} - q_j = \sum_{i=0}^{n} R_{ji}, j \in \{1, \ldots, n\}, \ -(3.3)$$

$$\sum_{i=0}^{n} P_{ij} + b_j = \sum_{i=0}^{n} P_{ji}, j \in \{1, \ldots, n\}, \ -(3.4)$$

$$\sum_{i=1}^{n} R_{i0} = 0, \ -(3.5)$$

$$\sum_{i=1}^{n} P_{0i} = 0, \ -(3.6)$$

$$R_{ij} + P_{ij} \leq Cx_{ij}, i,j \in \{0, \ldots, n\}, \ -(3.7)$$

$$x_{ij} = \{0, 1\}, i,j \in \{1, \ldots n\}$$

$$R_{ij}, P_{ij} \geq 0, i,j \in \{1, \ldots, n\}$$

where,

d_{ij} = Cost to travel from node i to j.

q_i = The delivery demand of node i.

b_i = The pickup demand of node i.

C = Vehicle capacity.

Decision Variables:

1. x_{ij} = {**1**, if edge (i, j) present in any route. **0**, otherwise}.
2. R_{ij} = The amount of delivery goods on board on arc ij.
3. P_{ij} = The amount of pickup goods on board on arc ij.

The objective is to minimize the sum of the distances of all routes that satisfy the given constraints. The constraints (3.1) and (3.2) ensure that every node is visited exactly once (every node leads to only one other node, every node must be visited by only one other node). The constraints (3.3) and (3.4) ensure that the flow conservation is met (at a node *j*, the amount of delivery load after servicing delivery of node *j*, must be equal to pickup load at the same node. The pickup load condition is similar in nature). The constraints (3.5) and (3.6) ensure that the depot has zero pickup and delivery load. The constraint (3.7) ensures that the pickup and demand loads do not exceed the capacity of the vehicle

at all nodes in the routes.

In [84]:

```
```

E-N13-K4.vrp - Christofides and Eilon

In [109]:

```
```

Out[109]:

13

In [110]:

```
```

In [111]:

```
Gurobi Optimizer version 9.0.2 build v9.0.2rc0 (mac6
4)
Optimize a model with 231 rows, 507 columns and 1431
nonzeros
Model fingerprint: 0xb8164ab9
Variable types: 0 continuous, 507 integer (169 binar
y)
Coefficient statistics:
  Matrix range     [1e+00, 6e+03]
  Objective range  [5e+00, 5e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 2e+03]
Presolve removed 51 rows and 87 columns
Presolve time: 0.01s
Presolved: 180 rows, 420 columns, 1236 nonzeros
Variable types: 0 continuous, 420 integer (132 binar
y)
Found heuristic solution: objective 662.0000000

Root relaxation: objective 2.185667e+02, 254 iterati
```

In [112]:

```
[(0, 0), (0, 1), (0, 2), (0, 6), (0, 8), (1, 3), (2,
7), (3, 12), (4, 11), (5, 10), (6, 0), (7, 4), (8, 5
), (9, 0), (10, 0), (11, 0), (12, 9)]
variablesR {(151, 164): 5600.0, (159, 261): 6000.0,
(146, 246): 1400.0, (142, 239): 5200.0, (130, 254):
4400.0, (161, 242): 4300.0, (156, 217): 2900.0, (128
, 231): 1700.0, (148, 232): 1600.0, (128, 252): 3100
.0, (163, 247): 3300.0, (163, 236): 1800.0}
variablesP {(161, 242): 1200.0, (156, 217): 1700.0,
(128, 231): 4100.0, (148, 232): 2600.0, (145, 215):
5700.0, (128, 252): 2600.0, (163, 247): 1200.0, (163
, 236): 3400.0}
[(0, 0)]
[(0, 1), (1, 3), (3, 12), (12, 9), (9, 0)]
[(0, 2), (2, 7), (7, 4), (4, 11), (11, 0)]
[(0, 6), (6, 0)]
[(0, 8), (8, 5), (5, 10), (10, 0)]
[[0, 1, 3, 9, 12], [0, 2, 4, 7, 11], [0, 6], [0, 5,
8, 10]]
```

