2. Terminologies and Operators of GA

In the following sections we will discusses the basic terminologies and operators used in Genetic Algorithms to achieve a good enough solution for possible terminating conditions.

2.1 Key Elements

Individuals : The two distinct elements in the GA are individuals and populations. An individual is a single solution while the population is the set of individuals currently involved in the search process. Each individual in the population is called a string or ***chromosome,*** in analogy to chromosomes in natural systems. The chromosome, which is the raw 'genetic' information that the GA deals.

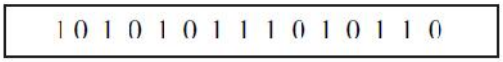


Fig. 2.1 Representation of a chromosome

Genes: Genes are the basic "instructions" for building a Generic Algorithms. A chromosome is a sequence of genes. Genes may describe a possible solution to a problem, without actually being the solution. A gene is a bit string of arbitrary lengths and can be represented as shown in the Fig. 2.2. The location of gene in a chromosome is called ***locus.***

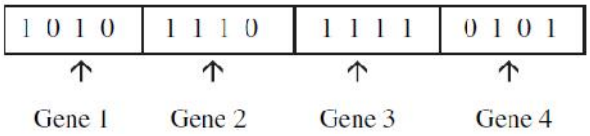


Fig. 2.2 Representation of gene

Fitness: The ***fitness*** function is the evaluation function that is used to determine the fitness of each chromosome. The fitness function is problem specific and user defined. For calculating fitness, the chromosome has to be first decoded and the objective function has to be evaluated. The fitness not only indicates how good the solution is, but also corresponds to how close the chromosome is to the optimal one.

Populations: The population is the set of individuals currently involved in the search. The two important aspects of population used in Genetic Algorithms are:

1. The initial population generation.
2. The population size.

Population	Chromosome 1	1 1 1 0 0 0 1 0
	Chromosome 2	0 1 1 1 1 0 1 1
	Chromosome 3	1 0 1 0 1 0 1 0
	Chromosome 4	1 1 0 0 1 1 0 0

Fig. 2.3 Population

For each and every problem, the population size will depend on the complexity of the problem. It is often a random initialization of population is carried. Ideally, the first population should have a gene pool as large as possible in order to be able to explore the whole search space. Population being combination of various chromosomes is represented as in Fig. 2.3. The above population consists of four chromosomes.

2.2 Genetic Algorithms Requirements

In order to apply a genetic algorithm, there are several questions that need to be answered:

- How is an individual represented?
- How is an individual's fitness calculated?
- How are individuals selected for breeding?
- How are individuals crossed-over?
- How are individuals mutated?
- How big should the population be?
- What are the "termination conditions"?

Most of these questions have problem-specific answers. The last two, however, can be discussed in a more general way.

Population size

The size of the population is highly variable. The larger the population, the more possible solutions there are, which means that there is more variation in the population. Variation means that it is more likely that good solutions will be created. Therefore,

- The population should be as large as possible.
- The limiting factor is, of course, the running time of the algorithm. The larger the population, the longer the algorithm takes to run.

Stopping condition

The GA can be terminated according to several stopping criteria:

1. A fixed number of generations have occurred;
2. All individuals in the population converge to the same string;
3. No improvements in the fitness value are found after a certain number of generations.

Once the stopping criterion is met and the new generation is evolved, the old generation can be discarded.

Since most of the other questions are dependent upon the search problem, we will look at example problem that can be solved using genetic algorithms: finding a mathematical function's maximum.

***Example -1*:** One application for a genetic algorithm is to find values for a collection of variables that will maximize a particular function of those variables. For this example, let's assume that we are trying to determine the variables that produce the maximum value for the function:

$$\mathbf{f(w, x, y, z) = w^3 + x^2 - y^2 - z^2 + 2yz - 3wx + wz - xy + 2}$$

To use GA, we need to answer the previously mentioned questions

How is an individual represented?

A simple way to do this is to simply have an array of four values (integers or floating point numbers, either way). So, we can simply choose a size in bits for each variable, and then concatenate the four values together into a single bit string.

For example, we will choose to represent each variable as a four-bit integer, making our entire individual a 16-bit string. Thus, an individual such as

1	1	0	1	0	1	1	0	0	1	1	1	1	1	0	0

represents a solution where w = 13, x = 6, y = 7, and z = 12.

How is an individual's fitness calculated?

Next we consider how to determine the fitness of each individual. The fitness function is a function that measures the value of the individual relative to the rest of the population.

In our example, an obvious evaluation function would be to simply calculate the value of ***f*** for the given variables. For example, assume we have a population of 4 individuals:

1010 1110 1000 0011
0110 1001 1111 0110
0111 0110 1110 1011
0001 0110 1000 0000

The first individual represents w = 10, x = 14, y = 8, and z = 3, for an f value of 671. The values for the entire population can be seen in the following table:

Individual	w	x	y	z	***f***
1010 1110 1000 0011	10	14	8	3	671
0110 1001 1111 0110	6	9	15	6	-43
0111 0110 1110 1011	7	6	14	11	239
0001 0110 1000 0000	1	6	8	0	-91

The fitness function can be chosen from many options. The key is that the fitness of an individual should represent the value of the individual relative to the rest of the population, so that the best individual has the highest fitness.

How are individuals selected for breeding(crossover)?

The key to the selection process is that it should be probabilistically weighted so that higher fitness individuals have a higher probability of being selected. One possibility is to use the ordinal method for the fitness function, then calculate a probability of selection that is equal to the individual's fitness value divided by the total fitness of all the individuals. Clearly this is giving the better individuals more chances to be selected.

A similar approach could be used with the average fitness calculations. This method makes the probability more dependent on the relative evaluation functions of each individual.

How are individuals crossed-over?

Once we have selected a pair of individuals, they are "breed" or in genetic algorithm language, they are crossed-over. Typically two children are created from each set of parents. One method for performing the cross-over is described here, but there are other approaches. Two locations are randomly chosen within the individual. These define corresponding substrings in each individual. The substrings are swapped between the two parent individuals, creating two new children. For example, let's look at our four individuals again:

1010 1110 1000 0011
0110 1001 1111 0110
0111 0110 1110 1011
0001 0110 1000 0000

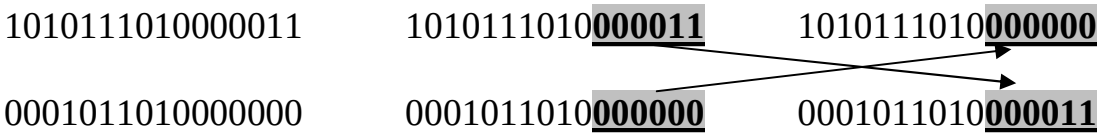
Let's assume that ***the first and third individuals*** are chosen for cross-over (making sense, as these are the two top individuals).

Keep in mind that the selection process is random, however. ***The fourth and fourteenth bits*** are randomly selected to define the substring to be swapped, so the cross-over looks like this:

1010111010000011 101**01110100000**11 101**10110111010**11

0111011011101011 011**10110111010**11 011**01110100000**11

Thus, two new individuals are created. We need one more cross-over. Assume that the ***first and fourth*** individuals are selected this time. Note that an individual may be selected multiple times for breeding, while other individuals might never be selected. Further assume that the ***eleventh and sixteenth bits*** are randomly selected for the cross-over point. We would see a second cross-over like this:



So, our second generation population is:

1011 0110 1110 1011
0110 1110 1000 0011
1010 1110 1000 0000
0001 0110 1000 0011

How are individuals mutated?

Finally, we need to allow individuals to mutate. When using bit strings, the easiest way to implement the mutation is to allow every single bit in every individual a chance to mutate. This chance should be very small, since we don't want to have individuals changing dramatically due to mutation. Setting the percentage so that roughly one bit per individual will change on average is probably a reasonably good number.

The mutation will consist of having the bit "flip" – a 1 changes to a 0 and a 0 changes to a 1. In our example, assume that the bold, italic, and shaded bits have been chosen for mutation:

1011011011101011 → 1011011011101011
0110111010000011 → 0110101010000011
1010111010000000 → 1010111010010000
0001011010000011 → 0101011010000001

Finally, let's look at the new population:

Individual	w	x	y	z	f
1011011011101011	11	6	14	11	1199
0110101010000011	6	10	8	3	51

1010111010010000	10	14	9	0	571
0101011010000011	5	6	8	3	15

The average evaluation value here is 459, versus an average of 194 for the previous generation.

2.3 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.

2.3.1 Binary Encoding

The most common way of encoding is a binary string, which would be represented as in Fig. 2.4. 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.

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

Fig. 2.4 Binary encoding

2.3.2 Octal Encoding

This encoding uses string made up of octal numbers (0–7) which would be represented as in Fig. 2.5.

Chromosome 1	03467216
Chromosome 2	15723314

Fig. 2.5 Octal encoding

2.3.3 Hexadecimal Encoding

This encoding uses string made up of hexadecimal numbers (0–9, A–F) as illustrated in Fig. 2.6.

Chromosome 1	9CE7
Chromosome 2	3DBA

Fig. 2.6 Hexadecimal encoding

2.3.4 Permutation Encoding (Real Number Coding)

In permutation encoding, every chromosome is a string of integer/real values, which represents number in a sequence. Fig. 2.7 represents permutation encoding.

