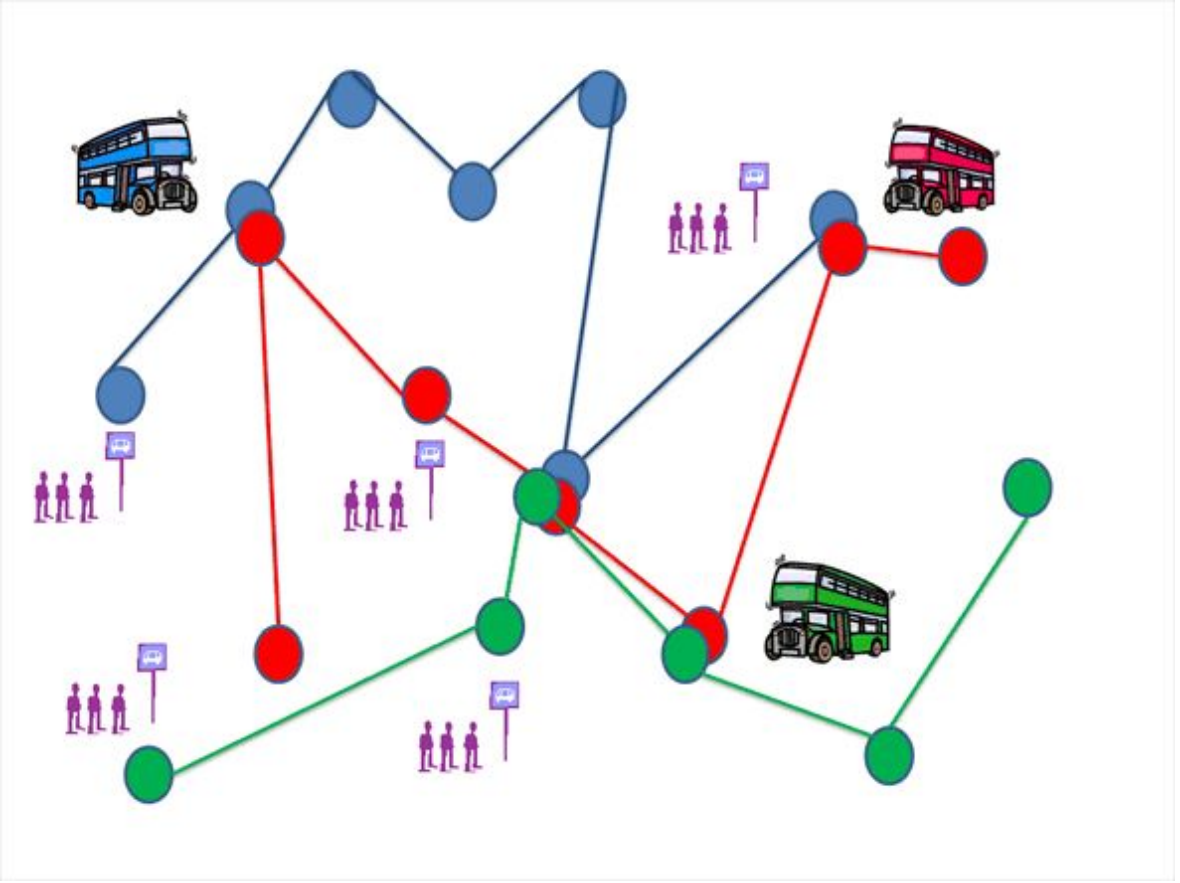
BOSCH's Route Optimization Algorithm

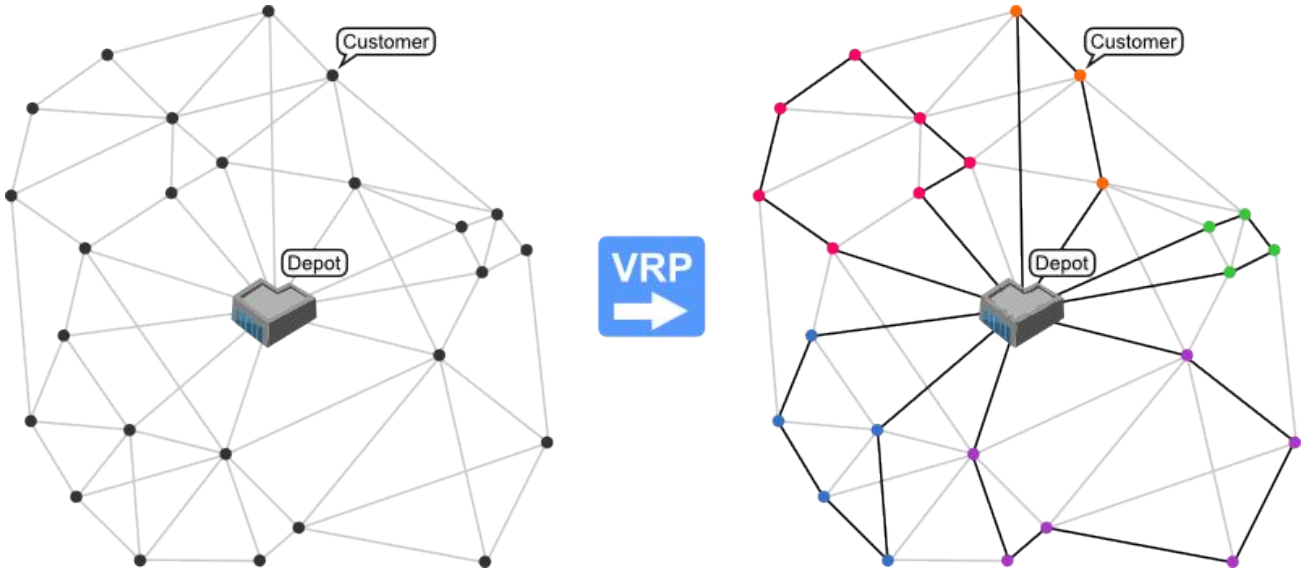
IIT Hyderabad

Problem Statement

- Develop an algorithm for generating efficient routes.
- The routes should cover all the pickup points given.
- These routes should satisfy following constraints
  - Minimum operational cost
  - Time window for travel
  - Only certain capacity of vehicles allowed
  - Every vehicle should get 85% occupancy for a trip.
  - Total distance travelled for a vehicle in a day is limited.



