From Genetics to Genetic Algorithms

Solution to Optimisation Problems Using Natural Systems

Jitendra R Raol and Abhijit Jalisatgi

Genetic algorithms are search procedures inspired by natural selection and genetics that can be **used** to obtain global and robust solutions **to** optimisation problems. They find applications in computer science, engineering, economics, linguistics, psychology, biology, etc.

Robustness, the ability to strike a balance between efficiency and efficacy, is necessary for any system to survive in many different environments. Nature usually offers good soluti whenever robustness is 'required. Biological systems are more robust, efficient and flexible than the most sophisticated artificial systems. Aitificial systems have to learn from biological ones to improve rheir performance and carry out their functions for ionger periods of time. Genetic algorithm are based on principies drawn from natural systems.

Generic algorithms (GAs) are computational optimisation schemes with an unconventional approach. They were developed by John Holland and his colleagues at rhe University of Michigan. The algorithms solve optimisation problems imitating nature in the way it has been working for millions of years on the evolution of life. Inspired by biological systems, GAs adopt the rules of natural selection and generics to achieve robustness. Acting on the premise of survival of the fittest, a population of possible solutions is combined in a manner similar to the mixing of chromosomes in a natural genetic system. The fitter population members pass on their structures as genes (*Resonance*, Vol.1, No.1, p 40) in far greater quantities rhan less fit members. The net effect is evolution of the population towards an optimum.

J. R. Raol is a scientist with NAL. His research interests are parameter estimation, neural networks, genetic algorithms and fuzzy systems and their applications to aerospace problems. He also writes poems in English.

Abhijit Jalisatgi is a graduate research student from Karnataka Regional Engineering College, Surathkal working an the implementation of GAs on parallel computers.

Genetic algorithms are based on principles drawn from natural system

GAs operate by combining the information present in different possible solutions for a given problem, in such a way that a better solution is obtained in future generations. Terminologies used in natural genetic systems (NGS) and (GAs) are given in the box below. Often NGS terminology is used freely in the description of GA.

Terminology in Natural Genetic Systems **and** Genetic **Algorithms**

NGS (biological system)	GA (artificial genetic system)
chromosome	string
gene	feature **or** detector
allele	feature value
locus	string position
genotype	structure
phenotype	parameter set, alternative solution, a decoded structure

The ***strings*** of artificial genetic **systems** are analogous to chromosomes.

Comparison of Natural Systems **and GAs Terminology**

Chromosomes are long stretches of **DNA** that carry the genetic information needed to build an organism (*Resonance*, Vol.1, No.1). The ***strings*** of artificial genetic systems are analogous to chromosomes. Chromosomes are composed of ***genes.*** **Each** gene is a unit of information that takes different values called *alleles* at different *loci.* In artificial systems, strings are composed of *features* or *detectors* that assume different values (0 or **1** in case of binary coding) located at different positions on the string. The total genetic package is called the genotype whereas it is called a structure in artificial genetic systems. In natural systems, the organism formed by the interaction of the total genetic package with its environment is called the phenotype. In artificial genetic systems, the structures decode to form a particular parameter set or a possible solution.

Operations in a GA

Chromosomes : Chromosomes represent encoding of information in a string of finite length on which the algorithm operates. Each chromosome consists of a string of bits. Each bit may be binary 0 or 1, or it may be a symbol from a set of more than two elements. Generally for function optimisation problems, chromosomes are constructed from binary strings. A few examples of encoded parameters in a binary string of length 6 are shown alongside.

Parameters (Numeric Values)	String
6	000110
12	001100
34	100010
20	010100

Population and ***Fitness*** : GAs operate by maintaining *a* population of such possible solutions with chromosomes. Population members are called individuals. Each individual is assigned a fitness value based on the cost function. Better individuals (solutions) have higher fitness values and weaker individuals (solutions) have lower fitness values. A simple GA is composed of the operations discussed next.

Chromosomes represent encoding of information in a string of finite length on which the algorithm operates.

Initialisation and Reproduction : A population of possible initial solutions is created by randomly selecting information from the search space and encoding it. Selecting the information means assigning random values to the decision variables of the cost function. Reproduction is a process in which individual strings are copied according to their fitness values.

Cost Function, Decision Variables, Search Space

In most of the practical optimisaiion problem, ihe aim is to find optimal parameters to increase the production and/or to reduce the expenditure/ loss, i.e. to get maximum profit by reorganising the system and its parameters. Since this will finally reflect on the cost, it is represented by the cost function (*Figure 1*). A carefully written and convergent computational algorithm (*Resonance*, Vol.1, No.1 Introduction to 'Algorithms') would eventually find an optimum solution. The parameters of the system that decide the cost are called decision variables. The search space is a Euclidean space (*Resonance*, Vol.1, No.1 'Geometry: The Beginnings') in which parameters take different values and each point in this space is a possible solution of the problem.

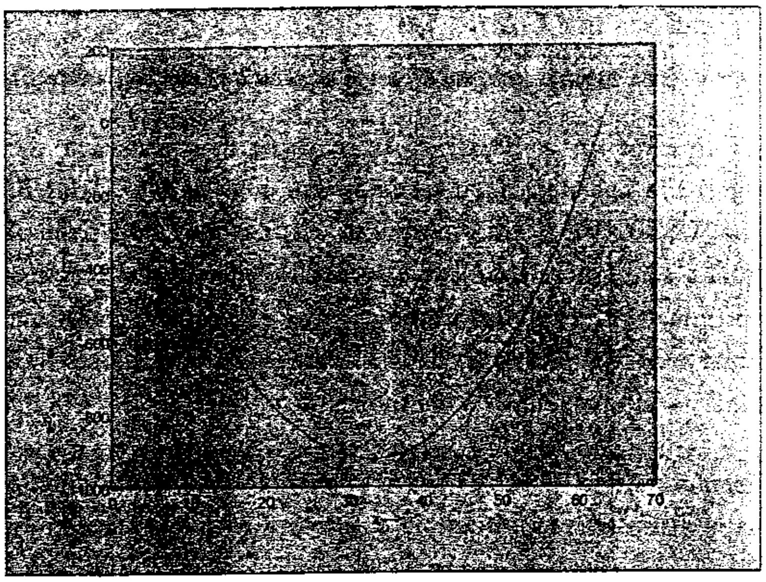


Figure 1 A cost function f(x) has x as parameter or variable which when optimally determined, will yield either maximum or minimum value of the cost function depending upon the nature of the problem. The function has a global maximum of 100 at x = 0. Also it has local maxima of 37 at x = 63 and minimum of –924 at x = 32.

This means that strings with a higher fitness have higher probability of contributing one or more offspring to the next generation.

Crossover : After reproduction, a simple crossover may proceed in two steps: i) members of newly reproduced strings in a mating pool are mated at random, and ii) each pair of strings undergoes crossover. In a crossover operation, a sire is selected randomly along the length of the chromosome, and each chromosome is split into two pieces by breaking at the crossover site. The new chromosomes are then formed by joining the top piece of one chromosome with the tail piece of another (*Figure 2*). Thus as in natuial genetics, the population members are allowed to mate in a probabilistic manner, with each combination producing offspring that are similar but not identical to both parents.

In simple GAs, a mutation is a random alteration of a value of the string position.

Mutation : A mutation operator is included in most GAs. In simple GAs, a mutation is a random alteration of a value of the string position. This operator helps to gain information that is not available to the rest of the population. The purpose of the mutation operator is to prevent loss of important information by effectively increasing the population diversity.

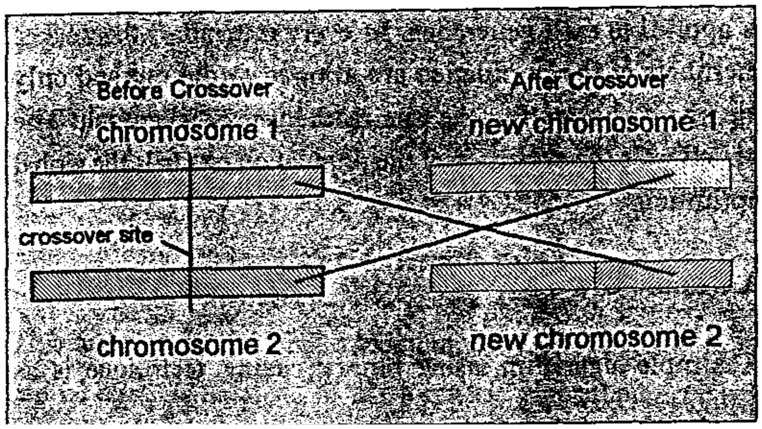


Figure 2 Crossover operation is exchanging the high performance notions to form new ideas in the search for better performance. This operation is included in GAs because it efficiently builds new ideas from the best partial solutions of previous trials

Generation : Each iteration in this optimisation procedure is called a generation. In each generation, pairs of individual chromosomes are chosen for crossover operation. The fitness determines the likelihood for reproduction and crossover probability for new offspring is applied to decide which individual will undergo crossover. Mutation is randomly carried out during the crossover operation (during or after has a subtle distinction). A new population evolves from these operations.

Survival of the fittest : The individuals may be fitter or weaker than other population members and must be evaluated and placed at their relevant ranked place in the population. This ranking process will involve some form of sorting routine that

GAs vs Conventional Search Methods

In conventional search methods, a point is chosen in a search space, and the next point is obtained using the gradient of the cost function at that point. One may get to the local peaks or troughs in multimodal search spaces (See *Figure 3*). Many search techniques require a lot of auxiliary information in order to work properly. For example, gradient techniques need derivatives of the cost function with respect to search variables {change in $f(x)$/change in x} in order to climb up (down) the current peak (trough) (*Figure 1*). GAs work simultaneously from a rich database of points, climbing many peaks in parallel. Thus the probability of finding a local peak is reduced as compared to other methods. GAs only require cost function values associated with individual strings, to perform an effective search for better solutions. They can be applied to a large range of optimisation problems that require cost function minimisation or maximisation.

is applied to each generation. In every generation the weaker members of the population are allowed to die out and only members who are fit take part in the genetic operation. The net effect is the evolution of the population towards the global optimum.

A Simple GA

A simple algorithm using binary coding technique is as follows:

1. Create population of N samples from a chosen search space – denoting the decision variables.
2. Produce series of 0s and 1s to create chromosomes – encoding the decision variables.
3. Calculate cost function values and assign fitness to each individual.
4. Sort the individuals according to respective fitness values and apply the reproduction operator.
5. Carry out crossover operation taking two chromosomes at a time.
6. Mutate the chromosomes during crossover with a given probability of mutation.
7. Replace weakest member of new population by the best member of old population.
8. Replace old generation by new population.
9. Repeat step 3 to step 8 for a given number of generations.

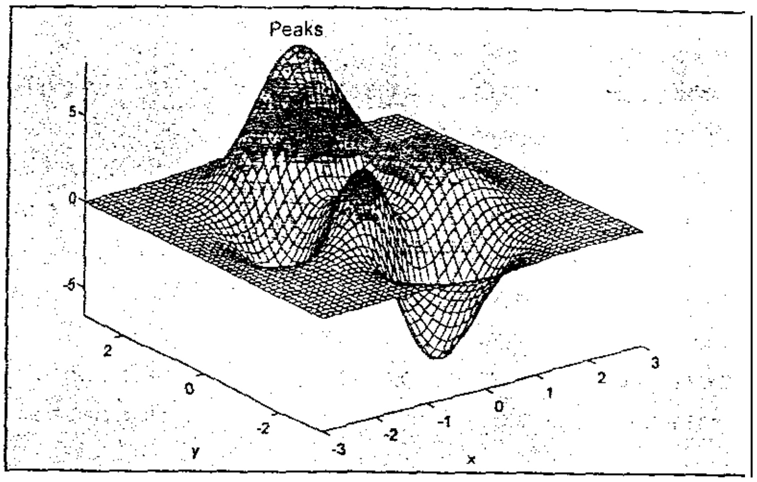


Figure 3 Multimodal function-surface has more than one maximum or minimum. The aim of a GA is to obtain the values of parameters that give a global optimum.

Simulation of a G.4

Consider the problem of maximising the function (*Figure* 1)

$$f(x) = x^2 - 64x + 100,$$

where x varies from 0 to 63.

This function has a global maximum value of 100 at $x = 0$, as can be easily computed. To use a generic algorithm, decision variables of the problem have to be coded in strings of finite length. For this problem, we can encode the variables as a binary string of length 6. We create an initial population with 4 samples by randomiy selecting them from the interval 0 to 63 and encode each sample. A binary string of length 6 car, represent any value from 0 to 63 (2^6–1); hence string length is chosen as 6 for the example. Four encoded samples in the initial population are:

5	=	000101
60	=	111100
33	=	1051001
8	=	001000

These individuals are sorted according to their fitness values. (They are arranged in the descending order of their fitness values). In this case, the fitness value is the same as the cost function value. These sorted individuals are given as:

No.	x	String	Fitness
1	60	111100	–140
2	5	000101	–195
3	8	001000	–348
4	33	100001	–923

To use a genetic algorithm, decision variables of the problem have to be coded in strings of finite length.

In the 1st generation, the 1st and 2nd strings are crossed over at site 3, (crossover site is randomly selected) to get two new strings:

crossover site	new strings	fitness of new strings
|		
111|100	111101	–83
000|101	000100	–140

Similarly rhe 3rd 2nd **41h** strings are crossed over at crossover sire 2, to get:

crossover site	new strings	fitness of new strings
|		
00|1000	000001	37
10|0001	101000	–860

No.	x	String	Fitness
1	1	000001	37
2	61	111101	**-83**
3	**4**	**0001**00	–140
4	40	101000	**-860**

We see that in one generation fitness is improved from –140 to 37 ($f(1) > f(60)$). Before proceeding to the next step, the weakest member of the population **is** replaced by the fittest member of previous population, i.e. **,** smng 10 10 0 0 that has fitness –860 is replaced ***by*** string 1 1 1 1 0 0, whose fitness ***is*** –140.

In the 2nd generation, the 1st and 2nd strings arecrossed over ar site 1, to get:

crossover site	new strings	fitness of new strings
|		
0|00001	011101	–915
1|11101	100001	–923
|		

Similarly **rhe** 3rd and 4th strings are crossed over at crossover site 3 to get:

crossover site	new strings	fitness of new strings
000|100	000100	–140
111|100	111100	–140

Replacing the weakest member by the fittest member of the previous population (string 1 0 0 0 0 1 with fitness value of –923 is replaced by string 0 0 0 0 0 1 with fitness value of 37) and sorting than according to the fitness values, we get:

No.	x	String	Fitness
1	1	0 0 0 0 0 1	37
2	4	0 0 0 1 0 0	–140
3	60	1 1 1 1 0 0	–140
4	29	0 1 1 1 0 1	–915

In the third generation, the above process of crossover (at site=4) is repeated (not shown here). The new set of strings in the population, after replacement of the weakest by the fittest member of the previous population is given as:

2		0 0 0 0 0 1	37
3	El	1 1 1 1 0 1	–83

For simplicity, the probability of the mutation is chosen as 0 (zero). We see that as genetic operations continue from one generation to rhe nest, improved solutions evolve. At x=0, $f(x)$=100, which is the desired result. Here, the problem which could have been solved using conventional computational optimisation algorithms, is used to illustrate GA operations for the sake of simplicity.

Stopping Strategies

One can stop the search after a certain number of generations. However, for a fixed population size, more generations might be needed for convergence of the GA. The search can be stopped when no further improvement in fitness is detected. There are several types of stopping criteria and the research for an optimal criterion continues. Since GAs employ multiple concurrent search points, it is important to maintain the diversity of such points. A convergence criterion can be based on the assumption that as the GA progresses there will be a time when a large number of generations is needed to bring a small improvement in the fitness value. This can be ascertained from the gradient of a plot of best fitness against the number of generations needed to obtain that fitness. One can do an effective search if one exploits important similarities in the coding used in GAs. This leads one to the important notion of a *similarity template* or *schema*.

GAs Without Coding the Parameters

GAs become complex because of the efforts involved in encoding and decoding the chromosomes. Also they do not provide the variety of easy options, required to tackle the spectrum of problems faced in science and engineering which involve floating point numbers. For higher dimensional problems this leads to very long chromosomes. In such cases real number parameters can be directly used in genetic operations with some modifications in crossover and mutation operations. A crossover operation may involve only averaging of the information. A mutation is done by adding a small noise to the information. For a cost function varying with a parameter X, if 'A' and 'B' are two individuals with X_a and X_b as parameters, when they are crossed over, the new individual 'C' is created with parameter $X_c = (X_a + X_b)/2$. The best individual in population 'A' with parameter X_a will undergo a mutation to create a new individual 'D' with parameter X_d, where $X_d = X_a + d_m \zeta$. Here d_m is a constant and ζ is a number chosen randomly between –1 to 1. In each generation the population is refreshed by taking new samples from the search space. A simple algorithm that does not code parameters can be obtained from the genetic algorithm described earlier, by incorporating the new crossover and mutation operations as discussed above.

