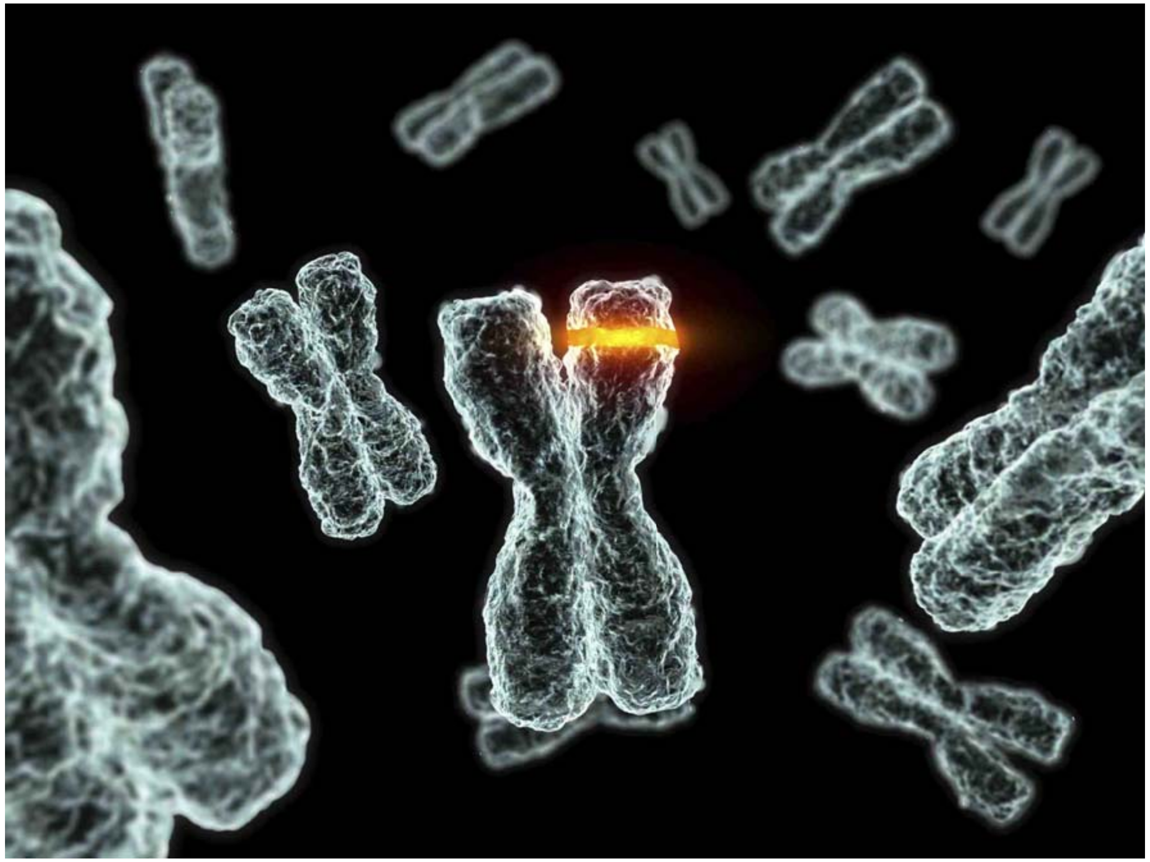
AS PER THE SYLLABUS OF BPUT FOR SEVENTH SEMESTER OF AE&IE BRANCH.