Problem Statement



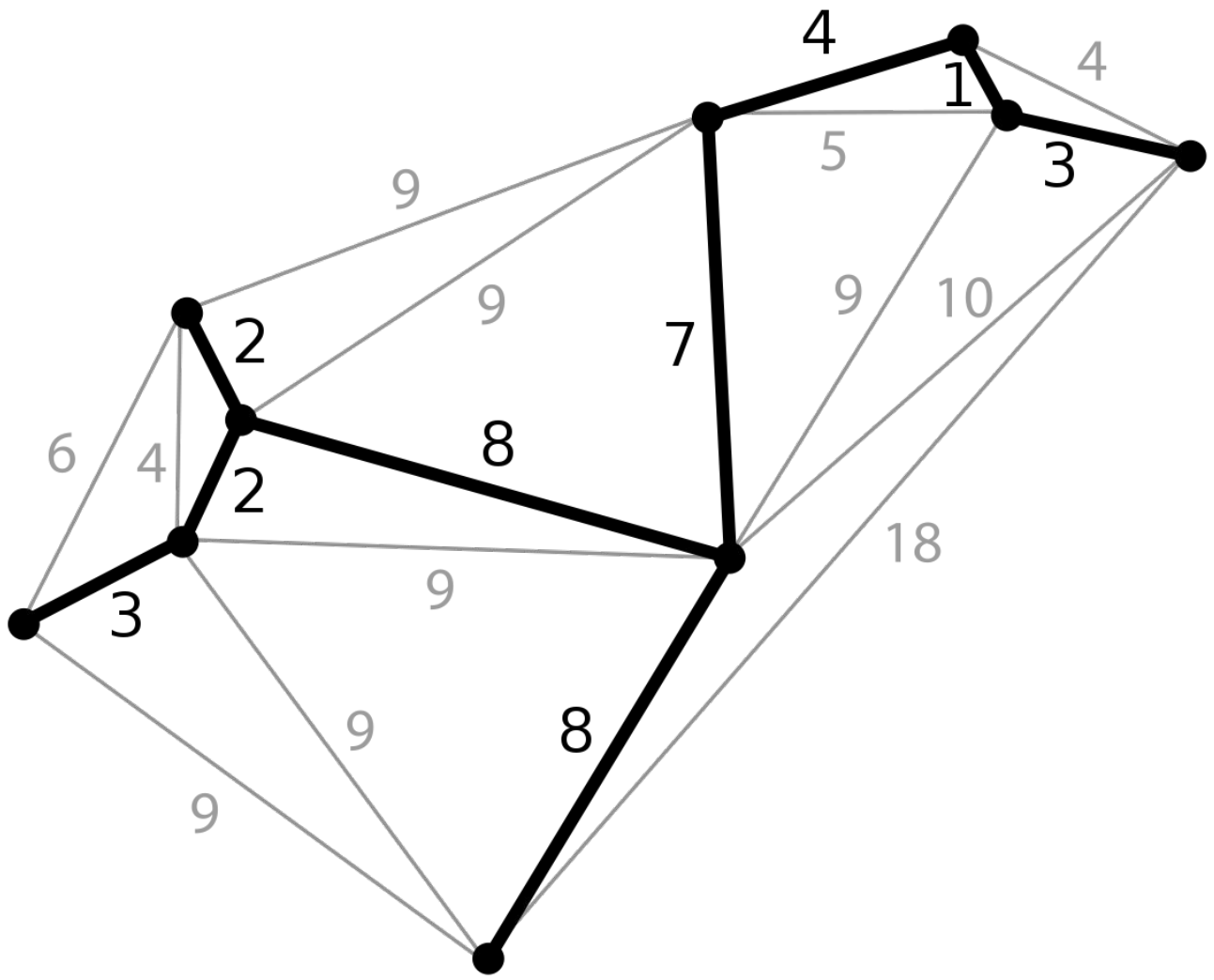
Need of automation

- New Staff addings
- Old staff shifting
- Trial and error for different bus capacities
- Modelling of operational cost is complex

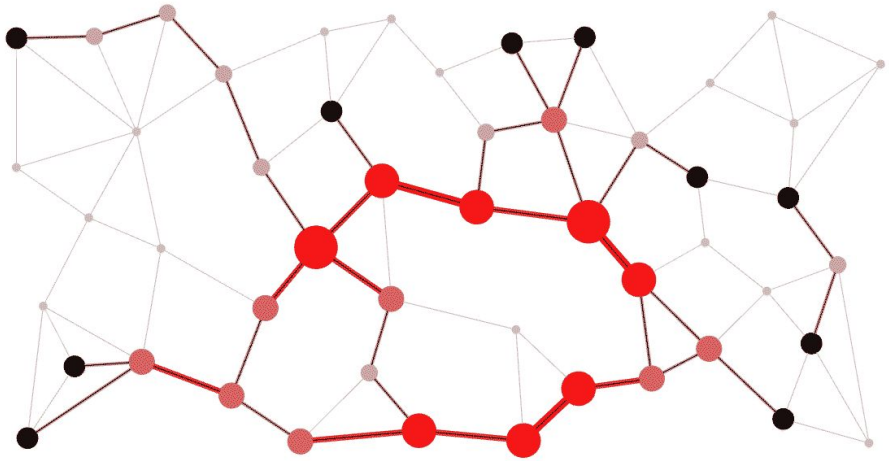
BRUTE FORCE ?

$O(N! \times K^N)$

NP - Hard



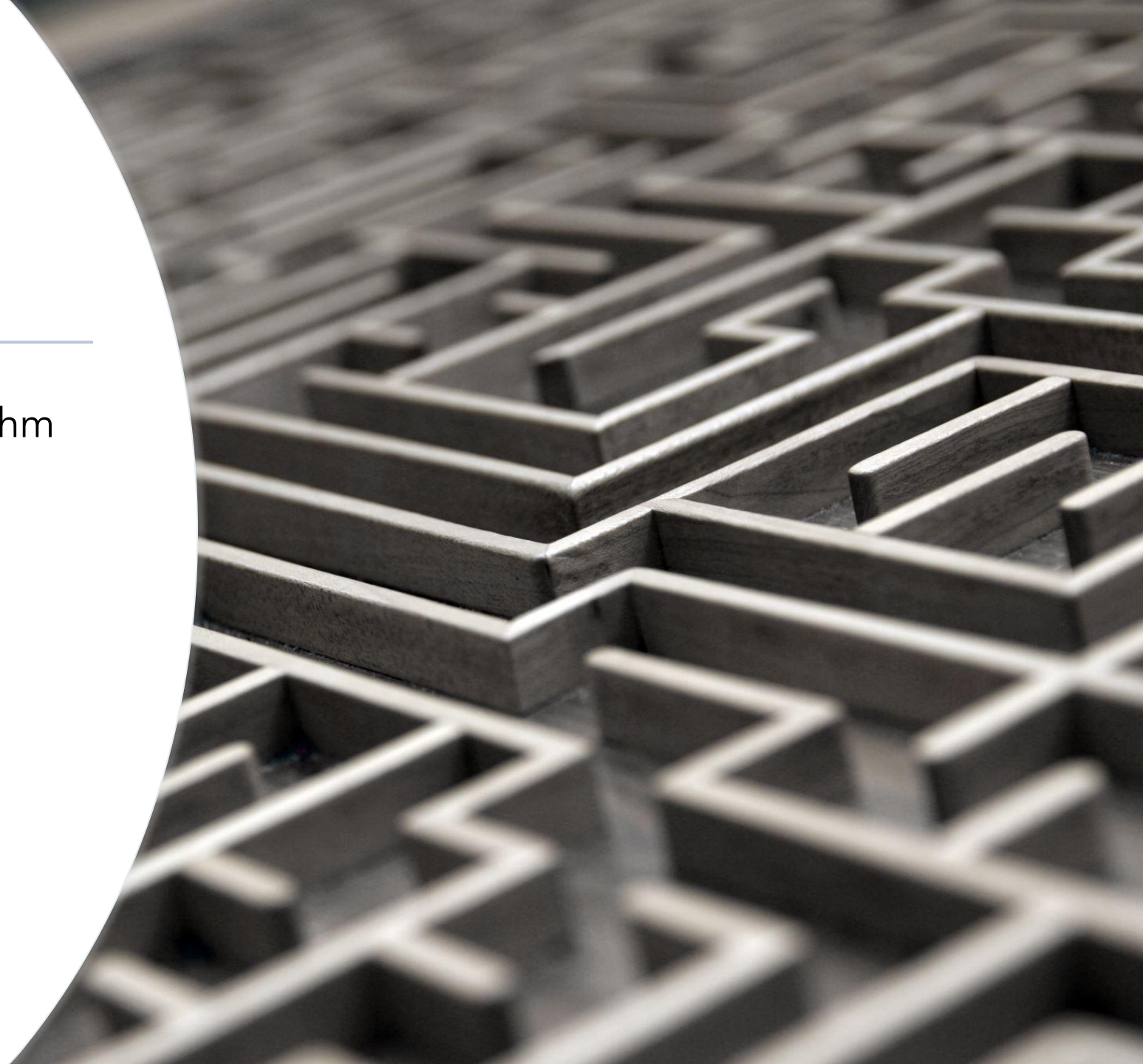
combinatorial optimization



seeks to improve an algorithm by using mathematical methods either to reduce the size of the set of possible solutions or to make the search itself faster

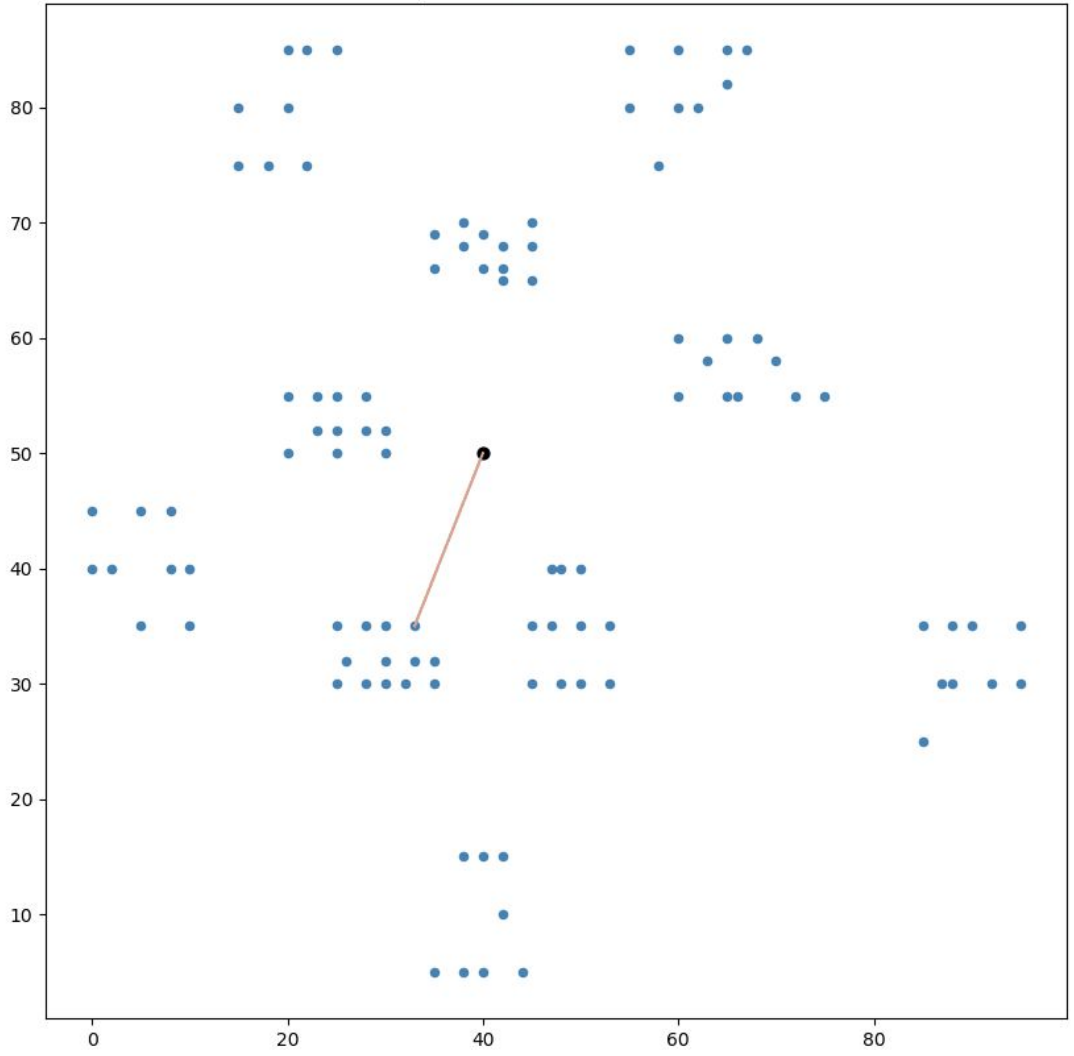
OUR Approach

- Initial solution using greedy algorithm
- Local Search
- Guided local search



Greedy Approach

Starting from a route "start" node, connect it to the node which produces the cheapest route segment, then extend the route by iterating on the last node added to the route.



