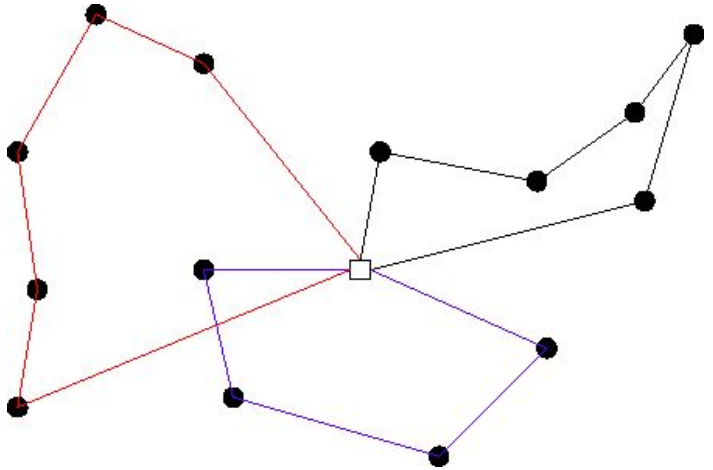
Tabu Heuristics for Vehicle Routing Problem

Presented by:-
Navadiya Dhruvik (171IT225)
Neeraj (171IT226)
Rakesh Pavan (171IT154)
Yogesh Choubey (171IT252)

Vehicle Routing Problem

Vehicle Routing Problem or simply VRP is a well known combinatorial optimization problem and a generalization of the travelling salesman problem. A definition of the problem is this: We have a number of customers that have a demand for a delivery. Which are the optimal (minimal) routes for a fleet of vehicles starting from a single point (depot) to deliver the requested goods in all customers. Finding optimal solution is a NP-hard problem so heuristic strategies are proposed for approximation of the optimal solution.



1. Types of VRP:-

The objective function of a VRP can be very different depending on the particular application several variations and specializations of the vehicle routing problem exist:

- ➔ **Vehicle Routing Problem with Pickup and Delivery (VRPPD)**.
- ➔ **Vehicle Routing Problem with LIFO**
- ➔ **Vehicle Routing Problem with Time Windows (VRPTW)**
- ➔ **Capacitated Vehicle Routing Problem**

Capacitated VRP

Let *G = (V, A)* be a graph where *V* is the vertex set and *A* is the arc set. One of the vertices represents the depot at which a fleet of m identical vehicles of capacity *Q* is based, and the other vertices customers that need to be serviced. With each customer vertex vi are associated a demand *qi*. With each arc *(vi, vj)* of *A* are associated a cost *cij*. The *CVRP* consists in finding a set of routes such that:

- Each route begins and ends at the depot;
- Each customer is visited exactly once by exactly one route;
- The total demand of the customers assigned to each route does not exceed Q;
- The total cost of the routes is minimized.

Algorithms for solving the Vehicle Routing Problem

Heuristic Algorithms

The term **heuristic** is used for algorithms which find solutions among all possible ones ,but they do not guarantee that the best will be found,therefore they may be considered as approximately and not accurate algorithms.These algorithms,usually find a solution close to the best one and they find it fast and easily.Sometimes these algorithms can be accurate,that is they actually find the best solution, but the algorithm is still called heuristic until this best solution is proven to be the best.The method used from a heuristic algorithm is one of the known methods,such as greediness,but in order to be easy and fast the algorithm ignores or even suppresses some of the problem's demands.

