Lab 5

Stimulated Annealing Algorithm

code:-

import numpy as np

import math

import random

def objective\_function(x):

"""Objective function to minimize: f(x) = x^2"""

return x \*\* 2

def simulated\_annealing(initial\_state, initial\_temp, cooling\_rate, max\_iterations):

"""Simulated Annealing algorithm to find the minimum of the objective function."""

current\_state = initial\_state

current\_energy = objective\_function(current\_state)

best\_state = current\_state

best\_energy = current\_energy

temp = initial\_temp

for iteration in range(max\_iterations):

# Generate a new candidate state by perturbing the current state

candidate\_state = current\_state + random.uniform(-1, 1)

candidate\_energy = objective\_function(candidate\_state)

# Calculate energy difference

energy\_diff = candidate\_energy - current\_energy

# If the candidate state is better, or accepted with a certain probability

if energy\_diff < 0 or random.uniform(0, 1) < math.exp(-energy\_diff / temp):

current\_state = candidate\_state

current\_energy = candidate\_energy

# Update best state found

if current\_energy < best\_energy:

best\_state = current\_state

best\_energy = current\_energy

# Cool down the temperature

temp \*= cooling\_rate

# Print the current state and temperature for debugging

print(f"Iteration {iteration + 1}: Current State = {current\_state:.4f}, Current Energy = {current\_energy:.4f}, Temperature = {temp:.4f}")

return best\_state, best\_energy

# Get user input for parameters

try:

initial\_state = float(input("Enter the initial state (starting point): "))

initial\_temp = float(input("Enter the initial temperature: "))

cooling\_rate = float(input("Enter the cooling rate (between 0 and 1): "))

max\_iterations = int(input("Enter the number of iterations: "))

# Validate cooling rate

if cooling\_rate <= 0 or cooling\_rate >= 1:

raise ValueError("Cooling rate must be between 0 and 1.")

# Execute the simulated annealing algorithm

best\_state, best\_energy = simulated\_annealing(initial\_state, initial\_temp, cooling\_rate, max\_iterations)

# Output the best state and energy found

print(f"Best State: {best\_state:.4f}, Best Energy: {best\_energy:.4f}")

except ValueError as e:

print(f"Invalid input: {e}")  
output:-



Observation:





