The power of Genetic Algorithm

Malay Kumar Pakhira*

1. Introduction

After a long time since the development of computing devices and computing techniques, computer scientists turned their attention to borrow ideas from nature and naturally occurring events in order to attain the supreme power of nature, at least partially. It is observed that rules of nature are unanimously applicable to all nature directed events. These rules are so firm that any departure from them may cause great chaos. The algorithms that nature follows always try to converge to the most stable states only. When in any natural system, this stability is disrupted artificially, the nature never fails to bring back the earlier stable conditions. Nature always tries to maintain a perfect balance among its creatures so that the biological ecosystem is properly maintained. Nature always wants its creatures to be the fittest for the environment. It passes them through the algorithms of adaptation and evolution. Nature is always kind to its creatures and it always applies its supreme power for their betterment. Since human beings are the most matured creatures of nature, they gained power to learn the rules of nature and apply those rules to their lives. Most of the secrets of nature are no longer that much secret to the human being as they were thousands of years ago. Man gained scientific knowledge from natural events and applied it for development. The idea of genetic survival of the fittest has been simulated in the extremely powerful optimization technique called the *Genetic Algorithm.* In the last half century period, almost all efforts in developing problem solving systems are oriented towards the use of any of the above mentioned artificially simulated natural events. In this article we shall look first at the history of the development of Genetic Algorithm and then learn how to write an evolution program and how it works.

2. Genetic Algorithm: Brief history of development

Genetic Algorithms (GAs) are randomized search methods that work upon the principle of biological evolution processes discovered by Charles Darwin in 1859. The famous scientist observed the natural evolution processes in animals of the world and developed his theory on *Survival of the Fittest* through generations. The idea that a child inherits the parental characteristics, is discovered by another famous scientist Gregor Johann Mendel, known as the Father of Genetics, in 1865. He devised the checker-board logic to explain exchange of genetic substances lying in chromosomes and proved it experimentally. Mendelian laws became known to the scientists only after they were rediscovered in 1900 by Heugo De Vries. T. Morgan and his collaborators completed the theory of genetics. In 1920, it was proved that Mendel's law of genetics and Darwin's theory of natural selection are supplementary to each other and both of them occur naturally for every sexually reproducing organisms. The genetic algorithm that we use today is developed by mixing the phenomena of genetic exchange and biological evolution. The algorithm has been proposed by John Holland in 1975 in his book on *"Adaptation in Neural and Artificial Systems".*

3. Basics of Genetic Algorithm

GAs perform guided random search in large multimodal search space in order to provide the optimal or a near optimal solution (the limitation that only a near optimal solution is guaranteed is due to the fact that this is a probabilistic algorithm). The value of the objective function is defined in terms of the parameters defining the problem space. Unlike the conventional search methods, GAs deal with multiple representative solutions simultaneously. GAs are found to provide near optimal solutions in various fields of applications e.g., pattern recognition, image processing, machine learning, VLSI design etc.

To use GA to solve a problem, initially a set of representative solutions are generated randomly or by using domain specific knowledge. This set of solutions

* *Lecturer, Department of Computer Science and Technology*

is called the initial population. Each member of this population is called a chromosome which is coded as a binary string of finite length. The binary string is generally a concatenation of the binary encoded parameters of the objective function. A collection of chromosomes or solution strings is called a population. We start with an initial population of size P. In each iteration of the algorithm, a new population of the same size is generated from the current population using two basic operations. These two operations are selection and reproduction. Reproduction is a combination of the genetic operators i.e., cross-over and mutation.

In the elitist model of the GA, adopted in this article, the best string obtained in the current iteration is copied into the population for the next iteration.

String Representation :

In an implementation, we can use a chromosome structure that will represent a particular solution of the problem. Each string or chromosome will be of length n, where n is the number of variables whose optimal values will determine the optimal value for the objective function. For example, S ={ x1, x2,x3,x4,x5,x6,x7,x8} is a string of eight variables.

Initial Population :

A set of P solutions is generated randomly and taken as the initial population. The length of each chromosome is fixed and is equal to the number of variables. In this generation process care is taken so that values of all the variables must lie within the specific ranges i.e., any variable xi should be selected in such a way so that $x_i^{min} \leq xi \; x_i^{max}$. If in the random generation process any string does not satisfy the above condition, it is rejected and a new string is generated. Information regarding the values of x_i^{min} and x_i^{max} are obtained from the problem specification.

Fitness evaluation :

The fitness of a chromosome is computed as the value of a predefined fitness function. In general every optimization problem uses the objective function as the fitness function. Let A and B be two strings and f^A and f^B are values of the objective function f. If $f^A > f^B$, then the former string or chromosome is a fitter candidate than the latter.

Selection :

The selection operation mimics the survival of the fittest concept of natural genetic systems. The chromosomes that possess better fitness value will have higher probability of appearing in the population for the next generation. For selection, we used the roulette wheel strategy which ensures that the probability of selection of the fittest candidate will be the maximum.

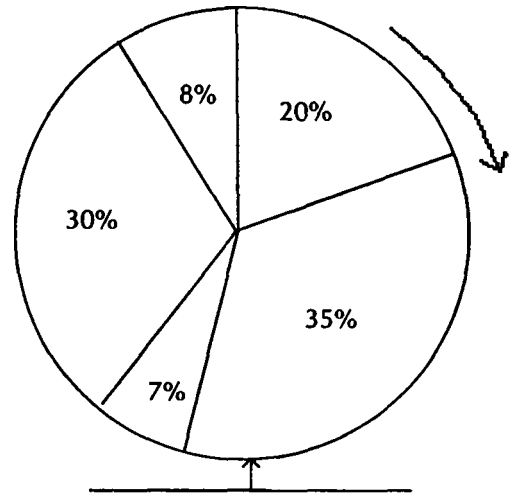

Figure 1. A roulette wheel

Consider the above figure. If we rotate the wheel, the probability that the wheel comes to a stable state with the arrowhead pointing to any marked region on the wheel is determined by the percentage of area covered by that region. Similarly, in the selection scheme if the sum total of fitness of all the strings be represented by the total area of the wheel, where the marked areas represent the fitness of individual strings, then the probability of their selection depends on their fitness values.

Crossover :

Crossover exchanges information between two parent strings and generates two offsprings for the next population. A pair of chromosomes is selected randomly from the parent population pool and crossover is performed. We have selected the single point cross-over, where the gene position at which a chromosome may be fragmented is random. This will provide the maximum possible cover in the search space. Let the

two parent chromosomes be $P_i = \{x_{i1}\ x_{i2}\ x_{i3}\ x_{i4}\ x_{i5}\ x_{i6}\ x_{i7}\ x_{i8}\}$ and $P_j = \{x_{j1}\ x_{j2}\ x_{j3}\ x_{j4}\ x_{j5}\ x_{j6}\ x_{j7}\ x_{j8}\}$. Now if the randomly generated crossover position be 5 then after crossover P_i and P_j will be replaced in the new pool by: $C_i = \{x_{i1}\ x_{i2}\ x_{i3}\ x_{i4}\ x_{i5}\ x_{j6}\ x_{j7}\ x_{j8}\}$ and $C_j = \{x_{j1}, x_{j2}, x_{j3}, x_{j4}, x_{j5}, x_{i6}, x_{i7}, x_{i8}\}$. There may arise one problem in the crossover process. The child may have fewer groups than the parents. If this occurs then we should reject the child by any other member of the child pool already generated or by a fresh string generated using the initial population generation technique. One can also attempt the crossover process for a number of times until a suitable child is generated or may select any of the parents to be copied in the child pool.

Example of crossover:

String A : 10001|00110110101000111

String B : 11111|1111000011111101

If crossover point is the fifth gene from the left, then after crossover we get two offsprings as:

String A': 10001|11110000111111101

String B': 11111|00110110101000111

Mutation :

By mutation, we can alter the value at any arbitrary gene position. Every gene position will have the equal probability of undergoing this kind of changes. A new string is generated from the old one by the mutation process. Since invalid strings can be generated in this process too, we should keep a step for checking as is done in the crossover phase.

The mutation process introduces some extra variability into the population. Even if it is performed with a very low probability, it has an important role in the generation process. Any accidental loss of information due to crossover may be recovered by this process.

Example of mutation:

String A: 10001001110110101000111

If mutation point is the sixth gene from the left then after mutation we get the new string as:

String A': 10001101101101010001111

Elitist strategy: Using this strategy we can preserve the best information obtained thus far in the generation process. This is implemented in the following manner. We copy the best string in the current pool to the worst string in the child pool. In this way we can avoid any loss of information. This also ensures a faster convergence to the global solution.

4. Basic parameters and operators of Genetic Algorithm

For every GA procedure there are a number of parameters. Some of the parameters are *objective parameters,* which in the coded form represent one of the infinitely different solutions. The target is to find out the best possible representative solution maintained by a finite sized population. The value of the objective function for a particular solution is determined by the corresponding parameter values.

Other parameters are called *strategic parameters,* which are related to the degree and rate of convergence of the algorithm itself. These parameters include population size, tournament size, crossover probability, mutation probability, maximum generations etc.

In GA we maintain a fixed population size. Each member of the population is called a chromosome. Normally, initial population is generated randomly and in some cases domain specific knowledge is applied to generate them. The chromosome is generally a concatenation of the encoded parameters values. Encoding is done in binary for basic GA. However, other symbolic or numeric values are possible for EP (Evolution Programs).

The selection operator is responsible for the selection of the fittest candidates(chromosomes) of the current generation (g) in the population to be represented in the next generation (g+1). Obviously the selection depends on their figure of merit over the fitness of the chromosome to optimize the objective function. The selection operator selects a chromosome from the current population by using a probabilistic function p(.). The probability of selection of the i^{th} chromosome is

$$p(X_i) = F(X_i) / F(X_{\ i})$$

Where, $F(X_i)$ = fitness of the i^{th} chromosome. This kind of selection scheme is called *proportionate selection or the roulette wheel* selection strategy.

By crossover, features of two selected mates from the parent population are intermixed and this enriches the next generation by generating newer chromosome patterns called child. If chromosomes are of fixed length L, then we can select the crossover point to be the r^{th} position in the gene patterns, where $1 \leq r \leq L$, which is generally randomly selected. This type of cross-over is called single point cross-over. This is done with a certain cross-over probability, P_c. The value of P_c is selected from the range (0-1). Note that L = n x gene-size. Gene-size is the number of bits to represent a gene position in the chromosome.

Mutation is another operator in GA which is having no less significance than the crossover and selection operators. It is needed as a fine tuning operator. It makes minor changes to a chromosome so that any chance of not reaching the global solution is removed. Generally the mutation probability P_m is kept very low, in the range of (0-0.5). The mutation operator may operate in different ways. In the binary representation for chromosomes, it generally toggles one or more bit position(s). For other representations, it works differently but its role is to make minor changes to the chromosome patterns. In case of real representation of chromosomes, generally Gaussian distribution function is used to update a gene. In GA, the mutation operator acts as a background operator and is typically used to recover lost patterns.

Thus using the above three operators, we can get a new population P(g+l) from the current population P(g). This process is repeated till some stopping criteria is satisfied. The basic GA is listed in Fig 2.

5. The Algorithm

```
begin
   g =0
   initialize P(g)
   evaluate P(g)
   termination_condition = false
   while termination_condition = false do
      begin
       g =g+l
       selects parents from P(g)
       crossover
       mutation
       evaluate P(g+l)
      end
end
```

Figure 2. The basic genetic algorithm

6. Powers and weaknesses of Genetic Algorithm

The major strength of GA is its wide applicability. Scientists and researchers in various fields of science and technology are using it without demur. It is found that the softwares based on GA perform much better than other commercially available softwares. Another strong point is its inherent parallelism. It can process a number of representative solutions at the same time and this is the secret that starting with a number of completely random configurations it can reach a very stable solution within a reasonable time. Goldberg said in his book on *Genetic Algorithms in Search, Optimization and Machine Learning* that:

"In a world where serial algorithms are usually made parallel through countless tricks and contortions, it is no small irony that genetic algorithms (highly parallel algorithms) are made serial through equally unnatural tricks and turns."

GAs are perceived as weak methods because of its lack of general applicability to all kinds of problems. However, in presence of nontrivial constraints they change rapidly into strong methods. We must go far beyond the present scenario to frame the GA as a great general problem solver. Continuous efforts of scientists and researchers in this direction are revealing newer and newer paths to reach this goal.

I can only show you the door, Neo.
You have to walk through it.
— Morpheus to Neo, The Matrix