9 Genetic Algorithms

"Genetic algorithms are good at taking large, potentially huge search spaces and navigating them, looking for optimal combinations of things, solutions you might not otherwise find in a lifetime."

Salvatore Mangano
Computer Design, May 1995

9.1 Introduction

Origin with a protozoa (prime unicell animal) to existence of human (most developed living being) in nature as a result of evolution, is the main theme, adopted by genetic algorithms (GA), one of the most modern paradigm for general problem solving. Since the paradigm simulates the strategy of evolution, it is surprisingly simple but powerful, domain free approach to problem solving. GAs are gaining popularity in many engineering and scientific applications due to their enormous advantages such as adaptability, ability to handle non-linear, ill defined and probabilistic problems. As the approach is domain free, it has wide scope of applications and most of the optimization problems can be handled successfully with this approach.

The emergence of massively parallel computers made these algorithms of practical interest. There are various well known programs in this class like evolutionary programs, genetic algorithms, simulated annealing, classifier systems, expert systems, artificial neural networks and fuzzy systems. This chapter discusses a genetic algorithm – which is based on the principle of evolution (survival of fittest). In such algorithms a population of individuals (potential solution) undergoes a sequence of transformations like mutation type and crossover type. These individuals strive for survival; a selection scheme, biased towards fitter individuals, selects the next generation. After some number of generations the program converges to the optimal value.

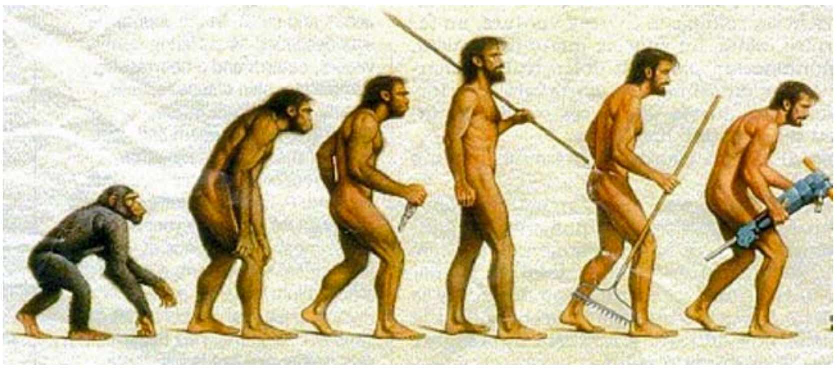
Genetic algorithm has been applied to various problems in electrical power systems such as generation scheduling (Orero and Irving 1996, 1996a, 1998;

Huang 1998), Economic load dispatch (Song and Xuan 1998), reactive power optimization (Iba 1994), distribution network planning (Miranda et al. 1994), alarm Processing (Wen et al. 1998), Electrical long term load-forecasting (Chaturvedi et al. 1995) and optimal control problems. Genetic algorithms are more robust than existing directed search methods. Another important property of GA based search methods is that they maintain population of potential solutions – all other methods process a single point of the search space like hill climbing method. Hill climbing methods provide local optimum values and these values depend on the selection of starting point. Also there is no information available on the relative error with respect to global optimum. To increase the success rate in hill climbing method, it is executed for large number of randomly selected different starting points. On the other hand, GA is a multi-directional search maintaining a population of potential solutions and encourages information formation and exchange between these directions. The population undergoes a simulated evolution and at each generation the relatively good solution reproduce, while the relatively bad solutions die out. To distinguish between different solutions we use an objective function which plays the role of an environment.

9.2 History of Genetics

Genetics is a science which deals with the transfer of biological information from generation to generation. Genetics deals with the physical and chemical nature of these informations itself. Geneticists are concerned with whys and hows of these transfer of biological information, which is the basis for certain differences and similarities that are recognized in a group of living organisms. What is the source of genetic variations? How are difference distributed in populations? Why not all variations among living things however are inherited? All these are concern with genetics.

Long before human began to wonder about genetic mechanism, they already operating effectively in nature. Population of plants and animals are now known to have built in potentials for consistency and change that are dependent on genetics. Change that are established through these mechanism over long period of time in a population of living things is called EVOLUTION.



Many potential changes have been accomplished by human interventions in genetic mechanism that now accrue to benefit human beings. By selective breading, domesticated organisms have been made to serve human society increasingly better. Improve quantity and quality of milk, eggs, meat, wool, maize, wheat, rice, cotton and many other sources of food, fiber and shelter at least to the success of human intervention in genetic mechanism.

The mechanism of genetics is entirely based on gene. The gene concept however, had been implicit in model's visualization of a physical element or factor that acts as the foundation of development of a trait. He first postulated the existence of genes from their end effects, as expressed in altered characteristics. The word "allelmorph", shortened to "allele" is used to identify the member of paired genes that control different alternative traits. The gene is characterized as an individual unit of genetic mechanism. Genes replicate themselves and reproduce chromosomes, cells and organisms of their own kind. Gene is the part of chromosome. Some chromosomal genes work together, each making a small contribution to height, weight or intelligence, etc. Genes not only have a basic role in the origin and life of individual organisms, but they also, through variation in gene frequencies, cause change in populations.

Let us have a quick look at the brief history of genetics:

"The fundamental principle of natural selection as the main evolutionary principle long before the discovery of genetic mechanism has been presented by C. Darwin. He hypothesized fusion or blending inheritance, supposing that parental qualities mix together.

This theory was first time objected by Jenkins. He mentioned that there is no selection in homogenous populations. It is simply a nightmare called Jenkins nightmare.

In 1865, Gregor Johann Mandel discovered the basic principles of transference of hereditary factors from parents to offspring and explained the Jenkins nightmare. The Danish biologist Wilhelm Johannsen called these factors genes. It is now known as the genes not only transmitted hereditary traits but also mastermind the entire process of life. The genes are located in the chromosome (thread-like bodies) which are themselves situated in the nucleus of the cell. They are always found in pairs. Chromosomes vary in number according to species. The fruitfly, for example, has 4 pairs or 8 chromosomes in all, and the garden pea has 7 pairs (14 in all), mice have 20 pairs (Lawrence 1991) and humans 23 pairs (Brest et al. 2006).

Genetics was fully developed by Morgan and his collaborators. They proved experimentally that chromosomes are the main carriers of hereditary information, which later proved that Mendelian laws to be valid for all sexually reproducing organisms.

1920s Cetverikov proved that Mendel's genetics and Darwin's theory of natural selection are in no way conflicting and that their marriage yields modern evolutionary theory.

Prof. John Holland of the University of Michigan, Ann Arbor envisaged the concepts of GA algorithms and published a seminal work (Holland 1975).

There after, number of other researchers (Davis 1991; Goldberg and Holland 1989; Michalewiccz 1992) contributed to developing and improve the original GA.

9.3 Genetic Algorithms

The beginnings of genetic algorithms can be traced back to the early 1950s when several biologists used computers for simulations of biological systems (Goldberg and Holland 1989). However, the work done in late 1960s and early 1970s at the University of Michigan under the guidance of John Holland led to genetic algorithms as they are known today. GA vocabulary is being borrowed from natural genetics. The idea behind genetic algorithms is to do what nature does. Genetic algorithms (GAs) are stochastic algorithms whose search methods inspired from phenomena found in living nature. The phenomena incorporated so far in GA models include phenomena of natural selection as there are selection and the production of variation by means of recombination and mutation, and rarely inversion, diploid and others. Most Genetic algorithms work with one large panmictic population, i.e. in the recombination step each individual may potentially choose any other individual from the population as a mate. Then GA operators are performed to obtain the new child offspring; the operators are:

i. Selection
ii. Crossover,
iii. Mutation,
iv. Survival of fittest (Heistermann 1990; Michalewiccz 1992; Muzhlenbein 1989; Holland 1973; Nowack and Schuster 1989).

9.3.1 Selection

As in natural surroundings it holds on average: “the better the parents, the better the offsprings” and “the offspring is similar to the parents”. Therefore, it is on the one hand desirable to choose the fittest individuals more often, but on the other hand not too often, because otherwise the diversity of the search space decreases (Braun 1990). GA researchers have developed a variety of selection algorithms to make it more efficient and robust. In the implementation of genetic algorithm the best individuals have been select using roulette wheel with slot sized according to fitness, so that the probability of selection of best strings are more as shown in Fig. 9.1a. Besides the roulette wheel selection, researchers have developed a variety of selection algorithms like proportionate selection, linear rank selection, tournament selection and stochastic remainder selection.

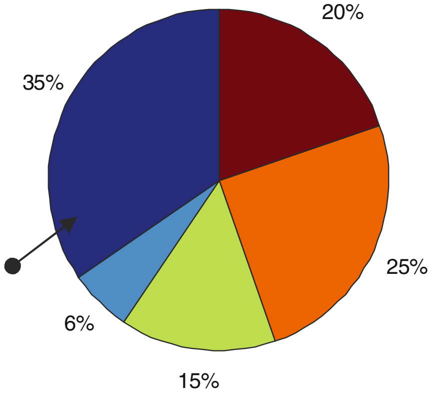


Fig. 9.1a. Roulette Wheel Selection

9.3.1.1 Roulette Wheel Selection

In roulette selection chromosomes are selected based on their fitness relative to all other chromosomes in the population as shown in Fig. 9.1a. One disadvantage of using roulette wheel is that its selective pressure reduces as population converges upon a solution, which reduces the convergence rate and may not allow finding the better solutions.

% Matlab sub-routine for roulette wheel selection

```
function x=roul(num_to_be_sel,popsize,pop,f);
         f=f/sum(f);
         roul_wheel(1)=f(1);
         for i=2:popsize
                roul_wheel(i)=roul_wheel(i-1)+f(i);
         end;
% create n=num_to_be_sel random numbers for roulette
           selection : r(i)
         r=rand(num_to_be_sel,1);
% determine selected strings according to roul_wheel :
sel_str(popsize,lchrom)
          for i=1:num_to_be_sel
            flag=1;
            for j=1:popsize
                    if (r(i)<=roul_wheel(j) & flag==1)
                        x(i,:)=pop(j,:);
                        flag=0;
                    end;
                end;
          end;
```

9.3.1.2 Tournament Selection

In this process of selection, one parent is selected by randomly comparing other individual in the same population and select with the best fitness. To select the second parent the same process is repeated. It is most popular selection method due to its simplicity (Baker 1987).

9.3.1.3 Linear Rank Selection

In this method the individuals are ordered according to their fitness values (Grefenstette 1986). The individuals of highest fitness are kept on the top and worst on the bottom. Then each individual in the population is assigned a subjective fitness based on linear ranking function as

f(r)=(popsize-rank)(max-min)/(Popsize-1)+min
where popsize – population size
rank – rank in the current population
max, min – maximum and minimum subjective fitness determined by the user.

Now this subjective fitness value is assigned to the individual and the selection is done on the basis of roulette wheel spinning. In this selection process the selective pressure is constant and does not change with generation to generation. However, in this process, it is necessary to sort the population according to their fitness values and the individuals of same fitness will not have the same probability of being selected.

9.3.2 Crossover

Obviously, selection alone can not generate better offsprings. To produce better new off springs a crossover operator is required. A crossover operator can be termed loosely as recombination or slice-exchange-merge operator. The most common type of crossover operator mentioned above is called single point crossover. In this operation select two parents and randomly selects a point between two genes to cut both chromosomes into two parts. This point is called crossover point. In crossover operation combine the first part of first parent and second part of second parent to get first offspring. Similarly, combine the first part of second parent and second part of first parent to get second offspring. These offsprings belong to the next population. The crossover operator has three distinct substeps:

a. Slice each of the parent strings in two substrings.
b. Exchange a pair of corresponding substrings of parents.
c. Merge the two respective substrings to form offsprings.

For example, suppose following two binary strings are mated together and undergoes the crossover operation. The strings are 1100000, 0101111. By a random choice the crossover site is fixed at 3 which is shown by a vertical bar. Then the effect of cross over will be as shown in Fig. 9.1b.

To increase the speed of convergence of GA, the population is divided from the middle and two halves (subgroups) are used in group cross over as shown in Fig. 9.1 c. Another type of crossover is multi-point cross over, in which two or more than two sites have been selected and exchange have been done as illustrated in Fig. 9.1d.

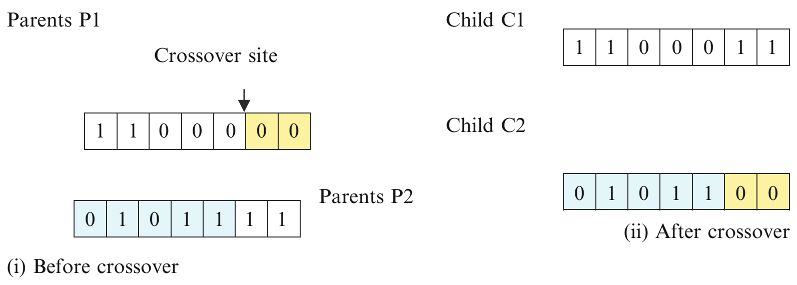


Fig. 9.1b. Single point cross over operation with two strings

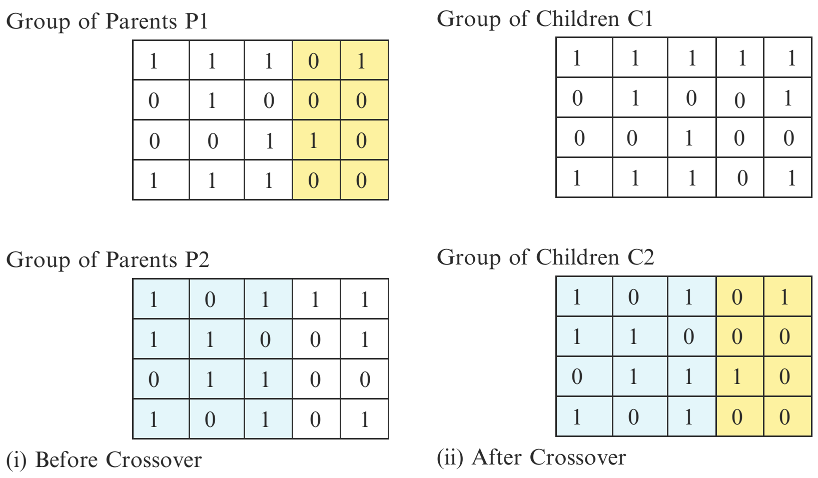


Fig. 9.1c. Single point Group Cross over operation

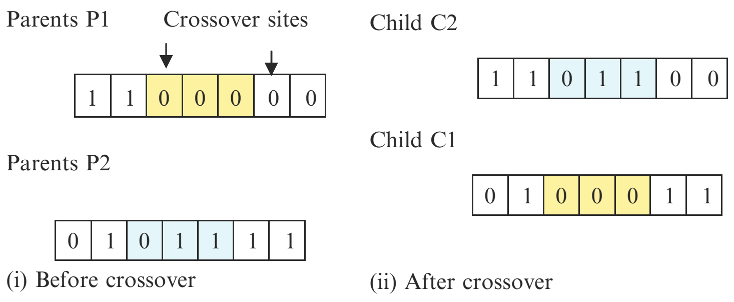


Fig. 9.1d. Multipoint crossover

% Matlab code for single point crossover operation

```
j2=2*i;
j1=j2-1;
a=rand(1);                    % Random number generation.
site=round(a*(lchrom-2)+1); % Random selection of crossover site
temp=sel_str(j1,site:lchrom);   % lchrom - length of cromosome
sel_str(j1,site:lchrom)=sel_str(j2,site:lchrom);
sel_str(j2,site:lchrom)=temp;
```

9.3.3 Mutation

The newly created individuals have no new inheritance information and the number of alleles is constantly decreasing. This process results in the contraction of the population to one point, which is only wished at the end of the convergence process, after the population works in a very promising part of the search space. Diversity is necessary to search a big part of the search space. It is one goal of the learning algorithm to search always in regions not viewed before. Therefore, it is necessary to enlarge the information contained in the population. One way to achieve this goal is mutation. The mutation operator M (chromosome) selects a gene of that chromosome and changes the allele by an amount called the mutation variance (mv), this happens with a mutation frequency (mf). The parameter mutation variance and mutation frequency have a major influence on the quality of learning algorithms. For binary coded GAs mutation is equivalent of flipping a bit at any particular position. Since, mutation is to be used sparingly its probability is very low. The mutation operation may be shown as in Fig. 9.1e. The group mutation and multipoint mutation may also be performed to improve the results.

% Matlab code for mutation operation

```
% sel_str is now the intermediate pop for mutation
for i=1:popsize
        for j=1:lchrom
                if (flip(pmute)==1)
                        if(sel_str(i,j)==0)
                                sel_str(i,j)=1;
                        else
                                sel_str(i,j)=0;
                        end;
                end;
        end;
end;
function y=flip(prob)
        a=rand(1);
```

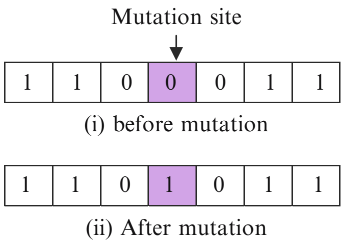


Fig. 9.1e. Single point mutation operation

```
if (a <=prob)
        y=1;
else
        y=0;
end;
```

9.3.4 Survival of Fittest

Further more we only accept an offspring as a new member of population, if it differ enough from the other individuals, that means here its fitness differ from all other individuals at least by some significant amount. After accepting a new individual we remove one of the worst individual (i.e. its fitness value is quite low) from the population in order to hold the population size constant.

To maximize the efficiency of GAs, three inherent parameters of GAs are to be optimized, the mutation probability **Pm**, the crossover probability **Pc**, and the population size **POPSIZE**. For GA parameter optimization several results have been obtained over the last few years. DeJong and Schuster proposed heuristics for an optimal setting of the mutation probability **Pm** (Nowack and Schuster 1989; Schuster 1985), Fogarty and Booker investigated time dependencies of the mutation and the crossover probability respectively (Fogarty 1989), Greffenstette Schaffer and Jong found optimal settings for all three parameters of the GA by experiment (Greffensette 1986; Schaffer et al. 1989; De Jong and Spears 1990). The brief description of these parameters are given below:

Duplicates

Individuals that represent the same candidate solution are known as duplicate individuals. It has been mentioned (Davis 1991) that eliminating duplicates increases the efficiency of a genetic search and reduces the danger of premature convergence.

9.3.5 Population Size

A group of individuals (chromosome) collectively comprise is known as population. Population size is the number of individuals (chromosome) in the population maintained by a GA. As discussed by De Jong and Spears (1990) [30]

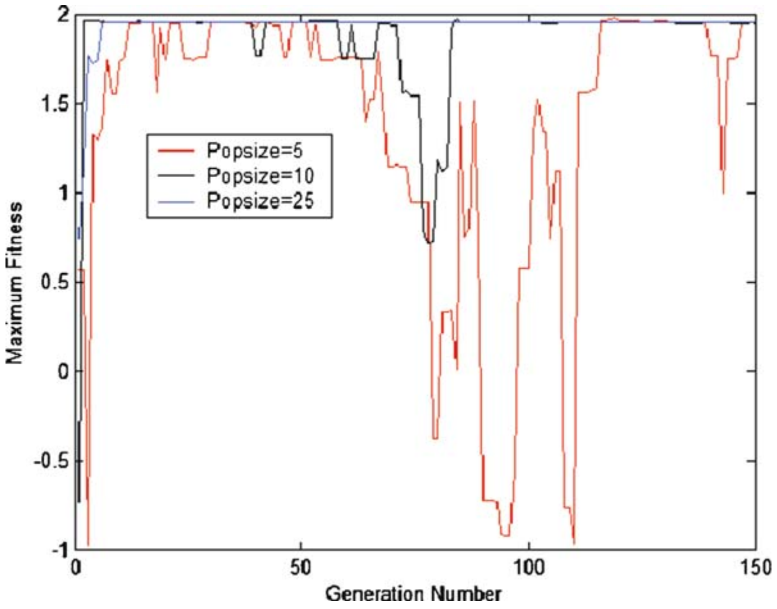


Fig. 9.2. Effect of population size on maximum fitness

that the choice of population size has a strong interacting effect on the results. Smaller population size tends to become homogeneous more quickly and there is a danger of premature convergence upon a suboptimal solution. With large population size the crossover productivity effect is much less dramatic, hence takes longer time to converge upon a solutions.

Usually the population size for GA varying from tens to thousands and it is noted that this parameter is mostly problem dependent. If the problem in hand is simpler then smaller population size can also serve the purpose, but if the problem is complex, large population size is required and it is also necessary to run for large number of generations.

The effect of population size on maximum fitness value of GA is shown in Fig. 9.2. Form the figure it is clear that the GA performance is good for population size 50, 80 and 100. The optimal performance of GA is obtained at popsize equals to 50. The average fitness is also compared for different population sizes as shown in Fig. 9.3.

9.3.6 Evaluation of Fitness Function

The evaluation function of a GA is used to determine the fitness of chromosomes in the population. The binary coded chromosomes also known as a genotype. To find the fitness of binary coded chromosomes, they must be decoded first and then evaluated the fitness. But in case of real coded chromosome which is also called as phenotype and for them, there is no need of decoding is required.

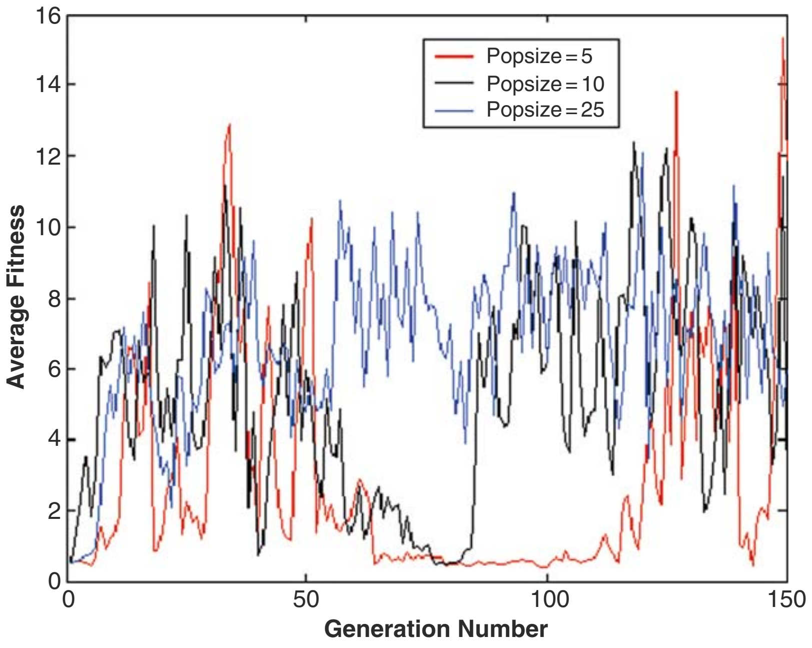


Fig. 9.3. Effect of population size on average fitness

9.4 Effect of Crossover Probability on GA Performance

For better results, it is advisable to select the crossover rate quite large than mutation rate. This is the usual practice to take crossover rate 20 times greater than the mutation rate. Crossover rate generally ranging from 0.25 to 0.95. The effect of crossover probability (pcross) on GA performance in terms of average fitness is shown in Fig. 9.4.

9.5 Effect of Mutation Probability on GA Performance

Schaffer (Mbamalu and Hawary 1993) found experimentally that mutation probability (Pm) is approximately inversely proportional to the population size. Mutation rate generally varying from 0.001 to 0.05. The effect of mutation rate is shown in Fig. 9.5. If the mutation rate is high then there are more fluctuations in the fitness value. On the other hand if the mutation rate is low then it the search area is reduced. Hence, the optimal value of mutation rate is selected for good performance of GA or one can dynamically change its value.

Maximum number of generations

The selection of maximum number of generations is a problem dependent parameter. For complex problems, the maximum number of generations is large enough, so that the results should converge to optimal value (Greffensette 1986).

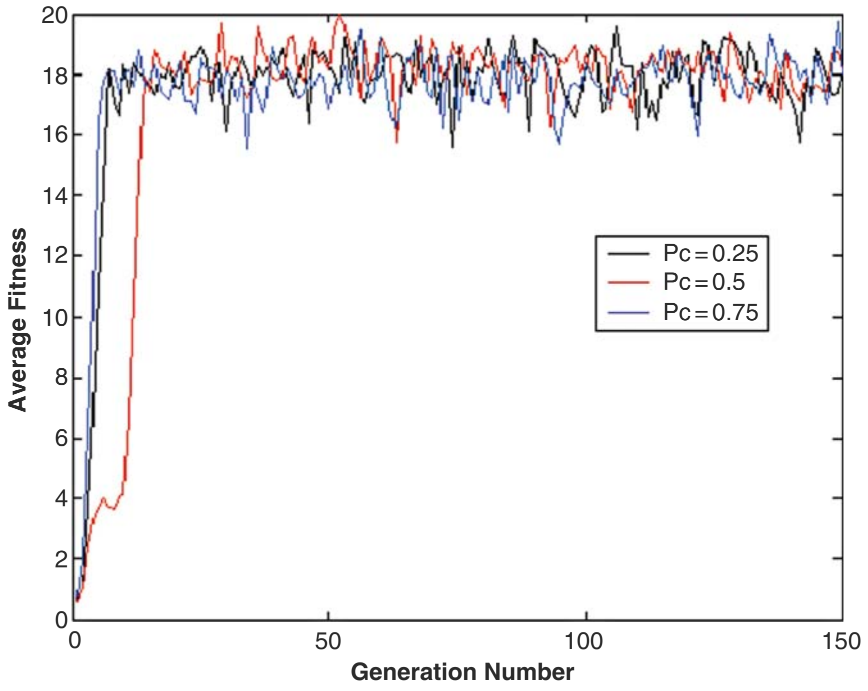


Fig. 9.4. Effect of crossover probability on average fitness

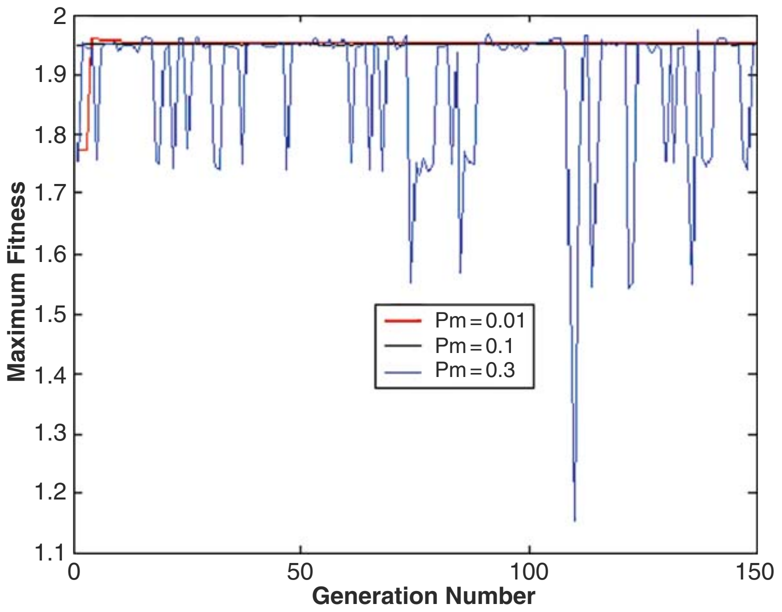


Fig. 9.5. Effect of mutation probability on maximum fitness

Length of chromosome (lchrome)

The value of lchrome is dependent to the precision required and can be calculated with the help of the following expression –

$$2^{lchrome} = (\max parm - \min parm) * 10^r$$

Where, r is number of places after decimal, up to which the precision is required.

Max parm – Upper bound of parameter
Min parm – Lower bound of parameter

9.6 Main Components of GA

A GA (or any evolutionary program) for a particular problem must have the following five components:

1. A genetic representation for potential solutions to the problem (Coding).
2. A way to create an initial population of potential solution.
3. To evaluate rank of a solution define an objective function.
4. To alter the composition of offspring's define genetic operators.
5. Define GA parameters like population size, probabilities of genetic operators, etc.

Coding

In order to solve any problem with genetic algorithms, variables are first coded in some string structures. There are some of the studies in which directly variables values are taken, but most of the GAs work with binary coded variable strings (Fig. 9.6).

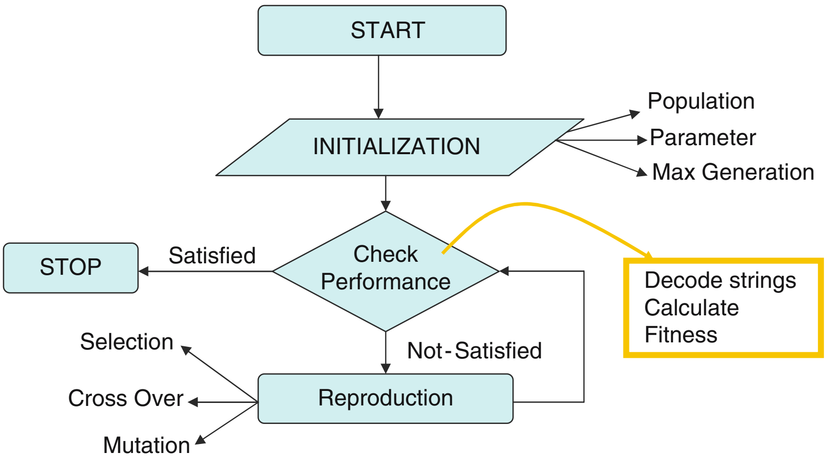


Fig. 9.6. Flow chart of simple genetic algorithm

Example 1. Minimize the surface area of a cylindrical closed end container, with the following constraints:

i. Volume is $5\,\text{m}^3$
ii. Radius of cylinder is not less than 0.5 m and not more than 1.0 m.

Solution

A. *Problem formulation*
It is a double variable optimization problem, which could be modified into a single variable optimization problem.

$$\begin{aligned}\text{Volume of container V} &= \pi\ r^2 l = 5\text{m}^3\\ L &= 5/\pi\ r^2\\ \text{Surface area A} &= 2\pi\ r^2 + 2\pi r\ l\\ &= 2\pi\ r^2 + 2\pi\ r\ (5/\pi\ r^2)\\ &= 6.28\ r^2 + 10/r\end{aligned}$$

B. *Implement GA code in MATLAB*
C. *Given*
Upper bond ub = 1.0
Lower bond lb = 0.5
D. *Data preparation*
i. Size of population popsize = 30
ii. Length of chromosome lchrom is determined as:

$$2^{\text{lchrom}} \geq (\text{ub} - \text{lb}) * 10^{\text{dp}}$$

If the precision required upto four places after decimal dp = 4.
Hence, lchrom = 13
iii. Since it is a minimization problem and genetic algorithms evaluate the strings on the basis of fitness function, which is maximization function.
Objective function = C − A.
Where C is a constant, whose value is more than the area A.

$$= 50 - \text{A}$$

iv. Number of maximum generation = 10
v. Cross over probability Pc = 0.5
vi. Mutation probability Pm = 0.01
E. Generate initial population

Generate a random binary matrix of size (popsize × lchrom (= 30×13)).

F. Determine the fitness value for each chromosome in the population.
G. If the fitness is equal to some specified value then stop, otherwise perform GA operations (Table 9.1).

Table 9.1. Fitness values in different generations

Generation number	Maximum Fitness value	Minimum Fitness value
0	33.8156	33.2806
1	33.8158	33.3760
2	33.8158	33.3855
3	33.8158	33.4126
4	33.8158	33.4576
5	33.8158	33.4853
6	33.8158	33.5147
7	33.8158	33.6076
8	33.8159	33.5559
9	33.8159	33.6155
10	33.8159	33.6349

Overall maximum fitness value = 33.8159

Hence, the minimum surface area A = 50 − Maximum fitness = 50 − 33.819 = 16.1841 m^2

Alternate method

The calculus method may also be used to solve the above mentioned problem, because there is one variable in the objective function. The function which is to be minimized is

$$\text{Surface area } A = 6.28r^2 + 10/r$$

Differentiate the ara A with respect to r and equate it to zero.

$$\begin{aligned}&\partial A/\partial r = 0\\&6.28 * 2\,r - 10/r^2 = 0\\&r^3 = 10/(6.28 * 2) = 5/6.28\\&r = 0.9268\end{aligned}$$

Substitute the value of r in the expression of surface area.

$$\begin{aligned}A_{\min} &= 6.28 * (0.9268)^2 + (10/0.9268)\\&= 16.1840\,\mathrm{m}^2\end{aligned}$$

Hence it is very clear that the GA results are quite close to the results calculated from the calculus method. The calculus method is good for small size problems, but if the problem size is large and complex or large number of variables. Then calculus method may give good results.

9.7 Variants

The simplest algorithm represents each chromosome as a bit string. Typically, numeric parameters can be represented by integers, though it is possible to use floating point representations. The floating point representation is natural to evolution strategies and evolutionary programming. The notion of real-valued genetic algorithms has been offered but is really a misnomer because it does not really represent the building block theory that was proposed by Holland in the 1970s. This theory is not without support though, based on theoretical and experimental results. The basic algorithm performs crossover and mutation at the bit level. Other variants treat the chromosome as a list of numbers which are indexes into an instruction table, nodes in a linked list, hashes, objects, or any other imaginable data structure. Crossover and mutation are performed so as to respect data element boundaries. For most data types, specific variation operators can be designed. Different chromosomal data types seem to work better or worse for different specific problem domains.

When bit strings representations of integers are used, gray coding is often employed. In this way, small changes in the integer can be readily effected through mutations or crossovers. This has been found to help prevent premature convergence at so called *Hamming walls*, in which too many simultaneous mutations (or crossover events) must occur in order to change the chromosome to a better solution.

Other approaches involve using arrays of real-valued numbers instead of bit strings to represent chromosomes. Theoretically, the smaller the alphabet, the better the performance, but paradoxically, good results have been obtained from using real-valued chromosomes. A very successful (slight) variant of the general process of constructing a new population is to allow some of the better organisms from the current generation to carry over to the next, unaltered. This strategy is known as *elitist selection*.

Parallel implementations of genetic algorithms come in two flavours. Coarse grained parallel genetic algorithms assume a population on each of the computer nodes and migration of individuals among the nodes. Fine grained parallel genetic algorithms assume an individual on each processor node which acts with neighboring individuals for selection and reproduction. Other variants, like genetic algorithms for online optimization problems, introduce time-dependence or noise in the fitness function.

