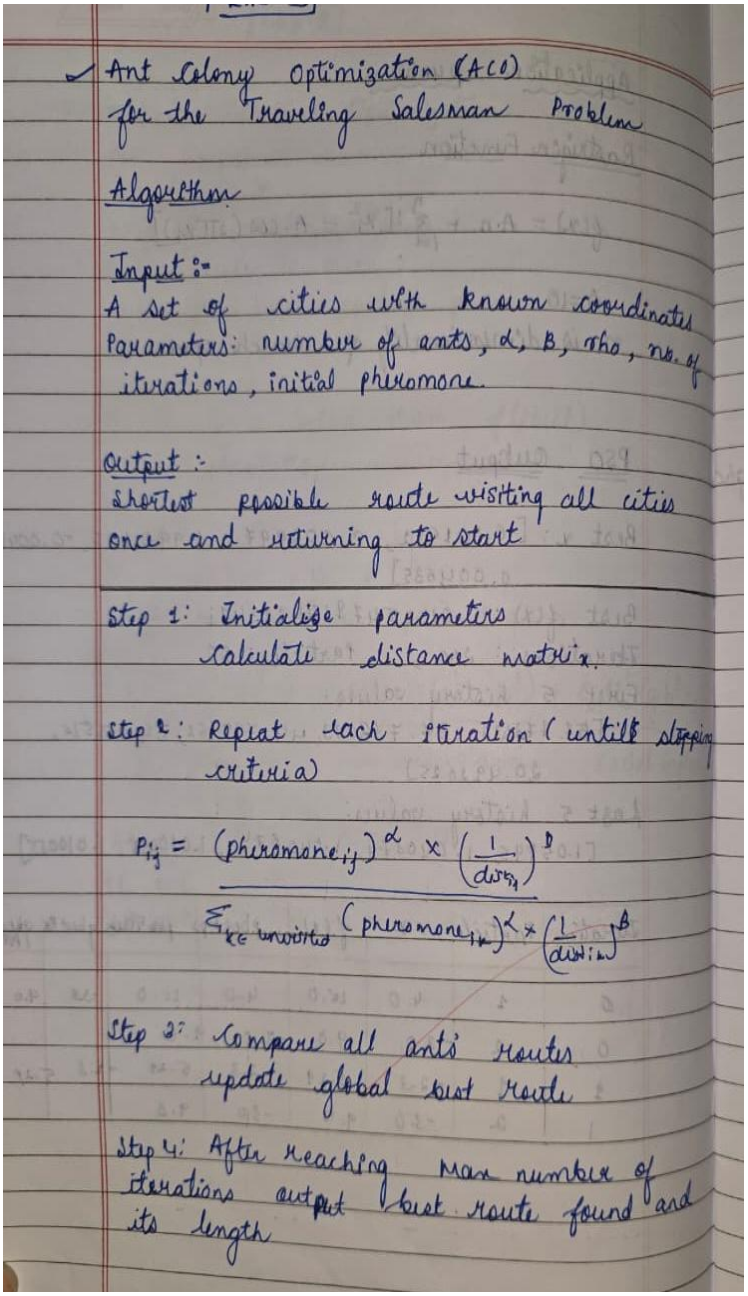
Program 3

Ant Colony Optimization for the Traveling Salesman Problem Ant Colony Optimization (ACO) for the Vehicle Routing Problem (VRP): It involves finding optimal routes for multiple vehicles to deliver goods to a set of customers from a central depot.

Algorithm:



Ant Colony Optimization (ACO)
for the Traveling Salesman Problem

Algorithm

Input :-
A set of cities with known coordinates
Parameters: number of ants, α, β, rho, no. of iterations, initial pheromone

Output :
shortest possible route visiting all cities once and returning to start

Step 1: Initialise parameters
Calculate distance matrix.

Step 2: Repeat each iteration (until stopping criteria)

$$P_{ij} = \frac{(\text{pheromone}_{ij})^{\alpha} \times \left(\frac{1}{\text{dist}_{ij}}\right)^{\beta}}{\sum_{k \in \text{unvisited}} (\text{pheromone}_{ik})^{\alpha} \times \left(\frac{1}{\text{dist}_{ik}}\right)^{\beta}}$$

Step 3: Compare all ant's routes
update global best route

Step 4: After reaching max number of iterations output best route found and its length

Example
Initialise

city	Coordinates (x,y)
0	(1, 1)
1	(4, 1)
2	(4, 5)
3	(1, 5)

Distance Matrix

From / To	0	1	2	3
0	0	3.0	5.0	4.0
1	3.0	0	4.0	5.0
2	5.0	4.0	0	3.0
3	4.0	5.0	3.0	0

Initial pheromone matrix

From/To	0	1	2	3
0	0.1	0.1	0.1	0.1
1	0.1	0.1	0.1	0.1
2	0.1	0.1	0.1	0.1
3	0.1	0.1	0.1	0.1

Sol

Iteration	Best Route	Length	Example Pheromone on edge (0,1)	Example Pheromone on edge (0-3)
1	0 → 1 → 2 → 3 → 0	14	0.12/4	0.05
2	0 → 3 → 2 → 1 → 0	14	0.0607	0.1321
3	1 → 0 → 3 → 2 → 1	14	~0.09	~0.1

10/10/25.

