1. Introduction

Genetic algorithms work on two types of spaces alternatively: coding space and solution space, or in other words, genotype space and phenotype space. Genetic operators (crossover and mutation) work on genotype space, while evolution and selection work on phenotype space. The selection is the link between chromosomes and the performance of decoded solutions. The mapping from genotype space to phenotype space has a considerable influence on the performance of genetic algorithms.

The genetic algorithms provide a directed random search in complex landscapes. There are two important issues with respect to search strategies: exploration (investigate new and unknown areas in search space) and exploitation (make use of knowledge of solutions previously found in search space to help in find better solutions). This can be done by making genetic operators perform essentially a blind search; with a hope that selection operators direct the genetic search toward the desirable area of solution space.

One general principle for developing an implementation of genetic algorithms for a particular real word problem is to make a good balance between exploration and exploitation of the search space. To achieve this, all the operators and parameters of the genetic algorithms must be examined carefully. Addition heuristics should be incorporated in the algorithm to enhance the performance.

2. Genetic Algorithms

Genetic Algorithms (or simply GAs) are powerful and widely applicable stochastic search and optimization methods based on the concepts of natural selection and natural evaluation.

GAs work on a population of individuals represents candidate solutions to the optimization problem. These individual are consists of a strings (called chromosomes) of genes. The genes are a practical allele (gene could be a bit, an integer number, a real value or an alphabet character,…,etc depending on the nature of the problem). GAs applying the principles of survival of the fittest , selection , reproduction , crossover (recombining) , and mutation on these individuals to get , hopefully , a new butter individuals (new solutions) .

GAs are applied for those problems which either can not be formulated in exact and accurate mathematical forms and may contain noisy or irregular data or it take so much time to solve or it is simply impossible to solve by the traditional computational methods.

3. How Genetic Algorithms Work

Genetic algorithm maintains a population of individuals, say **P(t)**, for generation **t**. Each individual represents a potential solution to the problem at hand. Each individual is evaluated to give some measure of its fitness. Some individuals undergo stochastic transformations by means of genetic operations to form new individuals. There are two type of transformation:-

1) Mutation, which creates new individuals by making changes in a single individual.

2) Crossover, which creates new individuals by combining parts from two individuals.

The new individuals, called offspring **C(t)**, are then evaluated. A new population is formed by selecting the more fit individuals from the parent population and offspring population.

After several generations, genetic algorithm converges to the best individual, which hopefully represents an optimal or suboptimal solution to the problem. The general structure of the Genetic algorithms is as follow:

```
Begin
  {
  t=0;
  Initialize P(t);
  Evaluate P(t);
  While (not termination condition) do
  Begin
      {
      Apply crossover and mutation to P(t) to yield C(t);
      Evaluate C(t);
      Select P(t+1) from P(t) and C(t);
       t=t+1;
       }
  End
  }
End
```

The flowchart explains how genetic algorithms work is showing in the figure (1).

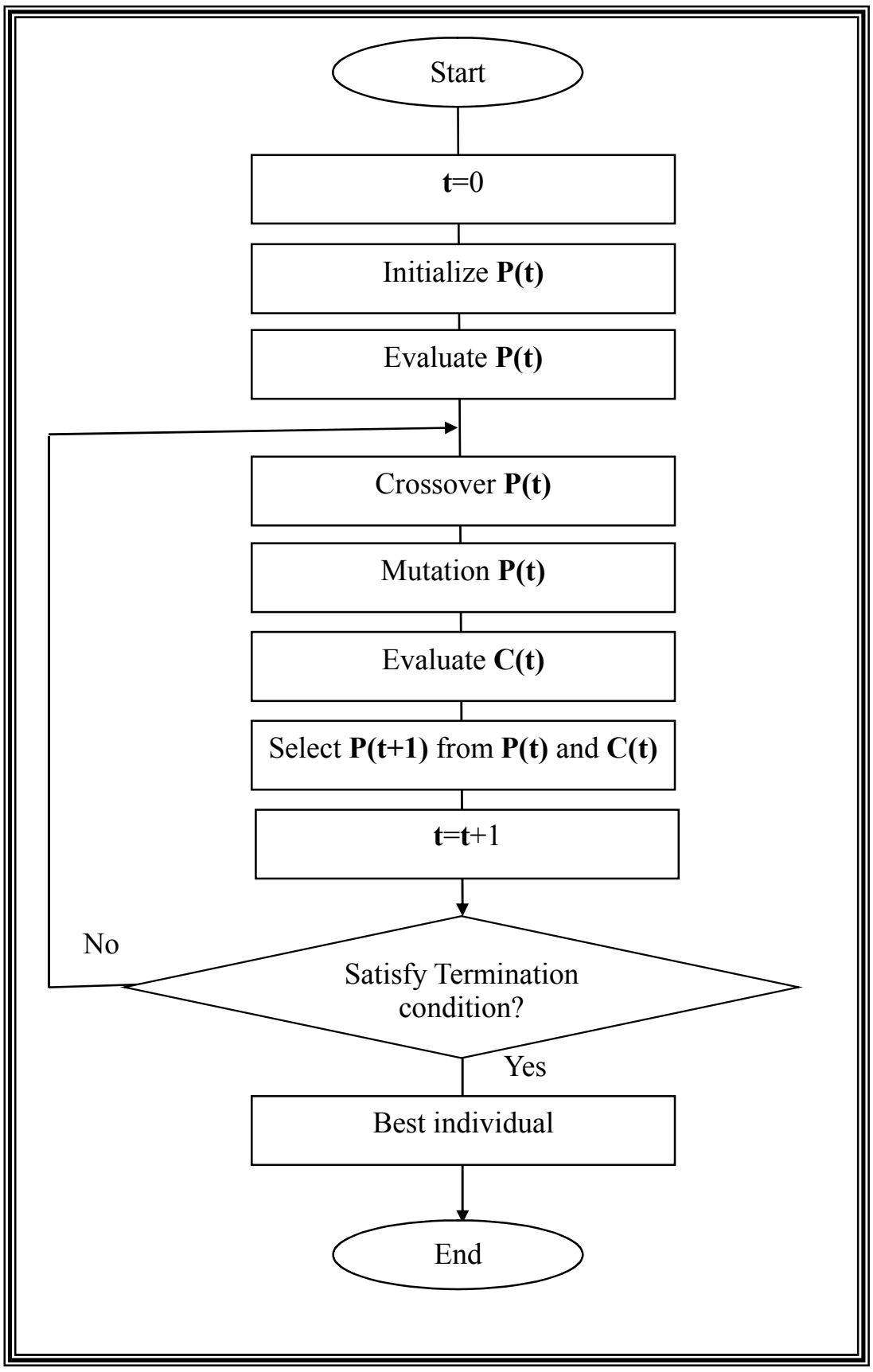


Figure (1) Flowchart explains the general structure of the genetic algorithms.

4. Encoding

How to encode the solutions of the problem into chromosomes is a key issue when using genetic algorithms.

One outstanding problem associated with encoding is that some individuals correspond to infeasible or illegal solutions to a given problem. This may become very severe for constrained optimization problems and combinatorial optimization problems.

It must be distinguished between two concepts: infeasibility and illegality, as shown in figure (2).

Infeasibility refers to the phenomenon that a solution decoded from chromosome lies outside the feasible region of given problem. Penalty methods can be used handle infeasible chromosomes [19]. One of these methods is by force genetic algorithms to approach optimal form both sides of feasible and infeasible regions.

Illegality refers to the phenomenon that a chromosome does not represent a solution to a given problem. Repair techniques are usually adopted to convert an illegal chromosome to legal one.

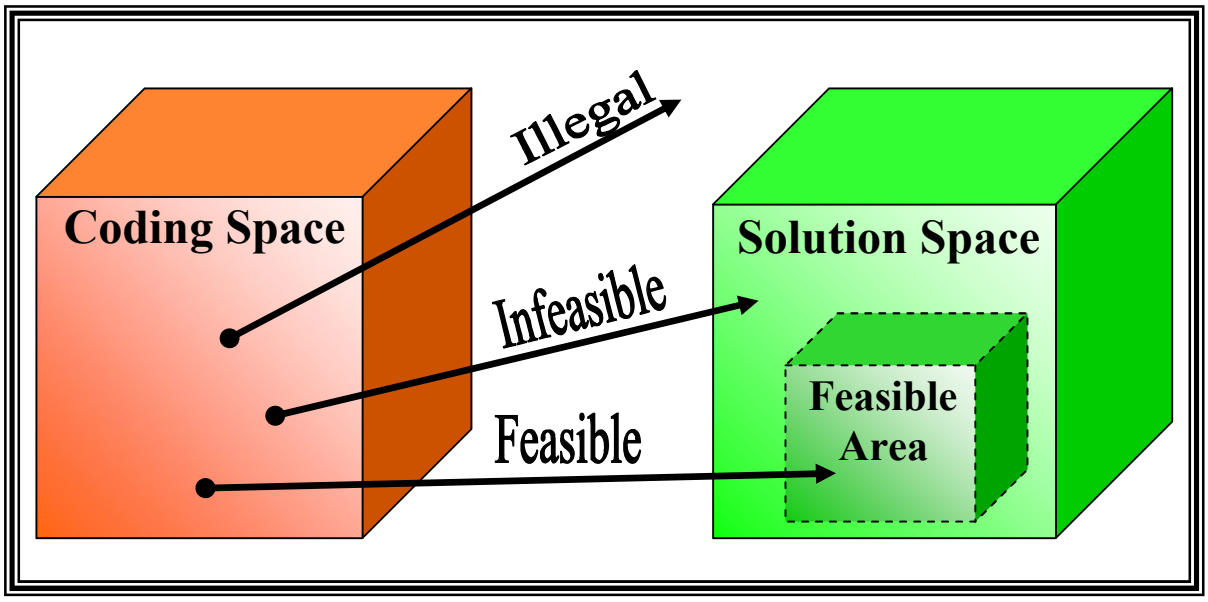


Figure (2) Infeasibility and illegality.

Various encoding methods have been created for particular problems to provide effective implementation of genetic algorithms. According to what kind of symbol is used as the alleles of a gene, the encoding methods can be classified as follows:

1) Binary encoding
2) Real-number encoding
3) Integer or literal permutation encoding

4.1 Binary Encoding

Binary encoding (i.e., the bit strings) are the most common encoding used for several of reasons. One is historical: in their earlier work, Holland and his students concentrated on such encodings and genetic algorithms practices have tended to follow this lead. Another reason for that was because much of existing GAs theories is based on the assumption of using binary encoding.

In spite of all that, binary encoding for function optimization problems is know to severe drawbacks due to the existence of Hamming cliffs, pairs of encoding having a large Hamming distance (The Hamming distance between two bit strings is defined as the number of corresponding positions in these bit strings where the bits have a different value) while belonging to points of minimal distance in phenotype space. For example, the pair 01111111111 and 10000000000 belongs to neighboring points in phenotype space but have maximum Hamming distance in genotype space. To cross the Hamming cliff, all bits have to be changed simultaneously. The probability that crossover and mutation will occur can be very small. In this sense, the binary code doses not preserve the locality of points in the phenotype space.

For many problems in the industrial engineering world, it is nearly impossible to represent their solution with binary encoding.

4.2 Real Number Encoding

Real number encoding is best used for function optimization problems. It has been widely confirmed that real number encoding perform better than binary encoding for function optimization and constrained optimizations problems. In real number encoding, the structure of genotype space is identical to that of the phenotype. Therefore, it is easy to form effective genetic operators by borrowing useful techniques from conventional methods.

4.3 Integer or Literal Permutation Encoding

Integer or literal permutation encoding is best used for combinational optimization problems because the essence of this kind of problems is to search for the best permutation or combination of items subject to constrains.

5. Genetic Algorithms Operators

There are two basic genetic algorithms operators which are crossover and mutation. These two operators are work together to explore and exploit the search space by creating new variants in the chromosomes. There are many empirical studies on a comparison between crossover and mutation. It is confirmed that mutation operator play the same important role as that of the crossover.

5.1 Crossover

One of the unique aspects of the work involving genetic algorithms (GAs) is the important role that Crossover (recombination) plays in the design and implementation of robust evolutionary systems. In most GAs, individuals are represented by fixed-length strings and crossover operates on pairs of individuals (parents) to produce new strings (offspring) by exchanging segments from the parents' strings. Traditionally, the number of crossover points (which determines how many segments are exchanged) has been fixed at a very low constant value of 1 or 2. Support for this decision came from early work of both a theoretical and empirical nature by Holland. In spite of this, other GAs problems were implemented using other types of crossover.

5.1.1 Single Point Crossover

A commonly used method for crossover is called single point crossover. In this method, a single point crossover position (called cut-point) is chosen at random (e.g., between the 4th and 5th variables) and the parts of two parents after the crossover position are exchanged to form two offspring [1, 24], as shown in figure (3).

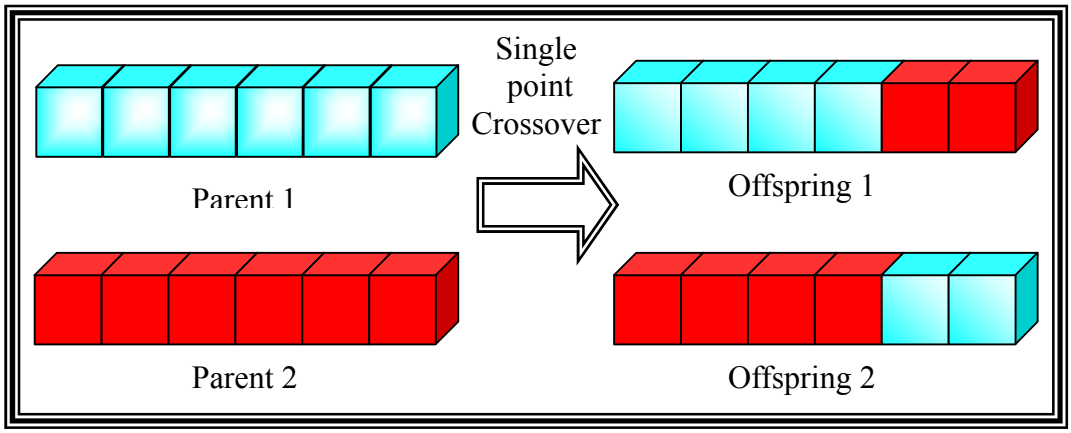


Figure (3) Single point crossover.

