B. Tech. Project
Search Algorithms for E-commerce

(Mentor : Prof. Praveen Paruchuri)
Submitted By**:**
Aditya Raj (201401075)
Aravind Ravishankar(201401121)

Motivation

To survive in today's market E commerce companies need to cut down their delivery costs while satisfying customer's requirements. Thus finding optimal delivery routes is a major and interesting problem to solve.

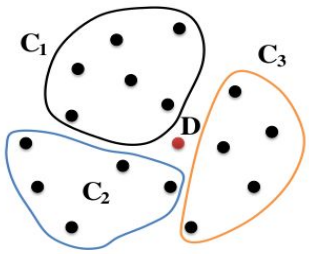
Problem Statement

- To identify an efficient routing between a set of storage depots to various customer locations satisfying their respective demands and time constraints.
- The goal is to minimize the total distance from all the depots respecting those constraints.

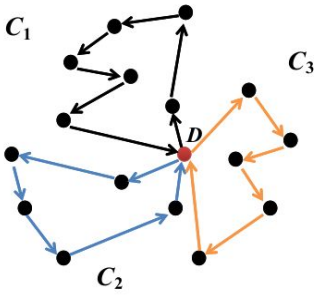
Algorithm: Beam Search based Approximate Algorithm

- **Why Beam Search?**
    - In the real world scenario the set of customers can be very large.
    - Optimal Algorithms like best first search have space issues since exploring all nodes is not efficient.
    - Beam search solves the issue and finds an approximate optimal solution.
    - We also use local search algorithms on top of the solution found by beam search to further improve the solution.

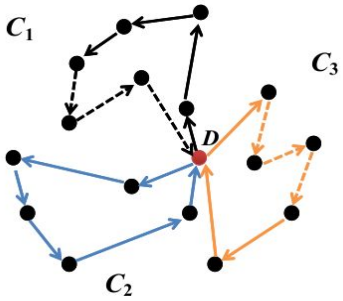
- **Algorithm Phase-1: K means Clustering**
    - Divide the set of customers into disjunctive clusters using K means clustering.
    - Each cluster will be assigned one delivery vehicle respecting its capacity constraints.



- **Algorithm Phase-2: Beam Search**
  - Determine a feasible route for the vehicle in each cluster using Beam search algorithm.
  - The time windows of the customers must be satisfied.



- **Algorithm Phase-3: Local Search**
  - Apply local search algorithms to further improve the solutions found using beam search. (covered in detail later)

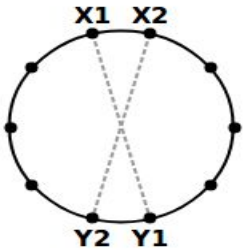


- **Error: Relaxed time windows**
  - Solution not found for many cases for strict time windows.
  - Introduced Error as a measure of time window violation, which depends on customer priority.
  - The algorithm now finds a solution even when the time windows are not satisfied, but with some error.

- **Extension to multiple depot locations**
    - Sorted customers based on 3 different heuristics:
        - Earliest time and Latest time and Priority
        - Distance
    - Assigned sorted customers to their nearest depot making sure that the total customer demand does not exceed the depot total capacity.

Local Search Analysis

- **2-OPT algorithm**
    - Given any route and 2 different pairs of consecutive nodes in the route, it can be shortened by exchanging 2 links (hence 2-OPT)
    - If the condition, $d(X1,\ Y1) + d(X2,\ Y2) < d(X1,\ X2) + d(Y1,\ Y2)$ holds and the time windows are satisfied, then we remove links X1-X2 and Y1-Y2 and replace with X1-Y1 and X2-Y2.

