International Journal of Combined Research & Development (IJCRD)
eISSN:2321-225X;pISSN:2321-2241 Volume: 1; Issue: 1; May -2013

Optimization Technique: Genetic Algorithm

Mr. Dharamvir
Dept. of Computer Applications
The Oxford College of Engineering
Bangalore-560068 , Karnataka , India

Abstract-This **paper presents detailed mechanics of genetic algorithm and its various applications. GAs can be used where optimization is needed. We mean that where there are large solutions to the problem but we have to find the best one. Given a specific problem to solve, the input to the GA is a set of potential solutions to that problem, encoded in some fashion, and a metric called a fitness function that allows each candidate to be quantitatively evaluated. These candidates may be solutions already known to work, with the aim of the GA being to improve them, but more often they are generated at random. The encoding of chromosomes very depends on the problem.**

Keywords**- Genetic algorithms, parameters, applications.**

I. INTRODUCTION

GAs developed by John Holland in the 1960s and subsequently studied by De-Jong, GoIdberg , Davis, Koza , Mitchell , to name only a few, have been originally proposed as a general model of adaptive processes, but by far the largest application of the techniques is in the domain of optimization . The original GAs (classical or canonical GAs) were typified by fixed-length binary representation of individuals, and fixed domain-independent operators.

Genetic Algorithms (GA) are stochastic search algorithms that borrow some concepts from nature. GA maintains a population pool of candidate solutions called strings or chromosomes. Each chromosome p is a collection of building blocks known as genes, which are instantiated with values from a finite domain. Let p & q denote the value of gene q in chromosome p in the population. Associated with each chromosome is a fitness value which is determined by a user defined function, called the fitness function. The function returns a magnitude that is proportional to the candidate solution's suitably and/or optimality. At the start of the algorithm, an initial population is generated. Initial members of the population may be randomly generated, or generated according to some rules. The reproduction operator selects chromosomes from the population to be parents for a new chromosome and enters them into the mating pool. Selection of a chromosome for parenthood can range from a totally random process to one that is biased by the chromosome's fitness. The cross-over operator oversees the mating process of two chromosomes. Two parent chromosomes are selected from the mating pool randomly and the cross-over rate, which is a real number between zero and one, determines the probability of producing a new chromosome from the parents. If the mating was performed, a child chromosome is created which inherits complementing genetic material from its parents. The cross-over operator decides what genetic material from each parent is passed onto the child chromosome. The new chromosome produced is entered into the offspring pool. This new chromosome may represent an unexplored point in the search space. The mutation operator takes each chromosome in the offspring pool and randomly changes part of its genetic make-up, i.e. its content. The probability of mutation occurring on any chromosome is determined by the user specified mutation rate. Chromosomes mutated or otherwise, are put back into the offspring pool after the mutation process. Thus each new generation of chromosomes are formed by the action of genetic operators (reproduction, cross-over and mutation) on the older population. Finally, the members of the population pool are compared with those of the offspring pool. The chromosomes are compared via their fitness value to derive a new population, where the weaker chromosomes may be eliminated. In exact, weaker members in the population pool are replaced by the fitter child chromosomes from the offspring pool. The heuristic for assessing the survival of each chromosome into the next generation is called the replacement strategy. The process of reproduction, cross-over, mutation and formation of a new population completes one generation cycle. A GA is left to progress through generations, until certain criteria (such as a fixed number of generations, or a time limit) are met. GAs were initially used for machine learning systems, but it was soon realized that GAs have great potential in function optimization. The general Evolutionary Algorithm (EA) has a structure as follows :

```
begin
        t: = 0;
        initialize P (t);
        evaluate P (t);
                while (not terminate) do,
                        t: = t+1;
                        p'(t): = select [P (t)];
                        P (t+1): = variation [P'(t) ];
                        Evaluate P (t+1);
```

end;
end.

