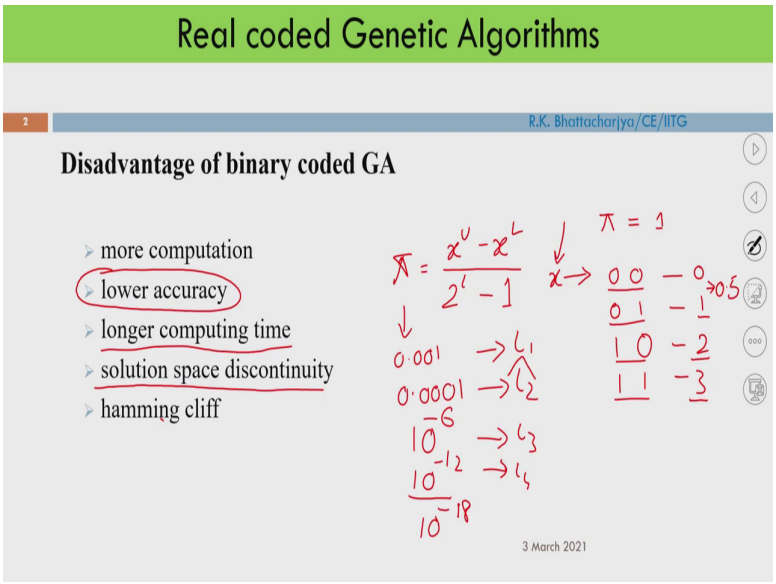
**Optimization Methods for Civil Engineering**
**Prof. Rajib Kumar Bhattacharjya**
**Department of Civil Engineering**
**Indian Institute of Technology, Guwahati**

**Lecture - 20**
**Real Coded Genetic Algorithms**

I welcome again to this course on Optimization Methods for Civil Engineering. So, already we have discussed about genetic algorithm. So, right now we know what is genetic algorithm; then we also discuss the different operators using genetic algorithm. And mainly so far we have discussed the binary coded genetic algorithm.

So, what we have done here. So, we have converted the solution to a binary string in order to apply crossover and mutation operators ok. So, let us see what is the disadvantage of binary coded genetic algorithm the main disadvantage of binary coded genetic algorithm.

(Refer Slide Time: 01:08)




So, I have listed some of the disadvantages ok. So, these are more competition. So, the computational complexity is more, because we are dealing with the string. So, the variable has been converted to binary string and if a and if you have seen that if you need if your precision is more or if you want more precision then the length of the string will be very large.

So, in that case it is really difficult to handle and then lower accuracy. So, what I have said here that accuracy has to be predefined; that means, I as I said that we are using the mapping something like that that pi equal to x upper minus x lower then 2 to the power l minus 1. So, this is the precision.

So, this precision you have to define already means it is a; it is a predefined precision and based on that the string length will also depend. So, if I use a pi value of 0.001 then I will get

a string length suppose L 1 and if I use that 0.0001 and this is suppose L 2. So, certainly this is L 2 is greater than L 1.

So, therefore, if you are using suppose if I my precision against 10 to the power minus 6 or 10 to the power minus 12 this length basically what you are getting L 3 and L 4 will be more ok. So, therefore, the we can say that it is either not accurate or accuracy has to be defined already.

So, in order to get the string length and as a result, so you need large computational time ok. So, if your precision is more suppose if I want 10 to the power minus 12 or 10 to the power minus 18. If I need precision at that level, so computational time will be very large. Another main disadvantage is that solution space discontinuity ok. So, what is solution space discontinuity? As already we have discussed that we are converting suppose I am using pi value of 1.

And my string length is 2. So, then suppose it is 0 0. So, this is there and I am getting and this value is 0 ok then 0 1 then this value is 1 then 1 0 is 2 and 1 1 is 3. So, what is happening suppose if I take the pi equal to 1; that means, my precision is 1 and if I am taking 2 bit then I can define these are the these are the possible values.

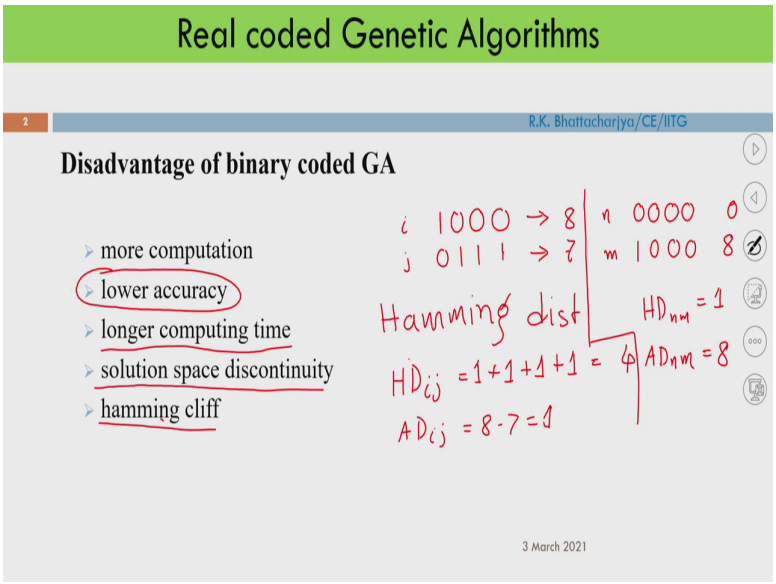
So, if the string the binary string is 0 0 then I will get 0 if it is 0 1 I will get 1 if it is one 0 I will get 2 and if it is 1 1 I will get 3. So, that means, in this case precision is 1. So, with this 2 bit and pi equal to 1. So, I will get either 0 or 1 or 2 or 3. So, that means, the if my if the variable is suppose this is the x variable and this x variable I have converted to binary bit with 2 bit string so, in that case the possible values are 0 1 2 3.

So, with this 2 bit I will not get if the x value optimal value is suppose 0.05 ok. So, I will not get this particular value either I will get 0 or I will get 1. So, what we are doing here. So, we are converting the continuous variable.

So, this is the continuous variable x is a continuous variable and this variable we have convert converted to discrete variable and that is basically where what we are doing the solution or

space discontinuity is the solution space is not continuous now after conversion to binary string. So, this is one of the issues the another problem is the hamming cliff problem. So, let me explain what is hamming cliff suppose if I have a string like this.

(Refer Slide Time: 05:51)




So, I have a string 1 0 0 0 and suppose another string this is 0 1 1 1 ok, so these are two string. Now the decoded value of these two strings. So, decoded value is this is 8 and this is 7 ok. Now, if I calculate the what is the difference between these two string. That means, if I say the what is the distance between these two solution ok.

So, first solution is 1 0 0 0 and second solution is 0 1 1 1. So, now, decoded value is 8 and 7. So, I can calculate the distance and this distance is 1 or I can say the difference between these two solution is 1, if I calculate the hamming distance, if I calculate the hamming distance ok.

So, what is hamming distance that if there is a similarity between the bits. So, in that case I will say the record is 0 and if there is if there is no similarity. So, 1 is 0 and 1 is 1 then record will be 1 basically. So, if I calculate the hamming distance of this particular string that is suppose if I say this string is i and this string is j. Now, if I calculate the hamming distance between i and j. So, what is this distance?

So, this is if I look at the first bit. So, this is not similar. So, therefore, record is 1 ok then for the second bit that is not similar. So, 1 is 0 and 1 is 1. So, this is again 1 then third bit is also 1 not similar and 4 bit is also not 1. So, therefore, the hamming distance is 4 ok. Now what is the real distance between these two? The actual distance ok I say that actual distance between i and j. So, this is 8 minus 7, so this equal to 1 ok.

So, for these two particular solutions the hamming distance is 4 and actual distance is 1. So, that means, if you look at the binary conversion the binary bit these two string or these two strings are not similar ok. So, they are far away, but if you consider their decoded value they two are very near these two solutions are very near.

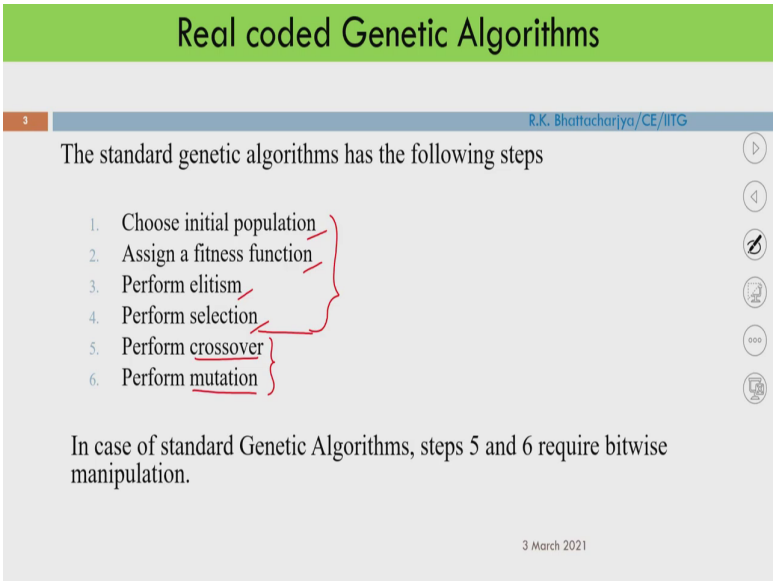
So, let us take another example problem suppose if I take 1 is 0 0 0 0. So, let us take another example problem and decoded value is 0 and if I take suppose 1 0 0 0 and here decoded value is 8. So, therefore, if I calculate the hamming distance ok. So, this is n and this is m. So, if I calculate that H hamming distance between n and m. So, this is your 1 ok.

