Genetic Algorithms:

Genetic Algorithm (GA) : is a search-based optimization technique based on the principles of **Genetics and Natural Selection**. It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise would take long time. It is frequently used to solve optimization problems, in research, and in machine learning.

Optimization refers to finding the values of inputs in such a way that we get the "best" output values. The definition of "best" varies from problem to problem, but in mathematical terms, it refers to maximizing or minimizing one or more objective functions, by varying the input parameters.

. Genetic Algorithms (GAs) are search based algorithms based on the concepts of natural selection and genetics. GA is a subset of a much larger branch of computation known as **Evolutionary Computation**.

In GAs, we have a **pool or a population of possible solutions** to the given problem. These solutions then undergo recombination and mutation (like in natural genetics), producing new children, and the process is repeated over various generations. Each individual (or candidate solution) is assigned a fitness value (based on its objective function value) and the fitter individuals are given a higher chance to mate and yield more "fitter" individuals. This is in line with the Darwinian Theory of "Survival of the Fittest". In this way we keep "evolving" better individuals or solutions over generations, till we reach a stopping criterion.

Advantages of GAs

GAs has various advantages which have made them immensely popular. These include −

- Does not require any derivative information (which may not be available for many real-world problems).
- Is faster and more efficient as compared to the traditional methods.
- Have very good parallel capabilities.
- Optimizes both continuous and discrete functions and also multi-objective problems.
- Provides a list of "good" solutions and not just a single solution.
- Always gets an answer to the problem, which gets better over the time.
- Useful when the search space is very large and there are a large number of parameters involved.

Limitations of GAs

Like any technique, GAs also suffers from a few limitations. These include:

- GAs are not suited for all problems, especially problems which are simple and for which derivative information is available.
- Fitness value is calculated repeatedly which might be computationally expensive for some problems.
- Being stochastic, there are no guarantees on the optimality or the quality of the solution.

• If not implemented properly, the GA may not converge to the optimal solution.

Definitions:

Parameter values: are encoded into binary strings of Fixed and finite length

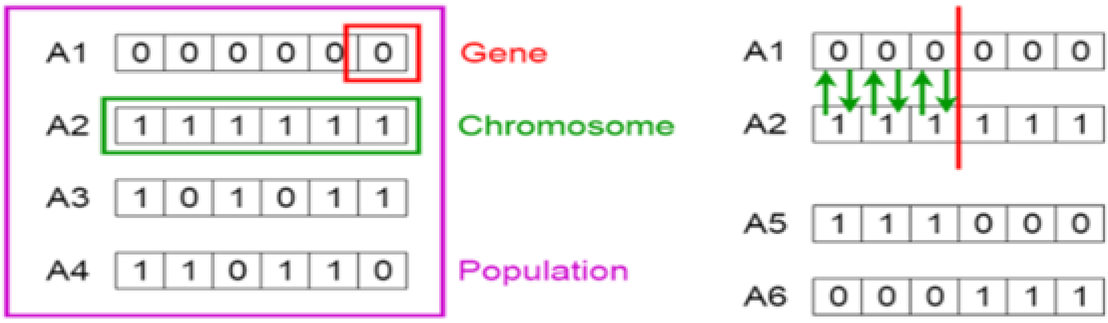
Gene: An individual is characterized by a set of parameters (each bit of the binary string)

Chromosome: represent a string of genes to create set of one or multiple chromosomes, a prospective solution to the given problem

Population: a group of individuals longer string lengths Improve resolution Requires more computation time