Results of the application of **a GA** (without coding of the variables to the function of *Figure 4)* show that, increase in the number of samples in the population, increases the success of reaching the global optimum. In this case the number of generations was 10 and the number of trials was 1000. The required accuracy was set to 0.0001. The percentage of success is defined as 100 times the number of trials in which the global minimum reached is divided by the total number of trials. For 15 samples the success mas 100%. Some of these results have been obtained using NAL's parallel computer (Flosolver).

Concluding Remarks

Nature has been using GAs for millions of years and along the way it has produced complex, intelligent living organisms capable of reproduction, self-guidance and repair. Although

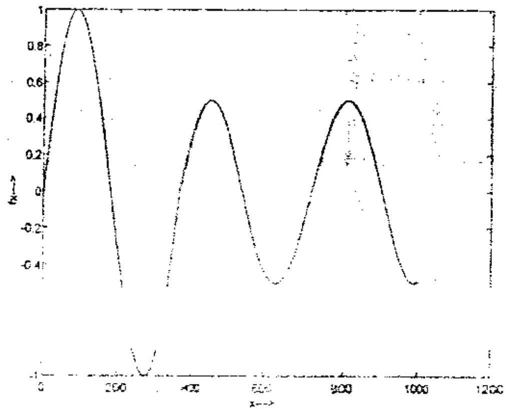


Figure 4 Function with more than one minimum. The aim is to search for a global minimum. Function: f(x) = sin(x), for 0<x<360; =[sin(x)] /2 for 360<x<1000. It has a global minimum f(x)= -1.0 at x=270. It has two local minima.

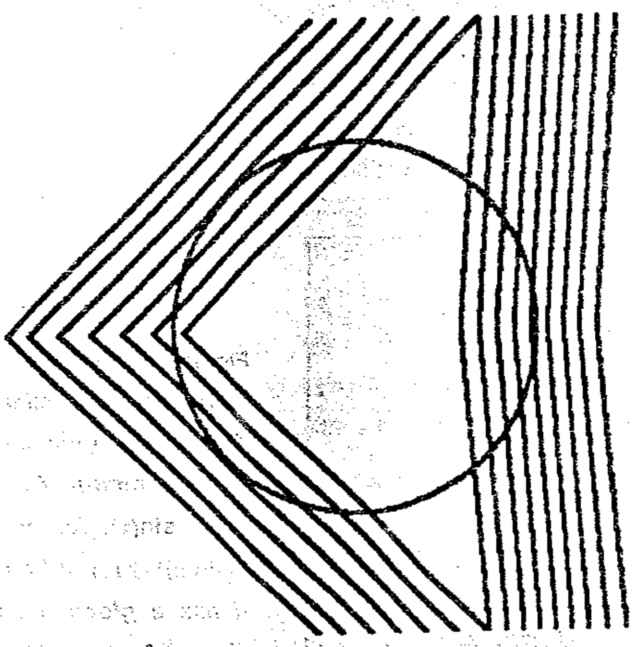
GAs are computationally simple, they have demonstrated their power and capability in optimising multi-modal, multi-dimensional and multi-objective problems. Hence they should find widespread applications in business, science and engineering. The price paid for global optimisation and robustness is the amount of computation required, which is not a problem in the age of fast/parallel computers. GAs use simple computational operations and yet are powerful tools for optimisation. Several versions/improvements in GAs are now emerging.

Address for correspondence
J R Raol
Scientist, FMCD
National Aeronautical Laboratory
Bangalore 560 017

Suggested Reading

D E Goldberg. Generic Algorithms in Search, Optimisation and Machine Learning, Addison-Wesley Publishing Company, Inc. 1989.

Zolliner illusion



Cross hatching has resulted in an illusion of distortion in a perfect circle.

G S Ranganath

678