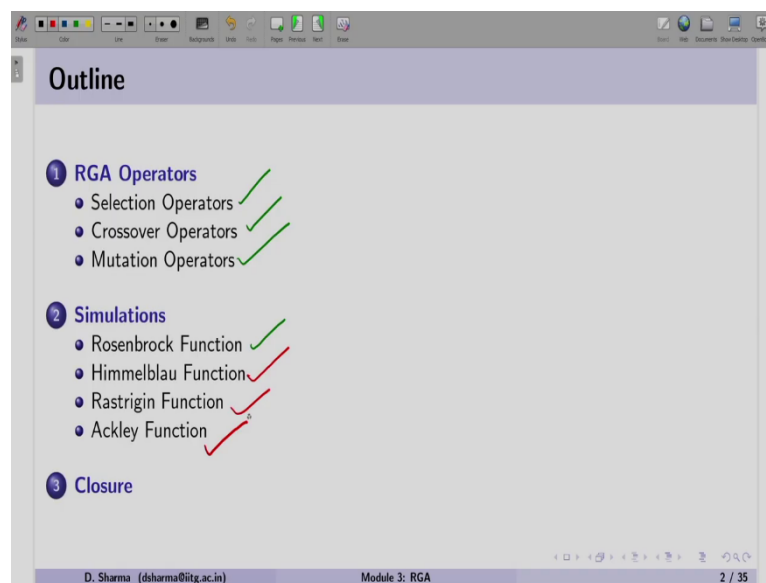


Evolutionary Computation for Single and Multi-Objective Optimization
Dr. Deepak Sharma
Department of Mechanical Engineering
Indian Institute of Technology, Guwahati

Module – 03
Lecture – 06
Operators and Simulations of Real-Coded Genetic Algorithm

Welcome to the session on Operators and Simulation of Real-Coded Genetic Algorithm.

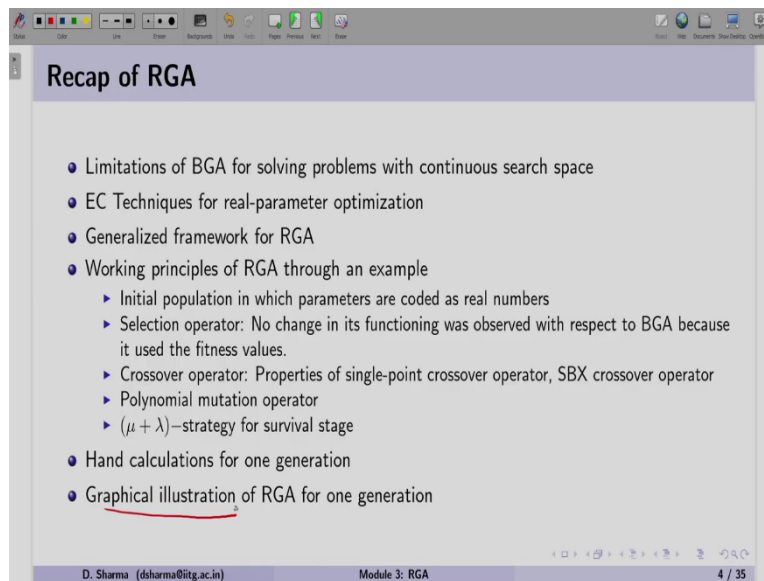
(Refer Slide Time: 00:41)



In this session, we will discuss various operators that will be required for RGA. So, here RGA stands for the real coded genetic algorithm. We will first discuss about the selection operators, thereafter the crossover and mutation operators.

The simulations of RGA will be shown using four functions that are Rosenbrock function, Himmelbau functions, Rastrigin function and Ackley function and thereafter, we will conclude this session.

(Refer Slide Time: 01:17)



So, before we start our different kinds of operators that are available for RGA, let us have a recap of the previous session. In the previous session, we found that when we are using binary coded GA for continuous search space there are certain limitations. Those limitations are corresponding to the precision or sometimes the hamming cliff problem. So, and moreover the binary coded GA makes the continuous search space into discrete.

Looking at those limitations we targeted EC techniques that can be used for real parameter optimization. So, in that category we have gone through RGA, evolutionary strategies, particles from optimization, differential evolution and etcetera. So, all these all these EC techniques are used for solving real parameter optimization problems.

We understood RGA using the generalized framework in which various operators and the processing of RGA we understand through this generalized framework. To understand the working principle of RGA we started we have taken an example of a Rosenbrock function, we want to minimize the function and for this particular Rosenbrock function we know where is the optimum.

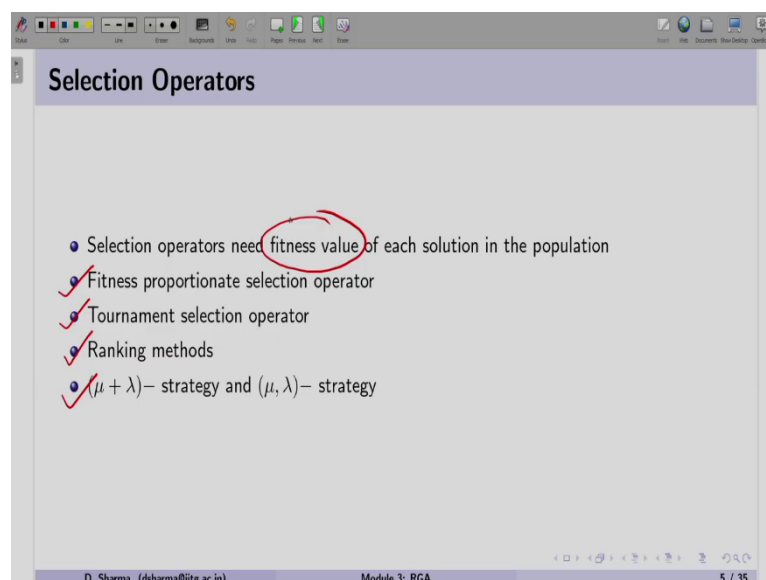
So, we started with the initial population. So, here the important part was that we have to we have to code the variable as a real number. Thereafter, we use the selection operator which is the binary tournament selection operator; after going through that we found that we do not have to change or we do not need any new selection operator for RGA because these selection operator work on the fitness. So, therefore, the functioning is not changing since we have already calculated the fitness of each solution.

Thereafter we discussed about the crossover operator. So, till binary coded genetic algorithm we have to deal with the binary strains. As soon as we work with the real numbers so we need to change the crossover, so that we can use it. In this particular section, we have gone through the various properties of single point crossover operator such as the averaging property and there was a spread factor beta.

So, those two properties were taken into the consideration and the authors came up with the operators called SBX crossover operator. Therefore, there is a polynomial mutation we talked about. In this polynomial operator we found that the probability distribution function was linear. Similarly, in SBX as well the probability distribution function was non-linear. That non-linear distribution allows to create solutions closer to the parent solutions.

And, finally, we use mu plus lambda strategy as a survival stage as an example there. All these processes as you can see all these operators and evaluations we have shown with the help of hand calculation. So, we show every calculation for a one generation and the same calculation we also showed with the help of graphical illustration. So, this graphical illustration helps us to know or understand how these solutions will be moving towards the optimum solution. With this recap, let us move to the operators now.

(Refer Slide Time: 05:17)



We start with the selection operator. As we know selection operator need the fitness value of each solution to select good or above average solutions. We have various methods available with us. For example, we can use fitness proportionate selection, tournament selection or the

ranking method; and, for combining the parent population and offspring population we use mu plus lambda strategies.

So, all these operators which we use either before variation operator or after a variation operator everyone needs the fitness value and this fitness is these this fitness value is calculated earlier. So, we found that there is no change in those selection operators. So, the selection operators which we have understood while learning binary coded GA or all of those operators are valid here.

(Refer Slide Time: 06:23)

Linear Crossover Operator

- Crossover operators for real parameters are also called as blending operators.

Linear crossover operator

One of the earliest implementations by Wright (1991):

| | |
|-------------------------------|---|
| Parent 1: $p_1 = x_i^{(1,t)}$ | Offspring 1: $0.5(x_i^{(1,t)} + x_i^{(2,t)})$ ✓ |
| Parent 2: $p_2 = x_i^{(2,t)}$ | Offspring 2: $(1.5x_i^{(1,t)} - 0.5x_i^{(2,t)})$ ✓ |
| | Offspring 3: $(-0.5x_i^{(1,t)} + 1.5x_i^{(2,t)})$ ✓ |

Diagram illustrating the linear crossover operator on a number line. The line has points $x_i^{(L)}$, p_1 , p_2 , and $x_i^{(U)}$. Offspring 1 is located between p_1 and p_2 . Offspring 2 is located to the left of p_1 . Offspring 3 is located to the right of p_2 . Red arrows indicate the distances from p_1 and p_2 to the offspring points.

Here, $x_i^{(1,t)}$ represents i -th decision variable of the randomly chosen first parent in the t -th generation.

D. Sharma (dsharma@itg.ac.in) Module 3: RGA 6 / 35

Now, let us come to the selection to the crossover operators. Now, crossover operators here these crossover operator for real parameters sometimes also referred as a blending operators. So, let us start with the first crossover operator, that is, the linear crossover operator as you can see here. This is one of the earliest implementation shown by the by Wright in 1991. So, for our simplicity we are considering p 1 as our parent 1, p 2 as a parent 2.

$$p_1 = x_i^{(1,t)}$$

$$p_2 = x_i^{(2,t)}$$

$$\text{Offspring 1: } (0.5(x_i^{(1,t)} + x_i^{(2,t)}))$$

$$\text{Offspring 2:} \quad (1.5x_i^{(1,t)} - 0.5x_i^{(2,t)})$$

$$\text{Offspring 3:} \quad (-0.5x_i^{(1,t)} + 1.5x_i^{(2,t)})$$

Now, in this case p 1 will be equals to the i-th decision variable for the solution selected in the t-th iteration. Similarly, this is the i-th variable for solution 2 and t-th iteration. So, before we move, we know that to perform crossover operator we have to select two or more solutions randomly from the population. So, in this case we selected two solutions as it is given here as 1 and 2 randomly and we are representing these two solution as p 1 and p 2.

Now, here in the figure you can see that when you have p 1 and a p 2, this crossover operator generates three offspring solution. So, the off spring solution 1 is actually at the middle of p 1 and p 2. So, we can say it is p 1 plus p 2 divided by 2. Now, if we measure this particular distance and say this is delta with a similar distance we created another solution.

So, you can see on the left and side of a p 1 offspring 2 is created on the right hand side of p 2 another offspring is created. Looking at the formula here, so, offspring 1 which is at the middle of p 1 and p 2, offspring 2 is on the left hand side of p 1 and offspring 3 is on the right hand side of the p 2. So, as the representation I explained you earlier this is for the i-th decision variable and in the t-th iteration.

(Refer Slide Time: 08:41)

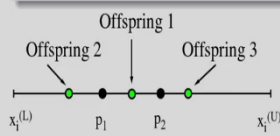
Linear Crossover Operator

- Crossover operators for real parameters are also called as blending operators.

Linear crossover operator

One of the earliest implementations by Wright (1991):

| | | |
|-------------------------------|---|--|
| Parent 1: $p_1 = x_i^{(1,t)}$ | Offspring 1: $0.5(x_i^{(1,t)} + x_i^{(2,t)})$ — ① | |
| Parent 2: $p_2 = x_i^{(2,t)}$ | Offspring 2: $(1.5x_i^{(1,t)} - 0.5x_i^{(2,t)})$ — ② | |
| | Offspring 3: $(-0.5x_i^{(1,t)} + 1.5x_i^{(2,t)})$ — ③ | |



Here, $x_i^{(1,t)}$ represents i -th decision variable of the randomly chosen first parent in the t -th generation.

- We choose two best offspring solutions among three.

D. Sharma (dsharma@nitg.ac.in) Module 3: RGA 6 / 35

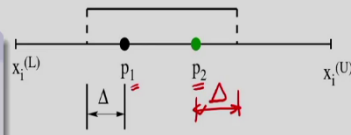
Now, in this case as you have realized that two parent solution are used to create three offspring solutions; as I am writing as 1, 2 and 3. Now, what we do here is, from these three solutions we choose two best solutions among the three solutions. So, this is the way linear crossover operator works. It is one of the simplest implementation here.

(Refer Slide Time: 09:13)

Blend Crossover Operator

Blend Crossover and Its Variants

- Blend crossover (BLX- α)
- The assumption is $p_1 = x_i^{(1,t)} < x_i^{(2,t)} = p_2$ for two parents.



- Here, $\Delta = \alpha(p_2 - p_1) = \alpha(x_i^{(2,t)} - x_i^{(1,t)})$

D. Sharma (dsharma@nitg.ac.in) Module 3: RGA 7 / 35

The another crossover operator which we have is called blend crossover operator. So, this blend crossover operator is also known as BLX alpha. Now, in this crossover operator we assume that p_1 is smaller than p_2 .

