Reaching Optimum Designs Through Processes Inspired by Principles of Evolution

Anu Taneja Amit Kumar

Abstract-Genetic Algorithms have recently become a popular artificial technique for solving complex optimization problems and a sophisticated tool for machine learning. This paper provides an introduction to genetic algorithms and brief applicability to problems. There is a focus on GAs used as an optimisation technique. GAs technique is based on natural evolution which provides a robust solution for a given problem. This paper explains how GAs approaches to optimization for a problem with implementation methods.

Keywords-
Darwin's Theory, EA- Evolutionary Algorithms, EP- Evolutionary Programming, GA- Genetic Algorithms, NP- Non Deterministic Polynomial, Optimization.

1. INTRODUCTION

Nature provides us different things. When we try to think about nature, there are complex mechanisms, operations involved in it. There are many scientific theories proposed on specific part of nature. One among the successful theory is DARWIN'S theory of EVOLUTION. The most important is that it predicted the need for a biological way for passing information between generations. That ultimately led to the discovery of the DNA molecule and within half a century the mapping of the human genome as well as that of other animals. In the other direction the ideas of evolution have given computer scientists ideas for new ways to program - the notion of genetic algorithms. Computer science is about coming up with solutions to problems and that is exactly what nature does over time - adapt animal species via natural selection to allow them to survive better in their changing environments. The idea is that to solve a problem into "digital DNA" and evolve a solution.

Genetic Algorithms are a way of solving problems by mimicking the same processes Mother Nature uses. They use same combination of selection, recombination and mutation to evolve a solution to a problem. Genetic Algorithms are one of the best ways to solve a problem for which little is known. A population is created with a group of individuals created randomly. The individuals in the population are then evaluated. The evaluation function is provided by the programmer and gives the individuals a score based on how well they perform at the given task. Two individuals are then selected based on their fitness value; the higher is the fitness, the higher are the chances of Selection. These individuals then "reproduce" to create one or more offspring, after which the offspring are mutated randomly. This continues until a suitable solution has been found or a certain number of generations have been passed, depending on the needs of programmer.

2. HISTORY

It was in the 1950/60s that several independent researchers were studying the idea that evolution could be used as an optimization tool for engineering problems. The idea behind it all was to evolve solutions to problems by using natural means based on "survival of the fittest". Evolutionary strategies were introduced in the mid 60s by Rechenberg as a method he used to optimize real-valued parameters for hardware devices. Owens, Fogel and Walsh developed evolutionary programming, a technique used where candidate solutions to problems or tasks were represented as finite-state machines which were evolved by randomly mutating their state-transition diagrams and then selecting the fittest. Together with genetic algorithms, these three areas form the backbone of evolutionary computation. Genetic algorithms were invented by Holland in the 60's and developed later in the 70's. This method was defined as a way to move from one population of chromosomes to another by utilizing natural selection and the operator of crossover, mutation, and inversion. Often, the term - genetic algorithm - is used to describe something very different from what was originally defined.

3. DEFINITION OF GENETIC ALGORITHMS

"Genetic Algorithms are nondeterministic stochastic search/optimization methods that utilize the theories of evolution and natural selection to solve a problem within a complex solution space". Genetic algorithms are basically computer-based problem solving systems which use computational models of some of the known mechanisms in "evolution" as key elements in their design and implementation. They are a member of a wider population of algorithm Evolutionary Algorithms (EA). The major classes of EAs are: GENETIC Algorithms, EVOLUTIONARY PROGRAMMING, EVOLUTION Strategies, CLASSIFIER SYSTEM, and GENETIC PROGRAMMING. They all share a common conceptual base of simulating the evolution of individual structures through methods of selection, mutation, and reproduction. The methodologies depend on the performance of the individual structures as defined by an environment. Genetic Algorithms are heuristic, which means it estimates a solution. We won't know if we get the exact solution, but that may be a minor concern. In fact, most real-life problems are like that: we estimate a solution rather than

calculating it exactly. GA's work within a Complex solution space: GAs can be used where optimization is needed. I mean that where there are complex large solutions to the problem but we have to find the best one. Like we can use GAs in findings best moves in chess, mathematical problems, and financial problems and in many more areas.

