International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:268
Vol.3, No.2, February 2014 DOI:10.15693/ijaist/2014.v3i2.115-119

Survey on Genetic Algorithms & Basic Operators

Krishna Lad
PG
Student,CHARUSAT,Changa
,Anand, Guajarat, India

Meenakshi Agrawal
PG
Student,CHARUSAT,Changa,
Anand, Guajarat, India

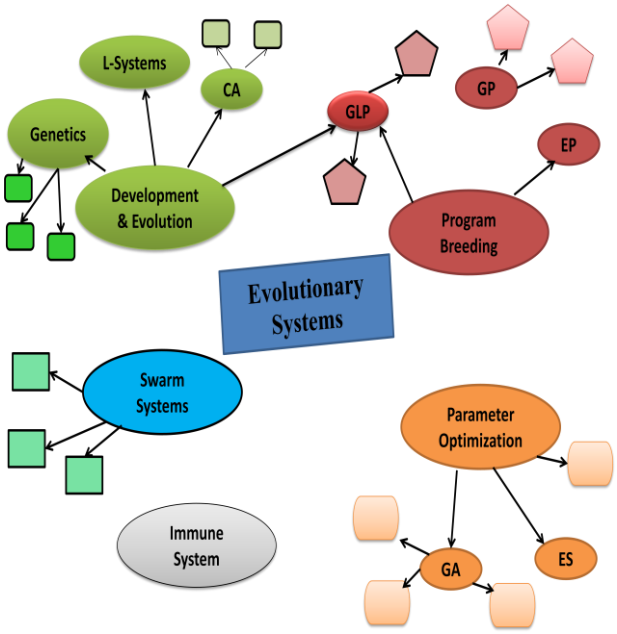
Mr. Mrudang Pandya
Assistant Professor,
CHARUSAT, Changa, Anand,
Gujarat, India

***Abstract*— Genetic Algorithm (GA) is the part of evolutionary computing, This paper examines the introduction of Genetic Algorithms, basic operators of GA, and applications of GA.**

Index terms -Application of GA, cross over, GA, Genetic Algorithm, mutation, recombination, selection

I. INTRODUCTION

Soft computing is a term applied to a field within computer science which is characterized by the use of inexact solutions to computationally hard tasks such as the solution of NP-Complete problems, for that there is no known algorithms that can compute an exact solution in polynomial time. Soft computing differs from hard computing in that, not like hard computing, it is tolerant



Evolutionary System

of uncertainty, imprecision, partial truth, and partially. Component of Soft Computing include Neural Network, Fuzzy logic, Evolutionary computation, Chaos theory etc... [1][6].

A. Genetic Algorithm (GA)

Genetic Algorithm (GA) are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. GA is a part of Evolutionary computing, a speedily growing area of Artificial Intelligence (AI). GAs is inspired by **Darwin's theory** about evolution - "**survival of the fittest**". It's provide an intelligent utilization of arbitrary search used to solve optimization problems. In GA solving problems mean looking for solutions, which is best among others. Optimization is a process that finds a best solution for a problem. There are three main centred factors for optimization problem are [1][6]

1) **Objective Function**: Which is either minimize or maximize [2]
2) **Set of Unknowns or variables:** That affects the objective functions [2].
3) **Set of Constrains:** That allows the unknowns to take on certain values but exclude others [2].

B. Simple Genetic Algorithm

Algorithm is started with a set of solutions called the population. It is also referred as chromosomes. Solutions from one population are taken and used to form a new population. This is motivated by hope that new population will be better than the old population. Solutions which are selected to form new solutions are selected according to their fitness- the more suitable they are the more chances they have to reproduce.[3]

Algorithm:

```
Simple Genetic Algorithm
{
        Initialize population
        Evaluate population/fitness of population
   (Chromosomes)
        while termination condition not glad
        {
                        Select parents for reproduction
                        Perform
crossover/recombination and Mutation
                        Evaluate                population
(Chromosomes)
        }
} [2]
```

Here, reproduction, cross over and mutation which are the basic operators which are applied in to GA. Now we discuss the detail about those operators in next section

II. OPERATORS OF GA

A genetic operators is an operators used in genetic algorithm to maintain genetic diversity knows as mutation and to combine existing solutions into others, crossover The main difference between them is that the mutation operators operate on one population that is they are single, while the reproduction operators are binary operators. Genetic variations are a necessity for the process of evolution. Genetic operators are used in

genetic algorithms analogous to those in the natural world; survival of the s**election**, or fittest; **reproduction** (crossover, also called recombination); and **mutation**. [3]

Type of Operators

A. Selection

Selection is the stage of a genetic algorithm in which individual genomes are chosen from chromosome for later breeding (recombination or crossover).[3][5]

A generic selection procedure can be implemented as follows:

- The fitness function is evaluated for every individual, providing fitness values, which are then normalized. Normalization defines dividing the fitness value of every individual by the sum of all fitness values, so that, sum of all resulting fitness values equals 1.
- Chromosomes are sorted by descending fitness values.
- Accumulated normalized fitness values are computed. The accumulated fitness of the last individual should be 1 (otherwise something went wrong in the normalization step).
- A value of random number *R* between 0 and 1 is chosen.
- The selected individual is the first one whose accumulated normalized value is greater than *R*. [3][2]

If this procedure is repeated until there are enough selected individuals, these selection methods are called fitness proportionate **selection** or ***roulette-wheel selection***. If instead of single pointer spun multiple times, there are multiple, equally spaced tips about a wheel that is spun at one time, it is called stochastic universal sampling. Constantly selecting the best individual of a randomly chosen subset is tournament selection. Taking the optimum half, third or another proportion of the individuals are truncation selection. [3]

There are other selection algorithms that don't consider all individuals for choice, but only those with a fitness value that is greater than a given (arbitrary) constant. Other algorithms choose from a restricted pool or chromosomes from one generation to the next. It is a biological crossover, upon which genetic algorithms are based. Cross over is a process of taking multiple parent solutions and producing a child solution from them. The most commonly used methods of selecting chromosomes for parents to crossover are : [3][5]

- Roulette wheel selection
- Boltzmann selection
- Rank selection
- Steady state selection
- Tournament selection

where only a certain percentage of the individuals are allowed, supported fitness value. [3]

B. Cross over

In Genetic Algorithms, Crossover is a genetic operator used to vary the programming of a chromosome

C. 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's analogous to biological mutation. Mutation change one or more gene values in a chromosome from its initial state. In mutation, the solution can change entirely from the previous solution. Hence GA can come to better solution than previous solution by using mutation. Mutation occurs through evolution according to a user-definable mutation probability. This probability should be set low. If it's set too high, the search will turn into a primitive random search.[3][2]

The classic example of a mutation operator involves likelihood that an arbitrary bit in a genetic sequence will be changed from its original state. A standard method of implementing the mutation operator involves generating a random variable for each bit in a sequence. This random variable says whether or not a particular bit will be modified. This mutation procedure, supported the biological point mutation, is referred as single point mutation. Any types are inversion and floating point mutation. When the genes encoding is unconfirmed as in transformation problems, mutations are interchange, inversions and scrambles.

The goal of mutation in GAs is preserving and introducing diversity. Mutation may allow the algorithm to avoid local minima by preventing the population of chromosomes from changing into too the same as one another, so slowing or even stopping evolution. This reasoning is also explain the fact that most GA systems avoid only taking the fittest of the population in generating the next but rather a random (or semi-random) choice selection with a weighting toward those that are fitter.[2][3][5]

❖ Flow Chart for Genetic Programming.

