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.

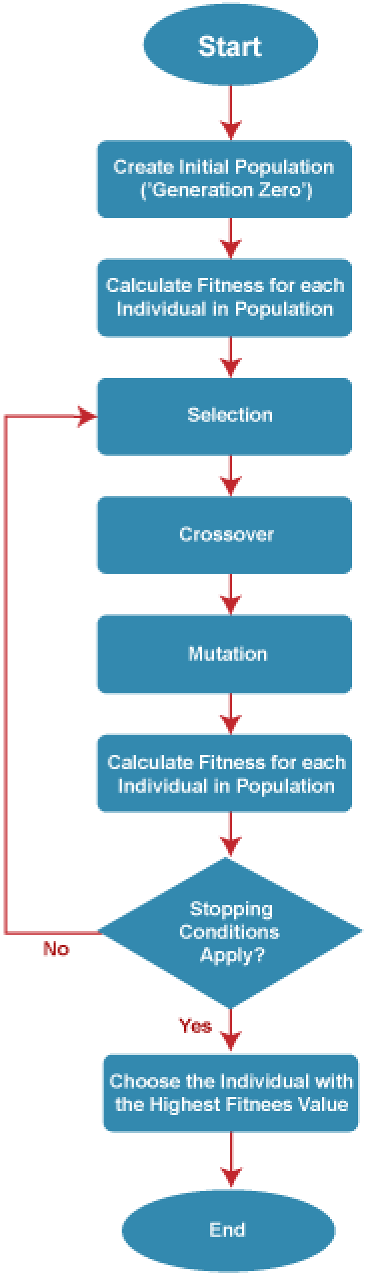
The genetic algorithm works on the evolutionary generational cycle to generate high-quality solutions. These algorithms use different operations that either enhance or replace the population to give an improved fit solution.

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:


Start
Create Initial Population ('Generation Zero')
Calculate Fitness for each Individual in Population
Selection
Crossover
Mutation
Calculate Fitness for each Individual in Population
Stopping Conditions Apply?
No
Yes
Choose the Individual with the Highest Fitnees Value
End

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
```

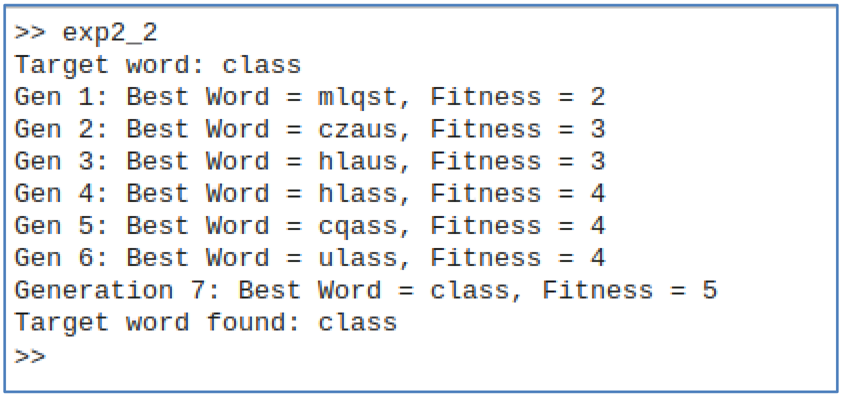
```
total_fitness = sum(fitness_scores);
   probabilities = fitness_scores / total_fitness;
   selected_indices = randsample(1:POPULATION_SIZE, POPULATION_SIZE, true, probabilities);
   selected_population = population(selected_indices);

   % 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 :



```
>> exp2_2
Target word: class
Gen 1: Best Word = mlqst, Fitness = 2
Gen 2: Best Word = czaus, Fitness = 3
Gen 3: Best Word = hlaus, Fitness = 3
Gen 4: Best Word = hlass, Fitness = 4
Gen 5: Best Word = cqass, Fitness = 4
Gen 6: Best Word = ulass, Fitness = 4
Generation 7: Best Word = class, Fitness = 5
Target word found: class
>>
```

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.