|  |  |  |  |
| --- | --- | --- | --- |
|  | Bansilal Ramnath Agarwal Charitable Trust's  Vishwakarma Institute of Information Technology  **Department of**  **Artificial Intelligence and Data Science** | | |
| Name: Siddhesh Dilip Khairnar | | | |
| Class: TY | Division: B | | Roll No: 372028 |
| Semester: V | | Academic Year: 2023-2024 | |
| Subject Name & Code: ADUA31201: Artificial Intelligence | | | |
| Title of Assignment: Write a program to apply simulated annealing algorithm to a simple 1-D x^2 objective function with the bounds [-5,5] | | | |
| Date of Performance:24-10-2023 | | Date of Submission: 11-11-2023 | |

**ASSIGNMENT NO. 5**

**CODE:**

import random

import math

# Define the objective function

def objective\_function(x):

    return x\*\*2

# Simulated Annealing parameters

initial\_temperature = 100.0

cooling\_rate = 0.95

min\_temperature = 0.1

iterations = 5

# Initial solution

x = random.uniform(-5, 5)

current\_energy = objective\_function(x)

best\_x = x

best\_energy = current\_energy

# Simulated Annealing algorithm

for iteration in range(iterations):

    temperature = initial\_temperature \* math.pow(cooling\_rate, iteration)

    # Generate a random neighbor within the bounds

    neighbor\_x = x + random.uniform(-1, 1)

    # Ensure the neighbor is within bounds

    neighbor\_x = max(-5, min(5, neighbor\_x))

    neighbor\_energy = objective\_function(neighbor\_x)

    # Calculate the energy (cost) change

    energy\_change = neighbor\_energy - current\_energy

    # Decide whether to move to the neighbor solution

    if energy\_change < 0 or random.random() < math.exp(-energy\_change / temperature):

        x = neighbor\_x

        current\_energy = neighbor\_energy

    # Update the best solution

    if current\_energy < best\_energy:

        best\_x = x

        best\_energy = current\_energy

    # Print progress

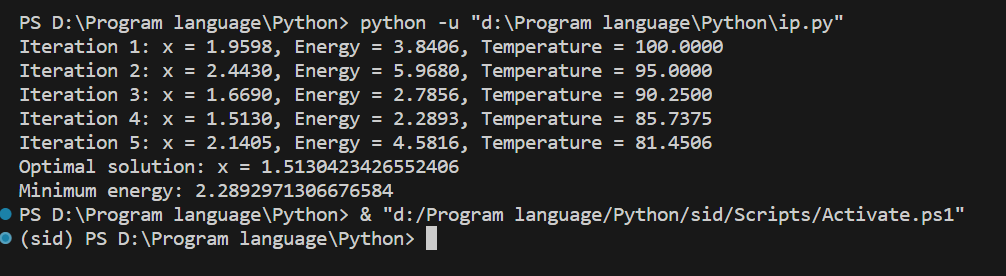
    print(

        f"Iteration {iteration + 1}: x = {x:.4f}, Energy = {current\_energy:.4f}, Temperature = {temperature:.4f}")

print("Optimal solution: x =", best\_x)

print("Minimum energy:", best\_energy)

**OUTPUT:**

****