4. NEED FOR GENETIC ALGORITHM

There are many tasks for which we know fast (polynomial) algorithms. There are also some problems that are not possible to be solved algorithmically. For some problems was proved that they are not solvable in polynomial time. But there are many important tasks, for which it is very difficult to find a solution, but once we have it, it is easy to check the solution. This fact led to NP-complete problems. NP stands for nondeterministic polynomial and it means that it is possible to "guess" the solution (by some nondeterministic algorithm) and then check it, both in polynomial time. If we had a machine that can guess, we would be able to find a solution in some reasonable time. Studying of NP-complete problems is for simplicity restricted to the problems, where the answer may be yes or no. Because there are tasks with complicated outputs, a class of problems called NP-hard problems has been introduced. This class is not as limited as class of NP-complete problems. For NP-problems is characteristic that some simple algorithm to find a solution is obvious at a first sight - just trying all possible solutions. But this algorithm is very slow (usually O (2^n)) and even for a bit bigger instances of the problems it is not usable at all. Today nobody knows if some faster exact algorithm exists. Proving or disproving these remains as a big task for new researchers. Today many people think, that such an algorithm does not exist and so they are looking for some alternative methods - example of these methods are "GENETIC ALGORITHMS".

5. IMPORTANCE OF GA OVER OTHER TECHNIQUES

For most problems in real life don't have any formula for solving the problem because it is too complex, or if you do, it just takes too long to calculate the solution exactly. An example could be space optimization - it is very difficult to find the best way to put objects of varying size into a room so they take as little space as possible. The most feasible approach then is to use a heuristic method. Genetic algorithms are different from other heuristic methods in several ways. The most important difference is that a GA works on a population of possible solutions, while other heuristic methods use a single solution in their iterations. Another difference is that GAs are probabilistic (stochastic), not deterministic.

6. WORKING OF GA

Before understanding the working of GA, let's understand some biological terms related to this.

Chromosome: A set of genes. Chromosome contains the solution in form of genes.
Gene: A part of chromosome. A gene contains a part of solution. It determines the solution. E.g.16743 is a chromosome and 1,6,7,4 and 3 are its genes.
Individual: Same as chromosome.
Population: No of individuals present with same length of chromosome.
Fitness: Fitness is the value assigned to an individual. It is based on how far or close a individual is from the solution. Greater the fitness value better the solution it contains.
Fitness function: Fitness function is a function which assigns fitness value to the individual. It is problem specific.
Breeding: Taking two fit individuals and intermingling the chromosome to create new two individuals.
Mutation: Changing a random gene in an individual.
Selection: Selecting individuals for creating the next generation.

6.1 BASIC DESCRIPTION

Genetic algorithm applies the rules of evolution to the individuals. Each individual in the GA population represents a possible solution to the problem. It selects the fit individuals according to fitness function then combines these individuals into new individuals. Using this method repeatedly, the population will hopefully evolve good solutions.
Specifically, the elements of a GA are:
1. Selection (according to some measure of fitness),
2. Cross-Over (a method of reproduction, "mating" the individuals into new individuals), and
3. Mutation (adding a bit of random noise to the off-spring, changing their "genes").
As we can see here, Darwin's principles have been a major inspiration to GAs. It can be performed through following cycle of stages.
i) Creation of a "population" of strings
ii) Evaluation of each string
iii) Selection of best strings and
iv) Genetic manipulation to create new population of strings.
This flowchart illustrates the basic steps in a GA:

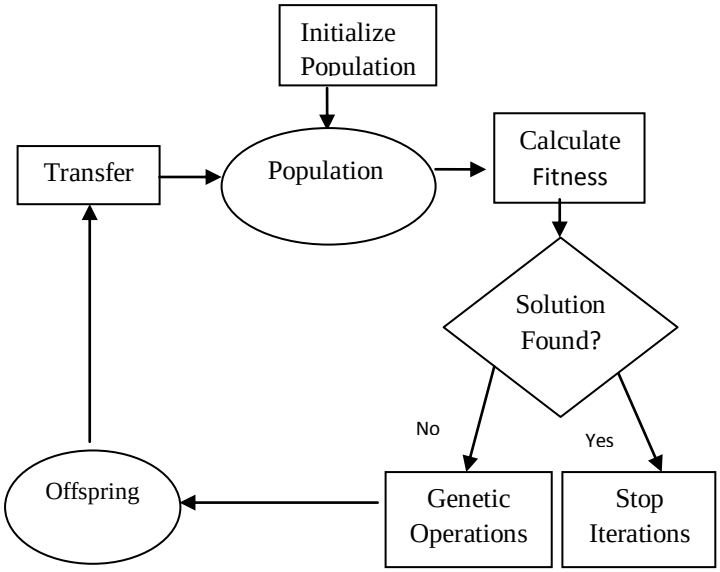

Fig.1 Flowchart of Genetic Algorithms

6.2 GENERAL ALGORITHM OF GA

The algorithm is almost same in most of the applications only fitness functions are different to different problems. The general algorithm is as follows:
START
Generate initial population.
Assign fitness function to all individuals.[Fitness]
DO UNTIL best solution is found
Select individuals from current generation [Elitism]
Create new offspring with mutation and/or breeding [New Population]
Compute new fitness for all individuals
Kill all the unfit individuals to give space to new off springs [Replace]
Check if best solution is found
LOOP
END

7. IMPLEMENTATION

Now let's concentrate on how all the steps are done:
Each cycle in Genetic Algorithms produces a new generation of possible solutions for a given problem. In the first phase, an initial population, describing representatives of the potential solution, is created to initiate the process.
1. The elements of the population are encoded into bit-strings, called chromosomes. Although encoding of chromosomes is done by many ways, binary encoding is most used. In binary encoding every chromosome is a string of bits, 0 or 1.
Chromosome A 10110010110010101011100101
Chromosome B 11111110000010000011111

2. The performance of the strings, often called fitness, is then evaluated with the help of some functions, representing the constraints of the problem. A fitness function is a particular type of objective function that prescribes the optimality of a solution (that is, a chromosome) in a genetic algorithm so that that particular chromosome may be ranked against all the other chromosomes. Depending on the fitness of the chromosomes, they are selected for a subsequent genetic manipulation process.

3. Selection process is mainly responsible for assuring survival of the best-fit individuals. Here individual genomes are chosen from a population for later breeding (recombination or crossover).
A generic selection procedure may be implemented as follows:
The fitness function is evaluated for each individual, providing fitness values, which are then normalized. Normalization means dividing the fitness value of each individual by the sum of all fitness values, so that the sum of all resulting fitness values equals 1.This can be done and represented through "Roulette wheel selection method".
Roulette wheel selection method: It is the likelihood of picking an individual. In this individuals are given a probability of being selected that is directly proportional to their fitness. Two individuals are then chosen randomly based on these probabilities and produce offspring.
Imagine a roulette wheel where all chromosomes are placed in the population, every chromosome has its place big accordingly to its fitness function, it looks like on the following figure.

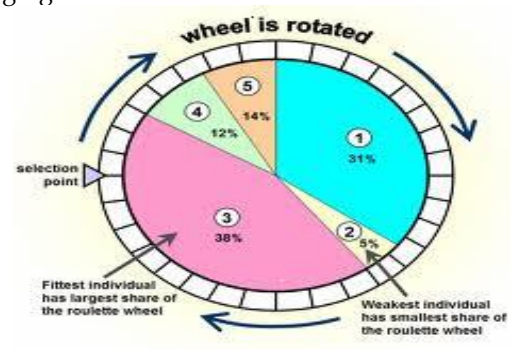

Fig.2 Roulette Wheel Selection Method

The population is sorted by descending fitness values.
• Accumulated normalized fitness values are computed (the accumulated fitness value of an individual is the sum of its own fitness value plus the fitness values of all the previous individuals). The accumulated fitness of the last individual should of course be 1 (otherwise something went wrong in the normalization step!).
• A random number R between 0 and 1 is chosen.
• The selected individual is the first one whose accumulated normalized value is greater than R .This step is repeated until chromosome is found. The selected chromosomes (individuals) are called parents.

4. After selection of the population strings is over, the genetic manipulation process consisting of two steps is carried out. In the first step, the crossover operation that recombines the bits (genes) of each two selected strings (chromosomes) is executed. The second step in the genetic manipulation process is termed mutation, where the bits at one or more randomly selected positions of the chromosomes are altered.

7.1 CROSS-OVER

The cross-over is the method for combining those selected individuals into new individuals. Remember that the individuals are simply strings of values. The crossover splits up the "parent" individuals and recombines them. Here's an example of how two "parents" cross over to make two "children". The simplest way how to do this is to choose

randomly some crossover point and everything before this point copy from a first parent and then everything after a crossover point copy from the second parent.

Chromosome1 11011 | 00100110110 11011 | 11000011110 child1

Chromosome2 11011 | 11000011110 11011 | 00100110110 child2

.

It is illustrated in figure below:

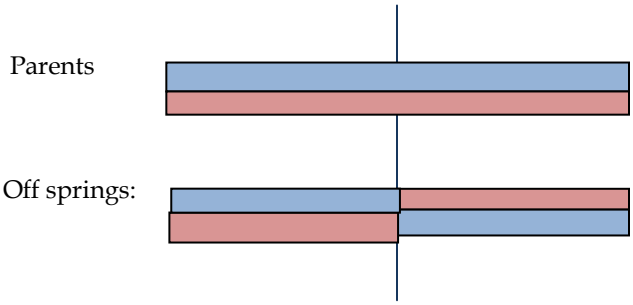

Fig.3 Cross Over

7.2 MUTATION

Mutation is used to maintain genetic diversity from one generation of a population of chromosomes to the next. It is analogous to biological mutation. The classic example of a mutation operator involves a probability that an arbitrary bit in a genetic sequence will be changed from its original state. A common method of implementing the mutation operator involves generating a random variable for each bit in a sequence. This random variable tells whether or not a particular bit will be modified. The purpose of mutation in GAs is preserving and introducing diversity. Mutation should allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. This reasoning also explains the fact that most GA systems avoid only taking the fitness of the population in generating the next but rather a random (or semi-random) selection with a weighting toward those that are fitter. Mutation is illustrated in below figure.

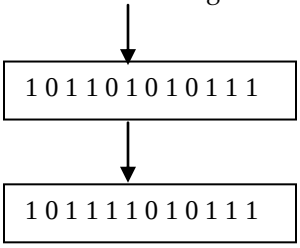

Fig.4 Mutation: Alteration of 5th bit

The off springs produced by the genetic manipulation process are the next population to be evaluated.

8. EXAMPLE: CPU SCHEDULING BY GENETIC ALGORITHMS

Scheduling in the operating system is a critical factor in the overall system efficiency. Process scheduling in an operating system can be stated as allocating processes to the processor so that throughput & efficiency of the system will be maximize and waiting time will be minimized. Typically scheduling problems are NP Complete problems i.e. algorithms that implement scheduling require exponential time to reach a solution. A genetic algorithm has the capability to find out the optimal job sequence which is to be allocated to the CPU. The algorithm starts with a population which is consists of several solution to the optimization problem. A member of population is called an individual. A fitness value is associated with each individual. Each solution in the population or an individual is encoded as a string of symbols. These symbols are known as genes & the solution string is called a chromosome. The values taken by genes are called alleles. Several pair of individual (parents) in the population mate to produce offspring by applying the genetic operator crossover. Selection of parents is done by repeated use of a choice function. A number of individuals & off springs are passed to a new generation such that the number of individual in the new population is the same as old population. A selection function determines which string forms the population in the next generation. Each serving string undergoes inversion with a specified probability.

Fitness function is Wi i.e. the waiting time of process where N is the total no. of processes.

Termination Criterion is that the genetic process will end if there is no change to the population's best fitness for a specified number of generations

Experimental Description

There are 6 jobs, which are to be considered. The number of possible sequences is 6! The total 10 sequences are selected out of 720 for the 6 jobs. Considering the number of jobs as 6 and the crossover point is 2 and 4. Let us consider following two individuals, which are marked as fit to generate next generation. 3 1 2 4 5 6 and 6 4 3 5 1 2

After cross over Child 1: 3 4 1 2 5 6

Child 2: 6 3 4 5 1 2

And let the inversion point be 2 and 5

After inversion Child 1: 3 5 1 2 4 6

Child 2: 6 1 4 5 3 2

After that we calculate the fitness of parent1, parent2, child1, child2 and two fittest individual are returned to the solution space. And above step is repeated until convergence criteria meet.

This example is an excellent illustration of how GA achieves the optimization.

9. ADVANTAGES AND DISADVANTAGES OF GAS

GA has number of advantages, some important among them are:

• Parallelism. GA works with multiple offspring's thus making it ideal for large problems where evaluation of all possible solutions in serial would be too time taking, if not impossible..
• Inductive. It can quickly scan a vast solution set. The inductive nature of the GA means that it doesn't have to know any rules of the problem - it works by its own internal rules. This is very useful for complex or loosely defined problems.
• Easy to implement. They are also easy to implement. Once you have some GA, you just have to write new chromosome (just one object) to solve another problem. With the same encoding you just change the fitness function and it is all. On the other hand, choosing encoding and fitness function can be difficult.

Disadvantages

• Certain optimization problems (they are called variant problems) cannot be solved by means of genetic algorithms. This occurs due to poorly known fitness functions which generate bad chromosome blocks in spite of the fact that only good chromosome blocks cross-over.
• There is no absolute assurance that a genetic algorithm will find a global optimum. It happens very often when the populations have a lot of subjects.
• Like other artificial intelligence techniques, the genetic algorithm cannot assure constant optimization response times. Even more, the difference between the shortest and the longest optimization response time is much larger than with conventional gradient methods. This unfortunate genetic algorithm property limits the genetic algorithms use in real time applications.

10. CONCLUSION AND FUTURE WORK

Genetic Algorithms are mainly inspired by DARWIN's theory of evolution. The whole concept is derived from single sentence "survival of the fittest". Nature itself provides best robust solutions to complex problems that we see day today in our life. If we understand those solutions and implement in complex problems, there might be no solution better than that. Genetic algorithms explore a far greater range of solutions to a problem than do conventional programs techniques like greedy techniques etc. They can be used when there is no other known efficient problem solving strategy, and the problem domain is NP-complete. These algorithms are extremely efficient, and are used in fields as stock exchange, production scheduling or programming of assembly robots in the automotive industry.
Finally, using Genetic Algorithms, there would be a no problem that remains unsolved even considering performance criteria.

The results provide an optimal solution for scheduling problems and this work can help us to design the effective algorithm for dynamic process scheduling in future.

REFERENCES

[1] "Introduction to Genetic Algorithms" by S.N. Sivanandam and S.N. Deepa.

[2] More Efficient Genetic Algorithm For Solving Optimization Problems-S. Ghoshray, K. K. Yen, Department of Electrical and Computer Engineering, Florida International University

[3] Tutorial GAs - A.A.R. Townsend.

[4]http://en.wikibooks.org/wiki/The_Computer_Revolution/Artificial Intelligence /Genetic Algorithms.

[5] David E.Goldberg, Genetic Algorithms in Search Optimization & Machine Learning, Second Reprint, Pearson Education Asia pte.Ltd, 2000.

[6] Genetic Algorithm approach to Operating System process scheduling Problem Dr.Rakesh Kumar, Reader Department of Computer Science and Application, Kurkshetra University, Haryana, India.

[7] K. R. Baker. Introduction to Sequencing and Scheduling. John Wiley and Sons, Inc., New York, 1974.

[8] William Stallings Operating Systems ISBN 0-13-031999-6

[9] Z.Michalewicz, "Genetic Algorithms + Data Structures =Evolutionary Programs", Springer- Verlag, Berlin, 1992.