TASK.5

import numpy as np

# Define the problem: Distance matrix between cities

distance\_matrix = np.array([

[0, 2, 2, 5],

[2, 0, 3, 4],

[2, 3, 0, 1],

[5, 4, 1, 0]

])

# Parameters

num\_ants = 4

num\_iterations = 100

alpha = 1 # Pheromone importance

beta = 2 # Distance importance

evaporation\_rate = 0.5

pheromone\_constant = 1

# Initialize pheromone levels

num\_cities = len(distance\_matrix)

pheromone = np.ones((num\_cities, num\_cities))

# Helper function: Calculate probabilities for next city

def calculate\_probabilities(current\_city, visited, pheromone, distance\_matrix):

probabilities = []

for city in range(num\_cities):

if city not in visited:

prob = (pheromone[current\_city][city] \*\* alpha) \* ((1 / distance\_matrix[current\_city][city]) \*\* beta)

probabilities.append(prob)

else:

probabilities.append(0)

probabilities = np.array(probabilities)

return probabilities / probabilities.sum()

# Main ACO loop

best\_path = None

best\_distance = float('inf')

for iteration in range(num\_iterations):

all\_paths = []

all\_distances = []

for ant in range(num\_ants):

visited = []

current\_city = np.random.randint(0, num\_cities)

visited.append(current\_city)

while len(visited) < num\_cities:

probabilities = calculate\_probabilities(current\_city, visited, pheromone, distance\_matrix)

next\_city = np.random.choice(range(num\_cities), p=probabilities)

visited.append(next\_city)

current\_city = next\_city

# Complete the tour by returning to the starting city

visited.append(visited[0])

all\_paths.append(visited)

# Calculate the total distance of the path

distance = sum(distance\_matrix[visited[i]][visited[i + 1]] for i in range(len(visited) - 1))

all\_distances.append(distance)

# Update best path

if distance < best\_distance:

best\_distance = distance

best\_path = visited

# Update pheromone levels

pheromone \*= (1 - evaporation\_rate) # Evaporation

for path, distance in zip(all\_paths, all\_distances):

for i in range(len(path) - 1):

pheromone[path[i]][path[i + 1]] += pheromone\_constant / distance

# Output the best path and distance

print("Best Path:", best\_path)

print("Best Distance:", best\_distance)

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.