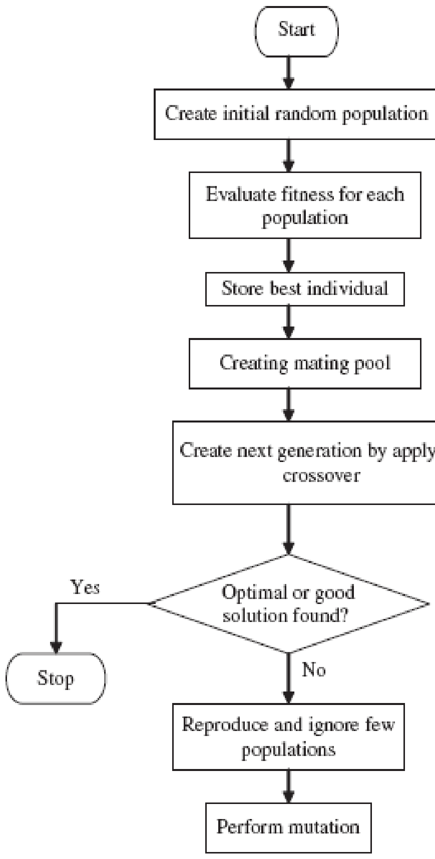
RIT, BERHAMPUR

SOFT COMPUTING (PECS 3401)-GENETIC ALGORITHM



What is Genetic Algorithm

The genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms(EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation,selection, and crossover. The main steps in GA are initialization, selection, reproduction and termination.


Start
Create initial random population
Evaluate fitness for each population
Store best individual
Creating mating pool
Create next generation by applying crossover
Optimal or good solution found?
Yes
Stop
No
Reproduce and ignore few populations
Perform mutation

Simple Generational Genetic Algorithm Pseudo Code

1. Choose the initial population of individuals
2. Evaluate the fitness of each individual in that population
3. Repeat on this generation until termination: (time limit, sufficient fitness achieved, etc.)
   i. Select the best-fit individuals for reproduction
   ii. Breed new individuals through crossover and mutation operations to give birth to offspring
   iii. Evaluate the individual fitness of new individuals
   iv. Replace least-fit population with new individuals

Advantages of Genetic Algorithm

1. Parallelism
2. Liability
3. Solution space is wider
4. The fitness landscape is complex
5. Easy to discover global optimum
6. The problem has multi objective function
7. Only uses function evaluations.
8. Easily modified for different problems.
9. Handles noisy functions well.
10. Handles large, poorly understood search spaces easily
11. Good for multi-modal problems Returns a suite of solutions.
12. Very robust to difficulties in the evaluation of the objective function.
13. They require no knowledge or gradient information about the response surface
14. Discontinuities present on the response surface have little effect on overall optimization performance
15. They are resistant to becoming trapped in local optima
16. They perform very well for large-scale optimization problems
17. Can be employed for a wide variety of optimization problems

Limitations of Genetic Algorithm

1. The problem of identifying fitness function
2. Definition of representation for the problem
3. Premature convergence occurs
4. The problem of choosing the various parameters like the size of the population,
   mutation rate, cross over rate, the selection method and its strength.
5. Cannot use gradients.
6. Cannot easily incorporate problem specific information
7. Not good at identifying local optima
8. No effective terminator.
9. Not effective for smooth unimodal functions
10. Needs to be coupled with a local search technique.
11. Have trouble finding the exact global optimum
12. Require large number of response (fitness) function evaluations
13. Configuration is not straightforward

Applications of Genetic Algorithm

1. Nonlinear dynamical systems–predicting, data analysis
2. Robot trajectory planning
3. Evolving LISP programs (genetic programming)

4. Strategy planning
5. Finding shape of protein molecules
6. TSP and sequence scheduling
7. Functions for creating images
8. Control–gas pipeline, pole balancing, missile evasion, pursuit
9. Design–semiconductor layout, aircraft design, keyboard configuration, communication networks
10. Scheduling–manufacturing, facility scheduling, resource allocation
11. Machine Learning–Designing neural networks, both architecture and weights, improving classification algorithms, classifier systems
12. Signal Processing–filter design
13. Combinatorial Optimization–set covering, traveling salesman (TSP), Sequence scheduling, routing, bin packing, graph coloring and partitioning

Biological Background

Chromosomes are the strings of DNA and serve as a model for the whole organism.
Genes are the constituent elements of the chromosomes that are responsible for encoding a particular pattern. E.g. colour of iris.
Alleles are the values of the that are assigned to genes. E.g. colour of iris is *blue*.
Locus is the position of a particular gene.
Genome is the complete set of genetic material present in the chromosome.
Genotype is a particular type of genome.

Encoding

Encoding is a process of representing individual genes. The process can be performed using bits, numbers, trees, arrays, lists or any other objects. The encoding depends mainly on solving the problem. For example, one can encode directly real or integer numbers.

Type of Encoding

Binary Encoding

Each chromosome encodes a binary (bit) string. Each bit in the string can represent some characteristics of the solution. Every bit string therefore is a solution but not necessarily the best solution. Another possibility is that the whole string can represent a number. The way bit strings can code differs from problem to problem.

Binary encoding

Chromosome 1	1 1 0 1 0 0 0 1 1 0 1 0
Chromosome 2	0 1 1 1 1 1 1 1 1 1 0 0

Binary encoding gives many possible chromosomes with a smaller number of alleles. On the other hand this encoding is not natural for many problems and sometimes corrections must be made after genetic operation is completed. Binary coded strings with 1s and 0s are mostly used. The length of the string depends on the
accuracy. In this,
• Integers are represented exactly
• Finite number of real numbers can be represented
• Number of real numbers represented increases with string length

Octal Encoding

This encoding uses string made up of octal numbers (0–7).

Chromosome 1	03467216
Chromosome 2	15723314

Octal encoding

Hexadecimal Encoding

This encoding uses string made up of hexadecimal numbers (0–9, A–F).

Chromosome 1	9CE7
Chromosome 2	3DBA

Hexadecimal encoding

Permutation Encoding (Real Number Coding)

Every chromosome is a string of numbers,which represents the number in sequence. Sometimes corrections have to be done after genetic operation is completed. In permutation encoding, every chromosome is a string of integer/real values, which represents number in a sequence.

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

Permutation encoding

Value Encoding

Every chromosome is a string of values and the values can be anything connected to the problem. This encoding produces best results for some special problems. On the other hand, it is often necessary to develop new genetic operator's specific to the problem. Direct value encoding can be used in problems, where some complicated values, such as real numbers, are used. Use of binary encoding for this type of problems would be very difficult. 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 A	1.2324 5.3243 0.4556 2.3293 2.4545
Chromosome B	ABDJEIFJDHDIERJFDLDFLFEGT
Chromosome C	(back), (back), (right), (forward), (left)

Value encoding

Value encoding is very good for some special problems. On the other hand, for this encoding is often necessary to develop some new crossover and mutation specific for the problem.

Tree Encoding

This encoding is mainly used for evolving program expressions for genetic programming. Every chromosome is a tree of some objects such as functions and commands of a programming language.

Breeding or Reproduction

The breeding process is the heart of the genetic algorithm. It is in this process, the search process creates new and hopefully fitter individuals. The breeding cycle consists of three steps:

1. Selecting parents.
2. Crossing the parents to create new individuals (offspring or children).
3. Replacing old individuals in the population with the new ones.

Selection

Selection is a method that randomly picks chromosomes out of the population according to their evaluation function. The higher the fitness function, the more chance an individual has to be selected. The selection pressure is defined as the degree to which the better individuals are favored. The higher the selection pressured, the more the better individuals are favored. This selection pressure drives the GA to improve the population fitness over the successive generations.

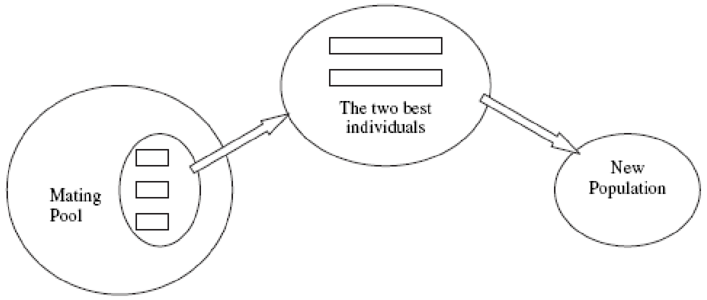


Fig. 3.10 Selection

The convergence rate of GA is largely determined by the magnitude of the selection pressure, with higher selection pressures resulting in higher convergence rates. Genetic Algorithms should be able to identify optimal or nearly optimal solutions under a wise range of selection scheme pressure. However, if the selection pressure is too low, the convergence rate will be slow, and the GA will take unnecessarily longer time to find the optimal solution. If the selection pressure is too high, there is an increased change of the GA prematurely converging to an incorrect (sub-optimal) solution. In addition to providing selection pressure, selection schemes should also preserve population diversity, as this helps to avoid premature convergence.

The various selection methods are discussed as follows:

Roulette Wheel Selection

Roulette selection is one of the traditional GA selection techniques. The commonly used reproduction operator is the proportionate reproductive operator where a string is selected from the mating pool with a probability proportional to the fitness. The principle of roulette selection is a linear search through a roulette wheel with the slots in the wheel weighted in proportion to the individual's fitness values.

The expected value of an individual is that fitness divided by the actual fitness of the population. Each individual is assigned a slice of the roulette wheel, the size of the slice being proportional to the individual's fitness. The wheel is spun N times, where N is the number of individuals in the population. On each spin, the individual under the wheel's marker is selected to be in the pool of parents for the next generation. This method is implemented as follows:

1. Sum the total expected value of the individuals in the population. Let it be T.
2. Repeat N times:
   a. Choose a random integer 'r' between o and T.
   b. Loop through the individuals in the population, summing the expected values, until the sum is greater than or equal to 'r'. The individual whose expected value puts the sum over this limit is the one selected.

Roulette wheel selection is easier to implement but is noisy. The rate of evolution depends on the variance of fitness's in the population.

Random Selection

This technique randomly selects a parent from the population. In terms of disruption of genetic codes, random selection is a little more disruptive, on average, than roulette wheel selection.

Rank Selection

The Roulette wheel will have a problem when the fitness values differ very much. If the best chromosome fitness is 90%, its circumference occupies 90% of Roulette wheel, and then other chromosomes have too few chances to be selected. Rank Selection ranks the population and every chromosome receives fitness from the ranking. The worst has fitness 1 and the best has fitness N. It results in slow convergence but prevents too quick convergence. It also keeps up selection pressure when the fitness variance is low. It preserves diversity and hence leads to a successful search. In effect, potential parents are selected and a tournament is held to decide which of the individuals will be the parent. There are many ways this can be achieved and two suggestions are,

1. Select a pair of individuals at random. Generate a random number, *R*, between 0 and 1. If *R* < *r* use the first individual as a parent. If the *R*>=*r* then use the second individual as the parent. This is repeated to select the second parent. The value of *r* is a parameter to this method.
2. Select two individuals at random. The individual with the highest evaluation becomes the parent. Repeat to find a second parent.

Tournament Selection

An ideal selection strategy should be such that it is able to adjust its selective pressure and population diversity so as to fine-tune GA search performance. Unlike, the Roulette wheel selection, the tournament selection strategy provides selective pressure by holding a tournament competition among Nu individuals.
The best individual from the tournament is the one with the highest fitness, which is the winner of Nu. Tournament competitions and the winner are then inserted into the mating pool. The tournament competition is repeated until the mating pool for generating new offspring is filled. The mating pool comprising of the tournament winner has higher average population fitness. The fitness difference provides the selection pressure, which drives GA to improve the fitness of the succeeding genes. This method is more efficient and leads to an optimal solution.

Boltzmann Selection

Simulation annealing is a method of function minimization or maximization. This method simulates the process of slow cooling of molten metal to achieve the minimum function value in a minimization problem. Controlling a temperature like parameter introduced with the concept of Boltzmann probability distribution simulates the cooling phenomenon.
Let fmax be the fitness of the currently available best string. If the next string has fitness f(Xi) such that f(Xi)>fmax, then the new string is selected. Otherwise it is selected with Boltz Mann probability, $P = \exp[-(fmax-f(Xi))/T]$
Where $T = To(1-\alpha)k$ and $k = (1 + 100*g/G)$; g is the current generation number; G, the maximum value of g. The value of α can be chosen from the range [0, 1] and To from the range [5, 100]. The final state is reached when computation approaches zero value of T, i.e., the global solution is achieved at this point. The probability that the best string is selected and introduced into the mating pool is very high.

Elitism

The first best chromosome or the few best chromosomes are copied to the new population. The rest is done in a classical way. Such individuals can be lost if they are not selected to reproduce or if crossover or mutation destroys them. This significantly improves the GA's performance.

Stochastic Universal Selection

Stochastic universal sampling provides zero bias and minimum spread. The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness exactly as in roulette-wheel selection. Here equally spaced pointers are placed over the line, as many as there are individuals to be selected. Consider *NPointer* the number of individuals to be selected, then the distance between the pointers are 1/*NPointer* and the position of the first pointer is given by a randomly generated number in the range [0, 1/*NPointer*].

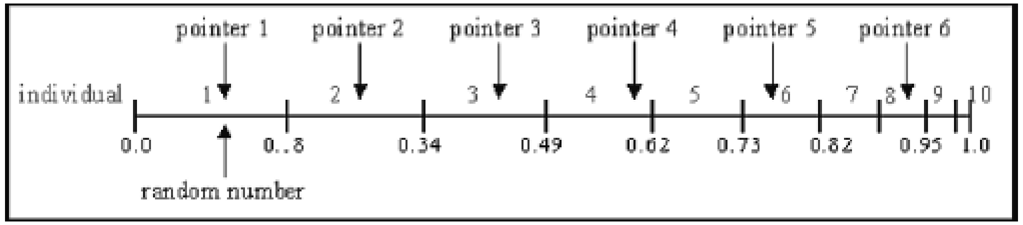


Fig. 3.11 Stochastic universal sampling

For 6 individuals to be selected, the distance between the pointers is 1/6=0.167. Figure 3.11 shows the selection for the above example. Sample of 1 random number in the range [0, 0.167]: 0.1. After selection the mating population consists of the individuals, 1, 2, 3, 4, 6, 8. Stochastic universal sampling ensures a selection of offspring, which is closer to what is deserved than roulette wheel selection.

Crossover (Recombination)

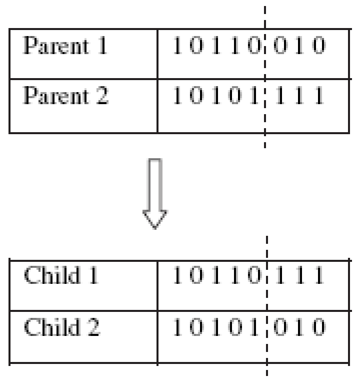
Crossover is the process of taking two parent solutions and producing from them a child. After the selection (reproduction) process, the population is enriched with better individuals. Reproduction makes clones of good strings but does not create new ones. Crossover operator is applied to the mating pool with the hope that it creates a better offspring. Crossover is a recombination operator that proceeds in three steps:

1. The reproduction operator selects at random a pair of two individual strings for the mating.
2. A cross site is selected at random along the string length.
3. Finally, the position values are swapped between the two strings following the cross site.

That is, the simplest way how to do that is to choose randomly some crossover point and copy everything before this point from the first parent and then copy everything after the crossover point from the other parent. The various crossover techniques are discussed as follows:

Single Point Crossover

The traditional genetic algorithm uses single point crossover, where the two mating chromosomes are cut once at corresponding points and the sections after the cuts exchanged. Here, a cross-site or crossover point is selected randomly along the length of the mated strings and bits next to the cross-sites are exchanged. If appropriate site is chosen, better children can be obtained by combining good parents else it severely hampers string quality.



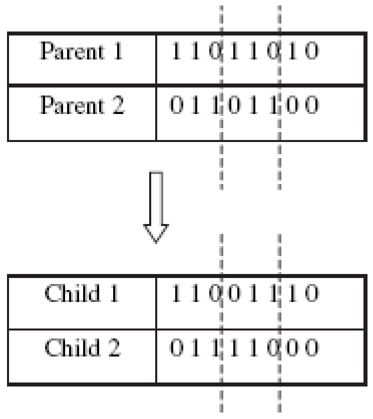
Parent 1	1 0 1 1 0 0 1 0
Parent 2	1 0 1 0 1 1 1 1

Child 1	1 0 1 1 0 1 1 1
Child 2	1 0 1 0 1 0 1 0

The above Fig. 3.12 illustrates single point crossover and it can be observed that the bits next to the crossover point are exchanged to produce children. The crossover point can be chosen randomly.