This probabilistic algorithm maintains a population P(t) = {x_{1t}…..x_{nt}} of n chromosomes at generation t. Each chromosome represents a potential solution, & is represented as some data structure, S. Each solution is evaluated for its objective or fitness function. Then a new offspring population P‘(t) is formed at generation (t+1), by selecting the more fit individuals. Some members of this new population are subjected to variation operators such as crossover & mutation. Genetic Algorithms consists of population of chromosomes, selection according to fitness, randomized crossover to produce in new offspring & mutation. ‘Inversion’ - Holland’s forth element of Genetic Algorithms is rarely used now a days as its advantages if any are not well established.

II. GENETIC ALGORITHM

A. Overview

As a special kind of stochastic search algorithms and optimization algorithms, genetic algorithm is a problem solving method that consist of following steps:

1. **[Start]** Generate random population of *n* chromosomes (suitable solutions for the problem)
2. **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population
3. **[New population]** Create a new population by repeating following steps until the new population is complete
    1. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
    2. **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
    3. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
    4. **[Accepting]** Place new offspring in a new population

    [Replace] Use new generated population for a further run of algorithm
4. **[Test]** If the end condition is satisfied, **stop,** and return the best solution in current population
    [Loop] Go to step **2**
5.
6.

B. Genetic Encoding

The chromosome should in some way contain information about solution which it represents.

- *Binary Encoding:* The most used way of encoding is a binary string. Each chromosome has one binary string. Each bit in this string can represent some characteristic of the solution.
  Chromosome11101100100110110
  Chromosome 21101111000011110

- *Permutation Encoding: In permutation encoding, every chromosome is a string of numbers, which represents number in a sequence.*

  Chromosome 1 135246798
  Chromosome 2 857623194

- *Value Encoding: 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.*
  Chromosome 11.345 4.234 5.347
  Chromosome 2ADBGJFTDBKE

- *TreeEncoding:In treeencoding every chromosome is a tree of some objects, such as functions or commands in programming language.*

C. Fitness Function

Each chromosome in the population is associated with a fitness value that is calculated using a fitness function. This value indicates how good the solution is for a particular chromosome. This information is then used to pick the chromosomes that will contribute to the formation of the next generation of solution. This fitness function depends on the problem.

D. Selection

This stochastic GA operator selects chromosomes form the current population to form a new population according to their fitness values. The chromosomes with higher fitness values have higher probability of contributing one or more offspring in the next generation. There are various selection schemes:

- *Roulette Wheel Selection:* This is conducted by spinning a biased roulette wheel, which is sized in proportion to the fitness of each chromosome. Chromosome with bigger fitness will be selected more times. This can be simulated by following algorithm.

1. *Calculate sum of all chromosome fitnesses in population - sum S.*
2. *Generate random number from interval (0,S) - r.*
3. *Go through the population and sum fitnesses from 0 - sum s. When the sum s is greater then r, stop and return the chromosome where you are.*

•*Rank Selection:* The previous selection will have problems when the fitnessesdiffers very much. For example, if the best chromosome fitness is 90% of all the roulette wheel then the other chromosomes will have very few chances to be selected. Rank selection first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness 1, second worst 2 etc. and the best will have fitness N (number of chromosomes in population).

•*Steady-State Selection:* Main idea of this selection is that big part of chromosomes should survive to next generation. In every generation are selected a few (good - with high fitness) chromosomes for creating a new offspring. Then some (bad - with low fitness) chromosomes are removed and the new offspring is placed in their place. The rest of population survives to new generation.

•*Elitism:* It first copies the best chromosome (or a few best chromosomes) to new population. The rest is done in classical way. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution.

E. Crossover

Crossover selects genes from parent chromosomes and creates a new offspring. 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.

1. Single point crossover: one crossover point is selected, binary string from beginning of chromosome to the crossover point is copied from one parent, the rest is copied from the second parent

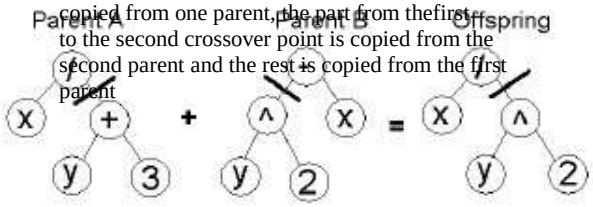
11001011+11011111 = **11001111**

In permutation encoding one crossover point is selected, till this point the permutation is copied from the first parent, then the second parent is scanned and if the number is not yet in the offspring it is added.

(1 2 3 4 5 **6** 7 8 9) + (4 5 3 6 8 **9** 7 2 1) = (1 **2 3 4 5 6** 8 9 7)

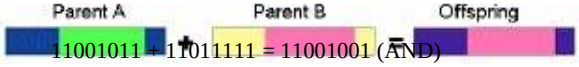
In tree crossover, in both parent one crossover point is selected, parents are divided in that point and exchange part below crossover point to produce new offspring

2. Two point crossover: two crossover point are selected,binary stringfrom beginning of chromosome to the first crossover point is copied from one parent, the part from thefirst to the second crossover point is copied from the second parent and the rest is copied from the first parent



11001011 + 11011111 = **11011111**

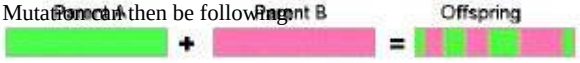
3. Uniform crossover: bits are randomly copied from the first or from the second parent



11001011 + 11011111 = 11001001 (AND)

F. Mutation

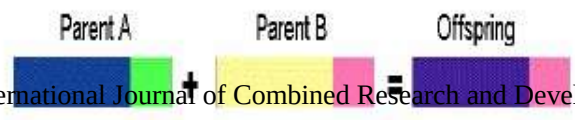
After a crossover is performed, mutation take place. This is to prevent falling all solutions in population into a local optimum of solved problem. Mutation changes randomly the new offspring. For binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1. Mutation can then be following:

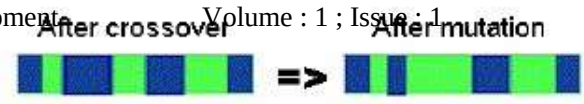


11001001 => 10001001

In permutation encoding two numbers are selected and exchanged.

(1 **2** 3 4 5 6 **8** 9 7) => (1 **8** 3 4 5 6 **2** 9 7)





In value encoding, a small number is added or subtracted to selected values.

(1.29 5.68 **2.86 4.11** 5.55) => (1.29 5.68 **2.73 4.22** 5.55)

III. PARAMETERS OF GA

1)*Crossover probability:* says how often will be crossover performed. If there is no crossover, offspring is exact copy of parents. If there is a crossover, offspring is made from parts of parents' chromosome. If crossover probability is 100%, then all offspring is made by crossover. If it is 0%, whole new generation is made from exact copies of chromosomes from old population (but this does not mean that the new generation is the same!).

2)*Mutation probability:* says how often will be parts of chromosome mutated. If there is no mutation, offspring is taken after crossover (or copy) without any change. If mutation is performed, part of chromosome is changed. If mutation probability is100%, whole chromosome is changed, if it is 0%, nothing is changed. Mutation is made to prevent falling GA into local extreme, but it should not occur very often, because then GA will in fact change to random search.

3)Population size: says how many chromosomes are in population (in one generation). If there are too few chromosomes, GA have a few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many chromosomes, GA slows down. Research shows that after some limit (which depends mainly on encoding and the problem) it is not useful to increase population size, because it does not make solving the problem faster.

IV. APPLICATIONS

A. Knapsack problem

There are things with given value and size. The knapsack has given capacity. Select things to maximize the value of things in knapsack, but do not extend knapsack capacity. This is example of binary encoding problem where each bit says, if corresponding thing is in knapsack.

B. Traveling salesman problem

There are cities and given distances between them. Traveling salesman has to visit all of them, but he does not to travel very much. Find a sequence of cities to minimize travelled distance. This is example of permutation encoding problem where chromosome says order of cities, in which salesman will visit them.

C. Finding weights for Neural network problem

There is some neural network with given architecture. Find weights for inputs of neurons to train the network for wanted output. This is example of value encoding problem where real values in chromosomes represent corresponding weights for inputs.

