OPEN ENDED LAB REPORT

Artificial Intelligence (CS-323)

Third Year-Computer and Information Systems Engineering

Batch: 2022

Group Members:

Rohit Dhanjee …..…… (CS-22085)

Asnaif Asim ……...…….... (CS-22092)

Hasnain Rizwan …….……...…..…. (CS-22096)

Submitted to: Miss Hameeza Ahmed

Submission Date: 25/11/2024

Table of Contents

Introduction:

Genetic algorithms simulate the process of natural selection which means those species that can adapt to changes in their environment can survive and reproduce and go to the next generation. In simple words, they simulate “survival of the fittest” among individuals of consecutive generations to solve a problem. Each generation consists of a population of individuals and each individual represents a point in search space and possible solution. Each individual is represented as a string of character/integer/float/bits. This string is analogous to the Chromosome.

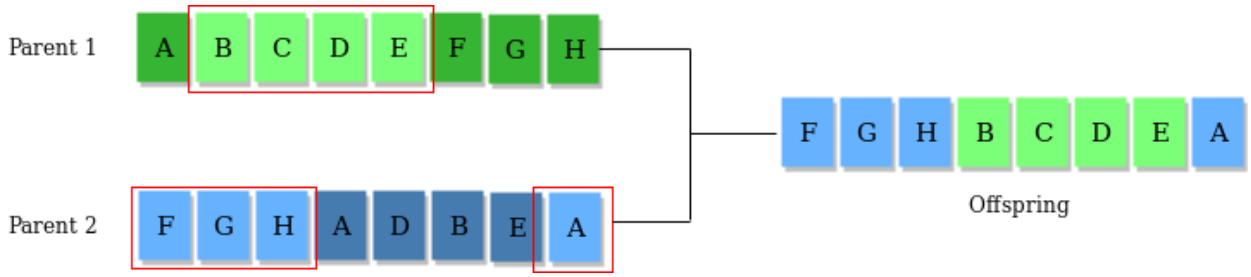
Operators of Genetic Algorithms:

Once the initial generation is created, the algorithm evolves the generation using following operators –

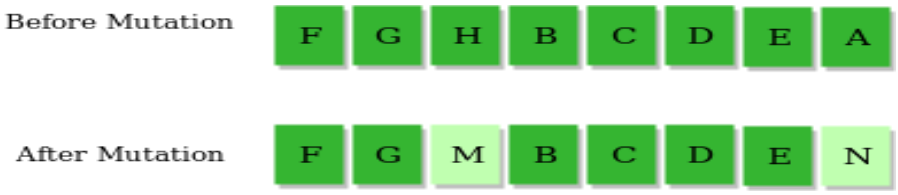
1) Selection Operator: The idea is to give preference to the individuals with good fitness scores and allow them to pass their genes to successive generations.

2) Crossover Operator: This represents mating between individuals. Two individuals are selected using selection operator and crossover sites are chosen randomly. Then the genes at these crossover sites are exchanged thus creating a completely new individual (offspring). For example

–



3) Mutation Operator: The key idea is to insert random genes in offspring to maintain the diversity in the population to avoid premature convergence. For example –



Simple Example of Genetic Algorithm: Finding the Maximum Value of a Function

Problem:

Maximize the function $f(x)=x^2$, where x is an integer in the range [0, 31].

Representation:

Each solution x is represented as a 5-bit binary string. For example, x=5 is represented as 00101.

Steps of Simulation:

1. **Initialization:** Generate a random population of binary strings (chromosomes).
2. **Fitness Evaluation:** Calculate the fitness of each chromosome as $f(x)=x^2$, where x is the integer decoded from the binary string.
3. **Selection:** Use roulette-wheel selection to select parents for reproduction, where the selection probability is proportional to the fitness.
4. **Crossover:** Perform single-point crossover between selected parents to produce offspring.
5. **Mutation:** Introduce random mutations in the offspring to maintain diversity.
6. **Repeat:** Continue the process for a predefined number of generations or until convergence.

Explanation of Output:

The program begins with a random population of binary strings, evaluates their fitness values, and evolves the population over multiple generations. Each generation should get closer to finding the maximum x^2, with x=31, represented as 11111.

Problem Statement:

Objective

Genetic algorithms (GAs) are inspired by the principles of natural selection and are used to solve optimization and search problems. This project applies a GA to generate a predefined target string starting from a random string population. The fitness of each individual string is evaluated based on its similarity to the target.
The goal is to replicate the target string:
"Can we meet today?"
using a genetic algorithm.

Relevance

The problem showcases how GAs work with non-numerical solutions, making it relevant for applications such as text generation, cryptography, and similar fields. The algorithm starts with a random population of strings and evolves the population to minimize the fitness score, which is defined as the number of mismatched characters between an individual string and the target.

Challenges:

- Representing solutions as chromosomes (strings).
- Defining a meaningful fitness function for string similarity.
- Implementing genetic operators tailored for text manipulation.

Implementation:

Parameters:

- **Population Size:** 100
- **Genes:** All valid characters, including uppercase, lowercase, spaces, and punctuation.
- **Target String:** *"Can we meet today?"*
- **Mutation Rate:** Controlled probabilistically to maintain diversity.

Algorithm Steps:

1. **Initialization:** Generate a population of random strings of the same length as the target.
2. **Fitness Function:** Measure string similarity by counting character mismatches.
3. **Selection:** Select parents based on fitness, with elitism preserving the top 10% of individuals.
4. **Crossover:** Combine genes from two parents to create offspring.
5. **Mutation:** Replace random genes with new characters to introduce diversity.
6. **Iteration:** Evolve the population until the target string is generated or a maximum number of generations is reached.

Results:

Simulation Results:

The algorithm effectively evolved the population to replicate the target string.

Output Example:

```
Generation: 2   String:  3Q4wcX LezU7n1a2$  Fitness: 15
Generation: 3   String:  3Q4wcX LezU7n1a2$  Fitness: 15
Generation: 19  String: {a2 we meet :odayS  Fitness: 4
Generation: 20  String: Lav we meet Ooday?  Fitness: 3
Generation: 114 String: Can we meet eoday?  Fitness: 1
Generation: 115 String: Can we meet eoday?  Fitness: 1
Generation: 116 String: Can we meet today?  Fitness: 0
```

Conclusion

The genetic algorithm successfully generated the target string. This project highlights the adaptability of GAs to non-numeric problems and demonstrates their effectiveness in optimizing solutions iteratively.