Optimization of Delivery Time and Route for Delivery Personnel

Team Members :

- Sahid Ali Raza 12200220047 (Class Roll No: 49)
- Shweta Yadav 12200220001 (Class Roll No: 66)
- Soumyadip Ghosh 12200220026 (Class Roll No: 24)
- Sheetal Sinha 12200220015 (Class Roll No: 65)

MENTOR – SUBHASHREE BASU

INTRODUCTION

This presentation provides a comprehensive project report on the development of an online delivery system aimed at providing fastest delivery services to the customers doors.

This project focuses on leveraging advanced technologies and data analysis techniques to improve the accuracy and reliability to find the shortest and fastest route for the delivery within minimum amount of time.

OUTLINE

★ PROJECT OBJECTIVE

★ IMPORTANCE

★ FLOW OF WORK

★ INTRODUCTION TO ALGORITHMS

★ RESULT ANALYSIS

Project Objective

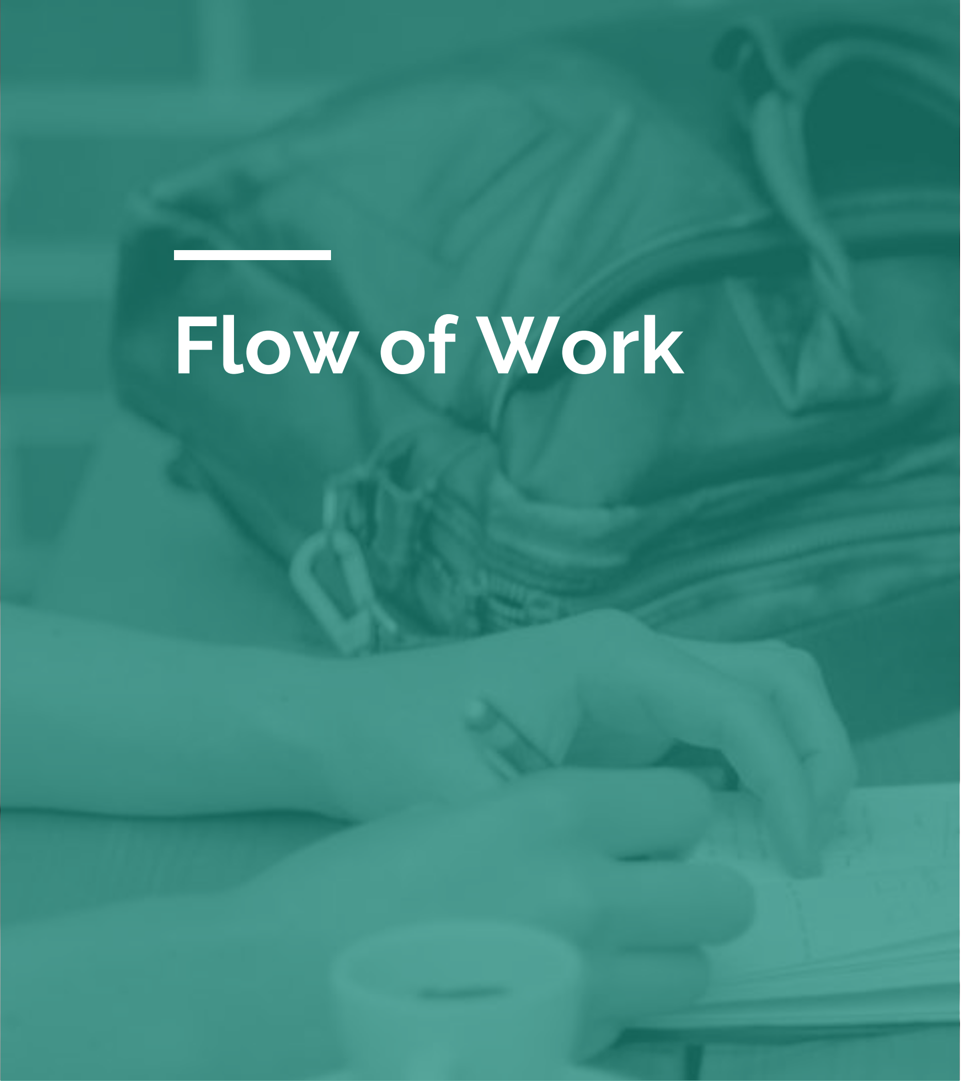
The following are the Objectives of this project :

- The main Objective is to Reduced Delivery Time and Route Optimization for a Dedicated Delivery personal.
- This Project Includes the Implementation Of Technology and Strategies to Streamline the Delivery process , Reduces Time and Route Optimization.
- This not only leads to Reduced Delivery times and Optimized Route but also improve the allover Performance and Job Satisfaction of Dedicated Delivery Personal , ultimately benefiting both the Organization and Customer.

Importance

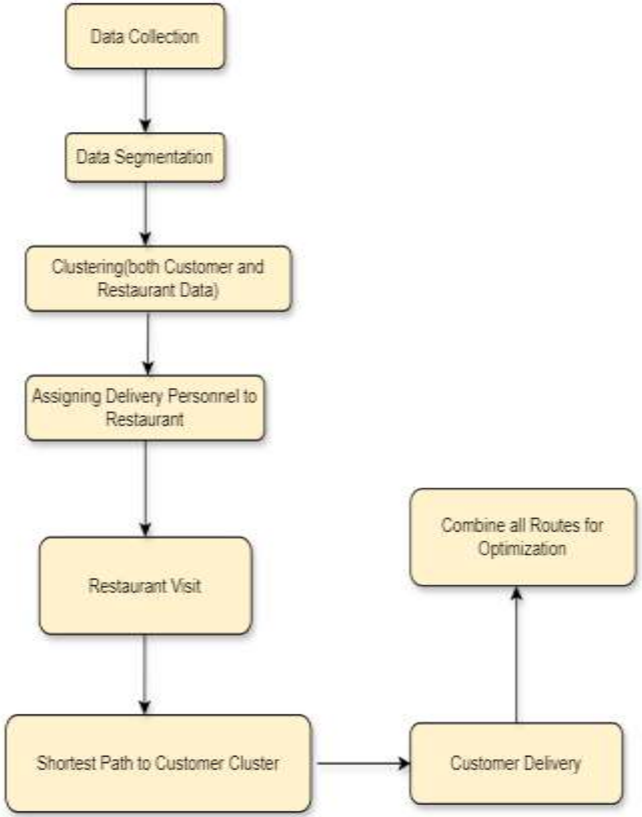
Ensuring client happiness requires prompt delivery. Modern consumers want their orders delivered quickly, and prompt delivery is essential to fulfilling those expectations. Because customers are more inclined to make extra purchases when they know their things will come swiftly, it can boost sales and encourage impulsive purchases.

