Inter IIT Tech Meet

BOSCH Route Optimisation Algorithm

Report

IIT DHARWAD

Team members:

1.Abhishek Raj

2.HarshRaj

3.Janhavikumari Dadhania

4.Rupesh Kalantre

5.Ashish Kupsad

Visualisation and Formation of Problem Statement

Aim of the algorithm is to design route/routes for given pickup points and one common drop point minimizing total cost and considering other constraints. Boarding points of all passengers are pickup points. Final routes of buses will start from one of the pickup points and will finally reach BOSCH bidadi covering all the pickup points.

Routing with Time Window :

In case every person has specific time window in which she/he will board the bus routing should strictly adhere to it. For example if person B1 boarding from location 1 has time window A to B then bus should not reach point 1 after B. Bus can reach point 1

before A but it should wait till time A for person B1.

Approach

Algorithm gives route or set of routes for buses based at one drop point and one source (here according to sample data drop point is BOSCH bidadi and bus starts from point which gives most optimal solution from the set of input) for number of geographically dispersed boarding points (here in sample data 29 boarding points are given).

For pickup route bus/buses starts from source point and covers all boarding points considering the given constraints (maximum distance per day per bus is 92 kilometers and bus capacity is 32) and finally reach BOSCH bidadi. Source point from which bus should start will be given by algorithm as output along with list of points that bus/buses should follow. And for drop route buses will follow the same path.

CONSTRAINTS:

[1] & [2] By default algorithm minimizes operational cost (operational cost is directly proportional to total distance covered by all buses) considering total capacity of buses and km/day allowed for one bus.

[3] Route with time windows :

In this case all boarding points will additionally have specific time window between which person boarding from that point will be available.

[4] Occupancy Constraint :

In order to reach 85% occupancy or more with given sample data maximum allowed distance constraint needs to be violated.

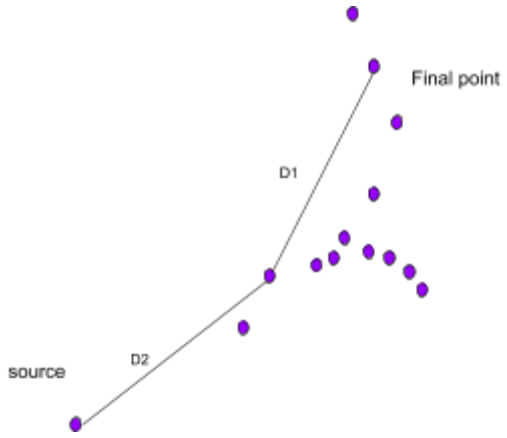
INPUT: Algorithm takes distance matrix as input. We have calculated real world shortest distance between two points using TomTom's api.

Time window array for all the boarding points if routing is to be done considering it.

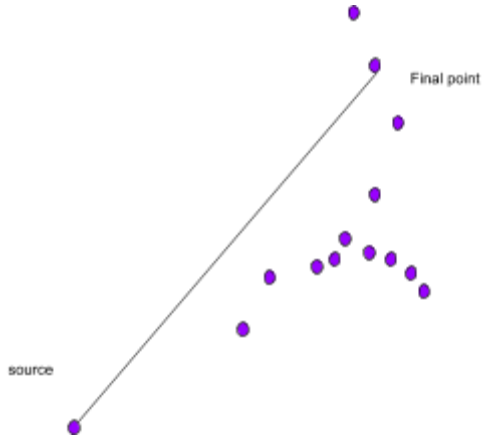
OUTPUT: Algorithm gives output as list of sequence of boarding points which buses should follow in order to minimise cost considering other constraints. It will also give the point from which all buses should start.

Algorithm

Step 1 : Sort the boarding point array with key as total distance from source and final point. Sorted array ensures that if we fail to add one point into mst then all the points following that point will also fail.

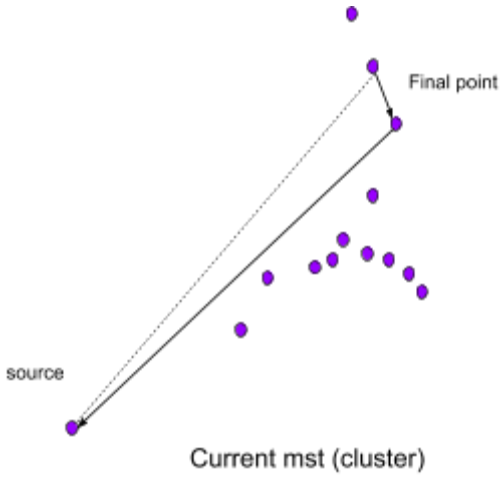


Step 2 : calculate total distance between source and final point.

