Date
Page

3. Algorithm

Initial pheromone values $\forall i, j \in [1, n]$: $z_j = z_0$

repeat.

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

randomly choose starting city $i_0 \in S$ for ant k $i \rightarrow i_0$

move

while $S \neq 0$ do

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

Choose next city i in tour with probability $p_{ij} = \frac{z^{\alpha}_{ij} \cdot n^{\beta}_{ij}}{\sum_{n \in S} z^{\alpha}_{in} n^{\beta}_{in}}$

update solution auction $\pi_k(i) \rightarrow j$

move to new city $i \rightarrow i$

end while

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

end for

for each solution π_k, $k \in \{1, \ldots, n\}$ do

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

end for

for all (i, j) do

evaporate pheromone $z_{ij} = s(1 - \rho) z_{ij}$

end for

determine best solution of iteration

$\pi^t = \text{argmin} f(\pi_k)$ $k \in [1, m]$

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

then set $\pi^* \leftarrow \pi^+$
end if
for all $(i,j) \in \pi^+$ do
reinforce $z_{ij} \leftarrow z_{ij} + \Delta/2$
end for
for all $(i,j) \in \pi^*$ do
reinforce $z_{ij} \rightarrow z_{ij} + \Delta/2$
end for
until condition for termination.

program:

```
import numpy as np.

def initialize_pheromones(n, tau0):
    return np.full((n,n), tau0)

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, m, alpha, beta, rho, Q, iterations):
    n = len(cities)
    tau = 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(m):
        tour = []
        visited = set()
        current_city = np.random.randint(0,n)
        visited.add(current_city)
        tour.append(current_city)

        while len(visited) < :
            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(next_city)
            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_tour = Q/tour_length
            for i in range(len(tour)-1):
                pheromones[tour[i], tour[i+1]] +=
delta_tour
                pheromones[tour[i+1], tour[i]] +=
delta_tour

            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(
all_tour_lengths)]
            return best_tour, best_tour_lengths

if __name__ == "__main__":
    cities = np.array([[0,0],[1,3],[4,3],
[6,1]])
```

```
distance = np.array([calculate_distance
(c1, c2)
 for c2 in cities] for c1 in cities])
m = 10
alpha = 1
beta = 2
rho = 0.5
Q = 100
iterations = 100

best_tour, best_tour_length = aco_tsp
(cities, m, 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), 8]

Best tour length: 15.075.

Sol:1