Businesses may see an increase in profitability by streamlining logistics and cutting operating expenses through the use of efficient delivery systems.



Flow of Work

FLOWCHART


Data Collection
Data Segmentation
Clustering(both Customer and Restaurant Data)
Assigning Delivery Personnel to Restaurant
Restaurant Visit
Shortest Path to Customer Cluster
Customer Delivery
Combine all Routes for Optimization

DATA COLLECTION

01

In the project's early stages, Collect the raw data containing Customer and Restaurant.

DATA SEGMENTATION

02

Segregate the raw data set into two distinct datasets : one for the Customer and another for the restaurants.

CLUSTERING

03

Employ a Clustering algorithm based on location and locality to group customers dataset into clusters based on their geographical proximity and apply the same clustering methodology for the restaurant dataset.

ASSIGNING DELIVERY PERSONNEL TO RESTAURANT

04

Determining the shortest path from the delivery boy's current location to the nearest restaurant in each cluster minimizes travel time and distance.

RESTAURANT VISIT

05

Directing the delivery boy to visit the nearest restaurant within each cluster streamlines the order pickup process. It ensures timely collection of orders, minimizing delays and optimizing resource utilization.

Shortest Path Computation (Customer Delivery)

06

Calculating the shortest path from visited restaurants to each customer within the respective cluster optimizes the delivery route. It minimizes travel time and distance, ensuring prompt and efficient order fulfillment..

Customer Delivery

07

Delivering orders to customers within each cluster following optimized routes enhances customer satisfaction and loyalty

Combine all the routes within the clusters and all the inter-cluster connections to find the optimized Route

08

Consolidating optimized routes from individual clusters and inter-cluster connections generates the final delivery route

Introduction to Algorithm

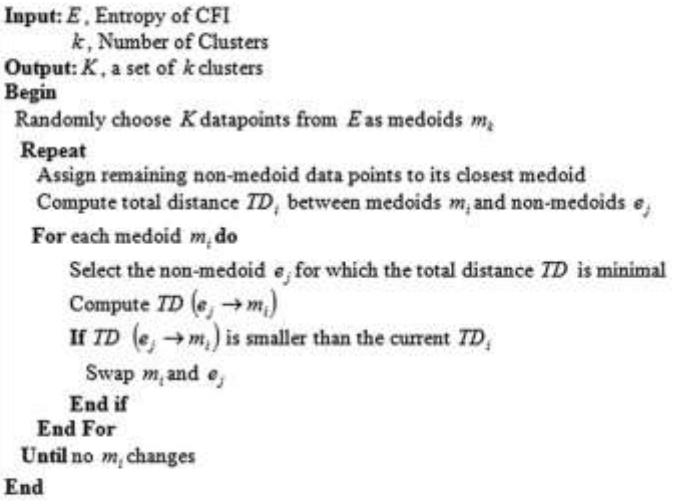
We gained a lot of knowledge about many algorithms in order to select the best ones for our project, including K-Means ,K-Medoids ,Birch, DB SCAN and the Nearest Neighbor method, Insertion Algorithm, Last Mile , First Mile, Time-based Assignment, Proximity Delivery Assignment, assigning delivery boys to clusters of customers, Vehicle routing which we used to solve the Traveling Salesman Problem.

The various algorithms are explained further :

K-MEDOIDS :

- ➔ K-medoids, also known as partitioning around medoids (PAM), is a popular clustering algorithm that groups k data points into clusters by selecting k representative objects within a dataset.
- ➔ The k-means clustering algorithm uses centroids.
- ➔ K-medoids is an alternative clustering algorithm that uses medoids instead.

Pseudocode:

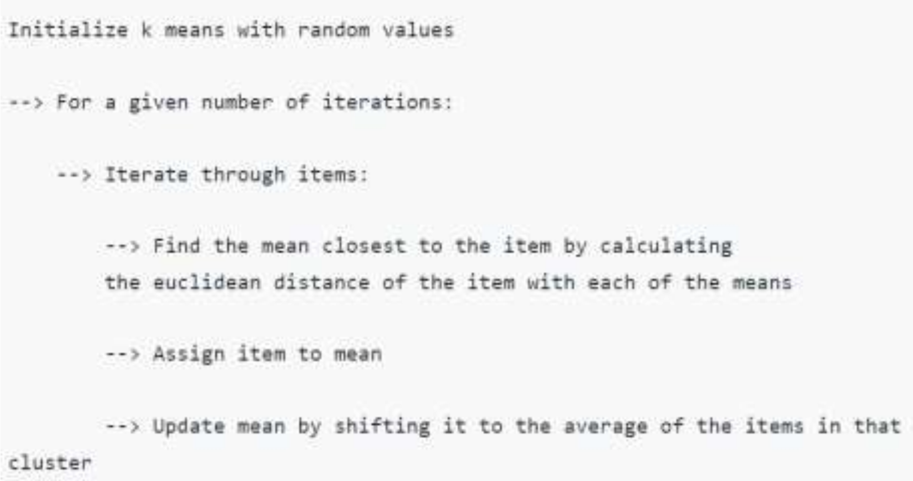


```
Input: $E$, Entropy of CFI
       $k$, Number of Clusters
Output: $K$, a set of $k$ clusters
Begin
  Randomly choose $K$ datapoints from $E$ as medoids $m_i$
  Repeat
    Assign remaining non-medoid data points to its closest medoid
    Compute total distance $TD_i$ between medoids $m_i$ and non-medoids $e_j$
    For each medoid $m_i$ do
      Select the non-medoid $e_j$ for which the total distance $TD$ is minimal
      Compute $TD\ (e_j \rightarrow m_i)$
      If $TD\ (e_j \rightarrow m_i)$ is smaller than the current $TD_i$
        Swap $m_i$ and $e_j$
      End if
    End For
  Until no $m_i$ changes
End
```

K-MEANS :

- ➔ K-means clustering is one of the simplest and popular unsupervised machine learning algorithms.
- ➔ Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.
- ➔ The K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.
- ➔ The 'means' in the K-means refers to averaging of the data; that is, finding the centroid.