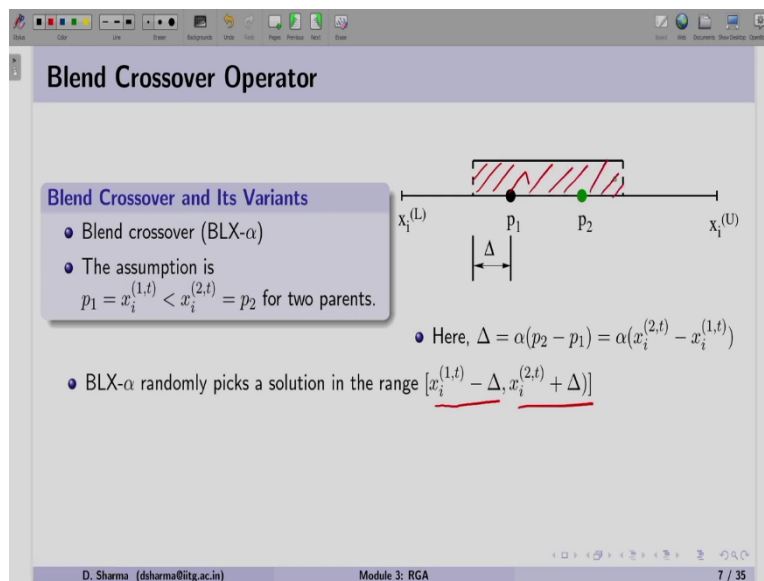
$$p_1 = x_i^{(1,t)} < x_i^{(2,t)} = p_2$$

$$\Delta = \alpha (p_2 - p_1) = \alpha (x_i^{(2,t)} - x_i^{(1,t)})$$

So, the same representation we are using for parent 1 and a parent 2, but with a condition that p 1 is a smaller than p 2. Now, in this crossover operator we can see in this figure we have p 1, p 2 and then we have a delta.

Similarly, we have the same quantity of a delta on the right hand side of p 2. So, the in this case thus delta is decided as alpha times of p 2 minus p 1 and as you know p 2 and a p 1 we are writing in terms of x i 2, t minus x i 1, t.

(Refer Slide Time: 10:11)



Now, looking at this particular figure we can see that the solution can be generated from x i 1, t minus delta to x i 2, t plus delta.

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

So, basically if we look at this region in the figure, in this particular region the solution is created and that is created randomly in this region.

(Refer Slide Time: 10:37)

Blend Crossover Operator

Blend Crossover and Its Variants

- Blend crossover (BLX- α)
- The assumption is $p_1 = x_i^{(1,t)} < x_i^{(2,t)} = p_2$ for two parents.
- Here, $\Delta = \alpha(p_2 - p_1) = \alpha(x_i^{(2,t)} - x_i^{(1,t)})$
- BLX- α randomly picks a solution in the range $[x_i^{(1,t)} - \Delta, x_i^{(2,t)} + \Delta]$
- Offspring: $x_i^{(1,t+1)} = (1 - \gamma_i)x_i^{(1,t)} + \gamma_i x_i^{(2,t)}$, where $\gamma_i = (1 + 2\alpha)u_i - \alpha$ *user-defined*
- u_i is a random number between 0 and 1.

D. Sharma (dsharma@itg.ac.in) Module 3: RGA 7 / 35

So, the offspring is created with the help of another factor called gamma and this gamma is calculated with the help of alpha value and the random number u i. So, we using these two values we can found the value of a gamma. Now, we know that u i is a random number. So, this random number will be generated by the computer. Now, in this case this alpha value this is a user defined value and we have to set this value here.

$$x_i^{(1,t+1)} = (1 - \gamma_i)x_i^{(1,t)} + \gamma_i x_i^{(2,t)}, \text{ where } \gamma_i = (1 + 2\alpha)u_i - \alpha$$

Now, are looking at the equation of the offspring you can make it out that the two solutions we are using a scaling function such as 1 minus gamma times of parent 1 plus gamma times of parent 2. So, this is scaling we use it to create an offspring. So, from this particular crossover operator we found that two cross two parent solutions are creating one offspring here.

(Refer Slide Time: 11:45)

Blend Crossover Operator

Blend Crossover and Its Variants

- Blend crossover (BLX- α)
- The assumption is $p_1 = x_i^{(1,t)} < x_i^{(2,t)} = p_2$ for two parents.

- Here, $\Delta = \alpha(p_2 - p_1) = \alpha(x_i^{(2,t)} - x_i^{(1,t)})$
- BLX- α randomly picks a solution in the range $[x_i^{(1,t)} - \Delta, x_i^{(2,t)} + \Delta]$
- Offspring: $x_i^{(1,t+1)} = (1 - \gamma_i)x_i^{(1,t)} + \gamma_i x_i^{(2,t)}$, where $\gamma_i = (1 + 2\alpha)u_i - \alpha$
- u_i is a random number between 0 and 1.
- It can be seen from the figure that γ_i is uniformly distributed for a fixed value of α .

D. Sharma (dsharma@itg.ac.in) Module 3: RGA 7 / 35

Now, once since we have included this gamma into the creation of the offspring solution, so, if we fixed this value of alpha throughout of throughout the iterations, we can find that the gamma i is uniformly distributed for a fixed value of alpha. So, this distribution as I showed you earlier so, this is the distribution where a random a solution will be generated using BLX alpha.

(Refer Slide Time: 12:17)

Blend Crossover Operators

BLX- α

- The value of $\alpha = 0.5$ is found to be performing better than other α values (Deb, 2001) over many test problems.
- Let us re-write the equation for BLX- α

$$x_i^{(1,t+1)} = (1 - \gamma_i)x_i^{(1,t)} + \gamma_i x_i^{(2,t)}$$

$$(x_i^{(1,t+1)} - x_i^{(1,t)}) = \gamma_i (x_i^{(2,t)} - x_i^{(1,t)})$$

$p_2 - p_1$

- It suggests that the location of offspring depends on the difference in parent solutions.

D. Sharma (dsharma@itg.ac.in) Module 3: RGA 8 / 35

$$x_i^{(1,t+1)} = (1 - \gamma_i)x_i^{(1,t)} + \gamma_i x_i^{(2,t)}$$

$$(x_i^{(1,t+1)} - x_i^{(1,t)}) = \gamma_i (x_i^{(2,t)} - x_i^{(1,t)})$$

Now, the question is what could be the value of alpha because I can use 1, 0.5, 3 or 7 anything value I can use it. So, in this particular study the author found that if we use alpha equals to 0.5, this particular value is working good for many test problems. So, that is why a guideline or a thumb rule is developed that we can take alpha value. However, we can change this value as per our problem and simulation.

Now, let us see what is the characteristic of this operator. So, the equation the original equation in the previous slide is written here. So, let us move some quantity on the right and side, keep some quantity on the left hand side. So, this particular equation is modified as what you can see here.

Now, the important point in this particular equation which we have rewritten, so, the in the square in the rectangular box you can realize it is similar to the difference between p 2 minus p 1; meaning that the difference between these two parent actually decide what could be the new solutions.

(Refer Slide Time: 13:43)

Blend Crossover Operators

BLX- α

- The value of $\alpha = 0.5$ is found to be performing better than other α values (Deb, 2001) over many test problems.
- Let us re-write the equation for BLX- α

$$x_i^{(1,t+1)} = (1 - \gamma_i)x_i^{(1,t)} + \gamma_i x_i^{(2,t)}$$

$$(x_i^{(1,t+1)} - x_i^{(1,t)}) = \gamma_i (x_i^{(2,t)} - x_i^{(1,t)})$$
 - It suggests that the location of offspring depends on the difference in parent solutions.
- The above condition signifies an adaptive search
 - In initial generations when random population is distributed over the entire search space, the difference is large and an offspring population with a large diversity is expected.
 - When population tends to converge in some region (later generations), difference is small thereby causing focused search.

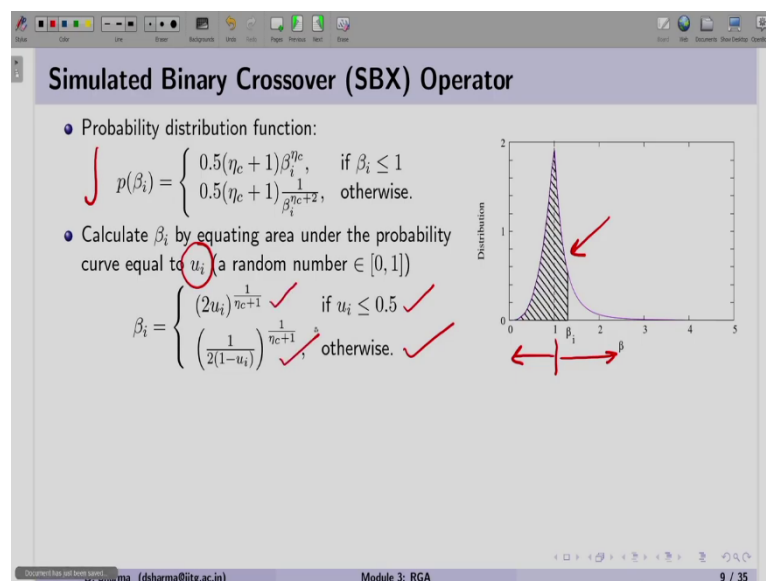
D. Sharma (dsharma@iitg.ac.in) Module 3: RGA 8 / 35

So, therefore, it says that the if we are going to use BLX alpha, the search is adaptive in nature. How? If supposed we are starting so, at the beginning when the solutions are randomly generated in the plane of variable, in that case we pick two solutions randomly. So, generally these solutions are little far from each other. So, when they are far this particular difference will be large.

Since this will be large so, the offspring solution that will be created will be little far from their parents. So, this property will help us to explore the search space initially and that is what we desire from EC techniques. But, in the later generations what you will see that the solutions will be close to each other.

When they are close to each other then the same difference will become small and therefore, the offspring will also be created close to the parent. So, in this case since the difference is small we can say the search is focused. So, this adaptive property is useful in all EC techniques if we can implement it.

(Refer Slide Time: 15:03)



Simulated binary crossover is the third crossover operator. This crossover operator we discussed in detail. This operator uses the non linear distribution probability distribution. You can see with respect to the value of beta i there are two functions which are used.

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

$$\beta_i = \{(2u_i)^{\frac{1}{(\eta_c+1)}}, \quad \text{if } u_i \leq 0.5 \left(\frac{1}{(2(1-u_i))} \right)^{\frac{1}{(\eta_c+1)}}, \text{ otherwise}$$

So, if you look at this particular figure so, this is the value 1. So, on the left hand side there is a another function, another side there is a another probability distribution function and these two probability distribution functions are non-linear in nature. The objective is the offspring should be created close to the parent solution.

And, how we how we can calculate the value of a beta i? As we have discussed earlier that we will be integrating this probability and this integration will give me the value of beta i. So, integrating this means the area under the curve as you can see here so, this area under the curve we can correlate with the random number given here as a u i.

So, after simplification we can get a value of a beta i based on random number. So, if random number is smaller than or equal to 0.5, we will be using the first equation otherwise we will be using the second equation.

(Refer Slide Time: 16:33)

Simulated Binary Crossover (SBX) Operator

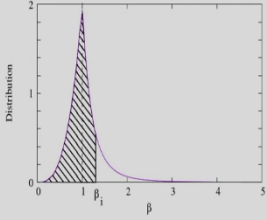
- Probability distribution function:

$$p(\beta_i) = \begin{cases} 0.5(\eta_c + 1)\beta_i^{\eta_c}, & \text{if } \beta_i \leq 1 \\ 0.5(\eta_c + 1)\frac{1}{\beta_i^{\eta_c+2}}, & \text{otherwise.} \end{cases}$$
- Calculate β_i by equating area under the probability curve equal to u_i (a random number $\in [0, 1]$)

$$\beta_i = \begin{cases} (2u_i)^{\frac{1}{\eta_c+1}} & \text{if } u_i \leq 0.5 \\ \left(\frac{1}{2(1-u_i)}\right)^{\frac{1}{\eta_c+1}}, & \text{otherwise.} \end{cases}$$
- Offspring are:

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

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



$x_i^{(1,t)} < x_i^{(2,t)}$ $p_1 < p_2$

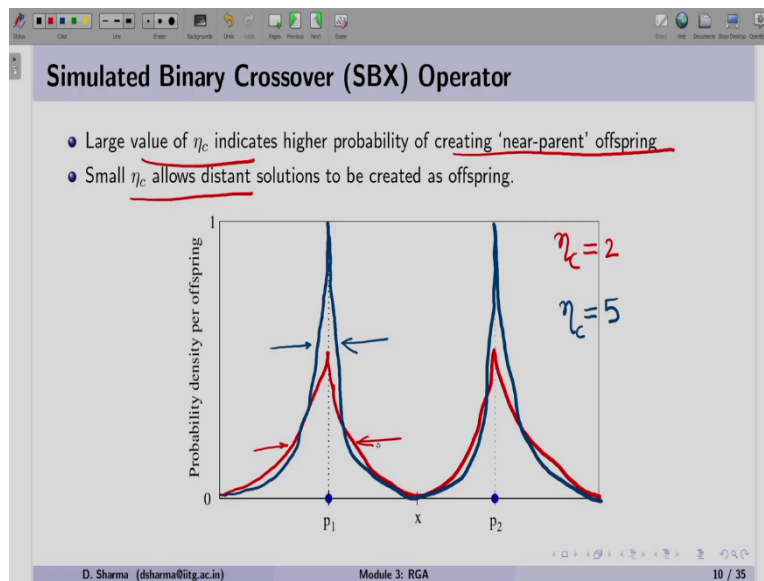
D. Sharma (dsharma@itg.ac.in) Module 3: RGA 9 / 35

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

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

Using this beta i value we used this we use this beta i value to create offspring. So, this equation is familiar to us as we can see these two quantities tells us about the average property and then we have a beta which we calculated using the probability distribution function and the difference between the two solution or two parent solutions. Here it is important to note that all this analysis we are doing by assuming that p 1 is smaller than p 2.

(Refer Slide Time: 17:13)



Now, let us see what how this SBX operator behaves. So, in this case let me draw the curve for two eta c value. So, this particular different eta c value we have gone through in the previous session. Now, here we are we have two solutions. So, using these two solution if we take different eta c value so, what will be the curve? So, let me take eta c is equals to 2.

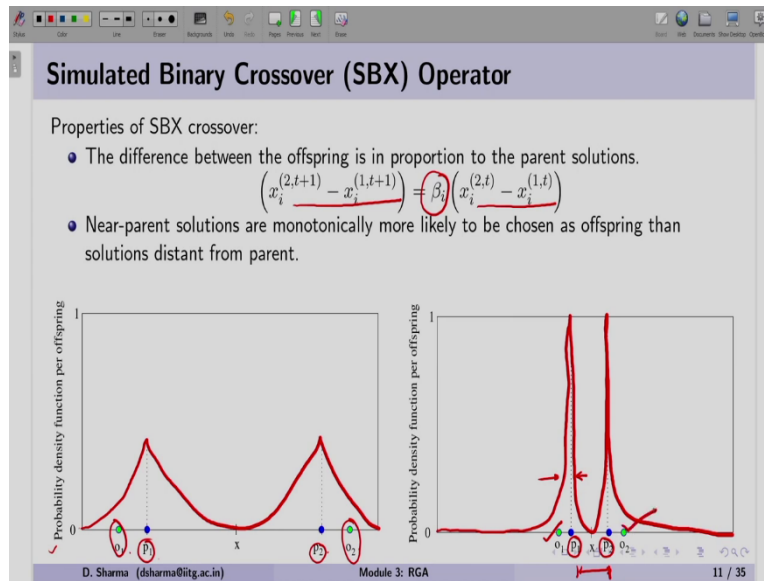
In this particular case this eta c for two value it will be similar to this curve and then it will move here and then come back here. So, this is for eta c equals to 2. Now, suppose we will take a larger value say eta c is equals to 5. So, what will happen? That as and when this eta c value is large so, you can see it is approaching the y-axis in this way than the mean then again for another solution it is approaching like this and then finally, coming here.

So, what we can see here is for the two different values of eta c and eta c equals to 2 and 5, we get two different curves. It is only because when we are integrating the probability the area under the curved maximum could be 1.

Now, as you can see that if we are using the eta c large eta c value that indicates that there is a higher probability of creating new a near apparent solution. It is because of the nature of the curve. So, when eta c is equals to 5 as you can see that the width of the curve as is reduced however, its peak has it is increased with respect to eta c equals to 2.

So, in this case when an offspring will be generated so, the solution will be close to the parent solution. In another case when eta c is small then we have a curve which is little wider than the eta c equals to 5. So, for a smaller value of eta c, the offspring solutions can be created little far from the parent solution.

(Refer Slide Time: 20:01)



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

Now, let us come to the property another property of SBX operator here. Now, the what we have done here is we have taken the difference between the two parent solution two offspring solutions that comes out to be the difference between the parent solution and this is multiplied by beta. So, in this case what we can see that the difference between the offspring is proportional to the difference between the parent solutions.

So, this is similar to the property as we have learned with the b BLX alpha. So, in this case our SBX operator also supports adaptive search, in the in that case in the initial generation when solutions are little far so, the offspring solutions will be generated little far. Similarly, when after few generations when solutions are moving towards the optimum solution so, they are close enough and this difference suggests that the offspring solution will be generated close to their parent solutions.

Now, let us take a two cases here; in one case as you can see in this plot y-axis the probability density and x-axis has parent 1, parent 2 similarly offspring 1 and offspring 2. So, since in this particular figure you can see that p 1 and p 2 they are little far so, what could be the distribution? So, if I redraw this, so, this is coming out to be this.

So, in this particular figure plot you can see that when the solutions are far, so they are so these two probability distribution for p_1 and p_2 since they are far so, the offspring solutions will be created far because the difference between p_2 and p_1 is large. Similarly, if we take the another case in the right hand side figure, so, p_1 and p_2 are quite close to each other. Let us take the another case where p_1 and p_2 are close to each other which you can see on the right hand side.

So, here so, the distribution will be similar to like this as you can see here. So, you can see there are peaks and then the curve is like this. So, in this case what we will see that when p_1 and p_2 are close to each other the peak of the curve has been raged. However, the width of the curve is reduced as compared to the figure on the left hand side. So, this will allow to create our offspring 1 and offspring 2 closer to the parent solution.

(Refer Slide Time: 23:05)

SBX Crossover Operator

- The preceding operator creates solution in the range of $[-\infty, \infty]$.
- However, we need to generate solution within the range that is $x_i^{(L)} \leq x_i \leq x_i^{(U)}$.
- Find the cumulative probability

$$P'_1 = \int_0^{\beta^{(L)}} P(\beta) d\beta,$$

$$P'_2 = \int_0^{\beta^{(U)}} P(\beta) d\beta,$$
- Here, $\beta^{(L)}$ and $\beta^{(U)}$ are the spread factors for the lower and upper bounds of variable x_i
- These spread factors are calculated as

$$\beta^{(L)} = \frac{p_1 + p_2 - 2x_i^{(L)}}{|p_2 - p_1|}$$

$$\beta^{(U)} = \frac{2x_i^{(U)} - p_1 - p_2}{|p_2 - p_1|}$$
- We can get two spread factors β'_1 and β'_2 using modified probability distributions $P(\beta)/P'_1$ and $P(\beta)/P'_2$, respectively.

D. Sharma (dsharma@itg.ac.in) Module 3: RGA 12 / 35

$$P'_1 = \int_0^{\beta^{(L)}} P(\beta) d\beta$$

$$P'_2 = \int_0^{\beta^{(U)}} P(\beta) d\beta$$

$$\beta^{(L)} = \frac{(p_1 + p_2 - 2x_i^{(L)})}{|p_2 - p_1|}$$

$$\beta^{(U)} = \frac{(2x_i^{(U)} - p_1 - p_2)}{|p_2 - p_1|}$$

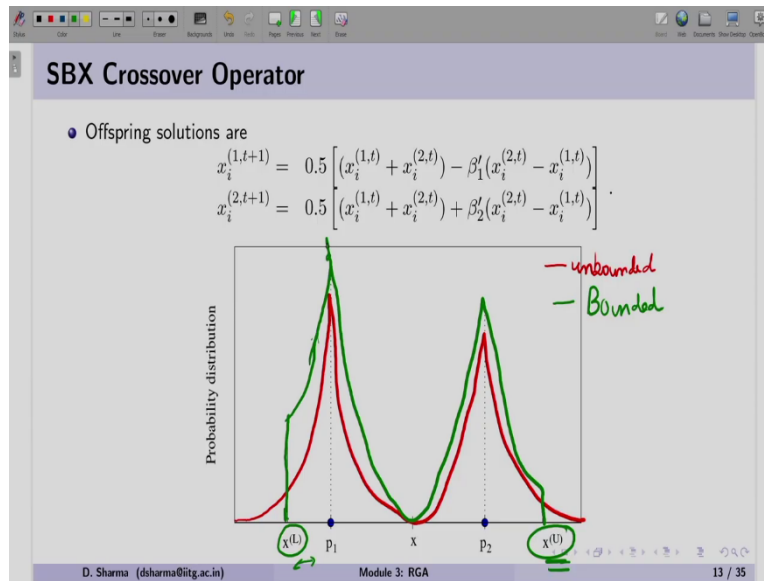
Now, the crossover operator that is SBX operator which we have discussed what we found that it can generate any solution between minus infinite to plus infinite. However, if we want what we want is that they since the variable bounds are given as between the lower and the upper bound we want that this crossover operator should generate solution within the bound.

So, in this case a small change has been made in this crossover operator, in which the cumulative proper probabilities are calculated with respect to beta U and beta L. What are those quantities? So, these quantities are corresponding to the upper and lower bound of the variable. So, you remember that we generally integrate from 0 to beta i, but in this case we are integrating from 0 to beta upper and beta lower, so that we can generate a solution within the range of x i.

Now, this beta U, beta L and a beta U we can find using the formula given here which is the summation of the two parents minus 2x i lower for beta lower. And, when we talk about the upper so, we have 2x i upper that is the upper limit on this variable and the difference between p 1 minus and p minus p 2 in this case the denominator remains the same that is the difference between p 2 minus p 1.

Now, using this modified probability we are going to calculate two quantities which are beta prime 1, beta prime 2. So, similar to area under the curve and we are equating with a random number the same approach is used, but with while respecting the lower and upper bound of the variable. So, our modified probability distribution as you can see here that will change.

(Refer Slide Time: 25:23)



When we integrate it we can get two values of beta 1 and beta 2. So, these two beta 1 and beta 2 values we are writing here. In this case the same formula as you can see this is the summation of the two parents and we have taken 0.5 outside. So, this is nothing, but the average.

$$x_i^{(1,t+1)} = 0.5 \left[(x_i^{(1,t)} + x_i^{(2,t)}) - \beta_1' (x_i^{(2,t)} - x_i^{(1,t)}) \right]$$

$$x_i^{(2,t+1)} = 0.5 \left[(x_i^{(1,t)} + x_i^{(2,t)}) + \beta_2' (x_i^{(2,t)} - x_i^{(1,t)}) \right]$$

And, then beta 1 prime we have multiplied with a difference, similarly for the offspring number 2 we have average property then plus so, here you have to make sure that it is beta 2 prime. So, another beta for the upper limit and this is the difference between the 2 parts. So, in this way we can generate 2 offspring solutions that will respect the bounds for the using the SBX crossover operator.

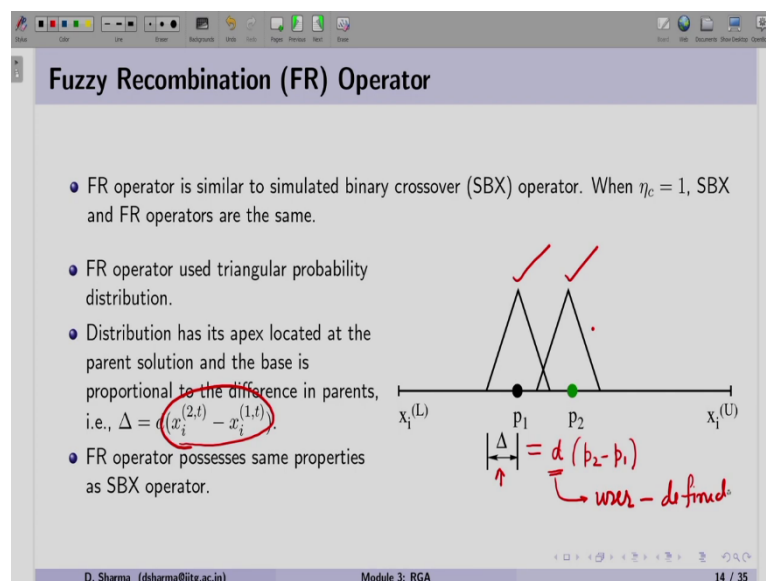
So, let us take two things now. First let me take the unbounded. So, unbounded means that the SBX crossover operator can generate a solution from the limit minus infinite to plus infinite. So, in this case you can see that this particular curve which is coming from the infinite going here and then this is coming here and then going here and then going towards the infinite.

So, this is the probability distribution you can expect when the solution is generated from minus infinite to plus infinite. Now, let us take the bounded. So, bounded means that the SBX crossover operator will generate two solutions between the bound.

Now, in this figure you can see the lower bound is given as well as the upper bound is given. So, how this particular solution will behave? So, here this will be, so, let me start from the bottom and then it will go it will follow the similar trend here. It will come at the middle now, it will go little higher and then and then it will finish it off.

So, this kind of a distribution for the bounded region when SBX is creating a solution between lower and an upper bound you can see as soon as we are fixing the lower and the upper bound here so, the peak is actually it has gone little up as compared to the bounded one. Why? Because the area under this particular curve should be one, now we have restricted this probability distribution from lower to upper so, the peak has been gone little high.

