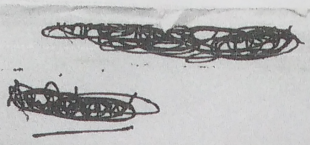


Chapter 2. GENETIC ALGORITHMS

2.1 Introductions and Overview

Genetic algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. Inspired by Darwin's theory of evolution John Holland proposed Genetic Algorithms (GAs) in 1970 as computer programs that mimic natural genetics. Genetic algorithms create an environment where populations of data can compete and only the fittest survive. GAs were invented by John Holland and developed by him and his students and colleagues. Holland's book "Adaptation in Natural and Artificial Systems" was published in 1975.

Genetic Algorithms are search methods that can be used for both solving problems and modeling evolutionary systems. Since it is heuristic (it estimates a solution) one won't know if the solution is exact. Most real-life problems are like that: one estimates a solution, don't calculate it exactly. Most problems don't have any formula for solving the problem because it is either too complex or it takes too long to calculate the solution exactly. A heuristic method is a more feasible approach. GAs differ from other heuristic methods in several ways. One important difference is that it works on a population of possible solutions (actually coded representation of solution), where other heuristic methods use a single solution in their iterations. Another difference is that GAs are probabilistic and not deterministic. There are two types of genetic algorithms:

1. Simple genetic algorithms (SGAs).
2. Elitist genetic algorithms (EGAs).

2.2.1 Simple Genetic Algorithms (SGAs)

Genetic algorithms are adaptive and robust computational procedures modeled on the mechanics of natural genetic systems. GAs act as biological metaphor and try to emulate some of the processes observed in natural evolution. They are viewed as randomized, yet structured, search and optimization (or search) techniques. GAs iteratively perform the following cycle of operations on coded representations of solutions (not directly on the solutions themselves), called population, until some termination condition is achieved: selection (including evaluation of each solution), reproduction (including crossover and mutation) and reduction/replacement of the old population with a new one.

To find out the optimal solution of a problem, a GA starts from a set of assumed solutions (chromosomes) and evolves different but better sets (of solutions) over a sequence of iterations. In each generation (iteration) the objective function (fitness measuring criterion) determines how good each solution is and based on these values, some of them are selected for reproduction. The number of copies reproduced by an individual parent is expected to be directly proportional to its fitness value, thereby embodying the natural selection procedure to some extent. The procedure thus selects better (highly fitted)

chromosomes and worse ones are eliminated. Hence the performances of GAs depend on the fitness evolution criterion to a large extent. Genetic operators are applied on these (selected) parent chromosomes and new chromosomes (offspring) are generated. The procedure terminates after some prefixed iterations (generations) or after all the strings in the population have attained a certain degree of homogeneity (a large number of strings have identical bits at most position). Simple genetic algorithms (SGAs) consider only the fitness value of the chromosome under consideration for measuring suitability for selection for the next generation, i.e., the fitness of a chromosome is a function of the functional value of the objective function. Fitness of a chromosome $x = g[f(x)]$, where $f(x)$ is the objective function and g is the another function which is by operating on $f(x)$ gives the fitness value [5].

2.2.2 Elitist Genetic Algorithms (EGAs).

In simple genetic algorithms (SGAs), we do not preserve the best possible solution obtained so far, thereby increasing the chance of losing the obtainable best possible solution. Elitist strategy overcomes this problem by occupying the best member of each generation into the next one. Though this strategy may increase the speed of dominance of a population by a potential string (string with high fitness value), it enhances the performance of a GA using generational replacement.

2.3 Biological Background

a) Chromosomes: All living organisms consist of cells. In each cell there is the same set of chromosomes. Chromosomes are strings of DNA and serve as a model for the whole organism. Chromosomes consist of genes, blocks of DNA. A complete set of genetic material (all chromosomes) is called genome. A particular set of genes in genome is called the genotype.

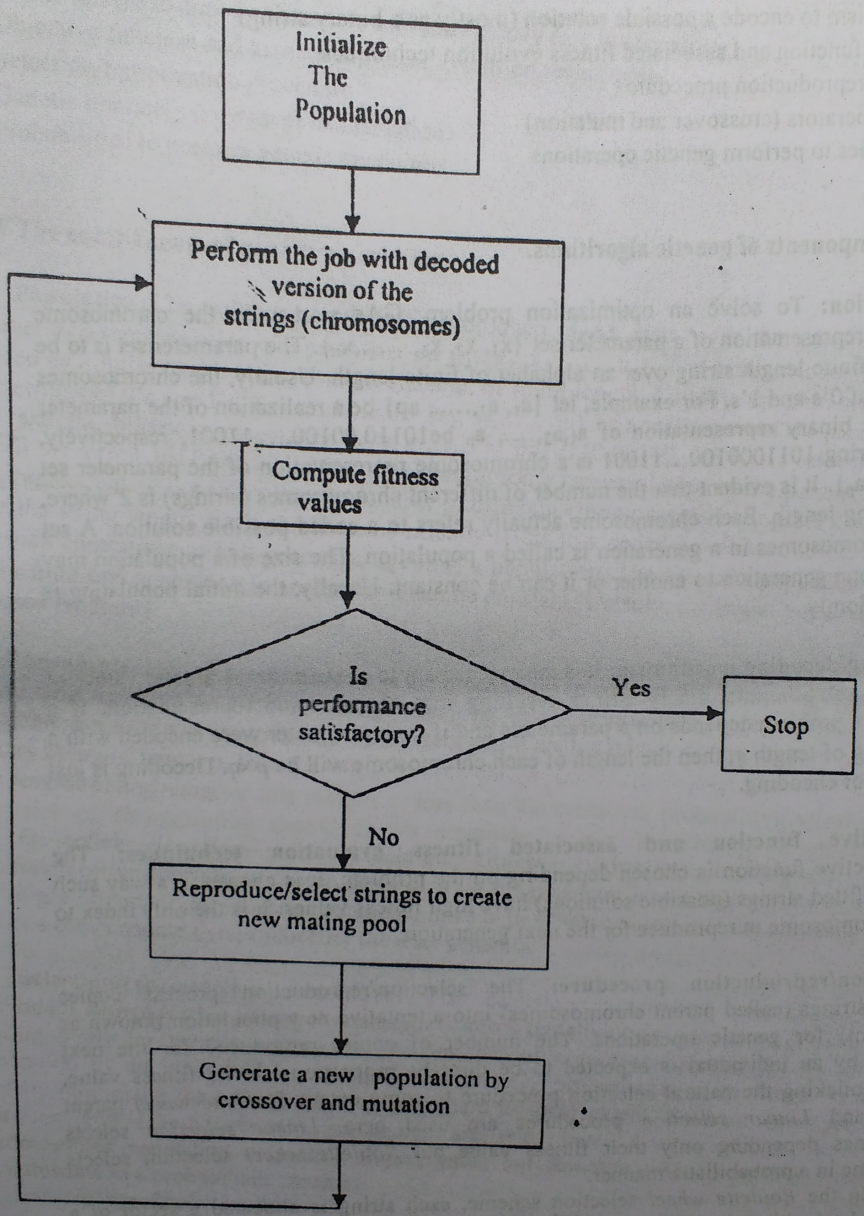
b) Reproduction: During reproduction, the first thing that occurs is recombination (or crossover). Genes from the parents combine in some way to create a whole new chromosome. The newly created offspring can then be mutated. Mutation means that the elements of DNA are a bit changed. These changes are mainly caused by errors in copying genes from parents.

c) Fitness: The fitness of an organism is measured by the success of the organism in its life.

d) Evolution: Evolution is a process that each and every species in nature is subjected to. In evolution, each species faces the problems of searching for a beneficial adaptation to adapt to the rapidly changing environment around them.

2.4 The Basic Structure of A Genetic Algorithm

FIGURE: 2.4.1 Basic Steps Of GA



A GA typically consists of the following components

- A population of strings or coded possible solutions (biologically referred to as chromosomes)
- A mechanism to encode a possible solution (mostly as a binary string)
- Objective function and associated fitness evolution techniques
- Selection/reproduction procedure
- Genetic operators (crossover and mutation)
- Probabilities to perform genetic operations

2.5 The components of genetic algorithms.

a) Population: To solve an optimization problem, GAs start with the chromosome (structural) representation of a parameter set $\{x_1, x_2, x_3, \ldots\ldots, x_n\}$. The parameter set is to be coded as a finite length string over an alphabet of finite length. Usually, the chromosomes are strings of 0's and 1's. For example, let $\{a_1, a_2, \ldots\ldots, ap\}$ be a realization of the parameter set and the binary representation of $a_1, a_2, \ldots, a_p$ be10110,00100,...,11001, respectively. Then the string 1011000100...11001 is a chromosome representation of the parameter set $\{a_1, a_2, \ldots, a_p\}$. It is evident that the number of different chromosomes (strings) is 2^l where, l is the string length. Each chromosome actually refers to a coded possible solution. A set of such chromosomes in a generation is called a population. The size of a population may vary from one generation to another or it can be constant. Usually, the initial population is chosen randomly.

b) Encoding/decoding mechanism: It is the mechanism to convert the parameter values of a possible solution into binary strings resulting into chromosome representation. If a solution of a problem depends on p parameters and if each parameter were encoded with a binary string of length q, then the length of each chromosome will be $p \times q$. Decoding is just the reverse of encoding.

c) Objective function and associated fitness evaluation techniques: The fitness/objective function is chosen depending on the problem. It is chosen in a way such that highly fitted strings (possible solutions) have high fitness values. It is the only index to select a chromosome to reproduce for the next generation.

d) Selection/reproduction procedure: The selection/reproduction process copies individual strings (called parent chromosomes) into a tentative new population (known as mating pool) for genetic operations. The number of copies reproduced for the next generation by an individual is expected to be directly proportional to its fitness value, thereby mimicking the natural selection procedure to some extent. *Roulette wheel* parent selection and *Linear selection* procedures are used here. *Linear selection* selects chromosomes depending only their fitness value but *Roulette wheel* selection selects chromosome in a probabilistic manner.

In the *Roulette wheel* selection scheme, each string is allocated a sector of a *Roulette wheel* with the angle subtended by the sector at the center of the wheel equal to

$2\pi f_i / f$, where f_i is the fitness value of the i^{th} string and f is the average fitness value of the population.

In *Linear normalization* selection procedure (which works better in a close competitive environment) the difference between successive fitness values has taken as 1, and the minimum fitness value has kept to 1. The number of copies produced by the i^{th} individual (chromosome) with normalized fitness value f_i in a population of size n is taken as round (c_i) where

$$c_i = \frac{n \times f_i}{\sum_{i=1}^{n} f_i}$$

and round(c_i) gives the nearest integer of the real number. If round(c_i) ≤ 1, then the number of copies reproduced is taken as 1, provided the size of the population does not exceed.

2.6 Genetic Operators

Genetic operators are applied on parent chromosomes and new chromosomes (called offspring) are generated. Frequently used genetic operators are described below [5].

a) Crossover: The main purpose of crossover is to exchange information between randomly selected parent chromosomes with the aim of not losing any important information (minimum disruption of the structures that are selected for genetic operation). Actually, it recombines genetic material of two parent chromosomes to produce offspring for the next generation. The crossover can be viewed as a three-step process. In the first step, pairs of chromosomes (called mating pairs) are chosen from the mating pool. The second step determines, on the basis of crossover probability (by generating a random number in [0,1]
And checking whether this number is less than the crossover probability), whether these pairs of chromosomes should go for crossover or not. Interchange of chromosome segments between mating pairs is done in the third step. The number of segments and the length of each segment to be exchanged depend on the type of crossover technique. Some of the commonly used crossover techniques are *one-point* crossover, *two-point* crossover, *multiple-point* crossover, *shuffle-exchange* crossover, *uniform* crossover etc.
To illustrate how the segments of two parent chromosomes are swapped, the one-point crossover is shown below:
Here, a position k is selected uniformly at random between 1 and l-1, where l is the string length (greater than 1). Swapping all character from (k+l) creates two new strings. If

a = 11000 10101 01000...01111 10001,
b = 10001 01110 11101... 00110 10100

be two strings (parents) selected for the crossover operation and the generated random number 11 (eleven). Then the newly produced offspring (swapping all characters after the position 11) will be

a' = 11000 10101 01101...00110 10100

b' = 10001 01110 11000...01111 10001

The two-point crossover is similar to that of one-point crossover except that we must select two positions and only the bits between these two positions are swapped. This crossover method can preserve the first and last parts of a chromosome.