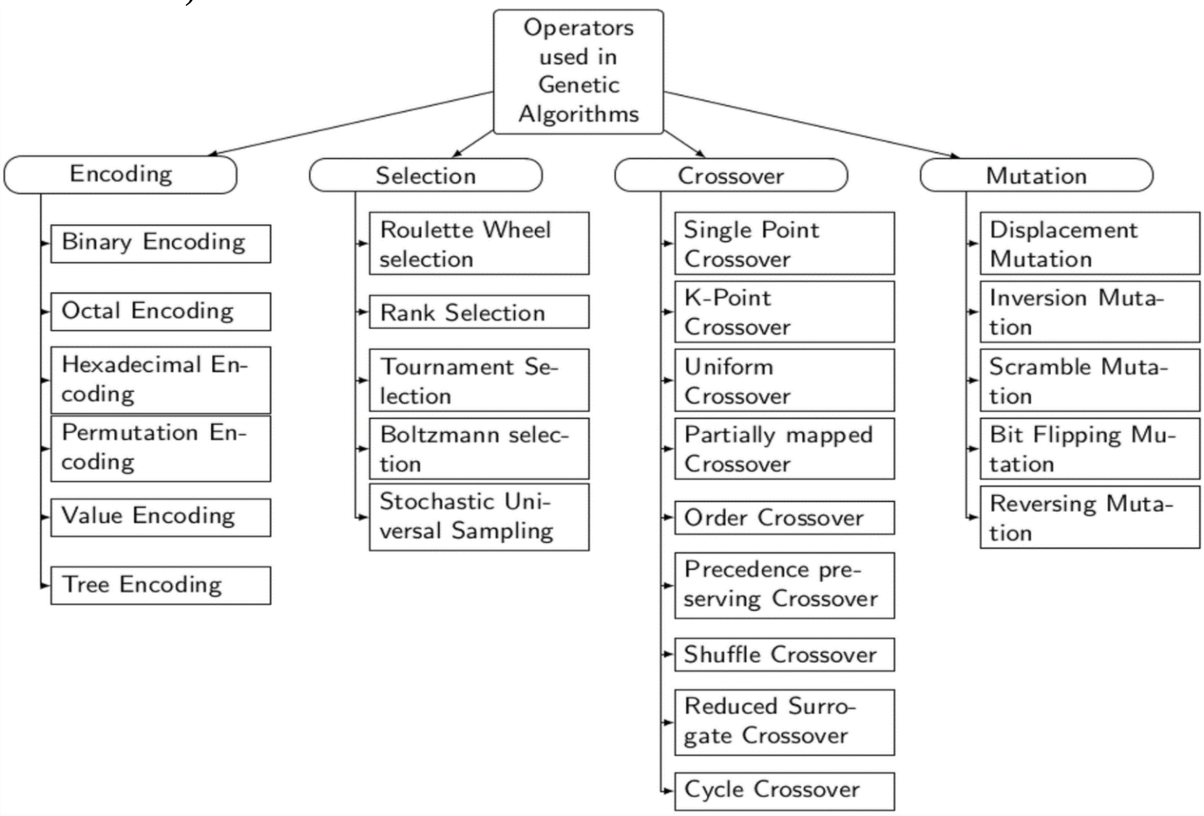
Fitness function: determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a fitness score to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

Sampling: Procedure which computes a new generation from the previous one and its offspring's.



Genetic operators

GAs used a variety of operators during the search process. Five phases are considered in a genetic algorithm. (Initial population, Fitness function, Selection, Crossover and Mutation)



Initial population (Encoding schemes): the set of genes of an individual is represented using a string, in terms of an alphabet. Usually, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome.

: the encoding scheme (i.e., to convert in particular form) plays an important role. The given information has to be encoded in a particular bit string. The encoding schemes are differentiated according to the problem domain. The well-known encoding schemes are binary, octal, hexadecimal, permutation, value-based, and tree.

Selection techniques:

Selection Mechanism for selecting individuals (strings) for reproduction according to their fitness (objective function value).Selection is an important step in genetic algorithms that determines whether the particular string will participate in the reproduction process or not. The convergence rate of GA depends upon the selection pressure. The well-known selection techniques are roulette wheel, rank, tournament, Boltzmann and stochastic universal sampling

- Roulette wheel selection maps all the possible strings onto a wheel with a portion of the wheel allocated to them according to their fitness value. This wheel is then rotated randomly to select specific solutions that will participate in formation of the next generation. However, it suffers from many problems such as errors introduced by its stochastic nature.
- Rank selection is the modified form of Roulette wheel selection. It utilizes the ranks instead of fitness value. Ranks are given to them according to their fitness value so that each individual gets a chance of getting selected according to their ranks. Rank selection method reduces the chances of prematurely converging the solution to a local minimum.
- Tournament selection technique: The individuals are selected according to their fitness values from a stochastic roulette wheel in pairs. After selection, the individuals with higher fitness value are added to the pool of next generation. In this method of selection, each individual is compared with all n-1 other individuals if it reaches the final population of solutions.
- Stochastic universal sampling (SUS) is an extension to the existing roulette wheel selection method. It uses a random starting point in the list of individuals from a generation and selects the new individual at evenly spaced intervals. It gives equal chance to all the individuals in getting selected for participating in crossover for the next generation.