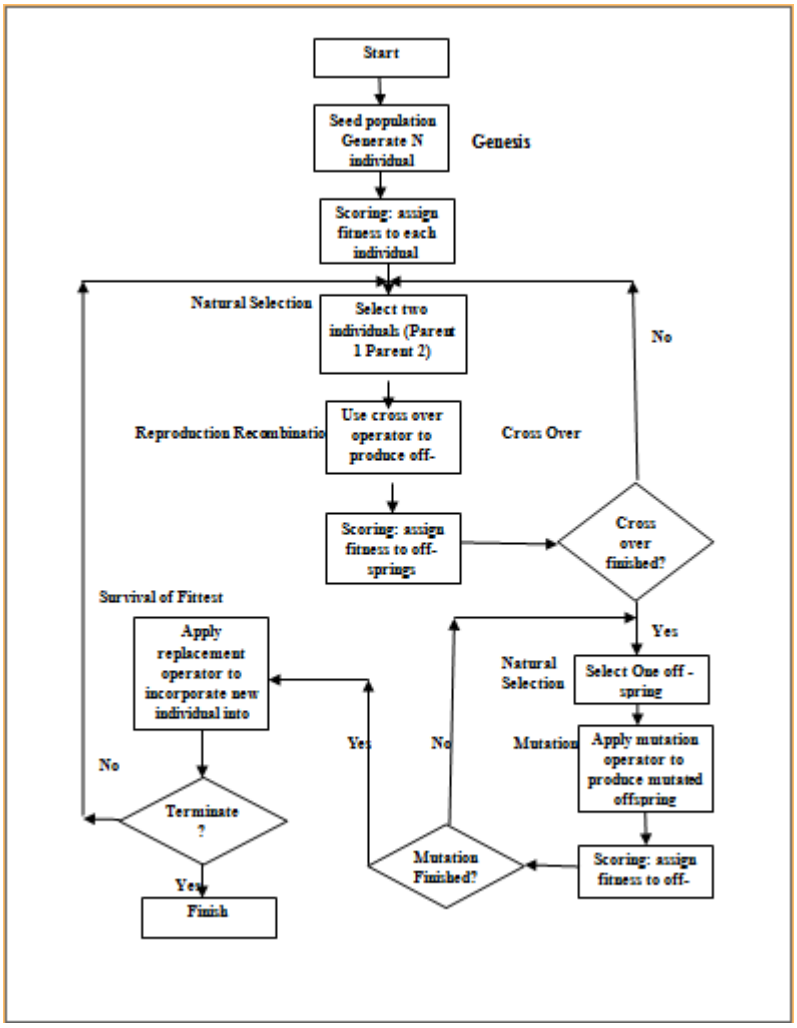


Figure 1: Genetic Algorithm – Program flow chart

Example:

Suppose we want to maximize the value of the function $f(x) = x^2$ on the interval [0, 31]. Our representation should be the set of all possible strings of length five that contain any permutation of 1 and 0. Changing these binary digits into decimal form represent that this representation is perfectly suited to our problem. For example, string 00000 represent to 0, the lowest value in our input parameter set, while the string 11111 represent to $1(2^4) + 1(2^3) + 1(2^2) + 1(2^1) + 1(2^0) = 31$, the largest value in our input set. [4]

Now we select at random some arbitrary number of the possible strings; these selected strings will become our initial population. In our example, we have chosen the following initial population:

01101
11000
01000
10011

Table 1 show the fitness value of string which we select above. [4]

String Number	String	Fitness	Percentage of Total
1	01101	13	20.3%
2	11000	24	37.5%
3	01000	8	12.5%
4	10011	19	29.7%
Total		64	100%
Maximum		24	

Table 1. [4]

After applying the Crossover operator the new population is generated which is better than previous population show in table 2.[4]

Parent (Before Crossover)	Crossover Site	New Population	Fitness	Percentage of Total
0 1 1 0 | 1	4	01100	12	15%
1 1 0 0 | 0	4	11001	25	31.25%
1 1 | 0 0 0	2	11011	27	33.75%
1 0 | 0 1 1	2	10000	16	20%
Total			80	
Maximum			27	

Table 2 [4]

Advantages & Disadvantages of GA

Advantages

- It will be solve every optimisation problem which can be described with the chromosome encoding.
- It solves issues with multiple solutions.
- As the genetic algorithm execution technique isn't dependent on the error surface, we can solve multi-dimensional, non-differential, non-continuous, and even non-parametrical problems.
- Structural genetic algorithm offers us the possibility to solve the solution structure and solution parameter problems at the same time by means of genetic algorithm.
- Genetic algorithm is a method which is very simple to understand and it practically doesn't demand the knowledge of mathematics.
- Genetic algorithms are simply transferred to existing simulations and models. [7]

Disadvantages

- Certain optimisation problems (they are called variant problems) can't be solved by means of GAs. This happens due to poorly known fitness functions which generate bad chromosome blocks in spite of the fact that only good chromosome blocks cross-over.
- There isn't absolute assurance that a genetic algorithm can find a global optimum. It happens fairly when the populations have lots of subjects.
- Like other artificial intelligence techniques, the genetic algorithm can't assure constant optimisation response times. Even more, the distinction between the shortest and therefore the longest optimisation response time period is much larger than with conventional gradient methods. This unlucky genetic algorithm property limits the genetic algorithms' use in real time applications.
- Genetic algorithm applications in controls which are performed in real time are limited because of random solutions and convergence. Therefore, it's unreasonable to use genetic algorithms for on line controls in real systems without testing them first on a simulation model. [7]