Two Point Crossover

Apart from single point crossover, many different crossover algorithms have been devised, often involving more than one cut point. It should be noted that adding further crossover points reduces the performance of the GA. The problem with adding additional crossover points is that building blocks are more likely to be disrupted. However, an advantage of having more crossover points is that the problem space may be searched more thoroughly.



Parent 1	1 1 0 1 1 0 1 0
Parent 2	0 1 1 0 1 1 0 0

Child 1	1 1 0 0 1 1 1 0
Child 2	0 1 1 1 1 0 0 0

In two-point crossover, two crossover points are chosen and the contents between these points are exchanged between two mated parents.

Multi-Point Crossover (N-Point crossover)

There are two ways in this crossover.One is even number of cross-sites and the other odd number of cross-sites. In the case of even number of cross-sites, cross-sites are selected randomly around a circle and information is exchanged. In the case of odd number of cross-sites, a different cross-point is always assumed at the string beginning.

Uniform Crossover

Uniform crossover is quite different from the N-point crossover. Each gene in the offspring is created by copying the corresponding gene from one or the other parent chosen according to a random generated binary crossover mask of the same length as the chromosomes. Where there is a 1 in the crossover mask, the gene is copied from the first parent, and where there is a 0 in the mask the gene is copied from the second parent. A new crossover mask is randomly generated for each pair of parents. Offsprings, therefore contain a mixture of genes from each parent. The number of effective crossing point is not fixed, but will average L/2 (where L is the chromosome length).

Parent 1	1 0 1 1 0 0 1 1
Parent 2	0 0 0 1 1 0 1 0
Mask	1 1 0 1 0 1 1 0
Child 1	1 0 0 1 1 0 1 0
Child 2	0 0 1 1 0 0 1 1

In Fig. 3.14, new children are produced using uniform crossover approach. It can be noticed, that while producing child 1, when there is a 1 in the mask, the gene is copied from the parent 1 else from the parent 2. On producing child 2, when there is a 1 in the mask, the gene is copied from parent 2, when there is a 0 in the mask; the gene is copied from the parent 1.

Three Parent Crossover

In this crossover technique, three parents are randomly chosen. Each bit of the first parent is compared with the bit of the second parent. If both are the same, the bit is taken for the offspring otherwise; the bit from the third parent is taken for the offspring. This concept is illustrated in Fig. 3.15.

Parent 1	1 1 0 1 0 0 0 1
Parent 2	0 1 1 0 1 0 0 1
Parent 3	0 1 1 0 1 1 0 0
Child	0 1 1 0 1 0 0 1

Crossover with Reduced Surrogate

The reduced surrogate operator constrains crossover to always produce new individuals wherever possible. This is implemented by restricting the location of crossover points such that crossover points only occur where gene values differ.

Shuffle Crossover

Shuffle crossover is related to uniform crossover. A single crossover position (as in single-point crossover) is selected. But before the variables are exchanged, they are randomly shuffled in both parents. After recombination, the variables in the offspring are unshuffled. This removes positional bias as the variables are randomly reassigned each time crossover is performed.

Precedence Preservative Crossover (PPX)

PPX was independently developed for vehicle routing problems by Blanton and Wainwright (1993) and for scheduling problems by Bierwirth et al. (1996). The operator passes on precedence relations of operations given in two parental permutations to one offspring at the same rate, while no new precedence relations are introduced.

Parent permutation 1	A	B	C	D	E	F
Parent permutation 2	C	A	B	F	D	E
Select parent no. (1/2)	1	2	1	1	2	2
Offspring permutation	A	C	B	D	F	E

Ordered Crossover

Ordered two-point crossover is used when the problem is of order based, for example in U-shaped assembly line balancing etc. Given two parent chromosomes, two random crossover points are selected partitioning them into a left, middle and right portion. The ordered two-point crossover behaves in the following way: child 1 inherits its left and right section from parent 1, and its middle section is determined

by the genes in the middle section of parent 1 in the order in which the values appear in parent 2. A similar process is applied to determine child 2. This is shown in Fig. 3.17

```
Parent 1 : 4  2 | 1  3 | 6  5      Child 1 : 4  2 | 3  1 | 6  5
Parent 2 : 2  3 | 1  4 | 5  6      Child 2 : 2  3 | 4  1 | 5  6
```