Boltzmann selection is based on entropy and sampling methods. It helps in solving the problem of premature convergence. However, there is a possibility of information loss. It can be managed through elitism. Elitism selection for improving the performance of Roulette wheel selection. It ensures the elitist individual in a generation is always propagated to the next generation.

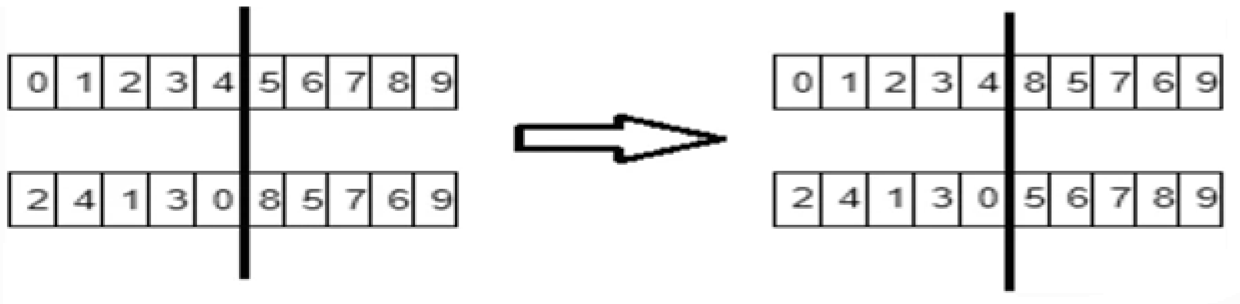
Selection Techniques	Pros	Cons
Roulette wheel	Easy to implement Simple Free from Bias	Risk of Premature convergence Depends upon variance present in the fitness function
Rank	Preserve diversity Free from Bias	Slow convergence Sorting required Computationally Expensive
Tournament	Preserve diversity Parallel Implementation No sorting required	Loss of diversity when the tournament size is large
Boltzmann	Global optimum achieved	Computationally Expensive
Stochastic Universal Sampling	Fast Method Free from Bias	Premature convergence
Elitism	Preserve best Individual in population	Best individual can be lost due to crossover and mutation operators

Crossover operators

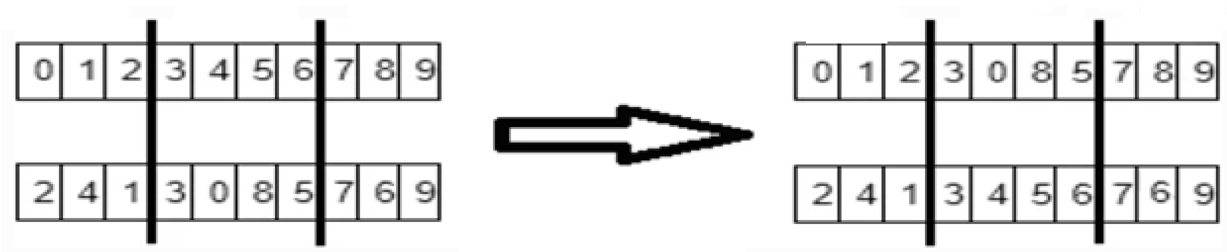
Crossover: Method of merging the genetic information of two individuals; if the coding is chosen properly, two good parents produces good children.

Crossover operators are used to generate the offspring by combining the genetic information of two or more parents. The well-known crossover operators are single-point, two-point, k-point, uniform, partially matched, order, precedence preserving crossover, shuffle, reduced surrogate and cycle.

