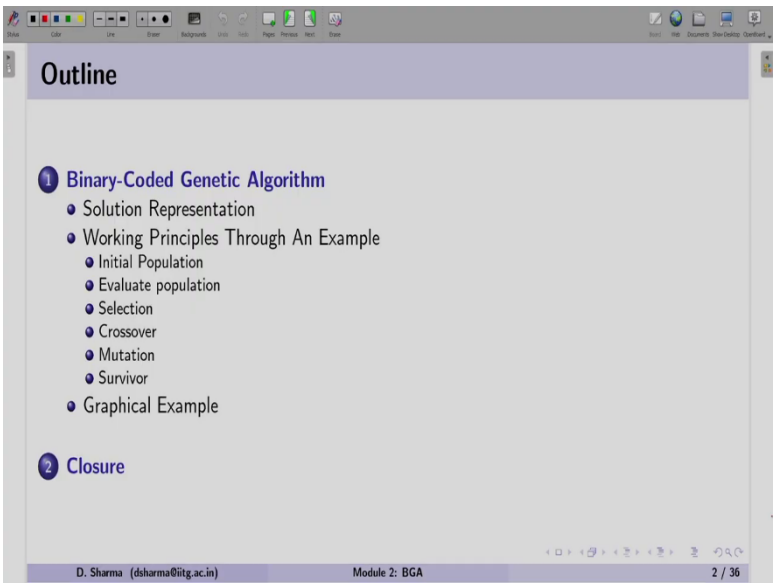
Evolutionary Computation for Single and Multi-Objective Optimization

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

Module - 02
Lecture - 03
Binary-Coded Genetic Algorithm

Welcome to the module 2 that is on Binary - Coded Genetic Algorithm. This particular module is divided into different sessions. In this particular session we will be targeting the introduction of the binary-coded GA. We will learn this particular algorithm using an example.

(Refer Slide Time: 00:58)

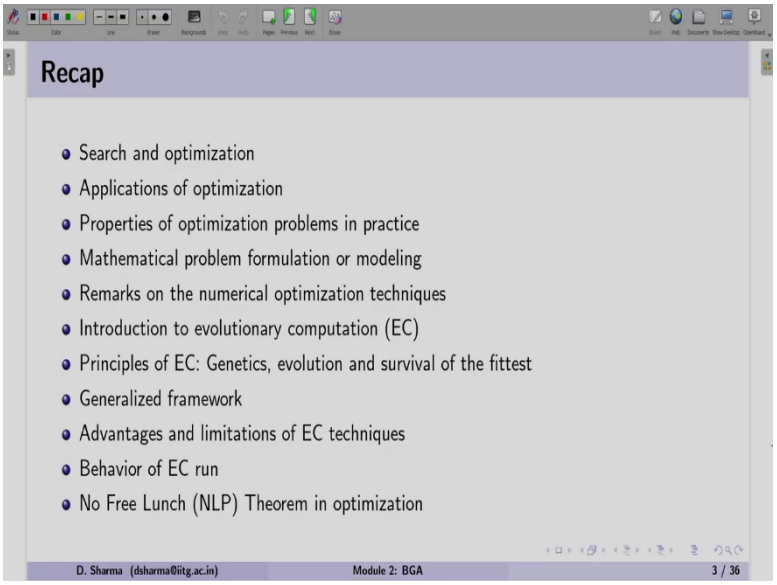


While we will be solving some hand calculation in this example we will go through the initial population after that we will evaluate it, then selection, crossover, mutation and the survival. You will realize that all these operators and the processes we have already discussed with the generalized scheme. We will follow that generalized scheme to understand how binary - coded GA works?

The same example which we have already solved that will be shown as a graphical example and finally, we will conclude this session. Before we start binary coded GA let us have a recap of the module 1.

In module 1 we started with the definition of optimization and searched an optimization, after that we have gone through various applications, those applications have been solved by one of the evolutionary computation techniques and while looking into those applications we realize that there could be different characteristics and properties of the problem.

(Refer Slide Time: 02:05)



So, therefore, we discuss the typical properties, we can have, when we solve any large and complex optimization problems. After finishing that we discussed the mathematical formulation. In that mathematical formulation, we discussed objective function and thereafter the constraints. We can have different kind of constraints, then variable bounds and the variable types.

In that particular formulation, we realize that we can have to identify different parameters. From those parameters, we have to identify the sensitive parameters that will help us to design our objective function and constraints. Those sensitive parameters are called decision variables.

Thereafter, we discussed two types of constraints; one was equality constraint, another one was inequality constraints, afterwards we discussed the objective function now since the objective function can be maximization or minimization. We use duality principle so that we can use the same algorithm for minimization as well as maximization.

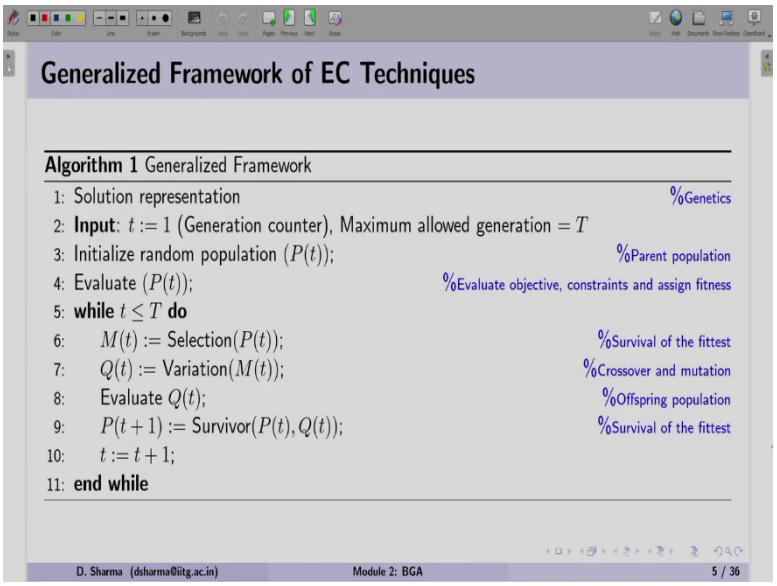
While going forward we also discussed the remarks on the numerical optimization techniques. Looking at those limitations that motivated us to come up with the flexible algorithm and there we have any scope of evolutionary computation techniques.

While discussing the evolutionary computation techniques we have gone through the principle of EC techniques where we discussed about genetics natural evolution and the survival of the fittest. And then we also discussed afterwards about the generalized framework.

We discussed that generalized framework because the same framework we will use to explain different EC techniques. We also talked about advantages and limitation this is important, because we should know at what particular time or what kind of a problem we should use EC techniques as well as when we are solving a problem we should also know what the limitations are.

For example, there are many parameters involved with EC techniques and there is no proof available. Thereafter we have gone through the behaviour of EC techniques on a one-dimensional landscape and finally, we also discussed the no free lunch theorem. Now, we are moving to binary coded genetic algorithm.

(Refer Slide Time: 04:58)



This is the generalized framework which we know as you can see in the step 1 we have to represent our solution and that is generally referred as the genetics. Since it is binary coded GA, we will be representing our solution using binary strain.

Thereafter there are certain input parameters as you can see in step number 2 that we have to set those input parameters for running the binary coded GA. Binary coded GA sometimes also refer to as BGA, once we are setting the input parameters then we have to create the initial population.

After creating the initial population which is generally referred as parent population we evaluate the population in step 4, after evaluating we are in the standard loop of the generation.

Here you can see at the step number 6, we do selection and this is a part of the survival of the fittest this selection we will be performing because we will be making the mating pool M t that will be required or that we will be using for performing the cross over. You can see that we have used the term M t so that is referred as the mating pool.

Performing after using the selection we will be selecting few good solutions and in step 7 we will move to the variation as you remember that will be required to create new solution in the population. So, since it is binary GA we will be looking into the cross over and mutation operators to generate new solutions.

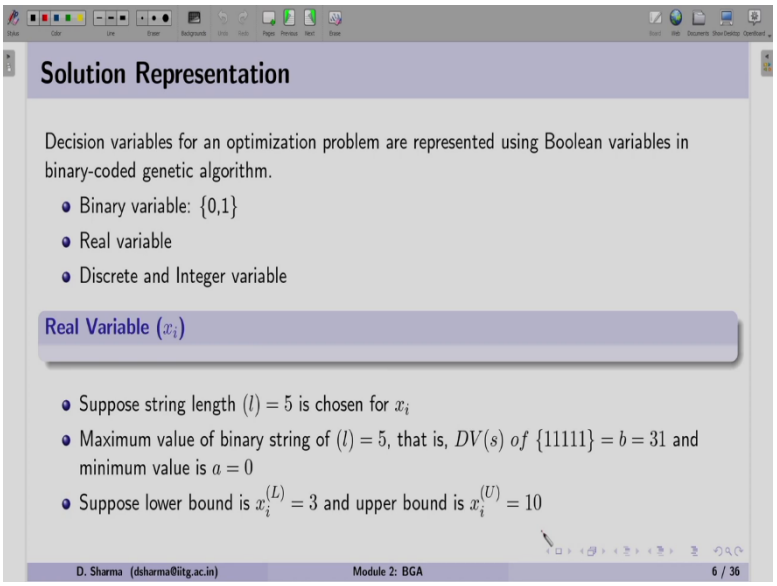
In step 8 we will again evaluate the population it is because the cross over and mutation will generate new solution and those new solution should be evaluated meaning that we have to calculate the objective, function, constraints and we have to assign a fitness.

Now, at the step number 9 you will realize that we have to perform survival, now looking at the two population that is P which is referred as a parent population, another population is offspring population now both the populations are different. What we have to do is either we have to select from one of the population the solution for the next generation or we can combine them.

This decision we will be taking in the survival stage and in this survival stage will see how we will select a good solution that will move to the second iteration. In step 10 we will increase the counter of generation by 1.

We will be working on this particular loop applying those operator one by one till the termination condition get satisfied. As of now the termination condition is decided based on the number of maximum generation. Now, let us move to the solution representation; now as we are using binary coded GA. The decision variables will be represented using the Boolean variables.

(Refer Slide Time: 08:19)



Now, as we know these Boolean variables are 0 and a 1 now these Boolean variables either I can use for 0 - 1 decision making. For example, in a transportation problem when we have to find the minimum cost from moving from source to destination, we can go from source to destination. In that case, we will be looking for those paths that will give us the minimum cost.

In that scenario, we make it 0 because we do not want to go through that path, but some path we want to make it 1. For a one particular path we have to decide whether we have to go or a not. The decision of yes and a no can be represented using the binary variable 0 1.

Similarly, there is a unit commitment problem in electrical engineering where the power generators have to switch it on or not, based on this status like 0 and a 1. There are various examples where we need such kind of binary variable.

Second is we can use the Boolean variables to represent the real variables. The real variables we are taking an example. I will be discussing this in detail in the following slides.

