Answer

1- The correct representation of a problem is vital to its solution.

a- Taking the problem of function optimization, discuss the suitability of binary, gray code and floating point representations.

Answer: Floating point is the faster and more optimized and don't need overhead or conversion and no encoding or decoding, and gray come after it in suitability because the rating jump then binary in last.

b- Calculate the number of bits necessary to represent a precision of 4 decimal places over a range of [1, 4].

Answer:

We have 10000 value for each range from 1 to 4 , It mean 10000 from (1 - 2) and 10000 from (2 - 3)
And 10000 from (3 -4)

Then

2 power n >= 3 * 10000

we must to get n (number of bits necessary to represent a precision of 4 decimal places)

c- Write down an algorithm to convert from binary to Gray code. Use it to convert binary (1111).

Answer:

1 – n is the size of binary string code
2 – B[n] is an input array that contain binary code
3 – G[n] is an output array of gray code
4 – G[0]=B[0]
5 – foreach i=1 to n-1
G[i] = B[i]   XOR B[i-1]

BinaryCode(1111)

1	G[0] = B[0]	
10	G[i] = B[i] XOR B[i-1]	i=1
100	G[i] = B[i] XOR B[i-1]	i=2
1000	G[i] = B[i] XOR B[i-1]	i=3

then 1000 is gray code

d- Write down an algorithm to convert from Gray to binary. Use it to convert Gray (1100) to binary.

Answer:

1 – n is the size of binary string code
2 – G[n] is an input array that contain gray code
3 – B[n] is an output array of binary code
4 – B[0]=G[0]

5 – foreach i=1 to n-1
B[i] = B[i-1] XOR G[i]

GrayCode(1100)

1	B[0] = G[0]	
10	B[i] = B[i-1] XOR G[i]	i=1
100	B[i] = B[i-1] XOR G[i]	i=2
1000	B[i] = B[i-1] XOR G[i]	i=3

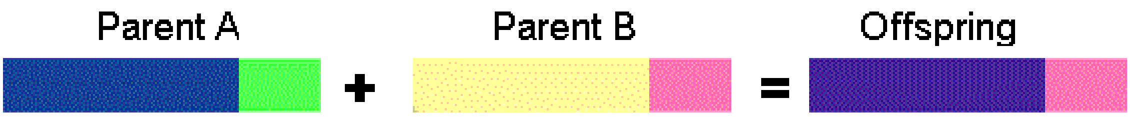
then 1000 is binary code

2- Crossover and mutation are the main operators of a Genetic Algorithm.

a- Differentiate between single-point and multiple-point crossover, on both binary and floating point representations.

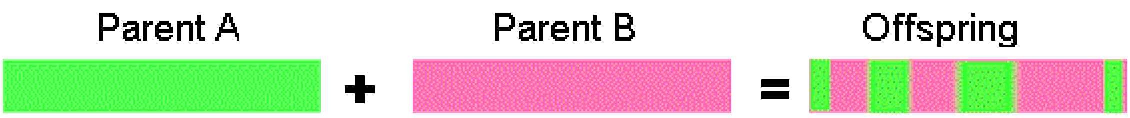
Answer:

Single point crossover - one crossover point is selected, binary string from beginning of chromosome to the crossover point is copied from one parent, the rest is copied from the second parent



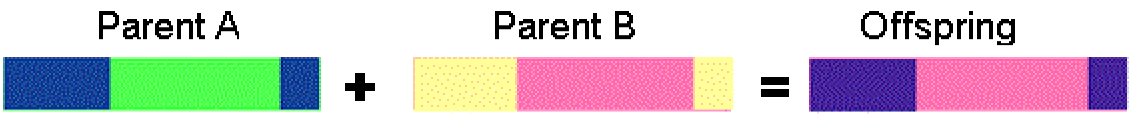
11001011+11011**111** = **11001111**

Multiple point crossover - bits are randomly copied from the first or from the second parent



1**10**010**11** + **1**10**111**01 = 11011111 b- **Show by example- using binary strings- how can a 2-point crossover be carried out.**

Two point crossover - two crossover point are selected, binary string from beginning of chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point is copied from the second parent and the rest is copied from the first parent



110010**11** + 11**0111**11 = **11011111**

c- **Explain the operation of the mutation operator on both binary and floating point representations.**

1 - Flip-bit Mutation on binary

We can switch a few randomly chosen bits from 1 to 0 or from 0 to 1. Mutation can then be following:

Original offspring 1 1101111000011110

Original offspring 2 1101100100110110

Mutated offspring 1 1100111000011110

Mutated offspring 2 1101101100110100

2 – Non-Uniform Mutation on floating point

In Non-Uniform Mutation we have the equation :

$$\Delta(t, y) = y.(1 - r^{(1-t/T)})$$

where t = current generation
T = max # of generation
b = Dependency factor
r = random

and

(y = UB – vi   or   y = vi – LB)

based on   r

if r >= 0.5

choose   y = UB – vi

else   choose   y =vi – LB

then the new vi after mutation

vi = vi + $\triangle$ (t, y)

d- Discuss the mechanics of non-uniform mutation on floating point representation- Apply using the following function:

$$\Delta(t, y) = y.(1 - r^{(1-t/T)})$$

where r is a random number from [0..1].

Answer:

where t = current generation
T = max # of generation
b = Dependency factor
r = random

and

(y = UB – vi or y = vi – LB)

based on r which r is random [0 - 1]

if r >= 0.5

choose y = UB – vi

else choose y =vi – LB

then the new vi after mutation

vi = vi + $\triangle$ (t, y)

e - Define the fitness f of bit string x with length m = 4, to be the integer represented by the binary number x. (eg. f(0011)=3, f(1111)=15). What is the average fitness of the schema 1* under f? What is the average fitness of schema 0*** under f?**

Answer:

Under schema 0*** then we have these fitness

f(0000) = 0
f(0001) = 1
f(0010) = 2
f(0011) = 3
f(0100) = 4
f(0101) = 5
f(0110) = 6
f(0111) = 7

then the average fitness of schema 0*** under f = (0 + 1 + 2 + 3 + 4 + 5 + 6 + 7) / 8

Under schema 1*** then we have these fitness

f(1000) = 8
f(1001) = 9
f(1010) = 10
f(1011) = 11
f(1100) = 12
f(1101) = 13
f(1110) = 14
f(1111) = 15

then the average fitness of schema 0*** under f = (8 + 9 + 10 + 11 + 12 + 13 + 14 + 15) / 8