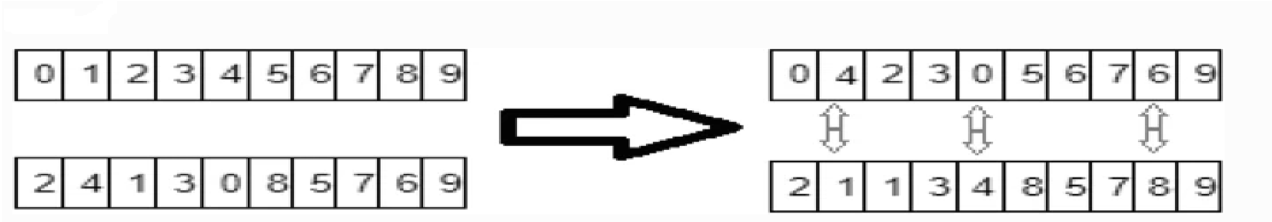
- In a single point crossover, a random crossover point is selected. The genetic information of two parents which is beyond that point will be swapped with each other. The genetic information after swapping. It replaced the tail array bits of both the parents to get the new offspring.



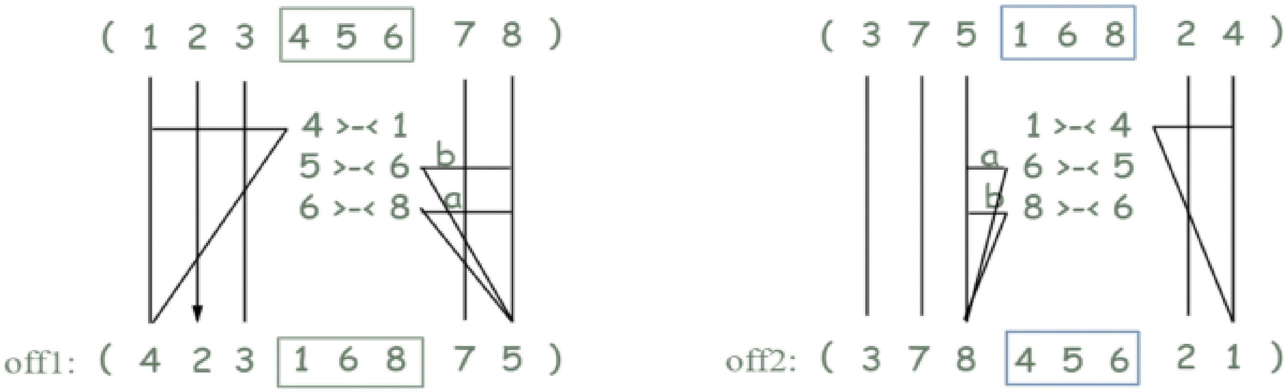
- In a two point and k-point crossover, two or more random crossover points are selected and the genetic information of parents will be swapped as per the segments that have been created. The middle segment of the parents is replaced to generate the new offspring.



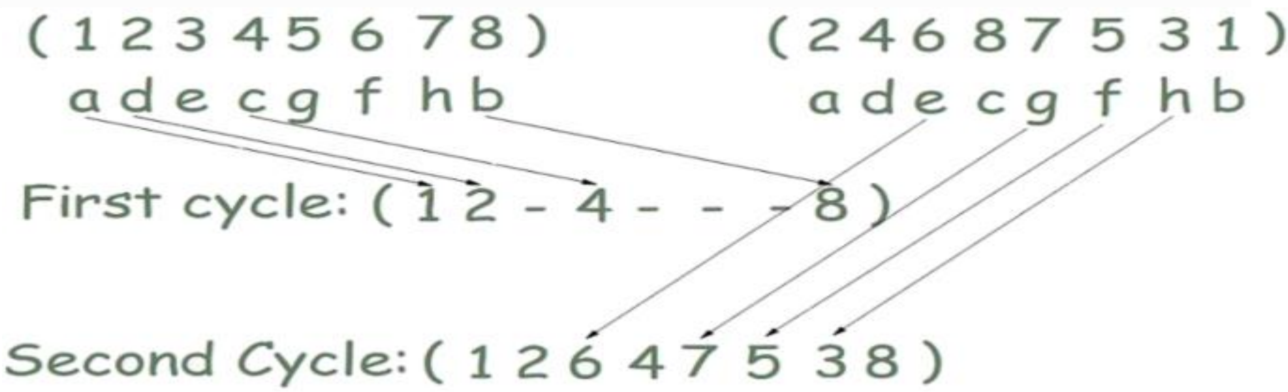
- In a uniform crossover, parent cannot be decomposed into segments. The parent can be treated as each gene separately. We randomly decide whether we need to swap the gene with the same location of another chromosome.



- Partially matched crossover (PMX) is the most frequently used crossover operator. It is an operator that performs better than most of the other crossover operators. Two parents are choosing for mating. One parent donates some part of genetic material and the corresponding part of other parent participates in the child. Once this process is completed, the left out alleles are copied from the second parent.



- Shuffle crossover shuffles the values of an individual solution before the crossover and unshuffles them after crossover operation is performed so.RCX is based on the assumption that GA produces better individuals if the parents are sufficiently diverse in their genetic composition. However, RCX cannot produce better individuals for those parents that have same composition. It attempts to generate an offspring using parents where each element occupies the position by referring to the position of their parents. In the first cycle, it takes some elements from the first parent. In the second cycle, it takes the remaining elements from the second parent.



Mutation

Mutation is the occasional introduction of new features in to the solution strings of the population pool to maintain diversity in the population. Though crossover has the main responsibility to search for the optimal solution, mutation is also used for this purpose.

Again, similar to the case of crossover, the choice of the appropriate mutation technique depends on the coding and the problem itself. We mention a few alternatives and again:

Inversion of single bits: With probability p_M, one randomly chosen bit is negated.

Bitwise inversion: The whole string is inverted bit by bit with prob. p_M.

Random selection: With probability p_M, the string is replaced by a randomly chosen one

GA Algorithm

In the genetic algorithm process is as follows

Step 1: Determine the number of chromosomes, generation, and mutation rate and crossover rate value

Step 2: Generate chromosome-chromosome number of the population, and the initialization value of the genes chromosome-chromosome with a random value >

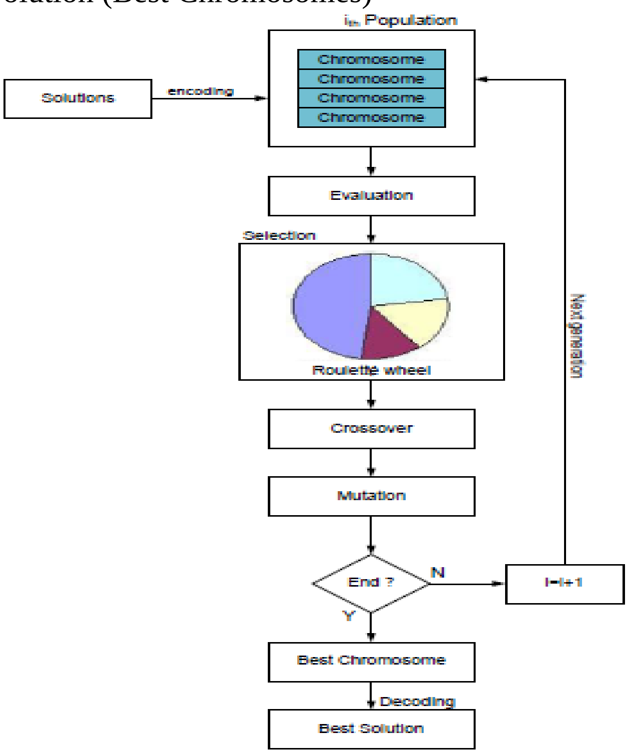
Step3 : Evaluation of fitness value of chromosomes by calculating objective function

Step 5: Chromosomes selection

Step 6: Crossover

Step 7. Mutation

Step 8. Solution (Best Chromosomes)



Numerical Example

Here are examples of applications that use genetic algorithms to solve the problem of combination. Suppose there is equality **a** + 2**b** + 3**c** + 4**d** = 30, genetic algorithm will be used to find the value of **a**, **b**, **c**, and **d** that satisfy the above equation. First we should formulate the objective function, for this problem the objective is minimizing the value of function **f**(**x**) where b

f(**x**) = ((**a** + 2**b** + 3**c** + 4**d**) - 30).

Since there are four variables in the equation, namely **a**, **b**, **c**, and **d**, we can compose the chromosome as follow: To speed up the computation, we can restrict that the values of variables **a**, **b**, **c**, and ***d*** are integers between 0 and 30.

a	b	c	d

Step 1: Initialization

For example we define the number of chromosomes in population are 6, then we generate random value of gene **a**, **b**, **c**, **d** for 6 chromosomes

Chromosome [1] = [**a**; **b**; **c**; **d**] = [12; 05; 23; 08]
Chromosome [2] = [**a**; **b**; **c**; **d**] = [02; 21; 18; 03]
Chromosome [3] = [**a**; **b**; **c**; **d**] = [10; 04; 13; 14]
Chromosome [4] = [**a**; **b**; **c**; **d**] = [20; 01; 10; 06]
Chromosome [5] = [**a**; **b**; **c**; **d**] = [01; 04; 13; 19]
Chromosome [6] = [**a**; **b**; **c**; **d**] = [20; 05; 17; 01]

Step 2. Evaluation

We compute the objective function value for each chromosome produced in initialization step:

F_obj [1] = Abs ((12 + 2*05 + 3*23 + 4*08) - 30)
= Abs ((12 + 10 + 69 + 32) - 30)
= Abs (123 - 30)
= 93

F_obj[2] = Abs((02 + 2*21 + 3*18 + 4*03) - 30)
= Abs ((02 + 42 + 54 + 12) - 30)
= Abs (110 - 30)
= 80

F_obj[3] = Abs((10 + 2*04 + 3*13 + 4*14) - 30)
= Abs ((10 + 08 + 39 + 56) - 30)
= Abs (113 - 30)
= 83

F_obj[4] = Abs((20 + 2*01 + 3*10 + 4*06) - 30)
= Abs ((20 + 02 + 30 + 24) - 30)
= Abs (76 - 30)
= 46

F_obj[5] = Abs((01 + 2*04 + 3*13 + 4*19) - 30)

= Abs ((01 + 08 + 39 + 76) - 30)

= Abs (124 - 30)

= 94

F_obj[6] = Abs((20 + 2*05 + 3*17 + 4*01) - 30)

= Abs ((20 + 10 + 51 + 04) - 30)

=Abs (85-30)

=55

Step 3. Selection

1. The fittest chromosomes have higher probability to be selected for the next generation. To compute fitness probability we must compute the fitness of each chromosome. To avoid divide by zero problem, the value of **F_obj** is added by 1.

Fitness [1] = 1 / (1+**F_obj**[1]) = 1 / 94 = 0.0106

Fitness [2] = 1 / (1+**F_obj**[2]) = 1 / 81 = 0.0123

Fitness [3] = 1 / (1+**F_obj**[3]) = 1 / 84 = 0.0119

Fitness [4] = 1 / (1+**F_obj**[4]) = 1 / 47 = 0.0213

Fitness [5] = 1 / (1+**F_obj**[5]) = 1 / 95 = 0.0105

Fitness [6] = 1 / (1+**F_obj**[6]) = 1 / 56 = 0.0179

Total = 0.0106 + 0.0123 + 0.0119 + 0.0213 + 0.0105 + 0.0179

= 0.0845

The probability for each chromosome is formulated by:

P[i] = **Fitness**[i] / **Total**

P [1] = 0.0106 / 0.0845 = 0.1254

P [2] = 0.0123 / 0.0845 = 0.1456

P[3] = 0.0119 / 0.0845 = 0.1408

P[4] = 0.0213 / 0.0845 = 0.2521

P[5] = 0.0105 / 0.0845 = 0.1243

P[6] = 0.0179 / 0.0845 = 0.2118

From the probabilities above we can see that Chromosome 4 that has the highest fitness, this chromosome has highest probability to be selected for next generation chromosomes. For the selection process we use roulette wheel, for that we should compute the cumulative probability values:

C [1] = 0.1254

C [2] = 0.1254 + 0.1456 = 0.2710

C [3] = 0.1254 + 0.1456 + 0.1408 = 0.4118

C [4] = 0.1254 + 0.1456 + 0.1408 + 0.2521 = 0.6639

C [5] = 0.1254 + 0.1456 + 0.1408 + 0.2521 + 0.1243 = 0.7882

C [6] = 0.1254 + 0.1456 + 0.1408 + 0.2521 + 0.1243 + 0.2118 = 1.0

Having calculated the cumulative probability of selection process using roulette-wheel can be done. The process is to generate random number **R** in the range 0-1 as follows.

R [1] = 0.201	**R** [2] = 0.284	**R** [3] = 0.099
R [4] = 0.822	**R** [5] = 0.398	**R** [6] = 0.501

If random number **R**[1] is greater than **C**[1] and smaller than **C**[2] then select **Chromosome**[2] as a chromosome in the new population for next generation:

NewChromosome [1] = **Chromosome** [2]

NewChromosome [2] = **Chromosome** [3]

NewChromosome [3] = **Chromosome** [1]

NewChromosome [4] = **Chromosome** [6]

NewChromosome [5] = **Chromosome** [3]

NewChromosome [6] = **Chromosome** [4]

Chromosomes in the population thus became:

Chromosome [1] = [02; 21; 18; 03]	**Chromosome** [2] = [10; 04; 13; 14]
Chromosome [3] = [12; 05; 23; 08]	**Chromosome** [4] = [20; 05; 17; 01]
Chromosome [5] = [10; 04; 13; 14]	**Chromosome** [6] = [20; 01; 10; 06]

In this example, we use one-cut point, i.e. randomly select a position in the parent chromosome then exchanging sub-chromosome. Parent chromosome which will mate is randomly selected and the number of mate Chromosomes is controlled using **crossover rate** (**ρc**) parameters. Pseudo-code for the crossover process is as follows:

```
k← 0;
while (k<population) do
      R[k] = random (0-1);
     if(R[k]< ρc) then
    Select Chromosome[k] as parent;
  End;
k = k + 1;
End;
End;
```

Chromosome **k** will be selected as a parent if **R**[**k**]<**ρc**. Suppose we set that the crossover rate is 25%, then Chromosome number **k** will be selected for crossover if random generated value for

Chromosome **k** below 0.25. The process is as follows: First we generate a random number **R** as the number of population.

R [1] = 0.191 **R** [2] = 0.259 **R**[3] = 0.760

R [4] = 0.006 **R** [5] = 0.159 **R**[6] = 0.340

For random number **R** above, parents are **Chromosome** [1], **Chromosome** [4] and **Chromosome** [5] will be selected for crossover.

Chromosome [1] >< **Chromosome** [4]

Chromosome [4] >< **Chromosome** [5]

Chromosome [5] >< **Chromosome** [1]

After chromosome selection, the next process is determining the position of the crossover point. This is done by generating random numbers between 1 to (length of Chromosome – 1). In this case, generated random numbers should be between 1 and 3. After we get the crossover point, parents Chromosome will be cut at crossover point and its gens will be interchanged. For example we generated 3 random number and we get:

C [1] = 1 **C** [2] = 1 **C** [3] = 2

Then for first crossover, second crossover and third crossover, parent's gens will be cut at gen number 1, gen number 1 and gen number 3 respectively, e.g.

Chromosome [1] = **Chromosome** [1] >< **Chromosome** [4]

= [02; 21; 18; 03] >< [20; 05; 17; 01]

= [02; 05; 17; 01]

Chromosome [4] = **Chromosome** [4] >< **Chromosome** [5]

= [20; 05; 17; 01] >< [10; 04; 13; 14]

= [20; 04; 13; 14]

Chromosome [5] = **Chromosome** [5] >< **Chromosome** [1]

= [10; 04; 13; 14] >< [02; 21; 18; 03]

= [10; 04; 18; 03]

Thus Chromosome population after experiencing a crossover process:

Chromosome [1] = [02; 05; 17; 01] **Chromosome** [2] = [10; 04; 13; 14]

Chromosome [3] = [12; 05; 23; 08] **Chromosome** [4] = [20; 04; 13; 14]

Chromosome [5] = [10; 04; 18; 03] **Chromosome** [6] = [20; 01; 10; 06]

Step 5. Mutation

Number of chromosomes that have mutations in a population is determined by the **mutation_ rate** parameter. Mutation process is done by replacing the gen at random position with a new value. The process is as follows. First we must calculate the total length of gen in the population. In this case the total length of gen is

total_gen = **number_ of _gen _in_ Chromosome** * **number of population**

= 4 * 6 = 24

Mutation process is done by generating a random integer between 1 and total_gen (1 to 24). If generated random number is smaller than mutation_rate (ρm) variable then marked the position of gen in chromosomes. Suppose we define ρm 10%, it is expected that 10% (0.1) of total_gen in the population that will be mutated:

number of mutations = 0.1 * 24 = 2.4 ≈ 2

Suppose generation of random number yield 12 and 18 then the chromosome which have mutation are Chromosome number 3 gen number 4 and Chromosome 5 gen number 2. The value of mutated gens at mutation point is replaced by random number between 0-30. Suppose generated random number are 2 and 5 then Chromosome composition after mutation are:

Chromosome [1] = [02; 05; 17; 01] **Chromosome** [2] = [10; 04; 13; 14]

Chromosome [3] = [12; 05; 23; **02**] **Chromosome** [4] = [20; 04; 13; 14]

Chromosome [5] = [10; **05**; 18; 03] **Chromosome** [6] = [20; 01; 10; 06]

Finishing mutation process then we have one iteration or one generation of the genetic algorithm. We can now evaluate the objective function after one generation:

Chromosome [1]= [02;05;17;01]

F_obj[1] = Abs((02 + 2*05 + 3*17 + 4*01) - 30)

= Abs ((2 + 10 + 51 + 4) - 30)

= Abs (67 - 30)

= 37

Chromosome [2] = [10;04;13;14]

F_obj[2] = Abs((10 + 2*04 + 3*13 + 4*14) - 30)

= Abs ((10 + 8 + 33 + 56) - 30)

= Abs(107 - 30)

= 77

Chromosome[3] = [12;05;23;02]

F_obj[3] = Abs((12 + 2*05 + 3*23 + 4*02) - 30)

= Abs((12 + 10 + 69 + 8) - 30)

= Abs(87 - 30)

= 47

Chromosome [4] = [20;04;13;14]

F_obj[4] = Abs((20 + 2*04 + 3*13 + 4*14) - 30)

= Abs((20 + 8 + 39 + 56) - 30)

= Abs(123 - 30)

= 93

Chromosome [5] = [10; 05; 18; 03]

F_obj[5] = Abs((10 + 2*05 + 3*18 + 4*03) - 30)

= Abs((10 + 10 + 54 + 12) - 30)

= Abs(86 - 30)

= 56

Chromosome [6] = [20; 01; 10; 06]

F_obj[6] = Abs((20 + 2*01 + 3*10 + 4*06) - 30)

= Abs ((20 + 2 + 30 + 24) - 30)

= Abs (76 - 30)

= 46

From the evaluation of new Chromosome we can see that the objective function is decreasing, this means that we have better Chromosome or solution compared with previous Chromosome generation. New Chromosomes for next iteration are:

Chromosome [1] = [02;05;17;01] **Chromosome** [2] = [10;04;13;14]

Chromosome [3] = [12;05;23;02] **Chromosome**[4] = [20;04;13;14]

Chromosome [5] = [10;05;18;03] **Chromosome**[6] = [20;01;10;06]

These new Chromosomes will undergo the same process as the previous generation of Chromosomes such as evaluation, selection, crossover and mutation and at the end it produce new generation of Chromosome for the next iteration. This process will be repeated until a predetermined number of of generations. For this example, after running 50 generations, best chromosome is obtained:

Chromosome = [07; 05; 03; 01]

This means that: **a** = 7, **b** = 5, **c** = 3, **d** = 1

If we use the number in the problem equation:

a + 2**b** + 3**c** + 4**d** = 30

7 + (2 * 5) + (3 * 3) + (4 * 1) = 30

We can see that the value of variable **a**, **b**, **c** and **d** generated by genetic algorithm can satisfy that equality.