(Refer Slide Time: 28:15)



So, as of now we have gone through this SBX crossover operator where we can generate solution from minus infinite to plus infinite as well as if we want to include the bounds. So, there are some changes needed to calculate beta 1 prime and beta 2 prime using the random number.

$$\Delta = d(x_i^{(2,t)} - x_i^{(1,t)})$$

Now, we are moving to the another crossover operator which is known as fuzzy recombination operator. This operator as you can see here it is similar to the binary crossover operator which as we say it is called SBX operator. So, when we fix eta c value is equals to 1, then both the operators will behave similar. So, what is the difference here?

So, in SBX crossover operator we use non-linear probability distribution on the either side of the beta. So, for beta equals to 1, we had one function and when beta is greater than 1 we had another function. So, instead of that as you can see from this figure they in these FR operator triangular shape probability distribution function is used and, this it is working is very similar to the SBX crossover operator.

Now, in the figure you can see there is a term called delta. This delta term has a significance because you can see that this delta depends on the difference between the two parent. So, it means that FR operator is also has this adaptive search capability in which when the solutions are far in the early generation, then the offspring solutions will be created little far. When parent solutions are close to each other when they are moving towards the optima then offspring solution will also be generated close to the minima.

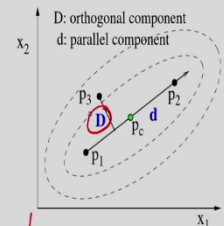
Now, using this d we have a delta which is also has let me write here as $d \cdot p_2 - p_1$. Now, this is the d factor which is a user defined factor. So, this we have to decide before we execute our function our algorithm.

So, in similar to SBX crossover operator we have eta c we have a BLX alpha where we have to set the value of alpha. This operator has a one user defined parameter called d. Now, all the properties what we have discussed earlier with the SBX operator all these properties are hold by the FR operator.

(Refer Slide Time: 31:05)

Unimodal Normally Distributed Crossover (UNDX) Operator

- It is proposed by Ono and Kobayashi, 1997
- In this crossover operator, three or more parent solutions are used to create two or more offspring solutions.
- Offspring are created from an ellipsoidal probability distribution with one axis formed along the line joining two of three solutions.



- Two parent solutions (p_1, p_2) are picked randomly and the parallel component is calculated as $d = p_2 - p_1$.
- The center of these parent solutions is $p_c = (p_1 + p_2)/2$.
- The third parent solution (p_3) is picked randomly and an orthogonal component (D) is found.

Offspring solution is calculated as $x_o = p_c + \zeta d + \sum_{i=1}^{N-1} \eta_i p_i D$, where ζ, η_i are random numbers and p_i is the orthogonal bases that span the subspace perpendicular to d .

D. Sharma (dsharma@nitg.ac.in) Module 3: RGA 15 / 35

Now, let us move to the new crossover operator which is called unimodal normally distributed crossover which we in short say UNDX operator. This particular operator was proposed by Ono and Kobayashi in 1997. Now, as you can see this particular crossover operator, need three or more parent solutions that will create two or more offspring solution; so we will see how.

Now, offspring solutions are created using the ellipsoidal probability distribution. So, one thing you can see here that the crossover operators are using some kind of probability distribution functions to generate offspring solutions. So, this ellipsoidal probability distribution function is used with one axis formed along the line joining the two of the three solutions. So, here we are assuming that we are working in a two variable for a two variable problem.

Now, let us see how it works. So, in this particular crossover operator we take two solutions as p_1 and p_2 and we picked it randomly. Now, when we pick it we find a component called d which is nothing, but p_2 minus p_1 . So, this particular component d as you can see in the figure this is referred as the parallel component.

Now, using p_1 and a p_2 we find the value of p_c . Now, this p_c is preferred as the centre of the two parents which is nothing, but p_1 plus p_2 divided by 2. So, as soon as we find this centre here, we pick the third solution which is p_3 and that is again randomly and this p_3 as

you can see that p_3 is lines in this particular figure here and the capital D we find and this will represent the orthogonal component.

So, using these parallel and orthogonal component, this crossover generates an offspring. So, you can see that x_0 represent it is x_{naught} , o stands for the offspring population or offspring solution, we have p_c which is the centroid, then we have a ζd . So, as we know d is our parallel component and then we have summations with respect to ζ and component of p_i 's and of D is our orthogonal component.

So, as you can see that ζ and η are the random numbers and p_i is the orthogonal basis that span the subspace perpendicular to the d . So, we are talking about all the orthogonal basis corresponding to the d which is deeper perpendicular or orthogonal component. So, in this way this crossover operator works and generates new solutions for EC techniques.

(Refer Slide Time: 34:25)

Crossover Properties

Case-1

Case-2

Desired properties expected from crossover operators

- The population mean should not change.
 - ▶ We know that most crossover operators do not use any fitness value. Therefore, a crossover operator do not steer the search in any particular direction.
 - ▶ We need a crossover operator that can generate offspring population such that the mean of offspring population should be the same as that of the parent population.
 - ▶ However, the variance can change.

D. Sharma (dsharma@itg.ac.in) Module 3: RGA 16 / 35

Now, as of now we have gone through various crossover operators, in the literature there may be few more crossover operators, but when we design a crossover operator we look for certain properties. So, one of the desired property is that the mean of the population should not change.

What we mean by that? That we know that; most crossover operators do not use any fitness value. Since they are not using any fitness value, meaning that, the crossover operator will not be performing any search in a particular direction. So, that is random in nature and this is

evident from the crossover operator whether it is a crossover operator for binary string or for a real number all of them are stochastic in nature and we perform the crossover operator.

Now, at this particular stage what we want is a crossover operator that can generate offspring population that such that the mean of the population should be the same as the parent solutions. However, the variance can change. So, let us take one example here.

So, let me take this is the mean here and we have various solutions. So, for just clarity I am taking the solutions little far, but if you try to find out the mean of these, so that it will become red this is the red point. Now, let me take another case here. Suppose we have this mean and the solutions are close to this particular solution. Now, in this case what you can realize that in both the cases mean will not change.

(Refer Slide Time: 36:49)

Crossover Properties

Desired properties expected from crossover operators

- The population mean should not change.
 - ▶ We know that most crossover operators do not use any fitness value. Therefore, a crossover operator do not steer the search in any particular direction.
 - ▶ We need a crossover operator that can generate offspring population such that the mean of offspring population should be the same as that of the parent population.
 - ▶ However, the variance can change.
- The population diversity should increase.
 - ▶ We know that selection operator reduces variance by selecting solutions using fitness values.
 - ▶ If crossover operator also reduces variance of the population, it may lead to a premature convergence.
 - ▶ Therefore, we need a crossover operator that should increase variance of the population.

D. Sharma (dsharma@nitg.ac.in) Module 3: RGA 16 / 35

So, I am assuming that both the population have the same mean. However, for this for the case 1 let me write here for this case 1 we have a larger variance, but for case 2 we have small variance. So, the crossover operator should be generated or developed in such a way that the mean should not change.

Second desired property is that the population diversity should increase. So, in every generation we performed crossover operator. Now, what we can realize here that since we are performing the selection operator. So, we are actually reducing the variance of the population because we are selecting only good and above average solution.

In this particular scenario, the crossover operator should not reduce the variance of the population it is only because that will lead to the premature convergence. It is because sometimes if there is an individual which is closer to the local optima, then this particular individual will get multiple copies and then after few iterations all solution will become same.

But, since it is close to the local optima then the whole population will converge to this a optima with which we consider as a premature convergence. So, in that case our crossover operator should not reduce the variance because that will help us to keep the diversity into our population. Therefore, as we suggested the crossover operator, that should increase the variance of the population.

(Refer Slide Time: 38:19)

Similarity in Different Crossover Operators

- Beyer and Deb (2000) conducted a study on using three types of crossover operators with their characteristics parameters.
- A flat landscape function was chosen for identical variance growth of operators.

| | SBX | BLX | FR |
|----------|-----|--------|-------|
| η_c | 2 | 0.662 | 1.095 |
| α | 3 | 0.500 | 0.707 |
| d | 5 | 0.419 | 0.433 |
| | 10 | 0.3801 | 0.226 |

- The study suggested that these crossover operators can perform similar for some characteristic parameter values.

D. Sharma (dsharma@nitg.ac.in) Module 3: RGA 17 / 35

As of now we have gone through various kinds of crossover operator for real parameters. So, in 2000 Beyer and Deb, they conducted a study where they were comparing different kind of a crossover operators and the performance of these crossover operator was tested on a flat landscape function, so that they can understand what is the what is the growth of the population. So, they are measuring the variance of the variance of the population.

In their analysis what they found is suppose because they have taken three crossover operators SBX, BLX and FR. And, we know that these are the three user defined parameters we have to fix before we move for during the simulation.

Now, on this flat landscape what they found is if they keep SBX eta c equals to BLX 0 point 0.06 and d as 1.095 the performance of the operator was similar all these operators. Similarly, when they kept eta c equals to 3 BLX is equals to 0.5 and d value 0.707, the performance of all these crossover was found to be similar.

So, these values suggest that there is a one characteristic parameter value for which the different kind of a crossover can behave similarly. So, that is a good study that help us to understand that this different kind of a crossover operator in certain situation can behave similarly.

(Refer Slide Time: 40:19)

Random Mutation Operator

- Michalewicz, 1992 proposed random mutation operator to create solution randomly from the entire search space.

$$y_i^{(1,t)} = r_i(x_i^{(U)} - x_i^{(L)}),$$

where r_i is a random number $\in [0, 1]$

- Independent of the parent solutions and equivalent to random initialization.

D. Sharma (dsharma@itg.ac.in) Module 3: RGA 18 / 35

After going through various types of crossover operator, now let us move to the mutation operator which is also of one of the kind of variation operator. So, the first of the first mutation operator is the random mutation operator. It was proposed by Michalewicz in 1992.

$$y_i^{(1,t)} = r_i(x_i^{(U)} - x_i^{(L)})$$

As you can see the new solutions created; so, y_i represents the new solution or the new value for the i -th variable that depends on r_i which is the random number and the difference between the upper bound and the lower bound.

The interesting part of this random mutation operator is that it is independent of the parent solution. So, we can say that we are this mutation operator is a random mutation operator and since it is independent of the parent solution, it can create solution anywhere in the search space.

(Refer Slide Time: 41:23)

Random Mutation Operator

- Michalewicz, 1992 proposed random mutation operator to create solution randomly from the entire search space.

$$y_i^{(1,t)} = r_i(x_i^{(U)} - x_i^{(L)}),$$

where r_i is a random number $\in [0, 1]$

- Independent of the parent solutions and equivalent to random initialization.
- A solution in the vicinity of parent solution with a uniform probability distribution (dashed-line)

$$y_i^{(1,t)} = x_i^{(1,t)} + (r_i - 0.5)\Delta_i$$

where Δ_i is a user-defined maximum perturbation allowed in the i -th decision variable.

- Care should be taken if $y_i^{(1,t)}$ takes value outside of the prescribed lower and upper limits.

D. Sharma (dsharma@itg.ac.in) Module 3: RGA 18 / 35

$$y_i^{(1,t)} = x_i^{(1,t)} + (r_i - 0.5)\Delta_i$$

To modify this particular mutation operator a small factor of a delta i is included into the formula. So, what this delta i says that? Now, let us look at the figure here. Now, this delta i says that on the either side of X i that is on the left side and the right hand side we have a uniform distribution. So, when we decide the delta i. So, this will tell me about how much perturbation. So, we call it as a maximum perturbation. So, this maximum perturbation is decided by the delta i value.

So, let us look at the equation now. This says that the solution y i created by the mutation is equals to the solution that is the parent solution and it depends on the random number and the delta i. Now, when random number is a smaller than 0.5, you know that we are talking about the left hand side of X i; when random number is greater than 0.5, then the solution will be generated on the right hand side of the X i.

Now, in this particular crossover operator as a remark we have to take care of the upper and the lower bound why because if supposed this X_i solution in the figure, if this solution is say somewhere here as you can see. So, let me write this as a p solution p.

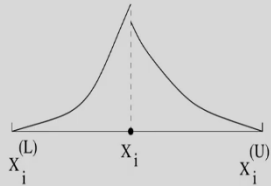
Now, if we take a delta value something like this, then it means that a solution can or the mutation operator can generate a solution which is beyond the lower bound. So, in this case what we say that suppose I consider a solution created by the mutation operator is y_i , so, if y_i is smaller than the lower bound of variable then we will say y_i is actually on the bound. So, we are keeping this solution on the bound.

Similarly, if this solution y_i is greater than the upper bound in this case we will keep this solution on the upper bound. So, such kind of simple implementation we can use it so that our solution generated by mutation should be within the bound.

