Genetic Algorithms

1.Introduction

Charles Darwin stated the theory of natural evolution in the origin of species. Over several generations, biological organisms evolve based on the principle of natural selection "survival of the fittest" to reach certain remarkable tasks.

In nature, an individual in population competes with each other for virtual resources like food, shelter and so on. Also in the same species, individuals compete to attract mates for reproduction. Due to this selection, poorly performing individuals have less chance to survive, and the most adapted or "fit" individuals produce a relatively large number of offspring's. It can also be noted that during reproduction, a recombination of the good characteristics of each ancestor can produce "best fit" offspring whose fitness is greater than that of a parent. After a few generations, species evolve spontaneously to become more and more adapted to their environment.

In 1975, Holland developed this idea in his book "Adaptation in natural and artificial systems". He described how to apply the principles of natural evolution to optimization problems and built the first Genetic Algorithms. Holland's theory has been further developed and now Genetic Algorithms (GAs) stand up as a powerful tool for solving search and optimization problems. Genetic algorithms are based on the principle of genetics and evolution.

The power of mathematics lies in technology transfer: there exist certain models and methods, which describe many different phenomena and solve wide variety of problems. GAs are an example of mathematical technology transfer: by simulating evolution one can solve optimization problems from a variety of sources. Today, GAs are used to resolve complicated optimization problems, like, timetabling, job shop scheduling, games playing.

2. A Simple Genetic Algorithm

An algorithm is a series of steps for solving a problem. A genetic algorithm is a problem solving method that uses genetics as its

model of problem solving. It’s a search technique to find approximate solutions to optimization and search problems. Basically, an optimization problem looks really simple.
GA handles a population of possible solutions. Each solution is represented through a chromosome, which is just an abstract representation. Coding all the possible solutions into a chromosome is the first part, but certainly not the most straightforward one of a Genetic Algorithm. A set of reproduction operators has to be determined, too. Reproduction operators are applied directly on the chromosomes, and are used to perform mutations and recombination over solutions of the problem. Appropriate representation and reproduction operators are really something determinant, as the behavior of the GA is extremely dependant on it. Frequently, it can be extremely difficult to find a representation, which respects the structure of the search space and reproduction operators, which are coherent and relevant according to the properties of the problems. Selection is supposed to be able to compare each individual in the population. Selection is done by using a fitness function. Each chromosome has an associated value corresponding to the fitness of the solution it represents. The fitness should correspond to an evaluation of how good the candidate solution is. The optimal solution is the one, which maximizes the fitness function. Genetic Algorithms deal with the problems that maximize the fitness function. But, if the problem consists in minimizing a cost function, the adaptation is quite easy. Either the cost function can be transformed into a fitness function, for example by inverting it; or the selection can be adapted in such way that they consider individuals with low evaluation functions as better. Once the reproduction and the fitness function have been properly defined, a Genetic Algorithm is evolved according to the same basic structure. It starts by generating an initial population of chromosomes. This first population must offer a wide diversity of genetic materials. The gene pool should be as large as possible so that any solution of the search space can be engendered. Generally, the initial population is generated randomly. Then, the genetic algorithm loops over an iteration process to make the population evolve.

Each iteration consists of the following steps:

- *SELECTION: The first step consists in selecting individuals for reproduction. This selection is done randomly with a probability depending on the relative fitness of the individuals so that best ones are often chosen for reproduction than poor ones.*
- *REPRODUCTION: In the second step, offspring are bred by the selected individuals. For generating new chromosomes, the algorithm can use both recombination and mutation.*
- *EVALUATION: Then the fitness of the new chromosomes is evaluated.*
- *REPLACEMENT: During the last step, individuals from the old population are killed and replaced by the new ones.*

The algorithm is stopped when the population converges toward the optimal solution.

The basic genetic algorithm is as follows:

- *[start] Genetic random population of n chromosomes (suitable solutions for the problem)*
- *[Fitness] Evaluate the fitness f(x) of each chromosome x in the population*
- *New population] Create a new population by repeating following steps until the New population is complete*

- [selection] select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to get selected).

- [crossover] With a crossover probability, cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.

- [Mutation] With a mutation probability, mutate new offspring at each locus(position in chromosome)

- *[Accepting] Place new offspring in the new population.*

- *[Replace] Use new generated population for a further sum of the algorithm.*
- *[Test] If the end condition is satisfied, stop, and return the best solution in current population.*

- *• [Loop] Go to step2 for fitness evaluation.*

Based on the foregoing discussion, the important criteria for GA approach can be formulated as given below:

- Completeness: Any solution should have its encoding
- Non redundancy: Codes and solutions should correspond one to one
- Soundness: Any code (produced by genetic operators) should have its corresponding solution
-Characteristic perseverance: Offspring should inherit useful characteristics from parents.

3.Comparison of Genetic Algorithm with Other Optimization Techniques

Genetic algorithm differs from conventional optimization techniques in following ways:
1. GAs operate with coded versions of the problem parameters rather than parameters themselves i.e., GA works with the coding of solution set and not with the solution itself.
2. Almost all conventional optimization techniques search from a single point but GAs always operate on a whole population of points(strings) i.e., GA uses population of solutions rather than a single solution fro searching. This plays a major role to the robustness of genetic algorithms. It improves the chance of reaching the global optimum and also helps in avoiding local stationary point.
3. GA uses fitness function for evaluation rather than derivatives. As a result, they can be applied to any kind of continuous or discrete optimization problem. The key point to be performed here is to identify and specify a meaningful decoding function.
4. GAs use probabilistic transition operates while conventional methods for continuous optimization apply deterministic transition operates i.e., GAs does not use deterministic rules.

4.Applications of Genetic Algorithm

A few applications of GA are as follows:
• Nonlinear dynamical systems–predicting, data analysis
• Robot trajectory planning
• Evolving LISP programs (genetic programming)
• Strategy planning
• Finding shape of protein molecules
• TSP and sequence scheduling

• Functions for creating images
• Control–gas pipeline, pole balancing, missile evasion, pursuit
• Design–semiconductor layout, aircraft design, keyboard configuration, communication networks
• Scheduling–manufacturing, facility scheduling, resource allocation
• Machine Learning–Designing neural networks, both architecture and weights ,improving classification algorithms, classifier systems
• Signal Processing–filter design
• Combinatorial Optimization–set covering, traveling salesman (TSP), Sequence scheduling, routing, bin packing, graph coloring and partitioning

Individuals

An individual is a single solution. A chromosome is subdivided into genes. A gene is the GA's representation of a single factor for a control factor. Each factor in the solution set corresponds to gene in the chromosome. Chromosomes are encoded by bit strings are given below in Fig.1 Genes are the basic "instructions" for building a Generic Algorithms. A chromosome is a sequence of genes. Genes may describe a possible solution to a problem.

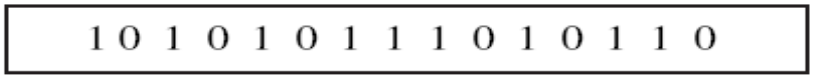

Fig.1 chromosome

Fitness

The fitness of an individual in a genetic algorithm is the value of an objective function . For calculating fitness, the chromosome has to be first decoded and the objective function has to be evaluated. The fitness not only indicates how good the solution is, but also corresponds to how close the chromosome is to the optimal one.

5.1 Populations

A population is a collection of individuals. A population consists of a number of individuals being tested.

The two important aspects of population used in Genetic Algorithms are:

1. The initial population generation.
2. The population size.

Population	Chromosome 1	1 1 1 0 0 0 1 0
	Chromosome 2	0 1 1 1 1 0 1 1
	Chromosome 3	1 0 1 0 1 0 1 0
	Chromosome 4	1 1 0 0 1 1 0 0

Fig.2 Population

5.2.Encoding

Encoding is a process of representing individual genes. The process can be performed using bits, numbers, trees, arrays, lists or any other objects. The encoding depends mainly on solving the problem. For example, one can encode directly real or integer numbers.

5.2.1 Binary Encoding

The most common way of encoding is a binary string, which would be represented as in Fig. 3 Each chromosome encodes a binary (bit) string. Each bit in the string can represent some characteristics of the solution. Every bit string therefore is a solution but not necessarily the best solution. Another possibility is that the whole string.

Chromosome 1	1 1 0 1 0 0 0 1 1 0 1 0
Chromosome 2	0 1 1 1 1 1 1 1 1 1 0 0

Fig.3 binary encoding

5..2.2 *Octal Encoding*

This encoding uses string made up of octal numbers (0–7).

Chromosome 1	03467216
Chromosome 2	15723314

Fig.4 octal encoding

5.2.3 Hexadecimal Encoding

This encoding uses string made up of hexadecimal numbers (0–9, A–F).

Chromosome 1	9CE7
Chromosome 2	3DBA

Fig.5 hexa encoding

5.2.4 Permutation Encoding (Real Number Coding)

permutation encoding, every chromosome is a string of integer/real values, which represents number in a sequence.

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

Fig.6 permutation encoding

5.2.5 Value Encoding

Every chromosome is a string of values and the values can be anything connected to the problem. This encoding produces best results for some special problems. On the other hand, it is often necessary to develop new genetic operator's specific to the

problem. Direct value encoding can be used in problems, where some complicated values, such as real numbers, are used. Use of binary encoding for this type of problems would be very difficult.
In value encoding, every chromosome is a string of some values. Values can be anything connected to problem, form numbers, real numbers or chars to some complicated objects.
Value encoding is very good for some special problems. On the other hand, for this encoding is often necessary to develop some new crossover and mutation specific for the problem.

Chromosome A	1.2324 5.3243 0.4556 2.3293 2.4545
Chromosome B	ABDJEIFJDHDIERJFDLDFLFEGT
Chromosome C	(back), (back), (right), (forward), (left)

Fig.7 value encoding

Breeding

The breeding process is the heart of the genetic algorithm. It is in this process, the search process creates new and hopefully fitter individuals. The breeding cycle consists of three steps:

a. Selecting parents.
b. Crossing the parents to create new individuals (offspring or children).
c. Replacing old individuals in the population with the new ones.