Date: 20.08.25

TASK:5

Implementation of Ant Colony Optimization to find the shortest and most efficient route for ride-sharing/delivery services using Python

Implementation of Ant Colony Optimization to Optimize Ride-Sharing Trip Duration using Python by following constraints.

•To forecast travel times between every pair of pick-up and drop-off locations.

•To find the shortest route that visits a set of locations.

•To implement optimization techniques are required to intelligently search the solution space and find near-optimal solutions.

Tools: Python

PROBLEM STATEMENT: **CO2**
S3

A logistics company is trying to improve the efficiency of its delivery trucks by minimizing the total travel distance required to visit multiple customer locations. Each truck must start from the depot, deliver to every customer exactly once, and return to the depot. The road network is represented as a distance matrix between all customer locations. Since finding the shortest possible route is a complex combinatorial optimization problem, the company decides to use the Ant Colony Optimization (ACO) algorithm. The algorithm simulates the behavior of ants searching for food, where ants explore different routes and deposit pheromones on shorter paths. Over time, the pheromone trails guide future ants to choose better routes, leading to the discovery of a near-optimal delivery path that reduces fuel consumption, delivery time, and overall operational cost.

ANT COLONY OPTIMIZATION

AIM

To implementation of Ant Colony Optimization to find the shortest and most efficient route for ride-sharing or delivery services using Python

ALGORITHM

1. Initialize pheromone levels on all routes.
2. Place ants at the starting city
3. For each ant:

   Build a path by moving to the next city based on:

   Pheromone strength (experience).

   Visibility (shorter distance is preferred).
4. Calculate the total distance (cost) of each ant's path.
5. Update pheromones:

   Evaporate some pheromone on all paths.

   Add pheromone to the paths used by ants (better paths get more).

6.Repeat steps 3–5 for a number of iterations.

7. Return the best path and its cost found during the process.

PROGRAM

Delivery optimization using Ant Colony Optimization

```
import numpy as np
from numpy import inf

# Distance matrix between depot + customer locations
# 5 cities (1 depot + 4 customers)
d = np.array([[0, 10, 12, 11, 14],
              [10, 0, 13, 15, 8],
              [12, 13, 0, 9, 14],
              [11, 15, 9, 0, 16],
```

```
            [14, 8, 14, 16, 0]])

iteration = 100    # number of iterations
n_ants = 5         # number of ants
n_citys = 5        # number of cities

# Parameters
e = 0.5   # evaporation rate
alpha = 1 # pheromone influence
beta = 2  # visibility influence

# Visibility = 1/distance
visibility = 1 / d
visibility[visibility == inf] = 0

# Initial pheromone levels
pheromone = 0.1 * np.ones((n_citys, n_citys))

# Routes for ants
routes = np.ones((n_ants, n_citys + 1))

for ite in range(iteration):
    routes[:, 0] = 1  # all ants start at city 1

    for i in range(n_ants):
        temp_visibility = np.array(visibility)

        for j in range(n_citys - 1):
            cur_loc = int(routes[i, j] - 1)
            temp_visibility[:, cur_loc] = 0

            # pheromone and visibility contributions
            pheromone_feat = np.power(pheromone[cur_loc, :], alpha)
            vis_feat = np.power(temp_visibility[cur_loc, :], beta)

            prob = pheromone_feat * vis_feat
```

```
            prob = prob / prob.sum()  # normalize

            cum_prob = np.cumsum(prob)
            r = np.random.random()
            city = np.nonzero(cum_prob > r)[0][0] + 1
            routes[i, j + 1] = city

        # last city not visited
        left = list(set([i for i in range(1, n_citys + 1)]) - set(routes[i, :-2]))[0]
        routes[i, -2] = left

    route_opt = np.array(routes)
    dist_cost = np.zeros((n_ants, 1))

    for i in range(n_ants):
        s = 0
        for j in range(n_citys - 1):
            s += d[int(route_opt[i, j]) - 1, int(route_opt[i, j + 1]) - 1]
        dist_cost[i] = s

    dist_min_loc = np.argmin(dist_cost)
    dist_min_cost = dist_cost[dist_min_loc]

    best_route = routes[dist_min_loc, :]
    pheromone = (1 - e) * pheromone

    for i in range(n_ants):
        for j in range(n_citys - 1):
            dt = 1 / dist_cost[i]
            pheromone[int(route_opt[i, j]) - 1, int(route_opt[i, j + 1]) - 1] += dt

print("Routes of all ants at the end:")
print(route_opt)
print("\nBest Path:", best_route)
print("Cost of the Best Path:", int(dist_min_cost[0]) + d[int(best_route[-2]) - 1, 0])
```

OUTPUT

```
========== RESTART: C:/Users/student/AppData/Local/Programs/Python/Python313/VTU26808.py ==========
=====
Routes of all ants at the end:
[[1. 2. 5. 3. 4. 1.]
 [1. 2. 5. 3. 4. 1.]
 [1. 2. 5. 3. 4. 1.]
 [1. 2. 5. 3. 4. 1.]
 [1. 2. 5. 3. 4. 1.]]

Best Path: [1. 2. 5. 3. 4. 1.]
Cost of the Best Path: 52
```

RESULT

Thus, the implementation of Ant Colony Optimization to find the shortest and most efficient route for ride-sharing or delivery services using Python was successfully executed and output was verified