(Refer Slide Time: 44:07)

Non-Uniform Mutation Operator

- Michalewicz, 1992 proposed a non-uniform mutation operator with the idea that
 - probability of creating a solution closer to the parent is more than the probability of creating one away from it.
- As generations (t) proceed, this probability of creating solutions closer to the parent gets higher and higher.



where

$$y_i^{(1,t)} = x_i^{(1,t)} + \delta_i^t$$

$$\delta_i^t = \begin{cases} (x_i^{(U)} - x_i^{(1,t)}) \cdot \left(1 - r_i^{(1-t/T)^b}\right) & \text{probability} \leq 0.5 \\ (x_i^{(1,t)} - x_i^{(L)}) \cdot \left(1 - r_i^{(1-t/T)^b}\right) & \text{otherwise} \end{cases}$$

- T is the maximum allowed generations and b is a non-negative user defined parameter.

D. Sharma (dsharma@itg.ac.in) Module 3: RGA 19 / 35

$$y_i^{(1,t)} = x_i^{(1,t)} + \delta_i^t$$

$$\delta_i^t = \left\{ \left(x_i^{(U)} - x_i^{(1,t)} \right) \left(1 - r_i^{\left(1 - \frac{t}{T} \right)^b} \right) \right. \\ \left. \left(x_i^{(1,t)} - x_i^{(L)} \right) \left(1 - r_i^{\left(1 - \frac{t}{T} \right)^b} \right) \right\}$$

Another mutation operator here we have non-uniform mutation. It is only because looking at the figure we have we have non-uniform or non-linear probability function. We can see from this figure that the non-linear probability distribution is used. In this case this because of this particular is spread this non-linearity will allow the new solution to be created close to the parent solution.

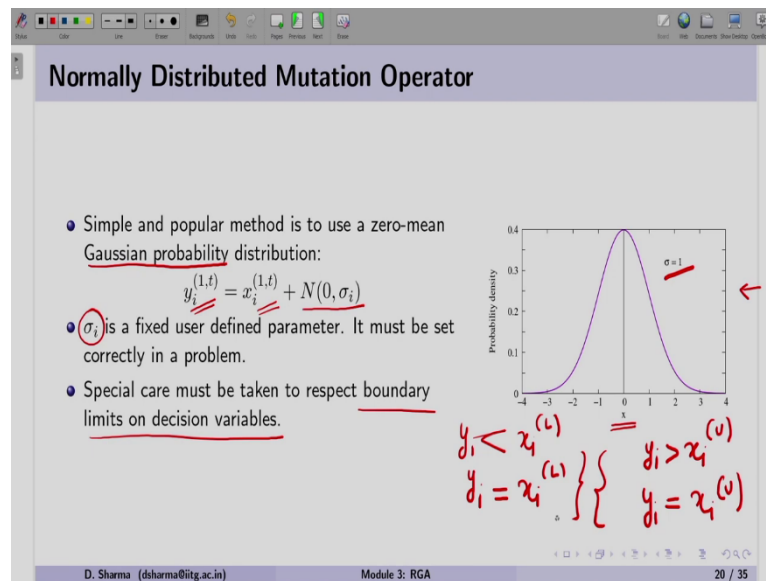
So, this particular operator was also proposed by Michalewicz in 1992 and it says that the probability of creating a solution closer to the parent is more than the probability of creating away from it. It is only because the non-linear probability distribution function is used here.

Now, along with this non-linear probability function there are some other terms that are also used. So, as you can see the new solution y_i is created with the help of the parent solution plus there is a δ_i parameter which is included. From the formula given here we can see that the δ_i depends on the difference between the two parents.

And, the another factor which is the $1 - r_i^b$ random number; so, r_i represents the random number and the power of the random number depends on the current iteration. So, t is the current iteration and the capital T is the maximum allowed generations. So, in this case you will realize that when t is 0 means at the beginning the power of r_i is 1, but when t reaches to capital T then r_i to the power will become 0.

So, in this case the random number has no contribution in generating the δ_i value. So, this T_{max} as I have mentioned, it is the maximum allowed generation and the power b is the non-negative user define parameter. So, you can see this b which is a non negative just to increase the number or to give the higher power. So, that will be affecting the r_i because everything is on r_i that is a random number here.

(Refer Slide Time: 46:41)



Thereafter the one of the simplest cross mutation operator which we can think is using normally distributed mutation operator. From the figure you can see this is the Gaussian distribution. In this Gaussian distribution we have taken like a sigma equals to 1 and the mean is at the 0. So, this is small perturbation we can use it with the solution.

Now, looking at the formula here; so, y_i the new solution is equals to the parent solution, but plus the perturbation using the Gaussian probability distribution as you can see here. Now, as we know that if we keep changing the value of this sigma i , then the spread of the curve will change. So, therefore, we have to find this sigma i value or we have to define it very carefully. So, your sigma i is going to be the user defined a parameter now.

Again for this mutation operator we have to take care of the lower and the upper bound as a recap. If the solution y_i is smaller than the lower limit, then we will keep y_i as is on the lower limit. Similarly, if a solution is created out of the upper bound then we will keep this solution on the upper bound. So, these simple conditions we can use it for our implementation. So, every time when a solution is mutated the it will be generated within the range of lower and upper bound.

(Refer Slide Time: 48:37)

Polynomial Mutation operator

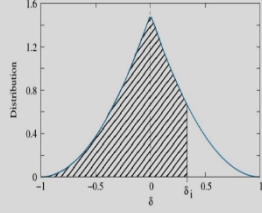
- Use polynomial distribution for perturbing a solution

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

- $\bar{\delta}_i$ is calculated from polynomial probability distribution

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

- Calculate $\bar{\delta}_i$ by equating area under the probability curve equal to r_i (a random number $\in [0, 1]$)

$$\bar{\delta}_i = \begin{cases} (2r_i)^{1/(\eta_m+1)} - 1, & \text{if } r_i < 0.5, \\ 1 - [2(1 - r_i)]^{1/(\eta_m+1)}, & \text{if } r_i \geq 0.5. \end{cases}$$


D. Sharma (dsharma@itg.ac.in) Module 3: RGA 21 / 35

$$y_i^{(1,t+1)} = x_i^{(1,t+1)} + (x_i^{(U)} - x_i^{(L)})(\delta_i)$$

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

$$\bar{\delta}_i = \begin{cases} (2r_i)^{\frac{1}{\eta_m+1}} - 1, & \text{if } r_i < 0.5 \\ 1 - [2(1 - r_i)]^{\frac{1}{\eta_m+1}}, & \text{if } r_i \geq 0.5 \end{cases}$$

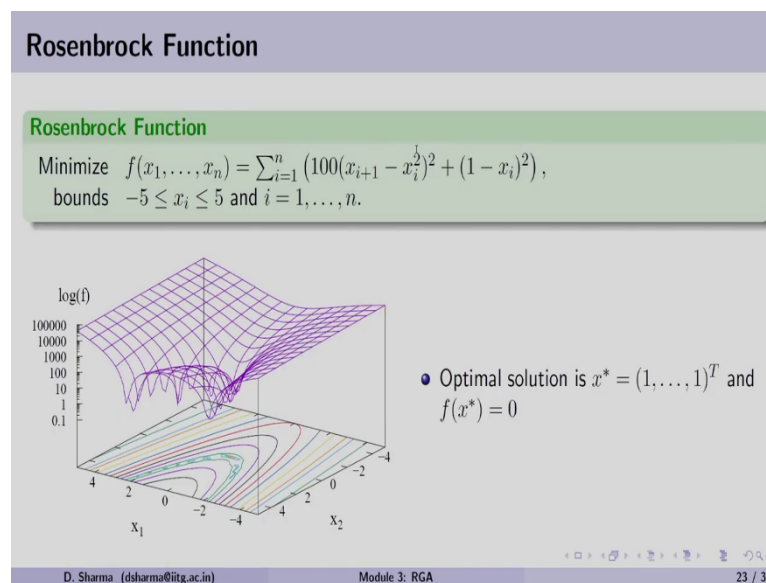
Come to the polynomial mutation: polynomial mutation is discussed in the previous session when we solved or we have an example we understood the principle working principle of RGA. So, as you can see in the polynomial mutation it is also following the same that the new solution is created with respect to the parent solution and the difference between the two the lower and the upper limit multiplied by the delta i value.

Now, looking at the figure on the right hand side you can see again the non-linear a probability function is used, so that it will allow the new solution to be created closer to the parent solution. Now, in this case the probability distribution as you can see here it is used

and when we equate the area under the curve as you can see in the figure on the right hand side we can calculate the data i value.

So, this delta i value as you can see if random number is smaller than 0.5 we have 1 formula when random number is greater than 0.5, we have another formula. We have come to the simulation of RGA.

(Refer Slide Time: 49:59)



Now, let us see how this RGA works of for different kind of problems we will start with the Rosenbrock function as you can see here.

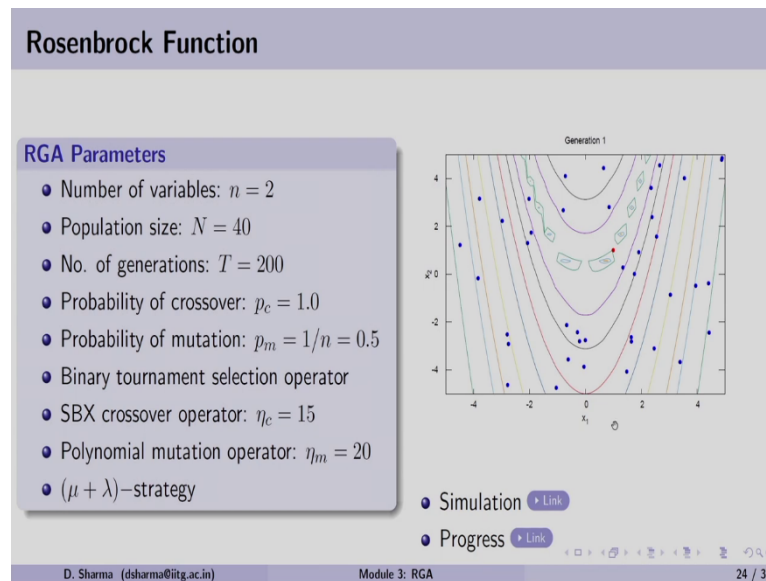
$$\text{Minimize } f(x_1, \dots, x_n) = \sum_{i=1}^n (100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$$

$$\text{bounds } -5 \leq x_i \leq 5 \text{ and } i = 1, \dots, n$$

So, we want to minimize the function. So, this Rosenbrock function is written in terms of n variables and the generic form of the Rosenbrock function is given here. The limits of the variables are taken as from minus 5 to plus 5.

Now, two variable problem as you can see in this figure on the left hand side; the y-axis is taken as the logarithmic of the function, so that we can see the we can see the surface of this function and in x and x 2 plane we can see the contours. For this particular function the optima is lying at 1, 1 and the function value is 0.

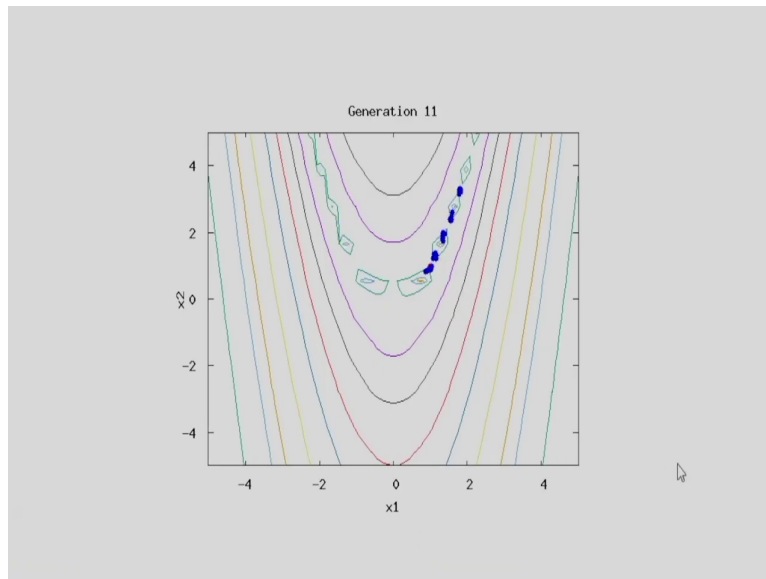
(Refer Slide Time: 50:57)



So, in the first simulation we kept the number of variable as 2, population size 40, number of iteration 200, probability of a crossover we keep it high because it has so many local optima's. We are keeping the probability mutation with a thumb rule 1 divided by n, n stands for number of variable. So, it comes out to be 0.5; then we use binary tournament selection operator and we also use SBX crossover operator to generate new solutions.

