Lab - I

Genetic algorithm for optimization

```
import random
import numpy as np

def fitness_function(x):
    return x**2
population_size = 10
generations = 50
mutation_rate = 0.1
crossover_rate = 0.8

def create_population(size):
    return np.random.uniform(-10, 10, size)

def evaluate_fitness(population):
    return np.array([fitness_function(ind) for ind in
                     population])
def select_parents(population, fitness):
    total_fitness = np.sum(fitness)
    selection_probs = fitness / total_fitness
    return population[np.random.choice(len(population),
        size=2, p=selection_probs)]
def crossover(parent1, parent2):
    if random.random() < cross_rate:
        alpha = random.random()
        child = alpha * parent1 + (1-alpha) * parent2
        return child
    return parent1
```

```
def mutate(child):
    if random.random() < mutation_rate:
        mutation_point = random.uniform(-10,10)
        return mutation_point
    return child
def genetic_algorithm():
    population = create_population(population)
    for generation in range(generations):
        fitness = evaluate_fitness(population)
    best_fitness = np.max(fitness)
    best_individual = population[np.argmax(fitness)]
    print(f"Generation {generation}: Best Fitness
    {best_fitness}, Best Individual = {best_individual}
    new_population = []
    for _ in range(population_size//2):
        parent1, parent2 = select_parents(population,fitness)
        child1 = crossover(parent1, parent2)
        child2 = crossover(parent2, parent1)
        child1 = mutate(child1)
        child2 = mutate(child2)
        new_population.extend([child1, child2])
    population = np.array(new_population)
final_fitness = evaluate_fitness(population)
best_individual = population[np.argmax(final_fitness)]
return best_individual, np.max(final_fitness)

Best_solution, best_fitness = generate_algorithm()
print(f"Best solution: {best_solution} with Fitness:
                {best_fitness}")
```

Output:

Best solution: -9.925 with Fitness: 98.5122 at Generation: 31

Algorithm:-

Step 1 :- Identify the objective fn to optimize in this case maximize $f(x) = x^2$

Step 2 :- Set the fol parameters population size, mutation rate, crossover rate, no of generation, lower bound upper bound

Step 3:- Generate an initial ~~po~~ population within the range of -10 to 10.

Step 4 :- For each individual in population compute fitness using fitness function $f(x) = x^2$

Step 5:- Use Roulette wheel 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 individuals

Step 10 :- After final generation, evaluate fitness of the population