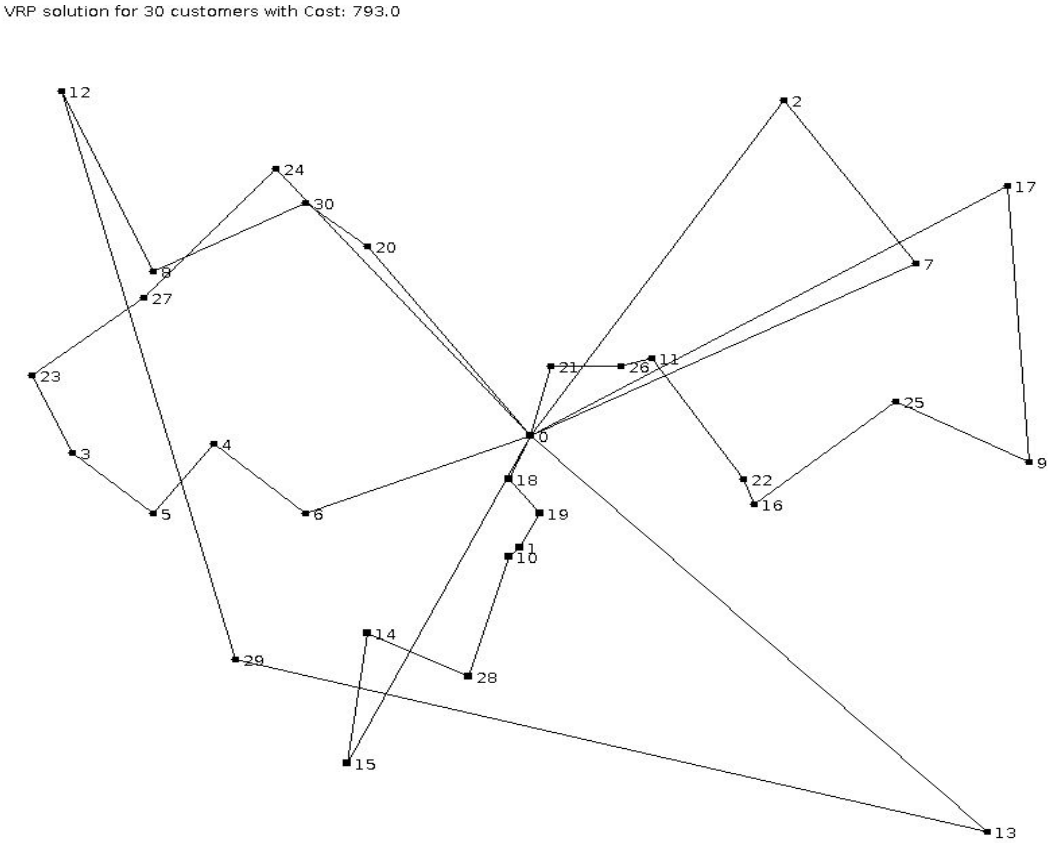
Meta-heuristic Algorithms

Meta-heuristics, on the other hand, are problem-independent techniques. As such, they do not take advantage of any specificity of the problem and, therefore, can be used as black boxes. In general, they are not greedy. In fact, they may even accept a temporary deterioration of the solution (see for example, the simulated-annealing technique), which allows them to explore more thoroughly the solution space and thus to get a hopefully better solution (that sometimes will coincide with the global optimum). Please note that although a meta-heuristic is a problem-independent technique, it is nonetheless necessary to do some fine-tuning of its intrinsic parameters in order to adapt the technique to the problem at hand.

Using Greedy Approach

Nearest Customer in terms of Euclidean Distance is selected as long as it’s capacity does not exceed.

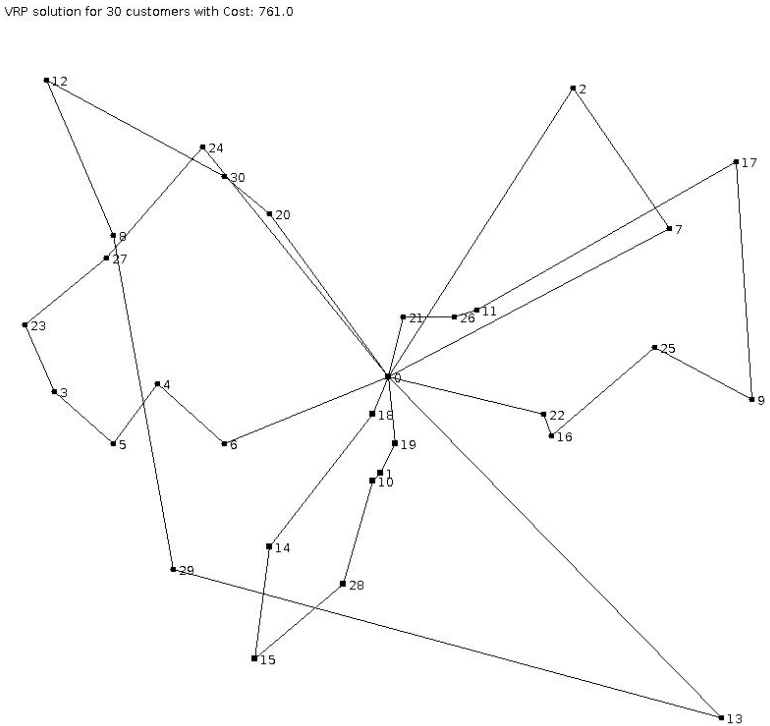
Route Selected using Greedy Approach



Using Intraroute Method

Normal greedy heuristic tries to explore neighbouring solutions of the current solution in hope of finding the local or global minima.

Route Selected using Intraroute Method

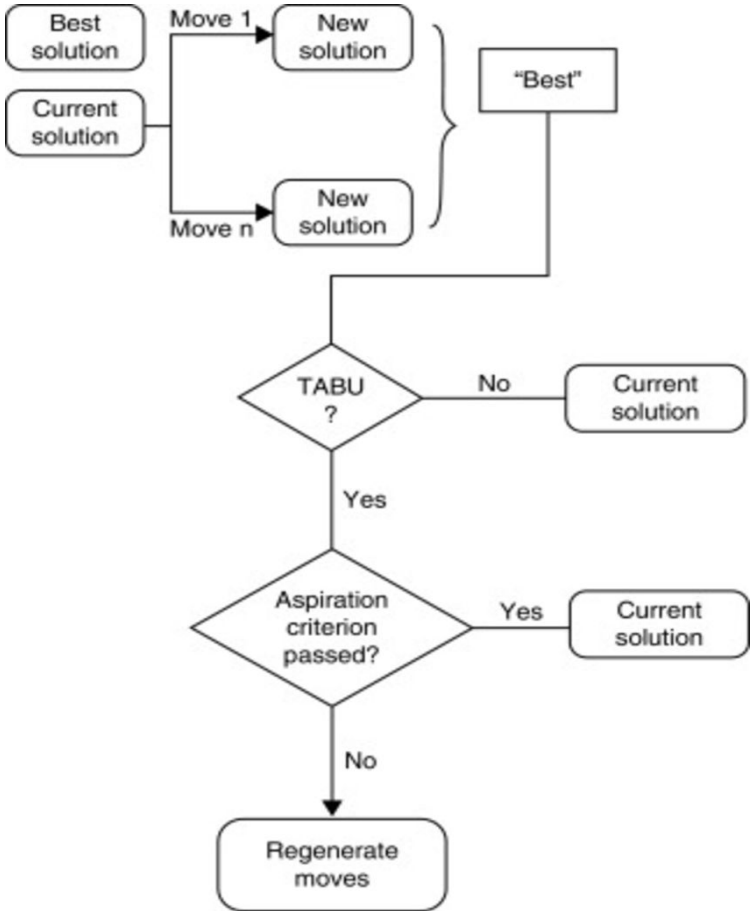


Tabu Search

Tabu search is a metaheuristic search method employing **local search methods**. Local (neighborhood) searches take a potential solution to a problem and check its **immediate neighbors** (that is, solutions that are similar except for very few minor details) in the hope of **finding an improved solution**. Local search methods have a tendency to become **stuck in suboptimal regions** or on plateaus where many solutions are equally fit.

Tabu search enhances the performance of local search by relaxing its basic rule. First, at each step worsening moves can be accepted if no improving move is available (like when the search is stuck at a strict local minimum). In addition, prohibitions (henceforth the term tabu) are introduced to discourage the search from coming back to previously-visited solutions.

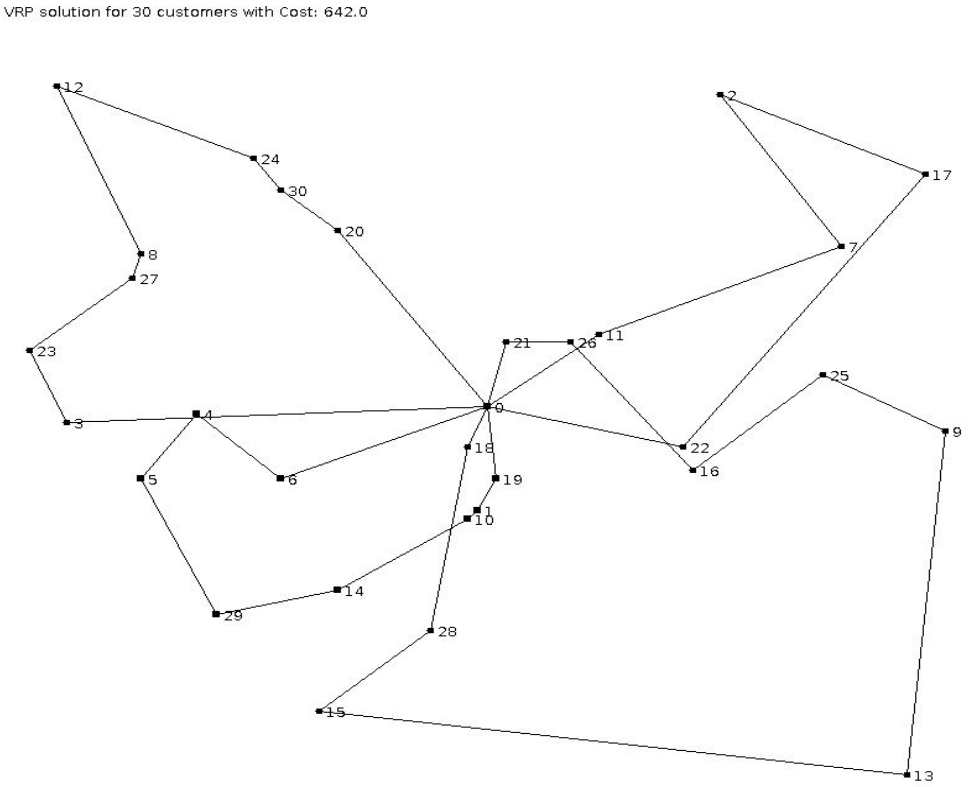
Tabu Search Flowchart



Tabu Search PseudoCode

```
 1 sBest ← s0
 2 bestCandidate ← s0
 3 tabuList ← []
 4 tabuList.push(s0)
 5 while (not stoppingCondition())
 6  sNeighborhood ← getNeighbors(bestCandidate)
 7  bestCandidate ← sNeighborHood.firstElement
 8  for (sCandidate in sNeighborHood)
 9          if ( (not tabuList.contains(sCandidate)) and (fitness(sCandidate) > fitness(bestCandidate)) )
10                  bestCandidate ← sCandidate
11          end
12  end
13  if (fitness(bestCandidate) > fitness(sBest))
14          sBest ← bestCandidate
15  end
16  tabuList.push(bestCandidate)
17  if (tabuList.size > maxTabuSize)
18          tabuList.removeFirst()
19  end
20 end
21 return sBest
```

Route Selected using Tabu Search



Thank You!