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

Fig. 2.7 Permutation (Real Number) Encoding

Permutation encoding is only useful for ordering problems. Even for this problems for some types of crossover and mutation corrections must be made to leave the chromosome consistent (i.e., have real sequence in it).

2.3.5 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. 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. On the other hand, for this encoding is often necessary to develop some new crossover and mutation specific for the problem.

Chromosome A	1.2324 5.3243 0.4556 2.3293 2.4545
Chromosome B	ABDJEIFJDHDIERJFDLDFLFEGT
Chromosome C	(back), (back), (right), (forward), (left)

Fig. 2.8 Value Encoding

2.4 Breeding (Crossover)

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:

a. Selecting parents.

b. Crossing the parents to create new individuals (offspring or children).

c. Replacing old individuals in the population with the new ones.

2.4.1 Selection

Selection is the process of choosing two parents from the population for crossing. After deciding on an encoding, the next step is to decide how to perform selection i.e., how to choose individuals in the population that will create offspring for the next generation and how many offspring each will create. The purpose of selection is to emphasize fitter individuals in the population in hopes that their off springs have higher fitness. There are different techniques that can be used in GA selection. Some of these techniques are introduced here.

2.4.1.1 Roulette Wheel Selection

Roulette selection is one of the traditional GA selection techniques.. 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.

Roulette Wheel selection process can be explained as follows:

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

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

2.4.1.2 Random Selection

This technique randomly selects a parent from the population. Random selection is a little more disruptive, on average, than roulette wheel selection.

2.4.1.3 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.

2.4.2 Crossover (Recombination)

Crossover is the process of taking two parent solutions and producing from them a child (offspring). 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:

- The reproduction operator selects at random a pair of two individual strings for the mating.
- A cross site is selected at random along the string length.
- 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:

2.4.2.1 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. Fig. 2.9 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.

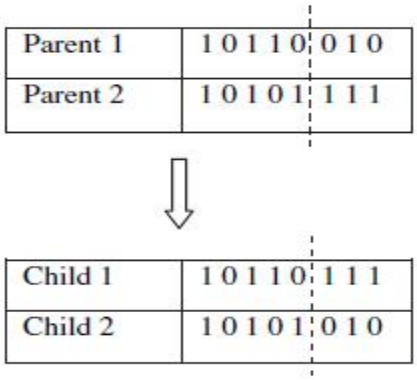


Fig. 2.9 Single point crossover

2.4.2.2 Two Point Crossover

In two-point crossover, two crossover points are chosen and the contents between these points are exchanged between two mated parents. In Fig. 2.10 the dotted lines indicate the crossover points. Thus the contents between these points are exchanged between the parents to produce new children for mating in the next generation.

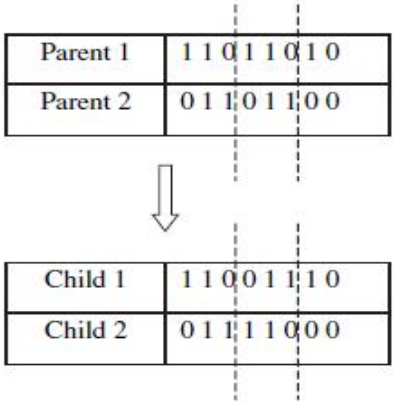


Fig. 2.10 Two – points crossover

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.

2.4.2.3 Uniform Crossover

In uniform 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).

In Fig. 2.11, 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.

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

Fig. 2.11 Uniform crossover

2.4.2.4 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. 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.

2.4.3 Mutation

After crossover, the strings are subjected to mutation. Mutation prevents the algorithm to be trapped in a local minimum. It introduces new genetic structures in the population by randomly modifying some of its building blocks. 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.

2.4.3.1 Flipping

Flipping of a bit involves changing 0 to 1 and 1 to 0 based on a mutation chromosome generated.

Fig. 2.12 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	**1** 0 1 1 **0** 1 0 **1**
Mutation chromosome	**1** 0 0 0 **1** 0 0 **1**
Child	**0** 0 1 1 **1** 1 0 **0**

Fig. 2.12 Mutation flipping

2.4.3.2 Interchanging

Two random positions of the string are chosen and the bits corresponding to those positions are interchanged. This is shown in Fig. 2.13.

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

Fig. 2.13 Interchanging

2.4.3.3 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. 2.14.

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

Fig. 2.14 Reversing

2.4.3.4 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 should not occur very often, because then GA will in fact change to random search.

2.4.4 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 offsprings are produced, a method must determine which of the current members of the population, if any, should be replaced by the new solutions.

2.4.4.1 Random Replacement

The children replace two randomly chosen individuals in the population. The parents are also candidates for selection.

2.4.4.2 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 population.

2.4.4.3 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.

2.4.4.4 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.