Genetic Algorithm: A Search of Complex Spaces

Namita Khurana, Anju Rathi, Akshatha.P.S
Lecturer in Department of (CSE/IT)
KIIT College of Engg., Maruti Kunj,
Sohna Road, Gurgaon, India

ABSTRACT
Living species solve very complex problem of optimization through the mechanism of evolution and natural selection. Genetic Algorithm has been a field of active interest and applied to solve problems almost in all the fields like Computer Science, Electrical Engg., Mechanical Engg., Optimization, Biology and Image Processing etc. One important application of Genetic Algorithm is to search complex spaces and function optimization. A genetic algorithm begins its search with random solution of the problem. The initial population is evolved to new population using Genetic operators like reproduction, crossover and mutation. A Genetic Algorithm keeps evolving the successive populations unless some criterion is met or a reasonable acceptable solution is found. In this paper Genetic Algorithm has been applied to schwefel function to find the best fit chromosome so far.

Keywords: Genetic Algorithm, Schwefel function, Crossover, Mutation, Selection

1. INTRODUCTION

1.1 Brief History of Genetic Algorithms

Humans being have normal tendency to look for the best possible or optimal solutions of the problems. In this attempt the earliest methods developed are either direct or gradient based. In direct methods the search is guided by the objective function and constraints. In the gradient based methods, like hill climbing, the first and second order derivatives are taken. The direct method fails where it is not possible to enumerate all possible values for the problem and hill climbing is successful only where not many peaks exist in the search space of a function. These traditional methods are not sufficient to be applied to a wider class of problems and lack global perspective. Moreover, most of these methods are serial in nature and cannot utilize the parallel computing environment[1].

Idea of evolutionary computing was introduced in the 1960s by I. Rechenberg in his work “Evolution strategies". His idea was then developed by other researchers. Genetic Algorithms (GAs) were invented by John Holland and developed by him and his students and colleagues. This lead to Holland's book "Adaption in Natural and Artificial Systems" published in 1975.

John Holland gave a new technique to search the complex spaces which is known as Genetic Algorithms. It is the adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics.

1.2 What is Genetic Algorithm?

GA’s are general purpose search algorithms that use the principles inspired by natural genetic population to evolve solutions to problems. The basic idea is to maintain a population of chromosomes, which represent candidate solutions to the concrete problem that evolves over time through a process of competition and controlled variation. Each chromosome in the population has an associated fitness to determine which chromosomes are used to form new ones in the competition process, which is called selection. The balance between exploration and exploitation or ,in other words, between creation of diversity and its reduction , by focusing on the individuals of higher fitness, is essential in order to achieve a reasonable behavior for GA’s in case of complicated optimization problem.

2. GENETIC ALGORITHM: WORKING PRINCIPLE

2.1 Working of Genetic Algorithm

Genetic algorithm starts working on a randomly generated set of solutions, known as initial population. Each solution is represented by a fixed length string of binary numbers (i.e 101010…). Fitness is associated with each solution. The fitness evaluation is based on the objective function. In this each string representing the solution is called chromosome, each bit of the string is called the gene. The set of strings is called population[1].

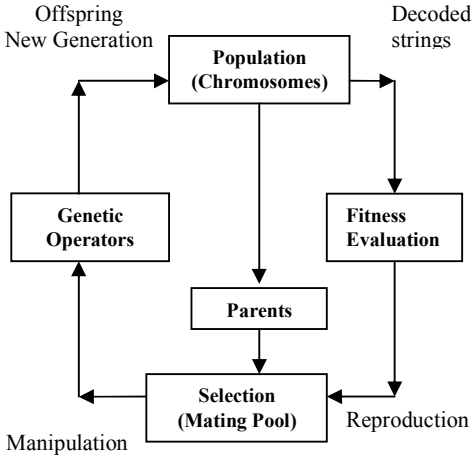


Figure1: A flowchart of working of genetic algorithm

2.2 Outline of the Basic Genetic Algorithm

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
   I. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
   II. **[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.
   III. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
   IV. **[Accepting]** Place new offspring in a new population
4. **[Replace]** Use new generated population for a further run of algorithm
5. **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population
6. **[Loop]** Go to step **2**

Genetic Algorithm generates new population of chromosomes by selecting the better fit solutions from existing population and applying genetic operators to produce new offspring of the solutions[6]. The process is repeated successively to generate new population iteratively. In this way every successive population is better fit then the previous population .This process is repeated until some criterion is met or a reasonably acceptable solution is found.

2.3 Representation and Encoding of the Population

When we start to solve a problem with the help of GA, encoding the solutions or chromosomes in the initial population is the first decision to be made. The encoding of the chromosomes is problem dependent. There are various encoding schemes like binary encoding, permutation encoding, value encoding and tree encoding. For searching the complex spaces, binary encoding is being used.

In binary encoding, each chromosome is a string of bits 0 or 1.let us take a simple example to explain the encoding. Let us take a simple example to explain the coding. Let $f(x) = x.\sin(10\pi.x) + 1.0$ where
$-1<=x<=2$; be the function to be optimized for the maximum value of x. The values of x ranging between -1 and 2. are encoded in binary strings. Let us take the length of the string l=16. Then there are 2^{16}(65536) possible values of x. these 2^{16}values are to be assigned to x. we assign the values as follows [6]:

0000000000000000 represents x=0
0000000000000001 represents $x=1*(\pi/2^1-1)$
0000000000000010 represents $x=2*(\pi/2^1-1)$
.
.
1000000000000000 represents $x=2^{15}*(\pi/2^1-1)$
1111111111111111 represents $x=\pi-(\pi/2^{16})$ just a value before π

For a value x we can generalize that

$$X = Xmin + \frac{(Xmax-Xmin)}{2^l - 1} * DV(Si)$$

where Dv(Si) is the decoded value of string Si.

2.4 Evaluation of the Fitness

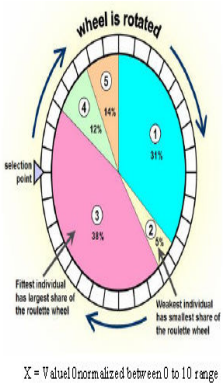
Each string in initial population or subsequent population is assigned a fitness value which is related to the objective function. For maximizing a function the fitness can be equal to string's objective function value. To find the optimal minimum the fitness will be equal to $1/(1+f(x))$.
The beauty of the binary coding is shielding between actual problem and the working of GA. The GA processes only the strings of bits which may represent any number of variables depending on the problem. We have to change only the definition of the coding.

2.5 Reproduction and Selection

Reproduction selects good strings from the population and puts them in mating pool. There are number of reproduction operators. The idea is to pick up the strings with above average fitness from current population and apply genetic operators to new strings for the successive population. One of the important techniques is the fitness proportionate selection. The chances of a string Si being selected to participate in reproduction are proportional to its fitness. This is performed by roulette wheel selection .Here; all the chromosomes are placed on an imaginary roulette wheel where each chromosome in the population gets a place big on the wheel proportionate to its fitness. A roulette wheel for five chromosomes is shown in fig below:

2.5.1 Roulette Wheel Selection

- Fitness level is used to associate a probability of selection with each individual solution.
- We first calculate the fitness for each input and then represent it on the wheel in terms of percentages.
- In a search space of 'N' chromosomes, we spin the Roulette Wheel.
- Chromosomes with bigger fitness will be selected more times [4].



No.	Chromosome	Value$_{10}$	X	Fitness f(x)	% of Total
1	0001101011	107	1.05	6.82	31
2	1111011000	984	9.62	1.11	5
3	0100000101	261	2.55	8.48	38
4	1110100000	928	9.07	2.57	12
5	1110001011	907	8.87	3.08	14
Totals				22.05	100

Example population of 5 for: $f(x) = -\frac{1}{4}x^2 + 2x + 5$

Value10: Value of chromosome to the base 10

Figure 2: Roulette Wheel Selection

Other techniques for selection are tournament selection, steady state selection rank selection.

2.5.2 Crossover and Mutation

Crossover and mutation are two basic operators of GA. Performance of GA very depends on them. Type and implementation of operators depends on encoding and also on a problem. Here the crossover and mutation are shown to work only on the binary encoding.

In crossover operation the right side portion of the strings are swapped among themselves to create two new strings to represent the two new chromosomes and solutions. The process is shown below[1]:

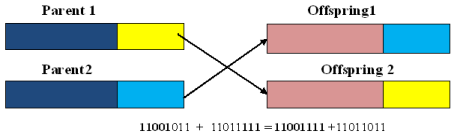


Figure 3: Crossover Operator

Figure 4: Mutation Operator

*** Mutation (2nd and 14th bits mutated in figure 4)**

There are many ways how to do crossover and mutation. Mostly a single point crossover is used in GA.the performance of GA depends on the retaining of good building blocks of the strings to successive generations. Crossover operator is responsible to search different ares of search space. In mutation some bits are inverted. The probability to mutate a bit is kept low. The mutation is used to kept diversity in the population.

After reproduction crossover and mutation are applied to whole population one generation is completed. The GA operators are applied in the hope that they will produce the better fit solutions. Even if some bad solutions are generated they would not survive over the generations and better solutions contribute more to generate new but even better fit off springs over successive generations.

3. SIMULATION FOR SCHWEFEL FUNCTION

3.1 A Simple Simulation

To illustrate the working of GA I've taken the example of schwefel function.
No. of variables: n variables[5].

Definition :

$$f(x) = 418.9829\ n - \sum_{i=1}^{n} (x_i \sin\sqrt{|x_i|})$$

Search Domain: $-500 <= x_i <= 500$ i=1,2,…….,n.
Number of local minima: several local minima.

3.2 Parameters of test simulation

The parameter setting is very important to make GA work successfully. For the parameter setting, the population should be large enough for adequate supply of building blocks over the generations. The probability of crossover and mutation should be set to allow adequate combining and mixing of building blocks but not to spoil too much of the good blocks contributing to the fitness[1].

3.3 Results of test simulation

These results has been found out by implementing the G.A in 'C' language[7].

Population Size (pop size) 80

Dimension 2

Cromosome length (lchrom) 10

Maximum num of generationsmaxgen) 20

Crossover Probability (pcross) 0.750000

Mutation Probability (pmut) 0.015000

Binary chromosomes	x1	x2	fitness
00101010100111000100	-167.64418	-361.19257	7.04940
00010101111111101110	414.95601	-31.76931	7.59671
00011110100011010010	-132.45357	-206.74487	7.07672
10110111000000011000	-268.32845	-406.15836	6.93579
10101000000110110010	-479.47214	-196.96970	6.91662
11100101110100001101	413.97849	190.12708	7.75242

.
.

0 Generation Stats
Max Fitness: 7.7524 — Average Fitness: 7.1287
Min Fitness: 6.4659 — sumfitness = 570.2925
No.of cross over occured: 0 — The best fit chromosome so far: 7.752424
No. of mutations: 0 — best x1 = 413.978495 best x2 = 190.127077

1 Generation Stats
Max Fitness: 7.4119 — Average Fitness: 7.0618
Min Fitness: 7.0494 — sumfitness = 593.1876
No.of cross over occured: 0 — The best fit chromosome so far: 7.752424
No. of mutations: 79 — best x1 = 413.978495 best x2 = 190.127077

2 Generation Stats
Max Fitness: 7.3665 — Average Fitness: 7.0607
Min Fitness: 7.0494 — sumfitness = 621.3397
No.of crossover occured: 0 — The best fit chromosome so far: 7.752424
No. of mutations: 79 — best x1 = 413.978495 best x2= 190.127077

3 Generation Stats
Max Fitness: 7.6341 — Average Fitness: 7.0735
Min Fitness: 7.0494 — sumfitness = 367.8195
No.of crossover occured: 1 — The best fit chromosome so far: 7.752424
No. of mutations: 78 — best x1 = 413.978495 best x2= 190.127077

.
.
.