In [113]:

```
'c' argument looks like a single numeric RGB or RGBA
sequence, which should be avoided as value-mapping w
ill have precedence in case its length matches with
'x' & 'y'.  Please use a 2-D array with a single row
if you really want to specify the same RGB or RGBA v
alue for all points.
/Users/shreyas/opt/anaconda3/lib/python3.7/site-pack
ages/ipykernel_launcher.py:24: MatplotlibDeprecation
Warning: Adding an axes using the same arguments as
a previous axes currently reuses the earlier instanc
e.  In a future version, a new instance will always
be created and returned.  Meanwhile, this warning ca
n be suppressed, and the future behavior ensured, by
passing a unique label to each axes instance.
'c' argument looks like a single numeric RGB or RGBA
sequence, which should be avoided as value-mapping w
ill have precedence in case its length matches with
'x' & 'y'.  Please use a 2-D array with a single row
if you really want to specify the same RGB or RGBA v
alue for all points.
```

E-N22-K4.vrp - Christofides and Eilon

In [94]:

In [95]:

In [96]:

```
Gurobi Optimizer version 9.0.2 build v9.0.2rc0 (mac6
4)
Optimize a model with 591 rows, 1452 columns and 420
3 nonzeros
Model fingerprint: 0x75260d28
Variable types: 0 continuous, 1452 integer (484 bina
ry)
Coefficient statistics:
  Matrix range     [1e+00, 6e+03]
  Objective range  [3e+00, 1e+02]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 2e+03]
Presolve removed 87 rows and 150 columns
Presolve time: 0.09s
Presolved: 504 rows, 1302 columns, 3864 nonzeros
Variable types: 0 continuous, 1302 integer (420 bina
ry)
Found heuristic solution: objective 1106.2205425

Root relaxation: objective 3.242301e+02, 1078 iterat
```

In [102]:

```
[(0, 0), (0, 9), (0, 12), (0, 13), (0, 19), (1, 20),
(2, 6), (3, 8), (4, 3), (5, 2), (6, 10), (7, 5), (8,
0), (9, 7), (10, 0), (11, 4), (12, 15), (13, 11), (1
4, 0), (15, 17), (16, 14), (17, 16), (18, 0), (19, 2
1), (20, 18), (21, 1)]
variablesR {(163, 236): 5100.0, (156, 217): 5600.0,
(129, 214): 4800.0, (129, 189): 6000.0, (155, 185):
1900.0, (146, 246): 1000.0, (142, 239): 100.0, (130,
254): 900.0, (159, 261): 1700.0, (148, 232): 600.0,
(163, 247): 3800.0, (161, 242): 4600.0, (128, 252):
2300.0, (164, 208): 4300.0, (128, 231): 3500.0, (147
, 193): 3400.0, (146, 208): 300.0, (141, 206): 2400.
0, (139, 182): 3900.0, (164, 193): 900.0, (151, 164)
: 3000.0}
variablesP {(155, 185): 1900.0, (146, 246): 3000.0,
(142, 239): 3400.0, (130, 254): 2700.0, (159, 261):
1900.0, (148, 232): 5100.0, (163, 247): 500.0, (145,
215): 5000.0, (161, 242): 100.0, (128, 252): 1900.0,
(164, 208): 1200.0, (128, 231): 1300.0, (147, 193):
```

