Genetic Algorithm & Fuzzy Logic

Semester-5

Practical-5

Name: Shivam Tawari

Roll no: A-58

Aim: Implement flipping mutation for binary coded GA.

Theory:

Mutation:

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. It is analogous to biological mutation. Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Hence GA can come to a better solution by using mutation.

Mutation occurs during evolution according to a user-definable mutation probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search.

The classic example of a mutation operator involves a probability that an arbitrary bit in a genetic sequence will be flipped from its original state.

A common method of implementing the mutation operator involves generating a random variable for each bit in a sequence. This random variable tells whether or not a particular bit will be flipped. This mutation procedure, based on the biological point mutation, is called single point mutation. Other types are inversion and floating-point mutation. When the gene encoding is restrictive as in permutation problems, mutations are swaps, inversions, and scrambles.

The purpose of mutation in GAs is to introduce diversity into the sampled population. Mutation operators are used in an attempt to avoid local minima

by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping convergence to the global optimum.

This reasoning also leads most GA systems to avoid only taking the fittest of the population in generating the next generation, but rather selecting a random (or semi-random) set with a weighting toward those that are fitter.

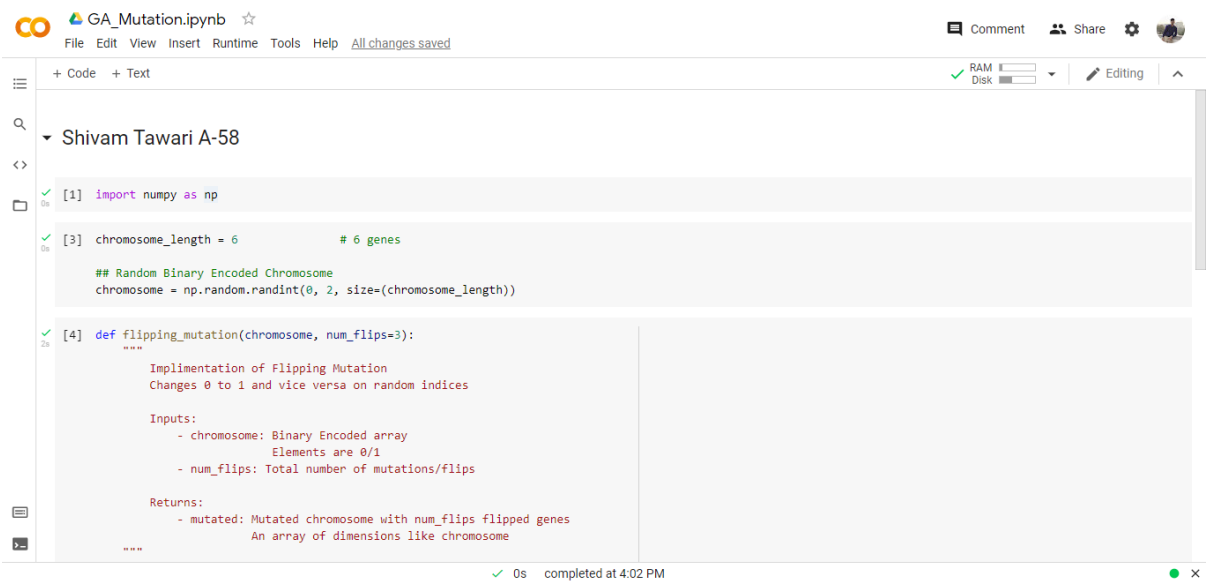
The Flipping Mutation ensue through bit flips at random positions.

Example:

1	0	1	0	0	1	0
				↓		
1	0	1	0	1	1	0

The probability of a mutation of a bit is 1/l, where l is the length of the binary vector. Thus, a mutation rate of 1 per mutation and individual selected for mutation is reached.

Code:



GA_Mutation.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

[4]

```
                                Elements are 0/1
                - num_flips: Total number of mutations/flips

            Returns:
                - mutated: Mutated chromosome with num_flips flipped genes
                           An array of dimensions like chromosome
        """
        indxs = np.random.choice(range(0, chromosome.shape[0]), num_flips, replace=False)

        mutated = chromosome.copy()

        for indx in indxs:
            if chromosome[indx] == 0:
                mutated[indx] = 1
            else:
                mutated[indx] = 0

        return mutated
```

Output:

GA_Mutation.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

[5]

```
mutated = flipping_mutation(chromosome)

print(f'Chromosome: {chromosome}')

print(f'Mutated: {mutated}')
```

```
Chromosome: [1 1 0 0 1 1]
Mutated: [1 0 1 0 0 1]
```

Conclusion: Hence, flipping mutation for binary coded GA has been implemented successfully.