18 Generation Stats
Max Fitness: 7.2747 — Average Fitness: 7.1041
Min Fitness: 7.0494 — sumfitness = 56.8328
No.of crossover occured: 1 — The best fit chromosome so far: 7.752424
No. of mutations: 39 — best x1 = 413.978495 ,best x2=190.127077

19 Generation Stats
Max Fitness: 7.2747 — Average Fitness: 7.1057
Min Fitness: 7.0494 — sumfitness = 56.8457
No.of crossover occured: 0 — The best fit chromosome so far: 7.752424
No. of mutations: 40 — best x1 = 413.978495, best x2 = 190.127077

20 Generation Stats
Max Fitness: 7.2747 — Average Fitness: 7.1057
Min Fitness: 7.0494 — sumfitness = 56.8457
No.of crossover occured: 1 — The best fit chromosome so far: 7.752424
No. of mutations: 40 — best x1 = 413.978495 best x2 = 190.127077

3.4 Function Graph for n=2

The graph for the schwefel function [5] is shown in figure 5

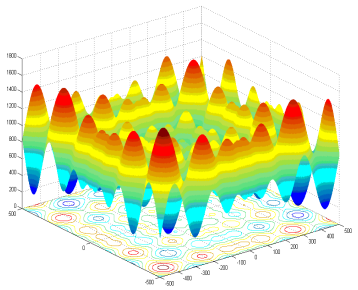


Figure 5: Function Graph

4. CONCLUSION

First of all it is important to get the idea about the working of Genetic Algorithm. So this paper gives the basic information about the G.A and tell us how to initialize the large number of population chromosomes, how to apply the operators like selection ,crossover and mutation on the population. The test simulations are the results after implementing all the operators on the initial population using the schwefel function. The simulation has been done for a number of generations .The results of the simulation gives the best fit chromosome found so far on the basis of fitness function.

5. FUTURE TRENDS

This paper is just the calculation of best fit chromosomes in the genetic algorithm by using the schwefel function. The future work in this paper can be on the comparison of encoding techniques in the genetic algorithms. The implementation can be done on different types of fitness functions. GA has a large number of application areas so it has a vast field for research work. To make Genetic Algorithm more effective and efficient it can be incorporated with other techniques within its framework to produce a hybrid Genetic Algorithm that can reap best from its combination. More research needs to be concentrated on the development of hybrid design alternatives for its efficiency enhancement.

6. REFERENCES

[1] D.E.Goldberg, "Genetic Algorithms in search, Optimization and Machine Learning", Addison Wesley Publishing Company, Ind. U.S.A, 1989.

[2] Charles C.Peck, Atam P. Dhawan, "Genetic algorithm as global random search methods: An alternative Perspective", Evolutionary Computation, Volume 3 Issue1, MIT Press, march 1995.

[3] Math world available at: http://mathworld.com.

[4] Pratibha Bajpai et al./Indian Journal of Computer Science and Engineering Vol 1 No 3 199-206, " Genetic Algorithm-an Approach to solve Global Optimization Problems".

[5] Test Functions available at: http://www-optima.amp.i.kyoto u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page2530.htm

[6] Melanie Mitche,"An introduction to genetic Algorithm", MIT press,1998.

[7] L.T.Leng, Guided Genetic Algorithm, Doctoral Dissertation.University of Essex, 1999.

[7] Yashwant Kanitkar "Pointers in c".

[8] Lawrence Davis. Handbook of Genetic Alogorithms. Van Nostrand Reinhold, NewYork, 1991.

[9] D. Whitley, D.Garrett, and J.Watson. Genetic Quad Search.

[10] Math works available at: www.mathworks.com

[11] Wikipedia (2004).Genetic Algorithm. Available from URL: http://en.wikipedia.org/wiki/Genetic_algorithm

[12] Holland, J. (1975). Adaptation In Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

[13] Schaffer, J.D. and Eshelman, L. (1993). Real-coded genetic algorithms and interval schemata. Foundations of Genetic Algorithms,2,ed.D.Whitley.Morgan Kaufmann, an Mateo, A.

[14] Schwefel, H.P. (1981). Numerical optimization of Computer models. John Wiley, New York.

[15] Baker, J. (1985). Adaptive selection methods for genetic algorithms. In proceedings of the International Conference on Genetic Algorithms and Their Applications.

[16] L.T.Leng,Guided genetic algorithm,Doctrol Dissertation. University of Essex,1999.

[17] L. Painton, J.Campbell, "Genetic Algorithms in optimization of system Reliability" Reliability, IEEE Transaction on Volume: 44 ,Issue 2, Digital Object Identifier.

[18] T.Weise, "Global Optimization Algorithms-Theory and Application".