In [107]:

```
'c' argument looks like a single numeric RGB or RGBA
sequence, which should be avoided as value-mapping w
ill have precedence in case its length matches with
'x' & 'y'.  Please use a 2-D array with a single row
if you really want to specify the same RGB or RGBA v
alue for all points.
/Users/shreyas/opt/anaconda3/lib/python3.7/site-pack
ages/ipykernel_launcher.py:24: MatplotlibDeprecation
Warning: Adding an axes using the same arguments as
a previous axes currently reuses the earlier instanc
e.  In a future version, a new instance will always
be created and returned.  Meanwhile, this warning ca
n be suppressed, and the future behavior ensured, by
passing a unique label to each axes instance.
'c' argument looks like a single numeric RGB or RGBA
sequence, which should be avoided as value-mapping w
ill have precedence in case its length matches with
'x' & 'y'.  Please use a 2-D array with a single row
if you really want to specify the same RGB or RGBA v
alue for all points.
```

E-N30-K4.vrp - Christofides and Eilon

In [118]:

In [119]:

```
Gurobi Optimizer version 9.0.2 build v9.0.2rc0 (mac6
4)
Optimize a model with 1047 rows, 2700 columns and 78
91 nonzeros
Model fingerprint: 0xf3fe0944
Variable types: 0 continuous, 2700 integer (900 bina
ry)
Coefficient statistics:
  Matrix range     [1e+00, 4e+03]
  Objective range  [1e+00, 1e+02]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+03]
Presolve removed 122 rows and 215 columns
Presolve time: 0.04s
Presolved: 925 rows, 2485 columns, 7397 nonzeros
Variable types: 0 continuous, 2485 integer (809 bina
ry)
Found heuristic solution: objective 2536.8168000

Root relaxation: objective 4.248668e+02, 2861 iterat
```

In [120]:

```
[(0, 0), (0, 15), (0, 20), (0, 21), (0, 26), (1, 6),
(2, 5), (3, 19), (4, 3), (5, 4), (6, 22), (7, 17), (
8, 12), (9, 14), (10, 23), (11, 10), (12, 11), (13,
7), (14, 8), (15, 16), (16, 13), (17, 9), (18, 0), (
19, 0), (20, 2), (21, 0), (22, 0), (23, 18), (24, 1)
, (25, 24), (26, 28), (27, 29), (28, 27), (29, 25)]
variablesR {(125, 355): 3625.0, (180, 360): 4225.0,
(159, 331): 1500.0, (188, 393): 3400.0, (210, 382):
250.0, (224, 370): 825.0, (175, 363): 400.0, (201, 3
70): 525.0, (214, 371): 625.0, (188, 357): 100.0, (1
15, 341): 2625.0, (125, 346): 1625.0, (126, 335): 22
25.0, (152, 349): 450.0, (129, 349): 550.0, (126, 34
7): 1500.0, (104, 354): 2775.0, (126, 338): 2075.0,
(119, 357): 3075.0, (116, 355): 2925.0, (119, 340):
2525.0, (218, 358): 3925.0, (153, 351): 150.0, (218,
382): 550.0, (215, 389): 1050.0, (184, 410): 3100.0,
(207, 392): 2850.0, (207, 406): 2950.0, (212, 394):
1850.0}
variablesP {(210, 382): 2150.0, (224, 370): 700.0, (
```

In [124]:

```
'c' argument looks like a single numeric RGB or RGBA
sequence, which should be avoided as value-mapping w
ill have precedence in case its length matches with
'x' & 'y'.  Please use a 2-D array with a single row
if you really want to specify the same RGB or RGBA v
alue for all points.
/Users/shreyas/opt/anaconda3/lib/python3.7/site-pack
ages/ipykernel_launcher.py:24: MatplotlibDeprecation
Warning: Adding an axes using the same arguments as
a previous axes currently reuses the earlier instanc
e.  In a future version, a new instance will always
be created and returned.  Meanwhile, this warning ca
n be suppressed, and the future behavior ensured, by
passing a unique label to each axes instance.
'c' argument looks like a single numeric RGB or RGBA
sequence, which should be avoided as value-mapping w
ill have precedence in case its length matches with
'x' & 'y'.  Please use a 2-D array with a single row
if you really want to specify the same RGB or RGBA v
alue for all points.
```

In [ ]: