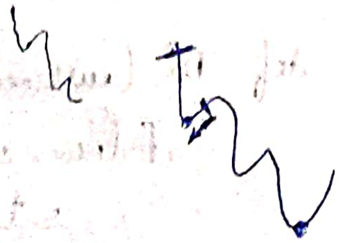
Lab-05



* Write an algorithm for simulated annealing

Objective function:- $x^2+5\sin x$

Step 1:-

```
def  SimulatedAnnealing (initial_state, initial_temp, cooling_rate, iterations)
     current_state = initial_state
     best_state = current_state
     best_cost = ObjectiveFun(current_state)
     temp = initial_temp
     while temp > 1 :
          for i <- 1 to iterations :
                new_state = Neighbour (current_state)
                curr_cost = ObjectiveFun (curr_state)
                new_cost = objective Fun (new_state)
                if AP(curr_cost, new_cost, temp)) > Random (0, 1)
                      curr_state = new_state
               if new_cost < best_cost :
                      best_state = new_state
                      best_cost = new_cost
          temp * = cooling_rate
     return (best_state, best_cost)
```

Step 2:-

```
def  ObjectiveFun (state):
     cost = 0
     for ele in state :
          cost + = ele^2 + sin(ele)
     return cost
```

Step 3:-

```
def  Neighbour (state):
     new_state = state.copy ()
     index = Random (0, length (state) - 1)
     new_state [index] + = Random (-1, 1)
     return new_state
```

22/10

```
def AP (curr_cost, new_cost, temp):
    if (new_cost < curr_cost):
        return 1
    else:
        return e^(-(curr_cost - new_cost)/temp)
```

$e^{-\Delta E k}$

Main function:-

```
def main():
    initial_temp = 1000
    coding_rate = 0.9
    iterations = "1000
    initial_state = [random.uniform (-10, 10) for _in is range(5)]
    best_state, best_cost = simu (initial_state, initial_temp, coling_rate, iterations)
    print (f"Best state : {best_state}")
    print (f"Best cost: {best_cost}")

if __name__ == "__main__":
    main()
```

Output:-

```
Best state: [-0.363922, -0.615657, -0.367044, -0.35904, -0.336619]
Best cost: -1.085552
```

22/10/24