III. APPLICATIONS OF GA

Genetic Algorithm (GA) have been used for problem-solving and for modelling. GAs are applied to several scientific, engineering problems, in business and entertainment, include: [8]

- **Optimization**: GAs utilized in a wide variety of optimization tasks, including numerical optimization, and combinatorial optimization problems such as travelling salesman problem (TSP), circuit design, job shop scheduling and video & sound quality optimization [8].
- **Automatic Programming**: GAs has been used to evolve computer programs for particular tasks, and to design other computational structures, for example, cellular automata and sorting networks [8].
- **Machine and robot learning**: GAs have been used for many machine- learning applications, like classification and prediction, and protein structure prediction. GAs has also been used to
- design neural networks, to go forward rules for learning classifier systems or symbolic production systems, and to design and control robots [8].
- **Economic models**: GAs has been used to model processes of innovation, the development of bidding strategies, and the emergence of economic markets [8].
- **Immune system models**: GAs are accustomed model various aspects of the natural immune system, including somatic mutation during an individuals lifetime and the discovery of multi-gene families during evolutionary time [8].
- **Ecological models**: GAs are accustomed model ecological phenomena such as biological arms races, host-parasite co-evolutions, symbiosis and resource flow in ecologies [8].
- **Models of social systems**: GAs is accustomed study evolutionary aspects of social systems, such as the evolution of cooperation, the evolution of communication, and trail-following behaviour in ants [8].

IV. CONCLUSION

Genetic Algorithms are straightforward to apply in wide range of problems, from optimization issues to inductive idea learning, scheduling, and layout problems. The results will be very good on some problems, and rather reduced on others. If only mutation is used, then performance of algorithm is very slow. Crossover makes it significantly faster. GAs are very helpful when the developer doesn't have precise domain expertise, because GAs possess the flexibility to explore and learn from their domain.

REFERENCES

[1]. "Genetic Algorithams: A Survey" , M.Srinivas,Motorola India Electronics Ltd.and Lalit M.Patnaik, Indian Institute of Science.
[2]. "Genetic Algorithms & Modeling" soft computing Course Lecture 37-40,notes,slide www.myreaders.info/,RCChakrabotry Aug.10,2010
[3]. http://en.wikipedia.org/wiki/Genetic_operator
[4]. http://legacy.earlham.edu/~chrisma/survey.htm
[5]. http://www.obitko.com/tutorials/genetic-algorithms/index.php
[6]. "Neural Network, Fuzzy Logic, and Genetic Algorithms - Synthesis and Applications", with G.A. Vijayalaksmi Pai and S. Rajasekaran (2005)
[7]. http://www.ro.feri.unimb.si/predmeti/int_reg/Predavanja/Eng/3. Genetic%20algorithm/_18.html
[8]. http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/tcw2/article1.html

Authors Profile

Krishna Lad received the **Diploma** in Computer engineering from the Govt. Poly. For Girls College, Surat, Technical Education Board, Ghandhinagar, Gujarat, India, in 2009. Received the **B.E** degree in Computer Engineering from the Sarvajanik Collage of Engineering & Technology (S.C.E.T), Surat, Gujarat Technological University, Ahmedabad, Gujarat, India, in 2012. Pursuing **M.Tech.** in Information Technology in Charotar University of Science & Technology, Changa, Anand, Gujarat, India. Her research interest includes Data Mining, Soft Computing, Networking, Software Engineering.

Meenakshi Agrawal Received the **B.E** degree in Information Technology from the Scope Collage of Engineering, Bhopal, Rajiv Gandhi Prodyogiki Vishvavidyalya (RGPV), , Madhya Pradesh, India. Pursuing **M.Tech.** In Information Technology in Charotar University of Science & Technology, Changa, Anand, Gujarat, India. Her research interest includes Networking, Neural Networks and fuzzy logic, Software Engineering.

Mr. Mrudang Pandya received the **B.E.** degree in Information Technology from sankalchand Patel Collage of Engineering (SPCE), North Gujarat Visnagar, India, in 2010. **M.Tech degree** in Information Technology from U.V Patel Collage of Engineering (UVPC) Ganpat University Kherwa, Mehsana, India in 2012. Presently working as a Assistant Professor, CHARUSAT, Changa, Gujrat, India. His research interest includes Soft Computing, Software Engineering, Enterprise Resource Planning, Image Processing, Data mining.