Experiment 07 : Implementation of Simple Genetic Application.

Learning Objective : Write a program for simple genetic algorithm.

Tools : Python

Theory :

One example of a simple genetic application is the breeding of plants and animals to produce desirable traits. This process is often referred to as selective breeding or artificial selection. In selective breeding, individuals with desirable traits are chosen to breed and produce offspring with those desirable traits. Over time, this can lead to the development of new varieties or breeds of plants and animals that exhibit specific characteristics.

For example, farmers may selectively breed cows for increased milk production or selectively breed crops for greater resistance to pests or drought. Similarly, dog breeders may selectively breed dogs for specific physical and behavioral traits.

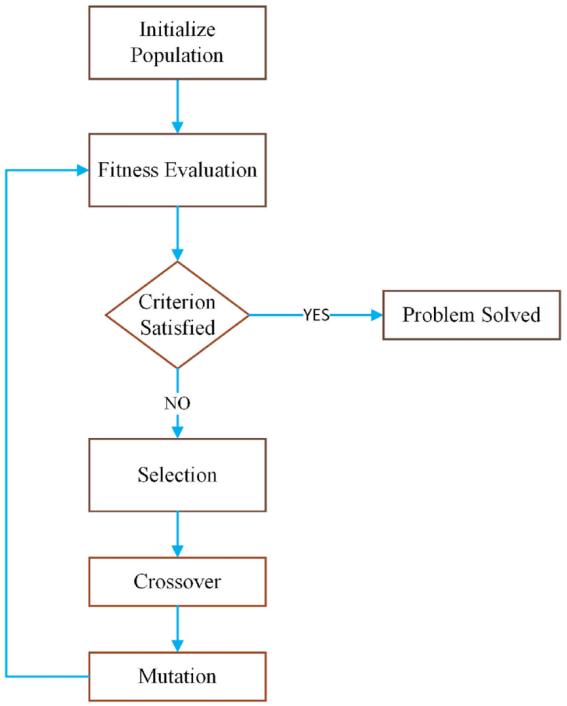
This process is similar to natural selection, where certain traits that are advantageous for survival and reproduction becomes more common in a population over time. However, in selective breeding, humans are deliberately choosing which traits to promote and which to eliminate, rather than relying on natural selection.

The use of genetic engineering is another example of a genetic application. Genetic engineering involves manipulating the DNA of an organism to introduce new traits or modify existing ones. This can be done by inserting or deleting genes, or by altering the expression of existing genes.

For example, scientists may use genetic engineering to create crops that are more resistant to pests or produce higher yields, or to create bacteria that can produce useful chemicals or enzymes. Genetic engineering is also used in medical research to create new treatments for genetic diseases or to develop vaccines.

Algorithm:

1. Select Np individuals from the previous population.
2. Create the mating pool randomly.
3. Perform Crossover.
4. Perform Mutation in offspring solutions.
5. Perform inversion in offspring solutions.
6. Replace the old solutions of the last generation with the newly created solutions and go to step (2).



Implementation :

[ ]:
```
import random

# Define the fitness function (change this to match your problem)
def fitness(individual):
    return sum(individual)

# Define the population size and length of the individuals
POPULATION_SIZE = 100
INDIVIDUAL_LENGTH = 10

# Create the initial population
population = []
for i in range(POPULATION_SIZE):
    individual = [random.randint(0, 1) for j in range(INDIVIDUAL_LENGTH)]
    population.append(individual)

# Evolution loop
for generation in range(10): # Change the number of generations here
    # Evaluate the fitness of each individual
    fitness_scores = [fitness(individual) for individual in population]

    # Select the fittest individuals
    selected = []
    for i in range(2):
        i1, i2 = random.choices(population, weights=fitness_scores, k=2)
        selected.append(i1 if fitness(i1) > fitness(i2) else i2)

    # Create the next generation by crossover and mutation
    population = []
    for i in range(POPULATION_SIZE):
        parent1, parent2 = random.choices(selected, k=2)
        child = [parent1[j] if random.random() < 0.5 else parent2[j] for j in␣
↪range(INDIVIDUAL_LENGTH)]
        if random.random() < 0.1: # Mutation rate
            child[random.randint(0, INDIVIDUAL_LENGTH-1)] = random.randint(0, 1)
        population.append(child)
```

```python
# Find the fittest individual in the final population
fitness_scores = [fitness(individual) for individual in population]
fittest = population[fitness_scores.index(max(fitness_scores))]

print("Fittest individual:", fittest)
print("Fitness score:", fitness(fittest))
print (population)
```

```
Fittest individual: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
Fitness score: 10
[[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 0, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 0, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,

1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1,
1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1]]
```

Result and Discussion :

Result and Discussion :

Learning Outcomes : Students should have the ability to

LO 7.1: Ability to understand genetic algorithms.
LO 7.2: Ability to implement the Simple Genetic Application.

Course Outcomes :

CO : Understand and implement Simple Genetic Application.

Conclusion :

Viva Questions :

Q1. What is selective breeding, and how is it used to produce desirable traits in plants and animals?
Q2. What is genetic engineering, and how is it used to introduce new traits or modify existing ones in organisms?

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	Total
Marks Obtained				