Lab - 3

Ant colony optimization

Algorithm

Initialize pheromone values $v_{i,j} = C_1, v_{i_0}, v_j / z_0$

prepare

for each ant $2 \in \{1, \ldots, n\}$

randomly choose starting city $i_0 \in S$ for and ant 2

move to starting city $i_0 \in S$ for any $2\ i \rightarrow i_0$ move

while $S \neq 0$ do

remove current city from selection set $S \rightarrow S \setminus \{t\}$

Choose next city i in tour with probability

$$p_{ij} = \frac{z_{i,j}^{\alpha} \cdot \eta_{i,j}^{\beta}}{\sum_{h \in S} z_{i,\eta}^{\alpha} \cdot \eta_{i,\eta}^{\beta}}$$

update solution allocation $\Pi_i(i) \rightarrow j$

move to new city $i \rightarrow j$

end while

finalize solution vector $\Pi_L(i) \rightarrow i_0$

end for

for each solution Π_L, $L \in \{1, \ldots, n\}$ do

calculate tour length $f(\Pi_L) \rightarrow \sum_{i=1}^{n} \alpha_{i\Pi_L(i)}$

end for

for all (i,j) do

evaporate pheromone $z_{ij} \rightarrow (1-\rho) \cdot z_{ij}$

end for

determine best solution of iteration $\pi^T = \arg\min_{q \in [1,m]} H(\pi_q)$

if Π^t better than current best π^*, i.e $f(\pi^t) < f(\pi^*)$

then set $\pi^* \mapsto \pi^t$

end if

for all $(i,j) \in \pi^t$ do

reinforce $>ij \mapsto z_{ij} + \Delta/2$

end for

for all $(i,j) \in \Pi^*$ do
reinforce $z_{ij} \rightarrow z_{ij} + \Delta/2$
end for
until condition for termination met

Program

```
import numpy as np
def initialize_pheromones(n, tau_0):
    return np.full((n, n), tau_0)
def calculate_distance(city1, city2):
    return np.linalg.norm(city1 - city2)
def calculate_probabilities(pheromones, distances, alpha, beta,
visited, current_city)
    n = len(pheromones)
    probabilities = np.zeros(n)
    for j in range(n):
        if j not in visited
            probabilities[j] = (pheromones[current_city, j]**
            alpha) * ((1 / distances[current_city, j])**
                beta)
    return probabilities / probabilities.sum()

def aco_tsp(cities, n, alpha, beta, rho, Q, iterations):
    n = len(cities)
    tau_0 = 1 / (n * np.mean([calculate_distance(cities[i],
        cities[j]) for i in range(n) for j in
        range(n)]))
    pheromones = initialize_pheromones(n, tau_0)
    best_tour = None
    best_tour_length = float('inf')
```

```
for _ in range (n):
    tour = [ ]
    visited = set ()
    current_city = np.random.randint (0, n)
    visited.add (current_city)
    tour.append (current_city)
    while len(visited) < n :
        probabilities = calculate_probabilities (
            pheromones, distances, alpha, beta, visited,
            current_city)
    next_city = np.random.choice (range (n), p=probabilities)
    visited.add (next_city)
    tour.append (next_city)
    current_city = next_city)
    tour.append (tour[0])
    tour_length = sum (distances [tour [i], tour [i+1]] for
        i in range (len (tour) - 1))
    all_tours.append (tour)
    all_tour_lengths.append (tour_length)
    delta_tau = Q / tour_length
    for i in range ( len (tour) - 1) :
        pheromones [tour [i], tour [i+1] += delta tau
        pheromones [tour [i+1], tour [i]] += delta tau
    pheromones *= (1-rho)
    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 [a] tour ~
                                length))

return best_tour, best_tour_length :
```

```
if __name__ == "__main__":
    cities = np.array([[0,0],[1,3],[4,3],[6,1]])
    distances = np.array([[calculate_distance(c1,c2)
    for c2 in cities] for c1 in cities])
    in=10
    alpha=1
    beta=2
    rho=0.5
    Q=100
    iterations=100
    best_tour, best_tour_length = aco_tsp(cities,
    alpha, beta, rho, Q, iterations)
    print("Best Tour:", best_tour)
    print("Best Tour Length:", best_tour_length)
```

Output:

```
Best Tour: [0, np.int64(1), np.int64(2), np.int64(3)
Best Tour Length: 15.073
```