So, because there is a this first bit is not similar other 3 bits are similar. So, therefore, this is 1 and other records are 0. So, hamming distance is 1, but the actual distance is between n and m is 8 ok. So, you just see in binary bit if you consider from binary conversion. So, these two strings are similar hamming distance is only 1, but actual distance is 8.

So, this problem is known as hamming cliff problem. So, this is this for n and in this case what may happen that sometime you need to alter more than one your bit in order to get a nearby solution ok. So, this problem is associated with the binary conversion.

So, you can also say that this is also a disadvantage of binary coded G A. So, these are some of the disadvantage of binary coded G A. Now, let us see how we can overcome this disadvantage suppose can we take can we apply the crossover and mutation directly on the real value.

(Refer Slide Time: 10:51)



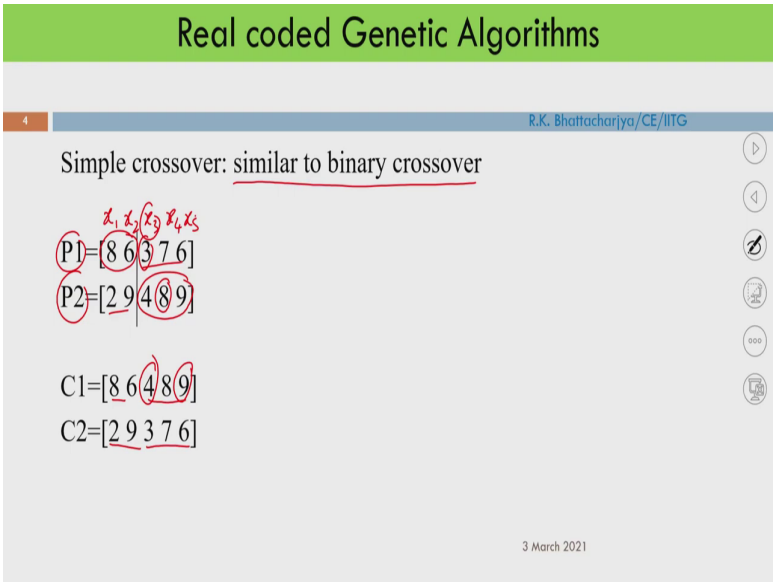

So, a standard genetic algorithm has the following steps the steps are the shows initial solution ok. So, I think now you know how to select the initial solution, how to generate the initial solution and then you define a fitness function. That means, I have to define the fitness value to each of the solution and then you have to perform elitism ok.

So, you want to preserve the better individual and then perform selection and then you have to perform these two operators that is crossover operator and mutation operator. Now, if you look at this part these 6 steps. So, step 1 to step 4 ok. So, up to here I do not need the binary

conversion ok binary conversion is not required, but I need the binary conversion to perform crossover and to perform mutation ok.

Now if I can have a different crossover that without doing binary cross without doing binary conversion if I can implement the crossover and mutation of operator then I actually I can overcome the disadvantages of binary coded G A. Let us discuss some of the technique that can be applied to perform crossover and mutation on the real values without converting to the binary string.

(Refer Slide Time: 12:19)