5.1.2 Multi Point Crossover

Multi-point crossover is a generalization of single point crossover, introducing a higher number of cut-points. In this case multi positions are chosen at random and the segments between them are exchanged [1, 24], as shown in figure (4).

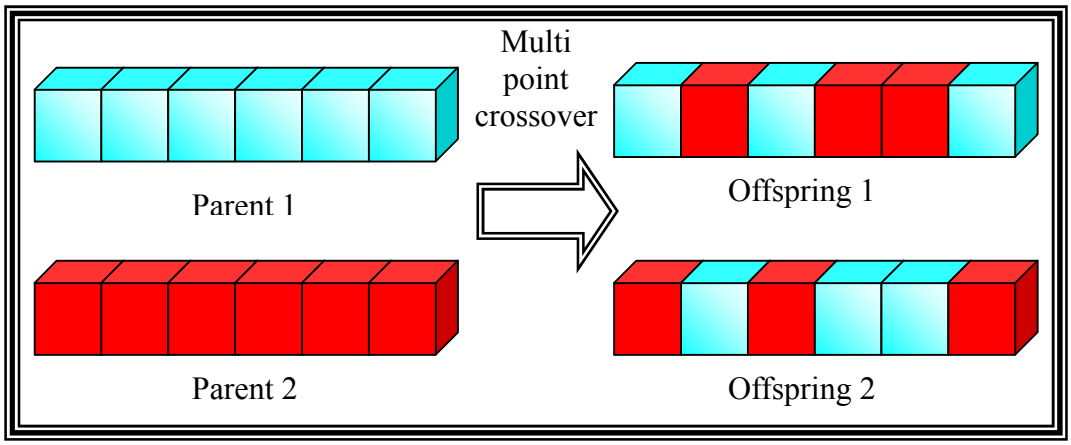


Figure (4) Multi point crossover.

5.1.3 Uniform Crossover

Uniform crossover does not use cut-points, but simply uses a global parameter to indicate the likelihood that each variable should be exchanged between two parents , as shown in figure (5).

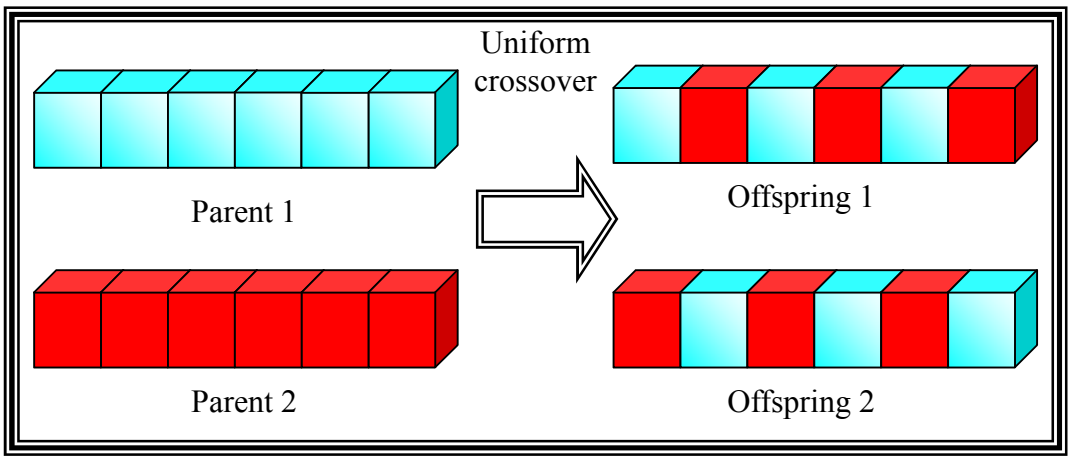


Figure (5) Uniform crossover.

5.2 Mutation

Mutation is a common operator used to help preserve diversity in the population by finding new points in the search pace to evaluate. When a chromosome is chosen for mutation, a random change is made to the values of some locations in the chromosome.

A commonly used method for mutation is called single point mutation. Though, a special mutation types used for varies problem kinds and encoding methods.

5.2.1 Single Point Mutation

Single gene (chromosome or even individual) is randomly selected to be mutated and its value is changed depending on the encoding type used, as shown in figure (6).

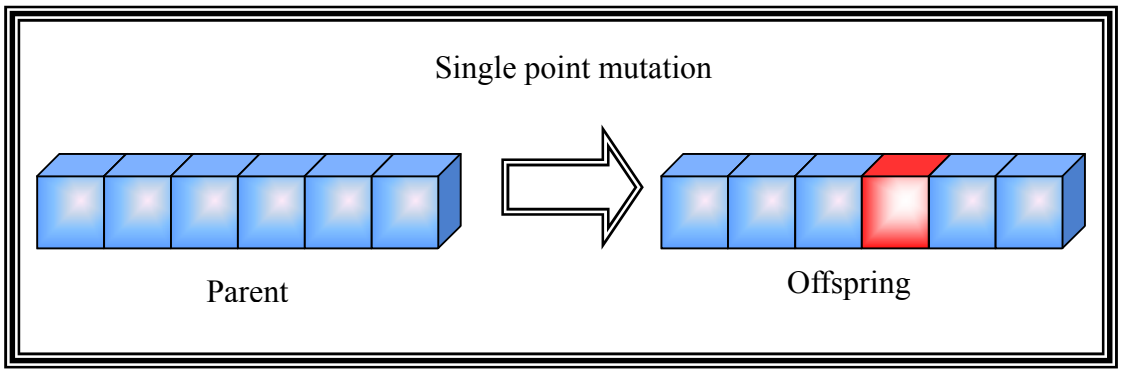


Figure (6) Single point mutation.

5.2.2 Multi Point Mutation

Multi genes (chromosomes or even individuals) are randomly selected to be mutated and there values are changed depending on the encoding type used, as shown in figure (7).

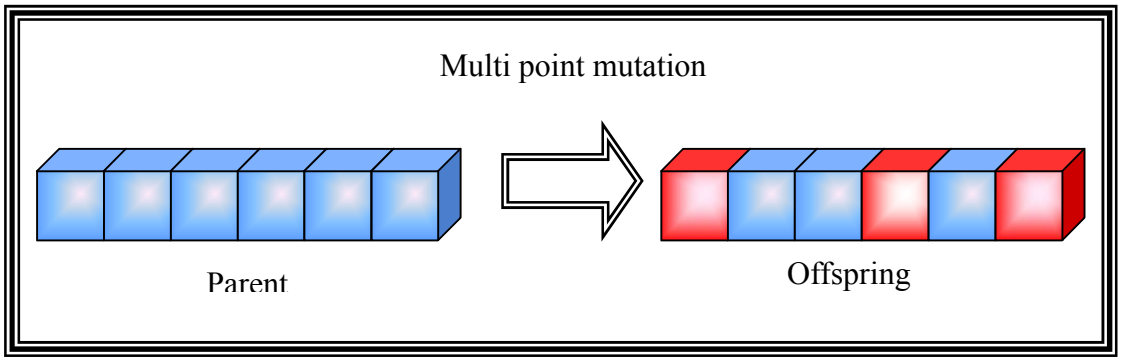


Figure (7) Multi point mutation.