Pseudocode:

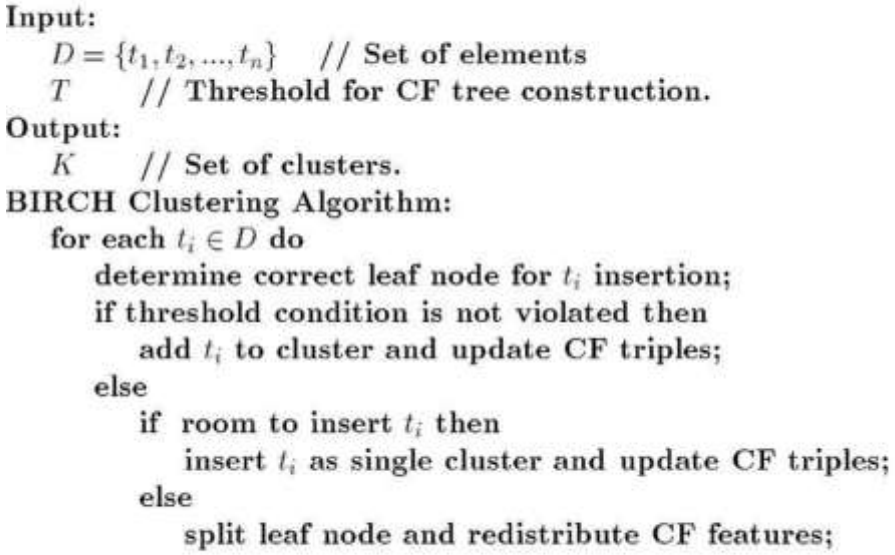


```
Initialize k means with random values

--> For a given number of iterations:

    --> Iterate through items:

        --> Find the mean closest to the item by calculating
        the euclidean distance of the item with each of the means

        --> Assign item to mean

        --> Update mean by shifting it to the average of the items in that
cluster
```

BIRCH ALGORITHM :

- ➔ Balanced Iterative Reducing and Clustering using Hierarchies also know as BIRCH is a clustering algorithm using which we can cluster large datasets by first producing a brief summary about the dataset that preserves information as much as possible.
- ➔ This brief summary is then clustered instead of clustering the whole dataset.
- ➔ Generally, there are two types of attributes involved in the data to be clustered: metric and nonmetric.
- ➔ BIRCH only considers metric attributes, which is one of its major drawback.
- ➔ A metric attribute is any attribute whose values can be represented in Euclidean space.

Pseudocode:



```
Input:
    $D = \{t_1, t_2, ..., t_n\}$    // Set of elements
    $T$    // Threshold for CF tree construction.
Output:
    $K$    // Set of clusters.
BIRCH Clustering Algorithm:
    for each $t_i \in D$ do
        determine correct leaf node for $t_i$ insertion;
        if threshold condition is not violated then
            add $t_i$ to cluster and update CF triples;
        else
            if  room to insert $t_i$ then
                insert $t_i$ as single cluster and update CF triples;
            else
                split leaf node and redistribute CF features;
```

DB SCAN

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is ideal for spatial data analysis, identifying clusters of various shapes and sizes without needing a predetermined number of clusters.
- The algorithm uses epsilon (eps) to define the radius for searching neighboring points and minimum points (MinPts) to specify the minimum points required to form a dense region.
- These are points with at least MinPts within their eps radius, serving as central points of clusters.
- These are points within the eps radius of core points but with fewer than MinPts within their own eps radius, thus lying on the cluster's periphery.
- Points that are neither core nor border points, representing outliers that do not belong to any cluster, effectively identifying noise in the data.

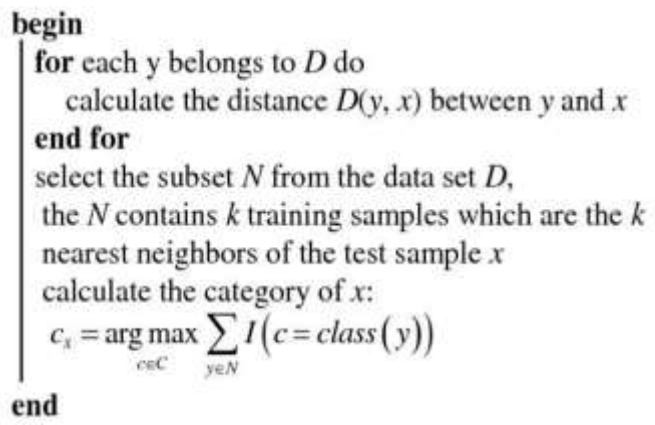
Pseudocode:

- **Initialize**: Start with an unvisited point.
- **Expand**: For each unvisited point, find all points within the epsilon (eps) radius.
- **Core Point Check**: If the number of points within eps is greater than or equal to MinPts, form a new cluster with this point as the core.
- **Cluster Expansion**: Recursively add all directly density-reachable points (within eps) to the cluster.
- **Mark Points**: Label points as core, border, or noise based on their reachability and density criteria.
- **Repeat**: Continue until all points are visited and appropriately labeled.

Travelling Salesman Problem(TSP) Using Nearest Neighbor Problem :

- ➔ The Travelling Salesman Problem (TSP) is an optimization problem that aims to find the shortest route between a set of cities.
- ➔ There is no known algorithm that can solve it for all possible inputs in polynomial time.
- ➔ As the number of cities increases, the number of possible tours becomes exponentially large, making exhaustive search for the optimal solution computationally infeasible.
- ➔ To find near-optimal solutions efficiently, several solutions have been developed, including Approximation Algorithms

Pseudocode:

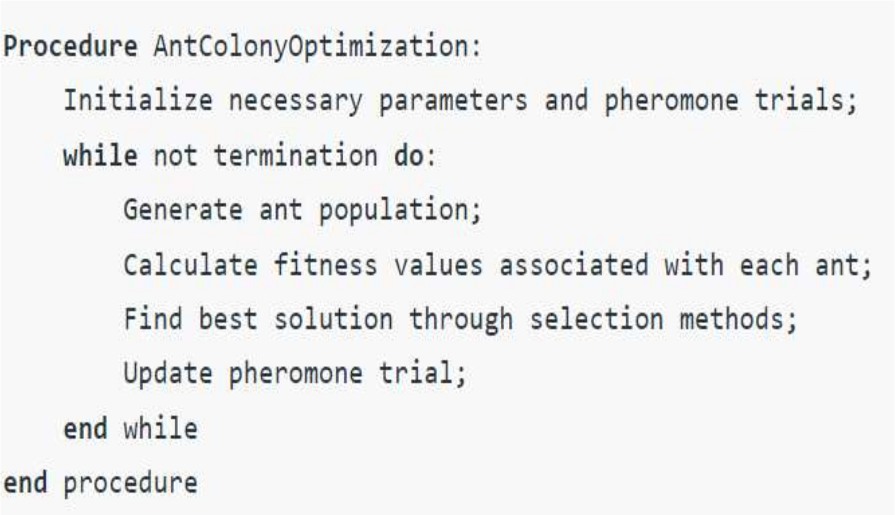


```
begin
  for each y belongs to D do
     calculate the distance $D(y, x)$ between $y$ and $x$
  end for
  select the subset $N$ from the data set $D$,
  the $N$ contains $k$ training samples which are the $k$
  nearest neighbors of the test sample $x$
  calculate the category of $x$:
  $c_x = \arg\max_{c \in C} \sum_{y \in N} I\left(c = class\left(y\right)\right)$
end
```

Ant Colony Optimization(ACO) :

- ➔ Ant colony optimization (ACO) is an optimization algorithm which employs the probabilistic technique and is used for solving computational problems and finding the optimal path with the help of graphs.
- ➔ This technique is derived from the behavior of ant colonies.
- ➔ To get the food, ants use the shortest path available from the food source to the colony.
- ➔ Now ants going for the food secret the pheromone and other ants follow this pheromone to follow the shortest route.