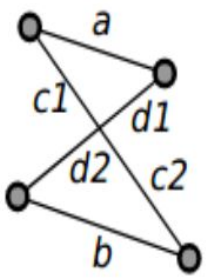


- **Observation**
    - Since our distance metric is Euclidean, the tour without intersections would be shorter than the tour with intersections

      $a < c1 + d1$

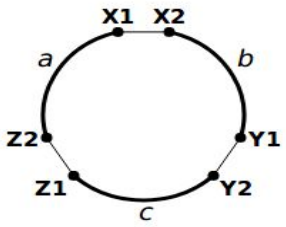
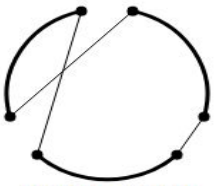
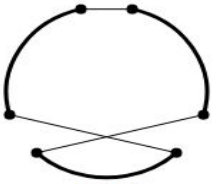
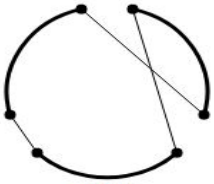
      $b < c2 + d2$

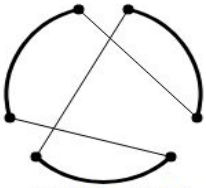
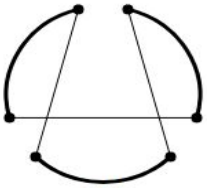
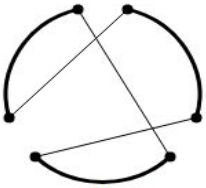
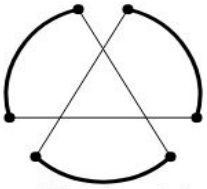
      $a + b < (c1 + c2) + (d1 + d2)$



- **Multiple Swaps: 3-OPT**

- In 2-OPT move we remove 2 links from cyclic tour, thus obtaining 2 open segments, which we manipulate and combine them to get a new tour.
- In 3-OPT we remove 3 links, obtaining 3 segments to manipulate.
- This gives us 8 combinations including the tour identical with the initial one.

<table>
<tr><th colspan="3">equals original tour</th></tr>
<tr><td colspan="3">case = 0</td></tr>
<tr><td colspan="3"> 
</td></tr>
<tr><th colspan="3">equivalent to a single 2-opt move</th></tr>
<tr><td>case=1</td><td>case=2</td><td>case=3</td></tr>
<tr><td> 
a'bc (=ac'b')</td><td> 
abc'</td><td> 
ab'c</td></tr>
<tr><th colspan="3">equivalent to two subsequent 2-opt moves</th></tr>
</table>

<table>
<tr><th colspan="3">equivalent to two subsequent 2-opt moves</th></tr>
<tr><td>case=4</td><td>case=5</td><td>case=6</td></tr>
<tr><td> 
ab'c' (=a'cb)</td><td> 
a'b'c</td><td> 
a'bc'</td></tr>
<tr><th colspan="3">equivalent to three subsequent 2-opt moves</th></tr>
<tr><td colspan="3">case=7</td></tr>
<tr><td colspan="3"> 
a'b'c' (=acb)</td></tr>
</table>

- **Why 3-OPT?**
  - Each 3-OPT move is either identical with 2-OPT move of is equal to a sequence of two or three 2-OPT moves
  - It is possible that there exists a sequence of 2-OPT moves that improves the tour but it begins with 2-OPT move that increases the length of the tour.
  - This sequence is not achieved when we use 2-OPT only because of that initial "bad" move.

- **Efficiency Issue**
  - Any K-OPT(performing K swaps) is better than (K-1)-OPT(performing K-1 swaps)
  - But Complexity for K-OPT for finding one single improvement is $O(n^k)$.
  - So, can we improve the efficiency of K-OPT?

- **First Improvement**
  - If for a given customer X1, we previously did not find any improving move and it still has the same neighbours, then chances that we will find an improvement now is small.
  - We use special flag for each of the customers called don't look bits. Initially all the flags are turned off, which means we allow searching.
  - If the search for improvement from customer X fails then the bit for that customer is turned on. If a move is performed which involves customer X, then the flag is turned off.
  - Now, when we are searching for candidates for customer Y, we skip all customers with their flag turned on.