Partially Matched Crossover (PMX)

PMX can be applied usefully in the TSP. Indeed, TSP chromosomes are simply sequences of integers, where each integer represents a different city and the order represents the time at which a city is visited. Under this representation, known as permutation encoding, we are only interested in labels and not alleles. It may be viewed as a crossover of permutations that guarantees that all positions are found exactly once in each offspring, i.e. both offspring receive a full complement of genes, followed by the corresponding filling in of alleles from their parents. PMX proceeds as follows:

1. The two chromosomes are aligned.
2. Two crossing sites are selected uniformly at random along the strings, defining a matching section

Where, the dots mark the selected cross points.

• The matching section defines the position-wise exchanges that must take place in both parents to produce the offspring.

• The exchanges are read from the matching section of one chromosome to that of the other.

• In the example, the numbers that exchange places are 5 and 2, 6 and 3, and 7 and 10.

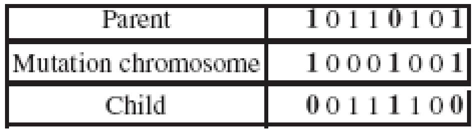
```
Name 9 8 4 . 5 6 7 . 1 3 2 1 0      Allele 1 0 1 . 0 0 1 . 1 1 0 0
Name 8 7 1 . 2 3 1 0 . 9 5 4 6      Allele 1 1 1 . 0 1 1 . 1 1 0 1

Name 9 8 4 . 2 3 1 0 . 1 6 5 7      Allele 1 0 1 . 0 1 0 . 1 0 0 1
Name 8 1 0 1 . 5 6 7 . 9 2 4 3      Allele 1 1 1 . 1 1 1 . 1 0 0 1
```

Crossover Probability

The basic parameter in crossover technique is the crossover probability (Pc). Crossover probability is a parameter to describe how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. If there is crossover, offspring are made from parts of both parent's chromosome. If crossover probability is 100%, then all offspring are 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!). Crossover is made in hope that new chromosomes will contain good parts of old chromosomes and therefore the new chromosomes will be better. However, it is good to leave some part of old population survive to next generation.

Mutation

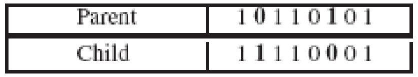
After crossover, the strings are subjected to mutation. Mutation prevents the algorithm to be trapped in a local minimum. Mutation plays the role of recovering the lost genetic materials as well as for randomly disturbing genetic information. It is an insurance policy against the irreversible loss of genetic material. Mutation has traditionally considered as a simple search operator. If crossover is supposed to exploit the current solution to find better ones, mutation is supposed to help for the exploration of the whole search space. Mutation is viewed as a background operator to maintain genetic diversity in the population. It introduces new genetic structures in the population by randomly modifying some of its building blocks. It also keeps the gene pool well stocked, and thus ensuring ergodicity. A search space is said to be ergodic if there is a non-zero probability of generating any solution from any population state. There are many different forms of mutation for the different kinds of representation. For binary representation, a simple mutation can consist in inverting the value of each gene with a small probability. The probability is usually taken about 1/L, where L is the length of the chromosome.



Parent	10110101
Mutation chromosome	10001001
Child	00111100

Flipping

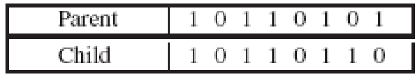
Flipping of a bit involves changing 0 to 1 and 1 to 0 based on a mutation chromosome generated. The Fig. 3.20 explains mutation-flipping concept. A parent is considered and a mutation chromosome is randomly generated. For a 1 in mutation chromosome, the corresponding bit in parent chromosome is flipped (0 to 1 and 1 to 0) and child chromosome is produced. In the above case, there occurs 1 at 3 places of mutation chromosome, the corresponding bits in parent chromosome are flipped and child is generated.



Parent	10110101
Child	11110001

Interchanging

Two random positions of the string are chosen and the bits corresponding to those positions are interchanged. This is shown in Fig. 3.21.



Parent	1 0 1 1 0 1 0 1
Child	1 0 1 1 0 1 1 0

Reversing

A random position is chosen and the bits next to that position are reversed and child chromosome is produced. This is shown in Fig. 3.22.

Mutation Probability

The important parameter in the mutation technique is the mutation probability (Pm). The mutation probability decides how often parts of chromosome will be mutated. If there is no mutation, offspring are generated immediately after crossover (or directly copied) without any change. If mutation is performed, one or more parts of a chromosome are changed. If mutation probability is 100%, whole chromosome is changed, if it is 0%, nothing is changed. Mutation generally prevents the GA from falling into local extremes. Mutation should not occur very often, because then GA will in fact change to random search.

Replacement

Replacement is the last stage of any breeding cycle. Two parents are drawn from a fixed size population, they breed two children, but not all four can return to the population, so two must be replaced i.e., once off springs are produced, a method must determine which of the current members of the population, if any, should be replaced by the new solutions. The technique used to decide which individual stay in a population and which are replaced in on a par with the selection in influencing convergence. Basically, there are two kinds of methods for maintaining the population; generational updates and steady state updates.

The basic generational update scheme consists in producing *N* children from a population of size *N* to form the population at the next time step (generation), and this new population of children completely replaces the parent selection. Clearly this kind of update implies that an individual can only reproduce with individuals from the same generation. Derived forms of generational update are also used like ($\lambda + \mu$)-update and (λ, μ)-update. This time from a parent population of size μ, a little of children is produced of size $\lambda \geq \mu$. Then the μ best individuals from either the offspring population or the combined parent and offspring populations (for (λ, μ)- and ($\lambda + \mu$)-update respectively), form the next generation. In a steady state update, new individuals are inserted in the population as soon as they are created, as opposed to the generational update where an entire new generation is produced at each time step. The insertion of a new individual usually necessitates the replacement of another population member. The individual to be deleted can be chosen as the worst member of the population. (it leads to a very strong selection pressure), or as the oldest member of the population, but those method are quite radical: Generally steady state updates use an ordinal based method for both the selection and the replacement, usually a tournament method. Tournament replacement is exactly analogous to tournament selection except the less good solutions are picked more often than the good ones. A subtile alternative is to replace the most similar member in the existing population.

Random Replacement

The children replace two randomly chosen individuals in the population. The parents are also candidates for selection. This can be useful for continuing the search in small populations, since weak individuals can be introduced into the population.

Weak Parent Replacement

In weak parent replacement, a weaker parent is replaced by a strong child.With the four individuals only the fittest two, parent or child, return to population. This process improves the overall fitness of the populationwhen paired with a selection technique that selects both fit and weak parents for crossing, but if weak individuals and discriminated against in selection the opportunity will never raise to replace them.

Both Parents

Both parents replacement is simple. The child replaces the parent. In this case, each individual only gets to breed once. As a result, the population and genetic material moves around but leads to a problem when combined with a selection technique that strongly favors fit parents: the fit breed and then are disposed of.

Maximizing the Function $f(x) = x*\sin(10*\Pi*x) + 10$

To find the solution of the functionMax $F(x) = x*\sin(10*\Pi*x) + 10$ with the constraint
$-1 < x < 2$ by using genetic algorithm.

Steps involved

```
Step1 : Generate the random number as n
Step2 : Initialize i, j to n and m respectively
Step3 : Max←1, x[i]←0, sum←0, m_max←1
Step4 : Compute x[i]←x[i]+(pp[i][j]/pow(10,j-1)) and
fx[i]=x[i]*sin(10*_*x[i])+10
sum=sum+fx[i];
Step5 : if (max>=fx[i])
Step6 : max←fx[i];
Step7 : until m_max>max
Step8 : Compute Maximum value
```

Maximize $f(x) = x2$

A C++ Program for maximizing $f(x) = x2$ using genetic algorithm, where x is
ranges from 0 to 31.

1. Generate Initial four populations of binary string with 5 bits length.
2. Calculate corresponding x and fitness value $f(x) = x2$.
3. Use the tournament selection method to generate new four populations.
4. Apply cross-over operator to the new four populations and generate new populations.
5. Apply mutation operator for each population.
6. Repeat the step 2 to 5 for some 20 iterations.
7. Finally print the result.