6 .Selection

Selection is the process of determining the number of times a particular individual is chosen for reproduction and, thus, the number of offspring that an individual will produce. The principle behind genetic algorithms is essentially Darwinian natural selection.

Selection provides the driving force in genetic algorithms. With too much force, genetic search will terminate prematurely. While with too little force, evolutionary progress will be slower than necessary.

Typically, a lower selection pressure is indicated at the start of genetic search in favor of a wide exploration of the search space, while a higher selection pressure is recommended at the end to narrow the search space. In this way, the selection directs the genetic search toward promising regions in the search space and that will improve the performance of genetic algorithms. Many selection methods have been proposed, examined and compared. The most common types are :

1) Roulette wheel selection
2) Rank selection
3) Tournament selection
4) Steady state selection
5) Elitism

6.1 Roulette Wheel Selection

Roulette wheel selection is most common selection method used in genetic algorithms for selecting potentially useful individuals (solutions) for crossover and mutation.

In roulette wheel selection, as in all selection methods, possible solutions are assigned a fitness by the fitness function. This fitness level is used to associate a probability of selection with each individual. While candidate solutions with a higher fitness will be less likely to be eliminated, there is still a chance that they may be. With roulette wheel selection there is a chance some weaker solutions may survive the selection process; this is an advantage, as though a solution may be weak, it may include some component which could prove useful following the recombination process.

The analogy to a roulette wheel can be envisaged by imagining a roulette wheel in which each candidate solution represents a pocket on the wheel; the size of the pockets are proportionate to the probability of selection of the solution. Selecting N individual from the population is equivalent to playing N games on the roulette wheel, as each candidate is drawn independently, as shown in figure (8).

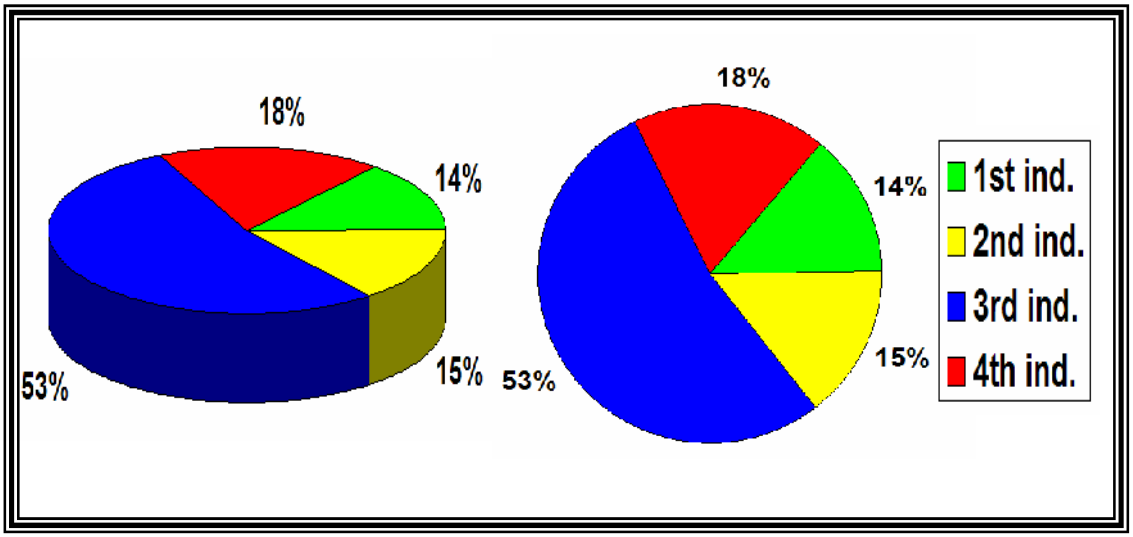
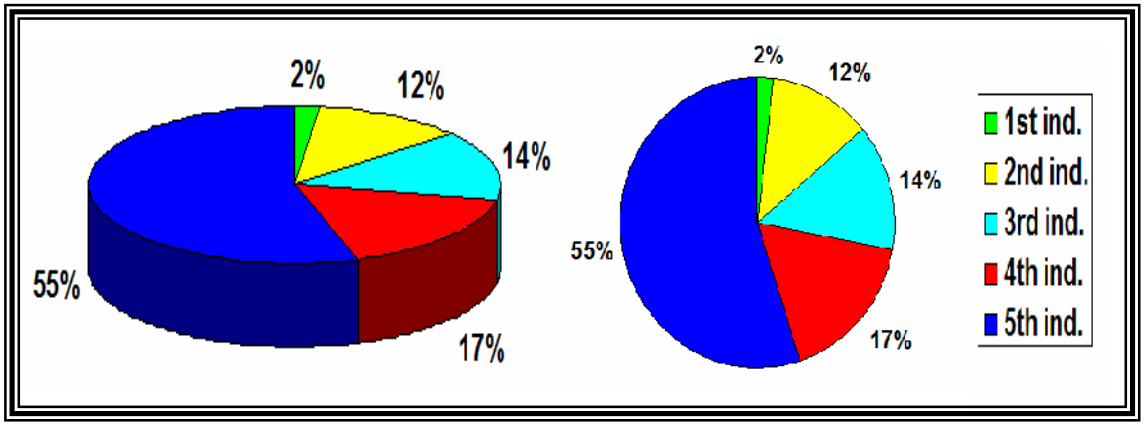


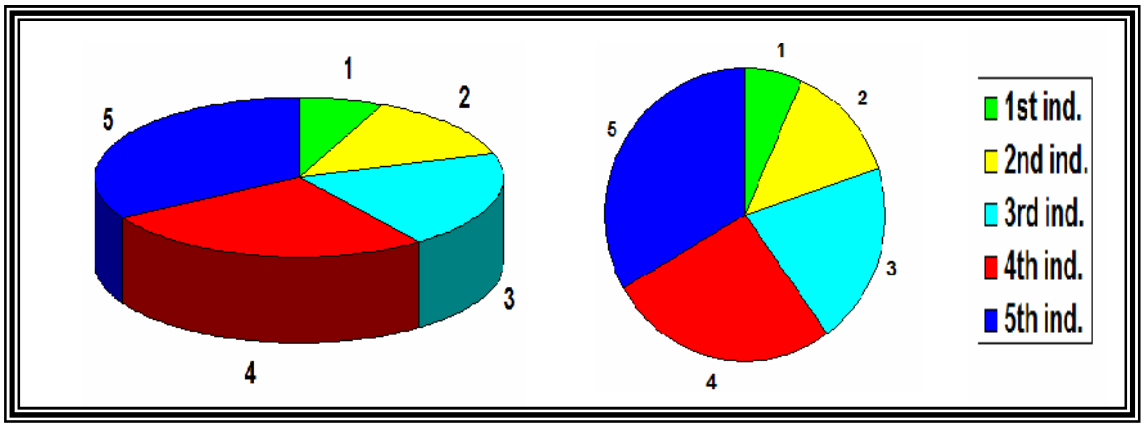
Figure (8) Roulette wheel selection.

