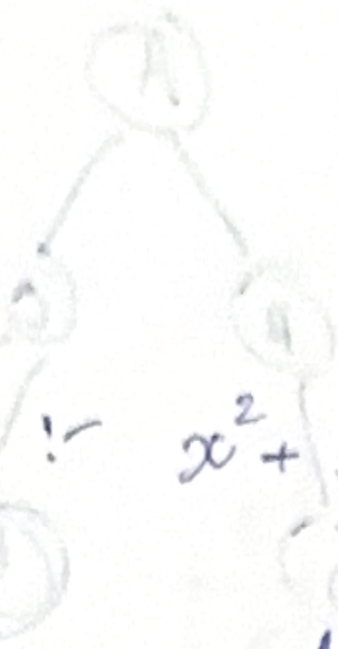
Lab - 5

Stimulated Annealing

Step 1 :-



Objective function :- $x^2 + 5\sin x$

function Simulated Annealing (initial_state, initial temperature, cooling_state. iterations)

current state . = initial_state

best_state = current_state

best_cost = Objective function (current_state)

temp = initial_temperature

while temp > 1 :

for i <= 1 to iterations

new_state = Neighbour (current state)

cur_cost = Objective Function(cur_state)

new_cost = Objective function (new_state)

if AP(cur_cost , new_cost, temp) > Random (0,1)

current_state = new_state

if new_cost < best_cost

best_state = new_state

best_cost = new_cost

temp * = cooling_rate

return (best_state , best_cost)

function Objective function (state):

    cost = 0

    for ele in state

        cost += ele² + 5 sin ele

    Return cost

function Neighbour (state)

    new_state = state - copy()

    index = Random (0, length(state) - 1)

    new_state [index] += Random(-1, 1)

    Return new state

function AP (curr_cost, new_cost, temp)

    if (new_cost < curr_cost);

        Return 1

    else

        return e^((curr_cost - new_cost) / temp)

Prog 22/10/24

~~objective function~~

def main():

    initial_temp = 1000

    cooling_rate = 0.9

    iterations = 1000

    initial_state = [random.uniform(-10, 10) for _ in range(2)]

best_state, best_cost = Simu (initial_state, initial_temp,
                              cooling_rate, iterations)

print(f"Best state: {best state}")

print(f"Best cost: {best_cost}")

Prog 22/10/24

Output :

Best state = [-0.2587, -0.13911, -0.1007, -0.0901,
-0.2483, 0.0074, -0.1024]

Best cost = 0.17660

Lab-6

→ 8 queens using Hill Climbing

① import random

② def - calculate_attacks ()

attack = 0

for i in range (len, state)

for j in range (len+1, state)

if (state(i) == state(j)] or abs state[i-j] = abs sta

attack += 1

return attack

③ def - hill climbing ()

state = random.randint (0,7) for in range

Calculat_attacks = current-attacks

④ for _ in range ()

neighbours = ()

for row in range (8)

for col in range (8)

state(row) != state(col)

neighbours = state

next_state = min(neighbours,

next_cutteak = Calculate_attack.