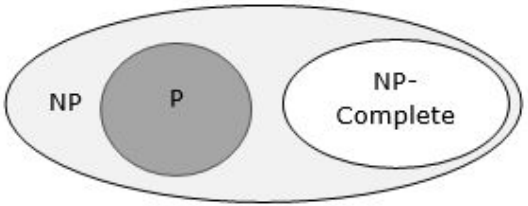
BOSCH

BOSCH'S ROUTE OPTIMIZATION

INTER IIT TECH MEET '19

TEAM IIT JAMMU

WHAT THE PROBLEM IS?



There are two types of algorithmic problems:

- solvable in polynomial time
- unsolvable in polynomial time, that is they have exponential time complexity.

Further Unsolvable in polynomial time are of three types: NP, NPC, NPHard

- A **NP hard** problem is- when every problem in NP is reducible to this problem
- A problem is in the **class NPC** if it itself is NP and is NP Hard also. That is, it is at least as hard as any NP problem.

Why we pointing out this is - **Our Vehicle Route Optimisation Problem is an NPC problem.**

Since we can't solve this optimisation problem in Poly-time, the present algorithms don't always guarantee the optimal solution and are based on different type of heuristics and objective minimisation.

Routing problem mainly is divided into two parts :- node-routing problems and arc routing problem.
In arc routing problem aim is to find the shortest path that traverse every arc in the graph.
In node routing problem aim is to include every node, not necessary traverse every arc.

Our problem comes under the node routing problem as we have to include all nodes which has passenger demand greater than zero.

The routing problem in which there is just one vehicle is called as Travelling salesman problem (TSP) and TSP with multiple vehicles is vehicle routing problem (VRP)

Constraints optimization is based on feasibility rather than finding an optimal solution and focuses in the constraints and variables rather than the objective function. The goal may simply be to narrow down a very large set of possible solutions to a more manageable subset by adding constraints to the problem.
For solving CP problem two solvers can be used - CP-SAT Solver and the Original CP solver.
In CP-SAT solver all constraints should be such that variables take only integer values.

When some variables can take non-integer value we can use mixed integer programming. For MIP we can use MIP Solver or CP-SAT Solver. MIP can be used when the feasible set is convex in nature and for non-convex feasible sets CP-SAT Solver gives faster and more accurate result.

Routing Solver

- **Search limits:-** We can terminate the solver after it reaches a specified limit it can be maximum length of time like time_limit.seconds, or the number of solution found like solution_limit.

In problems like this we assume a starting solution of the problem, there are many strategies that can be used to find an initial solution.

- **Path Cheapest Arc** - we start from the “start” node, connect it to the node which produces the cheapest segment, then extend the route by iteration on the last node added to the route.
- **Path Most Constrained Arc** - arcs are evaluated with a comparison-based selector which will favor the most constrained first.
  Christofides - it works on generic vehicle routing models by extending a route until no nodes can be inserted on it.

- **Cheapest Arc** :- it iteratively connects the first node in order that has no successor to a free node,such that the distance between them is minimal.

Heuristics is based on partial search which sometimes not completely escape the local optima therefore using meta-heuristics is better approach.

Different local-search methods are:

- **Guided Local Search** - Uses penalized cost function in order to escape local optima
- **Simulated Annealing** - probabilistic technique for approximating the global optimum of a given function, often used when the search space is discrete.
- **Greedy Descent** - accepts neighbor solutions only if the solution's cost decreases, until a local minima is reached.

OUR APPROACH

We are optimizing the routes a bus should take to minimize the overall cost, given the bus stop locations and no of people at each stop.

We are trying to find the no of buses that we need to buy based on our constraints. This will depend on the distance travelled cost (petrol price per unit distance), initial investment in bus (bus price, driver salary, etc) and the returns (say earning per unit distance travelled).

Once we have the route of each bus, by assuming speed one can find the time at which the bus will be present at the node.

THE ALGORITHM

CONSTANTS

F = fixed cost associated with a bus (Rs/bus)

r = return per sq of unit travel dist (Rs/km^2)

R = routing cost per unit travel dist (Rs/km)

t_{ij} = travel time between node i and j (min)
for $i \in DS$, $j \in SE$

P_i = demand (passengers) of node i, for $i \in S$

Q_k = capacity of the bus k, for $k \in V$

DATA SET

S = demand nodes (stops)

D = a bus's starting node, i.e. depot

E = a bus's ending node, i.e. company

DS = all nodes which permit on out- flow, i.e. depot and bus stops, $D \cup S$

SE = all nodes which permit on in-flow, i.e. ending point and bus stops, $E \cup S$

TD = the set of all nodes, $D \cup S \cup E$

V = the vehicle set

T = Constant larger than the total travel time of any feasible route

The decision variable represent binary (0-1) decisions of which buses to assign to which routes. The variable is 1 if a given bus is assigned to a specific route, and 0 otherwise.

