Lab - 02

04/10/24

program - 1.

python implementation of a genetic algorithm to solve a mathematical problem.

```
import numpy as np
import random

fitness = lambda x: x*np.sin(x)

population_size, generations = 10,20
crossover_rate, mutation_rate = 0.8,0.1
x_min, x_max = 0,10

population = np.random.uniform(x_min,x_max,
  population_size)

for generation in range(generations):
    fitness_values = np.array([fitness(x)for
x in population])
    fitness_values = fitness_values - np.min(
fitness_values)
    total_fitness = np.sum(fitness_values)
    selection_probs = fitness_values/total_
fitness if total_fitness > 0 else np.ones(
population_size)/population_size
    selected_population = np.random.
choice(population, size=population_size,
```

p = selection_probs)

offspring = []
for i in range(0, population_size, 2):
if random.random() < crossover_rate and i+1 < population_size:
parent1, parent2 = selected_population[i], selected_population[i+1]
alpha = random.uniform(0,1)
offspring.extend([alpha * parent1 + (1-alpha) * parent2, alpha * parent2 + (1-alpha) * parent1])
else:
offspring.extend([selected_population[i], selected_population[i+1]])

offspring = [x + random.uniform(-0.5, 0.5) if random.random() < mutation_rate else x for x in offspring]
population = np.clip(offspring, x_min, x_max)

best_individual = max(population, key=fitness)
print(f"Generation {generation+1}: Best Solution x = {best_individual}, f(x) = {fitness(best_individual)}")

best_individual = max(population, key=fitness)
print(f"In final Best Solution: x =

print("best_individual:", fitness(best_individual))

output:

Final best solution: x = 7.942354

f(x) = 7.911360

Sol-1

Genetic algorithm

Step 1: Identify the objective function to optimize in this case maximize $f(x) = x^3$
Step 2: Set the following parameters population size as 100 mutation rate as 0.1 crossover rate as 0.8 number of generations as 50, lower bound as -10, upper bound as 10.
Step 3: Generate an initial population of 100 random individuals within the range of -10 to 10.
Step 4: For each individual in the population compute etc fitness using fitness function $f(x) = x^3$
Step 5: Use Roulette where selection to select two parents from the population
Step 6: For the selected parents, perform crossover with a probability of 0.8
Step 7: For each offspring apply mutation with a probability of 0.8
Step 8: Collect the newly created offspring until the new population reaches the original population size
Step 9: Replace old population with new generation of individual.