- **Second Improvement**
  - Suppose that for some X1, X2, Y1, Y2:

    $$d(X2,\ Y2) \geq d(X1,\ X2)$$

    $$d(X1,\ Y1) \geq d(Y1,\ Y2)$$

  - But, for 2-OPT to hold,

    $$d(X1,\ Y1) + d(X2,\ Y2) < d(X1,\ X2) + d(Y1,\ Y2)$$

  - Therefore, one of the below 2 conditions must hold,

    $$d(X2,\ Y2) < d(X1,\ X2)$$

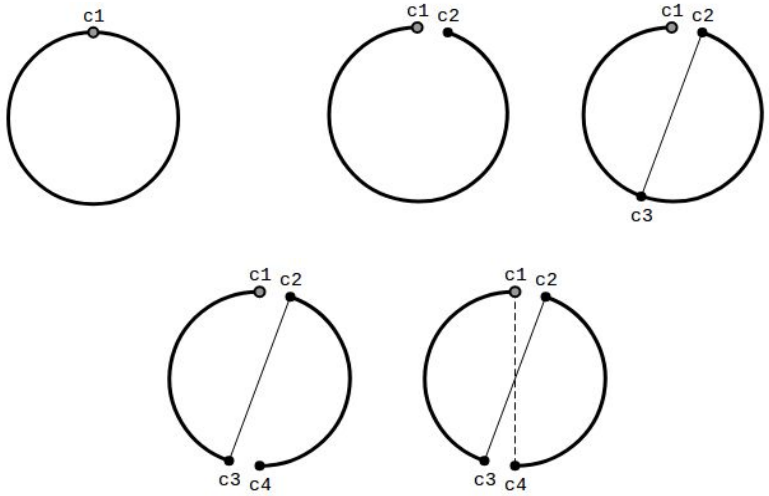
$$d(X1,\ Y1) < d(Y1,\ Y2)$$

- Let us analyze the first condition

$$d(X2,\ Y2) < d(X1,\ X2)$$

- So In 2-OPT, for a given vertex X1, considering it's neighbour X2, we can search around X2 for vertices Y2 which are closer than d(X1,X2) (Basically in a fixed radius neighbourhood) and accept the first improving 2-exchange move.

- **Sequential Moves**
  - Consider a sequence of 2-OPT moves
  - Start from certain customer C1 on tour.
  - Remove link between C1 and one of its tour neighbours, C2
  - Add link between C2 and some other customer C3
  - Remove link between C3 and its neighbor C4.
  - Add link between C4 and the first customer C1, to close the tour.



  - Note that steps 1 and 2 have the same scheme: remove a link between a city and it's route neighbor and then add link between this neighbor and some other city. Each step exchanges two links.
  - This can be done similarly for 3-opt as well.
  - Basically this is to show that every k-opt swap can be expressed as a sequence of moves where each move is equivalent to removing a link between a city and its tour neighbour and then add a link between this neighbour and some other city.

- So in K-OPT should I consider all possible K sequential moves(there are totally $O(n^k)$ moves) or is there a better solution?

- **Improving move condition**
  - A move is improving when it is valid and it improves a tour. Any k-opt move that improves a tour must fulfill the condition: Sum of lengths of links removed from tour must be greater than sum of links added to route.
  - In other words sum of links removed from tour must be greater than sum of links added to route.
  - Let us take a sequential move of a K-swap solution. Define

