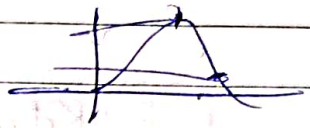
LAB 5:

Bafna Gold
Date: Page:

Simulated annealing



Step 1: Set T = T_initial
Step 2: set x_best = x_current
// Initialize the best solution

Step 3: while T > T_min
// Initialize T_min in begining,

Step 4: for i = 1 to max_iterations
// Initialize max_iterations.

Step 5 Generate a new solution
x_new by slightly modifying x_current.

Step 6 Calculate ΔE = f(x_new) − f(x_current),

Step 7 If ΔE < 0 (x_new is better).

Step 8. Accept x_new : x_current = x_new.

else:
Accept x_new with probability

P = exp(−ΔE / T) ≈
- if accepted, set
x_curr = x_new.

9. If f(x_curr) < f(x_best)
update x_best = current.

20

10. Reduce the temperature

$T = T * \alpha$.

11 Return x_best as the final solution.

Python code:

```
import random
import math

def objective_funct(x):
    return x**2

def s_a(in_sol, in_t, cool_rate, max_it):
    cur_sol = initial_sol
    curr_temp = init_temp
    best_sol = curr_sol
    best_val = objective_fun(best_sol)

    curr_value = best_val

    for it in range(max_it):

        new_sol = curr_sol + rand.uniform(-1,1)
        new_val = object_funct(new_sol)

        delta_val = new_val - curr_val

        if delta_val < 0 or rand.random()
            < math.exp(-delta_val / curr_t
```

```
curr_sol = new_sol
current_val = new_val

if new_val < best_val:
    best_sol = new_sol
    best_val = new_val

curr_t *= cool_rate

return best_sol, best_val

initial_sol = random.uniform(-10, 10)
initial_te. = 1000
cool_rate = 0.99
max_it = 1000

best_sol, best_val = simulated_an(in_sol, in_temp,
    cool_rate, max_it)

print(f"The best x: {best_sol}, It's corresponding
    f(x): {best_val}")
```

output:

The best x: -0.0008206 , It's corresponding f(x) = 6.7349e-07

dm
22/10/24