Encoding:

Before a genetic algorithm can be put to work on any problem, a method is needed to encode potential solutions to that problem in a form so that a computer can process.

- One common approach is to encode solutions as binary strings: sequences of l's and O's, where the digit at each position represents the value of some aspect of the solution.

Example :

A Gene represents some data (eye color, hair color, sight, etc.). A chromosome is an array of genes. In binary form a Gene looks like : (11100010)

a Chromosome looks like: Genel Gene2 Gene3 Gene4

(11000010, 00001110, 001111010, 10100011)

A chromosome should in some way contain information about solution which it represents; it thus requires encoding. The most popular way of encoding is a binary string like :

Chromosome 1 : 1101100100110110

Chromosome 2 : 1101111000011110

Each bit in the string represent some characteristics of the solution.

- There are many other ways of encoding, e.g., encoding values as integer or real numbers or some permutations and so on.
- The virtue of these encoding method depends on the problem to work on .

Binary Encoding:

Binary encoding is the most common to represent information contained. In genetic algorithms, it was first used because of its relative simplicity.

-In binary encoding, every chromosome is a string of bits : 0 or 1, like

Chromosome1: 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 1 1 1 0 0 1 0 1

Chromosome2: 1 1 1 1 1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1 1 1

- Binary encoding gives many possible chromosomes even with a small number of alleles i.e. possible settings for a trait (features).
- This encoding is often not natural for many problems and sometimes corrections must be made after crossover and/or mutation.

Example:

One variable function, say 0 to 15 numbers, numeric values, represented by 4 bit binary string.

Numeric value	4-bit string	Numeric value	4-bit string	Numeric value	4-bit string
0	0 0 0 0	6	0 1 1 0	12	1 1 0 0
1	0 0 0 1	7	0 1 1 1	13	1 1 0 1
2	0 0 1 0	8	1 0 0 0	14	1 1 1 0
3	0 0 1 1	9	1 0 0 1	15	1 1 1 1
4	0 1 0 0	10	1 0 1 0		
5	0 1 0 1	11	1 0 1 1		

Value Encoding:

The Value encoding can be used in problems where values such as real numbers are used. Use of binary encoding for this type of problems would be difficult.

1.In value encoding, every chromosome is a sequence of some values.

2.The Values can be anything connected to the problem, such as : real numbers, characters or objects.

Example:

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

3. Value encoding is often necessary to develop some new types of crossovers and mutations specific for the problem.

Permutation Encoding:

Permutation encoding can be used in ordering problems, such as traveling salesman problem or task ordering problem.

1. In permutation encoding, every chromosome is a string of numbers that represent a position in a sequence.

Example:

Chromosome A	153264798
Chromosome B	856723149

2. Permutation encoding is useful for ordering problems. For some problems, crossover and mutation corrections must be made to leave the chromosome consistent.

Applications :

1. The Traveling Salesman problem:
There are cities and given distances between them. Traveling salesman has to visit all of them, but he does not want to travel more than necessary. Find a sequence of cities with a minimal traveled distance. Here, encoded chromosomes describe the order of cities the salesman visits.

2. The Eight Queens problem :
There are eight queens. Find a way to place them on a chess board so that no two queens attack each other. Here, encoding describes the position of a queen on each row.

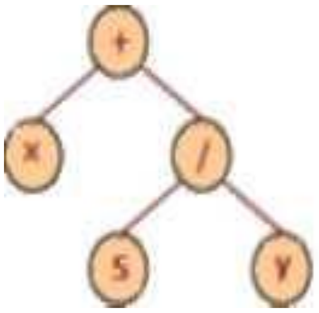
Tree Encoding:

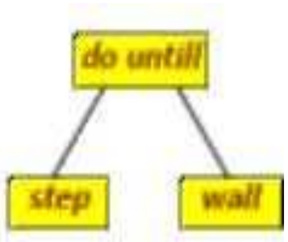
Tree encoding is used mainly for evolving programs or expressions. For genetic programming :

- In tree encoding, every chromosome is a tree of some objects, such as functions or commands in programming language.
- Tree encoding is useful for evolving programs or any other structures that can be encoded in trees.

- The crossover and mutation can be done relatively easy way. Example :

Chromosome A **Chromosome B**





(+ x (/ 5 y)) **(d o until step wall)**

Fig. Example of Chromosomes with tree encoding

Note : Tree encoding is good for evolving programs. The programming language LISP is often used. Programs in LISP can be easily parsed as a tree, so the crossover and mutation is relatively easy.

Reproduction, or Selection:

Reproduction is usually the first operator applied on population. From the population, the chromosomes are selected to be parents to crossover and produce offspring.

The problem is how to select these chromosomes ?
According to Darwin's evolution theory "survival of the fittest" - the best ones should survive and create new offspring.

