3-dim Tabu Search + ML Algorithm to Solve 2E-CVRPTW with Fuzzy Demand

1. Calculate the expected damands using K-NN and Initialization

updated\_demand <- mean\_demand

Training data <- Coordinates of customers

Labeling data <- updated\_demand

Training Using K-NN

expected\_demand <- predicted value

define penalty function with time window

initial\_solution <- [route1, route2] (each element of route1 and route2 is also list)

for each customer c:

find the suitable satellite s.

route2[s].append(c)

calculate required amount R of goods in each satellite.

For each satellite s:

N\_require\_urban <- int(R[s] / Q1) + 1

route1 <- [-1] \* N\_require\_urban

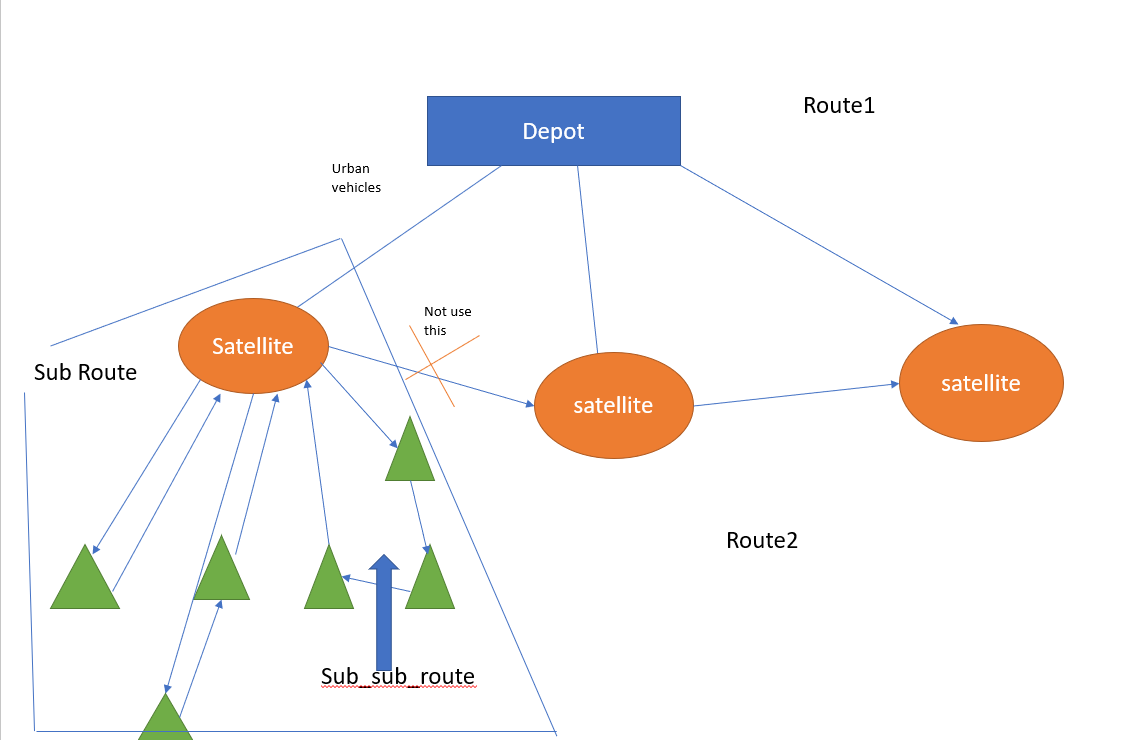
1. Implement the tabu search algorithm in sub route2

*It means, first we optimize the each sub route of each satellite in route2.  
the sameple of sub route of each satellite is [[0,1,4], [2,3], [6], [5,7]].*

*In the sample sub route, we used 4 city freighters and each element of sub route is the route of each frighters. (See the below figure)*

*According to the Bell Equation:*

*Min (route) =*



* Implement Tabu search algorithm in sub\_sub\_route
* Implement Tabu search algorithm in sub\_route

1. Optimize the Total Route using Tabu Genetic Algorithm

route1 = G(route2)

best\_solution = [route1, route2]  
best\_cost = calculate\_optimized\_cost (route2) (Implement Tabu Search)

bestcandidate <- [route1, route2]

tabuList ← []

tabuList.push([route1, route2])

while (not stoppingCondition())

sNeighborhood ← getNeighbors(bestCandidate)

bestCandidate ← sNeighborhood[0]

for (sCandidate in sNeighborhood)

if ( (not tabuList.contains(sCandidate)) and (calculate\_optimized\_cost(sCandidate) > calculate\_optimized\_cost(bestCandidate)) )

bestCandidate ← sCandidate

end

end

if (calculate\_optimized\_cost(bestCandidate) > best\_cost)

best\_solution ← bestCandidate

end

tabuList.push(bestCandidate)

if (tabuList.size > maxTabuSize)

tabuList.removeFirst()

end

end

return best\_solution

if expected\_demand < real\_demand:

best\_solution <- best\_solution with max\_demand

return best\_solution