b) **Mutation:** The main aim of mutation is to introduce genetic diversity into the population. Sometimes, it helps to regain the information lost in earlier generations. Like natural genetic systems, mutation in GAs is also made occasionally. A random position of a random string is selected and is replaced by another character from the alphabet. In the case of binary representation it negates the bit value and is known as bit mutation. For example, the third string of a, shown above, be selected for mutation. Then the transformed string after mutation will be

11100 10101 01000...01111 10001

Normally mutation rate is kept fixed. To sustain diversity (which may be lost due to crossover and very low mutation rate) into the population. Mutation is not always worth performing. High mutation rate can lead the genetic search to a random one. It may change the value of an important bit and thereby affect the fast convergence to a good solution. Moreover it may slow down the process of convergence at the final stage of GAs.

2.7 Probabilities to perform genetic operations:

The probability to perform crossover operation is chosen in a way so that recombination of potential strings (highly fitted chromosomes) increases without any disruption. Generally, the crossover probability lies between 0.6 to 0.9.

Since mutation occurs occasionally, it is clear that the probability of performing mutation operation will be very low. Typically the value lies between 0.001 to 0.01.

Crossover and mutation probabilities may be kept fixed throughout the operation of a GA or they may also be made adaptive (determined automatically depending on the environment) to improve the performance, if possible.

2.8 Distinguishing characteristics:

The following features facilitate GAs to enhance their applicability over most of the commonly used optimization and searching strategies:

- GAs work simultaneously on multiple points and not on a single point, which helps to introduce a large amount of implicit parallelism in the computational procedure. This feature also prevents a GA from getting stuck at local optimal, to some extent.
- GAs work with the coded parameter set not with the parameters themselves. Hence, varying the coding length of the parameters can control the resolution of the possible search space.
- In GAs, the search space need not be continuous (unlike a calculus-based search method).
- GAs are blind. They use only the objective function information and do not need any auxiliary information, like derivative of the optimizing function.
- GAs use probabilistic transition rules, not deterministic ones.

2.9 Advantages and Disadvantages

A GAs has a number of advantages. It can quickly scan a vast solution set. Bad proposals do not affect the end solution negatively as they are simply discarded. The inductive nature of the GAs means that it doesn't have to know any rules of the problem - it works by its own internal rules (operators). This is very useful for complex or loosely defined problems. GAs has drawbacks too, of course. While the great advantage of GAs is the fact that they find a solution through evolution, this is also the biggest disadvantage. Evolution is inductive; in nature life does not evolve towards a good solution - it evolves away from bad circumstances. This can cause a species to evolve into an evolutionary dead end. Likewise, GAs risk finding a sub-optimal solution. Eventually, the GAs will make every solution look identical. This is not ideal. There are two ways around this. The first is to use a very large initial population so that it takes the GAs longer to make all of the solutions the same. The second method is mutation. Mutation is when the GAs randomly changes one of the solutions. Sometimes a mutation can lead to a better solution that a crossover would not have found.

2.10. Applications of Genetic Algorithms

a) Optimization: Optimization is the art of selecting the best alternative among a given set of options. In any optimization problem there is an objective function or objective that depends on a set of variables. To reach an optimum' does not necessarily mean "maximum". It means the best value for the4 function. GAs are excellent for all tasks requiring optimization and are highly effective in any situation where many inputs (variables) interact to produce a large number of possible outputs (solutions).
Some example situations are: Optimization such as data fitting, clustering, trend spotting, path finding, ordering, music generation.

b) Management: Distribution, scheduling, project management, courier routing, container packing, and task assignment, timetables, strategy planning.

c) Financial: Portfolio balancing, budgeting, forecasting, investment analysis and payment scheduling.

d) Engineering: Structural design (e.g. beam sizes), electrical design (e.g. circuit boards), mechanical design (e.g. optimize weight, size & cost), process control, network design (e.g. computer networks).

e) Research & Development: Curve and surface fitting, neural network connection matrices, function optimization, fuzzy logic, population modeling, genetic synthesis, VLSI technology, machine learning, molecular modeling and drug design.