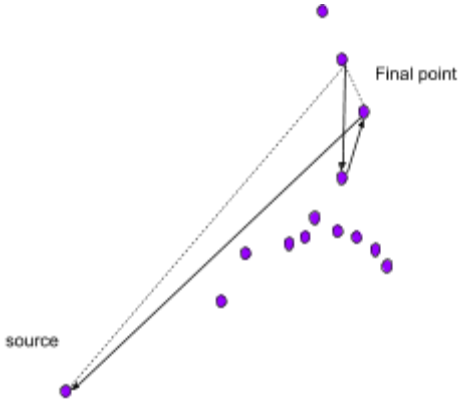


Step 3: one by one try to add points between source and final point into mst from sorted array considering the maximum distance allowed per trip and capacity of bus. If it is feasible to add point between any two points in current mst add it into mst.

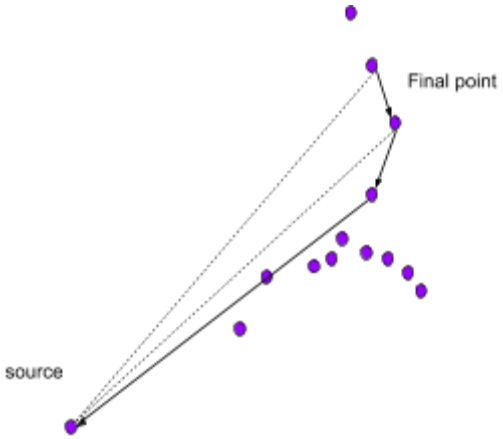
Optionally if boarding points have time window then consider that as well.



Current mst (cluster)



Try to add between every pair of edge

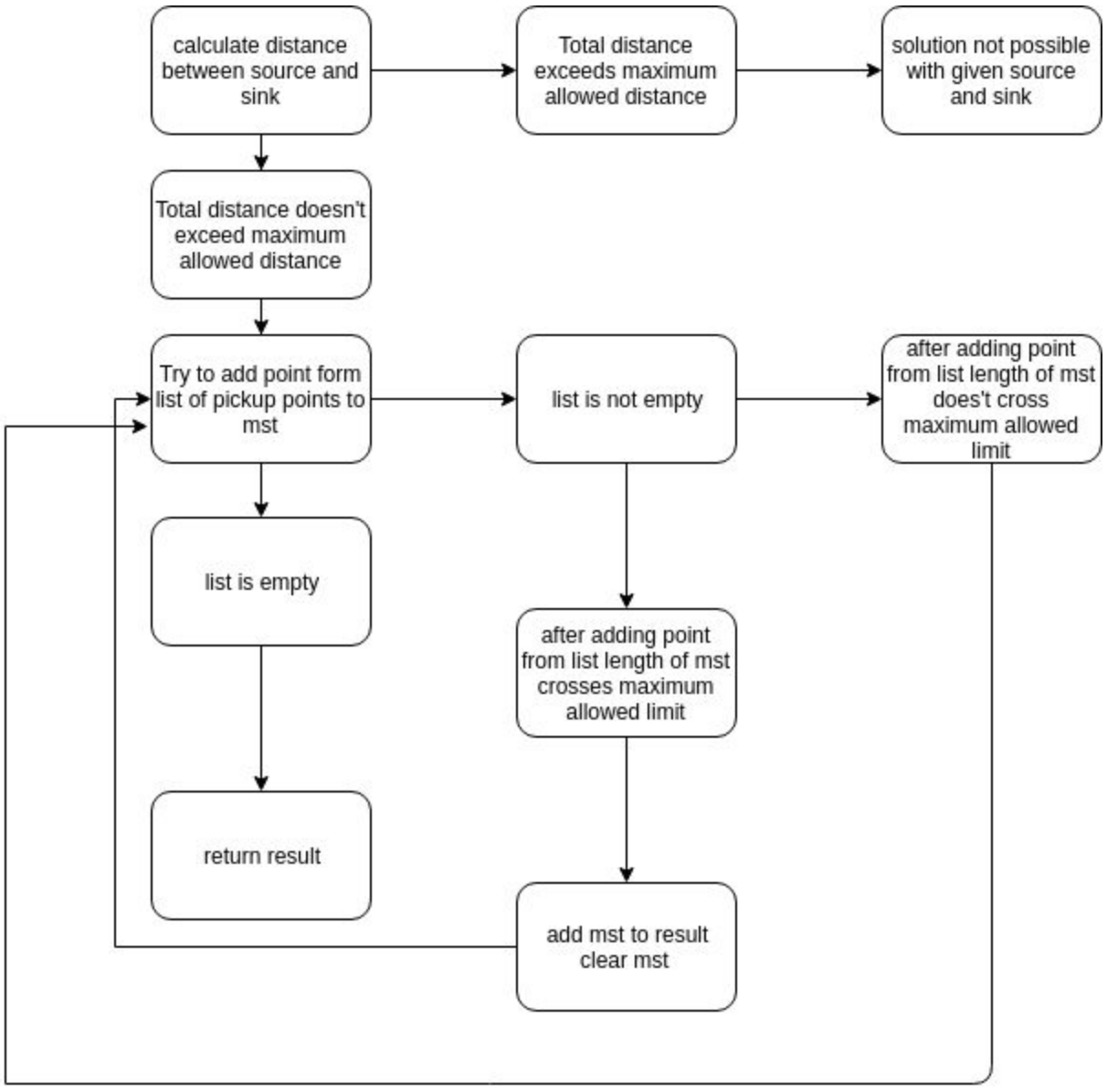


Try to add between every pair of edge

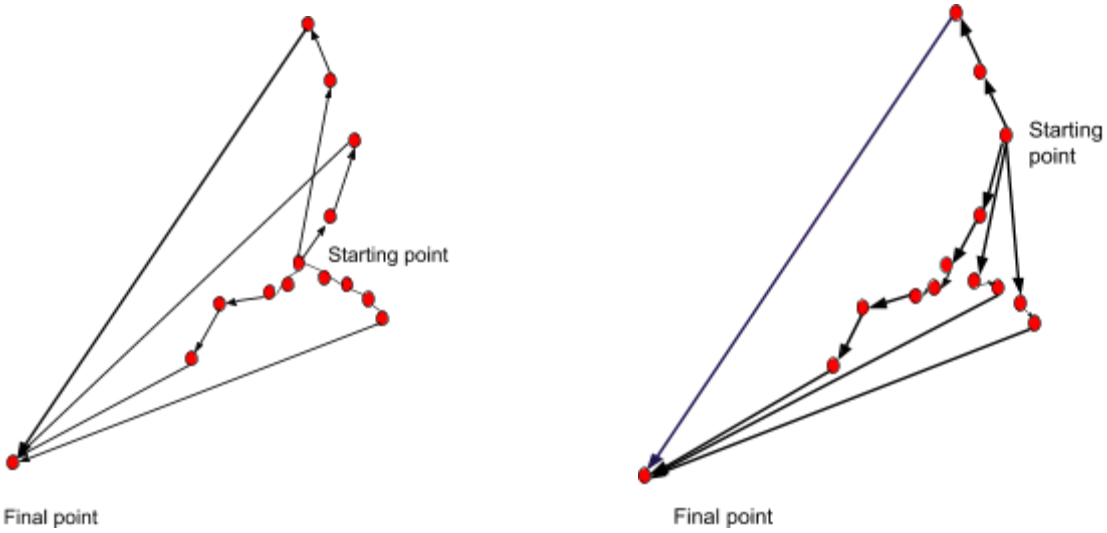
Step 4: If at any point length of mst becomes more then allowed length of path close that tree, add into final answer and initialise new tree. Closed tree is sequence of points for one bus.

Step 5: Continue the process till all points are included.

Cluster function



Step 6 : Try above process taking all points from boarding point list as final point one by one and final solution is most optimal of all.



In the above figure final routes are shown with respect to two different starting points. For different starting points the total distance covered by all the buses and total number of buses required will be different. As operational cost is directly proportional to total distance covered best solution is the one which gives routes with least total distance.

Implementation

We have implemented our algorithm in python language and psuedo code for algorithm is as below. Complete implementation can be found here : https://github.com/JanhaviDadhania/Solution-for-Vehicle-Routing-Problem-for-Employee-Transportation-Management

Psuedo-code:

```
main ( ) :

    final_tree = [ ]

    total_no_of_buses = INF

    for i in range(0 to total_no_of_points) :
```

```
            sink = i
            no._of_buses, mst = main_function ( )
            if no._of_buses < total_no_of_buses :
                total_no_of_buses = no._of_buses
                final_tree = mst
        return final_tree, total_no_of_buses



cluster (distance_matrix, source, sink, pickup_points, max_allowed_dist) :
        result = [ ]
        for points in pickup_points :
            try:
                current_mst = add_to_current_mst (distance_matrix, source, sink,
current_mst, point, max_allowed_dist)
            except :
                result.append(current_mst)
                current_mst = construct new mst


add_to_current_mst (distance, source, sink, current_mst, point, maximum_allowed_dist):
        current_mst = current_mst.add(point)
        if length.current_mst < maximum_allowed_dist :
            return current_mst
        else :
            raise exception
```

OR

```
add_to_current_mst (distance, source, sink, current_mast, point,
maximum_allowed_dist, time_window array) :

    current_mst = current_mst.add(point,time_window)

    if current_mst is feasible :

        return current_mst

    else :

        raise exception
```

Comparison

Comparison of our result with some standard algorithm results with given sample data as input.

1. **Ant Colony Optimisation**
    a. Shortest Path : 0 -> 19 ->18 -> 2 -> 12 -> 13 -> 10 -> 5 -> 7 -> 9 -> 6 -> 4 -> 1 -> 16 -> 11 -> 15 -> 17 -> 3 -> 8 -> 14

    b. Distance: 58602 metres

Reference: Akavall's Ant Colony Optimization[2]

2.**Tabu Search**
    a. Shortest Path : 0 -> 19 ->18 -> 2 -> 12 -> 13 -> 10 -> 5 -> 7 -> 9 -> 6 -> 8 -> 1 -> 16 -> 15 -> 11 -> 17 -> 3 -> 4 -> 14

    b. Distance : 58001 metres

Reference : Tabu search [3]

3. **Google's OR tools**

    a. Shortest Path: 0 -> 19 -> 18 -> 2 -> 12 -> 13 -> 14 -> 4 -> 3 ->

17 -> 11 -> 15 -> 16 -> 1 -> 8 -> 6 -> 9 -> 5 -> 7 -> 10

b. Distance : 59136 meters

Reference: Google OR tools VRP[4]

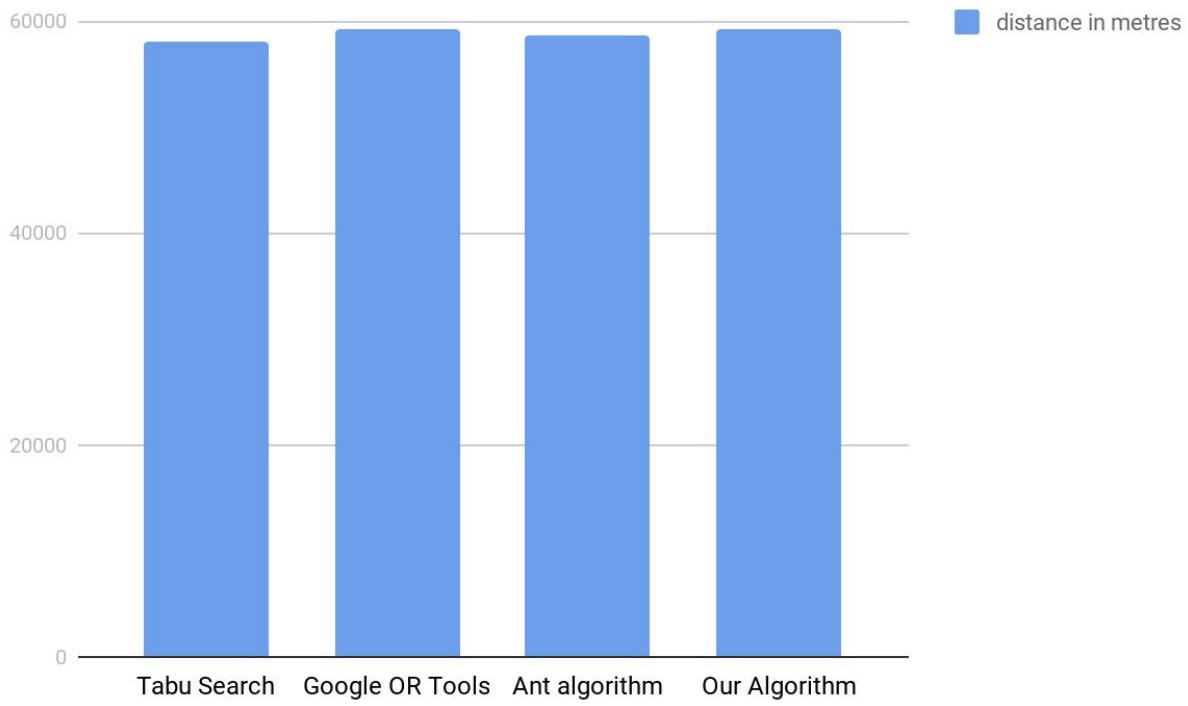
4. **Our Algorithm**

a. Shortest Path: 0 -> 18 -> 19 -> 10 -> 13 -> 12 -> 2 -> 14 -> 7 -> 5 -> 4 -> 9 -> 6 -> 1 -> 8 -> 3 -> 11 -> 15 -> 17 -> 16

b. Distance : 59315 meters

Here, we have violated maximum distance constraint in order to get one route as result.

Source code: [5]



Conclusion

Our algorithm when ran on given input data with maximum distance allowed per day as 92000 kilometres gives the output as below.

Input - 1. Distance matrix

2. Capacity of bus - 32

3. Maximum distance allowed per day - 92000 metres

4. 85% occupancy - not needed

Output

Number of Buses: 3

Route for Bus 1: 0 -> 6 -> 8 -> 3 -> 11 -> 15 -> 17 -> 1

BOSCH bidadi -> kathriguppe circle -> Bata show room -> kathriguppe -> jantha bazar -> jayanagar -> mantri apartment -> deve gowda petrol bunk

Distance travelled: 43258 meters

Route for Bus 2: 0 -> 7 -> 5 -> 4 -> 9 -> 16 -> 1

BOSCH bidadi -> Hosakerrali -> PESIT college -> sri kamkya theater -> ittamadu -> kedirenhalli -> deve gowda petrol bunk

Distance travelled: 38194 meters

Route for Bus 2: 0 -> 18 -> 19 -> 10 -> 13 -> 12 -> 2 -> 14 -> 1

BOSCH bidadi -> kodipalya -> kengeri uttarhalli main road -> Rajarajeshwari nagar temple -> kanthi sweets -> rajarajeshwari nagar double road -> channasandra RNSIT -> chowdeshwari temple -> deve gowda petrol bunk

Distance travelled: 37384 meters

Input - 1. Distance matrix

2. Capacity of bus - 32

3. Maximum distance allowed per day - 92000 metres

4. 85% occupancy - needed

Output - 0 -> 18 -> 19 -> 10 -> 13 -> 12 -> 2 -> 14 -> 7 -> 5 -> 4 -> 9 -> 6 -> 1 -> 8 -> 3 -> 11 -> 15 -> 17 -> 16

Locations : Kedirenhalli -> Mantri Apartment -> Jayanagar -> Jantha Bazar -> Kathrigupped -> Bata show room -> Deve Gowda Petrol Bunk -> Kathriguppe Circle -> Ittamadu -> Sri Kamkya Theater -> PESIT college -> Hosakerralli -> Chowdeshwari Temple -> channasandra RNSIT -> Rajarajeshwarinagar Double road -> Kanthi Sweets -> Rajarajeshwarinagar temple -> Kengeri uttarhalli main road -> Kodipalya -> BOSCH bidadi

Total number of buses required - 1

Total distance - 59315 metres

References:

[1] Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M.M., Soumis, F. “VRP with Time Windows”. In P. Toth and D. Vigo (eds.): The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications, vol. 9, Philadelphia, PA, 157-193. 2002.

[2] https://github.com/Akavall/AntColonyOptimization

[3] https://www.techconductor.com/algorithms/python/Search/Tabu_Search.php

[4] https://developers.google.com/optimization/routing/vrp

[5] https://github.com/JanhaviDadhania/Solution-for-Vehicle-Routing-Problem-for-Employee-Transportation-Management