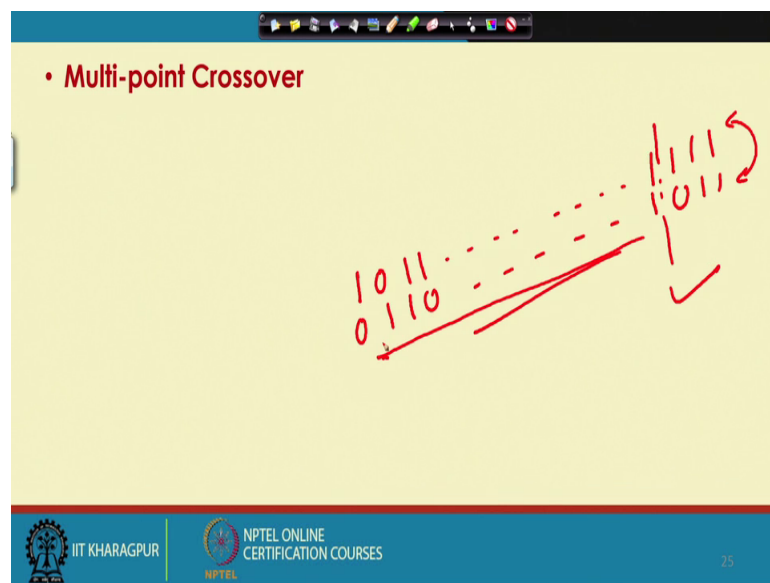


Traditional and Non-Traditional Optimization Tools
Prof. D. K. Pratihar
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture - 06
Binary – Coded Generic Algorithm (BCGA) (Contd.)

Let me start with another crossover operator; name, multi-point crossover. Now before we start discussing the principle of this particular crossover operator, let me take one example supposing that I am going to solve the optimization problem involving 10 real variables. And to represent each real variable supposing that I am using 10 bits. So, the GA string will be 10 multiplied by 10, 100 bits long. Now let me just show that particular the GA string.

(Refer Slide Time: 01:01)



Now, the GA string will look like this, supposing that this is the 100 bits long.

Now, at the GA string is 100 bits long. So, there will be 99 places for selecting the crossover site, if I use the single point crossover; now supposing that fortunately or unfortunately. So, we have selected the crossover site which is nothing, but here. So, this is nothing, but the crossover site in single point crossover. Now according to the principle of single point crossover, the bits which are lying on the left hand side of the crossover site, there will be no change and the bits which are lying on the right hand side. So, there will be swap in; and if you just follow this particular principle of single point

crossover. So, there is a possibility that in the children solution, there will be no change no diversity due to this particular the single point crossover. Because here, on this particular left side bits there will be no change.

So, we may not get the required diversity in the children solution compared to the parents. Just to remove this particular difficulty of the single point crossover, the concept of the multi point crossover has come. Now here actually what I do is, we try to select a number of crossover site at random using the random number generator. Now let us see what happens here.

(Refer Slide Time: 02:59)

• **Multi-point Crossover**

Here, the bits lying between alternate pairs of sites are interchanged.

```
101|1000|0111|0011|011|10 } Parents
011|1011|0001|1000|101|00

1011011 0111 1000 01100
0111000 0001 0011 10110
```

The slide shows two parent bit strings: 1011000011100110110 and 01110110001100010100. Five crossover sites are marked with vertical bars at positions 3, 6, 9, 12, and 15. The resulting child strings are 10110110111100001100 and 01110000001001110110. The bottom of the slide features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of the presenter.

Now, supposing that these 2 parents are going to participate in multipoint crossover, and as I told the crossover sites multiple number of crossover sites are selected at random. Now here I am selecting 1, 2, 3, 4, 5 crossover sites. Now let us see how to find out the children solution from these 2 parents. The principle is very simple, now what I do is first we concentrate on the leftmost crossover site; that means, this particular crossover site and the bits which are lying on the left hand side of the first crossover site, there will be no change and the bits which are lying between the first and second, here. So, there will be some swapping.

The next is your the bits lying between the second and third. So, there will be no change then the bits lying between the third and fourth. So, there will be swapping and the bits lying between fourth and fifth will remain the same, and the bits which are lying on the

right hand side of the last crossover site. So, there will be some swapping. And due to that, you will be getting the children solution which is nothing, but this. So, this is actually the children solutions which will be getting using the concept of this multi point crossover. Now here as I select a number of crossover sites at random. So, there is a possibility that, there will be some sort of diversification in the children solution compared to the parent's solutions. Now this is actually the merit or the plus point of the multipoint crossover or in comparison with the single point the crossover.