Pseudocode:

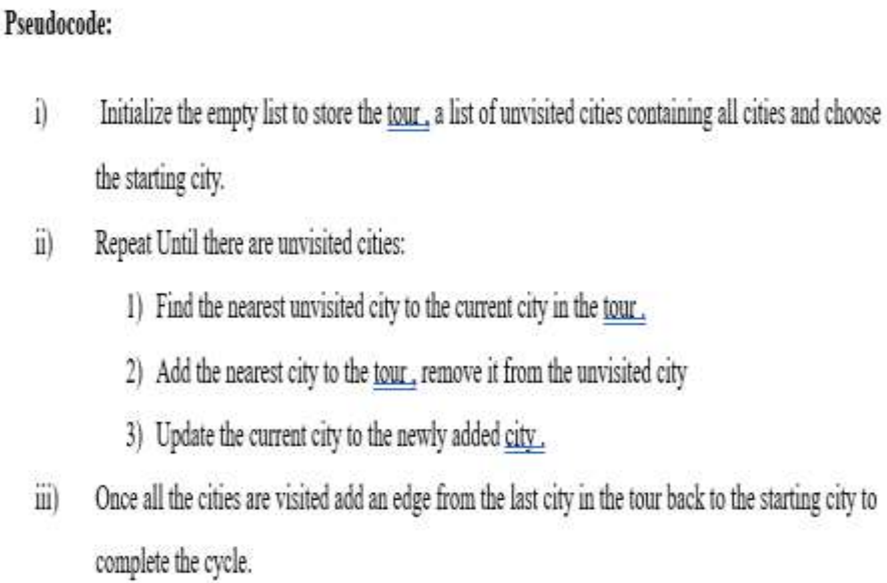


```
Procedure AntColonyOptimization:
    Initialize necessary parameters and pheromone trials;
    while not termination do:
        Generate ant population;
        Calculate fitness values associated with each ant;
        Find best solution through selection methods;
        Update pheromone trial;
    end while
end procedure
```

Insertion Algorithm:

Insertion Algorithm starts with an Empty route and Repeatedly inserts each city into partially constructed route at the position where it would lead to the least increase in total cost i.e Iteratively selects a city not yet in the route and insert it at the position where it minimize the total distance. The main goal of this algorithm is to minimize the total distance travelled while visiting each city exactly once and returning to the starting point.

Time complexity : O(n^3), where "n" is the number of cities.



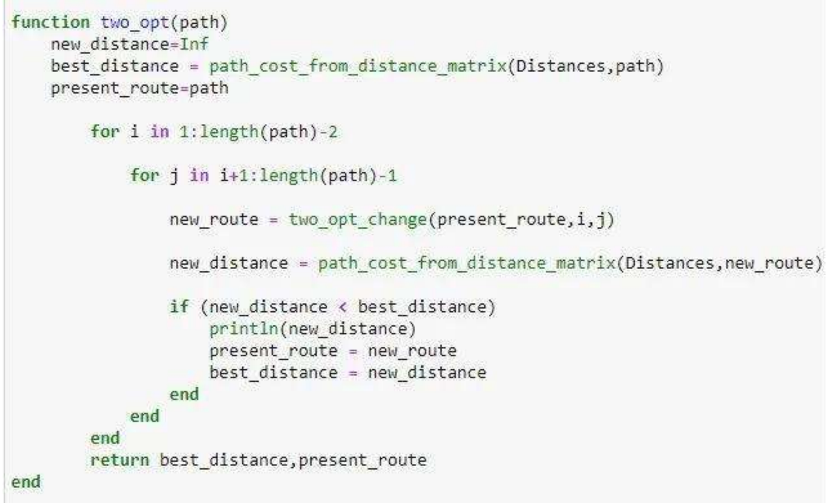
Pseudocode:

```
i)    Initialize the empty list to store the tour , a list of unvisited cities containing all cities and choose
      the starting city.
ii)   Repeat Until there are unvisited cities:
        1) Find the nearest unvisited city to the current city in the tour .
        2) Add the nearest city to the tour , remove it from the unvisited city
        3) Update the current city to the newly added city .
iii)  Once all the cities are visited add an edge from the last city in the tour back to the starting city to
      complete the cycle.
```

The 2-opt Algorithm :

- ➔ The 2-Opt algorithm removes two edges from the graph and then reconstructs the graph to complete the cycle.
- ➔ There is always only one possibility in adding two unique edges to the graph for the completion of the cycle.
- ➔ If the new tour length is less than the previous one, it is kept; otherwise, it is rejected.
- ➔ The 2-Opt heuristic is a simple improvement heuristic for the Traveling Salesman Problem.
- ➔ Time complexity : **O(n^3).**

Pseudocode:



```
function two_opt(path)
    new_distance=Inf
    best_distance = path_cost_from_distance_matrix(Distances,path)
    present_route=path

        for i in 1:length(path)-2

            for j in i+1:length(path)-1

                new_route = two_opt_change(present_route,i,j)

                new_distance = path_cost_from_distance_matrix(Distances,new_route)

                if (new_distance < best_distance)
                    println(new_distance)
                    present_route = new_route
                    best_distance = new_distance
                end
            end
        end
        return best_distance,present_route
end
```

Andean Condor Algorithm

- The Andean Condor algorithm is a metaheuristic optimization technique inspired by the natural behavior of Andean condors, large South American birds known for their long-distance flight. The algorithm is designed to solve complex optimization problems by iteratively exchanging and updating solutions based on their compatibility with the overall objective function. It is particularly useful for solving integer programming problems and has been applied in various fields, including scheduling, logistics, and telecommunications.

Pseudo code

- Initialize a list of data points. (e.g. Randomization (Machine-Cell Assignment)).
- Evaluate each points using a similarity function (e.g.Exploration).
- Select the most similar points to each other cell.
- Repeat steps 2-3 until no more points can be paired.
- The algorithm continuously checks whether the generated solution adheres to constraints for both machines and parts assigned to cells.