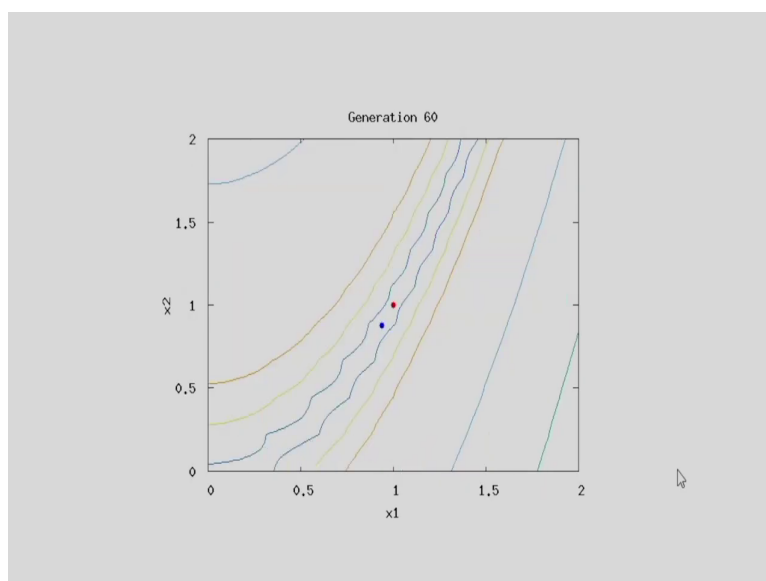
So, here eta c value is 15 and the polynomial mutation is used which has a eta m value 20. Finally, we used mu plus lambda strategy here. So, in when we generate the initial population you can see on the right hand side, the blue dots are the initial random solutions which are generated in x 1 and x 2 plane. So, we call it is a random initial population.

(Refer Slide Time: 52:15)



So, let us start with the simulation here. Now, as you can see the solutions are distributed.

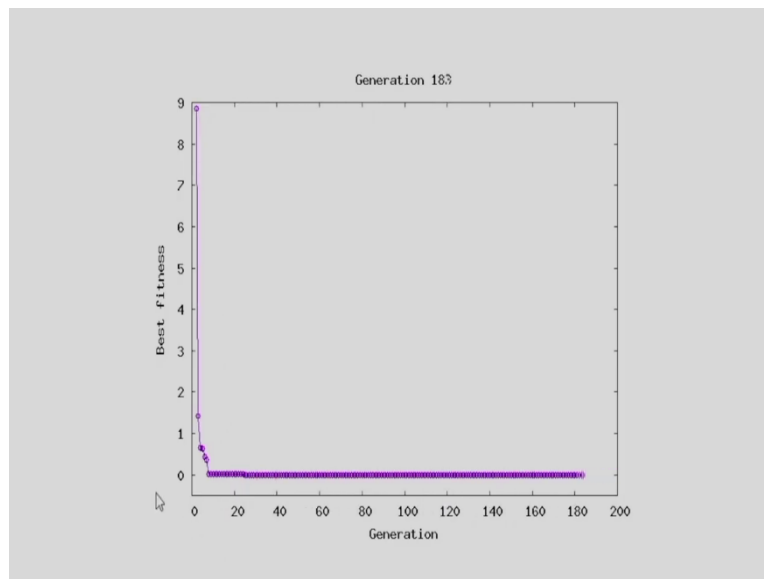
(Refer Slide Time: 52:19)



And, in 15 – 16 generations the solutions are already close to the optima and closed to 100 generation we got the solution. So, let us see one more time, so that we can understand how quickly these solutions are converged to the optimum solutions.

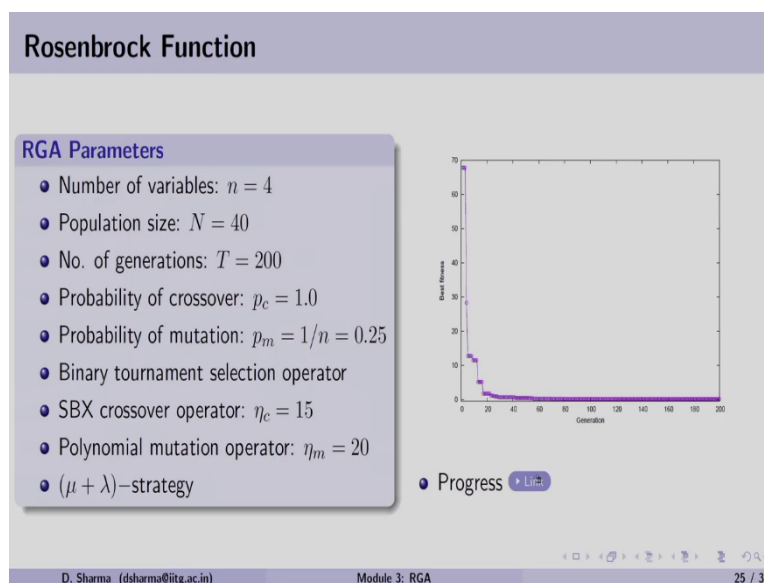
Since these variables are represented as a real number, so, any precision is possible and that is why RGA with the help of SBX crossover operator and polynomial mutation we got the solution early. Now, let us see the progress. So, the progress here is shown with respect to the best solution in the population.

(Refer Slide Time: 53:01)



So, you can see in the y-axis we have a best fitness in the population in every generation, an x-axis is our generation. So, here initially we can see a drastic improvement and thereafter since we have already close to the optima so, the performance is or the improvement is less. So, as you can see the performance is quite good at the in the initial generation and even at before 10 generation, we are very close to the optimum solution here.

(Refer Slide Time: 53:39)



In the next simulation, we are solving the same Rosenbrock function. We have taken the number of variable as 4. So, in the last example we took 2 variables, now the number of

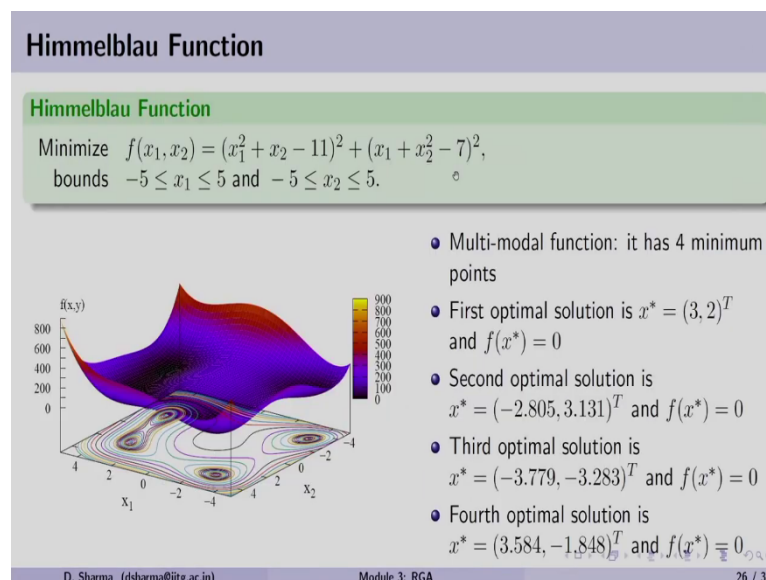
variable is 4. Population size we kept it same as 40; similarly the number of generation is 200; probability of crossover is 1. Now, the probability of mutation is reduced it is because we are using one thumb rule that the probability of mutation is equals to 1 divided by the number of variable. So, now, it is 0.25.

The same set of operators such as binary, tournament selection, SBX crossover, operator polynomial mutation and mu plus lambda strategy are used; eta c and eta m values are 15 and 20 respectively. Since it is a 4 variable so, we cannot see this simulation with respect to the contours of the function.

However, if you look into this figure we can see that the best fitness initially was close to say 67 and it reduces drastically by say 23 generations and then with a small improvement and close to 60 generation we got the optimum solution.

So, let us see how it worked. So, you we can see the improvement is drastic at the early generations and then slowly it is reaching to the minimum. Now, you can see the 0 is written here and before 60 generations a the RGA already found an optimum solution for 4 number of variables for Rosenbrock function.

(Refer Slide Time: 55:31)



Now, let us move to the another function which is Himmelblau function.

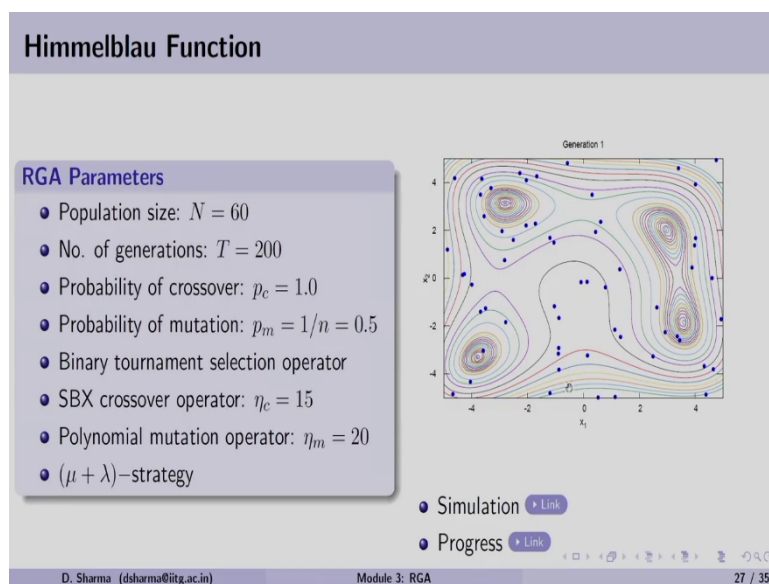
$$\text{Minimize } f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

$$\text{bounds} \quad -5 \leq x_1 \leq 5 \quad \text{and} \quad -5 \leq x_2 \leq 5$$

This function is a 2 variable function as you can see on the top and the bounce on both the variables is from minus 5 to plus 5. Looking at the surface on the left hand side so, this surface is based on the Himmelblau function and if you look at the contours you can locate 4 optimas.

So, basically it is a multimodal problem as I have mentioned on the right and side. So, it is a multimodal function which has 4 minimum point. Now, you can see that it has minimum point at 3, 2. Similarly, you have other 3 optimum solution where the function value is 0.

(Refer Slide Time: 56:25)

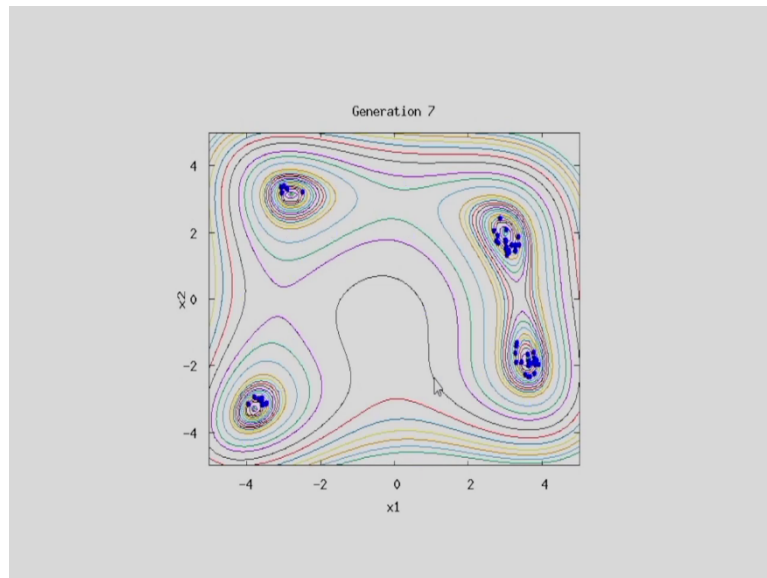


In this case we kept the population size 60 which is little larger than the Rosenbrock function just to make sure that we are solving a multimodal problem here. Number of a generation is kept a 200, probability of crossover is 1, probability of mutation is 1 by 1 by n as this problem is two variable. So, p m is equals to 0.5. We are using the same set of operators such as binary tournament selection, SBX crossover operator, polynomial mutation and mu plus lambda strategy.

Now, since it is a two variable problem we can we can show the simulation here. Now, looking at the figure on the right hand side this is the generation 1; this means we have this

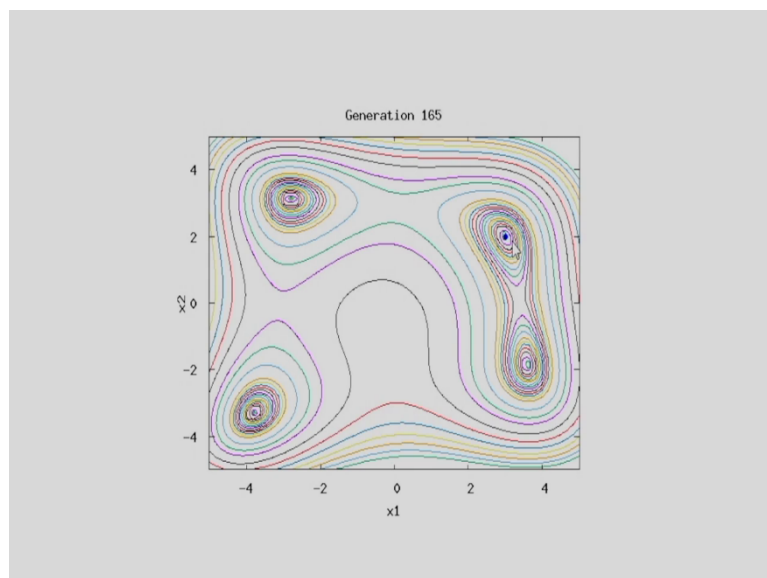
initial population which is generated randomly in the x_1 and x_2 plane. So, let us see how these solutions will progress towards the optimum solution.

