USN : 1BM22CS235

LAB-1 : Genetc Algorithm for Optmizaton Problems

CODE: import numpy as np import random

```
def objectve_functon(x):
    return x ** 2

populaton_size = 100
num_generatons =
50
mutaton_rate = 0.1
crossover_rate = 0.7
range_min = -10
range_max = 10

# Create inital populaton
def initalize_populaton(size, min_val,
max_val):
    return np.random.uniform(min_val, max_val, size)

# Evaluate fitness of the
populaton
def evaluate_fitness(populaton):
    return np.array([objectve_functon(x) for x in populaton])

# Selecton using rouleΣe-wheel
method
def selecton(populaton, fitness):
    total_fitness = np.sum(fitness)
    probabilites = fitness / total_fitness
    return populaton[np.random.choice(range(len(populaton)), size=2,
    p=probabilites)]
```

```
# Crossover between two parents
def crossover(parent1, parent2):
    if random.random() < crossover_rate:
        return (parent1 + parent2) / 2 # Simple averaging for crossover
    return parent1 # No crossover

# Mutaton of an individual
def mutate(individual):
    if random.random() <
        return np.random.uniform(range_min, range_max)
    mutaton_rate:
    return individual

# Genetc Algorithm functon
def genetc_algorithm():
    # Step 1: Initalize populaton
    populaton = initalize_populaton(populaton_size, range_min,
    range_max)

    for generaton in range(num_generatons):
        # Step 2: Evaluate fitness
        fitness =

        evaluate_fitness(populaton)
        # Track the best soluton
        best_index = np.argmax(fitness)
        best_soluton =

        populaton[best_index]
        best_fitness = fitness[best_index]
        # print(f"Generaton {generaton + 1}: Best Soluton = {best_soluton}, Fitness
{best_fitness}")
        =
```

```
        # Step 3: Create new
        populaton new_populaton = []
        for _ in range(populaton_size):
            # Select parents parent1, parent2 =
            selecton(populaton, fitness) # Crossover to
            create offspring offspring = crossover(parent1,
            parent2) # Mutate offspring offspring =
            mutate(offspring)
            new_populaton.append(offspring)



        # Step 6: Replace old populaton with new
        populaton populaton = np.array(new_populaton)

    return best_soluton,
    best_fitness
# Run the Genetc Algorithm best_soluton, best_fitness =
genetc_algorithm() print(f"Best Soluton Found: {best_soluton},
Fitness: {best_fitness}")
```

OUTPUT:

```
Best Solution Found: -9.290037411642935, Fitness: 86.30479510972536
```