- The Reproduction operators are also called Selection operators.

- Selection means extract a subset of genes from an existing population, according to any definition of quality. Every gene has a meaning, so one can derive from the gene a kind of quality measurement called fitness function. Following this quality (fitness value), selection can be performed.
- Fitness function quantifies the optimality of a solution (chromosome) so that a particular solution may be ranked against all the other solutions. The function depicts the closeness of a given 'solution' to the desired result.

Many reproduction operators exists and they all essentially do same thing. They pick from current population the strings of above average and insert their multiple copies in the mating pool in a probabilistic manner.

The most commonly used methods of selecting chromosomes for parents to crossover are :

- Roulette wheel selection.
- Rank selection.
- Boltzmann selection.
- Steady state selection.
- Tournament selection.

Roulette Wheel Selection:

Roulette-wheel selection, also known as Fitness Proportionate Selection, is a genetic operator, used for selecting potentially useful solutions for recombination.

In fitness-proportionate selection :

- the chance of an individual's being selected is proportional to its fitness, greater or less than its competitors' fitness.
- conceptually, this can be thought as a game of Roulette.

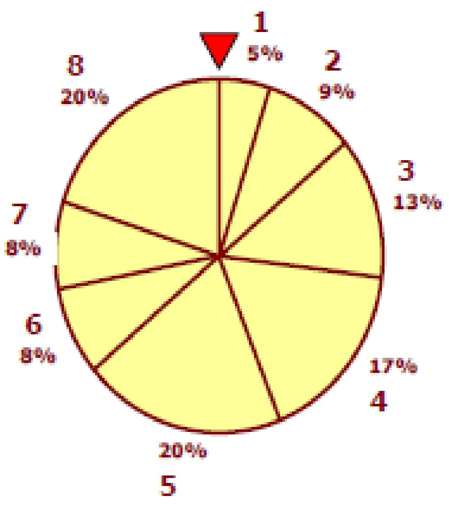


Fig. Roulette-wheel Shows 8 individual with fitness

The Roulette-wheel simulates 8 individuals with fitness values F_i, marked at its circumference; e.g.,

- the 5th individual has a higher fitness than others, so the wheel would choose the 5th individual more than other individuals .
- the fitness of the individuals is calculated as the wheel is spun $n = 8$ times, each time selecting an instance, of the string, chosen by the wheel pointer.

Probability of i th string is $p_i = F_i \;/\; (\sum_{j=1}^{n} F_j)$, where

n = no of individuals, called population size; **pi = probability** of ith string being selected; **Fi = fitness** for ith string in the population. Because the circumference of the wheel is marked according to a string's fitness, the Roulette-wheel mechanism is expected to make $\frac{F}{\bar{F}}$ copies of the ith string.

Average fitness = $\bar{F}$ F_j / n ; **Expected count** = $(n = 8) \times p_i$

Cumulative Probability$_5$ = $\sum_{i=1}^{N=5} p_i$

Example:

Evolutionary Algorithms is to maximize the function f(x) = x1 with x in the integer interval [0 , 31], i.e., x = 0, l, ... 30, 31.

1. The first step is encoding of chromosomes; use binary representation for integers; 5-bits are used to represent integers up to 31.
2. Assume that the population size is 4.
3. Generate initial population at random. They are chromosomes or genotypes; e.g., 01101, l1000, 01000, **10011.**
4. Calculate fitness value for each individual.
   (a) Decode the individual into an integer (called phenotypes), 01101 -» 13; 11000 -> 24; 01000 -> 8; 10011 -> 19;
   (b) Evaluate the fitness according to f(x) = x2,
   13 -* 169; 24 -> 576; 8 64; 19-> 361.
5. Select parents (two individuals) for crossover based on their fitness in pi. Out of many methods for selecting the best chromosomes, if roulette-wheel selection is used, then the probability of the ith string

in the population is pi = Fi / $\left(\sum_{i=1}^{n} F_i\right)$, where

Fi is fitness for the string i in the population, expressed as f(x); pi is probability of the string i being selected, n is no. of individuals in the population, is population size, n=4; n * pi is expected count

String No	Initial Population	X value	Fitness Fi f(x) = **X2**	Pi	Expected count n * Prob ***i***
1	0 110 1	13	169	0.1444	0.58
2	110 0 0	24	576	0.4923	1.97
3	0 10 0 0	8	64	0.0547	0.22
4	10 0 11	19	361	0.3085	1.23
Sum			1170	1.00	4.00
Average			293	0.25	1.00
Max			576	0.49	1.97

The string no 2 has maximum chance of selection.

Tournament Selection Example:

Tournament probability= $\frac{2n-2m+1}{n^2}$; where n is the size of the population, and m is the rank of the winning individual.

String No	Initial Population	X value	Fitness Fi f(x) = **X2**	Pi= $\frac{2n-2m+1}{n^2}$	Expected count n * Prob ***i***
1	0 110 1	13	169	$\frac{2\times4-2\times3+1}{4^2}=$ 0.1875	0.75
2	110 0 0	24	576	$\frac{2\times4-2\times1+1}{4^2}=$ 0.4375	1.75
3	0 10 0 0	8	64	$\frac{2\times4-2\times4+1}{4^2}=$ 0.0625	0.25
4	10 0 11	19	361	$\frac{2\times4-2\times2+1}{4^2}=$ 0.3125	1.25
Sum			1170	1.00	4.00
Average			293	0.25	1.00
Max			576	0.4375	1.75

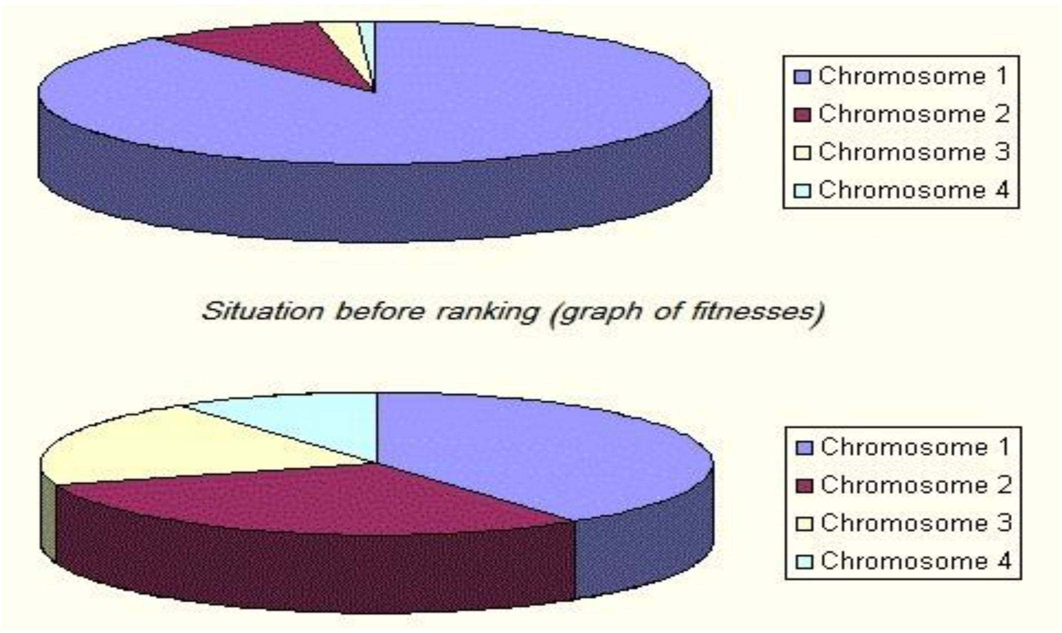
The string no 2 has maximum chance of selection.

Rank Selection:

The previous selection will have problems when the fitnesses differs very much. For example, if the best chromosome fitness is 90% of all the roulette wheel then the other chromosomes will have very few chances to be selected.

Rank selection first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness 1, second worst 2 etc. and the best will have fitness N (number of chromosomes in population).

You can see in following picture, how the situation changes after changing fitness to order number.



After this all the chromosomes have a chance to be selected. But this method can lead to slower convergence, because the best chromosomes do not differ so much from other ones.

Example:

String No	Initial Population	X value	Fitness Fi f(x) = **X2**	Rank	Probability Pi= $\frac{Rank}{Sum}$	Expected count n * Prob ***i***
1	0 110 1	13	169	2	$\frac{2}{10} = 0.2$	0.8
2	110 0 0	24	576	4	$\frac{4}{10} = 0.4$	1.6
3	0 10 0 0	8	64	1	$\frac{1}{10} = 0.1$	0.4
4	10 0 11	19	361	3	$\frac{3}{10} = 0.3$	1.2
Sum			1170	10	1	4.00
Average			293	2.5	0.25	1.00
Max			576	4	0.4	1.6

Steady-State Selection

This is not particular method of selecting parents. Main idea of this selection is that big part of chromosomes should survive to next generation.

Example:

String No	Initial Population	X value	Fitness Fi f(x) = **X2**	Steady State Selection	Selected Population
1	0 110 1	13	169	169	01101
2	110 0 0	24	576	576	11000
3	0 10 0 0	8	64	576	11000
4	10 0 11	19	361	361	10011
Sum			1170	1682	
Average			293	420.5	
Max			576	576	

Elitism

Idea of elitism has been already introduced. When creating new population by crossover and mutation, we have a big chance, that we will loose the best chromosome.

Elitism is name of method, which first copies the best chromosome (or a few best chromosomes) to new population. The rest is done in classical way. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution.