

**Department of Computer Science and Engineering(UG Studies)**

**PES University, Bangalore-560085**

|  |  |
| --- | --- |
| **Session :** Aug - Dec 2017  **Credits :** 0-0-2-0-1 | UE14CS405 : Machine Learning Lab |
| **Lab # :** 08 | Experiment on Genetic Algorithm |

**Introduciton:**

In computer science and operations research, a genetic algorithm (GA) is a meta heuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on bi o-inspired operators such as mutation, crossover and selection.

The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a *generation*. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

A typical genetic algorithm requires:

1. a genetic representation of the solution domain,
2. a fitness function to evaluate the solution domain.

A standard representation of each candidate solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case.

Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators.

**The basic process for a genetic algorithm is:**

1. Initialization - Create an initial population. This population is usually randomly generated and can be any desired size, from only a few individuals to thousands.
2. Evaluation - Each member of the population is then evaluated and we calculate a 'fitness' for that individual. The fitness value is calculated by how well it fits with our desired requirements.
3. Selection - We want to be constantly improving our populations overall fitness. Selection helps us to do this by discarding the bad designs and only keeping the best individuals in the population.
4. Crossover - During crossover we create new individuals by combining aspects of our selected individuals.
5. Mutation - We need to add a little bit randomness into our populations' genetics otherwise every combination of solutions we can create would be in our initial population. Mutation typically works by making very small changes at random to an individuals genome.
6. And repeat! - Now we have our next generation we can start again from step two until we reach a termination condition.

**Problem Description :**

The program tries to find a set of numbers between 0 and 9

and uses +,-, \* and / to get an expression to arrive at the specified input number.

**For example:**

If input number is 1000,

Then one of the correct expressions is:  
  
8\*5\*8+7\*3+9+8+2

(Evaluation is left to right, no precedence is given to any operator over another).

So, input is a number. Output is the expression using numbers

and operators to arrive at the number.

**TO DO:**

Only prototype for the below functions is given. Please fill the code.

1. roulette()

2. crossover()

3. mutate()

4. calcFitness()

Program takes 200 as input

**Some of the applications of Genetic Algorithms:**

Bayesian inference links to particle methods in Bayesian statistics and hidden Markov chain models

Bioinformatics Multiple Sequence Alignment

Bioinformatics: RNA structure prediction