(Refer Slide Time: 57:31)



Now, as of now the solutions are distributed along 4, but when we reach in to the 15th generation the solution were a two peak and after 50 generation we can see all the solutions are converged to the one peak.

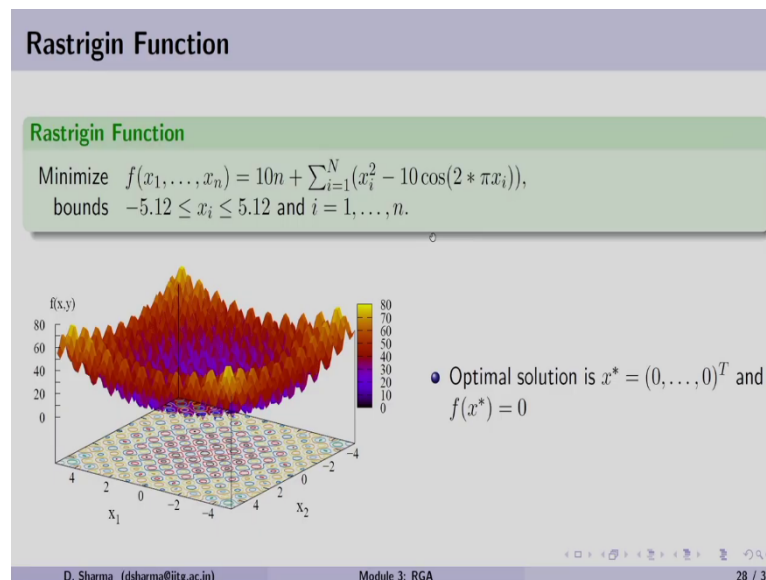
(Refer Slide Time: 57:47)



So, here the advantage is that you can see the solutions are quickly converged to converging to the optimum solution, but the disadvantage is that all solutions have converged to the one peak. As we have discussed earlier with the multimodal function that we should find out all the peak using EC techniques.

Since we do not have any fitness sharing kind of criterion here that will help us to solve multi model problem, so, this is the best example we can see that if we want to solve multimodal problem then we have to include some features for example, fitness sharing so that we can find the optimum solution at all four positions.

(Refer Slide Time: 58:41)



Now, let us move to the Rastrigin problem as you know this problem is difficult to solve because it has many local optima and a one global optima.

$$\text{Minimize } f(x_1, \dots, x_n) = 10n + \sum_{i=1}^N (x_i^2 - 10 \cos(2 * \pi x_i))$$

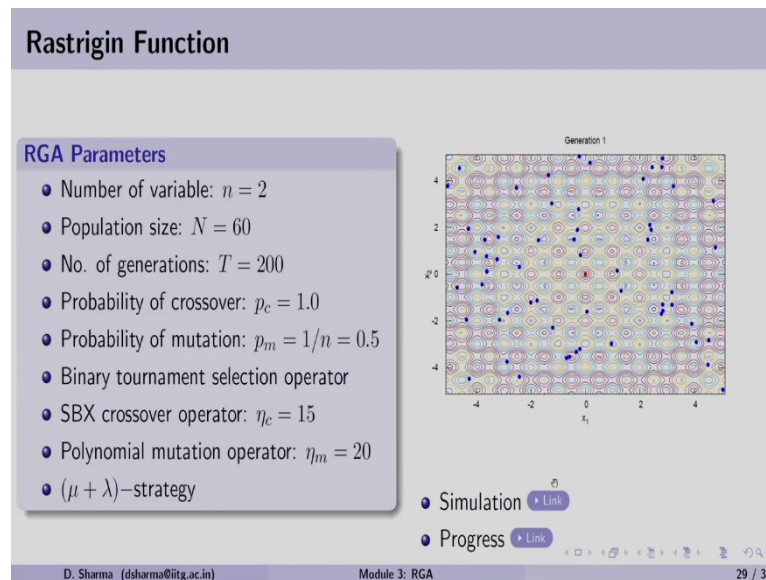
$$\text{bounds } -5.12 \leq x_i \leq 5.12 \text{ and } i = 1, \dots, n$$

For n variable function we can have an objective function which is given on the top and the variable the bounds on each variable is from minus 5.12 to plus 5.12.

Now, looking at the two variable function here, the surface you can make it out there are. So, many local minimas and from the contour itself it is clear that this problem is a difficult

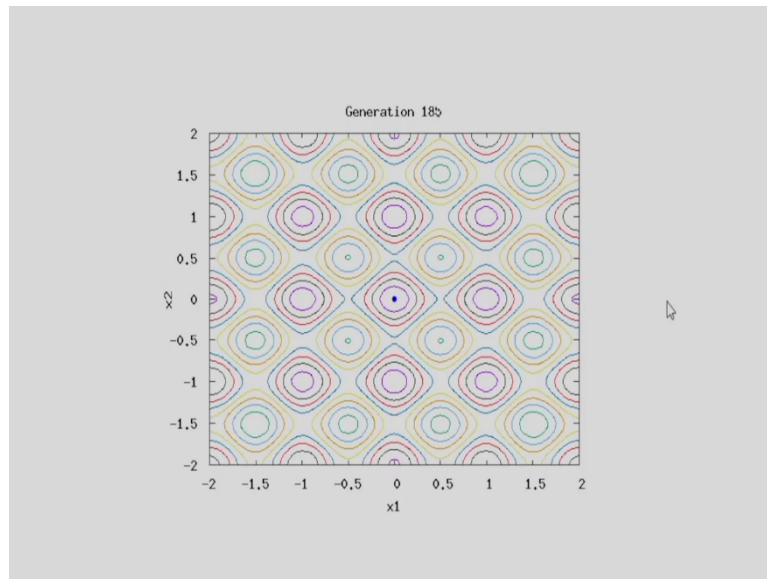
problem. So, the global optima for this problem is lying at the origin and the function value is 0.

(Refer Slide Time: 59:27)



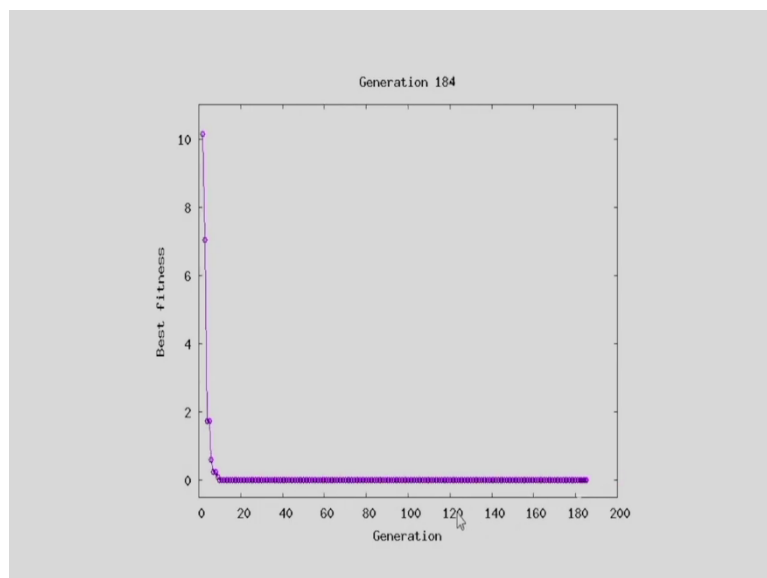
So, let us solve this Rastrigin function or a problem using RGA. Initially we have taken 2 variables, population sized 60, number of generation 200, probability of a crossover 1, probability of mutation 0.5 because number of variable is 2 and the same set of binary tournament selection, SBX crossover operator, polynomial mutation and mu plus lambda selection strategy. Now, the figure on the right hand side you can see the solutions are randomly generated and let us see how this RGA will work for this kind of a difficult problem.

(Refer Slide Time: 60:11)



Now, from the simulation you can see the solutions are quickly moving and in less than 30 generation we got the optimum solution using RGA. So, that is the advantage when we deal with the real numbers and when we have efficient crossover mutation operators. So, again you can see how quickly the solutions are converged to the optimum solution.

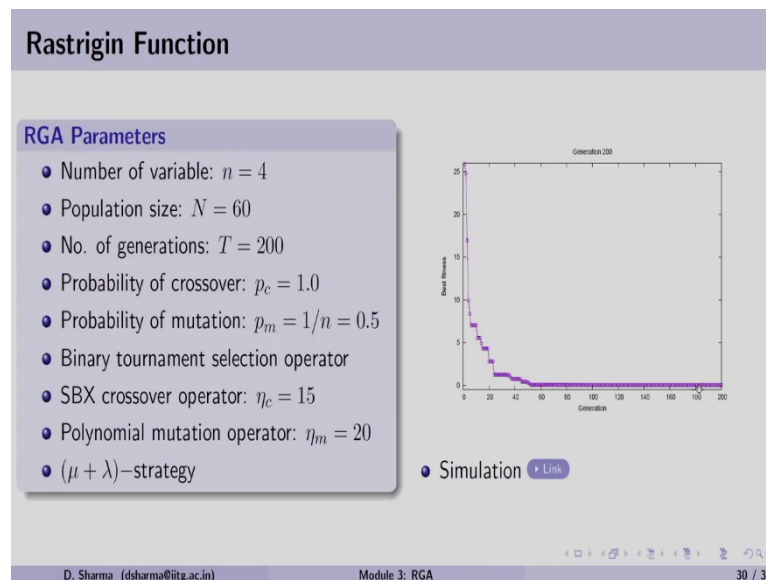
(Refer Slide Time: 60:41)



Now, coming to the progress let us see how the progress is visible or if we can observe with RGA. Now, initially it is started and within 10 or 15 generations, the solutions are already on the optimum solution and with the generation the rest of the solutions are converged to the optimum solution.

So, let us see one more time here how quickly the solutions are converged to the optimum solution. So, within 15 generation the optimum solution was found by using RGA. So, we can see that RGA is one of the efficient EC technique for solving real parameter optimization problem.

(Refer Slide Time: 61:23)



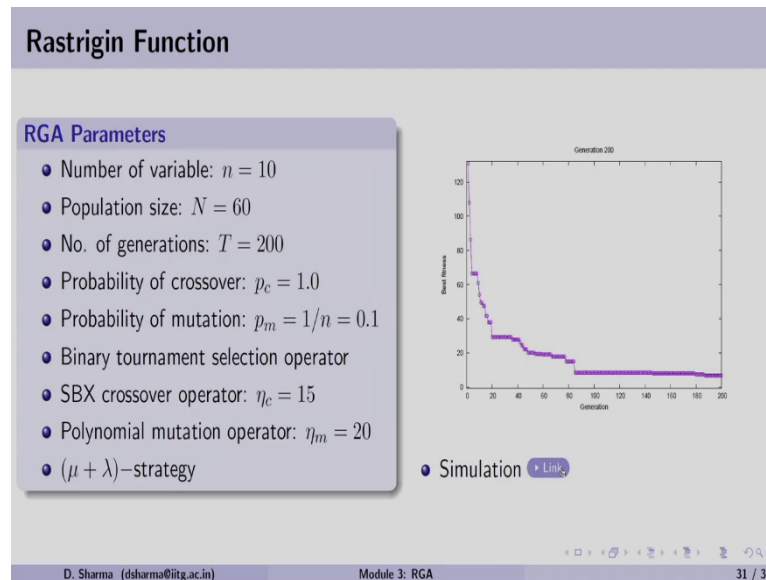
In another simulation, we have taken the number of variable as 4, population size is 60, number of a generation 200, probability of crossover 1, probability of mutation 0.5 and the same set of operators for selection and variation. Now, since it is a variable problem, we cannot see the simulation; however, we can see the progress as it is it can be seen in the figure.

So, it is started from say 25 close to the 25 value and almost close to 50 generation there is a drastic improvement in the fitness and thereafter, we since we have reached to the optimum solution, so, the progress can be seen as a straight line. Now, let us see this simulation the progress simulation.

So, here you can see how quickly the solutions have converged and almost after just 50 iterations we have reached to the optimum solution. So, after close to the 90 generation we have a little improvement that is just we are improving our optimum solution using RGA.

So, in this case we know that Rastrigin function itself is a difficult problem. Even we have increased the number of variable from 2 to 4; RGA is efficient to solve such kind of difficult problem where you have multiple local optimum solution and one global optimum solution.