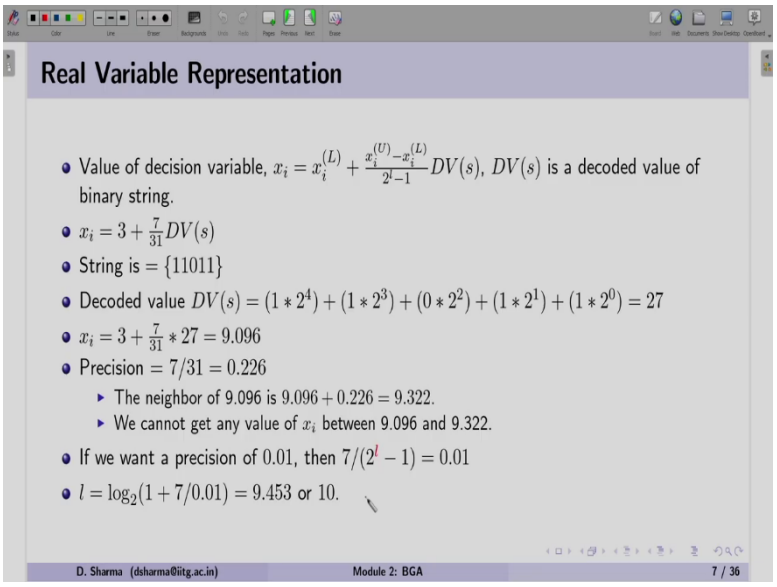
We can also use Boolean variable to represent discrete and integer variable, since these binary strings we are using it those binary string we can easily use to represent the discrete and integer variable. Let us take the example for the real variable and we are going to show how binary coded GA will work.

Let us start with real variable say x_i. Let us take just one variable now for this particular variable we chose that the string length equals to 5 and this string length will represent the value for x I which is real variable. Now, when we are using the string length as a 5 you will realize that if all the bit positions are one then the value of the binary string will be 31. This will be the maximum value of the binary string when all the bit positions are 1.

Similarly, when all the bit positions are 0, then in that case the value or the decoded value of the string is 0. This particular information we will need later in the stage. Since we are solving for a real variable x I let us assume that we have the lower limit on the variable as 3 and the upper limit as 10. Our decision variable should lie between 3 and 10.

$$x_i^{(L)} = 3 \; and \; x_i^{(U)} = 10$$

(Refer Slide Time: 11:18)



To represent a real variable using binary string we will be using this scaling function. What is this scaling function? In this scaling function you can see that x i is equals to x i plus lower bound plus in the numerator it is the upper bound minus lower bound divided by 2 to the power *l* minus 1.

$$x_i = x_i^{(L)} + \frac{x_i^{(U)} - x_i^{(L)}}{2^l - 1} DV(s)$$

l is the binary string here and this particular division will multiply with the decoded value of the string. Let us take an example how it will work.

For the current example you remember that our lower limit is 3 and the upper limit is 10, if I am going to use that value our x i value can be written as 3 plus 7 by 31. Now, how this 7 has come? 10 minus 3 will give me 7 and in the denominator it is 31, why because it is 2 to the power 5 which is 32 minus 1 gives me 31 and then we are going to multiply with decoded value of the string.

Let us assume that we have a string as 1 1 0 1 1 0 1 1 for this particular binary string let us see what is the length what is the; what is the decoded value. As we know how we are going to decode the binary string. When we are decoding this particular binary string here the decoded value will come out to be 27, once we decoded the value we will be in substituting this value in our equation. This equation if we use it, this is the scaling function scaling equation.

Here x_i equals 3 plus 7 divided by 31 multiplied by the decoded value which is 27. Overall x i value will be 9.096. What is the interesting thing you can find that a binary string of length 5 we are using and that binary string is giving us a real number and that real number we can use for our problem.

Now, let us talk about the precision, now this precision as you can see that we have written as 7 divided by 31 how we can get it. Let us look at the formula on the top. This comes from the numerator, the upper bound minus lower bound divided by 2 to the power l. Now, since the for the given problem we know these values. The precision will become 7 by 31. Now, when we calculate this value it is coming out to be 0.226.

Now, what is the idea behind precision or what is the significance of the precision? Now, suppose we are looking for the next number that we can get it using binary string. Here the first statement says that the neighbor of 9.096 is so we have this number plus precision. That is coming out to be 9.322. What it says that if I am going to use a binary string and suppose instead of 27 let me have the decoded value 28. That is the next number after decoding the binary string.

What you will realize that the next number will be 9.322 for x i variable. It means that I cannot have any value between 9.096 and 9.322 which says that the if I am going to use the binary string to represent a real number in this case the search space is already discreet why because we are using those binary string decoding those values and putting into our scaling function as you can see on the top.

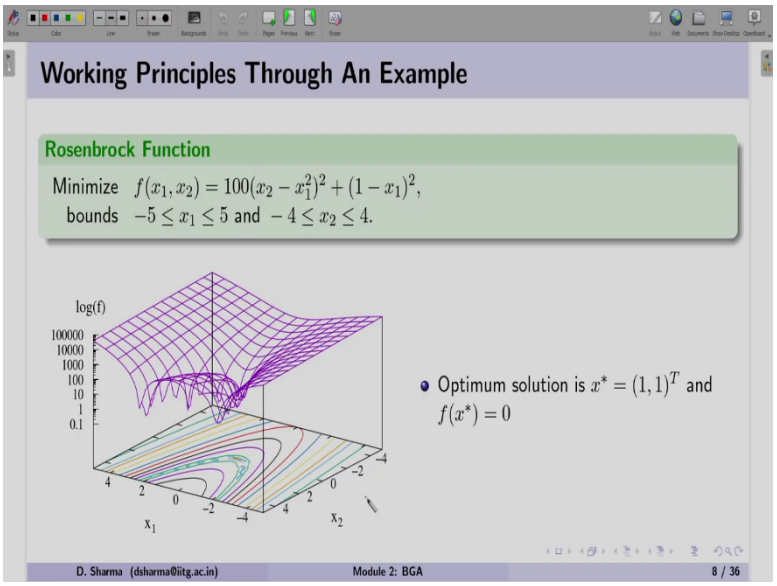
Now, suppose that the precision we are working on right now is 0 it is 0.226 it is not it is not a good for our problem; we want to increase this precision. If we want to increase it then definitely we have to increase the bit the string length, how we can calculate it? We are again going to use the same formula which is written on the top.

In this particular formula, you can see that lower minus upper bound minus lower bound gives me 7 divided by 2 to the power 2 to the power l minus 1. Now, here for precision of 0.01 we are looking for a value of a l that should give me this equivalent precision. So, we can solve this equation as the left hand side is equals to 0.01.

Now, this is the simple equation if we solve it we can get the value of l as 9.453. So, equivalently we can take a binary string length of 10. So, there are certain observation, first observation is if we have to improve the precision then the binary string length will increase. If we are looking for say 10 to the power minus 8, you can see that the binary string length will be too big.

The second observation is that once we use the binary string for the real variable, the space the search space is already discrete. This discreteness comes because we are dealing with the binary string using the decoded value and putting into the value for x i and that is why the search space is discrete already by using the binary coded GA.

(Refer Slide Time: 17:47)



So, let us understand binary coded GA with the help of an example here, now in this particular example we have chosen the Rosenbrock function here. So, you can see the function.

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

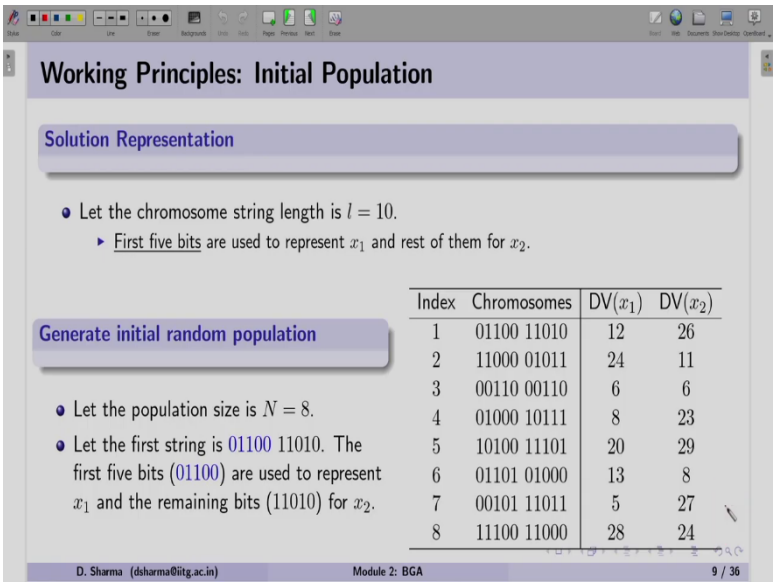
$$bounds\ -5 \leq x_1 \leq 5\ and\ -4 \leq x_2 \leq 4$$

So, Rosenbrock function we want to minimize the function has given here and the bounds we have decided as x 1 will be lying between minus 5 to 5 and x 2 will be lying between minus 4 to plus 4. Now, in this particular figure you can see that the y axis is drawn as a logarithmic scale and x 1 and x 2 as our 2 variables.

Now, looking at the surface here you can see that this particular problem has a many local optimum solutions. So, which you can realize from the contours of this particular problem.

Now, in these problems, there are so many local optima problems; however, the optima of this Rosenbrock function is that it is lying at 1 1 for a 2 variable problem and the function value at the optima is 00. So, this problem is little complex and this is what we will solve using binary-coded GA.

(Refer Slide Time: 19:11)



So, as you remember in our generalized framework we have the first step is called solution representation. Now, in this solution representation we are using a binary string length of 10 as you can see since we have two variables.

So, what we will be doing is the first 5 bits will be using to represent x 1 and rest of the bits will be used for x 2. So, let us see that. So, once we identified that we will use 10 binary string length of 10 let us generate the initial population. So, the after fixing the input parameters we have to start with the initial random population.

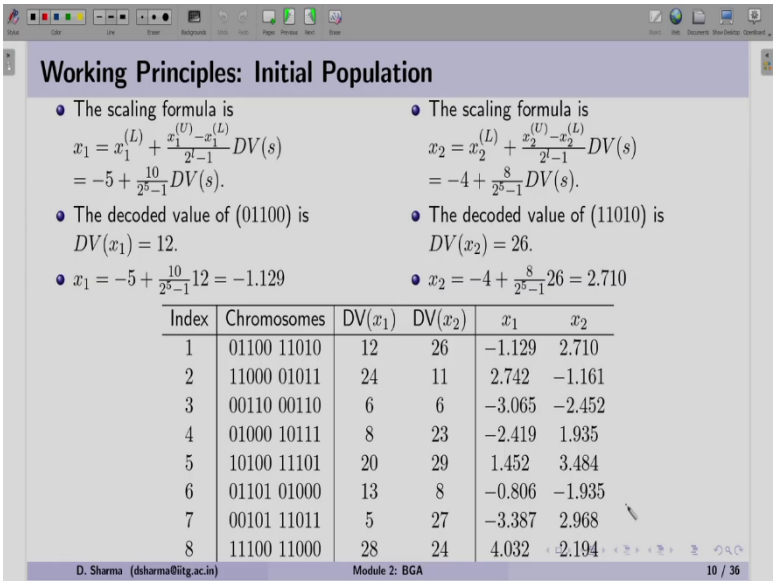
So, here let us assume that we are working with N equals 8; this means we have 8 solutions in the population nowhere since we are using the binary string length of 10. So, I am using two colour coding one is blue, another is black. So, first 5 bits which are colored in blue will be used for x 1 and the black the bits which are in the black color will be used for x 2.