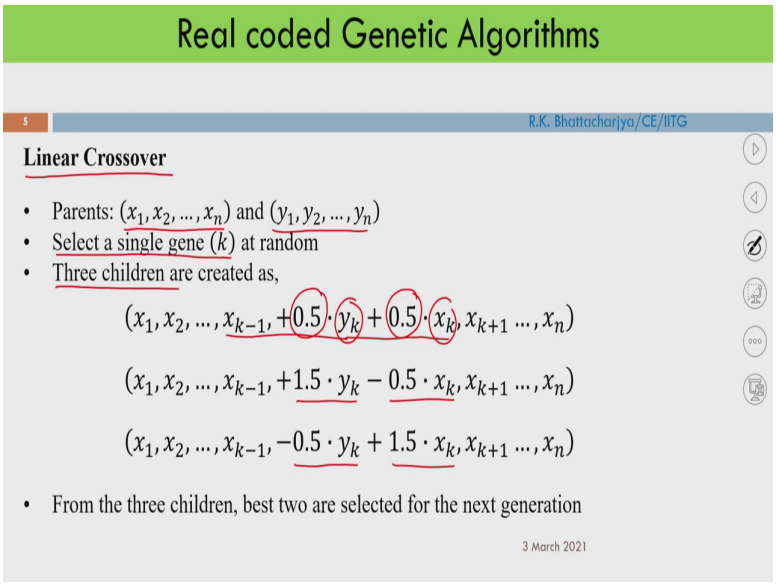

So, the first one is it is a simple crossover. So, this is similar to binary crossover suppose I have 2 parents ok. So, P 1 and P 2 and these are all variable suppose this is your x 1 and this is x 2 this is x 3 and this is x4 and this is x5. So, I have total 5 variable and the values are 8 6 3 7 6 and here 2 9 4 8 9.

So, one of the simplest way is that you select the crossover side randomly. So, you are selecting a crossover side randomly and then this swap this variable ok. So, what I can get that now 86 will come with 4 8 6. So, I will get 8 6 4 9 and then this 2 9 will come with this. So, I will be getting 2 9 3 7 6. So, I am getting 2 children ok C 1 and C 2 these are new solution, but question is that. So, what we are doing here? So, we are just sending the variable.

So, actually we are not getting a new value of any of the variables. So, earlier what happened x 3 as a 3 value. So, now, you are getting this is 4, but you are not getting near three suppose 3.1 or 3.2 or something like that or 4.1 and 4.2 something like that similarly for the this x 4 variable.

So, I am getting 8 earlier it was 7 and now I am getting 9, but actually I am not getting a new solution ok. So, this is a different combination of the initial your solution ok. So, I am not getting a new solution. So, therefore, the performance of this particular crossover is not good ok.

(Refer Slide Time: 14:16)




So, let us see another crossover. So, we call it linear crossover. So, in this case so, we have 2 parents that is x 1, x 2 up to xn and and y 1, y 2, yn. So, we are taking two solution randomly and then we will select a single gene ok. So, out of this x1 to xn, so 1 to n. So, let us select this is kth gene and that is randomly.

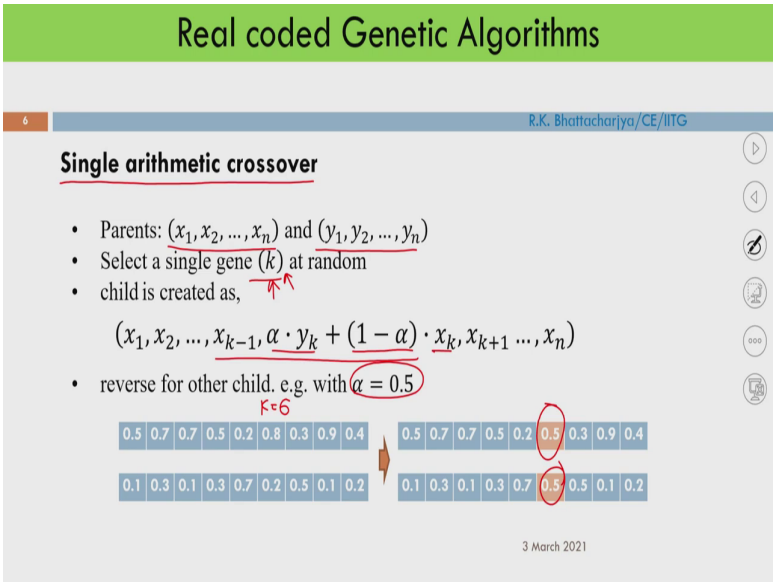
So, I can generate a random number between 1 and n and I can get this value of k and then what we have done here. So, in this case we are creating 3 children using these two solutions ok. So, first one is in the kth solution because this random number is generated as kth and the kth solution. So, what we are doing here. So, we are creating three solution like this this is 0.5 of yk and 0.5 of xk ok.

Similarly, 1.5 of yk minus 0.5 of xk, then minus 0.5 of yk plus 1.5 of xk. So, I am getting this 3 solution and here you just see. So, for this kth variable, so you are getting some new

solution ok so, out of these 3 children. So, best two are selected for the next generation. So, crossover in crossover what we are doing we are trying to create two new solution from the parent solution.

So, from 2 parents we are trying to create 2 children. So, here we are creating 3 solutions and out of these 3, 2 are selected for the next generation best 2 are selected for the next generation. So, this crossover operator is better than the simple one. So, here we are actually getting a new value of a particular variable. So, this is your linear crossover.

(Refer Slide Time: 16:21)



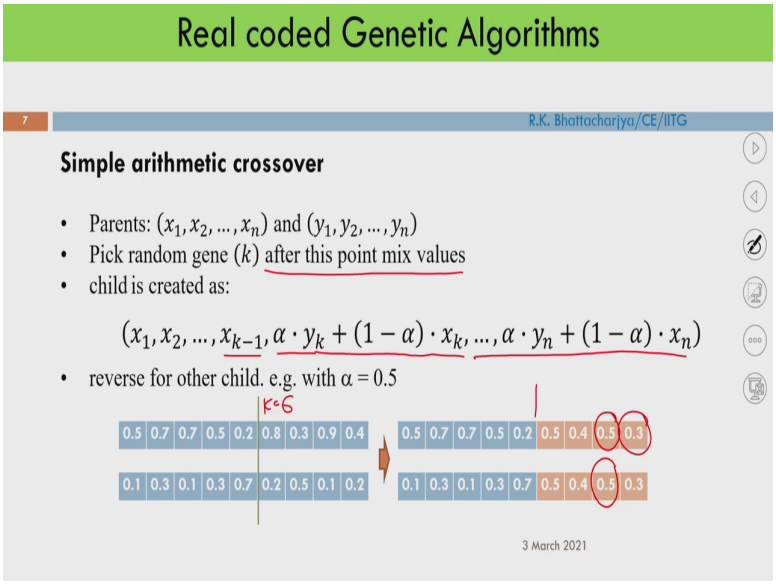

Now, next there is another one we call it single arithmetic crossover. So, here also this is what we are doing. So, we are taking two parents that is x 1, x 2 up to xn and y1, y2 to yn and. So, here we are selecting a single gene randomly. So, we are generating a random number

between 1 and n and suppose this is your the value is k and then we are creating a child using the using this process.

So, what we are doing here? So, we are multiplying alpha into yk for the kth gene ok alpha equal to alpha into yk plus 1 minus alpha into xk. So, this is similar to the earlier one. So, earlier one we have taken 0.5 then 0.5 minus 1.5 then my minus 0.5 and 1.5, but here what we are doing? We are we are just taking alpha into yk plus 1 minus alpha into yk.

So, now you can take a suitable value of alpha suppose for 0.5, what is happening? So, if the k is the sixth one ok this is. So, this is k equal to 6 ok sixth one. So, then what is happening this is if I take 0.5 then this value will be sent to 0.5 and 0.5. So, earlier it was 0.8 and 0.2. So, new value after crossover we are getting 0.5 and 0.5 ok. So, this is one of the operators. So, here also we are getting a new value of that particular variable.

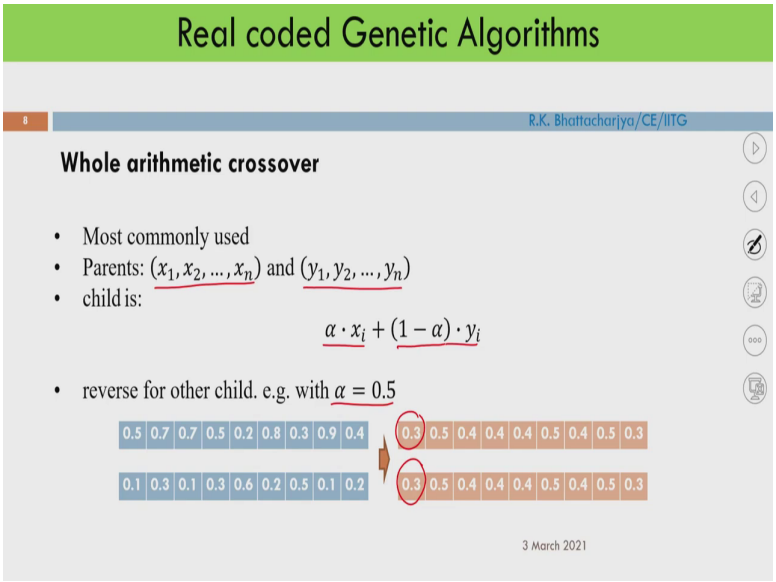
(Refer Slide Time: 18:11)



Similarly this is simple arithmetic crossover again. So, what we can do basically. So, in earlier case we have actually sends the value of the kth variable, but in this case what we have done. So, from the kth after this point so, we are mixing the values ok suppose here from basically kth 1 ok. So, up to k minus 1 we are not doing anything and after k so, we are actually multiplying this by alpha y k plus 1 minus alpha xk upto the last variable ok.

So, here this is k equal to 6, so you can see. So, after that we are doing the crossover. So, here the k equal to 6 I am getting 0.5 and 0.5 for the 7 variable I am getting 0.4 and 0.4 and for the 8 variable I am getting 0.5 and 0.5 and for the last variable I am getting 0.3 and 0.3. So, what we are doing? We are mixing the values after the kth variable ok.

(Refer Slide Time: 19:24)

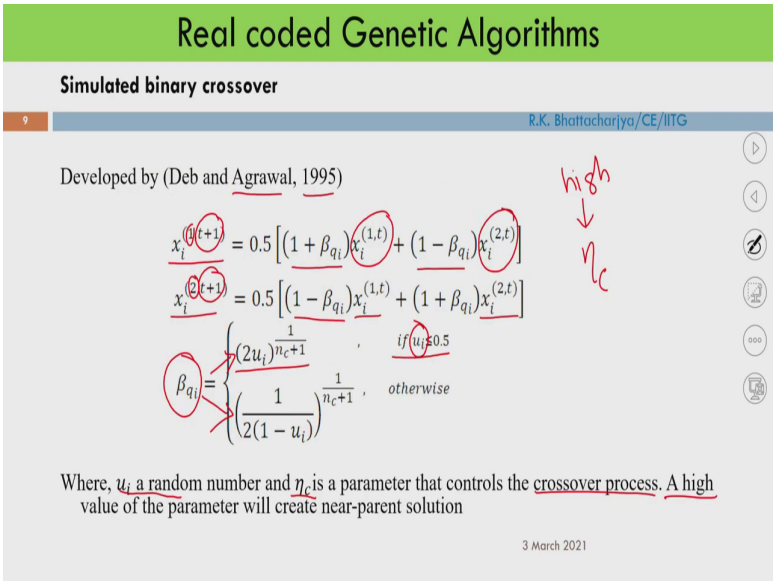



So, this is another crossover and also we can do what. So, we can implement it in all the for all the variables. So, here also we have selected two parents x and y and then you are

implementing alpha xi plus 1 minus alpha yi to all the variables here. So, alpha equal to for alpha equal to 0.5.

So, here actually you are changing all of them. So, first one it was 0.5 and 1. So, you are getting 0.3 and 0.3. So, like that you are implementing it for all the variable. So, these are some of the simple crossover. So, arithmetic crossover you can implement.

(Refer Slide Time: 20:05)




Now, this is another one. So, given by professor Kalyan Mahadev and his student Agrawal, so in 1995. So, what he has done he has used a probability distribution and then he is trying to generate two children. So, that is xi at t plus 1 that means, for the next generation ok this is the first child and this is the second child.

So, this is given by 0.5 1 plus beta qi into xi. So, this is one parent and this is another parent ok and this is 1 minus beta q this is xi this is the second parent similarly. So, here 0.5 this is 1 minus beta q and xi the parent 1 and this is parent 2 into 1 plus beta q.

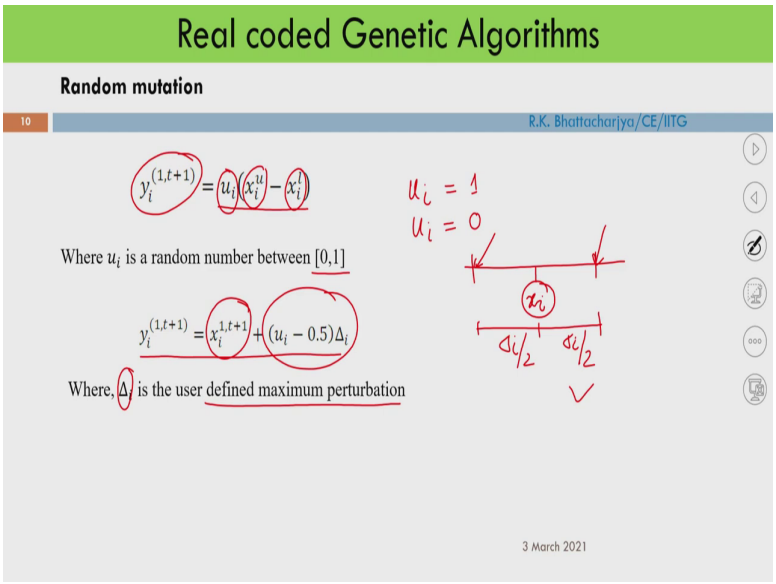
So, he has calculated beta q using a probability distribution and beta q is twice ui to the power 1 by eta c plus 1 and if the ui. So, ui is a random number so, you are generating a random number between 0 and 1 and if ui is less than 0.5. So, you are using beta q is this and otherwise beta q value is this ok. So, here ui is a random number and eta c is a parameter of that distribution that controls the crossover process.

So, this is basically you have some control over the crossover process a high value of eta ok. So, high value of eta c will create near parent solution. So, what will be done? If you are taking a high value of eta c. So, high value of eta c so, high value of eta c then it will create a child near the parents and if you are taking low value then it will create a your children will be far away from the parent.

If you look at the binary crossover. So, as I have already discussed that whatever solution you are creating. So, these 2 solution; that means, 2 children from parents 2 parents you are creating 2 solutions and 2 children and these 2 solutions are near the parents. So, that we have discussed. So, crossover is actually creating 2 children near the parents and now using this crossover operator given by professor Kalyan Mahadev. So, what we can do by putting a high value of eta c. So, I can create a solution near the parent.

So, I have some control over it. So, I can choose a suitable value of eta c and can actually generate a solution either near the parents or away from the parents. So, this crossover is basically simulating the binary crossover and therefore, this crossover is known as simulated binary crossover.

(Refer Slide Time: 23:13)




Now, let us discuss some of the real mutation ok. So, in case of binary mutation. So, we are going bit by bit; that means, what we are doing suppose we are generating a random number and if the random number is less than the mutation probability. So, we are mutating it, mutating it means we are sending the value of that particular bit if it is 0 we are putting in 1 and if it is 1 we are putting it 0.

So, we are changing the value of that particular bit ok. So, in case of real coded G A. So, we can generate a new solution using some random distribution, let us see how we can implement the mutation operator on real values. So, first one is the random mutation. So, what we can do basically I can create a random value between lower bound and upper bound.

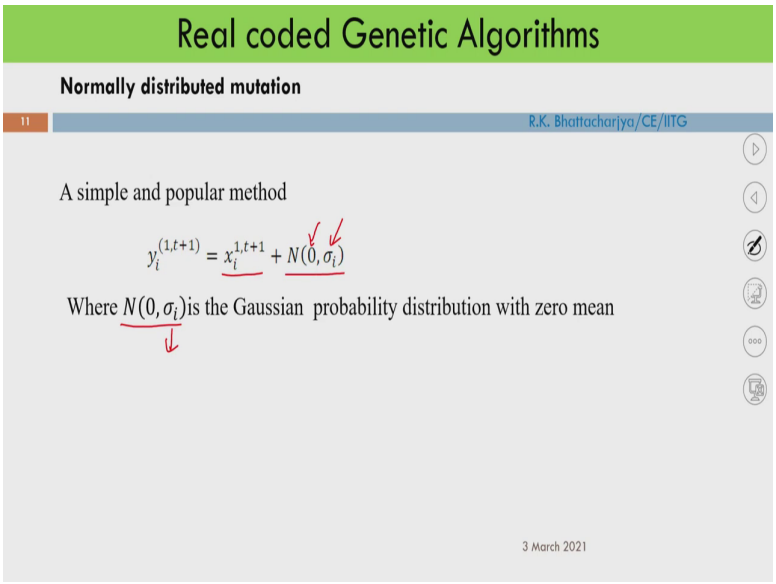
So, here xu is the upper bound and xl is the lower bound and ui is a random number between 0 and 1. So, I can generate a random number between 0 and 1 and I can create a mutated value between upper bound and lower bound and this is the mutated value.

So, this is yi t plus 1. So, in this case as I have said so, I can create a random number and I can do that, but here what is happening. So, I am not taking any solution. So, actually I am not creating or I am not mutating a solution I am creating a new solution here. So, in order to mutate a solution I can use this. So, this is the solution which I would like to mutate and this is the perturbation I can actually add. So, here delta i is the user defined maximum perturbation.

So, suppose this is your solution. So, this is the solution xi. So, xi is the solution and this is delta i by 2 and this is delta i by 2. So, now, u is a random number. So, ui is a random number and if the random number is 1 that is the maximum value of this random number, so then what will happen?

So, you are getting 0.5 delta I, so that means, you will get this value ok. So, this value is 0.5 delta i and if you are if ui is 0 then you are getting minus 0.5 delta i. So, therefore, using a random number between 0 and 1 so, I can generate a new solution using this mutation operator.

(Refer Slide Time: 26:21)

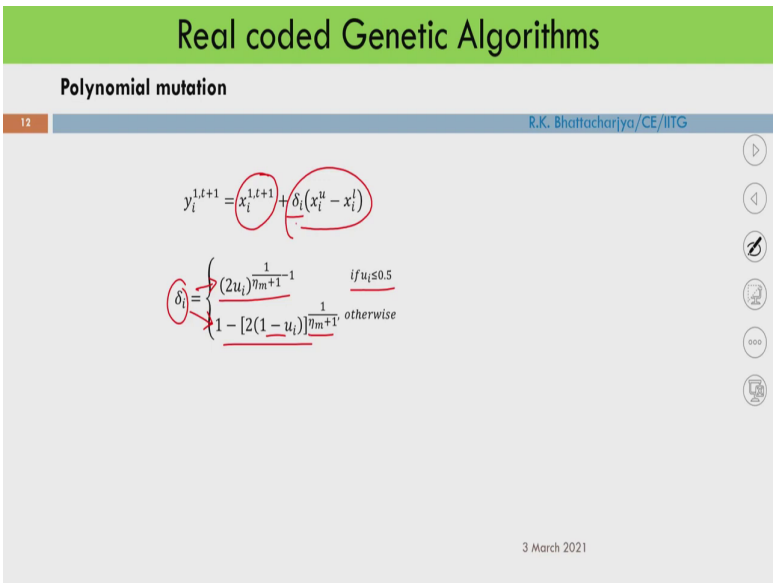



Another way is that. So, I can also use a normal distribution to generate a new solution or to get a mutated solution. So, here suppose I would like to mutate this xi t plus 1. So, this is the variable I would like to mutate and then I am putting a I am generating a random number with mean zero and standard deviation sigma i and I can create a new mutated solution for xi. Now here the strength is the sigma. So, if you are taking higher value of sigma.

So, in that case, so your mutated solution may be far away from the parent solution ok and if you are using a smaller value of sigma you are creating a solution near the original solution. So, I can take a suitable value of sigma and I can create a solution either near that particular solution or away from that particular solution, but if you are taking a smaller value of sigma then actually you are doing the local source. So, you are basically looking is there any better solution near the particular solution.

But if you are using higher value of sigma so, in that case you are also exploring the region where optima may be there. So, you are exploring the other region far away from the current solution and you are looking for a global optimal solution.

(Refer Slide Time: 27:55)




So, another one is the polynomial mutation. So, this was given by professor Kalyan Mahadev he has used a probability distribution to generate this delta i and. So, this is the perturbation part. So, you are adding you are with the current solution.

So, you are adding this part 2 get another solution or to get a mutated solution and here this is delta i into x upper minus x lower and this delta i is given by 2 to 2 into u to the power 1 by eta m plus 1 minus 1.

So, if ui is less than 0.5. So, ui is a random number, so you are generating and if the random number is less than 0.5. So, you are delta is calculated using this equation or otherwise you are calculating you are calculating the delta i using this equation and which is 1 minus 2 into 1 minus ui and this is 2 to the power 1 by eta m.

So, here eta m is a parameter of this particular mutation operator and using eta m you can actually control the mutated value ok. So, I can use this, but for my work ok whatever example I will discuss. So, I will be using this polynomial mutation and also I will be using simulated binary crossover for real coded G A ok.

So, for real coded G A so, we will use simulated binary crossover and I and here actually using eta c you can control the crossover operation. So, that we can do so, I will be using this particular crossover operator and then I will also be using polynomial mutation.

So, this is all about the real coded genetic algorithm. So, here there is no difference between the selection operator of real coded genetic algorithm and binary coded genetic algorithm. So, selection operator is the same, but only difference with the crossover and mutation operator. So, we are not using binary crossover or binary mutation. So, rather we are using real crossover and real mutation.

So, I have discussed different crossover operator and finally, we have discussed the simulated binary crossover. So, why we are calling it simulated? Because it is actually the performance of this crossover is similar to binary crossover, but it is a; it is a real crossover and then similarly that polynomial mutation also similar to the binary mutation.

So, I can I mean we have a control over it we have a control over the mutation operation as well as the crossover operation. So, I can create a solution near the parents or I can also create a solution away from the parents. So, other things the selection operator elitism operator then fitness function. So, this is similar for both real coded g a G A and binary coded G A.

So, in the next class I will discuss the application of G A for multi model function ok. So, in case of multimodal function there are more than one optimal solution. So, we have more than one optimal solution.

Now, if you are applying the classical optimization technique. So, you will get one of them. So, one of the local optimal solution, but and if you are applying simple genetic algorithm. So, you will get the global optimal solution.

So, you will not get the other local optimal solution. So, enter j population will converge to the global optimal solution. So, in the next class we will discuss how this simple genetic algorithm can be applied or can be implemented for finding all the local optimal solutions of a multimodal function.

Thank you.