    g1 = distance(c1, c2) - distance(c2, c3) # gain from step 1

    g2 = distance(c3, c4) - distance(c4, c5) # gain from step 2

    g3 = distance(c5, c6) - distance(c6, c7) # gain from step 3

    and so on .. $G_k = \sum_{i=1}^{k} g_i$

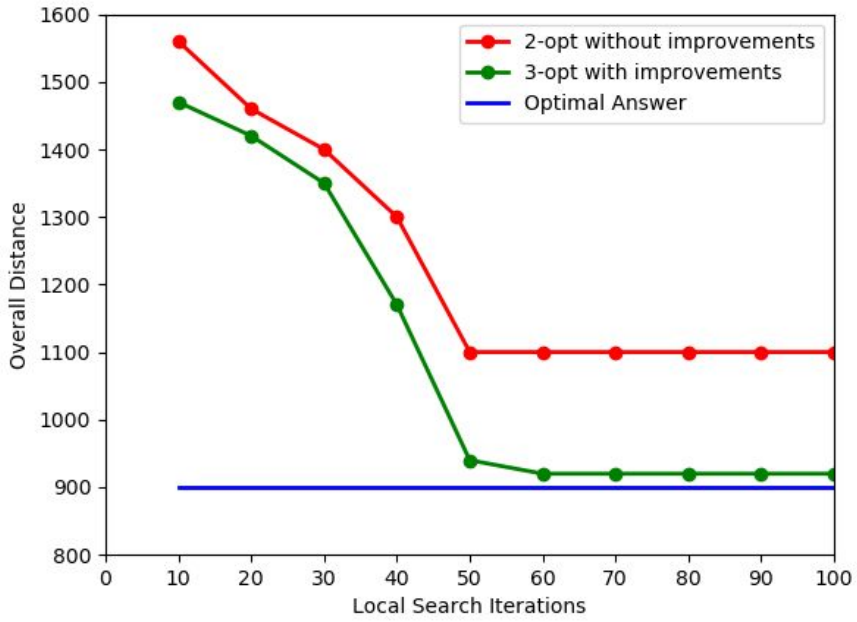
  - For solution to be improving $G_k > 0$
  - Although some of g1, g2, g3... may be negative, when this *sum* of numbers is positive then the move is an improving move

- **Observations**
  - We should note that sequential moves are *cyclic*, we can start from any step of move and apply them one by one, until we make them all.
  - So is there a specific order in which we can process these sequential moves instead of checking all possible orders for a given set of sequential moves?
  - Theorem: If a sequence of numbers has a positive sum, there is a cyclic permutation of these numbers such that *every partial sum* is positive.
  - In particular, then, since we are looking for sequences of gains g_i's that have positive sum, *we need only consider sequences of gains whose partial sum is always positive*. This gain criterion enables us to reduce enormously the number of sequences we need to examine!
  - Thus we found a huge reduction in the number of moves we have to check for a K-opt solution

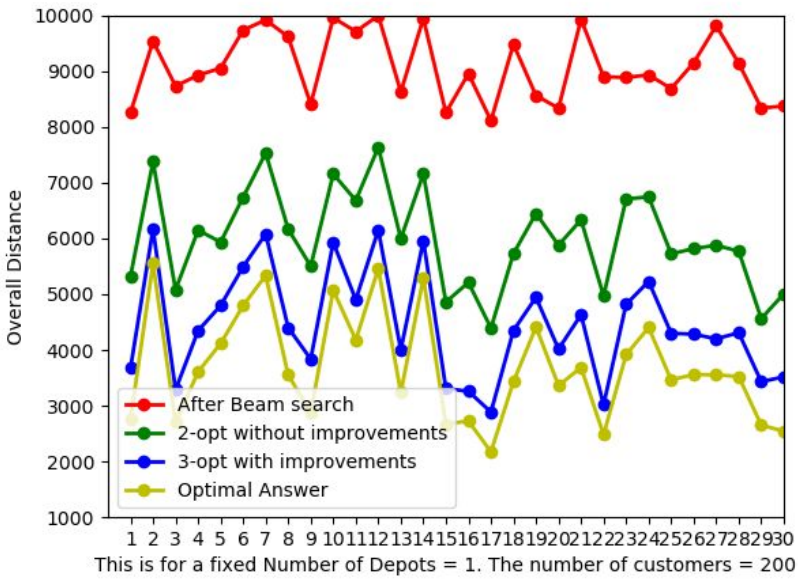
- Therefore during process of building a sequential move we check partial sum of gains. If this sum remains positive before the last, closing step, then a sequence of exchanges is *promising* and we can continue, even if it is not valid move now.

