ANT COLONY OPTIMIZATION

```
import numpy as np
import random

def euclidean_distance(city1, city2):
    return np.linalg.norm(np.array(city1) - np.array(city2))

def solve_tsp_aco(cities, num_ants, num_iterations, alpha, beta,
evaporation_rate, pheromone_deposit_amount):

    num_cities = len(cities)
    distance_matrix = np.zeros((num_cities, num_cities))
    for i in range(num_cities):
        for j in range(num_cities):
            if i != j:
                distance_matrix[i, j] = euclidean_distance(cities[i],
cities[j])

    pheromone_matrix = np.ones((num_cities, num_cities))
    best_tour = None
    best_tour_length = float('inf')

    for iteration in range(num_iterations):
        all_tours = []
        all_tour_lengths = []

        for ant in range(num_ants):
            current_city = random.randint(0, num_cities - 1)
            tour = [current_city]
            visited = set([current_city])
            tour_length = 0

            while len(tour) < num_cities:
                next_city = None
                probabilities = []
                total_probability = 0

                for city_j in range(num_cities):
                    if city_j not in visited:
                        pheromone = pheromone_matrix[current_city,
city_j] ** alpha
                        heuristic = (1.0 /
distance_matrix[current_city, city_j]) ** beta
                        probability = pheromone * heuristic
                        probabilities.append((city_j, probability))
                        total_probability += probability
```

```
                if total_probability > 0:
                    probabilities = [(city, prob / total_probability)
for city, prob in probabilities]
                    next_city = random.choices([city for city, prob in
probabilities], [prob for city, prob in probabilities])[0]

                if next_city is not None:
                    tour.append(next_city)
                    visited.add(next_city)
                    tour_length += distance_matrix[current_city,
next_city]
                    current_city = next_city
                else:
                    break


            if len(tour) == num_cities:
                tour_length += distance_matrix[current_city, tour[0]]
                all_tours.append(tour)
                all_tour_lengths.append(tour_length)

        pheromone_matrix *= (1 - evaporation_rate)
        for i in range(len(all_tours)):
            tour = all_tours[i]
            tour_length = all_tour_lengths[i]
            if tour_length < float('inf'):
                for j in range(num_cities):
                    city1 = tour[j]
                    city2 = tour[(j + 1) % num_cities]
                    pheromone_matrix[city1, city2] +=
pheromone_deposit_amount / tour_length
                    pheromone_matrix[city2, city1] +=
pheromone_deposit_amount / tour_length

        if all_tour_lengths:
            min_tour_length = min(all_tour_lengths)
            if min_tour_length < best_tour_length:
                best_tour_length = min_tour_length
                best_tour = all_tours[np.argmin(all_tour_lengths)]

    return best_tour, best_tour_length


cities = [(0, 0), (1, 5), (6, 2), (8, 8), (3, 4)]

num_ants = 10
num_iterations = 100
alpha = 1.0
beta = 5.0
```

```
evaporation_rate = 0.5
pheromone_deposit_amount = 100.0

best_tour, best_tour_length = solve_tsp_aco(cities, num_ants,
num_iterations, alpha, beta, evaporation_rate,
pheromone_deposit_amount)

print("Best tour found:", best_tour)
print("Best tour length:", best_tour_length)
```

OUTPUT:

```
Best tour found: [4, 1, 0, 2, 3]
Best tour length: 26.38732236919894
```