**Experiment-2.2**

**Aim:** Implementation Genetic Application – Match WordFinding.

**Theory:**

Genetic Algorithms(GAs) are adaptive heuristic search algorithms that belong to the larger part of evolutionary algorithms. Genetic algorithms are based on the ideas of natural selection and genetics. These are intelligent exploitation of random searches provided with historical data to direct the search into the region of better performance in solution space. They are commonly used to generate high-quality solutions for optimization problems and search problems.

Genetic algorithms are based on an analogy with the genetic structure and behavior of chromosomes of the population. Following is the foundation of GAs based on this analogy –

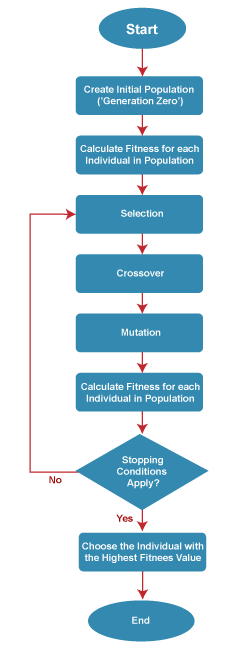
1. Individuals in the population compete for resources and mate
2. Those individuals who are successful (fittest) then mate to create more offspring than others
3. Genes from the “fittest” parent propagate throughout the generation, that is sometimes parents create offspring which is better than either parent.
4. Thus each successive generation is more suited for their environment.

The genetic algorithm works on the evolutionary generational cycle to generate high-quality solutions. These algorithms use different operations that either enhance or replace the population to give an improved fit solution.

It basically involves five phases to solve the complex optimization problems, which are given as below:

* Initialization
* Fitness Assignment
* Selection
* Reproduction
* Termination

Following is the workflow of Simple Genetic Algorithm:



**Code for Experiment :**

% Parameters

TARGET\_WORD = 'class';

POPULATION\_SIZE = 100;

MUTATION\_RATE = 0.1;

MAX\_GENERATIONS = 1000;

fprintf('Target word: %s\n', TARGET\_WORD);

% Initialize population

population = cell(POPULATION\_SIZE, 1);

for i = 1:POPULATION\_SIZE

population{i} = generateRandomWord(length(TARGET\_WORD));

end

% Main loop

for generation = 1:MAX\_GENERATIONS

% Calculate fitness for each individual

fitness\_scores = zeros(POPULATION\_SIZE, 1);

for i = 1:POPULATION\_SIZE

fitness\_scores(i) = calculateFitness(population{i}, TARGET\_WORD);

end

% Check if target word is found

[best\_fitness, idx] = max(fitness\_scores);

best\_word = population{idx};

if strcmp(best\_word, TARGET\_WORD)

fprintf('Generation %d: Best Word = %s, Fitness = %d\n', generation, best\_word, best\_fitness);

fprintf('Target word found: %s\n', best\_word);

break;

end

% Selection: Roulette wheel selection

total\_fitness = sum(fitness\_scores);

probabilities = fitness\_scores / total\_fitness;

selected\_indices = randsample(1:POPULATION\_SIZE, POPULATION\_SIZE, true, probabilities);

selected\_population = population(selected\_indices);

% Crossover and Mutation

new\_population = cell(POPULATION\_SIZE, 1);

for i = 1:2:POPULATION\_SIZE

parent1 = selected\_population{i};

parent2 = selected\_population{i+1};

point = randi(length(TARGET\_WORD) - 1) + 1;

child1 = [parent1(1:point) parent2(point+1:end)];

child2 = [parent2(1:point) parent1(point+1:end)];

% Mutation

if rand < MUTATION\_RATE

mutate\_index = randi(length(TARGET\_WORD));

child1(mutate\_index) = char(randi([97, 122])); % random lowercase letter

end

if rand < MUTATION\_RATE

mutate\_index = randi(length(TARGET\_WORD));

child2(mutate\_index) = char(randi([97, 122])); % random lowercase letter

end

new\_population{i} = child1;

new\_population{i+1} = child2;

end

population = new\_population;

% Print best word in each generation

fprintf('Gen %d: Best Word = %s, Fitness = %d\n', generation, best\_word, best\_fitness);

end

if generation == MAX\_GENERATIONS

fprintf('Max generation reached. Target word not found.\n');

end

% Function to generate a random word of given length

function word = generateRandomWord(length)

letters = 'abcdefghijklmnopqrstuvwxyz';

word = letters(randi(numel(letters), [1, length]));

end

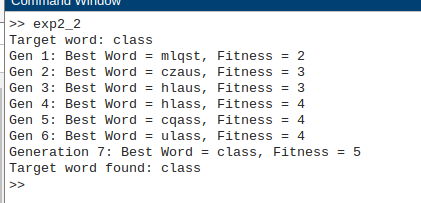
% Function to calculate fitness of a word

function fitness = calculateFitness(word, target\_word)

fitness = sum(word == target\_word);

end

**Result/Output :**



**Learning outcomes:**

1. Learned about Genetic Algorithm and its workflow.
2. Learned how to implement word-matching using genetic algorithm..
3. Learned to implement Genetic Algorithm in MATLAB.