6.2 Rank Selection

In ranking selection, the individuals in the population are sorted from best to worst according to their fitness values. Each individual in the population is assigned a numerical rank based on fitness, and selection is based on this ranking rather than differences in fitness. The advantage of this method is that it can prevent very fit individuals from gaining dominance early at the expense of less fit ones, which would reduce the population's genetic diversity and might hinder attempts to find an acceptable solution. The disadvantage of this method is that it required sorting the entire population by rank which is a potentially time consuming procedure. Rank selection effect is shown in figure (9) (a and b).



(a)



(b)

Figure (9) Rank selection effect. (a) Before ranking. (b) After ranking.

6.3 Tournament Selection

This method randomly chooses a set of individual and picks out the best individual for reproduction. The number of individual in the set is called the tournament size. A common tournament size is 2, this is called binary tournament.

By adjusting tournament size, the selection pressure can be made arbitrarily large or small. For example, using large Tournament size has the effect of increasing the selection pressure, since below average individuals are less likely to win a tournament while above average individuals are more likely to win it.

6.4 Steady State Selection

The steady state selection will eliminate the worst of individuals in each generation. It work as follow; the offspring of the individuals selected from each generation go back into the pre-existing population, replacing some of the less fit members of the previous generation .

6.5 Elitism

Elitism is an addition to many selection method that force genetic algorithms to retain some number of the best individual at each generation. It improves the selection process and save the best individuals.

With elitist selection, the quality of the best solution in each generation monotonically increases over time. Without elitist selection, it is possible to lose the best individuals due to stochastic errors (due to crossover, mutation or selection pressure).

7. Genetic Algorithms Parameters

One of the more challenging aspects of using genetic algorithms is to choose the configuration parameter settings. Discussion of GA theory provides little guidance for proper selection of the settings. The population size, the mutation rate, and the type of recombination have the largest effect on search performance. They are used to control the run of a GA. They can influence the Population and the Reproduction part of the GAs. In traditional GAs the parameters has fixed values .

Some guidelines are used in selecting these parameter settings are given in the following subsections.

7.1 Population Size

The population size is one of the most important parameters that plays a significant role in the performance of the genetic algorithms. The population size dictates the number of individuals in the population. Larger population sizes increase the amount of variation present in the initial population at the expense of requiring more fitness evaluations. It is found that the best population size is both application dependent and related to the individual size (number of chromosomes within). A good population of individuals contains a diverse selection of potential building blocks resulting in better exploration .

If the population loses diversity the population is said to have "premature convergence" and little exploration is being done. For larger individuals and challenging optimization problems, larger population sizes are needed to maintain diversity (higher diversity can also be achieved through higher mutation rates and uniform crossover) and hence better exploration. Many researchers suggest population sizes between 25

and 100 individual, while others suggest that it must be very much larger (1000 individual or more).

7.2 Crossover Rate

Crossover rate determines the probability that crossover will occur. The crossover will generate new individuals in the population by combining parts of existing individuals. The crossover rate is usually high and 'application dependent'. Many researchers suggest crossover rate to be between 0.6 and 0.95 .

7.3 Mutation Rate

Mutation rate determines the probability that a mutation will occur. Mutation is employed to give new information to the population (uncover new chromosomes) and also prevents the population from becoming saturated with similar chromosomes, simply said to avoid premature convergence. Large mutation rates increase the probability that good schemata will be destroyed, but increase population diversity. The best mutation rate is 'application dependent'. For most applications, mutation rate is between 0.001 and 0.1 [16,36], while for automated circuit design problems, it is usually between 0.3 and 0.8.

8. How Crossover and Mutation Work

Each encoding method have some different in implementation of crossover and mutation. The effect of mutation and crossover on each encoding type will be explained in the following subsections:-

8.1 Binary Encoding

1) Crossover

The single point, multi point, and uniform crossover can implement in binary encoding as explained in section (5.1) before. The effect of each one of these crossover type is shown in the figure (10) (a, b and c).