D. Programs (Finding a function from given values)

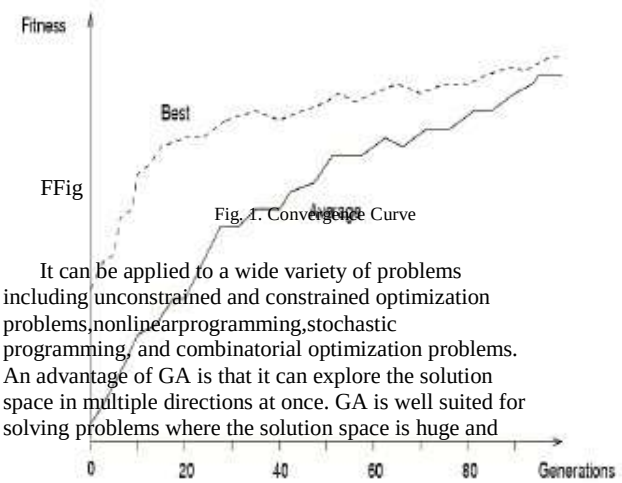
Some input and output values are given. Task is to find a function, which will give the best (closest to wanted) output to all inputs. This is example of tree encoding problem where chromosomes are functions represented in a tree.

E. Shortest path problem

It is defined as that of finding a minimum-length (cost) path between a given pair of nodes. It is a classical research topic. When network is very big, genetic algorithm is efficient. This is example of value encoding problem where each chromosome is encoded as a series of node IDs that are in the path from source to destination.

V. CONCLUSION

The Genetic Algorithm is a relatively simple algorithm that can be implemented in a straightforward manner. The idea behind GA is to have the chromosomes in the population to slowly converge to an optimal solution as shown in fig. 1.



FFig

Fig. 1. Convergence Curve

It can be applied to a wide variety of problems including unconstrained and constrained optimization problems,nonlinearprogramming,stochastic programming, and combinatorial optimization problems. An advantage of GA is that it can explore the solution space in multiple directions at once. GA is well suited for solving problems where the solution space is huge and

time taken to search exhaustively is very high They perform well in problems with complex fitness. If the function is discontinuous, noisy, changes over time or has many local optima, then GA gives better results. GA has ability to solve problems with no previous knowledge (blind). The performance of GA is based on efficient representation, evaluation of fitness function and other parameters like size of population, rate of crossover and mutation and the strength of selection.

REFERENCES

[1] Goldberg D. E., *"Genetic Algorithms in Search, Optimization, and Machine Learning,"* Addison-Wesley, Reading, MA, 1989.

[2] Neubauer, *"The circular schema theorem for genetic algorithms and two-point crossover,"* in Proc. of Genetic Algorithms in Engineering Systems: Innovations and Applications, pp. 209–214, Sept. 1997.

[3] Holland J.H., *"Adaptation in Natural & Artificial Systems,"* Ann. Arbor: The Uni. of Michigan press, 1975.

[4] Holland J.H., *"Outline for a logical theory of adaptive systems"* J. Assoc. Computer. Mach., vol.3. Pp.297-314, 1962.

[5] Michalewicz Z., *"Genetic Algorithms + Data Structures = Evolution Programs,"* Berlin: Springer – Verlag, 1992.

[6] Salman Yousof et.al. *"A Parallel Genetic Algorithm for Shortest Path Routing problem,"* International Conference on Future Computer and Communication, 2009

[7] Lixia Hanr et.al., *"A Novel Genetic Algorithm for Degree-Constrained Minimum Spanning Tree Problem,"* IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.7A, July 2006.

[8] J. B. Kruskal, *"On the Shortes Spanning Tree of a Graph and the Traveling salesman Problem,"* Amer. Math. Soc., vol. 7, pp. 48-50, 1956.

[9] R. Prim, *"Shortest Connection Networks and Some Generalization,"* Bell Syst. Tech. J., vol. 36, pp. 1389-1401, 1957.

[10] Chang Wook Ahn & R. S. Ramakrishna, *"A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations,"* IEEE Transactions On Evolutionary Computation, Vol. 6, No. 6, December 2002.

[11] Yee Leung, Guo Li, and Zong-Ben Xu, *"A Genetic Algorithm for the Multiple Destination Routing Problems,"* IEEE Transactions on Evolutionary Computation, Vol. 2, on 4,November 1998.