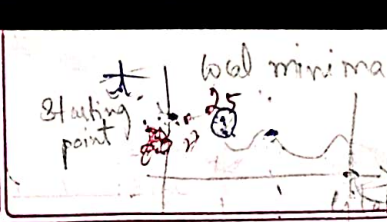

temp decreases in each iteration.

delta value < 0
↳ accepted.

PAGE :
DATE : / /

Lab - 5

Simulated Annealing Algorithm

```
import math
import random
```

Step 1: Initialize the initial solution & an initial temperature

Step 2: Take a random step from initial state.

Step 3: Evaluate new solution

comparing with the previous solution

which is b/w new & current solution

Step 4: Case 1:

If the new solution is better ($\Delta E < 0$) accept it

Case 2:

If the new solution is worse ($\Delta E > 0$) accept it with a probability that decreases with temperature

$$P(\text{accept}) = e^{-\Delta E/T}$$

This allows the algorithm to escape local optima by accepting worse solutions at high temperature.

checks solution

Where E is Energy or objective function value

T is temperature

↳ controls the exploration.

Cooling :-

Step 5: Reduce the temperature according to a cooling schedule

Step 6: Terminate :- Continue until the temperature drops below a threshold.

Proceed

$$f(x) = (x-3)^2$$

x = 10 T = 10

↳ temp decrease by 10%. So it is 0.9

Current solution = 10

Current energy = $f(10) = (10-3)^2 = 49$ - Generate new solution by taking random step $x = 9.5$, $f(9.5) = (9.5-3)^2 = 42.25$

$9.5 < f(10)$ The new solution is better & it is accepted

```
import math
import random
def objective_function(x):
    return (x - 3) ** 2
def simulated_annealing(objective_function, initial_solution,
                        initial_temperature, cooling_rate, stopping
                                    temperature, max_iterations):
    current_solution = initial_solution
    current_value = objective_function(current_solution)
    best_solution = current_solution
    best_value = current_value
    temperature = initial_temperature
    iteration = 0
    while temperature > stopping_temperature and iteration <
                                                max_iteration:
        new_solution = current_solution + random.uniform(-1,1)
        new_value = objective_function(new_solution)
        delta_value = new_value - current_value
        if delta_value < 0:
            current_solution = new_solution
            current_value = new_value
        else:
            probability = math.exp(-delta_value / temperature)
            if random.random() < probability:
                current_solution = new_solution
                current_value = new_value
        if current_value < best_value:
            best_solution = current_solution
            best_value = current_value
        temperature = temperature * cooling_rate
        iteration += 1
        print(f"iteration: {iteration}, Temperature: {temperature:4f}
        current solution: {current_solution:4f}, Best solution: {best_
                                                    solution:4f}")
```

PAGE :
DATE : / /

return best_solution, best_value
initial_solution = 10
initial_temperature = 1000
cooling_rate = 0.95
stopping_temperature = 1e-8
max_iterations = 10
best_solution, best_value = simulated_annealing(objective_function, initial_solution, initial_temperature, cooling_rate, stopping_temperature, max_iterations)
print(f"Best solution: x = {best_solution:.4f}, f(x) = {best_value:.4f}")

o/p :-

iteration : 1, Temperature : 950.000000, current solution : 9.475315, Best Solution : 9.475315
iteration : 2, Temperature : 902.500000, current solution : 9.475315, Best Solution — " —
iteration : 3, Temperature : 857.375000, current solution : 9.509608, Best Solution — " —
iteration : 4, Temperature : 814.506250, current solution : 9.636614, Best Solution — " —
iteration : 5, Temperature : 773.780937, current solution : 10.451081, Best — " —
iteration : 6, Temperature : 735.091891, current solution : 10.182316, — " —
iteration : 7, Temperature : 698.337296, current solution : 10.154454, — " —
iteration : 8, Temperature : 663.420431, current solution : 10.600446, — " —
iteration : 9, Temperature : 630.249410, current solution : 10.879282, — " —
iteration : 10, Temperature : 598.736939, current solution : 11.743971, Best Solution : 9.475315

PAGE :
DATE : / /

$f(x) = (x-3)^2$

$x = 2$ $T = 10$ cooling rate $= 0.9$ [temperature decrease by 10% in each iteration]

Stopping Temperature : $T_{stop} = 0.01$

iteration 1 :- $f(2) = (2-3)^2 = 1$

Generating new sol^{n} by taking random step

new sol^{n} $x = 2.3$

New energy: $f(2.3) = (2.3-3)^2 = 0.49$

$f(2.3) < f(2)$ ∴ The new sol^{n} is better, so we accept it

Best solution is updated to $x = 2.3$ energy 0.49.

cooling : update the temperature :- $T = 10 \times 0.9 = 9$

iteration 2 : current sol^{n}: $x = 2.3$ energy $= 0.49$

A random step $x = 1.9$

$f(1.9) = (1.9-3)^2 = 1.21$

The new sol^{n} is greater than 0.49

$T = 9$, there's a chance we will accept this worse sol^{n}.

$P(\text{accept}) = e^{-(1.21-0.49)/9} \approx 0.92$

$P = 0.92$ is high, the algorithm accepts the worse sol^{n} $x = 1.9$.

Cooling : $T = 9 \times 0.9 = 8.1$

22/10/24 good

Output

Clear

```
Iteration: 1, Temperature: 950.0000, Current Solution: 10.4345, Best
    Solution: 10.0000
Iteration: 2, Temperature: 902.5000, Current Solution: 10.8673, Best
    Solution: 10.0000
Iteration: 3, Temperature: 857.3750, Current Solution: 10.5390, Best
    Solution: 10.0000
Iteration: 4, Temperature: 814.5062, Current Solution: 9.9643, Best Solution
    : 9.9643
Iteration: 5, Temperature: 773.7809, Current Solution: 10.5985, Best
    Solution: 9.9643
Iteration: 6, Temperature: 735.0919, Current Solution: 11.5420, Best
    Solution: 9.9643
Iteration: 7, Temperature: 698.3373, Current Solution: 11.0790, Best
    Solution: 9.9643
Iteration: 8, Temperature: 663.4204, Current Solution: 11.9115, Best
    Solution: 9.9643
Iteration: 9, Temperature: 630.2494, Current Solution: 11.1281, Best
    Solution: 9.9643
Iteration: 10, Temperature: 598.7369, Current Solution: 10.3773, Best
    Solution: 9.9643
Best solution found: x = 9.9643, f(x) = 48.5011

=== Code Execution Successful ===
```