Local Search

A 2-opt move consists of eliminating two edges and reconnecting the two resulting paths in a different way to obtain a new tour.

There is only one way to reconnect the paths that yield a different tour. Among all pairs of edges whose 2-opt exchange decreases the length we choose the pair that gives the shortest tour.

This procedure is then iterated until no such pair of edges is found.

Local Search 1 : Best Accept

```
1:  input: starting solution, s_0
2:  input: neighborhood operator, N
3:  input: evaluation function, f
4:  current ⇐ s_0
5:  done ⇐ false
6:  while done = false do
7:     best_neighbor ⇐ current
8:     for each s ∈ N(current) do
9:        if f(s) < f(best_neighbor) then
10:          best_neighbor ⇐ s
11:       end if
12:    end for
13:    if current = best_neighbor then
14:       done ⇐ true
15:    else
16:       current ⇐ best_neighbor
17:    end if
18: end while
```

Need for Meta-Heuristic methods

- These local search algos gets stuck at local minimal points but we need to strive for a global minimum.

- A metaheuristic is an iterative generating process, controlling an underlying heuristic, by combining (in an intelligent way) various strategies to explore and exploit search spaces (and learning strategies) to find near-optimal solutions in an efficient way

- The meta heuristic methods can be used, to sit on top of local search algorithm to change it's behaviour and escape from local minimas and plateus

Guided Local Search

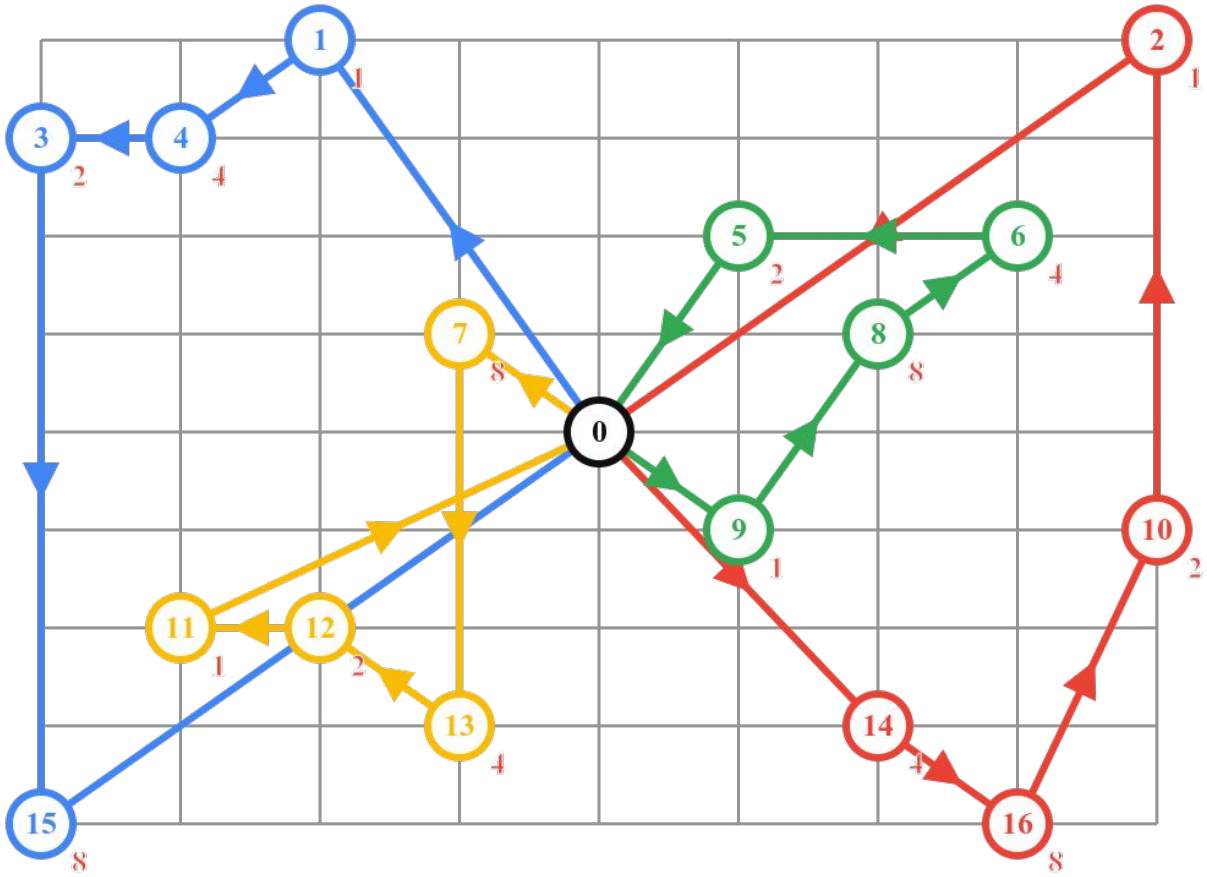
- The strategy for the Guided Local Search algorithm is to use penalties to encourage a Local Search technique to escape local optima and discover the global optima.
- A Local Search algorithm is run until it gets stuck in a local optima.
- The features from the local optima are evaluated and penalized, the results of which are used in an augmented cost function employed by the Local Search procedure.
- The Local Search is repeated a number of times using the last local optima discovered and the augmented cost function that guides exploration away from solutions with features present in discovered local optima.

Procedure

- Augmented Cost Function : $h(s) = g(s) + \lambda \cdot \sum_{i=1}^{M} f_i$
- Penalties are only updated for those features in a locally optimal solution that maximize utility
- utility for a feature: $U_{feature} = \frac{C_{feature}}{1+P_{feature}}$

Pseudocode

Input: $Iter_{max}, \lambda$
Output: S_{best}
$f_{penalties} \leftarrow \emptyset$
$S_{best} \leftarrow$ `RandomSolution()`
For $(Iter_i \in Iter_{max})$
  $S_{curr} \leftarrow$ `LocalSearch`$(S_{best}, \lambda, f_{penalties})$
  $f_{utilities} \leftarrow$ `CalculateFeatureUtilities`$(S_{curr}, f_{penalties})$
  $f_{penalties} \leftarrow$ `UpdateFeaturePenalties`$(S_{curr}, f_{penalties}, f_{utilities})$
  If (`Cost`$(S_{curr}) \leq$ `Cost`(S_{best}))
    $S_{best} \leftarrow S_{curr}$
  End
End
Return (S_{best})



Implementation

Preprocessing

Defining Routing Model

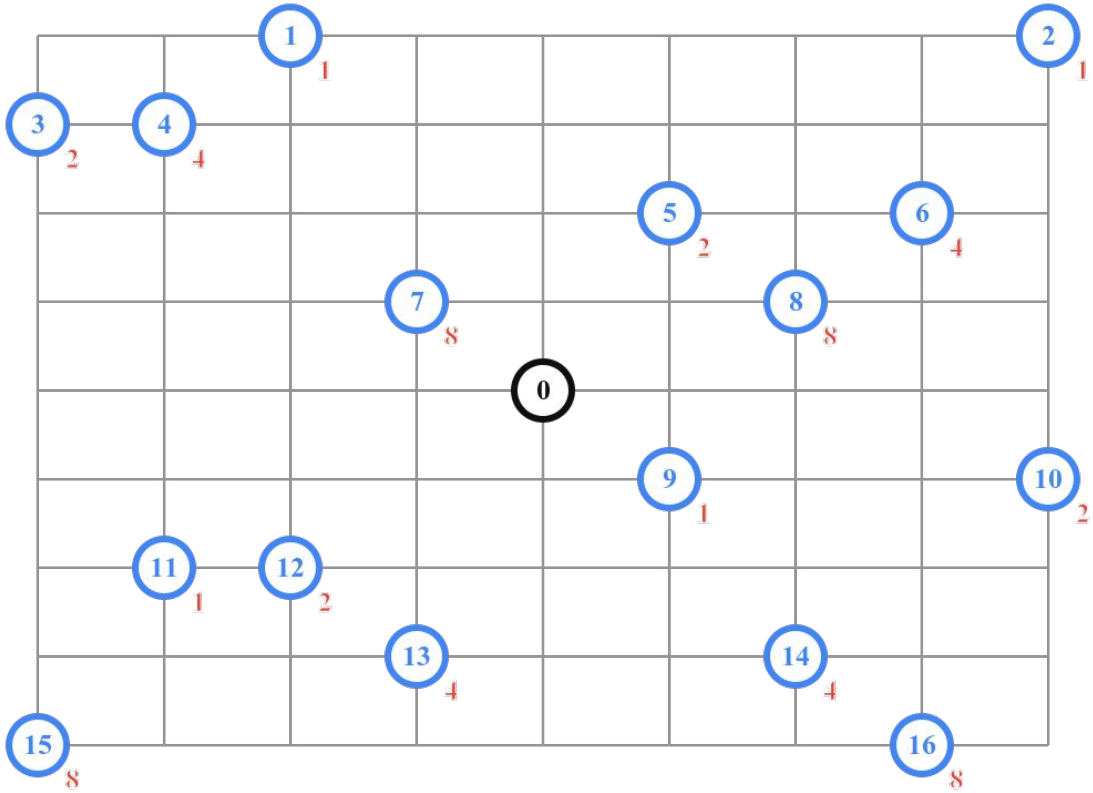
Adding constraints

Specifying Search Strategies

Solving

PREPROCESSING

- Generate Distance Matrix – Google maps Distance matrix API.
- Listing pickup points and number of pupil at each pickup point.
- Defining number of vehicles and their capacities.



Distance Constraint

```
1    dimension_name = 'Distance'
2    routing.AddDimension(
3        transit_callback_index,
4        0,  # no slack
