Implement Ant colony optimization by solving the Traveling salesman problem using python Problem statement- salesman needs to visit a set of cities exactly once and return to the original city. The task is to find the shortest possible route that the salesman can take to visit all the cities and return to the starting city.

```
import numpy as np
import random


# Define the distance matrix (distances between cities)
# Replace this with your distance matrix or generate one based on your problem
# Example distance matrix (replace this with your actual data)
distance_matrix = np.array([
    [0, 10, 15, 20],
    [10, 0, 35, 25],
    [15, 35, 0, 30],
    [20, 25, 30, 0]
])


# Parameters for Ant Colony Optimization
num_ants = 10
num_i
evapo
phero
heuri
```

Run cell (Ctrl+Enter)
cell executed since last change
executed by a paliwan
20:34 (0 minutes ago)
executed in 0.04 s

```
# Initialize pheromone matrix and visibility matrix
num_cities = len(distance_matrix)
pheromone = np.ones((num_cities, num_cities))  # Pheromone matrix
visibility = 1 / distance_matrix  # Visibility matrix (inverse of distance)
```

```
<ipython-input-4-c226b380e0f8>:4: RuntimeWarning: divide by zero encountered in divide
  visibility = 1 / distance_matrix  # Visibility matrix (inverse of distance)
```

```
# ACO algorithm
for iteration in range(num_iterations):
    ant_routes = []
    for ant in range(num_ants):
        current_city = random.randint(0, num_cities - 1)
        visited_cities = [current_city]
        route = [current_city]

        while len(visited_cities) < num_cities:
            probabilities = []
            for city in range(num_cities):
                if city not in visited_cities:
                    pheromone_value = pheromone[current_city][city]
                    visibility_value = visibility[current_city][city]
                    probability = (pheromone_value ** pheromone_constant) * (visibility_value ** heuristic_constant)
                    probabilities.append((city, probability))

            probabilities = sorted(probabilities, key=lambda x: x[1], reverse=True)
            selected_city = probabilities[0][0]
            route.append(selected_city)
            visited_cities.append(selected_city)
            current_city = selected_city

        ant_routes.append(route)

        # Update pheromone levels
    delta_pheromone = np.zeros((num_cities, num_cities))
    for ant, route in enumerate(ant_routes):
        for i in range(len(route) - 1):
            city_a = route[i]
            city_b = route[i + 1]
            delta_pheromone[city_a][city_b] += 1 / distance_matrix[city_a][city_b]
            delta_pheromone[city_b][city_a] += 1 / distance_matrix[city_a][city_b]

    pheromone = (1 - evaporation_rate) * pheromone + delta_pheromone


# Find the best route
best_route_index = np.argmax([sum(distance_matrix[cities[i]][cities[(i + 1) % num_cities]] for i in range(num_cities)) for cities in ant_routes])
best_route = ant_routes[best_route_index]
shortest_distance = sum(distance_matrix[best_route[i]][best_route[(i + 1) % num_cities]] for i in range(num_cities))


print("Best route:", best_route)
print("Shortest distance:", shortest_distance)
```

```
Best route: [0, 1, 3, 2]
Shortest distance: 80
```

Start coding or generate with AI.

Run cell (Ctrl+Enter)
cell executed since last change

executed by a paliwan
20:34 (0 minutes ago)
executed in 0.04 s