So, this particular crossover operator is more efficient particularly for the problem having a large number of variables. Now I am just going to discuss with another or crossover operator which is very popular and this is known as uniform cross over. Now this uniform crossover is actually a slightly modified version, and I should say a sophisticated version of this particular the multipoint crossover. Now let us see; how does it work.

(Refer Slide Time: 05:44)

Parents

```

10110000111001101110
01110110001100010100
  
```

Let us assume that 2nd, 4th, 5th, 8th, 9th, 12th, 18th and 20th bit positions are selected for swapping.

Children

```

11110000011101101110
00110110101000010100
  
```

template

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now the principle is very simple, supposing that these 2 are the parents which are going to participate in this particular the uniform cross over. So, what I do is we start from the leftmost bit position and; that means, I am here. So, here we take the help of some set of coin tossing with probability 0.5 for appearing head. Now if head appears then there will be swapping of the bits and if it is a failure then the bits will remain the same.

Now, this particular procedure is followed at each of the bit position. Now in computer program, now how to implement this particular the technique; the principle is very simple. So, what we do is at each of the bit position we just take the help of 1 random number generator. Now this particular random number generator will generate a number lying between 0 and 0.5. So, if sorry lying between 0 and 1.0. Now if the number generated by the random number generator is found to lie between 0 and 0.5. So, that will be a success; that means, success mean the head will appear and there will be swapping of the bits. So, this particular principle is followed just to get the children solutions from the parents. Now if I just consider this 2 parents and if I consider that head has appeared at a particular bit position, now let us see how to find out the children solution.

Let us assume that the head has occurred at the position that is the second one, the fourth one, the fifth one, 8 9th 12, 18, and 20th position; that means, in this positions there is a success in coin tossing that mean the head has appeared, there will be swapping. Now if I see the parent position, this is the parent position second. So, this is 0 1. So, 0 1. So, it is a success. So, this will become 1 and 0 then comes here the fourth one. So, this is the fourth position. So, this is the success. So, there will be a swapping. So, it is 1 1. So, here on the children solution this will remain same as 1 1. Then comes your; the fifth position is a success, once again there will be swapping. So, 0 0 will become 0 0 here, now this particular procedure is followed.

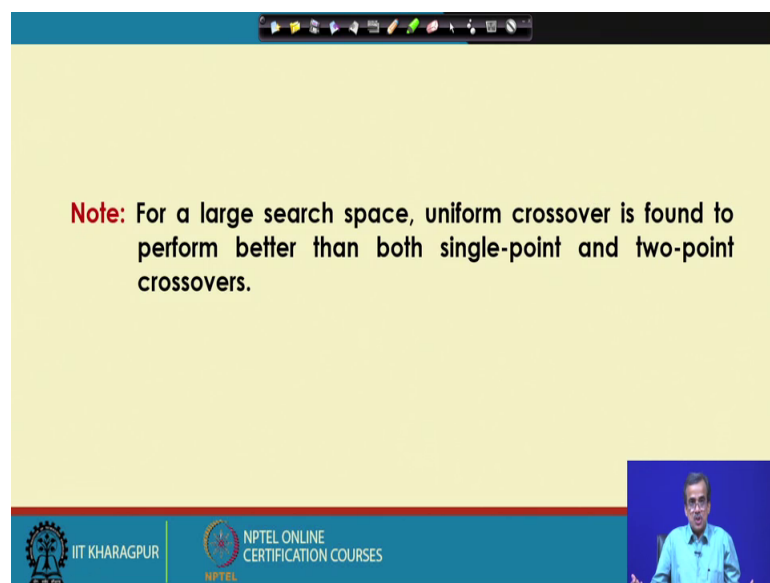
And if we follow this particular principle, starting from the 2 parents we will be getting the children solution like this, now how to implement. So, the method which I have already discuss is one of the possible methods. Now if you see the literature there is another method with the help of which. So, this particular the uniform crossover can be implemented very easily. The method is as follows; now supposing that say I have got 20 bits and will have to implement the uniform crossover. Now what I do is we take the help of a template, now this template is nothing, but actually or this is a just like your; the plate sort of thing, where there are some marked position 20 positions for the 20 bits.

Now, supposing that. So, this is the template now here on this template. So, this is the position for the first bit, this is the position for second bit and. So, on and might be this is the position for the twentieth bit. Now what I do is, here at all these 20 positions we generate 1 and 0 using the random number generator. Now supposing that there is 1 here,

there is 0 here, there is 1 here and in between there are some zeros and ones. Now the principle is as follows if there is a 1 at a particular position there will be a swapping of the bits and if there is 0.

So, there will be no swapping at the bits will remain intact and this particular procedure is followed for all the 20 bits positions. And by using this template I can also or implement the concept of this particular the uniform crossover and here in this particular uniform crossover starting from the 2 parents. So, I will be getting the children solution.

(Refer Slide Time: 10:52)



Note: For a large search space, uniform crossover is found to perform better than both single-point and two-point crossovers.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, if I compare the performance of this particular your uniform crossover with your the single point crossover or say 2 point crossover, now we are getting some advantage in uniform crossover in the sense that the problem having the large number of variables. So, this uniform crossover is found to perform better compared to both single point and 2 point crossovers. Now we have seen the way the different crossover operators are working and using the crossover operator how to get the children solution. And as I told several time using this crossover operator, there will be exchange of properties and we will be getting some diversity in the children solutions. Now let us try to concentrate on another very powerful operator which is known as mutation.

Now, if you see what happens in biology? So, this biological mutation is well known and let me take a very simple example of this biological mutation. Now generally the crows are black in color, but if you just try to find out if and if a fortunate enough if and if you

get 1 crow, which is having white color. So, that could be due to with the biological mutation. The concept of this biological mutation has been copied in genetic algorithm in the artificial way and here also we use the principle of mutation. Now let us see how to implement this particular the mutation.

(Refer Slide Time: 12:38)

• **Step5: Mutation**

- (i) Bring a local change over the current solution
- (ii) 1 is converted into 0 and vice-versa
- (iii) Helps the GA to search the globally optimal solution
- (iv) Mutation probability p_m is generally kept to a low value

$$\frac{0.1}{L} \leq p_m \leq \frac{1}{L}$$

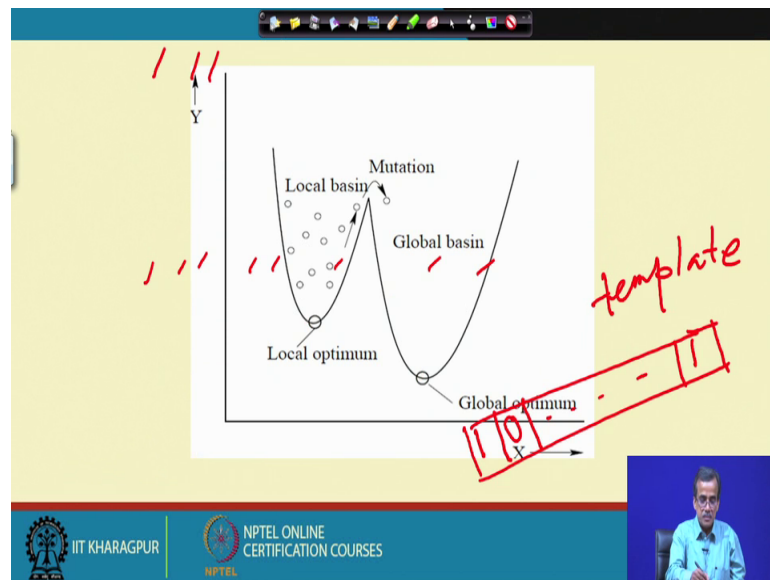
L=100
0.001 ≤ p_m ≤ 0.01

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 29

Now, this mutation can bring local change around the current solution, and by doing that it can help to overcome the local minima problem. Now I am just going to take 1 example just to show you; how does it work. So, here in mutation this is the bitwise nutrition. So, what I do is if there is 1 that is converted into 0 and vice versa.

Now, as I told it helps the GA to overcome the local minima problem.

(Refer Slide Time: 13:18)



Now, let us see; how can it overcome this particular the local minima problem. Let me take the example of a function having only one variable. Now here in this particular function say Y is a function of only one variable $f(x)$. Now if I plot and supposing that I am getting this type of local basin and that type of global basin, now here in genetic algorithm what I do is we start with a population of solution selected at random, now supposing that unfortunately all the initial solutions are lying in this particular the local basin.

And if the solutions lying on the local basin and if I run GA for a large number of iteration, there is a possibility GA is going to hit this particular as the optimal solution. But this is not the globally optimal solution, this is the locally optimal solution whereas, the globally optimal solution could be here, because this is a minimization problem. So, this is the globally minimum solution and that is the global locally minimum solution.

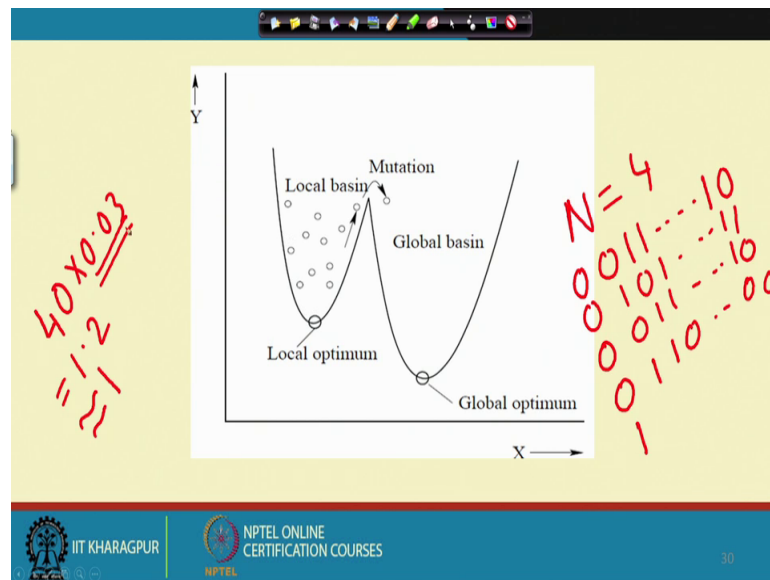
Now, if I run GA for a large number of generation, there is no guarantee that I will it will be able to hit the globally minimum solution. Now it will be able to hit the globally minimum solution if and only if we can push at least one solution from local basin to the global basin. Now it is this particular operator the mutation operator, which can push at least 1 solution from the local basin to the global basin, and if it can push 1 solution to the global basin there is a possibility GA through a number of iteration it is going to hit this particular the globally minimum solution.

Now, this is the way actually the GA is going to help the GA. So, the mutation is going to help the GA to come out of the local minima the problem. Now let me go back to the previous slide just to show you. Now how to select that particular the probability of mutation; now here if you see the probability of mutation place a great role. Now if the p_m that is the probability of mutation is found to be or if it is selected to be a very low value.

The very purpose of using the mutation may not be soft, on the other hand if the probability of mutation is selected to be a very high value. So, this GA will become equivalent to the random walk method or the random search method; that means, will have to select this particular the p_m in a very careful way. And the thumb rule to select actually this particular p_m is as follows, that p_m should lie between 0.1 divided by capital L and 1 divided by capital L . Now here this L is nothing, but the length of the GA string now supposing that this L is equals to 100 , there are 100 bits in the GA string and here actually what I do is. So, 0.1 divided by L is nothing, but 0.001 and 1 divided by L is nothing, but 0.01 and this particular p_m should lie between this. So, the p_m should lie between 0.001 to 0.01 . So, generally we as I as I told generally we keep the value of this mutation probability to a low value, to we just select a low value.

Now, let me try to understand the utility of this particular the mutation operator in a slightly different way. Now supposing that I have got one GA whose task is to find out the optimal solution and for simplicity, let me assume that the population size that is your; the N that is found to be only 4 .

(Refer Slide Time: 17:55)



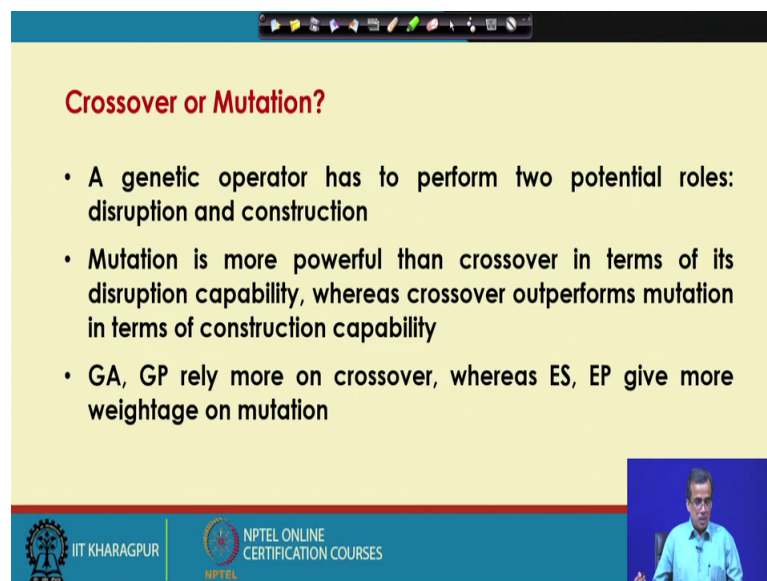
Say capital N is equals to 4 and let me take a very hypothetical example supposing that the GA strings are as follows, and in each of the GA string supposing that I am using only 10 bits. The GA string is as follows say 0 0 1 1 dot, dot, dot and the last is 1 0; the next is 0 1 0 1 1 1 the next is your 0 0 1 1 1 0 0 1 1 0 0 0 supposing that these are the GA strings and they are going to participate in crossover. Now if I a single point or 2 point or multipoint or uniform crossover. So, he will be getting some children solution. Now if I use this crossover operator, there is a possibility that I am going to miss the value 1 at the leftmost bit position.

Now, if I a single point crossover. So, at the leftmost bit position I will not be getting 1, if I use 2 point multipoint uniform at the left most bit position I will not be getting 1, but supposing that my globally optimal solution is such that if I want to indicate that, there must be 1 at the leftmost bit position; that means, if I want to hit that globally optimal solution, the condition is the leftmost bit position should be one; that means, there should be 1 here. Now the reality is not even a single the crossover operator which I have discussed we will be able to generate 1 on the leftmost bit position then how to overcome this particular problem. To overcome this particular problem, it is the mutation that is the bitwise mutation which is going to help us and there is a possibility that I will be getting 1 at the left most bit position using this particular the mutation operator.

So, this is actually the real strength of this particular the mutation operator and that is why we should use the mutation. Now supposing that I have got only such 4 GA string, and in each GA string are got only 10 bits; now 10 multiplied by 4, I have got 40 bits. Now if I have got only 40 bits and supposing that the mutation probability is say 0 point say 03, if it is 0.03. Now this is equal to your 1.2. 40 multiplied by 0.03. So, this is nothing, but 1.2 that means, out of this 40 bits, there will be mutation probabilistically only on 1 bit this is equivalent to one, but the as it is probabilistic. So, there could be mutation on 2 bits also, and at the same time there could be a chance that there will be no mutation. If I consider the mutation probability as 0.03 now; that means, we have understood that the mutation probability has got some role at this operator has got big role in the working principle of this particular the genetic algorithm.

Now, let us try to find out like whose contribution is more and let us try to compare the contribution of this particular the crossover and your; the mutation operator.

(Refer Slide Time: 22:03)



Crossover or Mutation?

- A genetic operator has to perform two potential roles: disruption and construction
- Mutation is more powerful than crossover in terms of its disruption capability, whereas crossover outperforms mutation in terms of construction capability
- GA, GP rely more on crossover, whereas ES, EP give more weightage on mutation

The slide also features a small video inset of a speaker in the bottom right corner and logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES at the bottom.

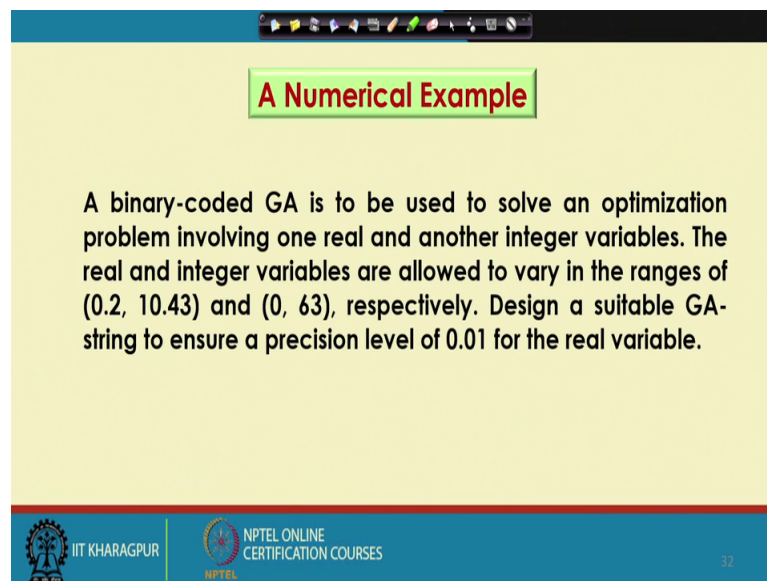
And my question is should I go for crossover or only mutation or both. Now to answer this actually the whole community of this nontraditional optimization tools, particularly those who are working on GA, actually the whole community was divided into 2 subgroups. Now 1 group used to believe that there should be crossover as well as mutation and another group used to believe there should be only mutation crossover is redundant. Now to answer that actually I am just going to discuss on this, this is so. Now,

if I want to have a very efficient GA search there should be construction as well as disruption. That means, will have to construct some GA string will have to destruct also because we want some diversification in the search process.

Now, if you see in terms of disruption capability, now this particular your; the mutation operator is more powerful in comparison with the crossover operator. But if you see in terms of the construction capability now the crossover operator is preferred to the mutation operator and that is why in genetic algorithm and genetic programming we actually considered both your crossover and mutation, but we give slightly more weightage on the crossover compared to the mutation.

On the other hand the techniques like evolutionary strategy or evolutionary programming they give more weightage on the mutation operator. In fact, they do not use the concept of this particular the crossover operator.

(Refer Slide Time: 24:29)



A Numerical Example

A binary-coded GA is to be used to solve an optimization problem involving one real and another integer variables. The real and integer variables are allowed to vary in the ranges of (0.2, 10.43) and (0, 63), respectively. Design a suitable GA-string to ensure a precision level of 0.01 for the real variable.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

32

Now, I am just going to solve one numerical example just to help you in understanding like how to carry out the calculation little bit in binary coded GA. Now here what I am going to discuss like how many bits are to be assigned to represent the real variable and to represent the integer variable, if you want a desired level of precision. Now the way I have form this particular the numerical example is as follows, I am just going to use 1 binary coded GA to solve 1 optimization problem having 2 variables.

Now, out of 2 variables there is one integer variable and the second one is the real variable now real variable is having the range 0.2 to 10.4 three say and the integer variable is having the range 0 to 63, now how to design a suitable GA string to ensure the precision level of 0.01 for the real variable and the precision level of 1 for the integer variable.

(Refer Slide Time: 25:43)

Solution:

For real variable

$$l_1 = \log_2 \left(\frac{x_1^{\max} - x_1^{\min}}{\epsilon} \right)$$

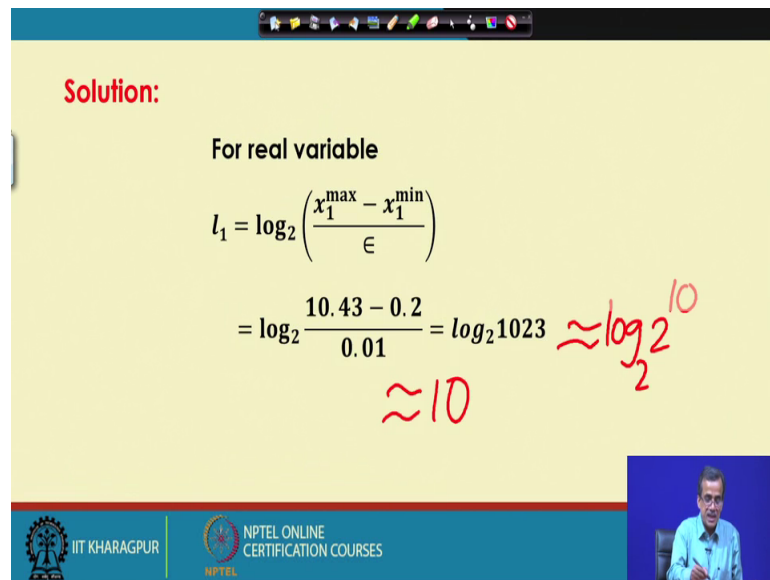
$y = f(z)$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us try to solve this particular the numerical example. Now how to determine how many bits are to be assigned to represent this particular the real variable. Now this particular formula we have already got we have already discussed how to derive that you have seen, now I am just going to use this particular the formula just to find out this how many bits are to be assigned to represent the real variable.

Now, here actually what I do is what is what I do is, we try to calculate this l_1 and to calculate this l_1 .

(Refer Slide Time: 26:30)



Solution:

For real variable

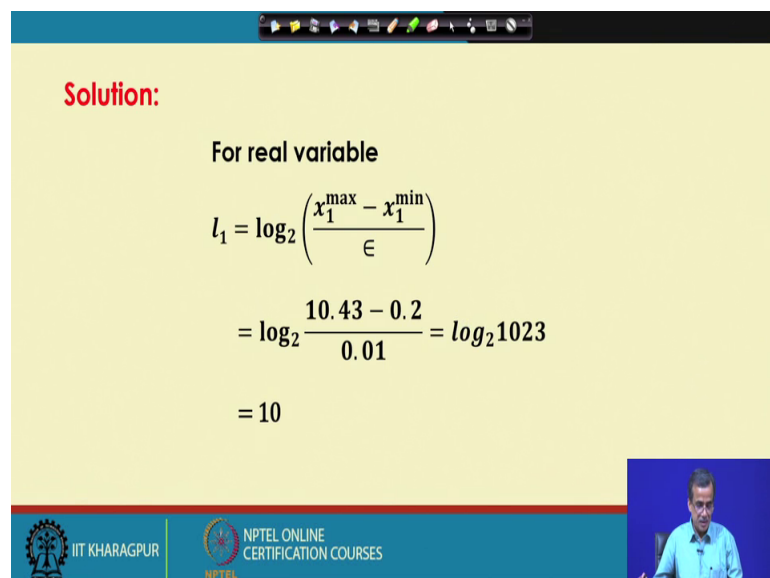
$$l_1 = \log_2 \left(\frac{x_1^{\max} - x_1^{\min}}{\epsilon} \right)$$
$$= \log_2 \frac{10.43 - 0.2}{0.01} = \log_2 1023 \approx \log_2 2^{10} \approx 10$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, we know the x_1 maximum that is 10.43 and we know x_1 minimum is 0.2 and the desired accuracy level is 0.01 and if you calculate this will become log base 2, $\log_2 1023$ and that is approximately equal to log base 2 two raised to the power 10 and that is approximately equal to your 10.

So, So, the number of bits which I am going to assign to represent this particular real variable is 10; that means, we are going to assign 10 bits to represent this particular the real variable.

(Refer Slide Time: 27:16)



Solution:

For real variable

$$l_1 = \log_2 \left(\frac{x_1^{\max} - x_1^{\min}}{\epsilon} \right)$$
$$= \log_2 \frac{10.43 - 0.2}{0.01} = \log_2 1023$$
$$= 10$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

(Refer Slide Time: 27:17)

For integer variable

$$l_2 = \log_2 \left(\frac{x_2^{\max} - x_2^{\min}}{\epsilon} \right)$$
$$= \log_2 \left(\frac{63 - 0}{1} \right) \approx 6$$
$$= \log_2 2^6 = 6$$

Handwritten diagram: A rectangular box representing a GA string is divided into two sections. The left section is labeled "10 bits" and "real". The right section is labeled "6 bits" and "integer".

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 34

Similarly, if you want to determine how many bits are to be assigned to represent the integer variable, I can use this particular formula. And if I just substitute the values x_2 maximum is 63, x_2 minimum is 0 and here the accuracy level is 1 and if you calculate this will become log base 2 63 which is approximately equal to the log base 2 63 is approximately equal to 6.4. So, this is nothing, but 2 raise to the power 6 and that is equals to 64. So, I will have to assign 6 bits to represent this particular the integer variable and once I have got this particular information, now I can design this particular your; the GA string. Supposing this is the GA string. So, the first the 10 bits will represent your; the real variable. So, this is going to represent the real the 10 bits and the remaining 6 bits are going to represent this particular your; the integer variable. So, this is the way actually we can represent the GA string in order to handle the problem having 1 real and 1 integer that is nothing, but the mixed integer optimization problem.

(Refer Slide Time: 28:55)

For integer variable

$$l_2 = \log_2 \left(\frac{x_2^{\max} - x_2^{\min}}{\epsilon} \right)$$
$$= \log_2 \left(\frac{63 - 0}{1} \right) \approx 6$$

Therefore, 10 and 6 bits are required to represent the real and integer variables, respectively.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 34

Thank you.