So, we can use our computer that can generate a binary string randomly. So, here what you can see. So, the index 1 will be representing our solution 1 and the chromosome which we have written as a binary string, a computer randomly generates this pattern of 0 and a 0 1.

Now, you know that since we are using binary coded binary string here we have to decode the value. So, if you are going to decode the 5 first 5 bits it is going to give you value 12 for x 1 and if you are going to decode the next 5 bits next 5 bits, then the value you are going to get 26 this process we have to use for the other population members.

So, second member again you will see that we are generated the binary string length of 10. So, the first 5 bit represents x 1 and the decoded value is 24 and the other 5 bits are used for x 2 the decoded value is 11. So, this process will we will continue and we will get all the population. Now, look at the table here that we generated the binary string randomly with the help of a computer. So, those binary strings once it is generated we have to find the decoded value of each of the solution.

(Refer Slide Time: 22:06)



$$x_1 = x_1^{(L)} + \frac{x_1^{(U)} - x_1^{(L)}}{2^l - 1} DV(s) = -5 + \frac{10}{2^5 - 1} DV(s)$$

$$x_2 = x_2^{(L)} + \frac{x_2^{(U)} - x_2^{(L)}}{2^l - 1} DV(s) = -4 + \frac{8}{2^5 - 1} DV(s)$$

Once we have identified the decoded value we have to find out what it is the real number. So, the as we have used one scaling formula. So, the same scaling formula we are using here.

Now, the scaling formula right now. So, here you can see that for x 1 it will be x 1 lower bound plus upper minus lower bound divided by 2 to the power L into the decoded value of the string. Now, since the since the lower and upper upper limit of x 1 is given as between minus 5 to plus 5. So, you can see the lower bound is minus 5 plus in numerator it is 10

because 5 minus 5 will become 10 and then in the denominator you can see it is 2 to the power 5 minus 1 because 5 bits are used for x 1 variable.

So, here you have to you have to pay a pay attention here. So, we have this scaling formula now, using this scaling formula we are going to decode we are going to use a binary string. So, as this is the first 5 bits for x 1 we decode this value put it into the formula we will get as minus 1.129. So, this is the 5 bits that we have used for x 1.

We will do the same practice for x 2 variable, same scaling formula we have used here and in this case since the x 2 is lying between minus 4 to plus 4. So, the lower and upper bound has being change, I purposely change these lower bound. So, that you can make it out make out a difference between the formula which is used for x 1 and the formula used for x 2.

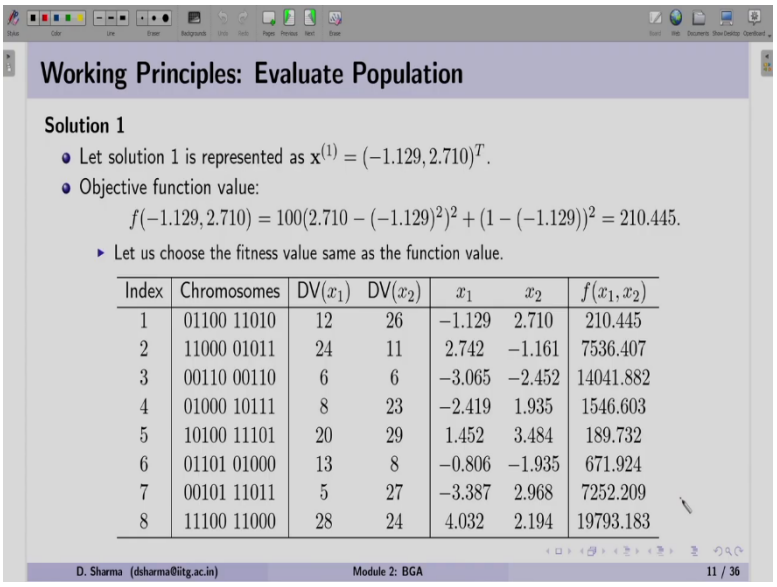
Here, the lower bound is minus 4 plus 4 minus times of 4 will give me 8 in the numerator and again since we are using 5 bits for x 2 variable. So, it is 2 to the power 5 minus 1 into the decoded value of the string, after that we are going to use the another 5 bits decoded value is 26 when we will put this decoded value into our formula we are we get x 2 as 2.710.

So, this is the population this is the solution number 1 and for that one we get the value of x 1 and x 2. If we see the table here now the solution 1 we already decoded the value here and these decoded value using this scaling formula I got the value of x 2 x 1 and x 2. Similarly, when for the solution number 2 we have the binary string we have the decoded value of x 1 and x 2 we can use the formulas given for x 1 here and x 2 here. So, I can use both these formulas to get the value the real number of x 1 and x 2.

So, in this particular example, solution 2 will become 2.742 and x 2 will become minus 1.161. So, we will continue this process and find the x 1 and x 2 value for all the solutions. So, in this particular table you can see that we have identified or calculated the value of x 1 and x 2 for all the solutions here.

Now, once the initial population is generated we have already decoded the value of a binary string, we also converted that binary string into the real number; now the next step as per our generalized framework is to evaluate the population. Now, when we are evaluating this particular population we will we are going to take solution one by one.

(Refer Slide Time: 26:21)



So, for solution one we represent the x 1 and x 2 variable as x 1 now you will realize that the representation is in the term of column vector. So, it is easy to follow. So, we have represented the solution in a column vector and x 1 and x 2 values are written here. Thereafter we already know that the objective function that is the Rosenbrock function.

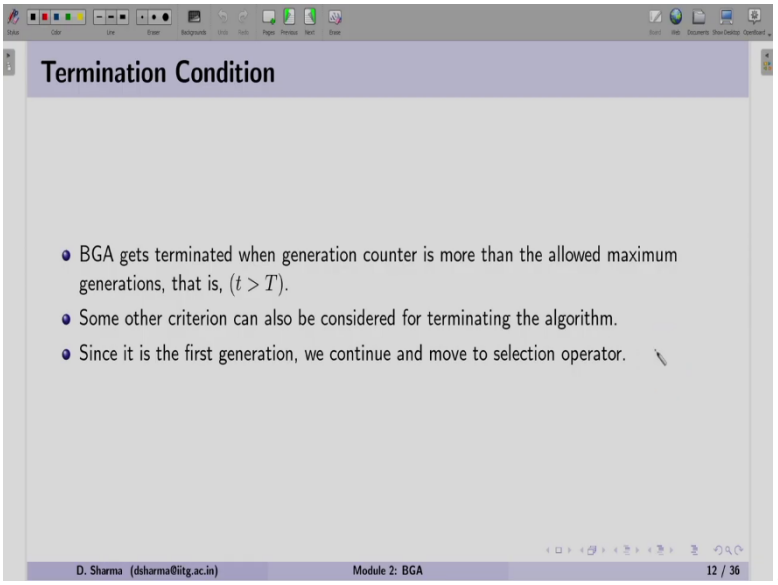
$$f(-1.129,\ 2.710) = 100\left(2.710 - (-1.129)^2\right)^2 + (1 - (-1.129))^2 = 210.445$$

So, Rosenbrock function we are going to put all the values of these x 1 and x 2 the function value that will come out to be 210.445. Now, since we do not have any constraint, we will take the fitness value of solution 1 same as the function value. So, this particular value will become the fitness value for the solution 1, we will follow the same procedure for the other solution.

So, this is the table that we have. So, I am extending this table now you can see the number of columns in the table is increasing so that we can understand. So, we generated the solution binary string decoded the value afterwards we get the value we calculated x 1 and x 2 value now we get the fitness value for solution 1, following the same step we can get the fitness value of solution 2 after substituting the value of x 1 and x 2 into the objective function. So, this is what we are going to get it for solution 2.

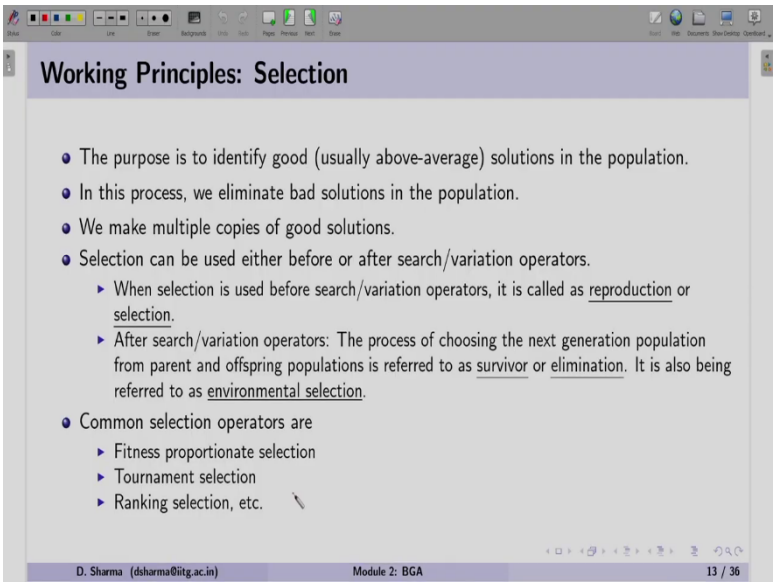
So, in this way we have we can calculate the objective function value for all the solutions. So, all these values you can see will be in these particular ranges, now after finding the fitness value you know that the first step is to decide whether we have to terminate the algorithm or a not.

(Refer Slide Time: 28:19)



Now, the termination condition generally defined using the number of iteration as and when t is greater than capital T we stop our algorithm. In some situations you can also use some other terminate terminating criterion as of now we are stick to the number of generations. Now, since we are at the first iteration. So, we will continue and move to the selection operator.

(Refer Slide Time: 28:51)



Now the first step inside the loop is the selection as you remember with our generalized scheme. The purpose of the selection scheme is to identify good usually above average solution in the population. When we are identifying who are good solutions who are above average solutions or who are bad solution our main task is that we have to remove or illuminate bad solutions.

When we remove them, the places of those bad solutions will be taken by the good solutions. So, we will be making the multiple copies of this good solution. So, that is the overall objective of identifying solutions, making multiple copies of good solutions, and deleting bad solutions.

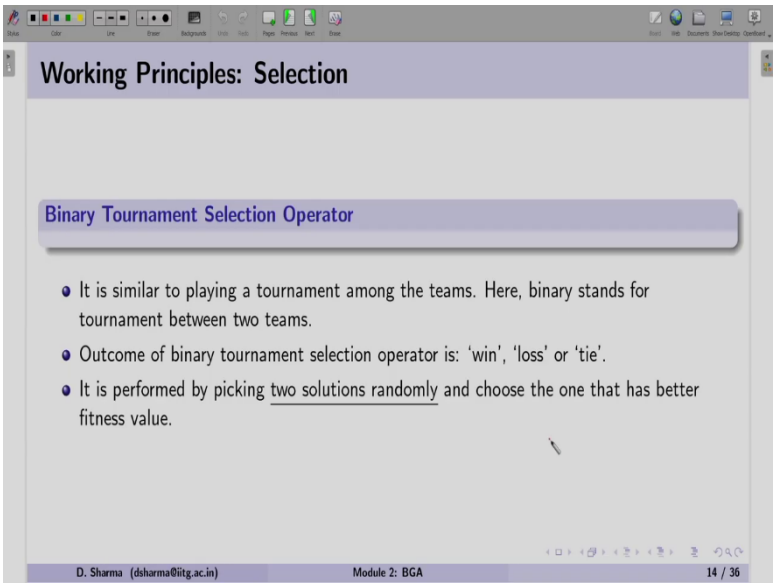
Now, you will realize that the selection can be done either before the variation operator and sometimes after the variation operators. So, when we perform the selection before the variation operator, those selection operators are referred to as reproduction or selection operator. The purpose of performing the selection before variation is to make a mating tool to apply our variation operators to create a new solution on good solutions using good solutions.

The other kind of operators we apply after variation operator are referred to as survivor or the elimination, this survivor and elimination we use it. So, we can choose a good solution from the population that will be that those solutions will be used for the next generation population.

So, we will be emphasizing the good solutions and those good solution will go to the next generation in the while loop of our framework. Sometimes this selection which we done after variation it is also referred as a environmental selection.

There are various selection operators exist in the literature for example, as you can see here we have proportionate fitness selection we have tournament selection or we have a ranking selection and there are few more selection operators. So, in this particular example where we are learning how binary coded is working, let us take tournament selection for selecting the good solution.

(Refer Slide Time: 31:29)

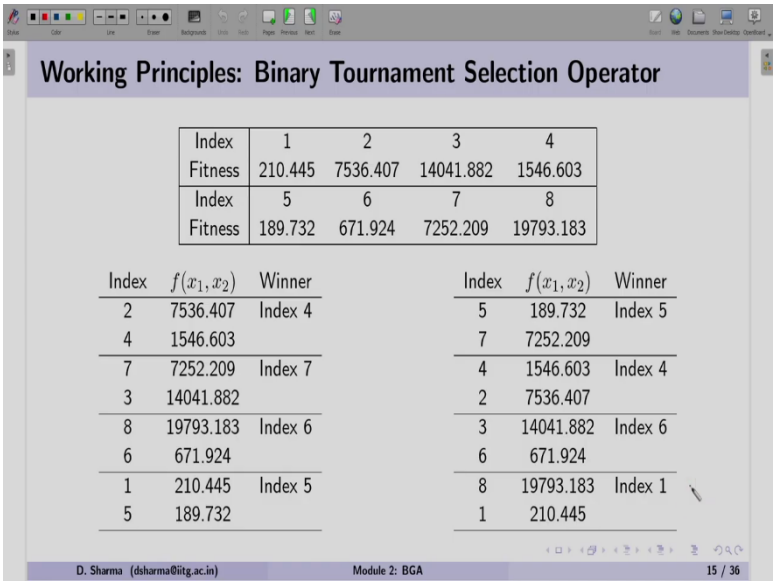


So, what is binary tournament selection? So, let us understand the word now as you know that there is a tournament between the two teams. So, generally we use this tournament word when team 1 is playing with the team 2, when these two teams are playing then this particular scenario is called as a tournament. Since we are performing, we have chosen two teams so they are referred to as binary. So, binary stands for twice.

Now, when there is a tournament between the two team what will be the outcome as you know that either team 1 will be winning over team 2 or team 1 will be losing the tournament over team 2 or there is a tie. So, these three outcomes which is win-loss and tie is used to select solution in binary tournament selection operator.

So, in this tournament selection operator we pick two solutions randomly. So, you can see that this operator is stochastic in nature because when we are performing the tournament, we are selecting two solutions randomly. Between these two solutions, we will choose a better, better in terms of fitness value. So, let us see how this binary tournament selection work.

(Refer Slide Time: 33:01)



Now, this is the table which you can see here; now, in this table, the index is written. So, 1 2 3 4 and their corresponding fitness value which we have calculated earlier are also mentioned.

Similarly, for the other four solutions, so total eight solutions and their fitnesses are written now, as you remember, in binary coded in binary tournament selection operator, we have to randomly pick two solutions.

So, let us assume that we picked two solutions 2 and a 4, now when we pick 2 and a 4 solution their fitness I have written why because this fitness will be useful or it is required to find which is the who is the winner. Now, since the fitness value of solution 4 which is 1546 is less than the fitness value of solution 2.

So, the index 4 or the solution 4 is the winner. So, we are going to do what we are going to do since we have selected 2 and a 4 we are removing from the pool. Now, we are left with six solutions, among those six solutions picked, we pick two solutions randomly this time let us assume that we picked solutions 7 and a 3 look at their fitness value, after comparing their

fitness value you can make it out that the solution 7 is the winner so this solution will be selected.

Now, since four solutions have been removed we are left with four solutions, among the four solution let us pick 8 and a 6 as randomly. Once we selected it we are going to compare the fitness value looking at the fitness value you can make it out that the solution 6 is the winner. Now, at the last we are left with two solutions that are 1 and a 5 now 1 and a 5 looking at their fitness value you can make it out that the solution 5 is the winner.

Now, there are certain observations here, first observation is the binary tournament selection operator which I have defined that is the binary tournament selection operator without replacement, without replacement means that as and when we selected 2 and a 4 for performing the tournament we are removing. So, we are left with six solutions then we are randomly picking two solutions once we selected one then we are removing these two as well.

So, we are not replacing any solution that is already selected for binary tournament selection. This particular tournament selection without replacement will help us or help all the solution to participate in the tournament. Second observation is that after performing the binary tournament selection, we have selected four solutions; however, the population size is 8.

So, we have selected since we have selected 4 what we have to do is we have to perform one more time the tournament selection. So, that we can get another four solutions so that our populations population size will become 8.

So, in that case the eight solutions which we have used earlier we again mix up. So, again we will refer this particular table assuming that all solutions are available to us. Now, in the second tournament let us pick two solutions randomly and suppose these two solutions are 5 and 7, when we are selecting 5 and a 7 comparing their fitness value you know that solution 5 is going to be the winner.

Then among the six solutions the remaining six solutions we picked 4 and a 2 in this 4 and a 2 comparing their fitness value you can make it out that the solution 4 is the winner. Thereafter, we have to randomly pick another two solutions from among the set of four. So, let us assume that we picked solution 3 and a 6 comparing their fitness value solution 6 is the winner and finally, we are left out with 8 and a 1 looking at their fitness index 1 is the winner.

Now, there are certain observation when we perform the tournament selection. So, here our observation is related to binary tournament selection. Look at the table on the top now in this particular table you can find who is the solution that has the best fitness comparing all the fitness value you will realize that solution 5 is the best solution among all the eight.

Now look at the number of copies given to the solution 5, now solution 5 you can see in tournament 1 we get one copy, in tournament 2 also we get one copy, apart from solution 5 we have two copies of solution 4 similarly, we have two copies of solution 6. Now, let us look at another solution that is worst. So, among the eight solutions here if you compare which one is the worst solution, now looking at the fitness value you can realize that the solution 8 is the worst solution based on its fitness value.

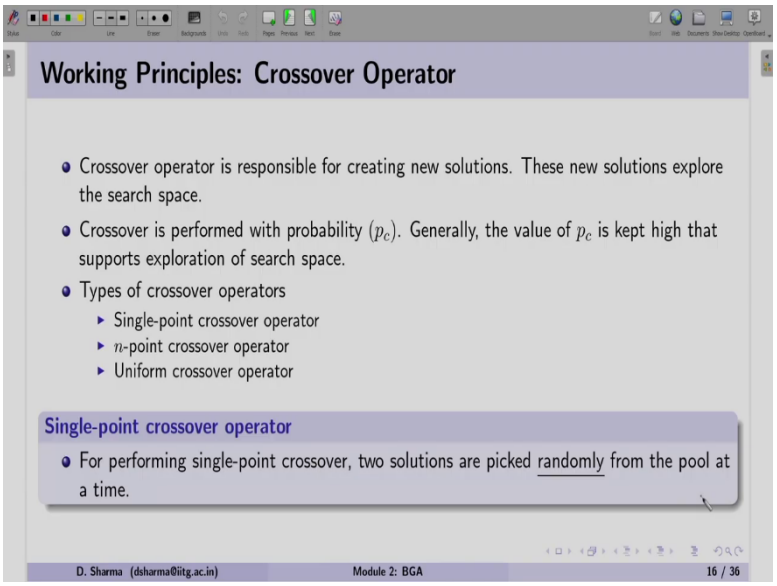
Now, let us see when we perform the tournament 1 is there any copy. So, you will realize that there is no copy of solution 1 8 sorry because 8 is the worst solution here. Similarly, if you look at the copy of solution 8 here, there is no copy here since it is worst. So, there are two observation, first few solutions are getting two copies, some solutions are getting one copy and some solution are not getting any copy.

Now, let us discuss the worst solution now since the worst solution is 8 as and when it is compared with any of the solutions, it will lose the tournament. So, whether it is tournament 1 or it is tournament 2 in both the cases, the worst solution will lose. So, they therefore, we are not going to get any copy of the worst solution, which is evident from this particular table.

Now, look at the best solution now as and when this solution 5 will be competing it will always be the winner, either it is in tournament 1 or it is in tournament 2 this says that the best solution will always get a two copy, because we are performing two tournaments tournament 1 here and tournament 2 here. So, since we are performing twice, the best solution will get a two copy and worst solution will get zero copies.

Now, how about the other solutions, why because solution 4 and solution 6 also got two copies. So, for the other solution we do not know whether they are going to get two copy, one copy or a zero copy, it is only because we are choosing the solution randomly. So, this selection operator's stochastic behaviour will decide whether the other solutions apart from the best and the worst they are going to get two, one or a zero-copy.

(Refer Slide Time: 41:01)

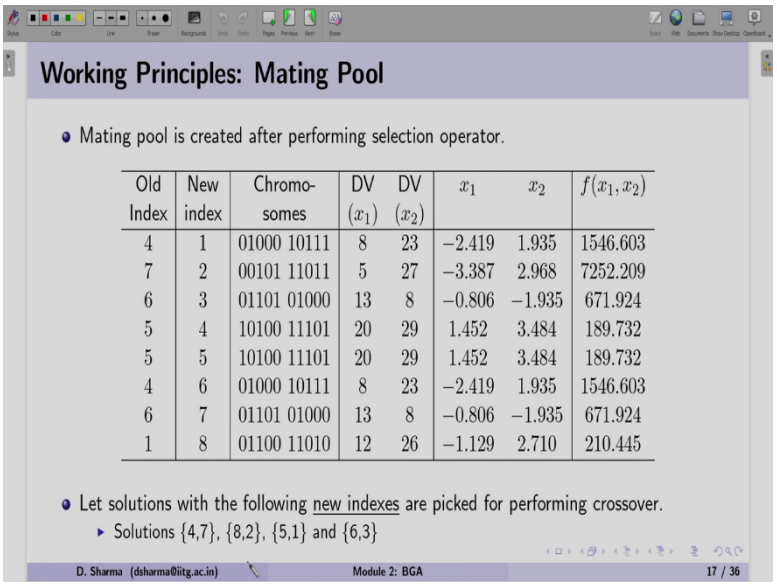


Once we have performed the selection of these good solutions, we will make a mating pool, now once those solutions are selected, we will perform crossover on it. So, what is the purpose of crossover? Operator crossover operator is responsible for creating new solutions. So, these new solutions are basically exploring the search space. Exploring means we will be creating or the crossover operator will be creating the solutions in those regions where there is no solution present.

Thereafter ah, what you will realize that generally we perform crossover operator with a probability p c. So, this p c probability of a crossover we keep it generally high for the problem. So, you will realize that we take crossover probability has 0.8, 0.9 and a 1. So, we are keeping high this is the thumb rule we generally follow. There are different types of crossover operators available in the literature for example, single point, n - point, uniform crossover operators.

So, in this particular example let us take single point crossover operator what is that, in the single point crossover we picked two solutions randomly. So, you can see that this operator is also stochastic in nature why because we are picking two solutions randomly, when we pick two solutions randomly from the pool we have to perform the crossover. So, let us see how.

(Refer Slide Time: 42:38)

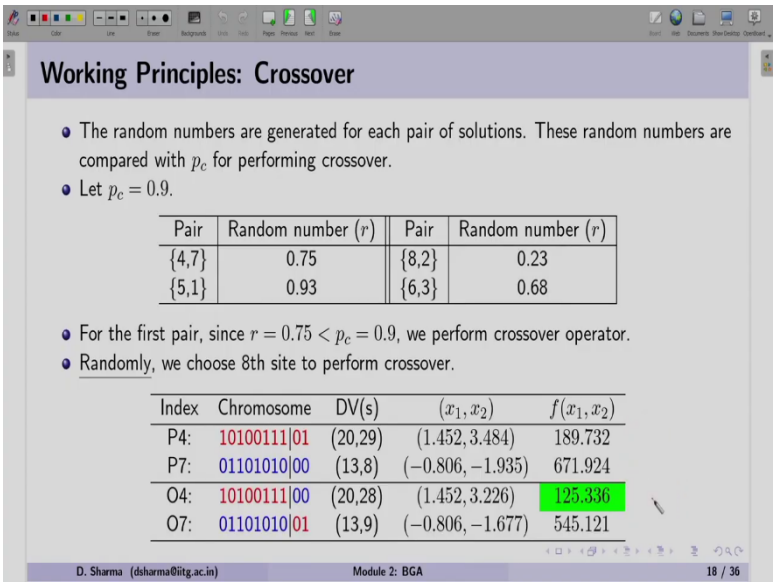


So, as for our reference this is the mating pool that we have created after binary tournament selection. Now, look at the old index first, now this old index is the solutions selected randomly by binary tournament selection. So, 4 5 6 5 this is the first sequence of tournament selection, 5 4 6 and 1 that was the sequence of another tournament. Now, we are putting all the two solution in the same sequence.

Now, to perform the crossover operator we are giving a new index as 1 to 8. So, in the following slides we will be following these new indexes when performing the crossover operator. So, the chromosome is straying x 1 x 2 value and their fitness values are given in this particular table, now as you know in the crossover operator we pick solution in a pair.

So, let us take the new indexes and we pick two solutions in a pair. So, randomly we will be picking 4 and 7. So, the solution 4 and solution 7 we picked it for crossover, thereafter we will pick 8 and 2 for performing the crossover 5 and 1 and finally, we will perform crossover between 6 and a 3 solutions.

(Refer Slide Time: 44:07)



Since we have the probability of a crossover whether to perform crossover or a not we generate random number. So, this random number is generated for each pair of the solution why, because this random number will tell us whether we want to perform the crossover or a not.

So, let us assume that the probability of a crossover is 0.9 as you can see here and the pair of the solutions say 4 7 the random number is say 0.5 for another pair is 0.23, 0.93 and 0.68. So, these are the random numbers we are going to use to perform crossover operator.

So, let us chose the first pair now here in the first pair since the random number which is 0.75 is less than the probability of a crossover which is 0.9 we perform the crossover operator. Now, here we randomly choose one of the sites say 8th site now as I am including the random the word called random you can see that even for performing the crossover we have included the stochastic nature into this particular operator.

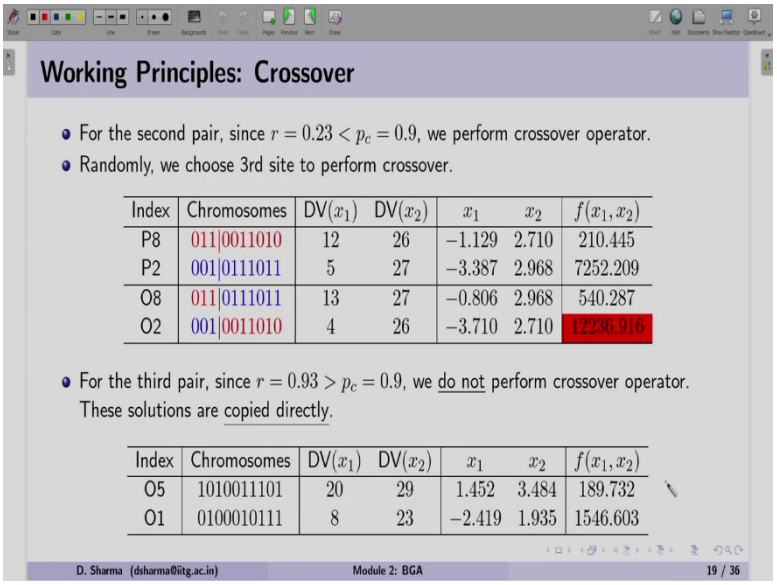
Now, here the index P and P 4 and P 7 are written P stands for parent. So, 4 and 7 parents are chosen these are the binary string I am using two color coding here just to make out how we are performing the single point crossover. Now, the 8th site as you know we are picking randomly. So, if you count this is the site 1, site 2, 3, 4, 5, 6, 7 and 8th. So, here at the 8th position we have drawn this particular vertical line and just for the understanding we have written other values as well.

Now, in this crossover operator you remember that in module 1 we discuss that the tail will be switched. So, you can identify from the color coding that for offspring now here O stands for offspring 4 is the same index number. So, O 4 and O 7 now for O 4 we have red then blue.

So, we have swapped the tail. Now, suppose you want to decode this particular string so you get the decoded value of O 4 and O 7 as given here thereafter we have to find what is the real values of x 1 and x 2 using the decoded value, after doing that you can see the fitness for O 4 is 125.336 and for the other solution it is 545.121.

So, what is the observation here is that when we are performing a crossover this crossover operator can generate a better solution than it is parent solution. So, this offspring is actually better than it is parent solution that is the one observation here.

(Refer Slide Time: 47:26)



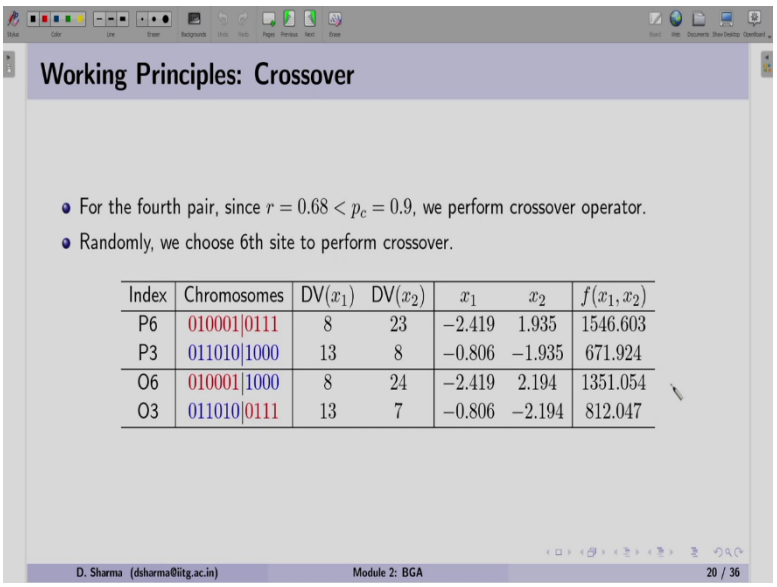
Now, let us take another pair, now here for the other pair the random number was 0.23 which is less than the probability of a crossover so we will perform it. Now, to perform the single point crossover we have chosen the third site randomly to perform it.

Now, another pair, the second pair was 8 and a 2 we have this binary string now the third site you can see as the vertical lines we have used it. So, you know that we are going to swap the tails. Now, in this case you can get 2 offspring as O 8 and O 2. So, when you are swapping the tail you can get the decoded value of O 8 and O 2 as given here, similarly we can get the value of x 1 and x 2 for both the solutions and get the fitness value.

Now, what you will realize, that in the first case when we pick two solutions perform the perform the single point crossover operator one of the solutions was better than the parent, here when we perform the crossover operator it is actually it is generated the worse solution than the parent solution. So, we can say that since the crossover operator is stochastic in nature, it can generate good and bad solutions.

Now, coming to the third pair now here, it is important to note that the random number is 0.93 which is greater than the probability of crossover since we do not perform crossover. So, since we are not performing what we can do, we will directly copy the same solution. So, instead of writing P 5 and a P 1 I have written O 5 and O 1 and the same solution same values we have copied we have copied for this pair.

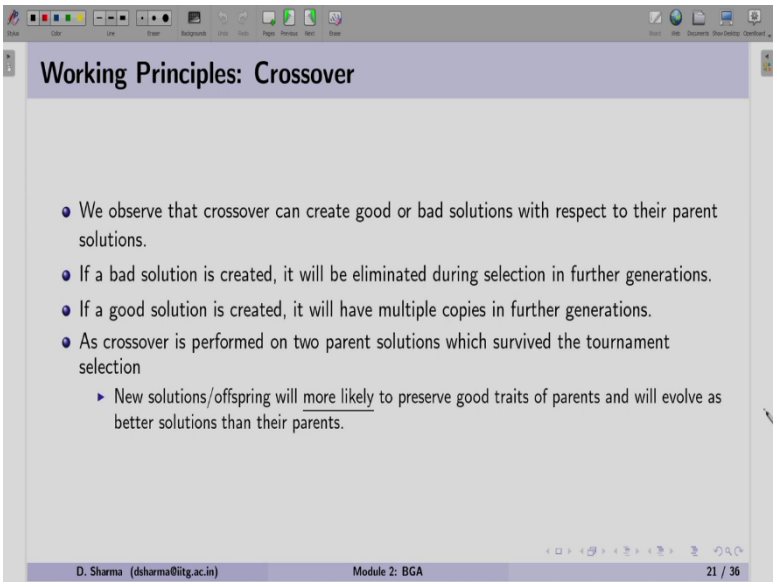
(Refer Slide Time: 49:30)



Now, move to the 4th pair in this particular pair the random number is 0.68 which is smaller than the probability of a crossover. So, it means that we are going to perform it. Now, here for performing the single point crossover we choose sixth site. So, as you can see, the parent 3 and a 6 are given and at the 6th site, we will swap the tail.

Now, when we are going to swap the tail we are getting the new solution as O 6 and O 3 these are offspring solutions. So, similarly we can decode the value of those strings for x 1 and x 2, we can calculate what x 1 and x 2 you can see here and finally, we can find what is the fitness value for these solutions is.

(Refer Slide Time: 50:22)



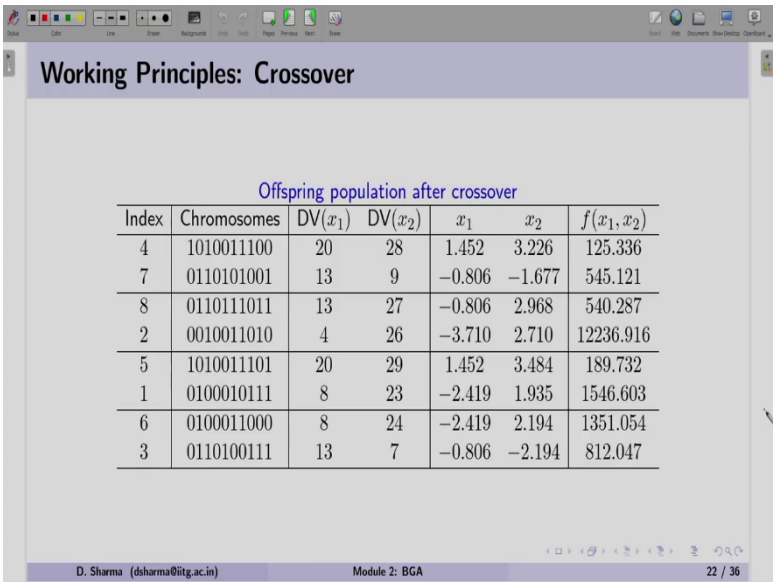
So, from this crossover operator, we have many observations that first of all, the crossover operator can generate good and bad solutions since it is stochastic to generate good and a bad. Now question comes that if it is generating good solution it is good for the algorithm, but if it is generating bad for the bad solution for the algorithm then how we are going to deal with it.

So, as we know that if the solution is bad and we have a one selection operator this bad solution will be eliminated in the further generation. Similarly, if there is a good solution, this good solution will get multiple copies because of the selection operator. So, we do not have to worry whether the crossover operator is generating good or a bad. So, let it be stochastic in nature good solution will be emphasized by this selection operator in further generation and bad solutions will be deleted.

Now, there is a important point which you can see here that a crossover is performed on two parent solutions that survive the tournament selection, this means that those solutions which are in the mating pool after binary tournament selection operator they are good in some sense and when we are performing the crossover operator. This means that these new solutions more likely to preserve good traits of parents and will evolve as a better solutions than their parent.

So, since we are writing more chances, this means that there is a higher probability that when two good solutions are chosen for crossover, the offspring solution will have good traits and generate even better solutions. That is what we have observed from the crossover operators.

(Refer Slide Time: 52:24)

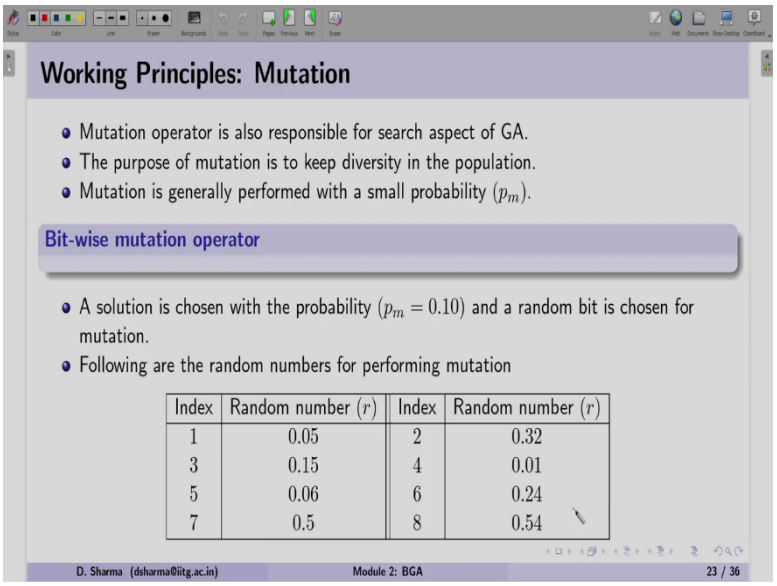


So, these are the offspring solutions which you can see here. So, the index 4 7 8 2 is the same. So, we are not changing here and decoding the decoded value and the x 1 and x 2 value similarly their fitness value. So, you remember that we are doing when we are evaluating it we are following the same sequence.

Now, here it is important to note that we are showing the fitness value as well as the decoded and x 1 value, when we are using binary GA we do not need the column corresponding to the decoded value x 1 and x 2 and the fitness, why because we perform the crossover operator randomly and whatever the solution is generated we take it.

So, this particular column decoded value x 1 and x 2 and the fitness value we are showing here just to explain how the crossover operator is working, but actually, we do not need it.

(Refer Slide Time: 53:27)

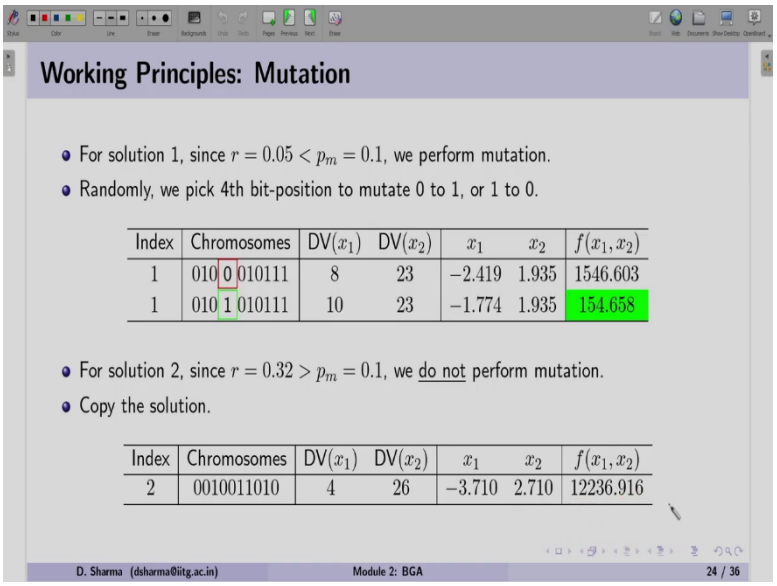


Now let us move to another operator. So, you remember that in variation we have two operators; crossover and mutation. So, crossover operator will generate new solutions and on those new solutions we perform mutations, mutation is generally referred as like perturbing a solution in the close vicinity. So, what is the purpose, again this mutation operator is responsible for the search aspect of GA, the purpose is we have to keep the diversity.

So, performing the mutation changing the bit position will introduce diversity because this diversity will help us help the algorithm not to stuck in any of the local optima. Mutation as you can see it is performed with the low probability. Now, for this example where we are solving a Rosenbrock function using binary coded GA we have taken bit wise mutation as an example here.

Now, bit wise mutation means that we are going to choose a bit randomly and we will mutate, now here for doing this analysis we are taking the probability of mutation as 0.1 which is quite less. Now, for performing the mutation operator you know that we need a random number because that random number will be compared with a mutation probability. So, for the different solution 1 2 3 4 to 8 the random numbers are generated. So, let us see how this mutation operator will work.

(Refer Slide Time: 55:07)



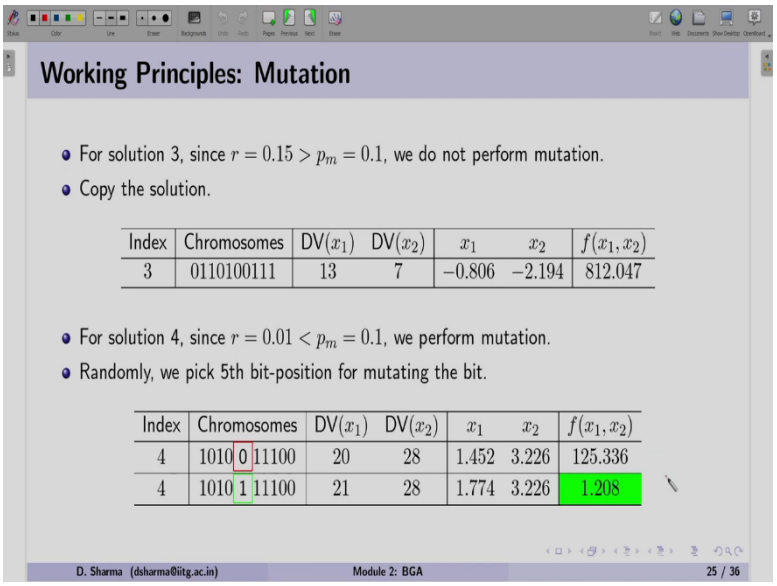
So, we pick the solution number 1 here since the random number here you can see 0.05 is less than the probability of a mutation we will perform it. So, in this mutation, we are going to do so if we pick a bit of 0 then we are going to convert we are going to convert into 1, or 1 to 0. So, let us see how.

For performing this mutation we have randomly. So, again this word randomly is coming it means that mutation operator is stochastic. So, we pick 4th bit to perform the a bitwise mutation. So, this is our solution number 1, in this solution 1 you can count it at the 4th position I have drawn a square here this particular bit will mutate the values of x 1, x 2 and fitness is given.

When we are mutating it so 0 will become 1 and you can see the decoded value of the new solution and the x 1 and x 2 and the fitness now what you can realize that after performing the mutation the fitness has improved, let us see the similar situation or the scenario with the other solutions.

Now, take the solution number 2, now here the random number is 0.32 which is greater than the probability of mutation. So, we do not perform it, since we are not performing it so we will we will copy the same solution which we found after crossover. So, the solution 2 after crossover is selected as it is.

(Refer Slide Time: 56:51)

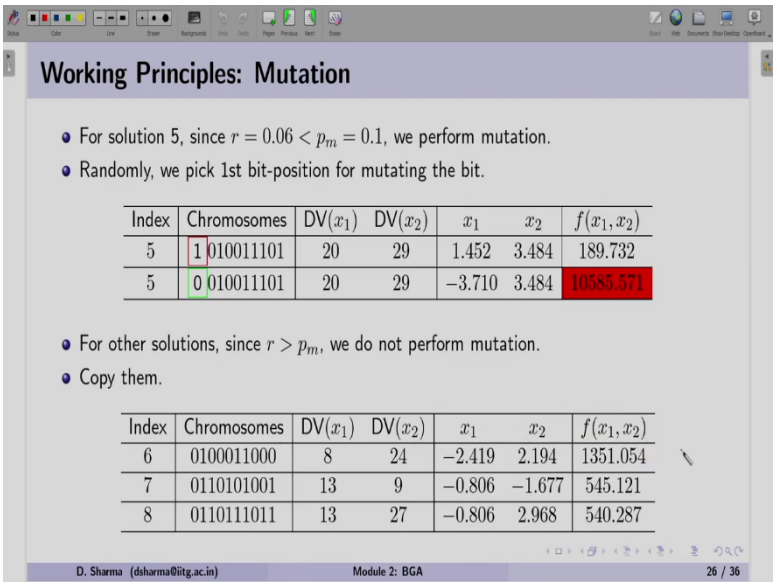


For solution 3 the random number is again greater than 0.1 we do not perform the mutation we copy the solution as we found after the crossover. Now, let us move to solution 4, in this solution 4 the random number is smaller than 0.1 meaning that we will perform mutation here nowhere when we are performing randomly we are picking the 5th bit position.

Now, in this 5th bit position you can see from this solution number 4 we have chosen the 4th bit. So, current currently it is 4 when we are going to perform the mutation it will be converted into 1.

Now, look at the decoded value of the binary string for x 1 and x 2 similarly the real values and we get the function value 1.208 which is quite smaller than it is parent. So, in the two case scenarios our objective function or the fitness value has improved by the mutation operator. Now, solution 6 has a random number less than 0.1.

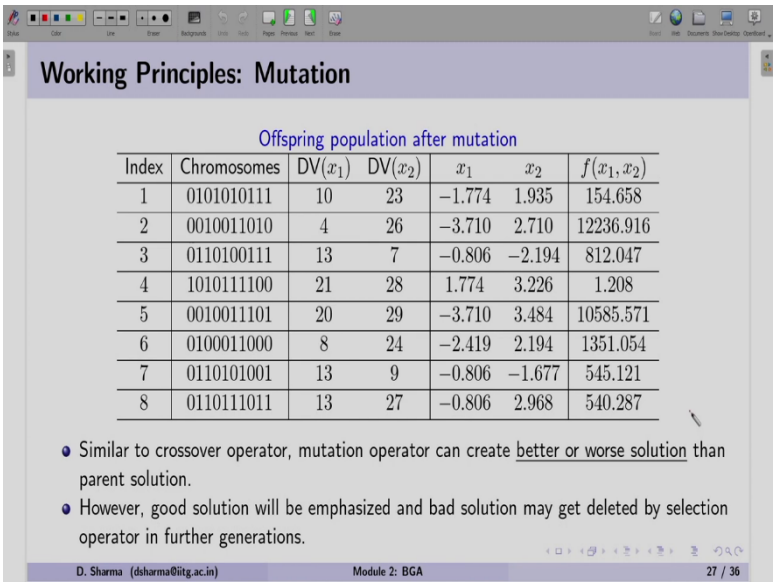
(Refer Slide Time: 57:55)



So, we perform it. So, let us pick the first bit as for performing the mutation. So, we picked it randomly this first bit now you know that since it is 1 it will be converted into 0. Now, you can see the decoded value for these strings and then decoded and the values of x 1 and x 2 finally, we get the fitness value. Here you realize that the fitness has increased tremendously. So, this means that the mutation can help generate the good solution and sometimes it can generate bad solution with respect to the parent solution.

For the other three solutions you will realize that the random number was greater than p m. So, we will not perform the mutation and select all these solution 6 7 8 as it is into our offspring population.

(Refer Slide Time: 58:55)

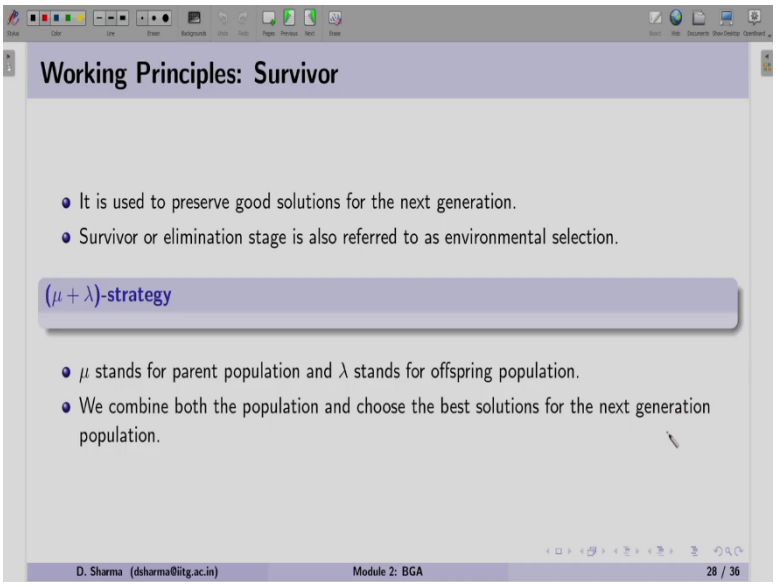


So, this is the offspring population that we get it after mutation. So, we are keeping the same index as we have used in the crossover and these are the new strings. So, you can see that crossover and mutation together can change the string. So, correspondingly x 1 and x 2 values are change and the fitness values for all those 8 solutions have been changed.

As we have discussed earlier the observation is this mutation operator can create better or worse solution than its parents what we have observed with the crossover both of them are stochastic and as the argument given earlier if the mutation operator is creating good solution then this good solution will be emphasized in further iteration to make multiple copies, but if it is generated bad solution, then this bad solution will be deleted by the selection operator.

So, we do not have to worry whether the crossover and mutation operators are generating good and a bad let them generate solution based on the stochastic nature. Now, coming to the survival stage, now survival stage is important why because you realize that we started with a population p which we say as a parent population and after performing crossover and a mutation we get another set of solution this population we referred as offspring population.

(Refer Slide Time: 60:35)

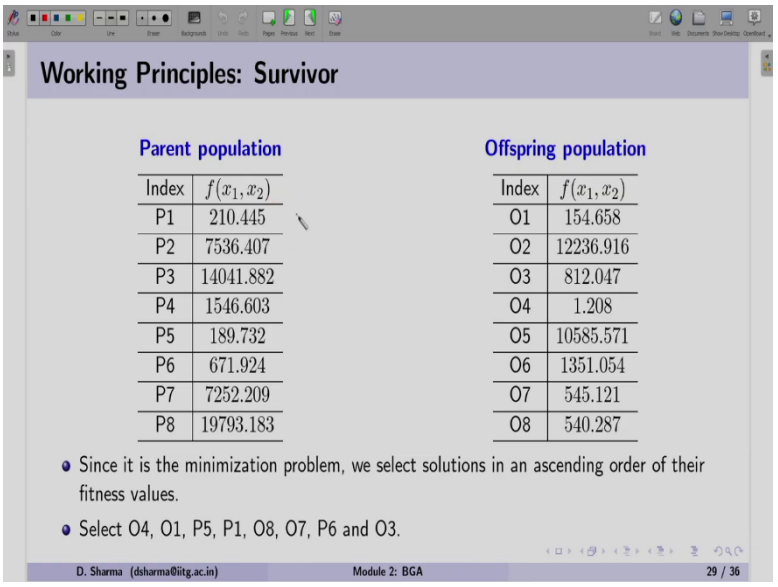


So, since we have two population members. So, we have to select the solutions that will be selected for the next generation. So, we can preserve the good solution for the next generation. As I have mentioned earlier that this particular stage is also referred as elimination stage or environmental selection. For performing the survival stage we are using one of this strategies called mu plus lambda strategies.

So, what is this mu plus lambda strategy that mu stands for the number of parent solution in a parent population? Mu lambda stands for number of solution in the offspring population. So, mu plus lambda says that let us combine parent population and offspring population and choose the best.

So, we will choose the best solution for the next generation, in this case what you will realize that this algorithm in by using mu plus lambda strategy it is elitist, elitist means that we are keeping the good solutions in our population. So, let us see how this mu plus lambda strategy works.

(Refer Slide Time: 61:49)



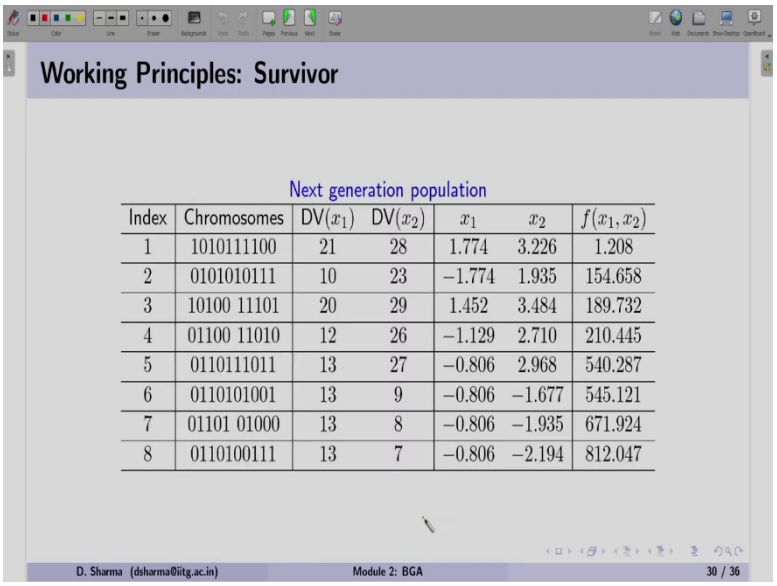
So, I have for our references we have I have shown the parent population. So, with 1 2 3 I have included P. So, that we can differentiate. So, P stands for parent. So, parent 1 to parent 8 their fitnesses are given here.

Similarly, after mutation, the population generated is referred to as the offspring population, which is why I have included O, the offspring. So, O 1 to O 8 and their fitness are given here. So, since we are solving the minimization problem for Rosenbrock function. So, we are going to select the solution based on the ascending order of their fitness.

Now, if you look both of the table you can see that the solution O 4 is the best. So, that should be selected first. So, if I follow this procedure you will realize that first we will select O 4 because it has the minimum fitness, then we will select O 1, then P 5 and after P 5 we will select P 1, then solution O 8, then O 7 and finally, O 3.

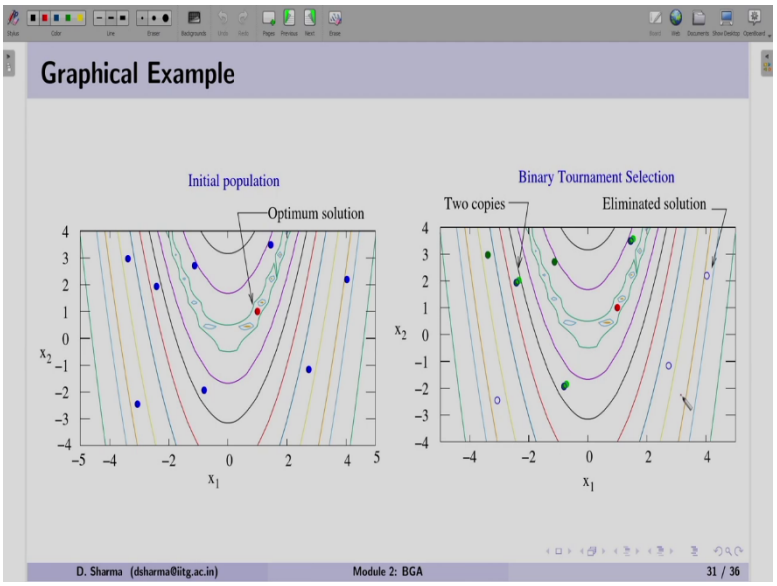
So, you can realize that all these solutions are arranged in the ascending order of their fitness. Now, how many solutions we are selecting here since our population size is 8. So, we are selecting 8 solutions here.

(Refer Slide Time: 63:20)



So, this is the next generation population after the survival so the index number is given their chromosomes the decoded value of their strings for x 1 and x 2, the value of x 1 and x 2 the real numbers and the fitness of those solutions. So, this is by hand calculation in one generation you can see that few solutions have been improved from the parent solution. So, let us see graphically how these solutions are improving and moving.

(Refer Slide Time: 63:52)



So, a graphical example is presented here in this example we have x 1. So, I am showing you the contours of the Rosenbrock function and we have x 1 variable and x 2 variable here.

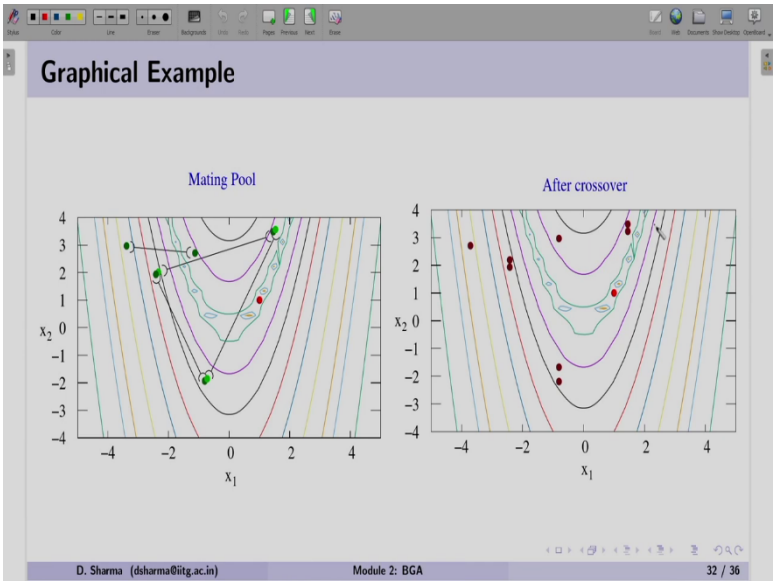
These blue dots represent the parent population which we started there and the red dot here you can make it out that it is our optimum solution.

So, the objective is we have to use binary coded G algorithm to converge to this optimum solution. So, in this graphical example we will show the one iteration of BGA after initial population and finding the fitness of these solutions we perform binary tournament selection.

In binary tournament selection you remember that few solutions will get two copies one copy or some solution will get 0 copies. Now, looking at here for example, this particular solution got two copies similarly this solution got two copy and similarly this solution got two copy. So, these are the same solutions that we have seen when we perform the binary tournament selection in the previous slides.

There are other solutions now in the initial population we use these blue dots now I am showing you the blue circles this solutions are eliminated. So, we have three solutions which are eliminated by the binary tournament selection and you know when they part they took part in the tournament they lose the tournament that is why they have 0 copies. So, these means these three solutions are deleted in the population.

(Refer Slide Time: 65:42)

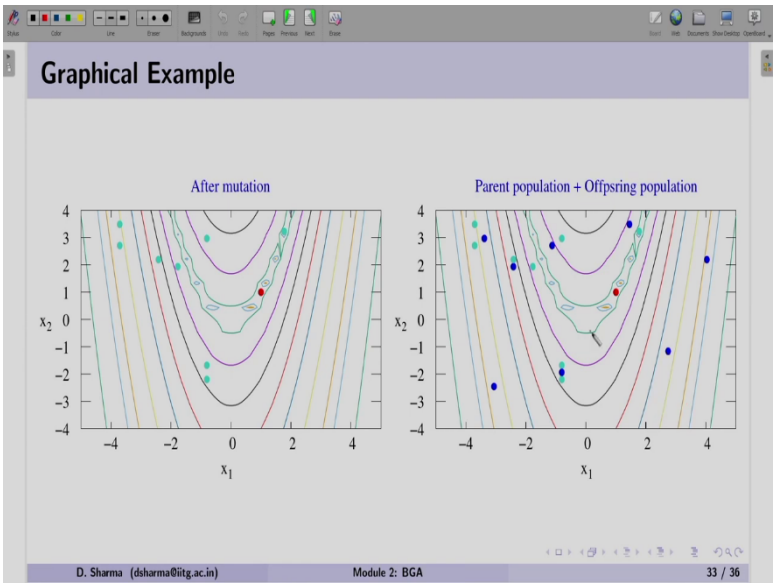


Once we selected we have this mating pool now you can see the two solutions here and the rest of the solutions. So, these lines have been drawn to tell you about the crossover between

the two parents. So, the same set of parents which we chose for crossover same set of solutions are shown with the line.

So, for example, if I am showing this particular line this means that there is a crossover between this solution as well as this solution. So, this these lines is these lines are telling us the crossover between the two solutions, after performing the bit after performing the single point crossover the brown dots which you can see here these are the solutions which are generated. So, after that these solutions will be used for performing the mutation. So, we will take these brown points and perform the mutation.

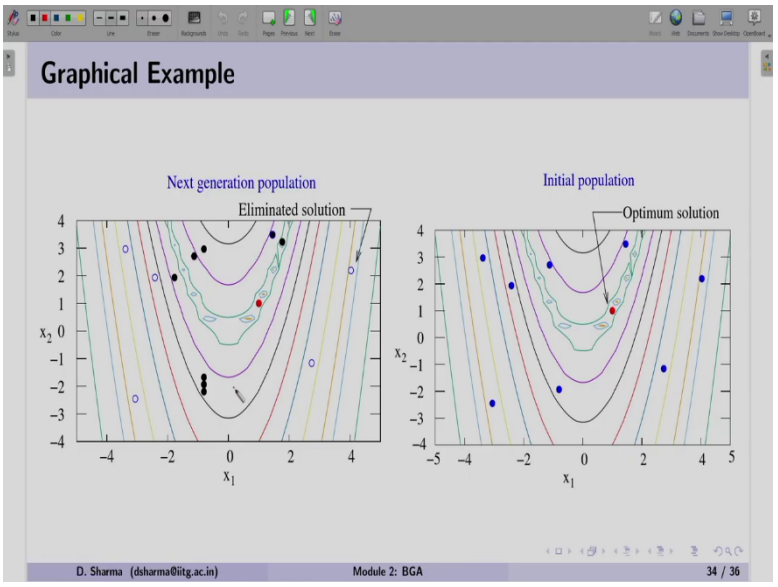
(Refer Slide Time: 66:44)



Now, in the mutation after performing the bitwise mutation, these are the solutions we got it, now these solutions you can see that they have since the binary string are changing. So, their x 1 and x 2 values also change in this range of x 1 and x 2.

So, once the offspring population is generated the next task is we have to select good solutions. So, basically we have to apply the survival stage, in that case we will combine the parent population and the offspring population. So, both the populations are combined you can see in different color codes that the blue solutions are represented parent population and the cyan color which has been used for other solution that represent offspring population.

(Refer Slide Time: 67:38)

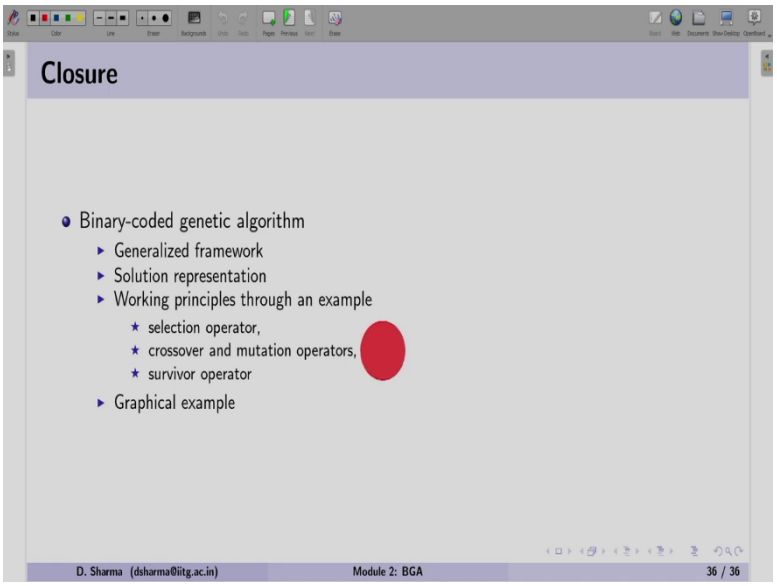


Now, you remember that in the survivor we sort the solutions in an ascending order and choose the solution, you will realize that these are the black dot solutions selected after the survival stage. Now, the solutions which are not filled with the color these are the solution that we have eliminated from the parent population plus the offspring population.

So, if we compare that in one generation how this solutions have been improved. So, this is the initial population we started all these solutions are randomly distributed in x 1 and x 2 plane here, but in gen in one generation you can see that the these black dots are already improving their fitness value and in if we follow the same steps of binary coded GA our algorithm will converge to the red dot which is the optimam solution for the Rosenbrock function.

So, we can expect binary coded GA to perform better and give us the optimum solution in few more generations. So, with this I am coming to the closure of the session in this particular session.

(Refer Slide Time: 69:01)



We have gone through the binary coded GA. So, we understood this binary coded GA with the help of the generalized framework. So, the steps mentioned in the generalized framework we followed one by one and we showed the hand calculation for one iteration.

So, in this binary coded GA our first task was the solution representation used using the binary string. So, in the current example we used 10 bit 10 the string length of 10 was used to represent the x 1 and x 2. To explain the working principle of binary coded GA we have gone through the selection operator the purpose and the working of binary tournament selection operator.

Thereafter we performed the variation operators on the mating pool and that variation operator includes crossover and mutation operator. So, after performing that, we generated the offspring population. Finally, we combined the parent population with the offspring population at the survival stage and then selected the best solution.

And in all the process we keep our population size 8 and the same example we have shown graphically to understand how these solutions which are distributed randomly initially, will be moving towards the optimum solution. With this introduction on binary coded GA I am concluding this session.

Thank you very much.