DATE:29.08.25

TASK 5
Ant Colony Optimization

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 school needs to plan the optimal route for its bus that starts from the school, picks up students from four different stops, and then returns. The distances between the stops and the school are given as a matrix. The school uses the Ant Colony Optimization algorithm to find the shortest route so that the bus consumes less fuel and students spend minimal time commuting. By continuously updating pheromone trails, the algorithm helps the bus driver identify the most efficient path. This not only saves travel costs but also ensures that students arrive at school on time.

DATE:29.08.25

TASK:5

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

AIM

To implementation of Ant Colony Optimization to find the shortest and most efficient route for school route planning using Python

ALGORITHM

- Start with the distance matrix of the school and all student stops.
- Initialize parameters: number of ants, iterations, pheromone influence (α), visibility influence (β), and evaporation rate.
- Initialize pheromone trails on all paths equally.
- For each iteration, place all ants at the school (starting point).
- For each ant, select the next stop using a probability rule based on pheromone level and visibility (1/distance).
- Repeat the selection until the ant has visited all stops exactly once.
- Make the ant return to the school to complete the tour.
- Calculate the total distance (cost) of the route for each ant.
- Update pheromone trails:
    - Evaporate old pheromone.
    - Add new pheromone inversely proportional to route length (shorter routes get more).
- After all iterations, choose the route with the minimum distance as the optimal school bus route.

PROGRAM

School Bus Route Planning

```
import numpy as np
from numpy import inf

# Distance matrix (School + 4 Stops)
# 1 = School, 2–5 = Stops
d = np.array([[0, 10, 12, 11, 14],   # distances from School
              [10, 0, 13, 15, 8],    # Stop 1
              [12, 13, 0, 9, 14],    # Stop 2
              [11, 15, 9, 0, 16],    # Stop 3
              [14, 8, 14, 16, 0]])   # Stop 4

iteration = 100    # number of iterations
n_ants = 5         # number of ants
n_citys = 5        # number of cities (School + Stops)

# 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 the School (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 unvisited city
        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

# Display final result
print("=== School Bus Route Planning with ACO ===")
print("Best Route (School -> Stops -> School):", best_route.astype(int))
```

print("Total Distance (km):", int(dist_min_cost[0]) + d[int(best_route[-2]) - 1, 0])

OUTPUT

```
=============== RESTART: C:/Users/student/Desktop/26005/task5.py ===============
=== School Bus Route Planning with ACO ===
Best Route (School -> Stops -> School): [1 2 5 3 4 1]
Total Distance (km): 52
```

RESULT

Thus, the implementation of Ant Colony Optimization to find the shortest and most efficient route for school route planning using Python was successfully executed and output was verified