5        3000,  # vehicle maximum travel distance
6        True,  # start cumul to zero
7        dimension_name)
8    distance_dimension = routing.GetDimensionOrDie(dimension_name)
9    distance_dimension.SetGlobalSpanCostCoefficient(100)
```

Creating Distance dimension to compute cumulative distances and assign costs

Vehicle Capacity Constraint

```
demand_callback_index = routing.RegisterUnaryTransitCallback(
        demand_callback)
    routing.AddDimensionWithVehicleCapacity(
        demand_callback_index,
        0,  # null capacity slack
        data['vehicle_capacities'],  # vehicle maximum capacities
        True,  # start cumul to zero
        'Capacity')
```

Creating Distance dimension to compute cumulative distances and assign costs

Specifying Search Strategies

```
search_parameters = pywrapcp.DefaultRoutingSearchParameters()
    search_parameters.local_search_metaheuristic = (
            routing_enums_pb2.LocalSearchMetaheuristic.GUIDED_LOCAL_SEARCH)
    search_parameters.time_limit.seconds = 1
    search_parameters.log_search = True

    # Solve the problem.
    assignment = routing.SolveWithParameters(search_parameters)
```

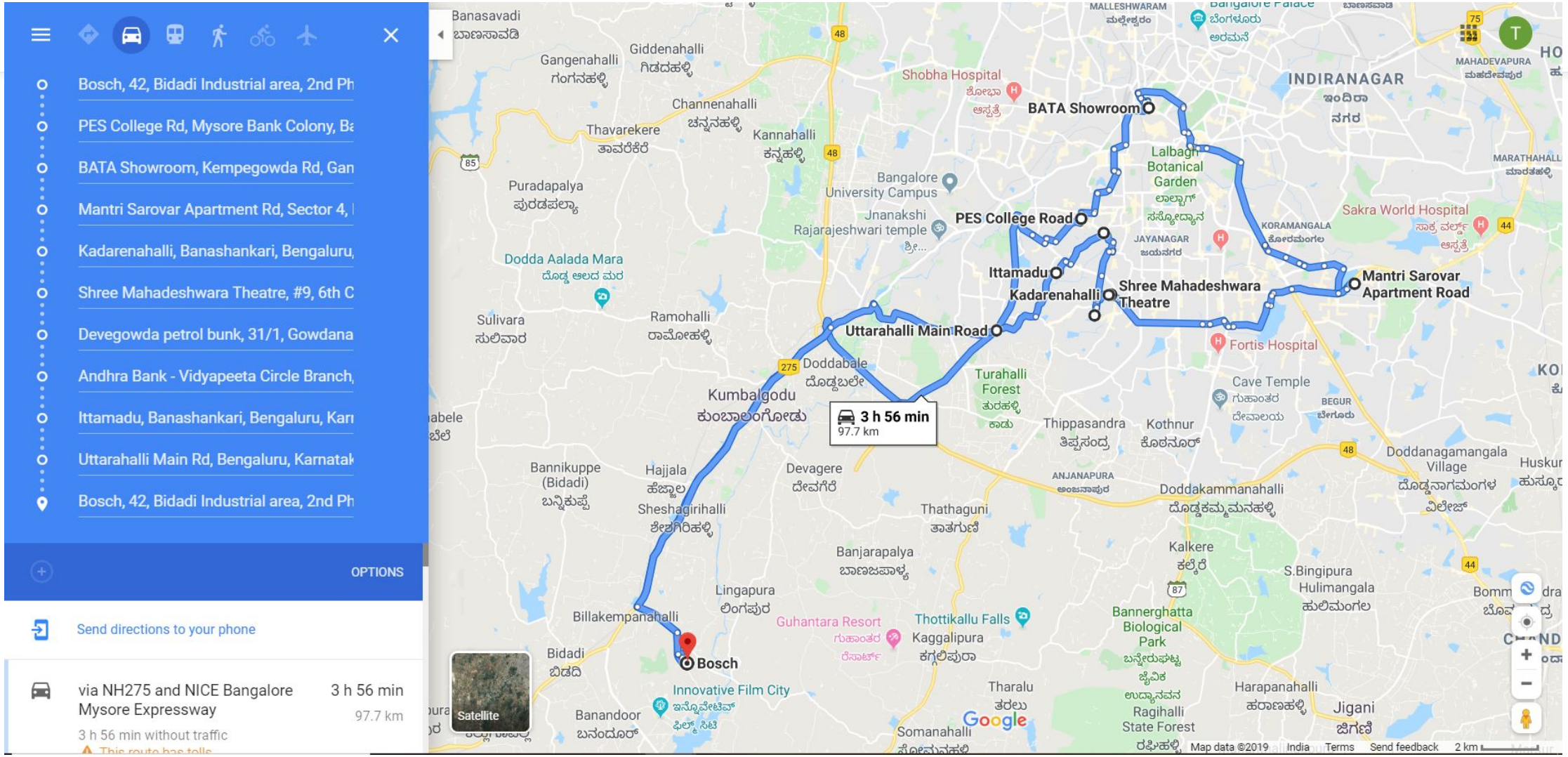
Declaring the heuristic method and solving the optimisation problem

