LAB 5

Simulated Annealing

import random

import math

def objective\_function(x):

    # Example objective function: f(x) = x^2

    return x \*\* 2

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

    current\_solution = initial\_solution

    current\_temp = initial\_temp

    best\_solution = current\_solution

    best\_value = objective\_function(best\_solution)

    # Initialize current\_value to the value of the initial solution

    current\_value = best\_value

    for iteration in range(max\_iterations):

        # Generate a new solution by perturbing the current solution

        new\_solution = current\_solution + random.uniform(-1, 1)  # small random step

        new\_value = objective\_function(new\_solution)

        # Calculate the change in objective value

        delta\_value = new\_value - current\_value

        # Accept the new solution with a certain probability

        if delta\_value < 0 or random.random() < math.exp(-delta\_value / current\_temp):

            current\_solution = new\_solution

            current\_value = new\_value

            # Update best solution found so far

            if new\_value < best\_value:

                best\_solution = new\_solution

                best\_value = new\_value

        # Cool down the temperature

        current\_temp \*= cooling\_rate

    return best\_solution, best\_value

# Example usage

initial\_solution = random.uniform(-10, 10)

initial\_temp = 1000

cooling\_rate = 0.99

max\_iterations = 1000

best\_solution, best\_value = simulated\_annealing(initial\_solution, initial\_temp, cooling\_rate, max\_iterations)

print(f"The best x: {best\_solution}, It's corresponding f(x): {best\_value}")

OUTPUT:

