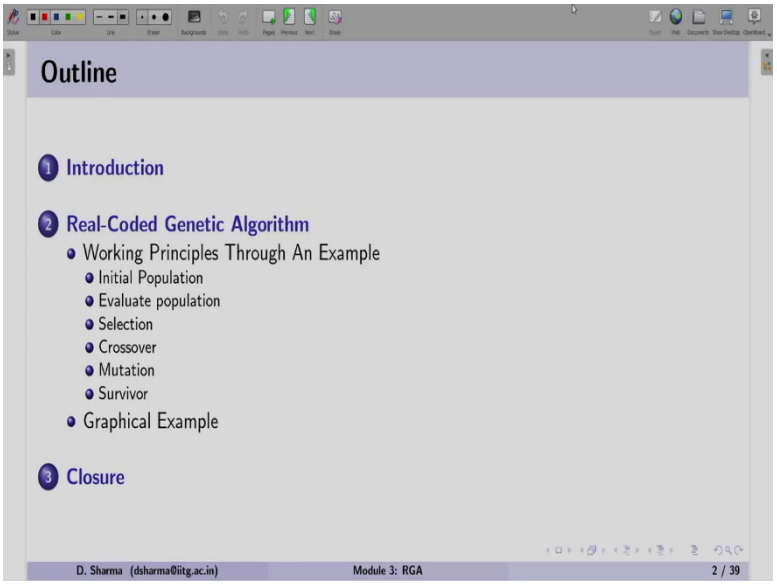
Evolutionary Computation for Single and Multi-Objective Optimization

Dr. Deepak Sharma
Department of Mechanical Engineering
Indian Institute of Technology, Guwahati

Module – 03
Lecture – 05
Real-Coded Genetic Algorithm

Welcome to module 3 in this module we will discuss Real Coded Genetic Algorithm. In this particular session we will be covering the introduction about real coded GA then we will understand RGA with the example of Rosenbrock function.

(Refer Slide Time: 00:53)



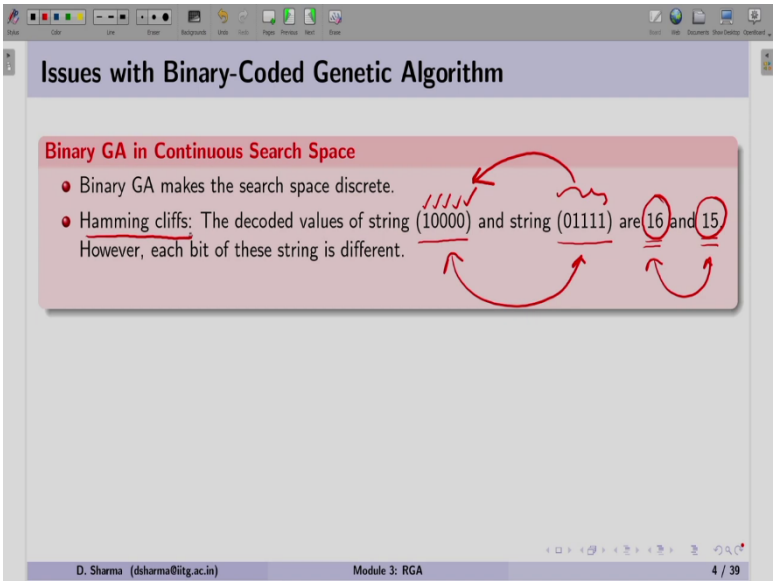
So, similar to binary coded GA we will generate the initial population, then we will evaluate it, after that we will be performing selection operator followed by crossover and mutation operators. Once the off spring is off spring population is generated we will combine them in the survival stage.

We will perform all the hand calculation for one generation and the same example we will see as a graphical illustration. So, and finally, we will close this session. So, let us start with the introduction. Now we have already gone through the binary coded GA.

So, in the last session as a recap, we have started with the generalized framework and binary coded GA was fitted into the generalized framework. We understand the binary coded GA

with the operators that are involved for optimizing the function. Here when we solve a problem using binary coded GA there are certain issues.

(Refer Slide Time: 02:19)



And when we are solving a continuous search space using binary coded GA what we can see here that a binary coded GA makes the search space discrete. When we say it is discrete meaning that, we are finding the real values with the help of binary string. Now when we are decoding a binary string we know that we get some integer value say for example, 14, 15, 16, 17.

So, when those integer values or the decoded value of binary string are used with scaling function then we find the value of a variable and the next variable. So, there is in between these two in the in between these two values we cannot get any value meaning that, the binary string is actually making our continuous search space into the discrete search space.

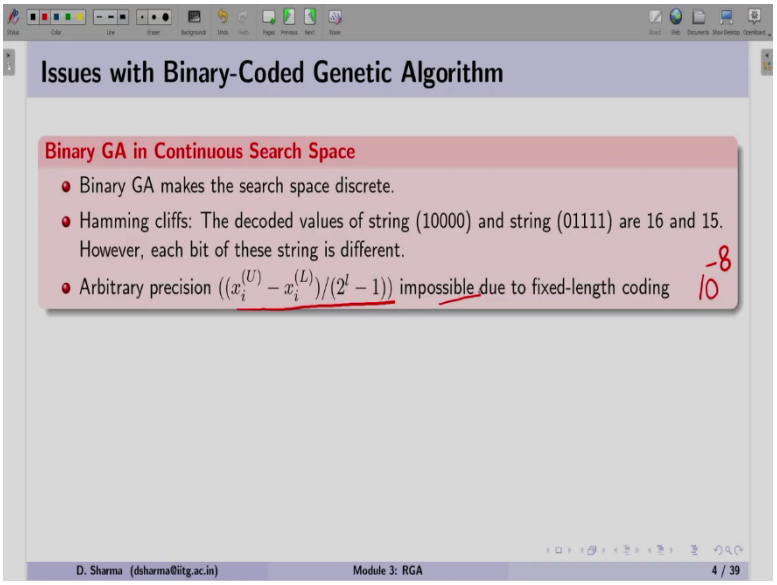
Another problem as we can see here that for example, we have a binary string given. So, 2 binary strings are given here in this case if we decode this binary string we can get a value of 16. The another binary string if we decode the value is 15. Now if I am going to compare these 2 values these 2 values our neighbours in a real; in a real sense or a in a real number.

However, if you use a these decoded value in our scaling function then 15 and 16 will give us the neighbouring value of x i, but if I compare the binary strings here you can see that we

have to change each and every bit to get this string. Say suppose currently the decoded value of a 15 we are using in a binary GA and assume that the optimize lying at the 16.

So, currently it means that this is the binary string we are using. Since we are using it and the optimize lying on 16 then this string has to convert into the 10000 meaning that we have to mutate or change each and every bit.

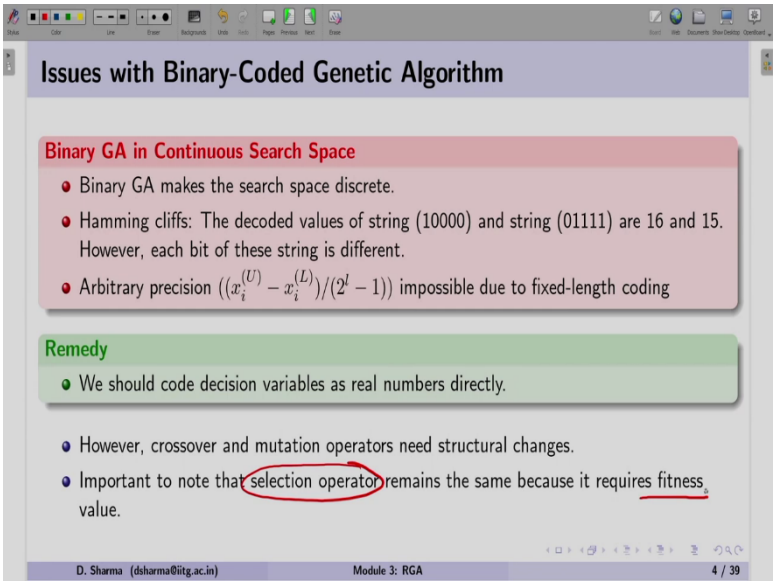
(Refer Slide Time: 04:55)



This particular problem is referred as a hamming cliff problem, another issue with the binary GA is the arbitrary precision. Now this particular formula we know that we used it for the further precision in the variable say x i. Now if suppose we are we want the precision in terms of say 10 to the power minus 8.

So, in that case you can see the binary string will be very large and if we are solving a multivariable problem then the binary string over all chromosome length including the binary string for all real numbers will be huge. So, practically it is impossible to have this fixed length coding with the arbitrary precision. So, what is the remedy here?

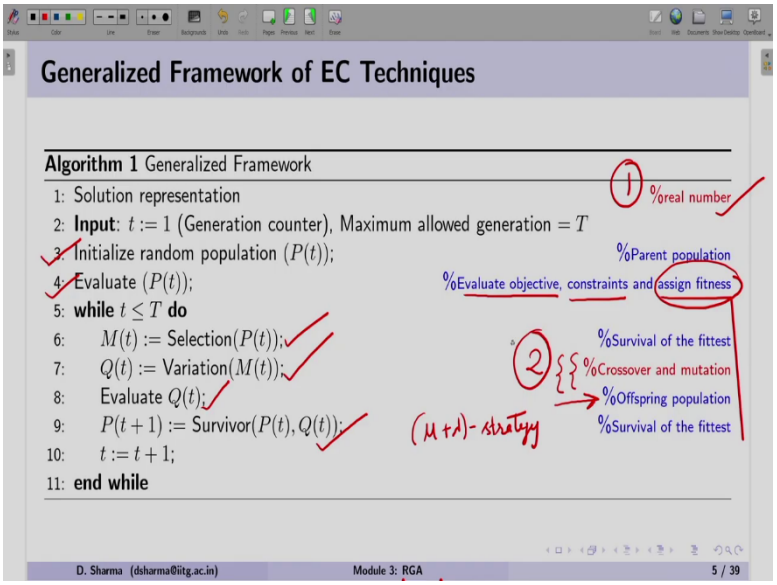
(Refer Slide Time: 05:47)



We can actually code this decision variable as a real numbers. So, when so we do not want any binary string here. So, we can remove it and we should use the real numbers as 2.53 or 1.67. Now once we are doing once we are coding the variable as a real number, we have to concentrate on crossover and mutation. Why? Because since we do not have any binary string now so, we have to come up with new operators that will work with the real numbers.

But, at the same time we have to see that we do not have to think about the selection operator it is because the selection operator requires only the fitness value. So, we do not have to worry about it. So, all the operators we have learned during binary coded GA same selection operators can be used with real coded GA.

(Refer Slide Time: 06:57)



Now, we will come to the generalized framework and let see where we have to change our algorithm or the code. Now the first step is the solution representation since it is RGA or real coded GA we have to represent all of them as a real number, then we have a certain input at step number 2.

Now in step number 3 we generalize the initial population, initial population meaning that we have to tell the x 1, x 2 and x 3 values should be the real numbers. So, within the range of those variables we can generate and similarly we can have a population initial random population thereafter we evaluate.

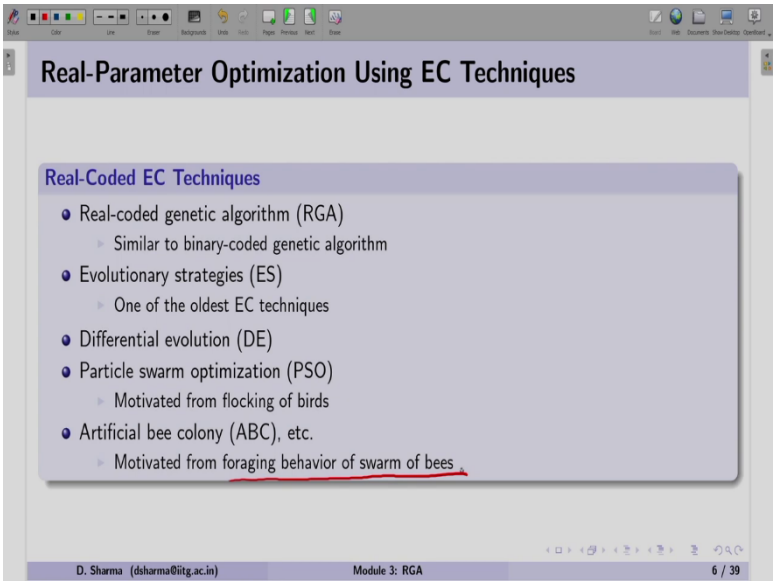
Now, the evaluation will not change with RGA it is because we already have x1, x2, x3 value directly and we have to just fit into our objective function. So, therefore, our we the objective function or the constraint that will not change, similarly the fitness assignment can be kept same. Now we are in the loop of the number of generation here, now in step 6 we have to perform the selection operator.

So, whether we perform binary tournament selection or fitness proportionate selection all selection operators need only the fitness value. So, fitness value is already calculated in the previous step. So, therefore, we can use the operator as it is. So, we do not have to change the selection operator for real coded GA. Thereafter any step number 7 we have variation. As we have understood that now we are working with the real numbers. So, therefore, we have to come up with new kind of operators.

So, therefore, I made crossover and mutation in red colour because these operators need to be changed thereafter we evaluate the population. So, this is similar to evaluating the objective function, constraint and assigning the fitness in this case the offspring solution. So, the we call this as a offspring solution or a offspring population and thereafter we have a survival stage at step number 9.

Now, in this again if you if we take mu plus lambda strategy that also depends on the fitness value only; so, in this case the mu plus lambda strategy or mu lambda strategy that we have understood with binary coded GA that will also remain the same. So, what we have identified through this framework? So, the places where we have we need to change is we have to represent our solution as a real number and 2nd we have to work on our crossover and mutation operators.

(Refer Slide Time: 10:25)



There are various real coded EC techniques. First one is the RGA we say it is called Real Coded Genetic Algorithm you will realize that this algorithm is similar to binary coded GA, only we have to change the crossover and mutation operator and solutions are directly represented in real numbers.

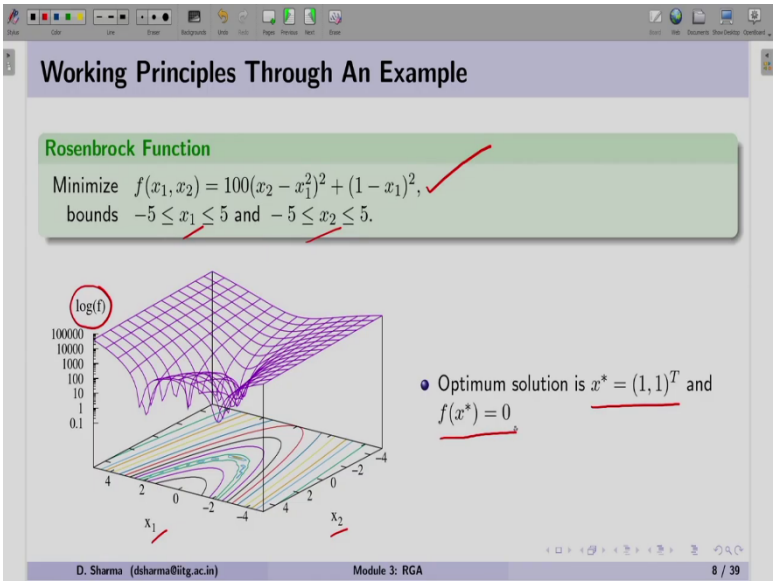
We can have evolutionary strategies as I have mentioned here this is one of the oldest EC techniques available. So, in early 60s genetic algorithm and EC techniques were evolved at different places. Than we have differential evolution; differential evolution works on the

vectors for example, mutant vectors etcetera and we also have particle swarm optimization, both these algorithms are very effective in solving a real parameter optimization problem.

As we know this particle swarm optimization it is motivated from the flocking of the birds. Similarly, we can have other algorithm like artificial bee colony we say its call ABC, it is motivated from foraging behaviour of swarm of the bees, in the literature we have various other EC techniques that can be used for real parameter optimization.

We will be focusing on few important algorithms or EC techniques. Now with this introduction let us move to real coded genetic algorithm. This algorithm we will be understanding with the help of an example. So, we have taken the same example of Rosenbrock function.

(Refer Slide Time: 12:29)

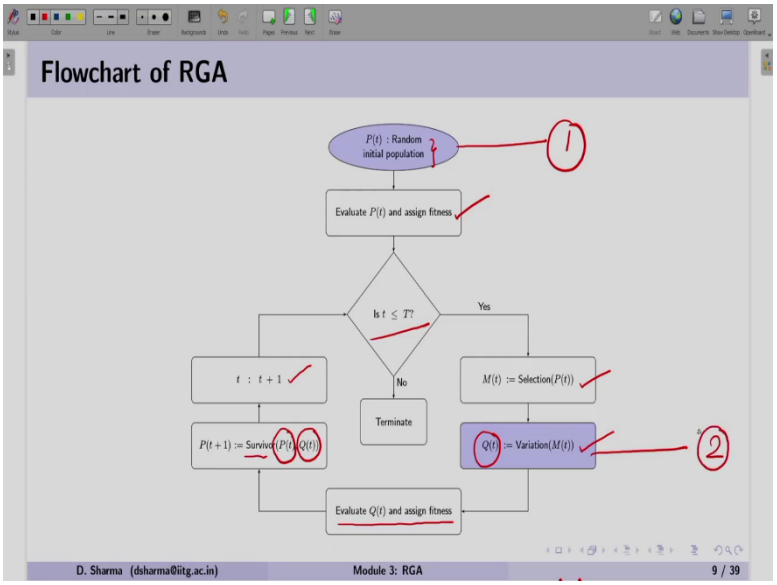


$$Minimize\ f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2$$
$$bounds\ -5 \leq x_1 \leq 5\ and\ -5 \leq x_2 \leq 5$$

So, for 2 variable function you can see the objective is pay objective function which we want to minimize and the variable bounds on x 1 and x 2 are also written. On the left hand side we

have plot the logarithmic of function as a third axis and we have x 1 and x 2 plane. Looking at this cut looking at this surface we can see that this particular function has many local optimas. However, the global optimize lying at 1 and a 1 and the function value at the optimal point is 0.

(Refer Slide Time: 13:11)



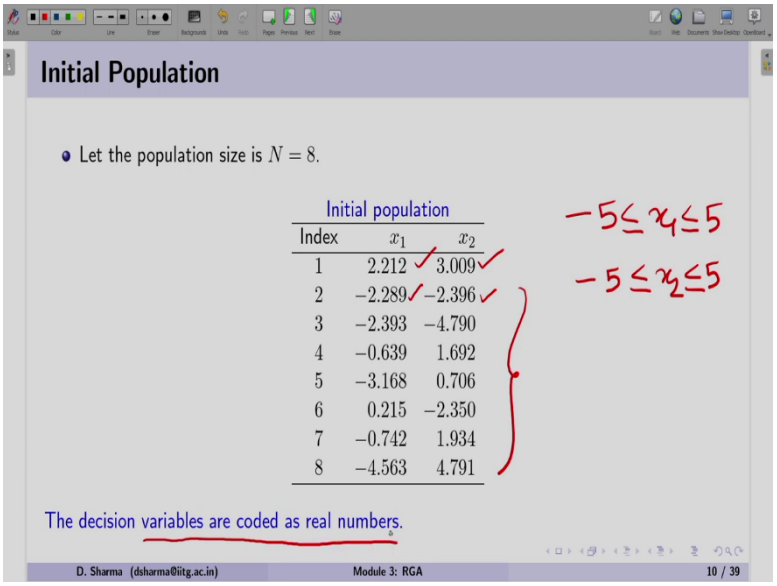
We will follow this RGA with the help of this flow chart. So, this flow chart we discussed earlier, in this flow chart you can see that it is similar to the generalized framework. So, we start with the random initial population where we are generating a population with the real numbers then we are going to evaluate the population and assign a fitness.

Here this is the decision box and we look for the number of generation when t is smaller than T we do selection, now at this stage we know various kind of selection operators we can use any one of them. Thereafter we have to perform the variation operator on the mating pool generated by the selection operator, thereafter the offspring population Q is generated then we have to evaluate this population and assign fitness to it.

The once at this particular stage you can see we have parent population, we have offspring population. So, if I assume that the size of a offspring parent population is 100, offspring population is also 100, then 100 plus 100 will become 200. So, this survival stage will help us to select best 100 solution for the next generation, we increase the counter by 1 and then we keep on moving in this loop till the termination condition gets satisfied.

Once it is satisfy we report our result. Here I have again made the 2 blocks. So, this is the area 1 where we need to change our algorithm and this is the 2nd part where we have to change our algorithm.

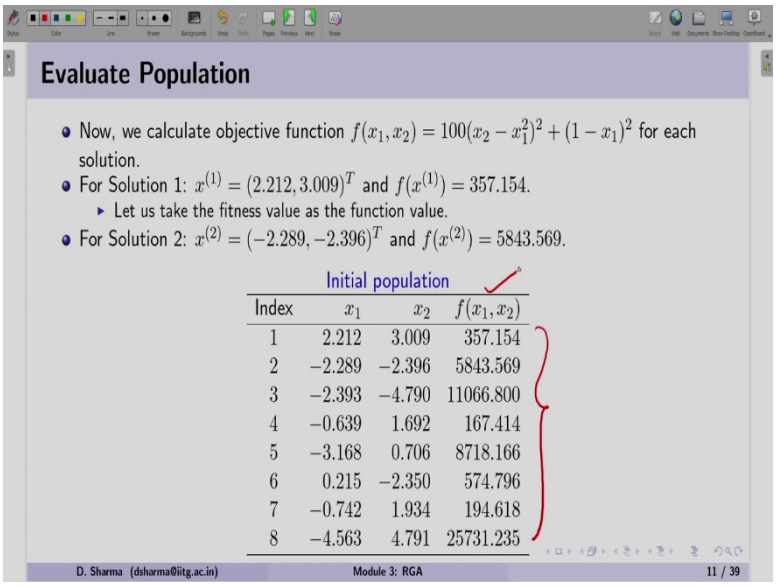
(Refer Slide Time: 15:15)



So, as per the flow chart we have to start with the initial population. So, let us assume that we have a population size of 8, then we generate the initial population. Generating initial population means that we generate a random number for x 1 between minus 5 to 5 it is a real number similarly we generate x 2 between minus 5 to plus 5 and that is also random.

So, here you can see these random numbers are generated and that is making say solution 1. Similarly for solution 2 and other solutions are generated randomly. So, as a reminder or a recap the decision variables are coded as real numbers here.

(Refer Slide Time: 16:13)



$$f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2$$

$$x^1 = (2.212, \&3.009)^T \text{ and } f(x^1) = 357.154$$

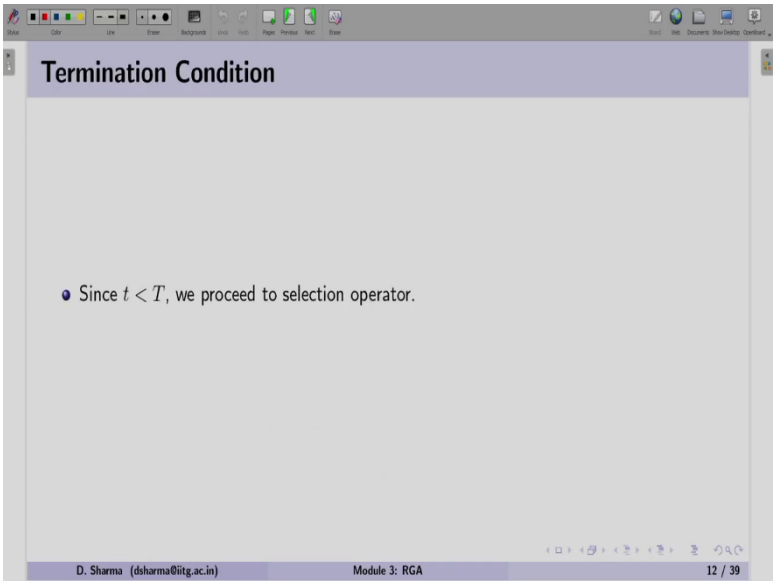
$$x^2 = (-2.289, -2.396)^T \text{ and } f(x^2) = 5843.569$$

Once the initial population is generated we have to evaluate the population; as of now the objective function we have taken as a Rosenbrock function. So, we have this two variable function. Let us take solution number 1. So, here I am representing the solution as a column vector. So, when as an when we are solving a problem specially the multi variable problem we can always represent our solution as a column vector.

So, the both the component of x 1 and x 2 are given here, putting into the value of x 1 and x 2 the objective function we can get the function value of solution 1. Now here let us assume that the fitness value is the same as the function value, this is the same assumption we took when we solve the same problem using binary coded GA. Similarly, I can take solution number 2 in this solution 2, I can directly include this value in the objective function and I can get the value.

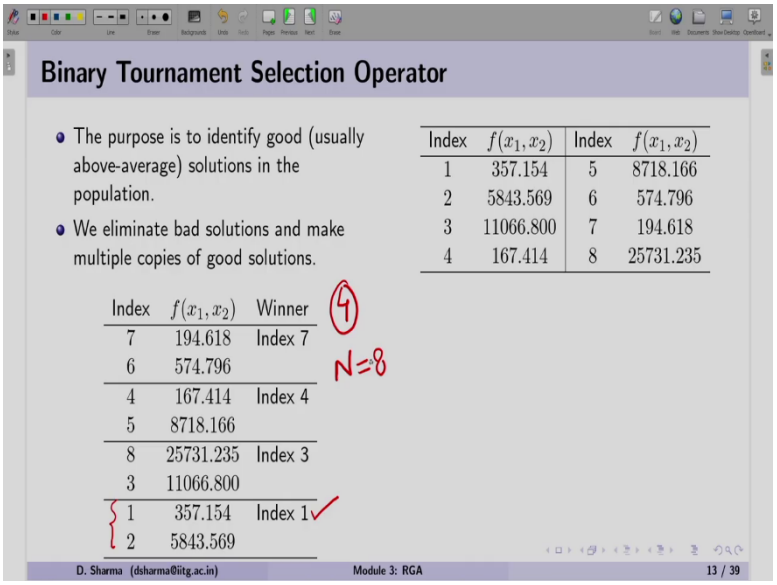
So, serially I have to calculate the objective function value which is the same as the fitness value for the solution for all the solution. Now in this case when we follow then you can see here all the solutions are evaluated their fitness values are given in the last column here; once we evaluate the population we look for the termination condition.

(Refer Slide Time: 17:51)



So, since it is the first generation let us move ahead. So, we will proceed to the selection operator.

(Refer Slide Time: 18:01)



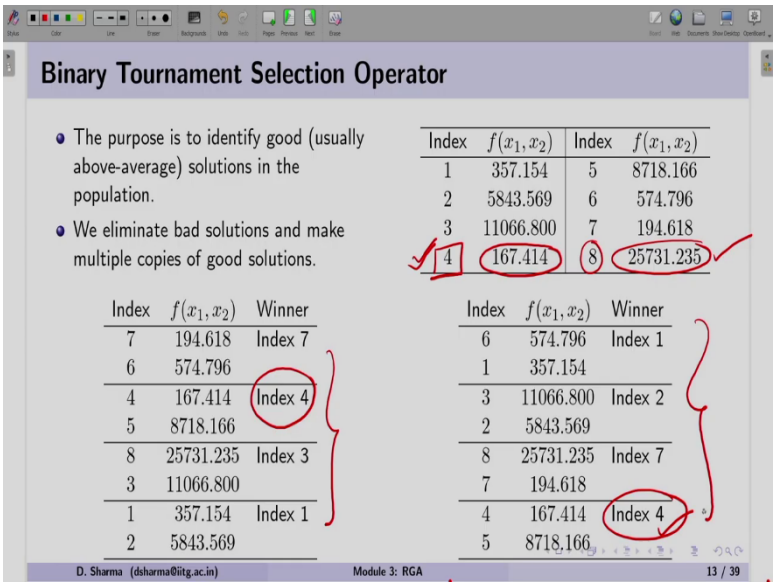
Now, coming to the selection operator as a recap we know the purpose of selection operator is to identify good above average solution in the population. It is because we can identify bad solutions those solutions will be deleted and we can identify good solutions so, that we can make the multiple copies of it. In this particular example we will again take the binary tournament selection operator which we discussed earlier.

For performing the binary tournament selection operator you can see that we only need the fitness value. So, the solutions all aid solution their fitness values are given in the table. Now as we know in the binary tournament selection operator we have to pick 2 solutions randomly, this binary tournament selection is without replacement. So, as an when we choose 2 solutions those solution will not be taken further in the tournament. So, let us take the first pair of solution.

So, looking so randomly we pick 7 and 6 solutions looking at their fitness we can make it out that the solution 7 has a less fitness. Since we are solving a minimization problem index 7 is the winner. Similarly now we are left with 6 solutions for among those 6 solution randomly we pick 4 and 5 and looking at their fitness value solution 4 is the winner. Now we are left with so 4 solutions randomly we pick solution 8 and 3, looking at their fitness value solution 3 is the winner.

Finally, we are left with solution 1 and a 2. So, we pick them and looking at their fitness value solution 1 is selected. So, when we perform the binary tournament selection for one time we have selected as of now 4 solutions and we know the population size is 8. So, what we have to do is, we have to again take all the solutions and perform the binary tournament selection.

(Refer Slide Time: 20:41)



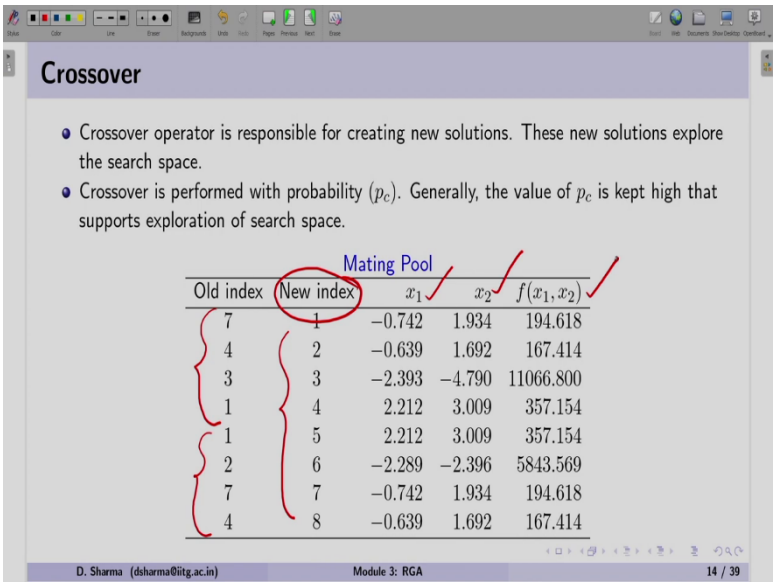
So, in this case we will take all the solution again and randomly pick 2 solutions. Here in the second tournament we pick 6 and a 1 at a random and then looking at their fitness value index 1 is the winner. Thereafter, we pick 3 and a 2 out of 6 solutions looking at their fitness index 2 is the winner. Thereafter we pick 8 and a 7 looking at their fitness solution 7 is the winner and thereafter we are left with 4 and a 5 and looking at their fitness value solution 4 is the winner.

Now, let us see the same observations which we found during the binary coded GA. Now looking at the table on the top so this is the solution; solution 4 has the best fitness. And among those 8 solution the worst fitness is corresponding to solution number 8. Now as you know if we are performing the tournament selection twice the best solution which is currently 4 should get a 2 copy; so let us see that.

So, we can identify we have a solution 4 in a tournament 1, similarly solution 4 in the tournament 2. So, we get 2 copies. Now let us look at the worst solution which is solution number 8. So, here we do not get any copy of a solution 8, similarly here also we would not get any copy of solution 8.

So, we have the same observation or the properties that the best solution in binary tournament selection operator will get 2 copies, the worst solution will get no copies and the other solutions like 7, 3, 1 and 2 they may get 2 copies, 1 copy or there will be no copy. After performing the tournament selection we use these selected solution to make a mating pool.

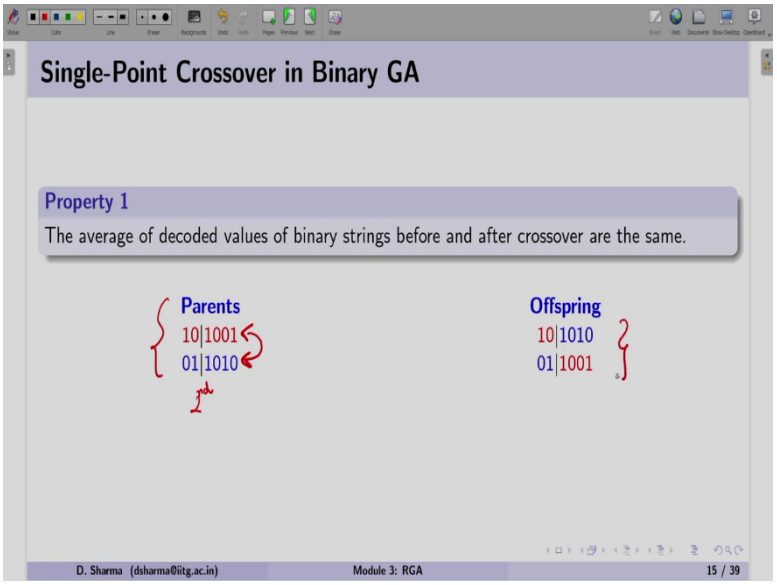
(Refer Slide Time: 22:59)



Now, once we perform the selection operator we have to perform crossover operator. Now we know the crossover is responsible for creating new solutions meaning that these new solutions will be exploring the search space so, that we can locate the minima for our given problem. Generally, crossover is performed with a probability p c and we keep this value high say point 8 point 9 or sometimes 1 as well.

So, here the mating pool is created as you can see that these 4 solutions were selected by tournament 1 the other 4 solutions were selected by tournament 2. So, we put together in a same sequence and now we are giving the new index here. So, all this new index we will be using in the further steps of real coded GA. In the mating pool we have given x 1 and x 2 and the fitness value we have given just for our references, although we understood that the fitness value is not needed in crossover as well as in mutation.

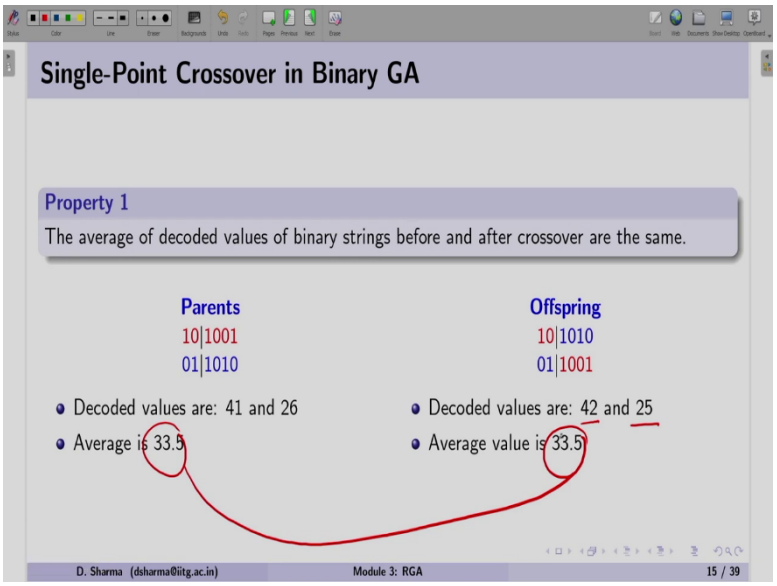
(Refer Slide Time: 24:21)



Now, the point is that we have gone through the crossover operators for binary coded GA. Now if we talk about the single point crossover operator there are certain properties. So, if those properties we can use it and we can devise a crossover operator for real numbers it can be useful for us.

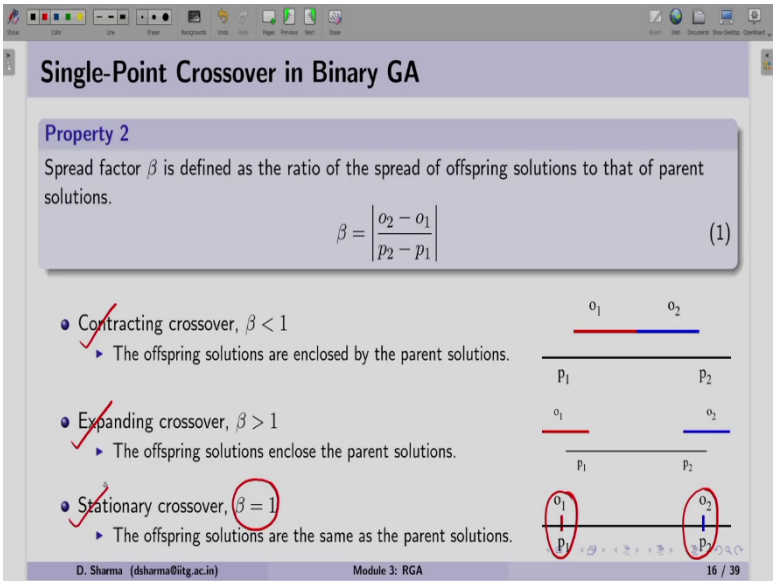
So, the property number 1 with single point crossover operator is that the average decoded value of the binary string before and after the cross over operator are the same let us see how. So, we have these 2 parents we have taken this 2nd side randomly and you know that we are going to flip the tail and this is 1 point crossover operator. Once we do it these are the offsprings which we get it. Let us decode the value.

(Refer Slide Time: 25:25)



Now, decoded value of the 1st string is 41 and the 2nd string is 26, if we get if we find the average value this is 33.5. Let us find for the offspring solutions. Now here the decoded value of the first solution is 42 another solution is 25 we find the average, now you can see the average is the same as after. So, this is the property of one point or single point crossover operator that the average value of decoded average of decoded value of binary string before and after are the same.

(Refer Slide Time: 26:15)



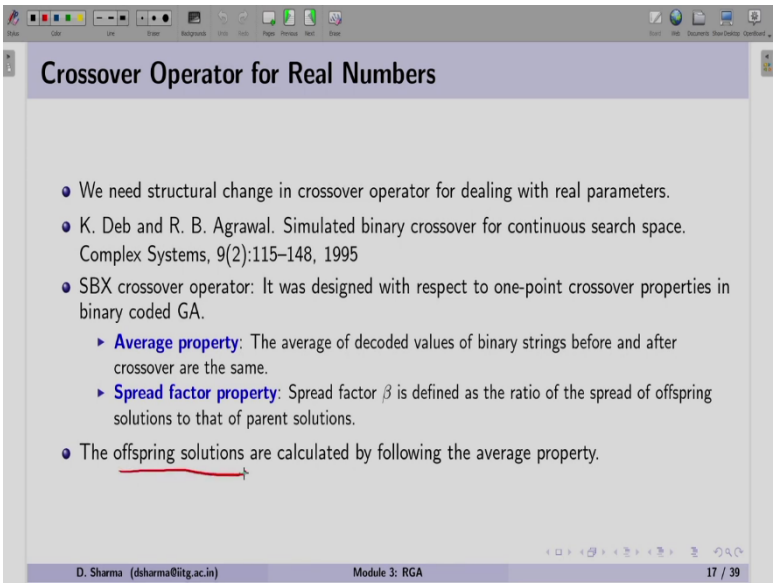
$$\beta = \left| \frac{o_2 - o_1}{p_2 - p_1} \right|$$

Let us move to the property number 2 of single point crossover. So, the property number 2 says that there is a spread factor beta here which is defined as the ratio of spread of offspring to that of parents. So, if we look at this formula equation number 1. So, the difference between o 1 and o 2 divided by p 2 and a p 1.

So, this spread factor will tell us about the property of crossover operator. So, let us take the first case, suppose in this case beta is smaller than 1; in this case as you can see in the figure here. So, these offspring o 1 and o 2 are enclosed by the parent solutions p 1 and a p 2 that is also evident from the equation 1.

Suppose if the beta is greater than 1, then the offspring solutions are enclosing their parent solution p 1 and p 2, the third case that is possible is call stationary crossover when beta is going to be 1. So, here you can see offspring solutions are the same as the parent solutions. So, we can have a 3 situations as you can see contracting crossover, expanding crossover or it can be a stationary crossover.

(Refer Slide Time: 27:45)



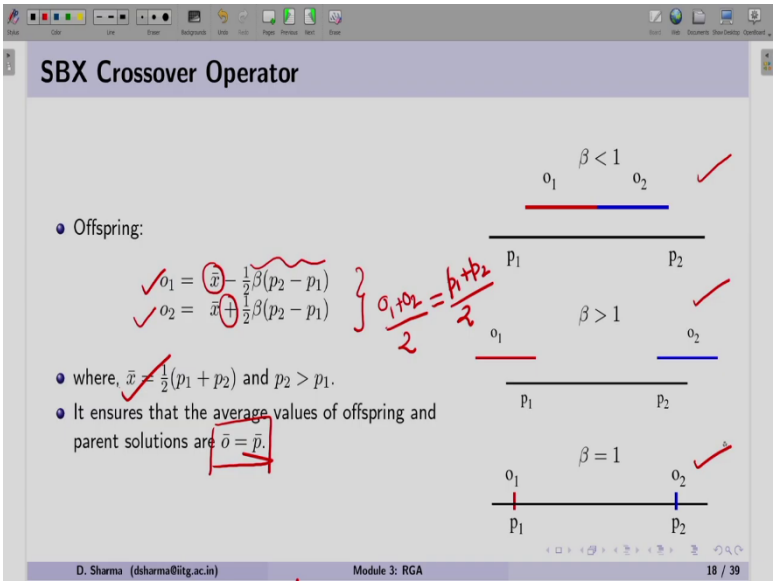
Now, as we have to move towards the crossover for real numbers. So, we have to change those crossover operator. So, in that case we have to change their structure. So, how they are

going to perform the crossover between the 2 solutions? Now here we have taken one of the operator which has been used with many algorithm and has shown success in solving different varieties of a problem in the literature.

So, this operator is called at simulated binary crossover operator, in short we call this as a SBX operator. Now, as you can see from the name. So, binary means it will be stimulating the one point crossover operator and that simulated crossover operator we use for continuous search space. Now as it is mentioned here, it is designed with respect to one point crossover properties in binary GA. So, what are those properties? Let us have a recap here.

Average, so, first is called average property which says that the average of decoded values of binary strings before and after crossover are the same. And there is a spread factor that says the ratio of the spread of offspring solution to that of parent solution. So, we use this beta factor here. Now using these two properties let us target the first property as a average the average property and find what could be the offspring solutions.

(Refer Slide Time: 29:27)

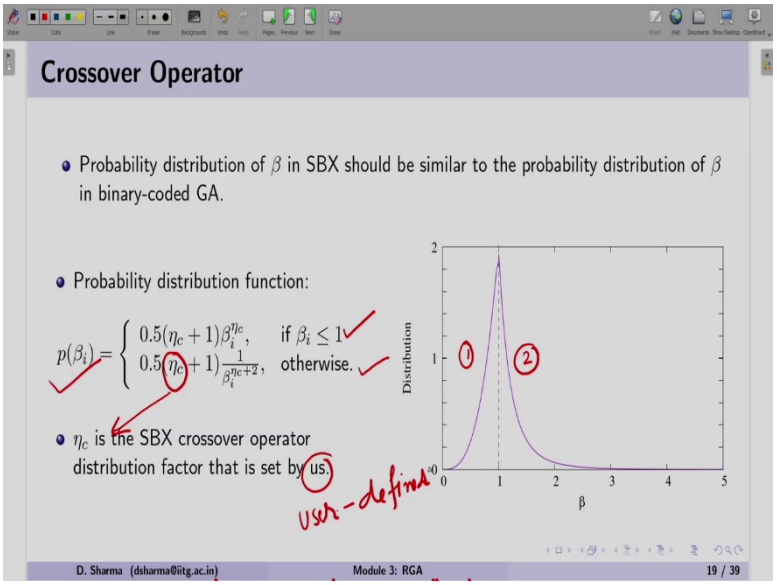


$$o_1 = \bar{x} - \frac{1}{2}\beta(p_2 - p_1)$$
$$o_2 = \bar{x} + \frac{1}{2}\beta(p_2 - p_1)$$
$$\bar{x} = \frac{1}{2}(p_1 + p_2) \; and \; p_2 > p_1$$

So, these authors come up with the idea of using a different offspring using the property of averaging. Now here in this formula you can see that the offspring 1 is equals to x bar. So, this x bar is represented by the average of 2 parents. So, p 1 plus p 2 divided by 2 then minus half times of beta p 2 minus p 1 similarly for o 2. So, only the sign plus has been change. What we can see here?

That if I take the average of o 1 plus o 2 divided by 2 this is the same as p 1 plus p 2 divided by 2 and that is why I have written here. So, by using this particular formula we can actually generate the offspring that is following the property of single point crossover operator. So, all the 3 cases for beta smaller than 1, beta greater than 1 or beta equals to 1 will be taken care.

(Refer Slide Time: 30:47)

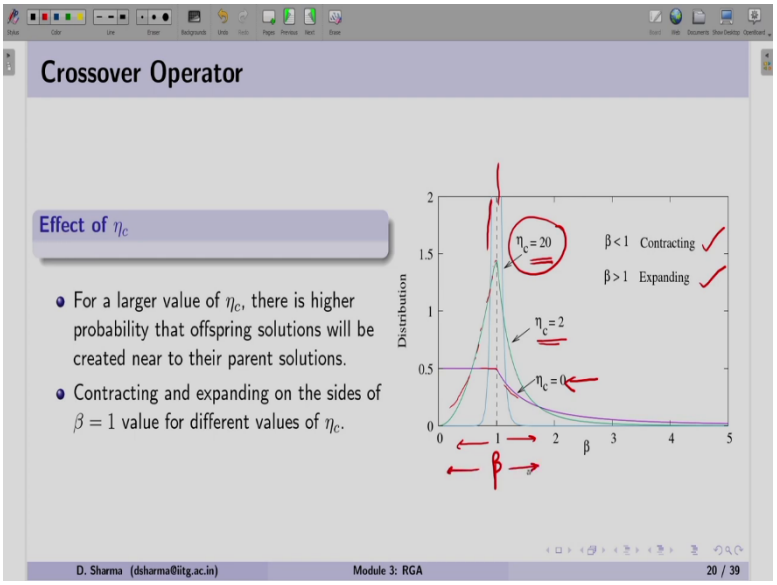


$$p(\beta_i) = \{0.5(\eta_c + 1)\beta_i^{\eta_c}, \ if\ \beta_i \le 1\ 0.5(\eta_c + 1)\frac{1}{\beta_i^{\eta_c+2}}, \ otherwise$$

Now, in this SBX crossover operator, so, the probably distribution of a beta in this operator we will should be similar to the properties of distribution of a beta in binary coded GA. So, we will be using this property as a for creating the SBX operator. So, these authors they come up with the probability distribution as you can see here which works when for the beta i smaller than 1 then this is the factor and otherwise we are going to use the another 1.

As you can see in the figure you have 2 distribution. So, this is the 1st part, this is the 2nd part. So, for these 2 distributions you can find how the beta probability distribution of a beta is distributed. Now in this formula you can find there is a one factor called eta c and this eta c we refer to as a SBX crossover operator distributor distribution factor and this should be set by us. It is it means that it is a user; it is a user defined parameter; once using the properties of single point crossover and a mutation operator.

(Refer Slide Time: 32:11)



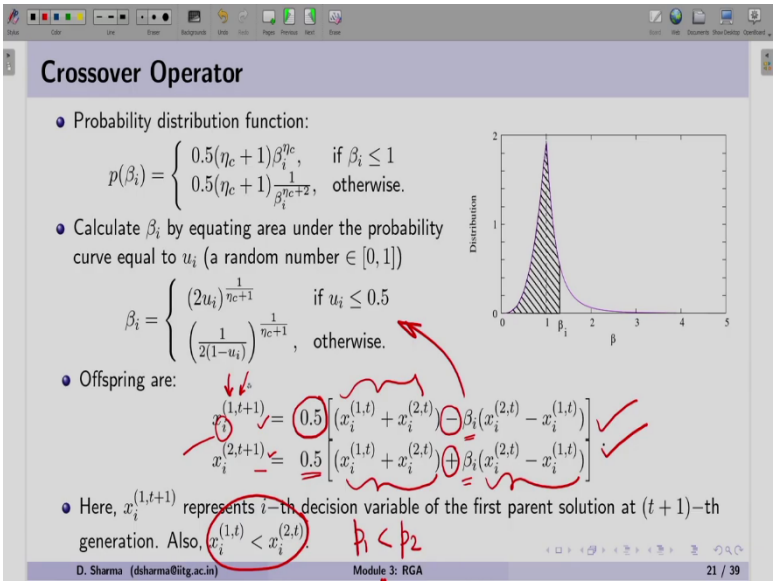
So, let us see how this SBX crossover operator effect based on eta c, it is only because this is the eta c value which is the user defined value. Now, let us see the figure on the right and side. So, the first plot is shown for eta c equals to 0, as you can see that before 1 they this is a constant the probability function is a line and then thereafter you can see the distribution.

When you increase the eta c value? The curve has actually increased and then you can see the distribution of probability function on the both side of beta equals 1 beta is smaller than 1 and beta greater than 1. Suppose if you take a very large value you can see a very steep curve and this curve is going on the top.

Now this probability function is made in such a way that the area under the probability distribution say from minus infinite to plus infinite should be equal to 1. So, basically area under the curve should be 1. In this particular case when we are taking eta c as a large value, that is why we get a very steep curve here for the beta distribution. Both the cases of

contracting and expanding we can see with respect to the beta value on the left and side and the right hand side.

(Refer Slide Time: 33:49)



$$p(\beta_i) = \{0.5(\eta_c + 1)\beta_i^{\eta_c} \qquad if\ \beta_i \le 1\ 0.5(\eta_c + 1)\frac{1}{\beta_i^{\{\eta_c+2\}}},\ otherwise$$

$$\beta_i = \{(2u_i)^{\frac{1}{\eta_c+1}}, \qquad if\ u_i \le 0.5\left(\frac{1}{2(1-u_i)}\right)^{\frac{1}{\eta_c+1}},\ otherwise.$$

$$x_i^{(1,t+1)} = 0.5[(x_i^{(1,t)} + x_i^{(2,t)}) - \beta_i(x_i^{(2,t)} - x_i^{(1,t)})]$$
$$x_i^{(2,t+1)} = 0.5[(x_i^{(1,t)} + x_i^{(2,t)}) + \beta_i(x_i^{(2,t)} - x_i^{(1,t)})]$$

So, as this probability distribution we have seen earlier. Now the question is, how we can use this probability distribution, which is non-linear right now to find out the 2 variables or 2 off springs? In this case suppose as you can see in the plot we will be finding the area under the curve. So, this particular edged line can be seen here.

So, in this way we can calculate the value for beta i we can calculate by equating the area under the probability curve equals to u i and u i is the random number. So, if we are going to integrate this beta distribution, then this integration or the area under the curve in this probability distribution will help us to calculate the factor called beta i.

Now, looking at these two formulas so, the upper part is valid when the random number is equal to or smaller than 0.5. So, in this one we need a random number and eta c, now eta c value will be decided by us. So, it is a user defined parameter and u i is the random number will be generated by a computer for us.

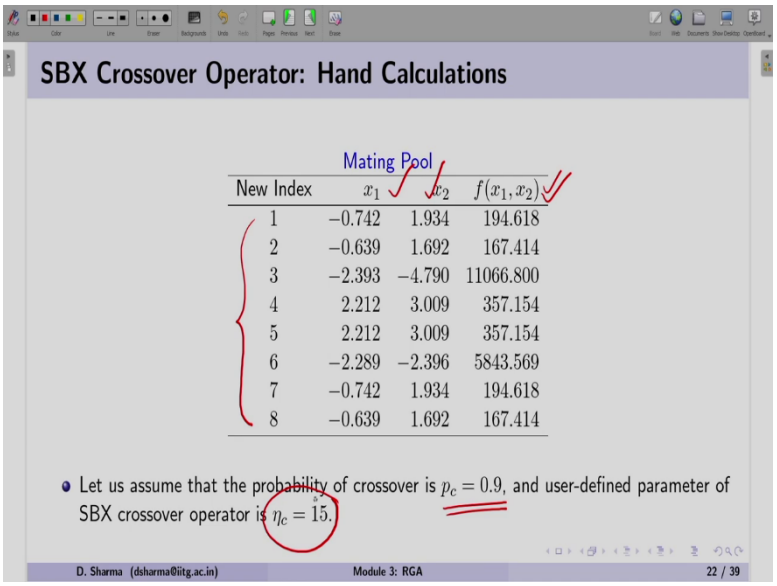
For u i greater than 0.5 value we have this particular formula. So, these two formulas will be using for calculating the beta i values. Now if we are going to follow the average property of 1 point crossover operator. So, the 2 offspring has you can see on the left hand side.

So, the 1 by 2 part which is 0.5 we have taken outside, this is the summation of 2 solutions p 1 and a p 2 which are represented as x i 1, t plus x i 2, t minus. So, first is minus times beta. So, beta we have calculated in the earlier step and then there is a difference which is p 2 minus p 1. Similarly the offspring 2 we have a factor half outside the bracket than we have the summation which is nothing but the average of the 2 parents. Now the sign is plus beta times of the difference between the 2 solutions.

Now, it is important to note that here that we have used this property of averaging it, but we have to make sure that when we are selecting the 2 solutions that x. So, the x i 2, t should be greater than xi 1, t meaning that p 1 should be smaller than p 2. In case it is other way around than we interchange the value of p1 and p 2. So, by following this condition we can use the formula given here for generating 2 offspring as x i 1, t and x i 2, t plus 1.

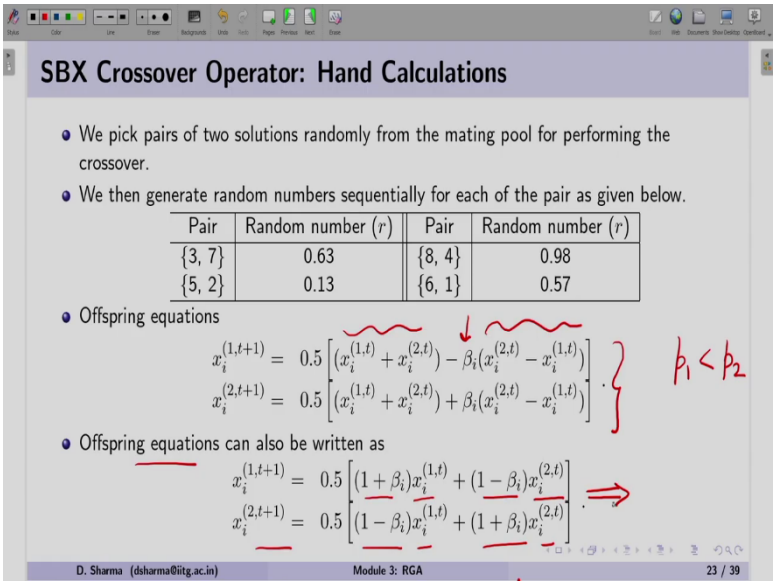
Now, the representation as I have mentioned here. So, the i is going to tell us that this is the; this is the ith variable, 1 will be telling me that this is the first solution and t you know that we use it for the generation counter. So, these 2 formulas we are going to use it to create offspring solutions.

(Refer Slide Time: 37:29)



Now, coming to the SBX crossover operator, now we will perform some hand calculations. So, the new index that we have used in the mating pool that we will take it into the crossover as well as in the mutation; for mating pool x 1 and x 2 are given fitness value we have given for our reference we do not need it as of now. Now, let us assume that the probability of a crossover operator is 0.9 and the 0 and the user defined parameter eta c we take as eta c equals to 15.

(Refer Slide Time: 38:07)



$$x_i^{(1,t+1)} = 0.5[\left(x_i^{(1,t)} + x_i^{(2,t)}\right) - \beta_i(x_i^{(2,t)} - x_i^{(1,t)})]$$
$$x_i^{(2,t+1)} = 0.5[(x_i^{(1,t)} + x_i^{(2,t)}) + \beta_i(x_i^{(2,t)} - x_i^{(1,t)})]$$

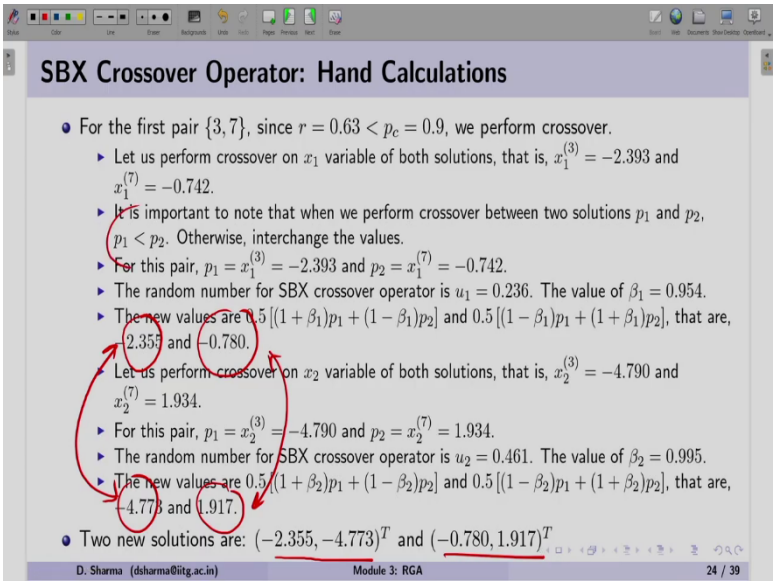
$$x_i^{(1,t+1)} = 0.5[\left(1 + \beta_i\right)x_i^{(1,t)} + \left(1 - \beta_i\right)x_i^{(2,t)}]$$
$$x_i^{(2,t+1)} = 0.5[\left(1 - \beta_i\right)x_i^{(1,t)} + \left(1 + \beta_i\right)x_i^{(2,t)}]$$

Now, when we perform SBX operator similar to 1 point crossover operator we have to pick 2 solutions randomly. So, these 2 solutions are picked from the mating pool. Now assume that we have picked say 3 and a 7 solutions together and for performing the crossover operator we generated the random number say 0.63.

This will help us to decide whether we have to perform the crossover operator or a not, similarly you can find the pair as well as their random numbers. Now here before we proceed to the hand calculation this is the formula which we have just seen in the previous slide.

Where we have the average of the 2 solutions and then we have a beta factor and we have a difference of these 2 solutions by knowing that p 1 is smaller than p 2. Now if I am going to rearrange the terms so the same offspring equations can be written as 1 plus beta p 1, 1 minus beta p 2. Similarly for the another offspring it is 1 minus beta p 1, 1 plus beta p 2. So, the same equation we can write it in our calculation this is the format we are going to use it.

(Refer Slide Time: 39:39)



So, let us pick the first 2 pair and that is 3 and a 7 and the random number that generated was 0.63. Since it is smaller than probability of a crossover we perform the crossover operator here. Now here it is important to note that we will be performing crossover operator variable wise. So, we will pick variable 1 first to perform the crossover operator. In this case the x 1 value of solution 3 is given here similarly x 1 value of solution 7 is given here.

So, here the terms which I am writing on the top it represent it is a solution 3. Similarly, this says that this is solution 7. Here, just as a reminder we have to make sure that we follow this relation that p 1 is smaller than p 2. Since p 1 is already smaller than p 2. So, we are again considering p 1 as x 1 3 and p 2 as x 1 7.

Now, in order to perform the crossover operator you know that we have to first generate a random number suppose the number given is 0.236 and if we put this value since it is smaller than 0.5 we will be using one of the formula here and we can get the value of a beta i as 0.954.

Now the 2 new values of offspring, so basically for variable 1 so, we will be using the formula here. So, instead of a using xi 1, t x i 2, t we are writing the formula in terms of p 1 and a p 2. So, these 2 formulas we will use it and we will get the x 1 value for a solution 3 and 7 as minus 2.355 and minus 0.780.

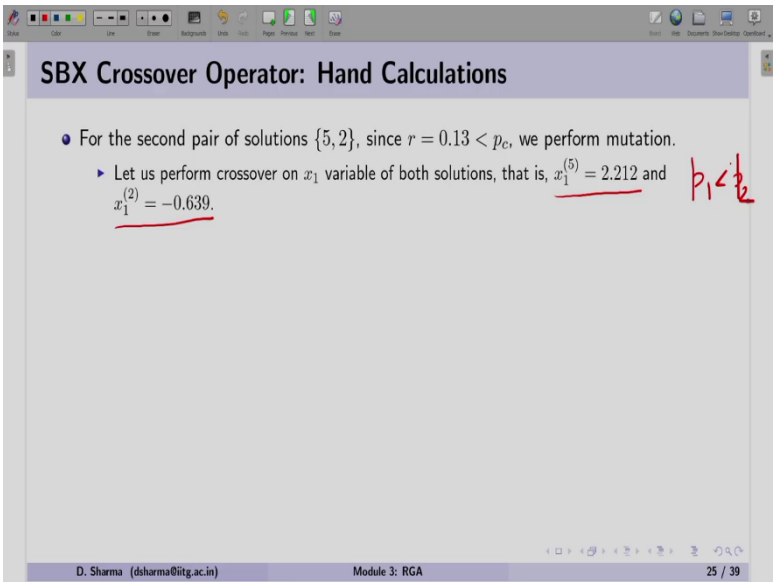
Since we are performing variable boys now we have to take the variable x 2 into our consideration; here in this case the variable x 2 for solution 3 is given and similarly variable 2

value for solution 7 is also given. Looking at the relation between p 1 and a p 2 we will assign p 1 as x 2 3 and p 2 as x 2 7.

In this case when we perform we have to again generate a random number. So, some number is given here say 0.461 and by putting this value into the formula beta 2 came out to be 0.995. Again the same set of equations for generating the offspring 1 and offspring 2 will get the value as minus 4.773 and 1.917. Now once we have generated different values of x 1 variable and x 2 variable we have to put together.

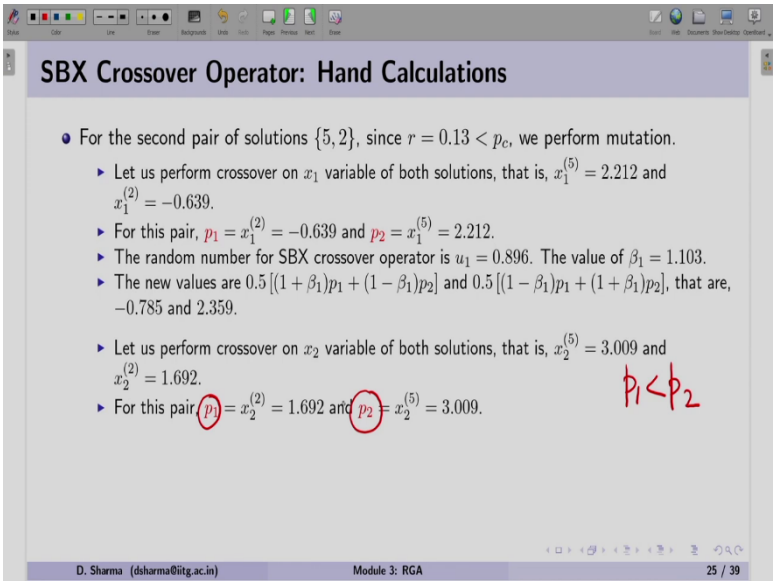
So, you can see that we have taken the first component together and that is make our first solution. Similarly, we took the second combination together and then we make the solution number 2, it is because there is no interchange between p 1 and a p 2 by following the relation p 1 should be smaller than p 2.

(Refer Slide Time: 43:23)



Now, the next pair we select is say 5, 2 just to repeat one more time. So, that we can understand how this crossover operator is working we will show the hand calculation for this set of pair. Now we have picked 5 and a 2 since the random number is smaller than probability of a crossover we perform crossover. In this case we have x 1 5 as 2.212 and x 1 2 as minus 0.639 since we have to follow the property that p 1 should be smaller than p 2.

(Refer Slide Time: 44:09)

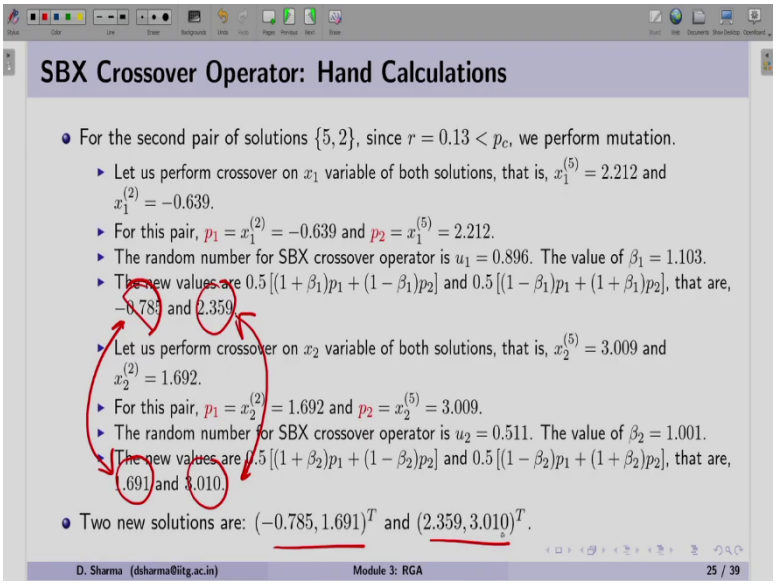


So, what we are going to do is we will interchange. So, now, you can see I have made it in a red colour p 1 and a p 2 because we have interchange the value although we picked 5 and a 2 in the sequence, but currently p 1 is x 1 2 and p 2 is x 1 5. Now for performing the SBX crossover operator we generate a random number and using this particular random number, now it is greater than 5.

So, we will be using the another formula to get a beta i value as 1.103. So, this beta i value we are going to use in the same set of formulas that will give me the first variable for 2 solution as minus 0.785 and 2.358, you have seen that we have performed the crossover operator for x 1 variable, now it is a turn for x 2 variable.

So, we take x 2 variable we find the second variable value for solution 5 and for solution 2. Again we have to follow the property of p 1 smaller than p 2. So, we have interchanged the value again and I have represented in a red colour of p 1 and a p 2.

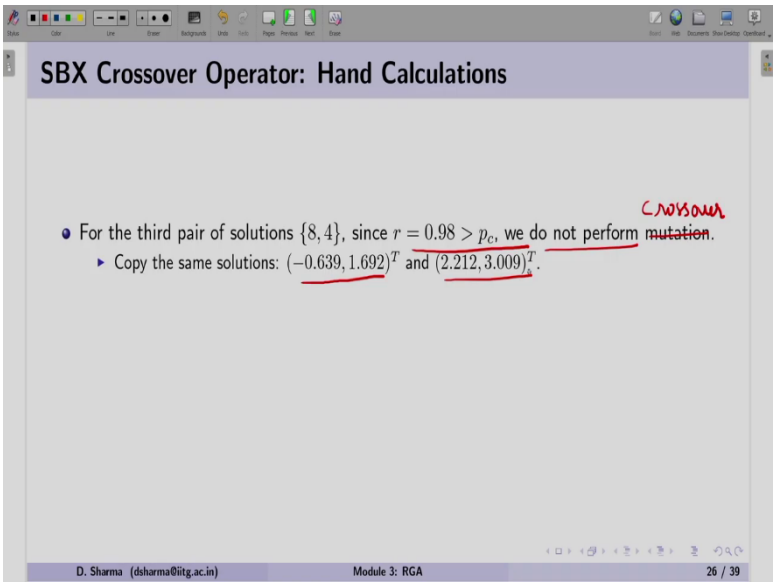
(Refer Slide Time: 45:35)



Once it is done we are going to generate a random number u 2 and since it is greater than 0.5 we will be using the second formula for a beta value and beta 2 will become 1.001, putting into the our standard formula of offspring 1 and offspring 2, this will create 2 values as you can see 1.691 and 3.010.

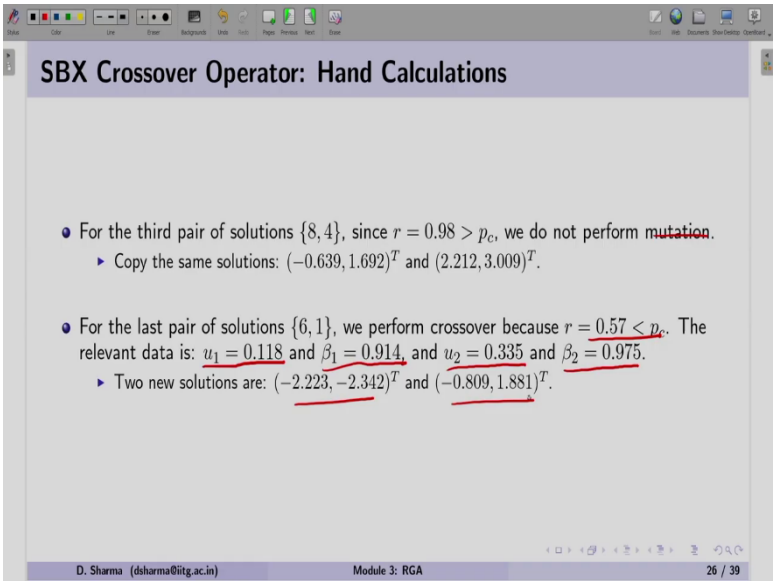
Similar to the previous case since in both the x 1 and x 2 variable both are interchange. So, we will take the first value here and the second value here and we will put together we will get a solution 1. Similarly, we take solution number 2 we put together we will get a solution number 2 here.

(Refer Slide Time: 46:27)



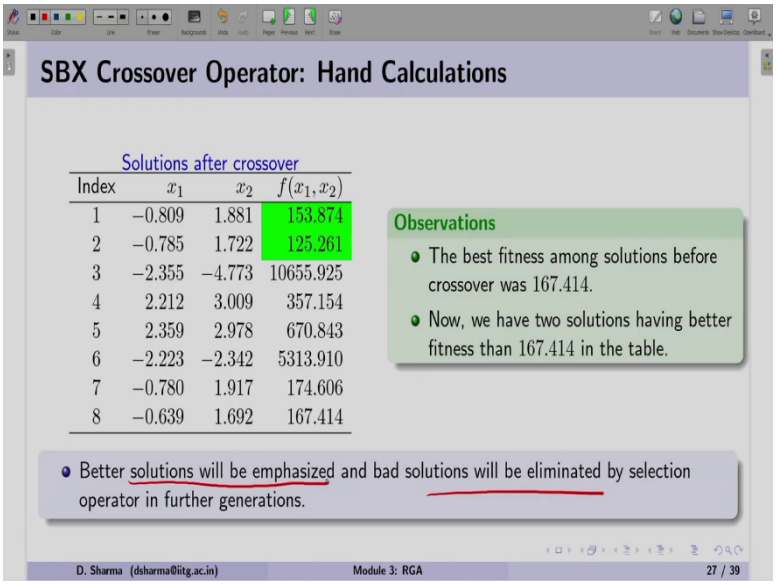
Now, let us look at the third pair now here we can observe that the random number is greater than p c. So, we will not perform any crossover. So, this is wrong this should be cross over. Now in this case when we are not performing the crossover operator we are going to copy the solution as it is. So, the column vectors as I told you earlier are copied as it is.

(Refer Slide Time: 47:01)



Now, we will pick the last pair which is 6 and a 1. Now at this stage we know how to perform the crossover operator. So, the hand calculation I have shown here. So, in this case the random number is smaller than p c we perform the crossover operator and the relevant data say for example, u 1 is created that with the help of that we can calculate beta 1 value then we generated u 2. That give me the value of a beta 2 putting into the formula we are going to get 2 solutions as given here.

(Refer Slide Time: 47:43)

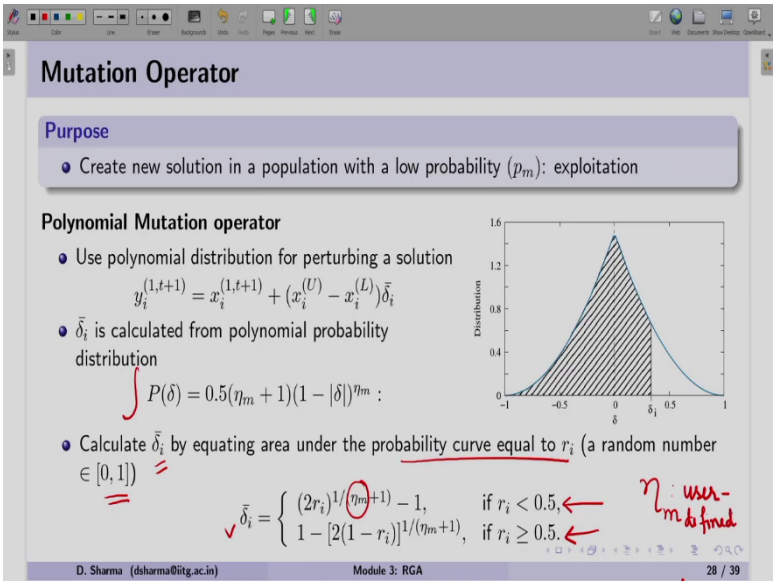


Now, putting all those solution in a table according to their index values x 1 and x 2 values are shown. Now the fitness values are shown for our purpose although we do not need it. Now, since we are showing this fitness value if there are certain observation. Now, the best fitness as you can see here the best fitness in the initial population was 167.414.

When we perform crossover operator we get these 2 solutions which are better than the best solution in the initial population. Similarly some other solutions are also generated. So, the observation here is the crossover operator can generate good as well as bad solutions.

When bad solutions are generated they will be eliminated by the selection operator in further generation, if better solutions are generated they will be emphasized and we can make multiple copies of those solutions. After performing the cross over operator let us move to mutation operator.

(Refer Slide Time: 49:09)



$$y_i^{(1,t+1)} = x_i^{(1,t+1)} + \left(x_i^{(U)} - x_i^{(L)}\right) \bar{\delta}_i$$

$$P(\delta) = 0.5\left(\eta_m + 1\right)(1 - |\delta|)^{\eta_m}:$$

$$\bar{\delta}_i = \{\left(2r_i\right)^{\frac{1}{(\eta_m+1)}} - 1, \quad if\ r_i < 0.5,\ 1 - \left[2\left(1 - r_i\right)\right]^{\frac{1}{(\eta_m+1)}}, \quad if\ r_i \geq 0.5$$

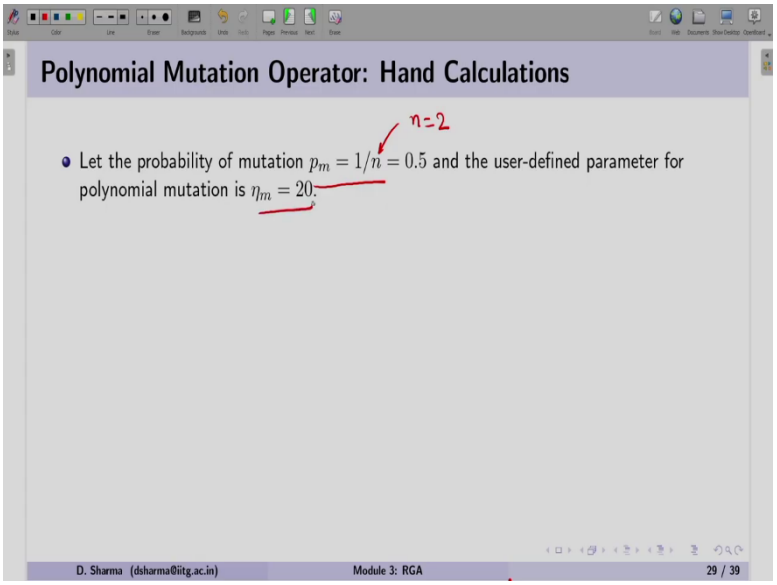
We know that we perform this mutation operator with a very low probability called p m, generally the mutation operator is used for exploitation. In this case we will be discussing one of the mutation operator called polynomial mutation operator.

As you can see in the formula that the mutated solution is made using the solution generated by the crossover operator the difference between the lower and upper bound and everything gets multiplied by delta i value. So, what is delta i? Again we are using some non-linear probability distribution as you can see here.

In this probability distribution we again find this delta i value by equating the area under this curve. So, the delta i value by equating the area under the probability curve to say random number and this random number will be lying between 0 to 1. So, if we are going to integrate it and then find the area we can get the delta i value based on the random number.

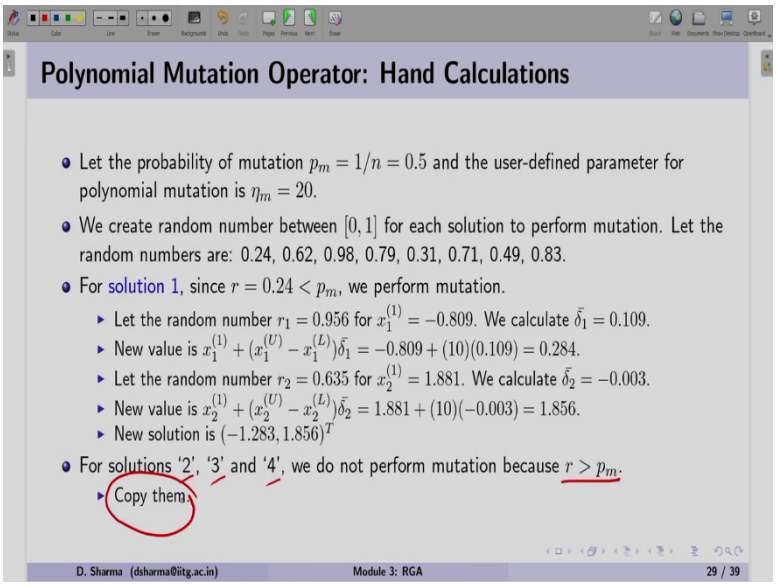
So, if random number is smaller than 0.5 we will use the first equation if it is more than 0.5 we will use the second equation. In mutation you can see that again there is a user defined parameter called eta m, this is again the distribution factor similar to SBX operator and this value has to be defined by the user. So, it is a user defined parameter.

(Refer Slide Time: 50:59)



Now let us perform some hand calculation using the polynomial mutation, here the probability of mutation is 1 by n. This n we have taken equals to 2 this is because we are currently solving a 2 variable problem; the user defined parameter eta c we have taken 20.

(Refer Slide Time: 51:21)



Now, to perform the polynomial mutation we generate a series of random number, that decide whether we will be performing the mutation or not. So, these are the set of the random numbers that a computer can generate between 0 and 1. Let us pick the solution number 1 and since the first random number is smaller than the probability of mutation so we perform this mutation here.

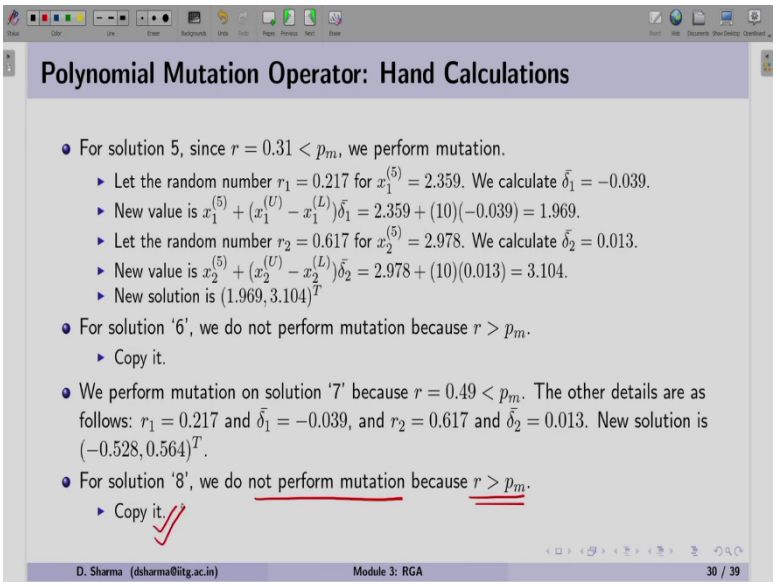
Again you can observe here we are going to perform this mutation operator variable wise. So, let us pick the variable 1 of solution 1 which is currently given here and we generated a random number 0.956. In this case if we calculate the delta i value it is coming out to be 0.109.

Now once we calculated we are going to put into our formula here and that will give me the value as 0.284. Now we will perform the mutation for the x 2 variable here the value is given here, let us assume that the random number is 0.635. We calculate delta 2 value which is coming out to be minus 0.003.

Putting into our same formula here we can get the new value as 1.856. So, these 2 values if we put together so these 2 new values for x 1 variable and x 2 variables, when we put together we get a new solution that is generated by mutation operator.

Similarly, if we look at the random numbers for solution number 2, 3 and a 4, we do not have to perform mutation it is because the random number is more than probability of mutation. So, what we are going to do that? We will copy this solution as it is. So, the column vector of this solution we copy.

(Refer Slide Time: 53:43)



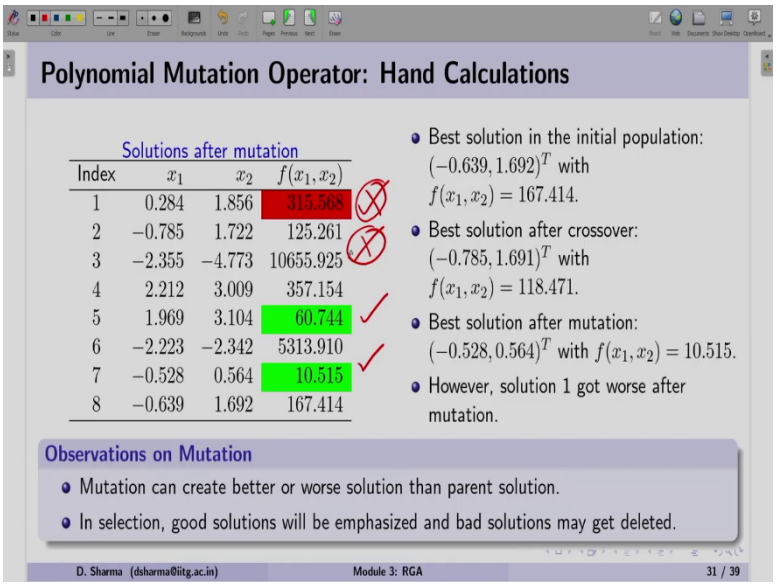
Now, come to the solution number 5 again the random number is a smaller than probability of mutation. So, we perform it in this case let us perform for the variable x 1 first take the value generated the random number as 0.217 putting into the formula give me the value delta 1 as minus 0.039.

Then we can include this into our formula and we can get the new mutated value for variable 1 is 1.969, thereafter we pick the second variable of solution 5 generated the random number as 0.617 calculate the delta 2 value as 0.013 and thereafter we can include into our formula here and the value the mutated value for variable 2 of solution 5 is 3.104.

So, we will again take these 2 values together and that will generate a new solution which is represented as a column vector here. Solution 6 we will not perform any mutation because the random number is more than probability of mutation. So, we will copy the solution as it is. For solution 7, since now we know how to perform the mutation operator. So, the data is given directly here. So, the random number is 0.5.

So, we perform the mutation, the other details as suppose the first random number is this and using this we can calculate delta 1 values thereafter we can generate the random number r 2 and the corresponding delta 2 value is given here. By using these two delta 1 and delta 2 values we get a new solution as given here. For solution 8 again we do not perform mutation it is only because, the random number is greater than mutation greater than the probability of mutation. So, we will copy this solution as it is.

(Refer Slide Time: 56:03)



Now, since we have performed the mutation operator let us put together all of them in a table. So, following the same index that we have used from the mating pool to the crossover now in mutation; the change in the value of x 1 and x 2 values are given for each solution. For our reference we are also showing the fitness value of these newly created solutions. Now let us have some observation, now the best fitness in the initial population corresponding to this solution is 167.414.

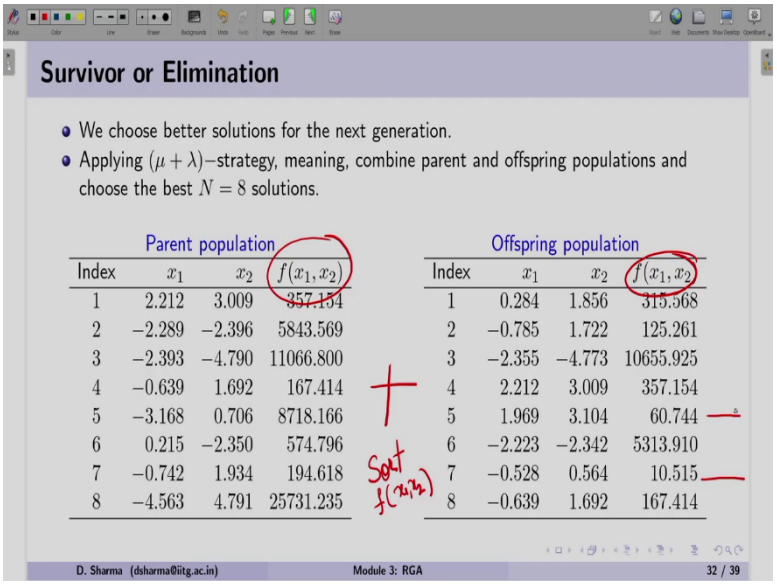
When we performed the crossover operator on the mating pool, we came across one particular solution which has a fitness of 118, which is better than the best fitness of initial population. On the crossover solutions we perform mutation operator and then we can realize that the best solution is corresponding to this solution which is solution number 7 and the fitness is 10.515; even what you can observe here that we have 2 solutions which are better than the solutions generated by a crossover operator.

At the same time you we can realize that the solution 1 after mutation this solution is worse than its original solution that was generated by crossover operator. So, that is why we made it in a red colour to make our observation as that as I have mentioned solution 1 got worse than worse after mutation.

So, the observation here is that mutation operator can create good as well as bad solutions in this case if the good solutions are created the (Refer Time: 58:13) solutions will get multiple

copies using selection operator in further generations. Those are bad solution as you can see here those solutions will be deleted in further generation because of the selection operator.

(Refer Slide Time: 58:33)

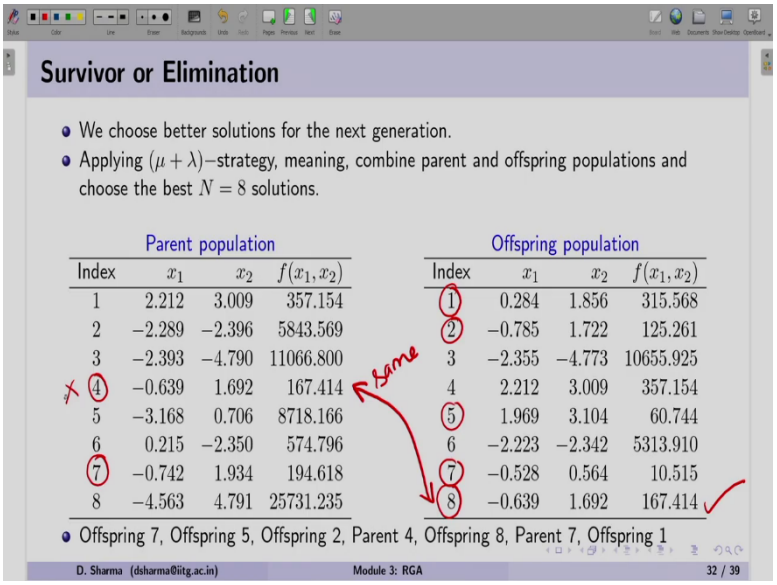


Now, at this stage what you can see that we have parent population of size 8 as well as offspring population of size 8. So, following the flow chart of real coded GA we have to perform the survival stage the purpose is we have to choose best solutions that can be taken to the further generation. Now in this case we will be using mu plus lambda strategy and in this strategy what we do here we combine parent and offspring population together and we choose the best end solution that is 8 right now.

So, these are the parent and offspring solutions given to us. Now we know at the survival stage we are going to combine them, when we combine them we have to look or we have to sort this solution based on the function value say x 1 and x 2. So, in this case we do not need x 1 x 2 value what we need only is the fitness values as given in the last column of the both table.

Now let us sort them if we sort them in an ascending order the best solution will be on the top because we are minimizing the function. In this case you can see the solution 1 is the minimum. So, that it should be on the top followed by solution 5 in the offspring. We have to follow this process till we select 8 solution.

(Refer Slide Time: 60:15)

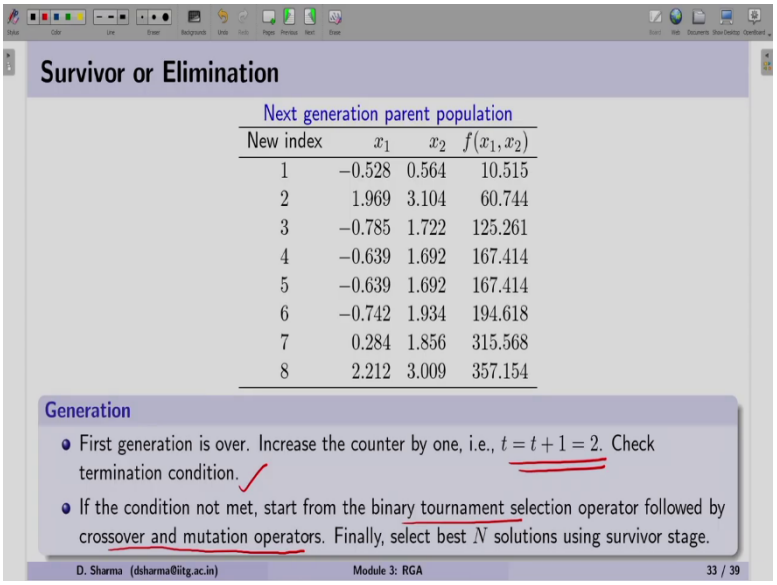


So, what we can see that solution 7 will be selected, thereafter 8 will be selected, then we will select offspring solution 2, then we will select parent solution 4, then offspring solution 8, parent solution 7 and finally, offspring solution 1. So, this solution in this particular sequence are already arranged in the ascending order of their fitness value.

So, these solutions will be selected and moved to the next generation. Now here one interesting point here you can see that the solution the offspring solution number 8 and the parent solution number 4 they are actually the same. So, in this case there are 2 strategies we can do. So, the first strategy which we are following in this that we will keep as it is we are not going to change it, another strategy is that we can copy the distinct solutions.

So, as soon as we have selected say 8 we may not select this particular solution number 4. So, that we will get different kinds of solution in the population; so, these two strategies we can use it for a present example we are following that we are copying exactly the same solution without removing the copies of the same solution in the population. So, this is the next generation population.

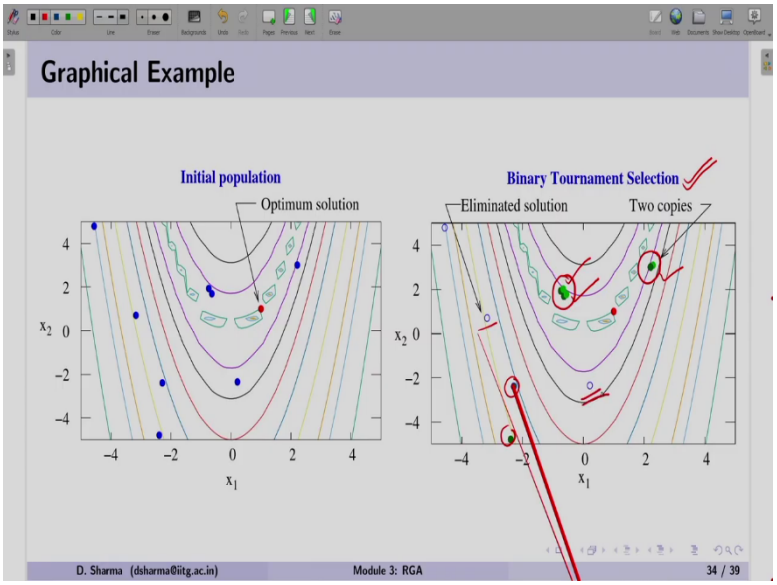
(Refer Slide Time: 61:57)



And here we have given the new index and all these solution as you can see they are already sorted in the based on their fitness values. Once the survival stage is over we have to increase a counter of number of a generation by 1. So, in this case the t will become 2 and we check the termination condition, if it is not met then we have to perform the same set of operator, such as binary tournament selection operator followed by crossover and a mutation operators as we have discussed and then survival stage.

We keep on following this operator in a same sequence till the termination condition is not satisfied. Once it is satisfied we have to terminate the algorithm and report the optimum solution for the given problem. Now let us come to the graphical example. So, the hand calculations which we did in the previous for solving the Rosenbrock function we will be see we can see how these solutions are moving towards the optima for one generation.

(Refer Slide Time: 63:15)

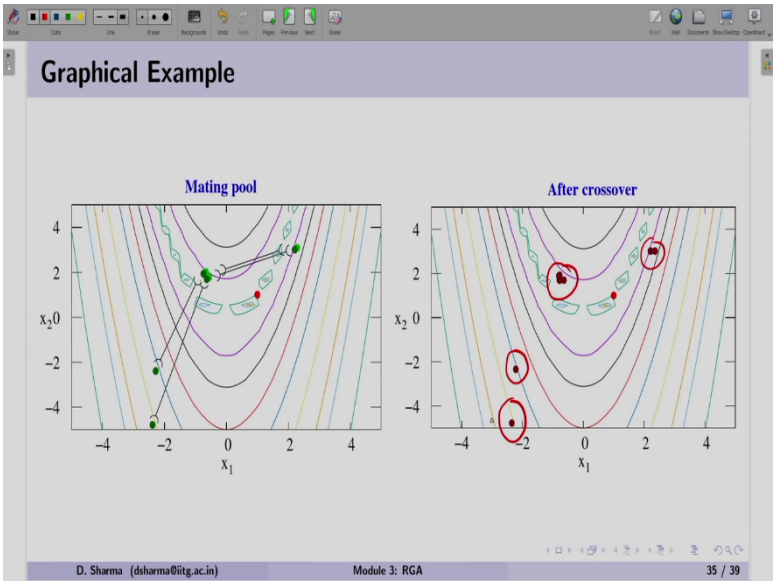


So, the as you can see in the figure we started with the initial population these blue dots represent the solutions or a randomly generated solution in the initial population. Once the initial population is done we after calculating the fitness we perform binary tournament selection. So, we are following exactly these steps which are given in the flow chart of RGA.

Now here you can see that there are 2 solutions are represented in different green colours meaning that, this solution get 2 copy, this solution get 2 copy, this also get 2 copy, this also get 2 copy and this solution will get 1 and 1 copy.

The solutions which are not filled with the colour all these solutions they are eliminated from the population and that is the purpose of binary tournament selection operator. Once we perform the binary tournament selection we get a mating pool because this mating pool is required to perform crossover operator.

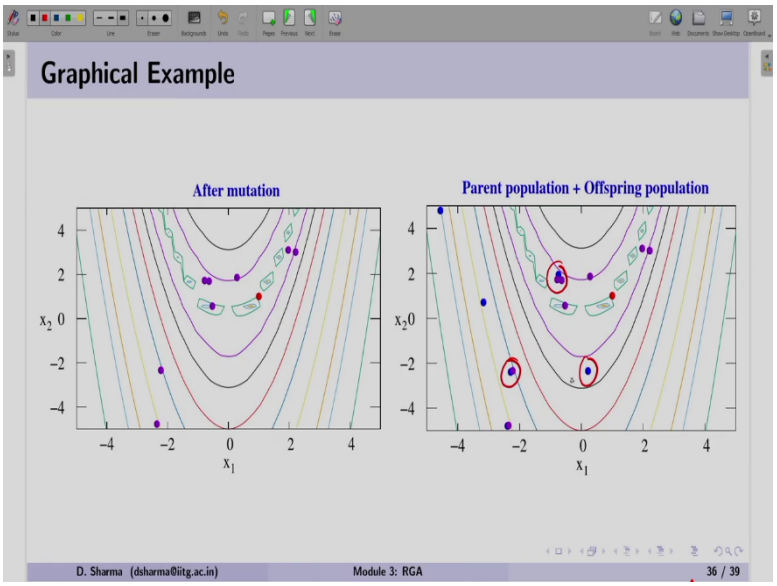
(Refer Slide Time: 64:23)



Now, certain lines have been drawn here these lines says that we are picking one solution from here and one solution from here randomly, so that we can perform the crossover operator. So, the set of so the pair of solutions which we selected for crossover operator same sort of solutions are shown in this figure.

After creating a mating pool we perform crossover operator in this particular session we have used SBX operator that created the solution. Now, you can see these brown points that are generated by the crossover operator. So, once these solutions are created than we perform mutation.

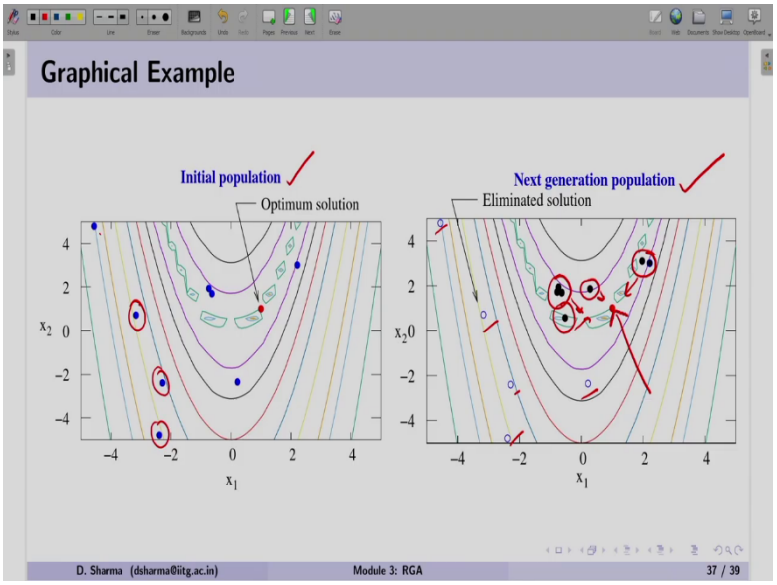
(Refer Slide Time: 65:13)



In our case we perform mutation using polynomial mutation. So, let us see these colours here, now these are the solutions which are generated by the mutation operators so we have 8 solutions. Now at this particular stage we created the offspring population which generally we represent as Q t.

So, this offspring population now has to combine with the parent population, so that we can use the survival stage. So, these 2 solutions this different colour coding will help you to identify that the blue and the purple colour dots those are parent and offspring solutions.

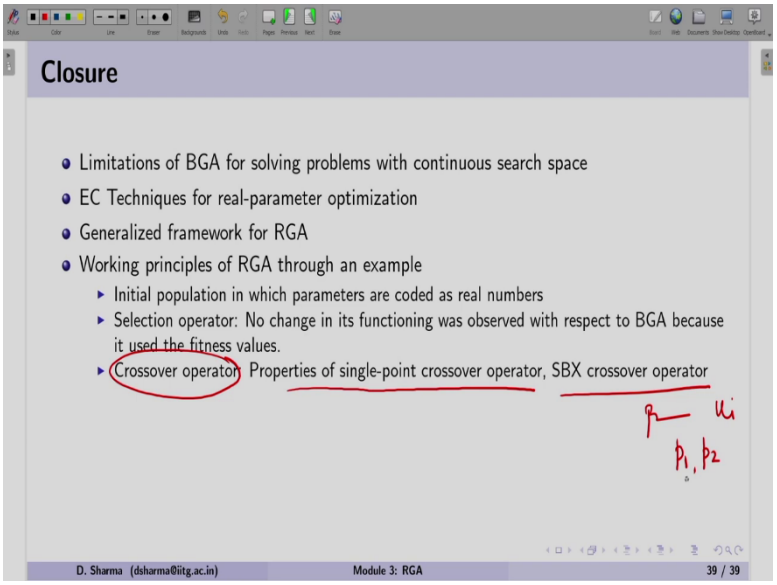
(Refer Slide Time: 65:57)



Now, thereafter we sort the population and we select it, now these are the black dots which are the solution we selected after survival stage. And the solutions which are not filled with a colour these are the solutions which we deleted with respect to the initial population. So, let us compare the initial population with respect to the next generation what you can see that at initial generation these solutions are basically generated randomly in the x 1 and a x 2 plane.

In just one iterations these solutions are moving towards the optimum solution. So, what we can expect that in further generations with the help of selection operator followed by crossover mutation and then the survival stage these solutions will finally, converge to the optimum solution as given here.

(Refer Slide Time: 67:03)



With this explanation on hand calculation and graphical example let us come to the closure of this session. So, what we have done in this session is we started with the limitation of binary coded GA, which makes our search space discreet even though we are working with the real number there are certain issues with the binary corded GA.

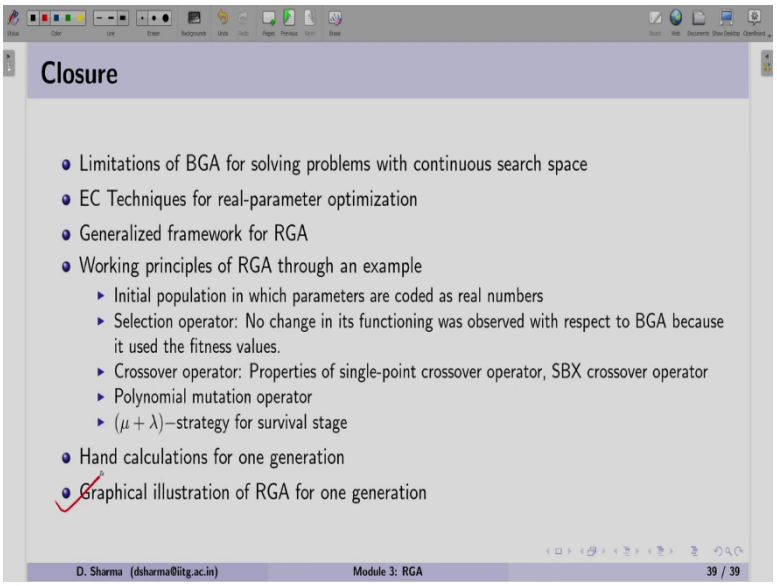
So, in order to remove those issues with binary coded GA there is another class of EC techniques which are referred as EC techniques for real parameter optimization. Then we fit our real coded GA with the on the generalized framework and thereafter we understood this RGA with the help of working example.

So, the first change that we did that all the real numbers or the real decision variables are represented using real numbers, thereafter we have a selection operator. So, this selection operator especially the binary tournament selection operator which we discussed here, we do not find any change its only because it works on the fitness value.

Then we made the structural change in the crossover operator. Why? Because we are now working on a real number rather than on a binary string; so, we use the property of a single point crossover operator. So, average property and a spread property beta that were used and we come up with the SBX crossover operator.

This SBX crossover operator has a non-linear probability distribution moreover, you can find that the beta value basically depends on the random number u i. So, this operator is already stochastic in nature, similarly when we pick solution p 1 and a p 2 that are also we chose randomly thereafter we discussed about the potential mutation operator here.

(Refer Slide Time: 69:19)



Thereafter we use mu plus lambda strategy all these calculations which we have shown that we have done some hand calculation for one generation. So, that we can understand how these operators work; and finally, the graphical illustration of RGA was shown; the purpose is when we see how the solution is started at the initial population.

And after performing selection operator, crossover, mutation and survival stage, these solutions have been moved to the another solutions which are relatively closer to the optimum. With these details I conclude this session on the Real Coded Genetic Algorithm.

Thank you.