import math
import random
def objective function (x)
    return (x-3)**2

→ x value is 10
$f(10) = (10-3)^2 = (7)^2 = 49$

def simulated annealing (objective function, is, itemp, cooling rate, st, iterations)

    current_sol = is // 10
    current_val = objective function (x, current_sol) // 49

    best_sol = current_sol
    best_val = current_val

    temp = itemp // 1000
    iterations = 0 // 10
while iterations < max_iteration and temp > stopping temp
    then generate a new value by taking a random value and adding to current_sol
ns = current_sol + random.uniform(-1, 1)
        10 + 0.5 = 10.5
dv = new_val - current_val
        10.5 - 10 = 0.5
if dv<0
    If dv is negative it means the new solution has a lower value which makes it better for minimization
    current_sol = new_sol
    current_val = new_val

else if the new solution is worse don't accept it

Next check if the solution found is the best one
    if current_val < best_val
        best_sol = current_sol
        best_val = current_val

Increment the iterations
values: is = 10
        max_iteration = 10
        stopping_temp = 1e-8
        itemp = 1000
        cooling rate = 0.95

After every iteration decrease the temp by 5%
temp = temp * cooling_rate

Practical

```
import math
import random

def objective_function(x):
    return (x-3)**2

def simulated_annealing(objective_function, initial_so
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_iterations:

        new_solution = current_solution + random.uniform(-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/temperat
            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:{temperatu
        Current solution: {current_solution:4f}      Best solution: {best_solu
```

```
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:.4f}, f(x) = {best_value:.4f}")
```

Output:

```
iteration : 1   Temperature : 950.0000, current soln : 9.4775
iteration : 2   Temperature : 902.0000  current soln : 9.5096
iteration : 3   Temperature : 857.3750  current soln : 9.6366
iteration : 4   Temperature : 814.5062  current soln : 10.4510
iteration : 5   Temperature : 773.7809  current soln : 10.1823
iteration : 6   Temperature : 735.0918  current soln : 10.1549
iteration : 7   Temperature : 698.337   current soln : 10.6004
iteration : 8   Temperature : 663.4204  current soln : 10.993
iteration : 9   Temperature : 630.2494  current soln : 10.8792
iteration : 10  Temperature : 598.7369  current soln : 11.7439
Best soln : 9.475315
Best soln : 9.475315
Best soln : 9.475315
Best soln : 9.475315
Best soln : 9.475315
Best soln : 9.475315
Best soln : 9.475315
Best soln : 9.475315
Best soln : 9.475315
Best soln : 9.475315
```

22/10/24