Last Mile Delivery

- **Data Import**: Read pickup location data from a CSV file, including location identifiers, latitudes, and longitudes.
- **Clustering**: Apply K-means clustering to group pickup locations based on spatial proximity, facilitating efficient route planning.
- **Batching**: Organize pickups within each cluster into batches, adhering to predefined vehicle capacity constraints.
- **Route Optimization**: Use a greedy algorithm to optimize pickup routes within each batch by evaluating all permutations of location sequences and selecting the one that minimizes total distance traveled.
- **Output**: Print optimized routes for each batch, detailing the journey from the initial depot to each pickup location and back, along with the total optimized distance for each cluster.

Pseudocode:

```
Procedure LastMileDeliveryPickup(locations_file, num_clusters, vehicle_capacity):
    Read locations from locations_file
    Perform K-means clustering with num_clusters
    Divide pickup locations into batches based on vehicle_capacity
    For each cluster:
        For each batch in batches:
            Find the optimal pickup route:
                Generate all permutations of pickup locations within the batch
                Calculate the total distance for each permutation
                Select the permutation with the minimum total distance as the optimal route
            Print the optimized route for the batch
            Print the total optimized distance for the cluster
```

First Mile Delivery

- **Data Collection**: Gather information about packages to be shipped, including sender addresses, package sizes, and delivery deadlines.
- **Routing**: Determine the optimal route for the delivery vehicle based on factors such as package locations, traffic conditions, and delivery deadlines.
- **Load Balancing**: Distribute packages evenly among delivery vehicles to optimize efficiency and minimize delivery times.
- **Tracking**: Implement a system to track the real-time location and status of delivery vehicles and packages, providing customers with accurate delivery updates.
- **Customer Interaction**: Facilitate communication between delivery personnel and customers, allowing for updates, rerouting requests, or special delivery instructions.

Pseudocode:

```
Procedure FirstMileDelivery(packages_info):
    Initialize delivery_vehicles
    For each package in packages_info:
        Assign package to the nearest available delivery vehicle
    For each delivery_vehicle in delivery_vehicles:
        Determine optimal route based on package destinations, traffic conditions, and deadlines
        Load packages onto the delivery vehicle, ensuring load balancing
        Depart from depot and follow the planned route
        Update package statuses and delivery vehicle location in real-time
        Arrive at each package destination and complete delivery
        Return to depot after completing all deliveries
    Return delivery status and any relevant updates to the main system
```

Proximity Delivery Assignment

- **Preprocessing**: Customer data is preprocessed, including filling missing values and extracting coordinates and timestamps.
- **Clustering**: Utilize KMeans clustering to group customers based on geographical proximity, with clusters equal to available delivery boys.
- **Delivery Assignments**: Assign each delivery boy to a specific cluster or region to optimize delivery routes.
- **Route Calculation**: Calculate routes for each delivery by computing geodesic distances between delivery boy and customer locations, considering average delivery boy speed.
- **After-Hours Orders Handling**: Direct delivery boys to the restaurant location before proceeding with delivery for orders placed after operating hours, ensuring adherence to specified time windows.

Pseudocode:

```
Procedure ProximityDeliveryAssignment(customer_data, delivery_boys):
    Preprocess customer_data: fill missing values, extract coordinates, and timestamps
    Perform KMeans clustering on customer_data with clusters equal to number of delivery boys
    Assign each delivery boy to a cluster based on geographical proximity
    For each delivery boy:
        Calculate route for deliveries:
            For each customer in assigned cluster:
                Calculate geodesic distance between delivery boy's location and customer's location
                Estimate time required for delivery considering average delivery boy speed
        Handle after-hours orders:
            If order is placed after restaurant's operating hours:
                Direct delivery boy to restaurant location before proceeding with delivery
    Output detailed information for each delivery, including distance traveled and time taken
```

Time-based Assignment

- **Nearest Neighbor Search**: Utilizes this technique to find the closest restaurant for each delivery boy based on geographical proximity.
- **Time-based Assignment**: Considers the availability of the restaurant and the order time of the customer to assign delivery boys to customer clusters.
- **Greedy Approach**: Implements a strategy where delivery boys are iteratively assigned without global optimization, focusing on local decisions to handle the assignment process efficiently.
- **Customized Solution**: Orchestrates a tailored solution aimed at optimizing delivery logistics by dynamically balancing spatial and temporal constraints

Pseudocode:

```
Procedure DeliveryLogisticsOptimization(delivery_boys, restaurants, customers):
    For each delivery_boy in delivery_boys:
        Find nearest restaurant using Nearest Neighbor Search
    For each customer in customers:
        For each delivery_boy:
            If restaurant is available and delivery boy can reach the customer in time:
                Assign delivery boy to customer cluster using Time-based Assignment
    GreedyAssignment(delivery_boys, customers)  # Greedy approach for assignment
Procedure GreedyAssignment(delivery_boys, customers):
    For each customer in customers:
        Find nearest available delivery boy
        Assign customer to the delivery boy
    Output optimized delivery logistics balancing spatial and temporal constraints
```

Assigning delivery boys to clusters of customers:

- Utilizes K-means clustering for grouping customers based on proximity.
- Assigns delivery boys to the nearest cluster, enhancing route efficiency.
- Calculates distances between clusters and delivery boys to optimize resource allocation.
- Implements iterative processes to refine cluster assignments and delivery boy allocations.
- Optimizes route planning within each cluster to minimize travel distances.
- Systematically improves delivery efficiency by minimizing overall travel distances and optimizing resource allocation.
- Evaluates algorithm effectiveness based on delivery efficiency metrics and iteratively refines the approach.

Pseudocode

```
1. Initialize cluster centers randomly.
2. Repeat until convergence:
    a. Assign each customer to the nearest cluster.
    b. Update cluster centers based on assigned customers.
3. Assign delivery boys to clusters based on proximity.
4. Optimize routes within each cluster for delivery boys.
5. Evaluate and refine algorithm based on efficiency metrics.
```

Vehicle Routing:

- **Input Data**: Gather information about delivery locations, vehicle capacities, time constraints, and any other relevant constraints.
- **Route Initialization**: Determine an initial route plan for each vehicle, which may be based on heuristics or optimization algorithms.
- **Route Optimization**: Utilize optimization techniques such as genetic algorithms, ant colony optimization, or simulated annealing to refine the initial routes and minimize total distance or time traveled.
- **Constraint Handling**: Ensure that the optimized routes adhere to constraints such as vehicle capacity, time windows for delivery, and any other specific requirements.
- **Dynamic Updates**: Implement mechanisms to dynamically update routes in response to real-time changes, such as new orders, traffic congestion, or vehicle breakdowns.

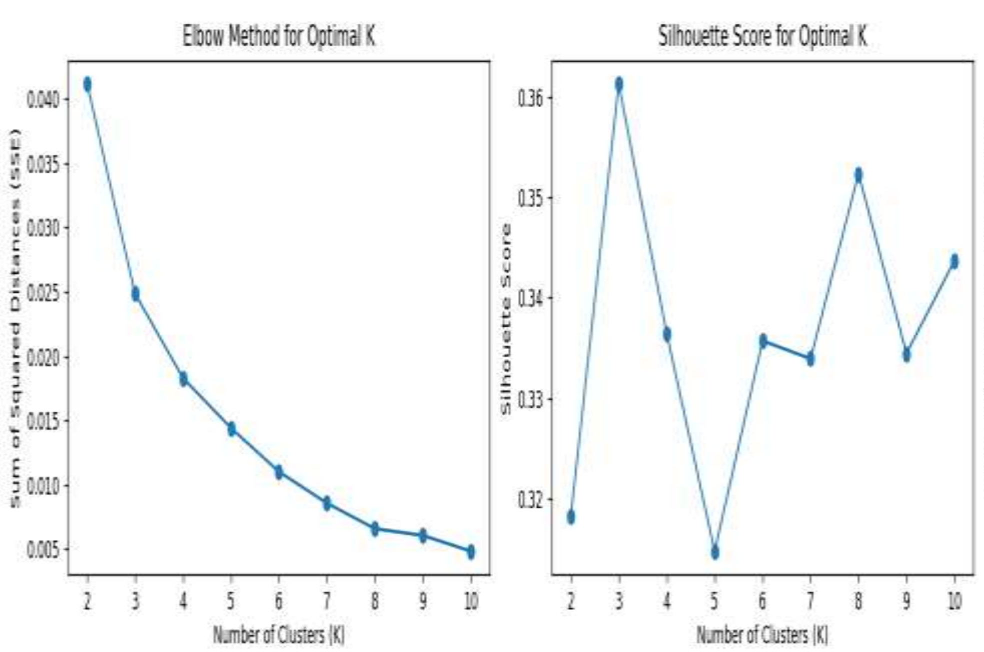
Pseudocode:

```
function can_serve(current_location, customer, vehicle):
    # Check if the vehicle can serve the customer based on capacity, time windows, etc.
    if vehicle.has_capacity_for(customer):
        if vehicle.is_within_time_window(current_location, customer):
            return True
        else:
            return False  # Customer not within vehicle's time window
    else:
        return False  # Vehicle does not have enough capacity for the customer
```

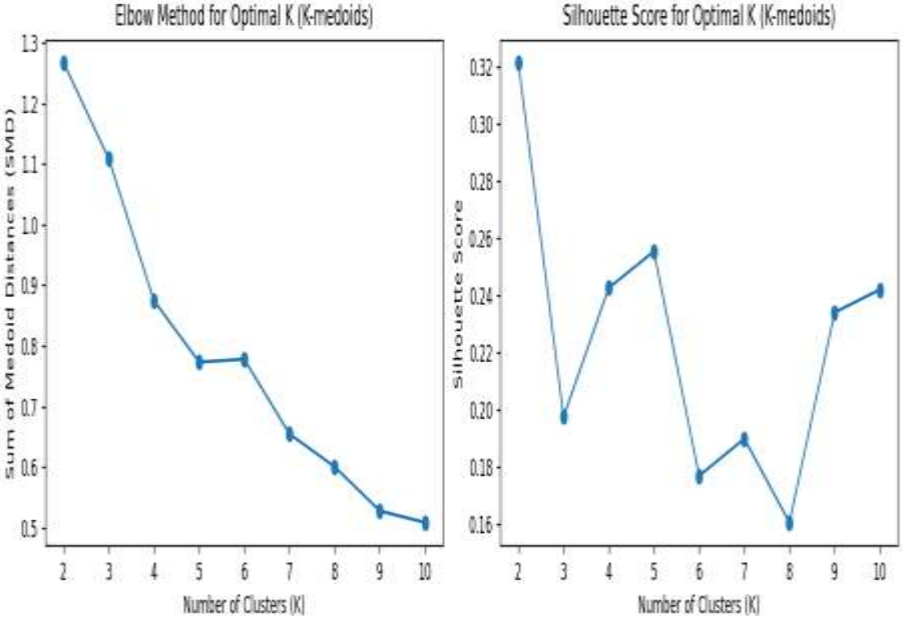
Implementation
08

We put all of the previously discussed algorithms into practice in order to generate outputs based on the data we had gathered and identify the best one.

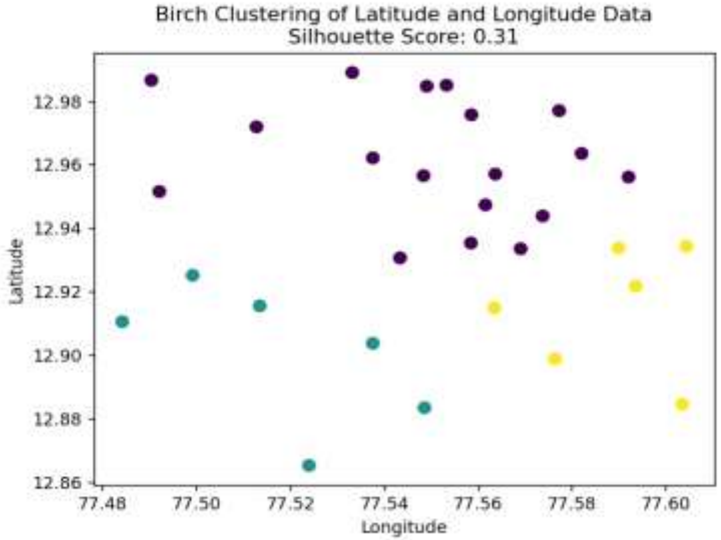
Outputs:



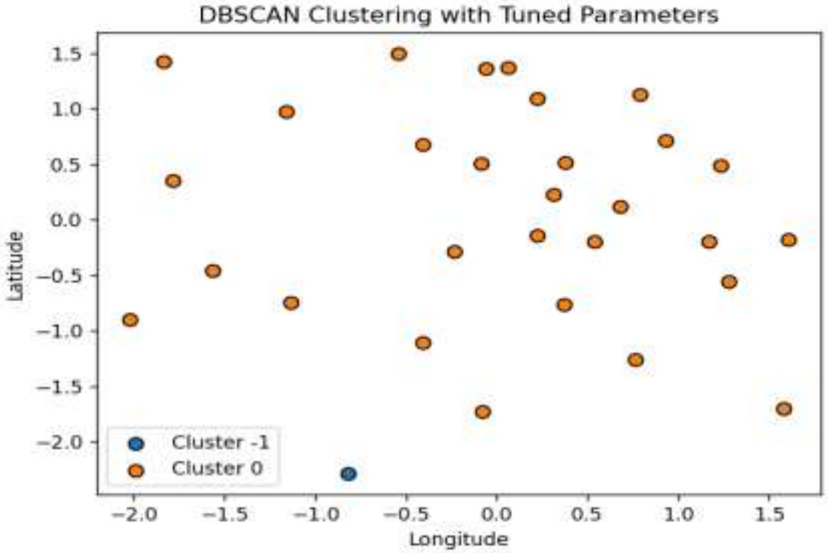
K-Means Algorithm



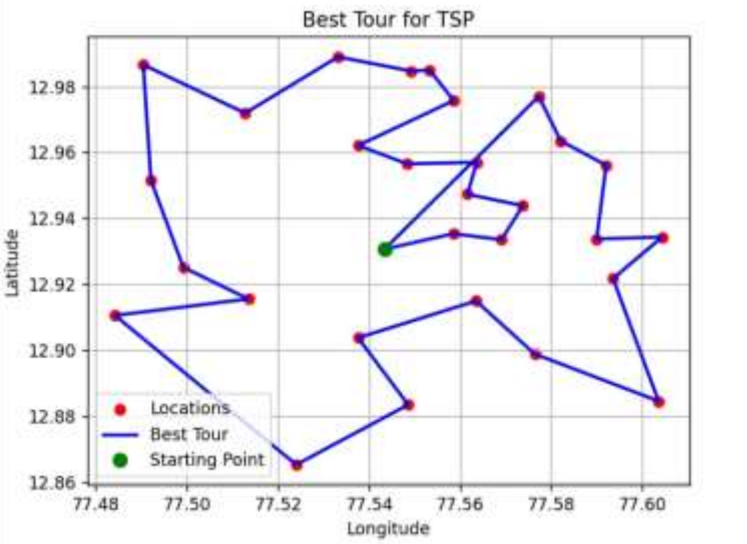
K-Medoid Algorithm:



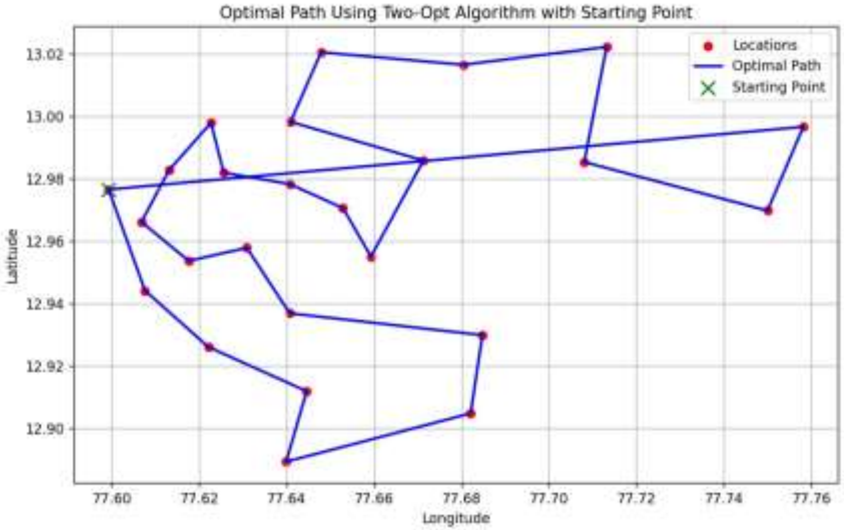
Birch Algorithm



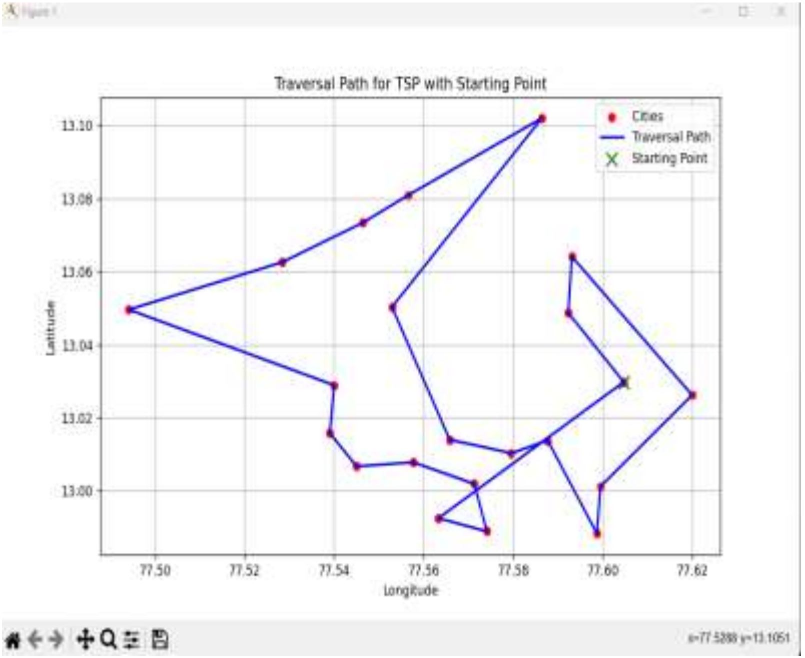
DBSCAN Algorithm



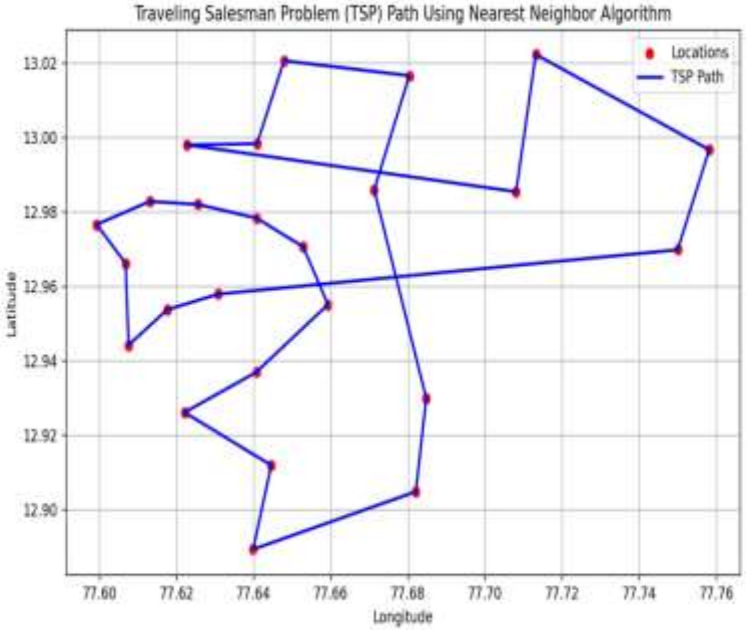
Ant Colony Algorithm



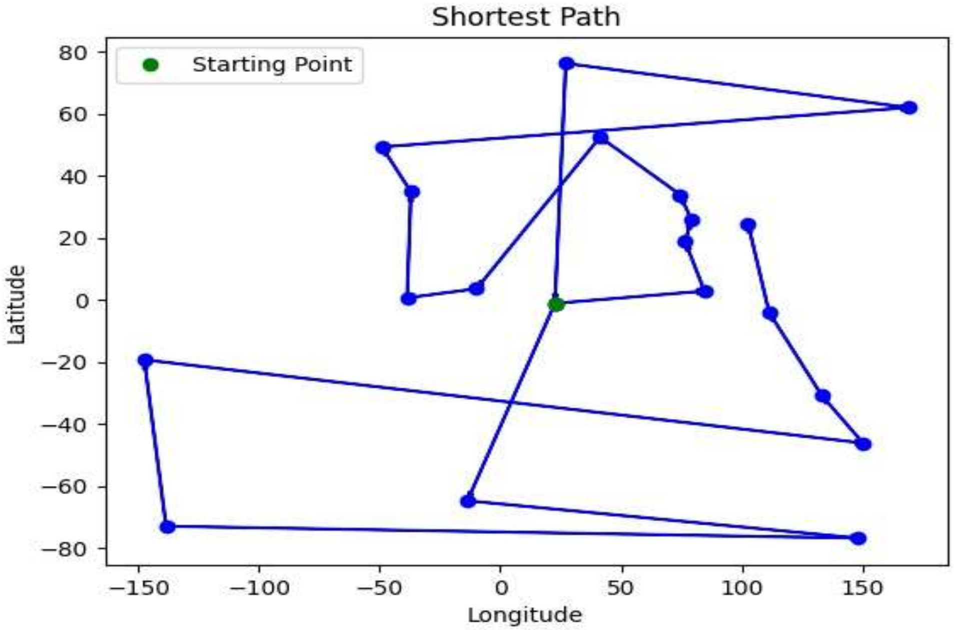
Optimal Path Using Two-Opt Algorithm With Starting Point



Insertion Algorithm



Nearest Neighbor Algorithm



Delivery Boy 1 traveled 7.65 kilometers to reach the restaurant.

Delivery Boy 2 traveled 5.08 kilometers to reach the restaurant.

Delivery Boy 3 traveled 5.76 kilometers to reach the restaurant.

Delivery Boy 1 is assigned to Cluster 1.

Delivery Boy 2 is assigned to Cluster 3.

Delivery Boy 3 is assigned to Cluster 2

Andean Condor Algorithm

Assigning delivery boys to clusters of customers

Vehicle at location (12.95404312, 77.56886512): Deliveries: Deliver to Customer 0 at location (12.9766, 77.5993) Deliver to Customer 1 at location (12.977, 77.5773) Deliver to Customer 3 at location (12.9473, 77.5616

Vehicle at location (12.91804385, 77.63768893): Deliveries: Deliver to Customer 2 at location (12.9551, 77.6593) Deliver to Customer 5 at location (12.9299, 77.6848

Vehicle at location (12.91192952, 77.63801917): Deliveries: Deliver to Customer 10 at location (12.8893, 77.6399) Deliver to Customer 22 at location (12.9119, 77.6446

Last Mile Delivery

Delivery Boy 1 traveled 8.83 kilometers to reach the restaurant. Delivery Boy 2 traveled 9.24 kilometers to reach the restaurant. Delivery Boy 3 traveled 9.91 kilometers to reach the restaurant.

Delivery Boy 1 traveled 4.14 kilometers to reach customer (Latitude 12.9766, Longitude 77.5993) and took 17.30 minutes.

Delivery Boy 1 traveled 2.70 kilometers to reach customer (Latitude 12.977, Longitude 77.5773) and took 15.38 minutes.

Boy 3 traveled 5.30 kilometers to reach customer (Latitude 12.9551, Longitude 77.6593) and took 20.29 minutes. Delivery Boy 1 traveled 1.09 kilometers to reach customer (Latitude 12.9473, Longitude 77.5616) and took 13.23 minutes. Delivery Boy 1 traveled 3.82 kilometers to reach customer (Latitude 12.985, Longitude 77.5533) and took 5.09 minutes

Proximity Delivery Assignment

Restaurant 1 to Customer: ([13.0014746, 77.6343764], [12.9983, 77.6409]) Restaurant 2 to Customer: ([12.97532366, 77.60504678], [12.9766, 77.5993]) Restaurant 3 to Customer: ([12.95954933, 77.64591224], [12.9706, 77.6529]) Restaurant 4 to Customer: ([12.91285665, 77.6352223], [12.9119, 77.6446] Restaurant 5 to Customer: ([12.95742391, 77.51870312], [12.9719, 77.5128]) Restaurant 6 to Customer: ([12.78641171, 77.74583794], [12.9048, 77.6821]) Restaurant 7 to Customer: ([12.9120841, 77.63851304], [12.9119, 77.6446]

First Mile

Vehicle 1 Route: ['Depot’]

Vehicle 2 Route: ['Depot', 'Customer 1’]

Vehicle 3 Route: ['Depot', 'Customer 3', 'Customer 2’]

Vehicle Routing

Result Analysis

- The K-means Clustering Algorithm achieved an accuracy of 0.9088, surpassing the performance of both the K-medoid and Birch algorithms. This outcome suggests that K-means successfully separated the data points into distinct and well- separated clusters , demonstrating clear and distinguishable boundaries between the clusters.
- Time complexity of **Ant colony** is greater than all the other algorithm but it always give the optimized result for large instances where as For small TSP instances where computational resources are limited, , the **Nearest Neighbor Algorithm** or the **Insertion TSP Algorithm** might be a good choice for quick approximations but it may not give the optimal result and **2-Opt Algorithm** is suitable for post-processing and improving solutions obtained from other algorithms, potentially enhancing the quality of the tour but it Can stuck in local optima and may require multiple iterations and also does not guarantee global optimality and most important **Andean Condor Algorithm** is designed to cater to the specific needs of delivery personnel and acknowledges variations in delivery requirements within clusters.

CONCLUSION

Reducing delivery time and optimizing routes for dedicated delivery personnel through clustering and optimization algorithms offers substantial benefits, from cost savings to improved customer satisfaction. When implemented effectively, these strategies can transform delivery services into more efficient and competitive operations.

THANK YOU