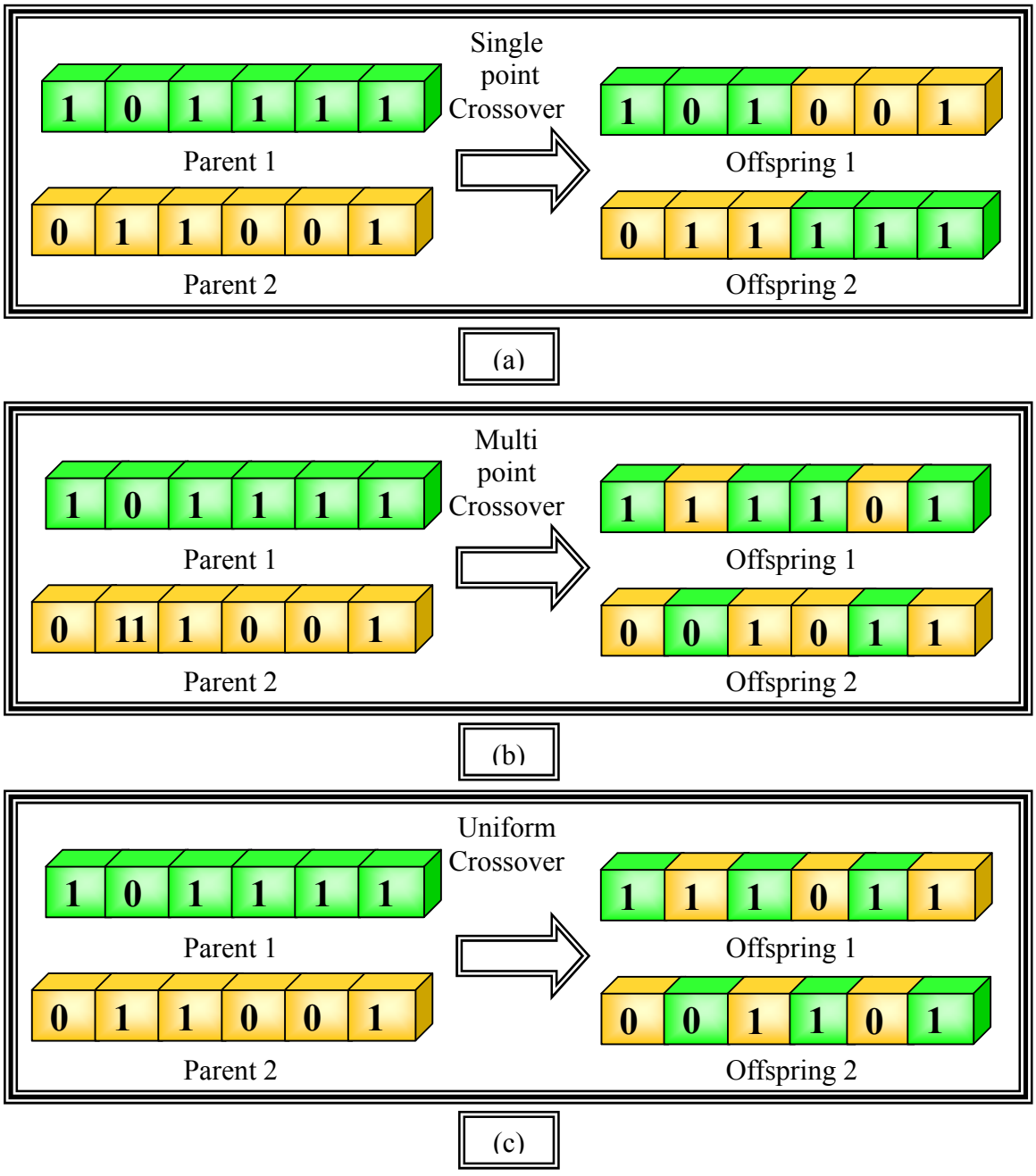


Figure (10) Explanation of crossover effect on binary string. (a) Single point crossover. (b) Multi point crossover. (c) Uniform crossover.

2) Mutation

The single point and multi point can implement in binary encoding as explained in section (5.2) before, the effect of mutation is shown in the figure (11) (a and b).

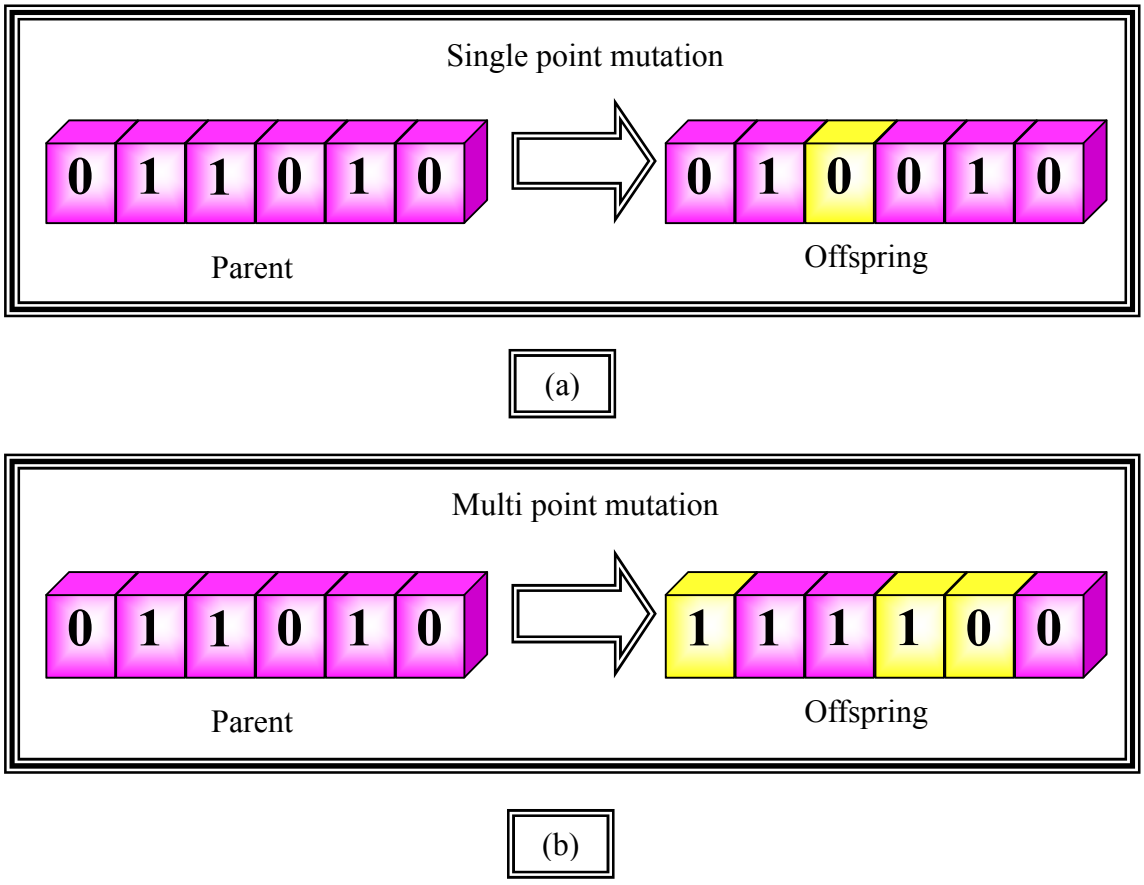


Figure (11) Explanation of mutation effect on binary string. (a) Single point mutation. (b) Multi point mutation.

8.2 Real Number Encoding

1) Crossover

Crossover in real number encoding has the same effect as in binary encoding, the single point, multi point, and uniform crossover in real number encoding is shown in figure (12)(a,b and c).

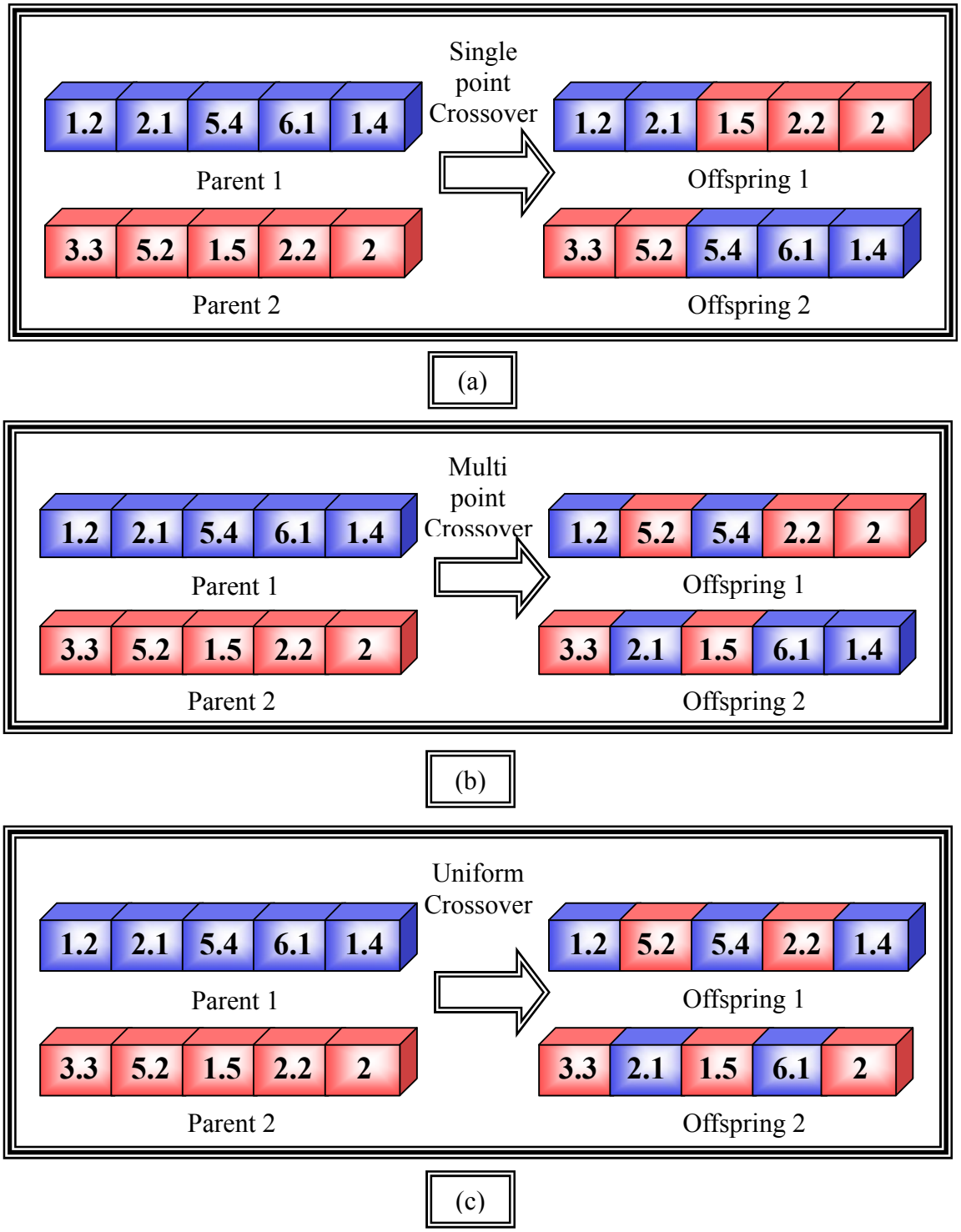


Figure (12) Explanation of the crossover effect on the real number string. (a) Single point crossover. (b) Multi point crossover. (c) Uniform crossover.

2) Mutation

In this type of encoding, there is several ways to implement mutation. This done, usually, by adding (or subtract) a random number to (or from) the mutated gene, but in another cases gene might by replaced by a new value generated randomly within the used real number limitation, as shown in figure (13) (a and b).

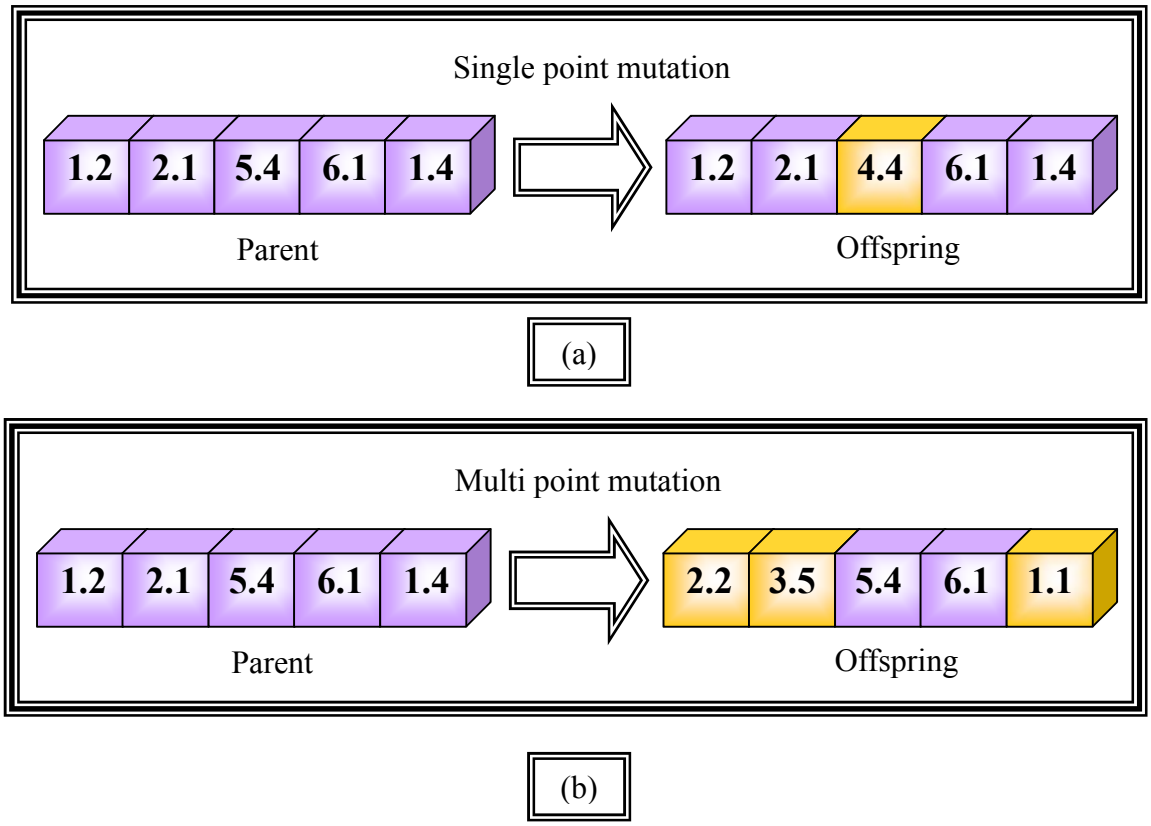


Figure (13) Explanation of the mutation effect on the real number string.
(a) Single point mutation. (b) Multi point mutation.

8.3 Integer or Literal Permutation Encoding

1) Crossover

Unlike the binary and real value encoding, this type has a special crossover rule, for single point crossover the crossover is done as follow: first copy a substring from first parent till crossover point to the first offspring, then copy all the symbols that not contents in that substring from the second parent, do the same to second parent to produce the second offspring. The same procedure can be done in multi point

crossover while uniform crossover is not usually used because it gives illegal offspring . The single point crossover is shown in figure (14).

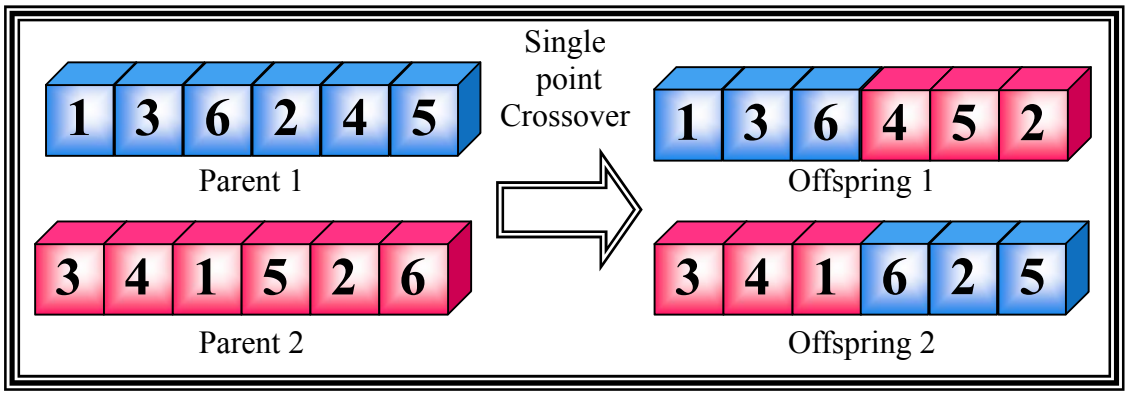


Figure (14) Explanation of single point crossover effect on Integer permutation encoding string.

2) Mutation

A special type of mutation is used, called reorder mutation, in which a pair of genes will select randomly and swap there contents. The mutation effect is shown in figure (15) (a and b).

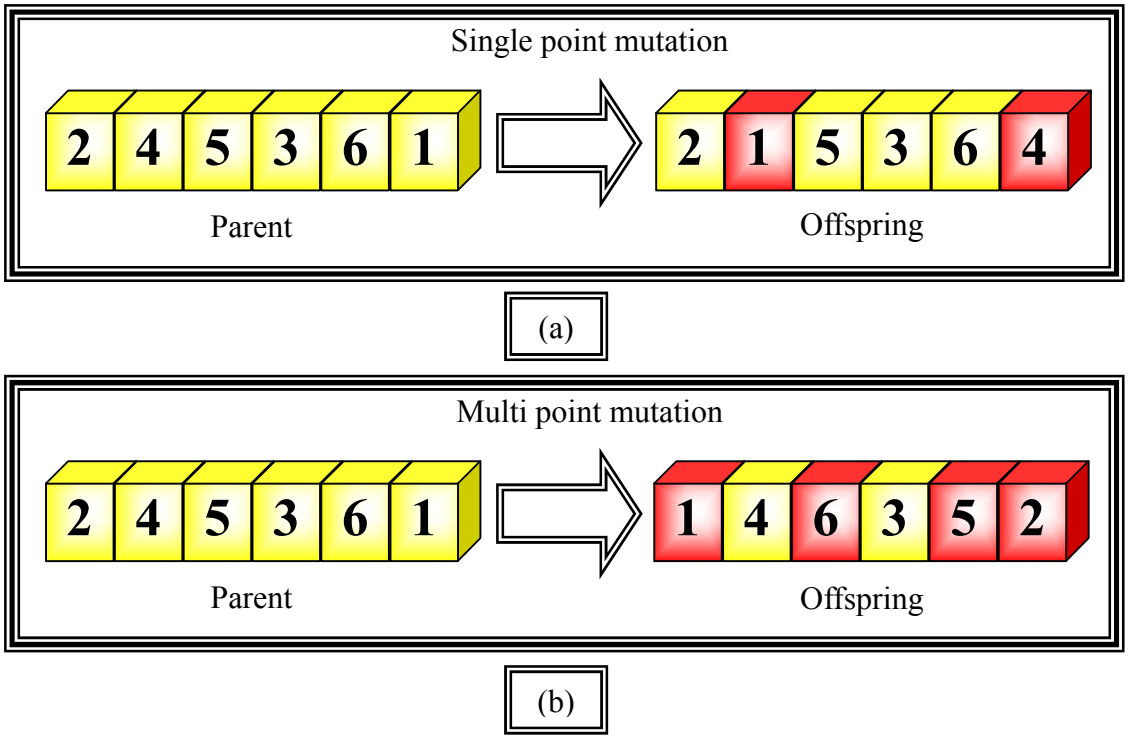


Figure (15) Explanation of the mutation effect on the Integer permutation encoding string. (a) Single point mutation. (b) Multi point mutation.

9. Application of Genetic Algorithms

Genetic algorithms (GAs) are adaptive methods which may be used to solve search and optimization problems. The power of GAs comes from the fact that the technique is robust and can deal successfully with a wide range of problem areas, including those which are difficult for other methods to solve. Therefore, the main ground for GAs is in difficult areas where no such solving techniques exist. Even where existing techniques work well, improvements can be made by mixing them with GAs.

GAs in various forms are implemented to wide rang of problems including the following:

1) Optimization: GAs have been used in a wide variety of optimization tasks, including numerical optimization and combinatorial optimization problems such as circuit design and job shop scheduling.

2) Automatic Programming: GAs have been used to evolve computer programs for special tasks and to design other computational structures cellular automata and sorting networks.

3) Machine and robot learning: GAs have been used for many machine learning applications, including classification and prediction tasks such as the prediction of dynamical systems, weather prediction , and prediction of protein structure. GAs have also been used to design neural networks, and to evolve rules for learning classifier systems or symbolic production systems and to design and control robots

4) Economic models: GAs have been used to model processes of innovation, the development of bidding strategies, and the emergence of economic markets.

5) Immune system models: GAs have been used to model various aspects of the natural immune system including somatic mutation during an

individual's lifetime and the discovery of multi-gene families during evolutionary time

6) Ecological models: GAs have been used to model ecological phenomena such as biological arms races, host-parasite co-evolution, symbiosis, and resource flow in ecologies.

7) Population genetics models: GAs have been used to study questions in population genetics, such as "under what conditions will a gene for recombination be evolutionarily viable (or practicable) ?"

8) Interactions between evolution and learning: GAs have been used to study how individual learning and species evolution affect one another.

9) Models of social systems: GAs have been used to study evolutionary aspects of social systems, such as the evolution of cooperation, the evolution of communication, and trail-following behavior in ants.

This list is by no means exhaustive, but it gives a flavor of the kinds of things for which GAs have been used, both for problem-solving and for modeling .

10. Advantages and Disadvantage of Genetic Algorithms

Perhaps it isn't obvious why such an algorithm should lead to accurate solutions for optimization problems. Crossover is a crucial aspect of any genetic algorithm, but it may seem that it will dramatically change parents with a high fitness function so that they will no longer be fit. However, this is not the case. As in biology, crossover can lead to new combinations of genes which are more fit than any in the previous generations. Other offspring will be created which are less fit but these will have a low probability of being selected to go on to the next generation. Creating new variants is the key to genetic algorithms, as there is a good chance of finding better solutions. This is why mutation is

also a necessary part of the genetic algorithms. It will create offspring which would not have arisen otherwise, and may lead to a better solution.

Other optimization algorithms have the disadvantage that some kind of initial guess is required and this may bias the final result. GAs on the other hand only require a search range, which need only be constrained by prior knowledge of the physical properties of the system. Effectively they search the whole of the solution space, without calculating the fitness function at every point. This can help avoid a danger in any optimization problem which is being trapped in local maxima or minima. There are two main reasons for this:

1) The initial population, being randomly generated, will sample the whole of the solution space, and not just a small area.

2) Variation inducing tactics, i.e. crossover and mutation, prevent the algorithm being trapped in one part of the solution space.

Genetic algorithms can be employed for a wide variety of optimization problems. They perform very well for large scale optimization problems which may be very difficult or impossible to solve by other traditional methods.

The disadvantage of genetic algorithms is that it, Sometimes, have trouble finding the exact global optimum because there is no guaranty to find best solution. Another drawback that GAs require large number of response (fitness) function evaluations depending on the number of individuals and the number of generations. Therefore, genetic algorithms may take long time to evaluate the individuals.