It can be quite effective to combine GA with other optimization methods. GA tends to be quite good at finding generally good global solutions, but quite inefficient at finding the last few mutations to find the absolute optimum. Other techniques (such as simple hill climbing) are quite efficient at finding absolute optimum in a limited region. Alternating GA and hill climbing can improve the efficiency of GA while overcoming the lack of robustness of hill climbing.

A problem that seems to be overlooked by GA-algorithms thus far is that the natural evolution maximizes mean fitness rather than the fitness of the individual (the criterion function used in most applications).

An algorithm that maximizes mean fitness (without any need for the definition of mean fitness as a criterion function) is Gaussian adaptation, provided that the ontogeny of an individual may be seen as a modified recapitulation of evolutionary random steps in the past and that the sum of many random steps tend to become Gaussian distributed (according to the central limit theorem).

This means that the rules of genetic variation may have a different meaning in the natural case. For instance – provided that steps are stored in consecutive order – crossing over may sum a number of steps from maternal DNA adding a number of steps from paternal DNA and so on. This is like adding vectors that more probably may follow a ridge in the phenotypic landscape. Thus, the efficiency of the process may be increased by many orders of magnitude. Moreover, the inversion operator has the opportunity to place steps in consecutive order or any other suitable order in favour of survival or efficiency. (See for instance (Mahalanbis et al. 1991)).

Gaussian adaptation is able to approximate the natural process by an adaptation of the moment matrix of the Gaussian. Gaussian adaptation may serve as a genetic algorithm replacing the rules of genetic variation by a Gaussian random number generator working on the phenotypic level. Population-based incremental learning is a variation where the population as a whole is evolved rather than its individual members.

9.8 Applications of Genetic Algorithms

GA is not only used for solving optimization problems, but there are number of GA applications as mentioned below:

1. Industrial design by parameterization
2. Scheduling problems such as manufacturing, facility scheduling, allocation of resources, etc.
3. System design
4. Time series prediction
5. Data base mining
6. Control system
7. Artificial life system
8. Various medical applications, such as image segmentation and modeling
9. Combinatorial optimization problems like travelling sales man problem, routing, bin packing, graph partitioning and colouring.
10. Trajectory planning of robots
11. Game playing like chase playing, prisoner's dilemma, etc.
12. Resource allocation problem
13. Graph colouring and partitioning, etc.

9.9 Summary

During the last few decades there has been growing interest in natural process based algorithm. In this chapter, we provided a brief introduction to the field of evolutionary computing and an overview of genetic algorithms (GAs).GA

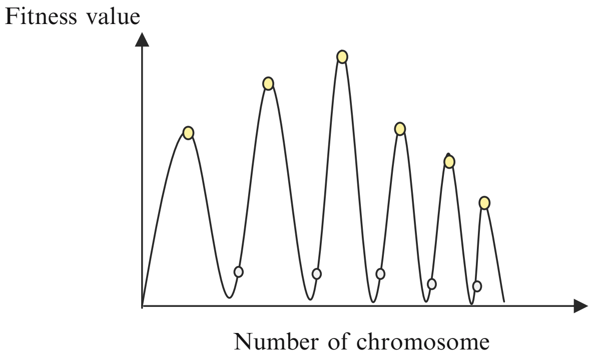


Fig. 9.7. Multiple solutions of GA for single population

describes the behaviour of genetic search. GA has a capability to provide multiple solutions for a given problem as shown in Fig. 9.7. These solutions (fitness value) improve from generation to generation.

9.10 Bibliography and Historical Notes

Some of good books on genetic algorithms are written by Goldberg and Holland (1988, 1989) and Lozano et al. 2006. The practical aspects and applications of genetic algorithms are given in Handbook of Genetic Algorithms edited by Davis (1991). The classical papers on genetic algorithms are given in Genetic Algorithms, edited by Buckles and Petry (1992) and Auger and Hansen (2005a,b) and Srinivas and Patnaik (1994a). Yaochu J. and Branke (2005) wrote a very lucid survey paper on evolutionary optimization. Albert et al. (2005) Hybrid Optimization Approach for a Fuzzy Modelled Unit Commitment Problem. Bath et al. (2007) had optimized the security constrained multi-objective optimal power dispatch.

Genetic programming (GP) is very computationally intensive and so in the 1990s it was mainly used to solve relatively simple problems. But more recently, thanks to improvements in GP technology and to the exponential growth in CPU power, GP produced many novel and outstanding results in areas such as quantum computing, electronic design (Garrison et al. 2006), game playing, sorting, searching (Smith 2002) and many more. GP has also been applied to evolvable hardware as well as computer programs (Eiben and Smith 2003; Fogel 2000; Auger and Hansen 2005a,b).

9.11 Exercises

1. What do you understand by genetic algorithms?
2. How does genetic algorithm work?
3. What do you mean by crossover and muation operations in GA. Write Matlab codes for these operations.

4. Mention different types of crossover operations and compare them.
5. Write Peuso codes for simple GA and implement simple GA using Matlab and study the effect of chromosome length, crossover and muation rates for the minimization of ocnnection length on a printed circuit board.
6. Simulated the above problem with different population size and compare the results.