Lab 3 – Ant colony optimization for Travelling salesman problem

```
import numpy as np
import random

def distance(city1, city2):
    """Calculate the Euclidean distance between two cities."""
    return np.sqrt((city1[0] - city2[0])**2 + (city1[1] - city2[1])**2)

def initialize_pheromones(num_cities, initial_pheromone):
    """Initialize the pheromone matrix with the given initial value."""
    return np.full((num_cities, num_cities), initial_pheromone)

def calculate_probabilities(current_city, unvisited, pheromones, distances, alpha, beta):
    """Calculate the probabilities of moving to the next city based on pheromones and distances."""
    probabilities = []
    for city in unvisited:
        tau = pheromones[current_city][city] ** alpha
        eta = (1 / distances[current_city][city]) ** beta
        probabilities.append(tau * eta)
    probabilities = np.array(probabilities)
    return probabilities / probabilities.sum()

def construct_solution(num_cities, pheromones, distances, alpha, beta):
    """Construct a solution for one ant by visiting cities probabilistically."""
    unvisited = list(range(num_cities))
    current_city = random.choice(unvisited)
    unvisited.remove(current_city)
    tour = [current_city]

    while unvisited:
```

```
        probabilities = calculate_probabilities(current_city, unvisited, pheromones, distances, alpha, beta)
        next_city = random.choices(unvisited, weights=probabilities)[0]
        tour.append(next_city)
        unvisited.remove(next_city)
        current_city = next_city

    return tour

def update_pheromones(pheromones, all_tours, distances, rho, Q):
    """Update pheromones based on the quality of the solutions."""
    pheromones *= (1 - rho)  # Evaporation
    for tour, tour_length in all_tours:
        pheromone_increase = Q / tour_length
        for i in range(len(tour)):
            from_city = tour[i]
            to_city = tour[(i + 1) % len(tour)]  # Circular tour
            pheromones[from_city][to_city] += pheromone_increase
            pheromones[to_city][from_city] += pheromone_increase

def calculate_tour_length(tour, distances):
    """Calculate the total length of a given tour."""
    return sum(distances[tour[i]][tour[(i + 1) % len(tour)]] for i in range(len(tour)))

def ant_colony_optimization(cities, num_ants, alpha, beta, rho, Q, iterations, initial_pheromone):
    """Solve the Traveling Salesman Problem using Ant Colony Optimization."""
    num_cities = len(cities)
    distances = np.array([[distance(cities[i], cities[j]) for j in range(num_cities)] for i in range(num_cities)])
    pheromones = initialize_pheromones(num_cities, initial_pheromone)

    best_tour = None
```

```
    best_length = float('inf')

    for _ in range(iterations):
        all_tours = []

        for _ in range(num_ants):
            tour = construct_solution(num_cities, pheromones, distances, alpha, beta)
            tour_length = calculate_tour_length(tour, distances)
            all_tours.append((tour, tour_length))

            if tour_length < best_length:
                best_tour = tour
                best_length = tour_length

        update_pheromones(pheromones, all_tours, distances, rho, Q)

    return best_tour, best_length

# Example usage
if __name__ == "__main__":
    cities = [(0, 0), (2, 0), (2, 2), (0, 2), (1, 1)]  # Define city coordinates
    num_ants = 10
    alpha = 1.0
    beta = 2.0
    rho = 0.5
    Q = 100
    iterations = 100
    initial_pheromone = 1.0

    best_tour, best_length = ant_colony_optimization(cities, num_ants, alpha, beta, rho, Q, iterations,
initial_pheromone)
```

```
print("Best tour:", best_tour)
print("Best length:", best_length)
```

OUTPUT:

```
Best tour: [1, 0, 4, 3, 2]
Best length: 8.82842712474619
```