EXP #3 : Ant Colony Optimization

Code:

```
import numpy as np

# City locations (x, y)
cities = [[4.897047, 45.465075],
          [9.105735, 45.027249],
          [15.396817, 43.972633],
          [21.010096, 43.328035],
          [25.505014, 44.361121],
          [16.924506, 43.190884],
          [22.560213, 44.385755]]

# Function to calculate Euclidean distance between two points
def euclidean_distance(point1, point2):
    # Convert lists to NumPy arrays for element-wise subtraction
    point1 = np.array(point1)
    point2 = np.array(point2)
    return np.sqrt(np.sum((point1 - point2) ** 2))

# Create pheromone matrix
pheromones = np.ones((len(cities), len(cities))) / len(cities)

# Number of ants
num_ants = 5

# Pheromone evaporation rate
evaporation_rate = 0.5

# Number of iterations
num_iterations = 100

# Function to choose next city
def choose_next_city(pheromones, cities, current_city, visited_cities):
    unvisited_cities = list(set(range(len(cities))) - set(visited_cities))

    probabilities = pheromones[current_city][unvisited_cities] / \
                (np.array([euclidean_distance(cities[current_city], cities[city]) for city in
unvisited_cities]) ** 2)
    probabilities /= sum(probabilities)

    next_city = np.random.choice(unvisited_cities, p=probabilities)
```

```
    return next_city

# Function to simulate a single ant
def simulate_ant(pheromones, cities, num_iterations):
    current_city = np.random.choice(len(cities))
    visited_cities = [current_city]

    for _ in range(num_iterations -1): # Reduce the number of iterations by 1
        next_city = choose_next_city(pheromones, cities, current_city, visited_cities)
        visited_cities.append(next_city)
        current_city = next_city

    return visited_cities

# Function to run ACO
def aco_tsp(pheromones, cities, num_ants, num_iterations):
    solutions = []

    for _ in range(num_iterations):
        for _ in range(num_ants):
            solutions.append(simulate_ant(pheromones, cities, len(cities)))

        pheromones *= evaporation_rate

        for solution in solutions:
            for i in range(len(solution) - 1):
                pheromones[solution[i]][solution[i + 1]] += 1 / euclidean_distance(cities[solution[i]],
cities[solution[i + 1]])

    return solutions

# Run ACO
solutions = aco_tsp(pheromones, cities, num_ants, num_iterations)

# Get the best solution
best_solution_indices = min(solutions, key=lambda x: sum(euclidean_distance(cities[x[i]],
cities[x[i + 1]]) for i in range(len(x) - 1)))
best_solution = [cities[i] for i in best_solution_indices]

# Print the best solution
print("Best TSP tour (cities visited):", best_solution)
```

Output:

```
Best TSP tour (cities visited):
 [[25.505014, 44.361121], [22.560213, 44.385755], [21.010096, 43.328035], [16.924506, 43.190884], [15.396817, 43.972633], [9.105735, 45.027249], [4.897047, 45.465075]]
```