$$X_{ij}^{k} = 1 \text{ (if vehicle k travels link ij)}$$
$$= 0 \text{ (otherwise)}$$
$$\text{for } k \in V,\ i \in DS\ J \in SE$$

The objective function The objective of this problem is to minimize the total cost incurred in serving all customers. For this, we need to minimize the number of buses used (fixed costs), and the total distance travelled by all the buses (operational costs)

$$Z = R*\left(\sum_{i\in DS}\sum_{j\in SE}\sum_{k\in V} x_{ij}^{k}\right) + F*\left(\sum_{k\in V}\sum_{i\in D}\sum_{j\in S} x_{ij}^{k}\right) - r*\left(\sum_{i\in DS}\sum_{j\in SE}\sum_{k\in V} (x_{ij}^{k})^2\right)$$

CONSTRAINTS

Each demand node (set s) should be served by only one bus.

Constraint 1: $\sum_{k \in V} \sum_{i \in DS} x_{ij}^{k} = 1 \quad \text{for } j \in S$

Constraint 2: $\sum_{k \in V} \sum_{j \in SE} x_{ij}^{k} = 1 \quad \text{for } i \in S$

If a bus enters a demand node, it must exit that node.

Constraint 3: $\sum_{i \in DS} x_{ip}^{k} - \sum_{j \in DE} x_{pj}^{k} = 0 \quad \text{for} \quad k \in V, p \in S$

The number of vehicle leaving depot must equal the number of vehicle entering the end point of the bus.

Constraint 4: $\sum_{i \in D} \sum_{j \in S} \sum_{k \in V} x_{ij}^{k} - \sum_{i \in S} \sum_{j \in E} \sum_{k \in V} x_{ij}^{k} = 0$

Each vehicle can leave one depot and arrive at the end point only once, which means that the number of routes should be less than or equal to the number of buses.

Constraint 5: $\sum_{k \in V} \sum_{j \in S} x_{ij}^{k} <= V \quad \text{for } i \in D$

Constraint 6: $\sum_{k \in V} \sum_{i \in S} x_{ij}^{k} <= V \quad \text{for } j \in E$

(As per given Problem)

There are upper and lower limits for the number of passengers on the bus as the capacity constraint.

Constraint 7: $\sum_{i \in DS} p_i \left(\sum_{j \in SE} x_{ij}^k \right) <= Q_k$ for $k \in V$

Constraint 8: $\sum_{i \in DS} p_i \left(\sum_{j \in SE} x_{ij}^k \right) >= 0.85 * Q_k$ for $k \in V$

Time window

9 : $tj \geq ti + tij - (1 - xijk)T$

$ai \leq ti \leq bi \ \forall \ i \in I$

$ti \geq 0$

We assume time to be a direct function of distance and hence apply constraints on maximum distance the route can take. The upper bound on the distance is given as 92 km / day

Constraint 9: $\sum_{i \in TD} \sum_{j \in TD} x_{ij}^k * l_{ij} <= 46$ km for all $k \in V$

ti = arrival time at node i. ai,bi = limits (time window)

RESULTS (sample data)

```
objective cost:  119

Route for vehicle 0 with minimum capacity 27:
 0 (Load(0)) ->  2 (Load(5)) ->  6 (Load(6)) ->  8 (Load(7)) ->  12 (Load(8)) ->  4 (Load(14)) ->  5 (Load(16)) ->  7 (Load(17)) ->  13 (Load(18)) ->  10 (Load(19)) ->  15 (Load(20)) ->  14 (Load(21)) ->
Distance of the route: 45m
Load of the route: 29


*****************************
Total distance of all routes: 45m
Total load of all routes: 29
[[0, 2, 6, 8, 12, 4, 5, 7, 13, 10, 15, 14, 1, 16, 3, 11, 9, 17], [0], [0], [0], [0], [0], [0], [0], [0], [0]]
```

```
5 (Load(16)) ->  7 (Load(17)) ->  13 (Load(18)) ->  10 (Load(19)) ->  15 (Load(20)) ->  14 (Load(21)) ->  1 (Load(22)) ->  16 (Load(24)) ->  3 (Load(26)) ->  11 (Load(28)) ->  9 (Load(29)) ->  17 Load(29)




 [0], [0]]
```

Using third party apps

Google maps can be used to find distance matrix, time matrix and to show visualisation. We need to buy the product to use the their services.

PROBLEMS FACED

- Understanding approaches towards the general NPC problems.
- Which solver to use - TORA, IBM’S CPLEX, Python based frameworks ?

- Problems with CPLEX - conflicts with mixed integer boolean (linear) type programming since objective function had a product of two decision variables

- Tens of iterations on framing the constraints since they resulted in conflicts with optimal solution and debugging had been a major issue. Also CPLEX demanded high computation power and time on every iteration.