National University of Computer & Emerging Sciences

Lab Manual

CS461: Artificial Intelligence Lab

Course Instructor	Dr. Hafeez-Ur-Rehman
Lab Instructor	Muhammad Yousaf
Semester	Spring 2021

Genetic Algorithm

Genetic Algorithm is a part of **Evolutionary** Algorithms, specifically used to generate high-quality solutions to

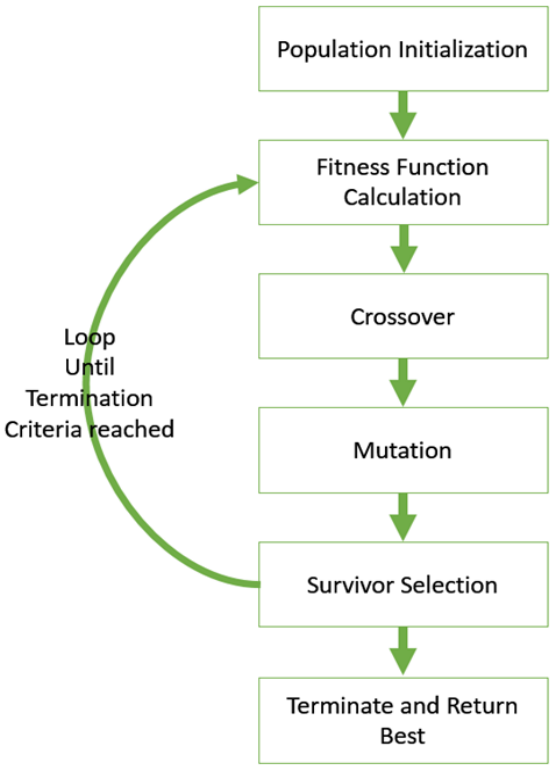
1. **Optimization problems** and
2. **search problems**

by relying on biologically inspired operators such as mutation, crossover, and selection.

Introduction:

Evolution usually starts from a population of randomly generated individuals in the form of iteration. (Iteration will lead to a new generation). In every iteration or generation, the fitness of each individual is determined to select the fittest. Genome fittest individuals selected are mutated or altered to form a new generation, and the process continues until the best solution has reached. The process **terminates** under 2 scenarios-

1. When maximum number of generations have been created
2. Fitness level reached is sufficient



Lab # 11: Genetic Algorithms

Step 1: Initial Population

Select any random states as initial population i.e.
No. of population is a random or your choice.
Values in each population represents a random state.

P1 =	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	Fitness = 1
P2 =	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	Fitness = 2
P3 =	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	Fitness = 2
P4 =	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	Fitness = 4

Calculate the Fitness Value of each Population

Fitness (fn) = No. of 1's in a population

Step 2: Selection of Parents

1. Parents will be selected random from the population in the first step.
2. High fitness value population will have high chances of selection as a parent i.e. P4 will have high weightage to be selected as randomly

Selection_Probability P(i) = Fitness of P(i)/Total Fitness of all Populations

Step 3: Modification

Crossover:

1. This is called convergence step because it will generate children (new states)
2. Create children from meeting of the parents
3. Generate two children/successor from two parents (new version of genetic algorithm produce one child instead)

Mutation:

1. Apply mutation on children in order to get the **new/updated** state (assuming it will quickly lead us to goal)
2. Mutation probability or mutation rate is fixed and chosen a very small value
   i.e. 0.01 means generate a random no. between (1-100), mutate the child if random no. is 1 else skip.
   i.e. 0.2 means generate a random no. between (1-10), mutate the child if random no. is 1 or 2. Skip otherwise
3. Mutation rate will be applied and checked for each digit/char in a population
   i.e. You will keep repeating this process for each array value in a single population means 16 times random no. will be generated and checked respectively.

Step 4: Evaluation

1. Compute the fitness values of newly generated children
2. Apply the goal test
3. Replace the old population with newly created population having new children
4. Repeat the steps 1-4 if goal is not found.

Advantages:

- Modular, separate from application
- Answer gets better with time
- Inherently parallel; easily distributed
- Genetic algorithms use fitness score, which is obtained from objective functions, without other derivative or auxiliary information.

Disadvantages:

- Genetic Algorithms might be costly in computational terms since the evaluation of each individual requires the training of a model.
- These algorithms can take a long time to converge since they have a stochastic nature.

Task

- We will start implementing problem of 2D array of 4x4
- Initial state = 4x4 array filled with all zero's
- Goal state = 4x4 array filled with all one's
- Fitness function of a state = no. of 1's in a state
- 1st step would be you'll convert 4x4 2d array into 1-D array of length 16
- You will process 1-D array only because genetic algorithm works on chromosomes of 1D arrays (strings of chars/digits)