(Refer Slide Time: 62:57)



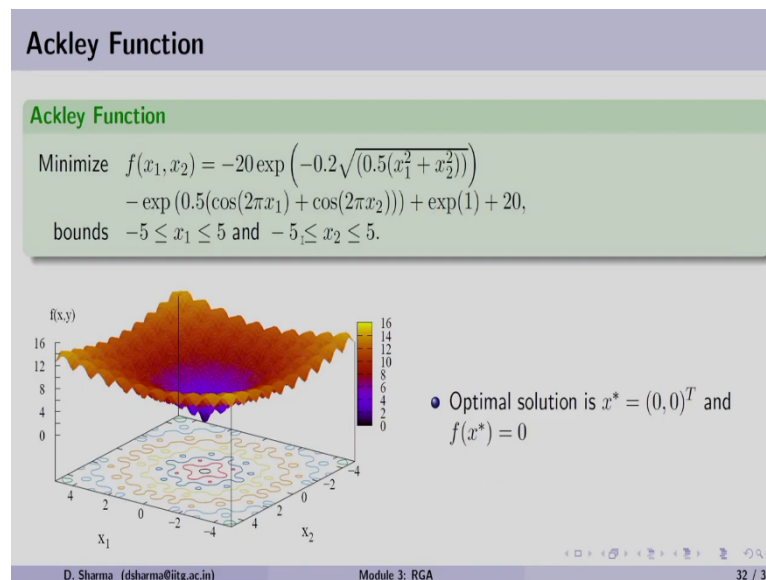
We also performed this RGA simulation for Rastrigin function having 10 number variables, the population size is 60, a number of generation is 200, probability of crossover is 1. Now, probability of mutation is 1 by n, so, 0.1 it is because the n is 10 basically the number of variable is 10.

We use the same set of selection operators for running the RGA. Now, let us see the progress as you can see the best fitness is started somewhere close to 130 it is keep on improving here and just after 80 generation it is almost like a straight line and they are further improvement close to 180 generation. So, let us see what is the progress here.

Now, you can see the progress is steady and the solutions are improving. So, best fitness is keep on improving with the number of generation. And now, after 80 generation as we have seen earlier there was no improvement and then there was a little improvement after 140 and 180. Important point here is that for 10 variable problem we can see in this figure that still we are little far from the a best fitness which is 0 which is the optimum solution for the given problem.

So, in this case, if we want to improve RGA we can work with the more number of population. So, currently we have taken 60 so we can take it little larger why because the number of variable is 10 which is a large number. Similarly, number of a generation can be increased to see how this RGA will perform.

(Refer Slide Time: 64:51)



Come to the Ackley's function.

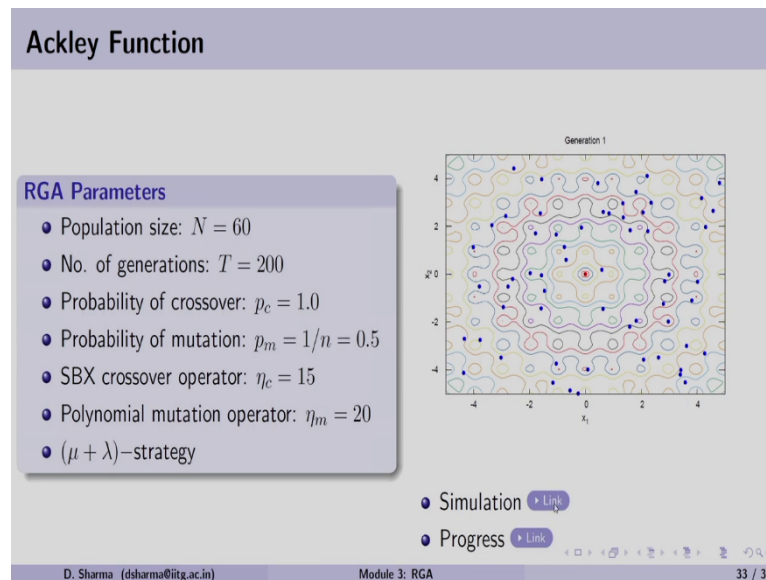
$$\begin{aligned} \text{Minimize } f(x_1, x_2) &= -20 \exp \left(-0.2 \sqrt{0.5(x_1^2 + x_2^2)} \right) \\ &- \exp(0.5(\cos(2\pi x_1) + \cos(2\pi x_2))) + \exp(1) + 20 \\ \text{bounds } &-5 \leq x_1 \leq 5 \text{ and } -5 \leq x_2 \leq 5 \end{aligned}$$

Now, Ackley function is again a two-variable function, looking at the equation you can see or observe that this function is going to have multiple local optima along with a global optimum. The variable bounds are given as between minus 5 to plus 5 or both the variables now look at the surface of Ackley function.

Now, small ups and downs in the surface suggest that there are many local optimum solution. Looking at the contour of x_1 and x_2 that says that yes there are many and we have a one

global optimum solution. So, the optimum solution is at the origin which is 0, 0 and the function value is 0.

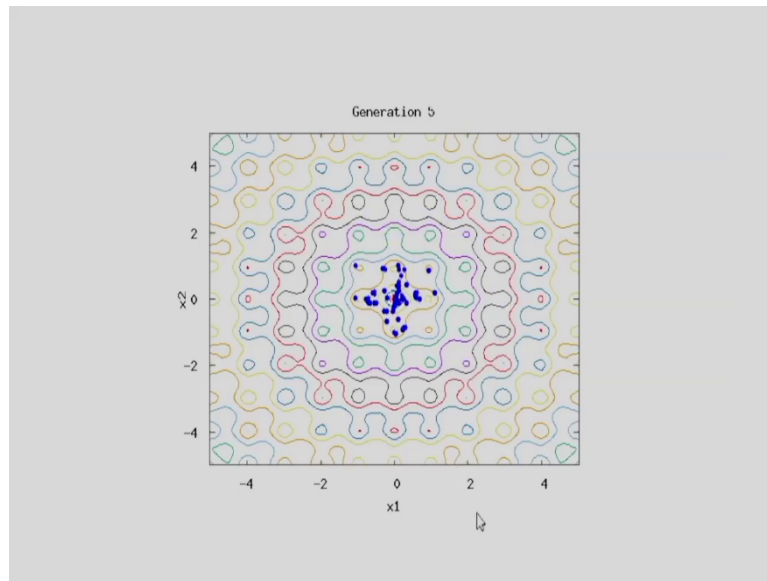
(Refer Slide Time: 65:39)



For running the Ackley problem we took the population size of 60, number of generation 200, probability of crossover is 1, probability of mutation is 0.5 and we have a SBX crossover operator with eta c 15, polynomial mutation 20 and mu plus lambda strategy we have use and we have taken.

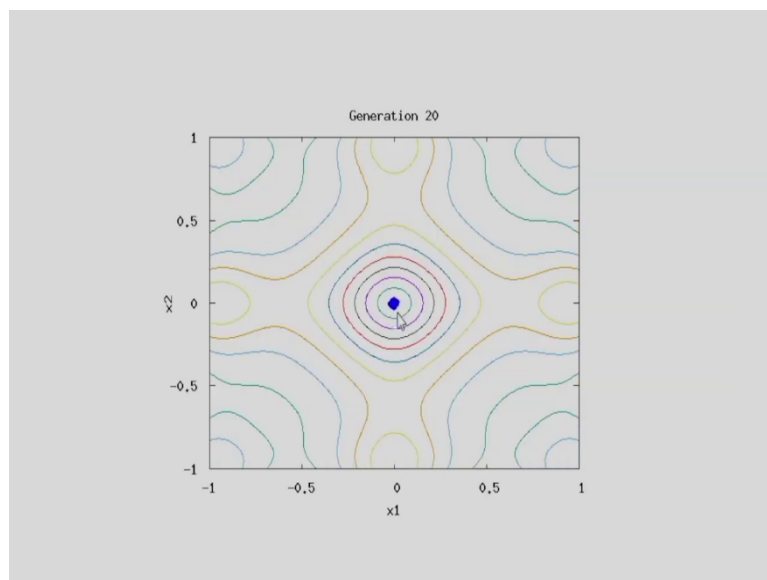
Now, here since it is the repetition, we did not mention that we have also used the binary tournament selection. So, binary tournament selection operator is also used here. Now, a in the generation 1 this is how the population members are distributed in x 1 and x 2 which are randomly generated. So, let us see how what is the simulation.

(Refer Slide Time: 66:33)



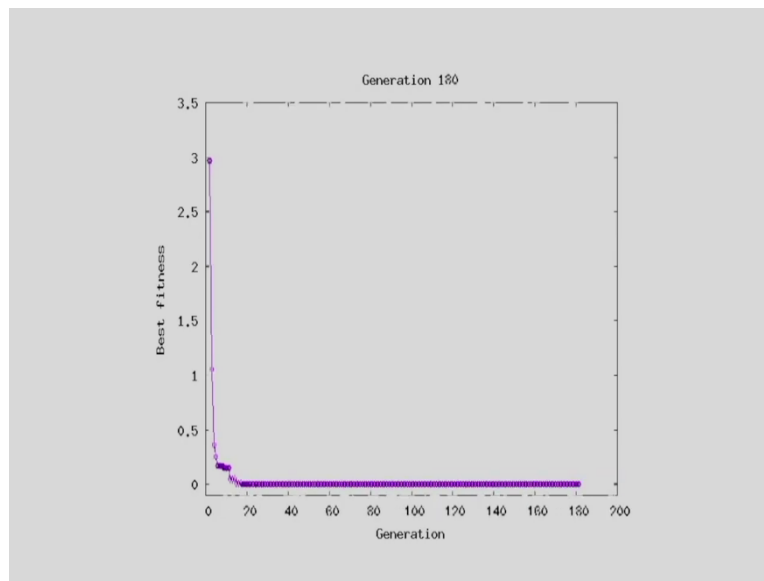
Now, looking at this how quickly the solutions have moved to the optimum solution in less than 30 or a 20 generation.

(Refer Slide Time: 66:35)



So, once it is generated now you can see within 5 or 10 generations, the solutions are actually converged near to the optimum solution. So, RGA is you we can say that RGA is able to solve the problems such kind of a problems which has so many local optima and one global optima.

(Refer Slide Time: 67:09)



Let us come to the progress. Now, in this progress we can see that how the fitness is improved quickly and close to 15 number of generations we have already reached to the optima and therefore, afterwards there is a straight line in the convergence plot or the progress plot. Let us see one more time how quickly it has converged. So, it is already just after the 50; 50th generations we the RGA has converged to the optimum solution.

(Refer Slide Time: 67:39)

Closure

- Various Operators for RGA
 - Selection operators
 - Crossover operators
 - ▶ Linear crossover operator
 - ▶ Blend crossover operator
 - ▶ SBX crossover operator
 - ▶ Fuzzy recombination operator
 - ▶ Unimodal normally distributed crossover operator
 - Properties of crossover operator
 - Similarity among crossover operators on a flat landscape function
- Mutation operators
 - ▶ Random mutation operator
 - ▶ Non-uniform mutation operator
 - ▶ Normally distributed mutation operator
 - ▶ Polynomial mutation operator
- Simulations and application of RGA
 - ▶ Rosenbrock function
 - ▶ Himmelblau function
 - ▶ Rastrigin function
 - ▶ Ackley function

D. Sharma (dsharma@nitg.ac.in) Module 3: RGA 35 / 35

Now, with this let us come to the closure of this session, in this particular session we have discussed various operators for RGA. We found that the selection operators which we have discussed earlier that are also applicable with the RGA without any change. Thereafter, we have discussed series of crossover operators starting with the linear crossover operator followed by the blend crossover so, BLX alpha. We discussed SBX crossover and thereafter fuzzy recombination operator.

The last three these three crossover operator says that the new solution that is created proportional to the difference between the parent solution. So, that is why their search was found to be adaptive. Thereafter we have gone through unimodal normally distributed crossover operator which use the ellipsoidal probability distribution.

The important part which we discussed is the properties of the crossover operator. So, when we want to generate or we want to develop a new crossover operator, two desired properties were discussed such that the mean should remain the same and to keep the diversity variance should not reduce.

Thereafter, we also discussed about a very good study on the similarity of the crossover operator. That suggests that these three crossover operators they behave very similar only we have to find that characteristic parameter value, so that these three operators can behave similarly.

Thereafter we discussed mutation operator. In these in this particular list we have gone through the random mutation operator followed by the non-uniform mutation operator; in this particular mutation operator non-linear probability function was used. Then we have a normal distributed mutation operator in which Gaussian distribution was used. It is considered as one of the simplest mutation operator to implement for real numbers.

Finally, the polynomial mutation was discussed in the list of mutation operators. The working of RGA or and the simulation of RGA was shown with the help of four examples. So, we have seen that the Rosenbrock function, Himmelblau function, Rastrigin and Ackley's function RGA was found to be good in finding or searching the minimum solution.

With Rosenbrock function when we increase the number of variable the performance is we have to check the performance and the parameter of RGA has to be changed. For Hemmelblau function we found that since it is a multimodal problem, then RGA needs some

changes, so that we can locate all four minimum point. In the present case RGA converged to the 1 minimum point out of 4.

For Rastrigin and Ackley function RGA was found to be good and when we take 10 number of variable for Rastrigin function, the solution found by RGA was close to the optima, but not reached the optimum point. So, in this case, the number of the number of the population size or the number of generations can be changed, so that RGA can work for such kind of problem. With these detail I conclude this session.

Thank you very much.