Results

```
Route for vehicle 1:
 bosch bidadi, Banglore: Load(0) ->  PESIT Collage, Banglore: Load(6) ->  Bata Show Room, Banglore:
 Load(14) ->  Mantri Apartment, Banglore: Load(16) ->  Kadirenahallli, Banglore: Load(20) ->  Chowd
eshwari Talkies, Banglore: Load(24) ->  Devegowda Petrol Bunk, Banglore: Load(26) ->  Kathergguppe C
ircle, Banglore: Load(28) ->  Ittamadu, Banglore: Load(31) ->  Uttarahalli road Kengeri, Banglore:
Load(32) ->  bosch bidadi, Banglore: Load(32)
Distance of the route: 97428m
Load of the route: 32

Route for vehicle 2:
 bosch bidadi, Banglore: Load(0) ->  Jantha Bazar, Banglore: Load(4) ->  Rajarajeshwarinagar Double
 Road, Banglore: Load(8) ->  Jayanagar, Banglore: Load(9) ->  Kattrigupe, Banglore: Load(14) ->  Ku
thriguppe, Banglore: Load(22) ->  Kathriguppe, Banglore: Load(30) ->  Kamakya Theatre, Banglore: Lo
ad(31) ->  Kanthi Sweets RR Nagar, Banglore: Load(32) ->  bosch bidadi, Banglore: Load(32)
Distance of the route: 72863m
Load of the route: 32

Route for vehicle 3:
 bosch bidadi, Banglore: Load(0) ->  Channasandra RNSIT, Banglore: Load(4) ->  Kodipalya (Uttarahal
li main Road), Banglore: Load(6) ->  HosaKerehalli, Banglore: Load(10) ->  Hoskeralli, Banglore: Lo
ad(18) ->  Hosakarehalli, Banglore: Load(19) ->  Rajarajeshwari temple, Banglore: Load(28) ->  bosc
h bidadi, Banglore: Load(28)
Distance of the route: 53762m
Load of the route: 28

Total distance of all routes: 224053m
Total load of all routes: 92
(env_tech)  pran@Skyera  ~/Documents/techmeet
```

S.No	Persons	Boarding point
1	B1	Devegowda Petrol Bunk
2	B2	Hoskeralli
3	B3	Channasandra RNSIT
4	B4	Kathriguppe
5	B5	Kamakya Theatre
6	B6	PESIT Collage
7	B7	Kathergguppe Circle
8	B8	HosaKerehalli
9	B9	Bata Show Room
10	B10	Ittamadu
11	B11	Rajarajeshwari temple
12	B12	Jantha Bazar
13	B13	Kuthriguppe
14	B14	Hosakarehalli
15	B15	Rajarajeshwarinagar Double Road
16	B16	Kathriguppe
17	B17	HosaKerehalli
18	B18	Kamakya Theatre
19	B19	Kanthi Sweets RR Nagar
20	B20	HosaKerehalli
21	B21	Chowdeshwari Talkies
22	B22	Jayanagar
23	B23	Kattrigupe
24	B24	Kadirenahallli
25	B25	Kathriguppe
26	B26	Mantri Apartment
27	B27	Channasandra RNSIT
28	B28	Kodipalya (Uttarahalli main Road)
29	B29	Uttarahalli road Kengeri

Visualisation - ROUTE #1



Checklist

- ✅ Minimise Total Travel Distance
- ✅ Bus Capacity Constrained
- ✅ Vehicle maximum distance limit
- ✅ Occupancy of each bus
- ✅ Time window (can be achieved by trading with max distance)

THANK YOU