Results

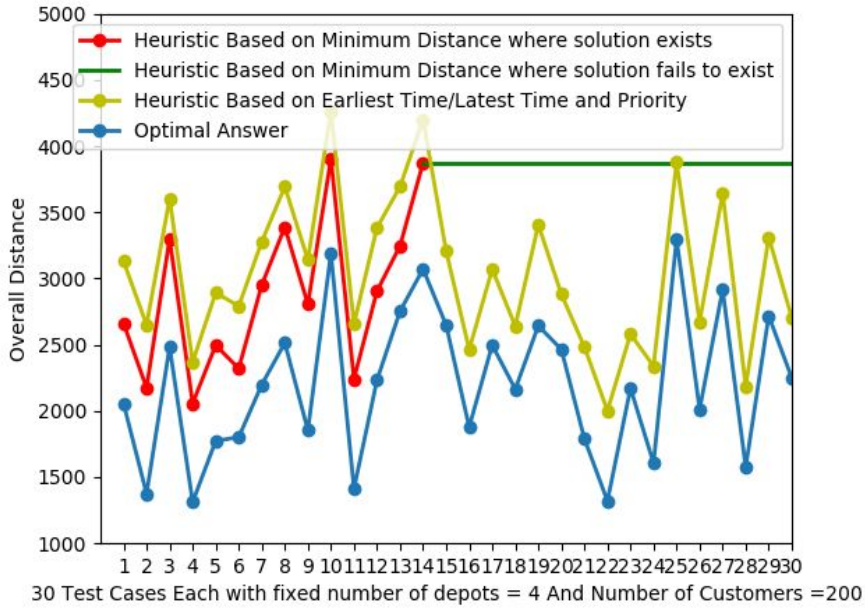
- **Local Search Improvement vs the number of iterations**
  - This is for one depot and 200 customers
  - As the iterations increase the local search solution improves but after a point it gets stuck at local minimum
  - It's clear that 3-OPT is better than 2-OPT.



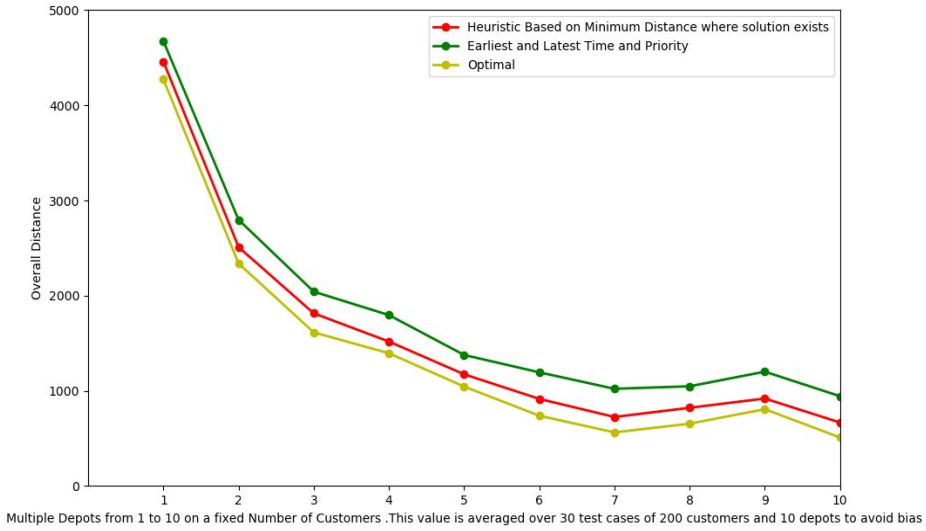
- **Local Search Improvement on various test cases**
  - The gap between beam search solution and 2-OPT solution is much more than the gap between the 3-OPT and 2-OPT solution



- **Heuristics Performance Comparison on various test cases**
  - In the heuristic criteria of minimum distance we just assigned based on minimum distance.
  - Even though it performs better than other heuristics, it fails to find a solution on certain cases highlighted by green



- **Heuristics Performance Comparison**
  - On an average the distance reduces by a factor of the number of depots



- **Local Search Time Performance Comparison**
  - Since 3-opt is $O(n^3)$ the general observation is that as number of customers double the time factor increase proportionately.
  - Also notice that the cyclicity observation decreased the time by a larger amount since for every cyclic permutation we reduced by a factor of 3 if there existed a solution
  - Other improvements were minor and did not much effect on the time