import time

from itertools import cycle

import numpy as np

from scipy.spatial.distance import pdist, squareform

import matplotlib.pyplot as plt

import matplotlib as mpl

import networkx as nx

import pyomo.environ as pyo

from pyomo.contrib.appsi.solvers.highs import Highs

np.random.seed(42) # Results should be always the same

N = 10

demands = np.random.randint(1, 10, size=N)

demands[0] = 0

capacity = 15

n\_vehicles = 4

coordinates = np.random.rand(N, 2)

distances = squareform(pdist(coordinates, metric="euclidean"))

distances = np.round(distances, decimals=4) # avoid numerical errors

model = pyo.ConcreteModel()

model.V = pyo.Set(initialize=range(len(demands)))

model.A = pyo.Set(initialize=[(i, j) for i in model.V for j in model.V if i != j])

model.K = pyo.Set(initialize=range(n\_vehicles))

model.Q = pyo.Param(initialize=capacity)

model.q = pyo.Param(model.V, initialize={i: d for (i, d) in enumerate(demands)})

model.c = pyo.Param(model.A, initialize={(i, j): distances[i, j] for (i, j) in model.A})

model.x = pyo.Var(model.A, model.K, within=pyo.Binary)

model.y = pyo.Var(model.V, model.K, within=pyo.Binary)

model.obj = pyo.Objective(

expr=sum(

model.x[i, j, k] \* model.c[i, j]

for (i, j) in model.A

for k in model.K

),

sense=pyo.minimize,

)

def arcs\_in(model, i):

if i == model.V.first():

return sum(model.x[:, i, :]) == len(model.K)

else:

return sum(model.x[:, i, :]) == 1.0

def arcs\_out(model, i):

if i == model.V.first():

return sum(model.x[i, :, :]) == len(model.K)

else:

return sum(model.x[i, :, :]) == 1.0

def vehicle\_assignment(model, i, k):

return sum(model.x[:, i, k]) == model.y[i, k]

def comp\_vehicle\_assignment(model, i, k):

return sum(model.x[i, :, k]) == model.y[i, k]

def capacity\_constraint(model, k):

return sum(model.y[i, k] \* model.q[i] for i in model.V) <= model.Q

model.arcs\_in = pyo.Constraint(model.V, rule=arcs\_in)

model.arcs\_out = pyo.Constraint(model.V, rule=arcs\_out)

model.vehicle\_assignment = pyo.Constraint(model.V, model.K, rule=vehicle\_assignment)

model.comp\_vehicle\_assignment = pyo.Constraint(model.V, model.K, rule=comp\_vehicle\_assignment)

model.capacity\_constraint = pyo.Constraint(model.K, rule=capacity\_constraint)