Step 5:

$x = 0.5 \rightarrow f(x) = 0.25$

$x = 0 \quad f(x) = 0$

Global minimum found!

Eg

① $f(x) = x_1^2 + x_2^2$

Number of nests = 25

discovery rate = 0.25

100 iterations

25 nests in range $[-5, 5]$

Problem: Welded - Beam Design

minimize fabrication cost (material + welding)

subject to constraints (stress, buckling, duct geometry).

$x_1 = h$ (weld thickness) (in)

$x_2 = l$ (weld length) (inches)

$x_3 = t$ = beam height

$x_4 = b$ = beam width

Minimize $f(x) = 1.10471\, x_1^2 x_2 + 0.04811\, x_3 x_4 (14.0 + x_2)$

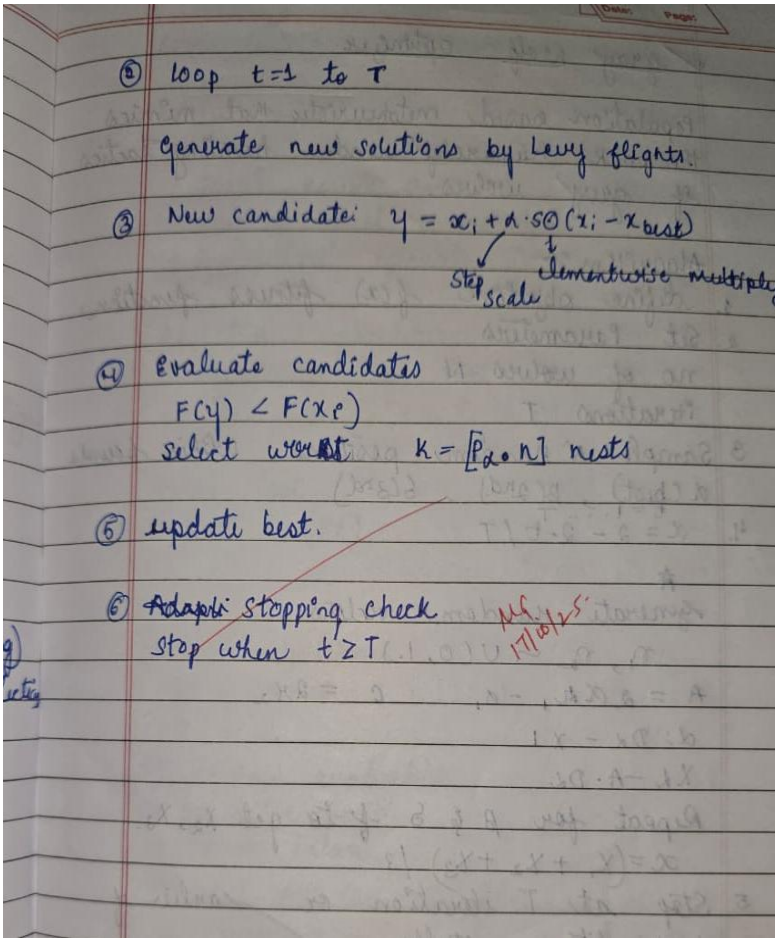
① Initialize.

Set parameters n, P_a, T, λ, M.

penalized fitness

$F(x) = f(x) + M \cdot \sum_i \max(0, g_i(x))^2$

record x_{best}.



Code:

```
import numpy as np


cities = np.array([
    [1, 1],
    [4, 1],
    [4, 5],
    [1, 5]
])


def distance_matrix(coords):
    n = len(coords)
    d = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            d[i][j] = np.linalg.norm(coords[i] - coords[j])
    return d
```

```
dist = distance_matrix(cities)

alpha = 1
beta = 2
rho = 0.5
num_ants = 5
iterations = 3


pher = np.ones_like(dist) * 0.1

def probability(i, visited):
    probs = []
    for j in range(len(dist)):
        if j not in visited:
            tau = pher[i][j] ** alpha
            eta = (1 / dist[i][j]) ** beta
            probs.append((j, tau * eta))
    total = sum(p for _, p in probs)
    return [(node, p / total) for node, p in probs]

def choose_next(probs):
    r = np.random.random()
    cum = 0
    for node, p in probs:
        cum += p
        if r <= cum:
            return node
    return probs[-1][0]

def route_length(route):
    length = 0
    for i in range(len(route)):
        length += dist[route[i]][route[(i + 1) % len(route)]]
    return length

best_route = None
best_len = float("inf")

for it in range(iterations):
    all_routes = []

    for k in range(num_ants):
        start = 0
        route = [start]
```

```
        while len(route) < len(dist):
            probs = probability(route[-1], route)
            nxt = choose_next(probs)
            route.append(nxt)

        all_routes.append(route)


    pher = (1 - rho) * pher
    for r in all_routes:
        L = route_length(r)
        if L < best_len:
            best_len = L
            best_route = r
        for i in range(len(r)):
            a = r[i]
            b = r[(i + 1) % len(r)]
            pher[a][b] += 1 / L

    print(f"Iteration {it+1}: Best Route = {best_route}, Length = {best_len}")
```

OUTPUT:

```
Iteration 1: Best Route = [0, 1, 2, 3], Length = 14.0
Iteration 2: Best Route = [0, 1, 2, 3], Length = 14.0
Iteration 3: Best Route = [0, 1, 2, 3], Length = 14.0
```