Genetic Algorithm & Fuzzy Logic

Semester-5

Practical-2

Name: Shivam Tawari

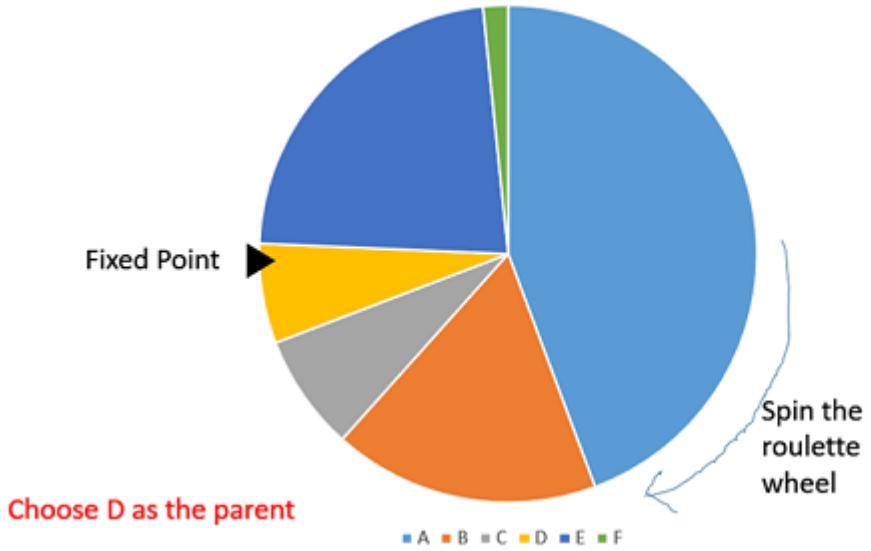
Roll no: A-58

Aim: Implement roulette wheel selection scheme for given f(x) optimization (maximization) using GA

Theory:

Roulette Wheel Selection:

In a roulette wheel selection, the circular wheel is divided as described before. A fixed point is chosen on the wheel circumference as shown and the wheel is rotated. The region of the wheel which comes in front of the fixed point is chosen as the parent. For the second parent, the same process is repeated.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

It is clear that a fitter individual has a greater pie on the wheel and therefore a greater chance of landing in front of the fixed point when the wheel is rotated. Therefore, the probability of choosing an individual depends directly on its fitness.

Implementation wise, we use the following steps −

- Calculate S = the sum of a finesses.
- Generate a random number between 0 and S.
- Starting from the top of the population, keep adding the finesses to the partial sum P, till P<S.
- The individual for which P exceeds S is the chosen individual.

Code:

```
[1]  # Shivam Tawari A-58
```

```
[2]  # Usual imports
     import numpy as np
```

```
[3]  def initial_pop(pop, length):
         """
             Function to Generate Initial Population
             Inputs:
                 - pop: Population Size (integer), greater than 0
                 - length: Length of Chromosome (integer), greater than 0
             Outputs:
                 - parents: Initial population
                           An Array of dimension (pop, length) with 0/1 entry
         """
         parents = np.random.randint(0, 2, size=(pop, length))
         return parents
```

```
[4]  pop = 10
     length = 6
```

```
[5]  # Generate initial population
     init_pop = initial_pop(pop, length)
     init_pop
```

```
array([[1, 1, 0, 0, 0, 0],
       [0, 0, 1, 0, 1, 0],
       [0, 1, 1, 0, 1, 1],
       [0, 0, 0, 1, 0, 1],
       [0, 0, 1, 0, 1, 1],
       [0, 1, 1, 0, 0, 1],
       [0, 1, 0, 0, 1, 0],
       [0, 0, 1, 1, 1, 1],
       [0, 0, 0, 0, 0, 1],
       [0, 1, 0, 1, 0, 0]])
```

```
[6]  # Assign Random Fitness Values
     init_fit = np.random.randint(0, 100, pop)

     init_fit
```

```
array([97, 51, 93, 30, 57, 29, 16, 99, 38, 44])
```

```
[7]  def roulette_selection(chromosomes, fitness, select=3):
         """
             Function to select n chromosomes, based on Roulette Wheel
             Input:
                 - chromosomes: an array of m chromosome
                 - fitness: fitness value of chromosomes
                 - select: selection of n chromosomes (<m)
             Output:
                 - selected: Selection of n chromosomes out of m
                             An array of dimension (n, length of chromosome)
         """
         if (select>chromosomes.shape[0]):
             # Display Warning and select with default value
             print(f'Number of selected chromosomes n: {select}, ' +
                   f'can not be greater than number of chromosomes m: {chromosomes.shape[0]}')
             select = 3
             print(f'Select default: {select} chromosomes')

         # Convert fitness to probabilities
         fitness_prob = fitness/np.sum(fitness, axis=0)

         # Select chromosomes
         selected_idx = np.random.choice(np.arange(chromosomes.shape[0]),
                                         select, p=fitness_prob)
         selected = chromosomes[selected_idx]

         return selected
```

```
[8]  roulette_selection(init_pop, init_fit, 4)
```

```
array([[0, 1, 1, 0, 1, 1],
       [0, 1, 1, 0, 0, 1],
       [0, 0, 1, 0, 1, 0],
       [1, 1, 0, 0, 0, 0]])
```

Conclusion: Hence roulette wheel selection scheme has been implemented successfully.