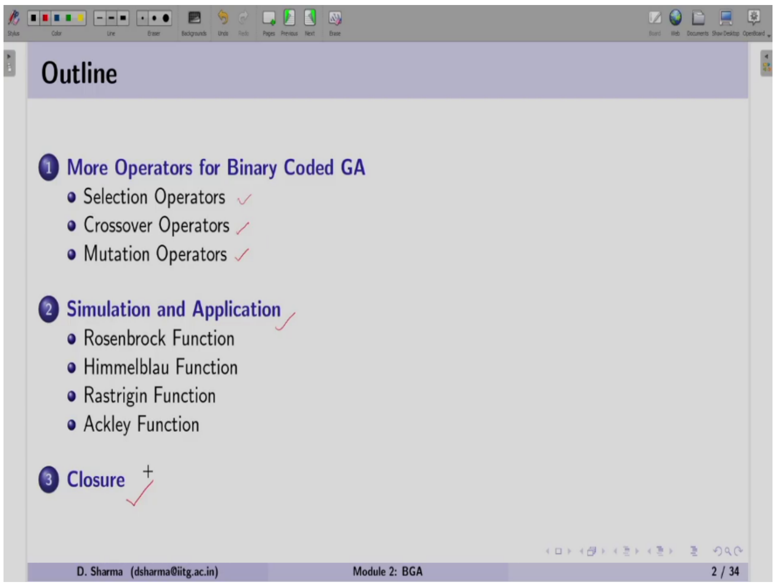
Evolutionary Computation for Single and Multi-Objective Optimization

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

Module - 02
Lecture - 04
Operators and Simulations of Binary-Coded Genetic Algorithm

Welcome to the session 2 of module 2. In this particular session, we will discuss more Operators that can be used with Binary Coded GA. Including that we will also discuss some simulations on mathematical functions for which the optima is known to us.

(Refer Slide Time: 00:59)



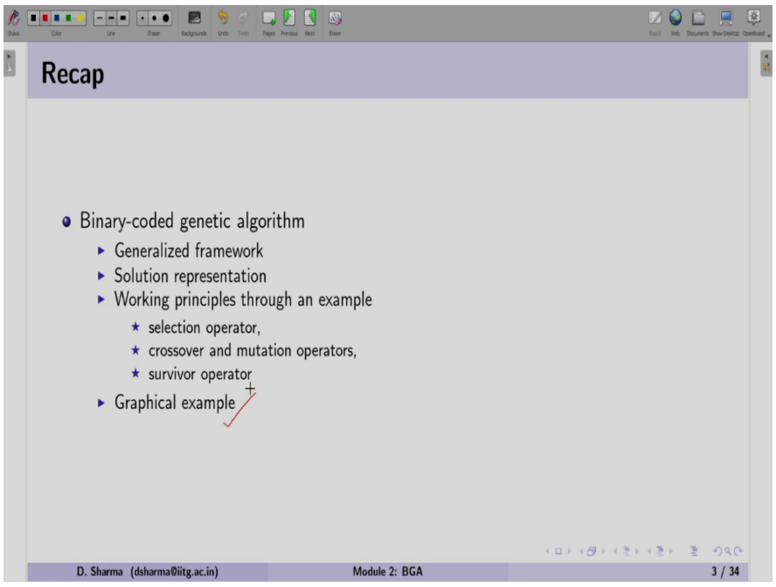
Outline

1. More Operators for Binary Coded GA ✓
   - Selection Operators ✓
   - Crossover Operators ✓
   - Mutation Operators ✓
2. Simulation and Application ✓
   - Rosenbrock Function
   - Himmelblau Function
   - Rastrigin Function
   - Ackley Function
3. Closure ✓

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 2 / 34

So, therefore, the outline of this particular session is decided as we will be talking about the selection operator, thereafter the crossover operators, and the mutation operators. So, various types of operator we will discuss and then we will be showing some application using binary coded GA.

(Refer Slide Time: 01:25)



Recap

- Binary-coded genetic algorithm
  - Generalized framework
  - Solution representation
  - Working principles through an example
    - selection operator,
    - crossover and mutation operators,
    - survivor operator
  - Graphical example

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 3 / 34

And, at the end we will close this module 2 on binary coded GA. So, before we move to the operators different types of operators, let us recap our previous session. In the previous session, we started with binary coded GA. And, we implemented and discuss the binary coded GA with the help of the generalised framework.

Using that generalised framework, we took a problem of Rosenbrock function and on this function, we started with the working principle of binary coded GA.

Since, in binary coded GA we use binary string. So, first we saw the solutions representing using binary string. When we have binary string? We can represent the decision like 0 or a 1, yes and a no on and off similarly. We can also use binary string for the discrete variable and the integer variable.

We took an example of real variable, where we took a binary string length of 5. And, accordingly we showed how this five-bit binary string, we decoded it and used the scaling formula, got the value of a real number.

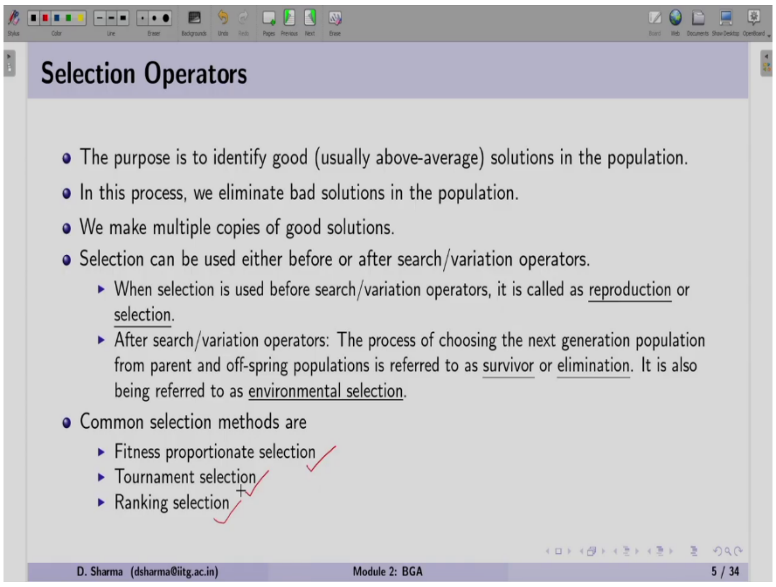
We use the lower and upper limit for that real number and put in the scaling formula and we get the value of the real number. There was a parameter called precision that also we discussed and we found that, if suppose we want more precision in that case we have to increase the length of binary string. After that, we took that Rosenbrock example in the working principle and then started with the selection operator.

In this selection operator, we discussed binary tournament selection and selected those solutions to making a mating pool. That mating pool then was used for the crossover and mutation operator. We discussed single point crossover operator and bitwise mutation operator. Afterwards, since the crossover and a mutation operator, they generated a population, which we refer as offspring population and we also started with a parent population.

So, since we have two population, that are parent and offspring population. So, we selected the best solutions from them. So, we used mu plus lambda strategy to select the best solution for the next iteration. In this particular process, we showed how the solutions were improved, some solutions were improved, using crossover and mutation some solutions have generated with the wrong value of the fitness.

The same example we took and we showed for the graphical example. So, just for one generation, we understood how the initial population was generated and using the selection followed by crossover and mutation operator, we get the solution. So, we found that the solutions in one generation started moving towards the optimum solution.

(Refer Slide Time: 05:13)



Selection Operators

- The purpose is to identify good (usually above-average) solutions in the population.
- In this process, we eliminate bad solutions in the population.
- We make multiple copies of good solutions.
- Selection can be used either before or after search/variation operators.
  - When selection is used before search/variation operators, it is called as reproduction or selection.
  - After search/variation operators: The process of choosing the next generation population from parent and off-spring populations is referred to as survivor or elimination. It is also being referred to as environmental selection.
- Common selection methods are
  - Fitness proportionate selection ✓
  - Tournament selection ✓
  - Ranking selection ✓

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 5 / 34

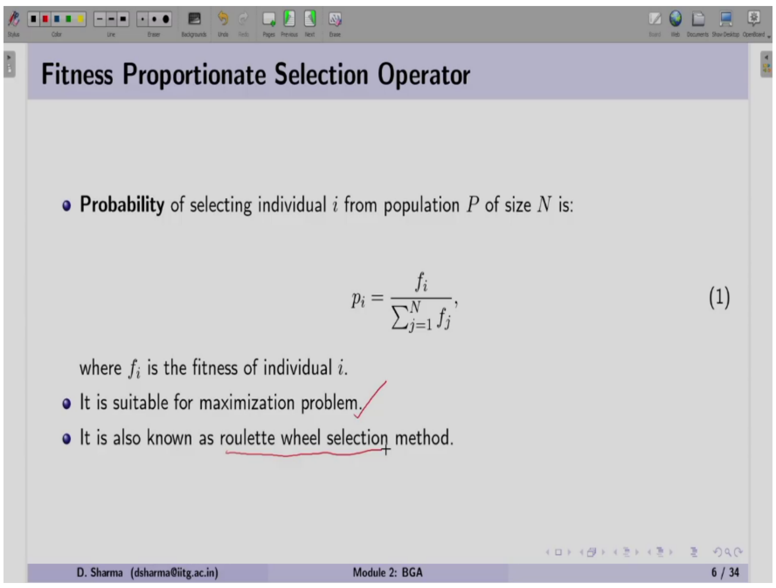
Now, let us begin with the more operators for a binary coded GA. In the previous session we discussed the selection operator. As a recap we know selection operator, the purpose of selection operator is to identify good and above average solutions. And, when we are doing this process, we eliminate bad solutions and make duplicate copies of the good solutions.

So, in this case we are emphasising the good solution and removing the bad solution. So, overall we are making a pool, as you remember that selection can be done before the variation operator and after the variation operator. When we perform before this before the variation operator, this selection process is generally referred to as reproduction or selection.

Thereafter, if we perform selection after variation operator, so, the main purpose is to find out best solutions that will be; that will be chosen for the next generation population.

This stage is referred as survival or the elimination. Sometimes it is also referred to as the environmental selection. In order to perform various kinds of selection or using different kind of selection operators, we have a list. For example, we can use fitness proportionate selection, we can use tournament selection, ranking method, and a few more. So, we will be focusing on few important selection operator in this session.

(Refer Slide Time: 07:05)



Fitness Proportionate Selection Operator

- **Probability** of selecting individual i from population P of size N is:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}, \quad (1)$$

where f_i is the fitness of individual i.

- It is suitable for maximization problem.
- It is also known as roulette wheel selection method.

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 6 / 34

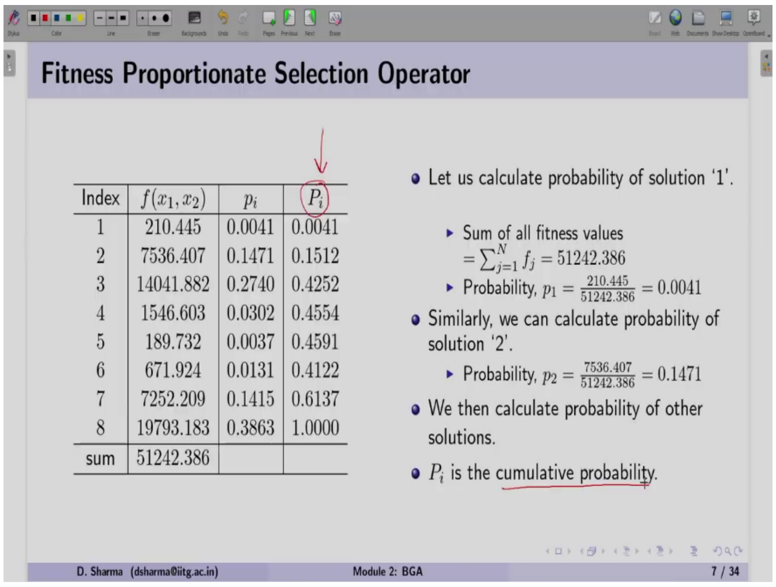
Let us begin with proportionate fitness selection. Now, here when we select a solution we before that, we find the probability of solution. Now, the probability of a solution and sometimes we also call it as a individual. So, the solution is the probability of a solution we calculate with the help of the fitness value. So, f i here represents the fitness of the solution, and this fitness is divided by the summation of fitness of all the solution in the population.

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

So, in this particular formula, you can realise that the value of p i depends on the value of the fitness of a solution. If, the fitness is large in that case the probability of selecting that solution is large. So, from this explanation you can get to know, that I can use the fitness proportionate selection operator for the maximization problem.

This operator is also known as the roulette wheel selection operator. So, there are different names some people refer as a fitness proportionate selection method and some refer to it as a roulette wheel selection method.

(Refer Slide Time: 08:37)



Fitness Proportionate Selection Operator

Index	$f(x_1, x_2)$	p_i	P_i
1	210.445	0.0041	0.0041
2	7536.407	0.1471	0.1512
3	14041.882	0.2740	0.4252
4	1546.603	0.0302	0.4554
5	189.732	0.0037	0.4591
6	671.924	0.0131	0.4122
7	7252.209	0.1415	0.6137
8	19793.183	0.3863	1.0000
sum	51242.386		

- Let us calculate probability of solution '1'.
  - Sum of all fitness values $= \sum_{j=1}^{N} f_j = 51242.386$
  - Probability, $p_1 = \frac{210.445}{51242.386} = 0.0041$
- Similarly, we can calculate probability of solution '2'.
  - Probability, $p_2 = \frac{7536.407}{51242.386} = 0.1471$
- We then calculate probability of other solutions.
- P_i is the cumulative probability.

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 7 / 34

So, let us see how it works? Here we have taken the same population that we have used with an example of binary coded GA. Now, the fitness of all these solutions have been taken. Now, it is important to note that, when we have to perform any selection, we do not need any binary string, or the decoded value of the binary string, or the value of x 1 and x 2. What we need is only the fitness value here, which is mentioned in the column number 2.

Now, in column number 3 we have mentioned the probability of each solution. So, let us see how we can calculate the probability of each solution? So, on the right hand side we will find the probability for this solution. Before that, we have to take the sum of all the fitness. So, if you look at this particular column number 2, if we take the summation of all these fitness values you will get this sum. This sum is the summation of the fitness of all solution.

$$= \sum_{j=1}^{N} f_j = 51242.386$$

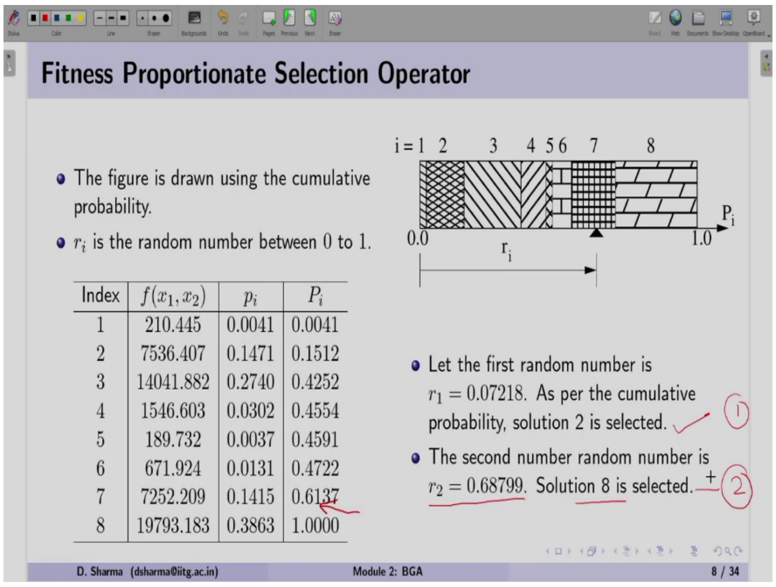
$$p_1 = \frac{210.445}{51242.386} = 0.0041$$

$$p_2 = \frac{7536.407}{51242.386} = 0.1471$$

Now, the probability of solution 1 is the numerator we get the same fitness of solution 1 here and the denominator is the summation. When you divide it you will get the probability as 0.0041, which is you can see in the 3rd column of the table. Similarly, suppose we follow the same procedure. In that case, we can find the probability of selecting solution 2 as in the numerator is the fitness of a solution 2, which you can see here. The denominator is the summation of the fitness. When we find it the probability is 0.1471.

So, this process will be followed and we can find the probabilities of all the solution by dividing the fitness with this summation of all the fitness. Now, it is here in the last column as you can see we are finding this capital P i, which is the cumulative probability. This cumulative probability we use to select solutions. So, let us see how we can implement fitness proportionate selection operator?

(Refer Slide Time: 11:25)



Fitness Proportionate Selection Operator

- The figure is drawn using the cumulative probability.
- r_i is the random number between 0 to 1.

Index	$f(x_1, x_2)$	p_i	P_i
1	210.445	0.0041	0.0041
2	7536.407	0.1471	0.1512
3	14041.882	0.2740	0.4252
4	1546.603	0.0302	0.4554
5	189.732	0.0037	0.4591
6	671.924	0.0131	0.4722
7	7252.209	0.1415	0.6137
8	19793.183	0.3863	1.0000

- Let the first random number is $r_1 = 0.07218$. As per the cumulative probability, solution 2 is selected. ①
- The second number random number is $r_2 = 0.68799$. Solution 8 is selected. + ②

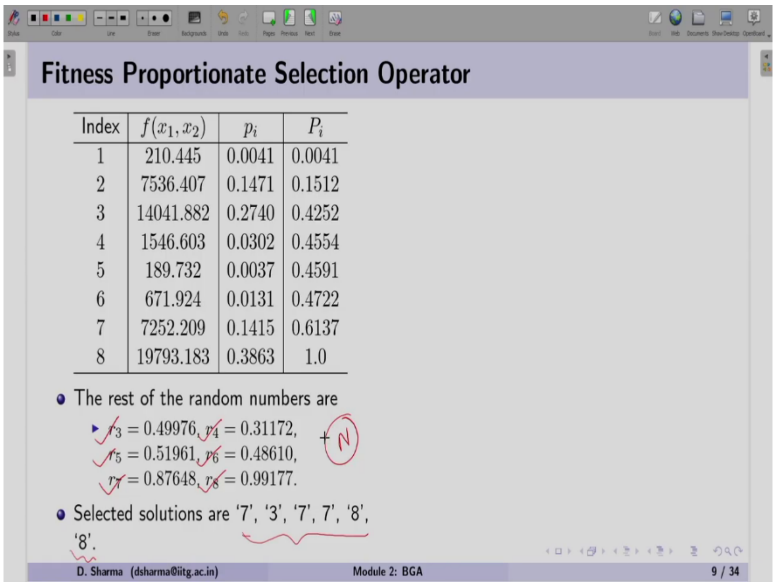
D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 8 / 34

Now, in the figure on the right hand side on the top, you can see this is solution 1, 2 and all these solutions till 8 all of them are given. Now, the rectangular boxes are shown here, representing the portion or the probability of the solution. So, corresponding to the probability these rectangles are shown here. On the x axis, we have the cumulative probability of capital P i. So, how will we select? We will generate a random number r i between 0 to 1.

Once, we generate it we can find where the counter is? So, you can see this particular counter. So, this counter will say, that we are going to select a solution. So, let us see how it works. So, in this case first of all we are going to calculate the probability of each individual, then we are going to find the cumulative probability, now we are generating random numbers. So, assume that the first random number is this. So, if you say 0.07218 is your first random number.

So, where you will be looking you will be looking at the cumulative probability. So, it will be coming here since it is coming here. So, we are going to select solution number 2. Similarly, if suppose you have the random number r 2 as given 0.68799 this says that, the counter will be here. In this case we are going to select solution 8. Now, by using this we have already selected solution number 1 and solution number 2. Now, we have to select few more solutions.

(Refer Slide Time: 13:31)



Fitness Proportionate Selection Operator

Index	$f(x_1, x_2)$	p_i	P_i
1	210.445	0.0041	0.0041
2	7536.407	0.1471	0.1512
3	14041.882	0.2740	0.4252
4	1546.603	0.0302	0.4554
5	189.732	0.0037	0.4591
6	671.924	0.0131	0.4722
7	7252.209	0.1415	0.6137
8	19793.183	0.3863	1.0

- The rest of the random numbers are
  - ✓ $r_3 = 0.49976$, ✓ $r_4 = 0.31172$,
    ✓ $r_5 = 0.51961$, ✓ $r_6 = 0.48610$,
    ✓ $r_7 = 0.87648$, ✓ $r_8 = 0.99177$.

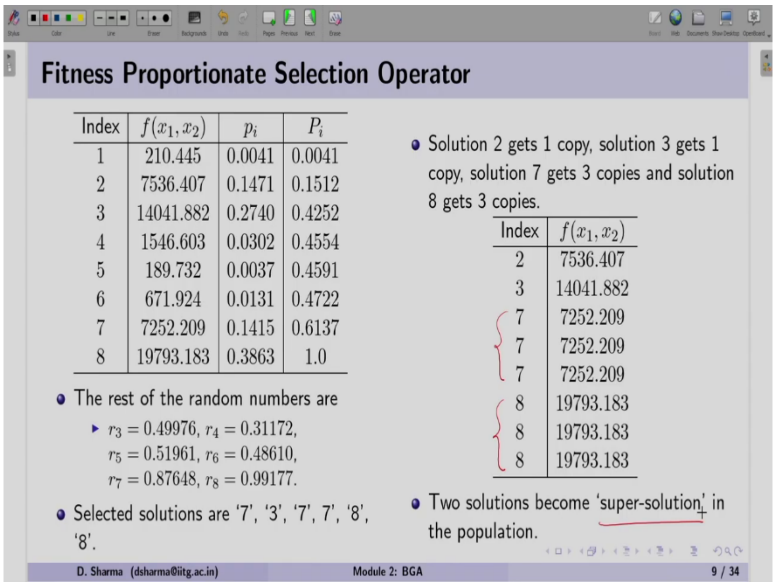
  $+$ N

- Selected solutions are '7', '3', '7', 7', '8', '8'.

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 9 / 34

Now, when we are going to follow the same procedure, let us assume that these are the probably, these are the random numbers, which are generated sequentially. When we generate it, the solution is 7, 3 and again 7, then one more time 7, 8 and finally, 8, all these solutions will be selected. Now, remember we have to remember here that we will be performing the selection operator such that we should select the solution equal to the number of N.

(Refer Slide Time: 14:13)



Fitness Proportionate Selection Operator

Index	$f(x_1,x_2)$	p_i	P_i
1	210.445	0.0041	0.0041
2	7536.407	0.1471	0.1512
3	14041.882	0.2740	0.4252
4	1546.603	0.0302	0.4554
5	189.732	0.0037	0.4591
6	671.924	0.0131	0.4722
7	7252.209	0.1415	0.6137
8	19793.183	0.3863	1.0

- The rest of the random numbers are
  - $r_3 = 0.49976$, $r_4 = 0.31172$, $r_5 = 0.51961$, $r_6 = 0.48610$, $r_7 = 0.87648$, $r_8 = 0.99177$.
- Selected solutions are '7', '3', '7', 7', '8', '8'.
- Solution 2 gets 1 copy, solution 3 gets 1 copy, solution 7 gets 3 copies and solution 8 gets 3 copies.

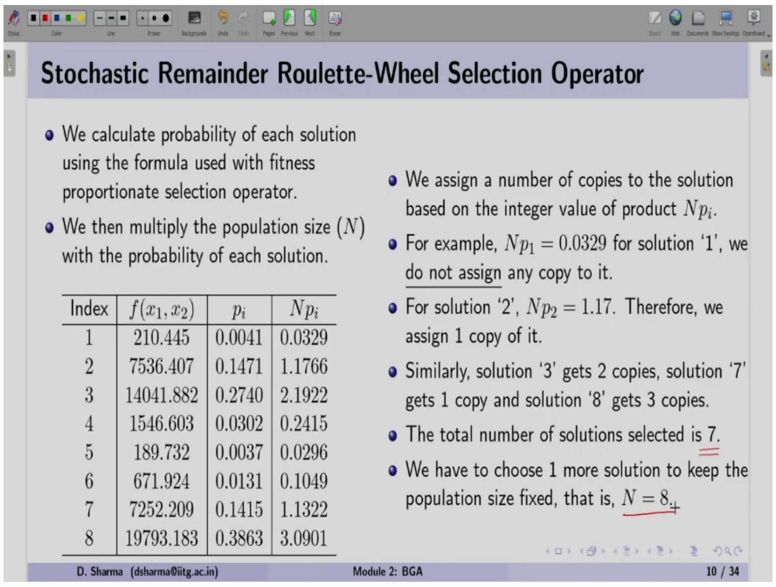
Index	$f(x_1,x_2)$
2	7536.407
3	14041.882
7	7252.209
7	7252.209
7	7252.209
8	19793.183
8	19793.183
8	19793.183

- Two solutions become 'super-solution' in the population.

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 9 / 34

So, let us copy all these solutions now. So, you can see that when we select a solution, solution 2 will get 1 copy, solution 3 will get 1 copy, solution 7 will get 3 copy, similarly solution 8 will get 3 copy. What you can understand here is these two solution especially 7 and 8 are the super solution. So, we will discuss, what is a super solution in the following slides.

(Refer Slide Time: 14:47)



Stochastic Remainder Roulette-Wheel Selection Operator

- We calculate probability of each solution using the formula used with fitness proportionate selection operator.
- We then multiply the population size (N) with the probability of each solution.

Index	$f(x_1, x_2)$	p_i	Np_i
1	210.445	0.0041	0.0329
2	7536.407	0.1471	1.1766
3	14041.882	0.2740	2.1922
4	1546.603	0.0302	0.2415
5	189.732	0.0037	0.0296
6	671.924	0.0131	0.1049
7	7252.209	0.1415	1.1322
8	19793.183	0.3863	3.0901

- We assign a number of copies to the solution based on the integer value of product Np_i.
- For example, $Np_1 = 0.0329$ for solution '1', we do not assign any copy to it.
- For solution '2', $Np_2 = 1.17$. Therefore, we assign 1 copy of it.
- Similarly, solution '3' gets 2 copies, solution '7' gets 1 copy and solution '8' gets 3 copies.
- The total number of solutions selected is 7.
- We have to choose 1 more solution to keep the population size fixed, that is, $N = 8$.

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 10 / 34

Another kind of selection operator we have is the stochastic remainder roulette wheel selection operator. In this selection operator we calculate probability of each solution using the formula used with fitness proportionate selection operator. So, the formula is going to remain the same.

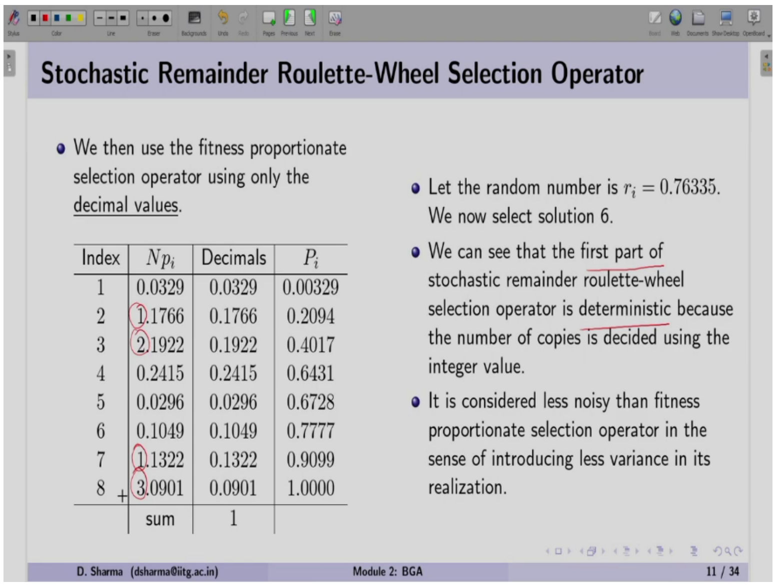
However, what we will do is, we will multiply the size of the population which is N, with the probability of each individual. So, here the same table we have drawn. So, we have index, we have fitness values and these are the individual probabilities. So, we use the same formula and we get the same probabilities.

In the last column you will see that, we have multiply the population size into the individual probability. So, this is the number which we get it. This selection operator says that, we assign several copies to the solution based on the integer value of the product of N p i. It means that, if we take a solution number 1. So, the value of N p 1 is 0.0329. Since we since at the integer value it is given 0, this means that we do not select this solution.

Now, let us look at solution number 2 here the value of N p 2 is 1.17. In this case we will select only the 1 copy of this solution. Similarly, the solution 3, so if we are going to follow. So, let us look into the table.

So, here the solution 2 we will get 1 copy, solution 3 we will get 2 copies, solution 7 we will get 1 copy and solution 8 we will get 3 copies. So, similarly we are going to get that many number of solution. If we count all the number of solutions, so, actually we selected 7 solution, and you remember that our population size is 8. So, we have to select one more solution.

(Refer Slide Time: 17:17)



Stochastic Remainder Roulette-Wheel Selection Operator

- We then use the fitness proportionate selection operator using only the decimal values.

Index	Np_i	Decimals	P_i
1	0.0329	0.0329	0.00329
2	1.1766	0.1766	0.2094
3	2.1922	0.1922	0.4017
4	0.2415	0.2415	0.6431
5	0.0296	0.0296	0.6728
6	0.1049	0.1049	0.7777
7	1.1322	0.1322	0.9099
8	3.0901	0.0901	1.0000
	sum	1	

- Let the random number is $r_i = 0.76335$. We now select solution 6.
- We can see that the first part of stochastic remainder roulette-wheel selection operator is deterministic because the number of copies is decided using the integer value.
- It is considered less noisy than fitness proportionate selection operator in the sense of introducing less variance in its realization.

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 11 / 34

In this case we use the fitness proportionate selection operator using the decimal values. So, here the same table we are drawing in which the N p we have written. Now, in this case you remember that these are the solutions, the copies of the solutions; we have already selected. So, we are going to remove them. So, we will be dealing only with the decimal places for all the solutions.

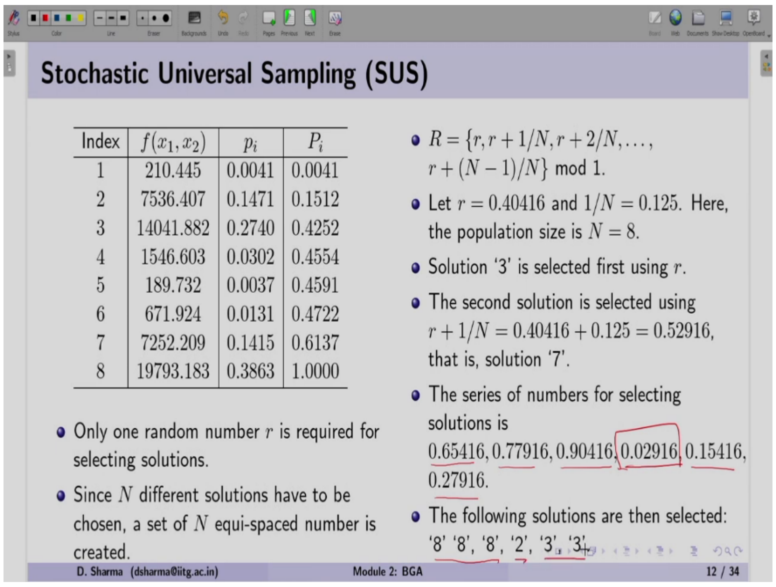
So, you can see that all the decimal values are given in this table. Then afterwards we will find the cumulative probability, as we have found it out you can see the summation is 1. Now, similar to the fitness proportionate selection method, we will generate random number and wherever it is lying the solution will be selected. So, let us assume that we have a random number of 0.76335.

If you look at this particular value so, the counter will be coming here. So, in this case we will be selecting solution number 6. So, we selected 7 solutions earlier, now by generating one random number, we selected one more solution, now the overall solution is equal to the size of the population.

Here, when we perform the stochastic remainder roulette wheel selection, you can realise that the first part of this operator is deterministic. It is because the numbers which we selected is based on the factor or the multiplication N p. So, since the probability was high, the solution got 1 copy, 2 copy or 3 copies, making the first part of this operator deterministic.

In the second part which we have just now discussed with the help of this random number and the decimal values, you can see that this is this stochastic. Since, it is since the first part was deterministic this operator is considered less noisy, than the fitness proportionate selection operator in the sense of introducing less variance in its realization.

(Refer Slide Time: 19:55)



Stochastic Universal Sampling (SUS)

Index	$f(x_1, x_2)$	p_i	P_i
1	210.445	0.0041	0.0041
2	7536.407	0.1471	0.1512
3	14041.882	0.2740	0.4252
4	1546.603	0.0302	0.4554
5	189.732	0.0037	0.4591
6	671.924	0.0131	0.4722
7	7252.209	0.1415	0.6137
8	19793.183	0.3863	1.0000

- Only one random number r is required for selecting solutions.
- Since N different solutions have to be chosen, a set of N equi-spaced number is created.
- $R = \{r, r + 1/N, r + 2/N, \ldots, r + (N-1)/N\}$ mod 1.
- Let $r = 0.40416$ and $1/N = 0.125$. Here, the population size is $N = 8$.
- Solution '3' is selected first using r.
- The second solution is selected using $r + 1/N = 0.40416 + 0.125 = 0.52916$, that is, solution '7'.
- The series of numbers for selecting solutions is $\underline{0.65416}, \underline{0.77916}, \underline{0.90416}, \boxed{\underline{0.02916}}, \underline{0.15416}, \underline{0.27916}$.
- The following solutions are then selected: '8' '8', '8', '2', '3', '3'.

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 12 / 34

We have another kind of selection operator, which is known as stochastic universal sampling. In short we say, it is called SUS operator. Now, in this operator, we will find the probability of each individual, the formula given in the fitness proportionate selection will be used, and thereafter we calculate the cumulative probability for each solution.

Now, in this selection operator, we are going to create only one random number. So, let us see how this random number is going to help us. So, since we will calculate just one, then we are making the N equi-spaced number. So, let us see how. So, we generate a random number. The second number will be r plus 1 by N. So, N is the population size.

$$R = \left\{r, r + \frac{1}{N}, r + \frac{2}{N}, \ldots, r + \frac{N-1}{N}\right\} mod\ 1.$$

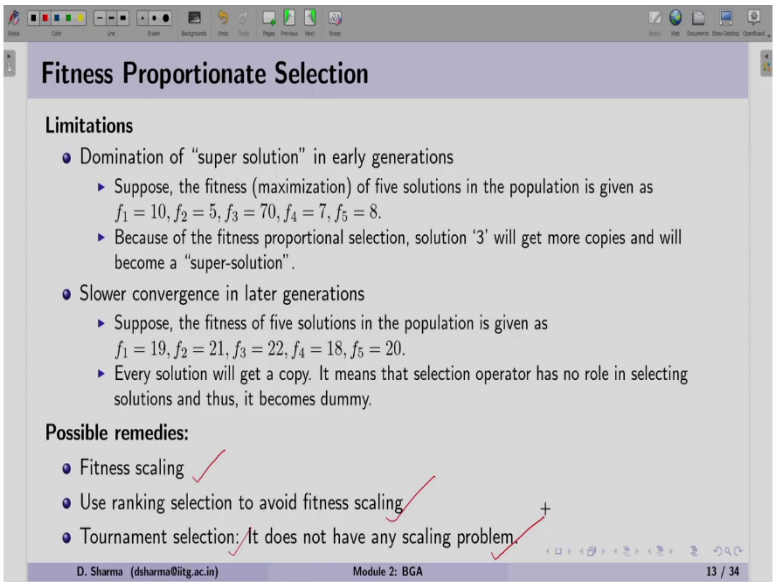
So, accordingly we have divided our range into equi-space and that is that depends on the size of N. Let us see how it is working.

Suppose, we have the random number as 0.40416 and if we calculate 1 by N we get a value 0.125, it is because the size of the population is 8. Now, since the value of a random number is 0.40, you can make it out that the counter will be here. So, in this case we will be selecting solution number 3. In the next stage what we will do is, we will have a random number plus 1 by N. So, this summation is going to give a value 0.52916 now if you look if you find where it is lying. So, this is the counter where we will get we can see the counter.

According to that the solution 7 is selected. So, this way we will be following and we can select the solution, using SUS operator. Now, when we perform this series of a number, we can get the numbers or get the number after 1 plus N. Now, once we have reached.

So, here you can note down that, once we are going we our number is going more than 1. So, we subtract to 1 and the value which we get we will be considering. So, it's like a circle and then we follow so, these numbers tells us that we are going to; we are going to select the 3 copies of solution 8, then solution 2 and then 2 copies of solution 3.

(Refer Slide Time: 23:05)



Fitness Proportionate Selection

Limitations

- Domination of "super solution" in early generations
  - Suppose, the fitness (maximization) of five solutions in the population is given as $f_1 = 10, f_2 = 5, f_3 = 70, f_4 = 7, f_5 = 8$.
  - Because of the fitness proportional selection, solution '3' will get more copies and will become a "super-solution".
- Slower convergence in later generations
  - Suppose, the fitness of five solutions in the population is given as $f_1 = 19, f_2 = 21, f_3 = 22, f_4 = 18, f_5 = 20$.
  - Every solution will get a copy. It means that selection operator has no role in selecting solutions and thus, it becomes dummy.

Possible remedies:

- Fitness scaling
- Use ranking selection to avoid fitness scaling
- Tournament selection: It does not have any scaling problem.

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 13 / 34

Now, there are certain limitations with the fitness proportionate selection operator, let us see, what are those. So, domination of super solution, so, this particular term we have used earlier.

So, let us see, what it means. Suppose, we are maximising our fitness and we have 5 solutions with us. Now, the fitness for each solution given as 10, 5, 70, 7 and 8.

Now, when we are comparing and finding the probability, you can make it out that the solution number 3, which has the fitness 70, which is a very big value compared to the fitness of other solution, will behave like this a super solution. Because this solution will get more copies and become dominant over the other solution, you will realise that there will be only one type of a solution in the population after a few generations.

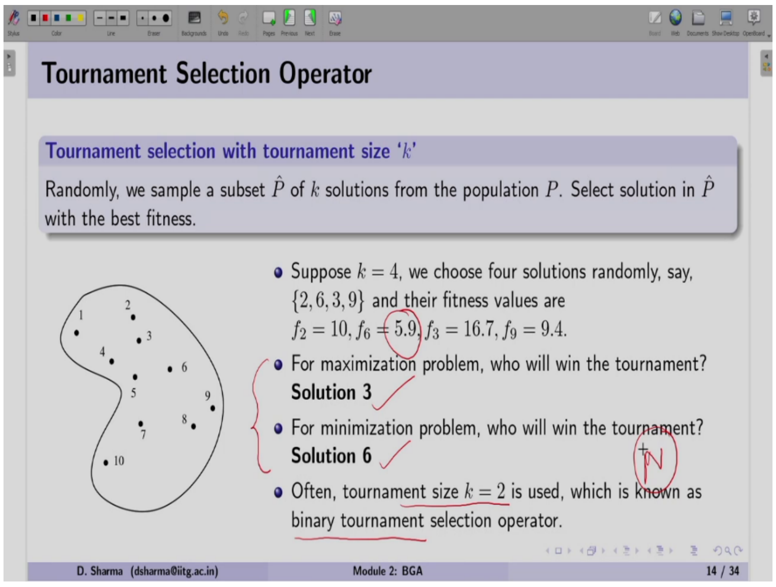
Another limitation is the slower convergence in later generation. So, again we take the fitness of 5 solutions and we are maximizing it. So, the f 1, f 2, f 3, f 4 and f 5 are given here, now let us compare them. Here, when you are going to calculate the probability of each solution, you can realise that these values like 19, 21, 22, 18 and 20 are very close to each other.

This means, that they are going to make an equal division on the cumulative probability scale. So, when we are going to generate a random number and selecting the solution, you will realise that everyone will get the copy. It means that, the selection operator is not performing anything and we are selecting the same solutions as we have earlier. So, in this case, you can realise that the selection operator will behave like a dummy because it selects all the solution given in the population.

So, these are the two limitations, where in one case, we have super individual, in which one individual has the largest fitness or very large value of fitness as compared to the other. In another case all solutions have very similar kind of fitness values. So, what are the possible remedies to the problem? We can use fitness scaling, but when we use fitness we have to perform some kind of scaling to select a solution. In this case, we can use a ranking selection and this ranking can avoid fitness scaling.

We also have a tournament selection. So, as we discussed in the previous session, we had a binary tournament selection operator. So, these tournament selection operators we drawn do not have any scaling problem.

(Refer Slide Time: 26:27)



Tournament Selection Operator

Tournament selection with tournament size 'k'

Randomly, we sample a subset $\hat{P}$ of k solutions from the population P. Select solution in $\hat{P}$ with the best fitness.

- Suppose $k = 4$, we choose four solutions randomly, say, $\{2, 6, 3, 9\}$ and their fitness values are $f_2 = 10, f_6 = 5.9, f_3 = 16.7, f_9 = 9.4$.
- For maximization problem, who will win the tournament? **Solution 3**
- For minimization problem, who will win the tournament? **Solution 6**
- Often, tournament size $k = 2$ is used, which is known as binary tournament selection operator.

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 14 / 34

Let us move to another type of operator, which does not behave like a proportionate fitness selection it is a tournament selection. Since, we have already gone through it, let us have a generalised definition of the tournament selection. Now, you can see that we have a tournament selection with the tournament size of 'k'. So, we are going to sample the k number of solutions randomly. So, we have a large population, we sample k solutions, and then select a solution based on the best fitness.

So, let us take an example here. Suppose, in this problem we have 10 solutions with us, in these 10 solutions let us choose we have a tournament size of a k. You may be wondering that earlier we discussed the tournament selection as a match between or tournament between the 2 team.

Since, this concept has been used we can simulate this particular operator for any number of tournament. So, we are taking in this example k as a 4. So, meaning that there are 4 solutions, they will be competing with each other and the solution which has the best fitness will be selected.

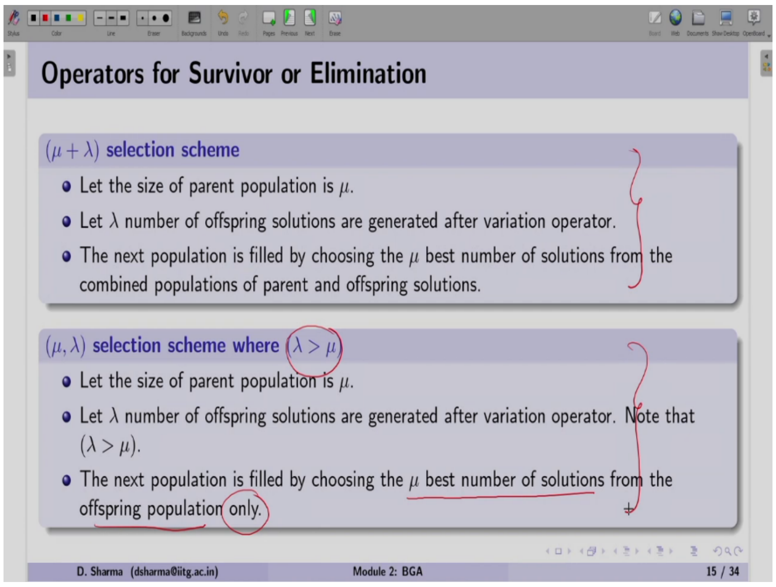
So, these four solutions suppose we have taken the four solutions: the randomly selected solutions and the fitness are given. For a maximization problem, if we ask a question which particular solution will win this tournament. Since, it is a maximization you can make it out that this fitness is the maximum 1, so, solution 3 is the winner. Similarly, for a minimization problem who will be the winner? Again we can see the fitness and we know that the solution 6 has the less fitness. So, we will select this solution.

What is interesting about this operator? That, whether it is a maximization problem or minimization problem, this operator can be used. Only the relationship we have to change. In maximization, we will be looking for the fitness for the more fitness, larger fitness; however, for the minimization problem, we will be looking for the least fitness.

So, if we change the relationship, we can use the same operator for both objective functions. However, you might have realised that the operator is designed for maximization problem only with the fitness proportionate selection method. Suppose you want to use for minimization problem. In that case, you have to use duality principle to convert minimization problem into maximization and then use it with the fitness proportionate selection method.

Now, often the tournament size we keep is 2 and that is why we call this tournament a binary tournament selection. So, we have discussed thoroughly in session 1, how this binary tournament selection works? Only important point you have to remember that, the number of tournament has to be done in such a way that the, we should select N number of solutions.

(Refer Slide Time: 30:13)



Operators for Survivor or Elimination

$(\mu + \lambda)$ selection scheme

- Let the size of parent population is μ.
- Let λ number of offspring solutions are generated after variation operator.
- The next population is filled by choosing the μ best number of solutions from the combined populations of parent and offspring solutions.

(μ, λ) selection scheme where $(\lambda > \mu)$

- Let the size of parent population is μ.
- Let λ number of offspring solutions are generated after variation operator. Note that $(\lambda > \mu)$.
- The next population is filled by choosing the μ best number of solutions from the offspring population only.

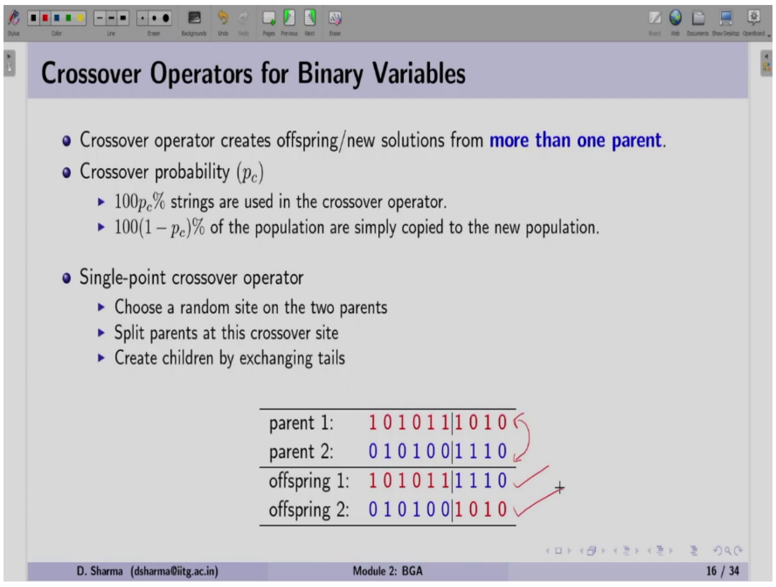
D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 15 / 34

We have operators for survivor or elimination. We discussed this mu plus lambda selection scheme, it was easy to implement. In this selection scheme we have mu number of parent population and after performing say variation operator, we created lambda number of offspring solutions. So, in this process what we do here is we combine both of them. So, combined population of parent and offspring solutions we took, we sort and select the best solution for the next generation population after sorting.

Second type of selection scheme which we can think of is the mu lamda selection scheme. In this case, the lambda should be more than mu, meaning we generate more offspring solutions than the parent solution. So, in this case we are going to select best mu solutions from the offspring population only. So, here we are not going to involve any parent solution, both the selection schemes support elitism.

Now, elitism is a word we use when we keep a good solution in our population. Now, in the first scheme, you can realise that when we combine parent and offspring population, we are not going to miss any of the good solutions. In the case of mu comma lambda is scheme, if the offspring solutions are not as good as parent solution, we may lose the best solution in this process.

(Refer Slide Time: 32:13)



Crossover Operators for Binary Variables

- Crossover operator creates offspring/new solutions from **more than one parent**.
- Crossover probability (p_c)
  - $100p_c\%$ strings are used in the crossover operator.
  - $100(1-p_c)\%$ of the population are simply copied to the new population.
- Single-point crossover operator
  - Choose a random site on the two parents
  - Split parents at this crossover site
  - Create children by exchanging tails

parent 1:	1 0 1 0 1 1|1 0 1 0
parent 2:	0 1 0 1 0 0|1 1 1 0
offspring 1:	1 0 1 0 1 1|1 1 1 0
offspring 2:	0 1 0 1 0 0|1 0 1 0

+

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 16 / 34

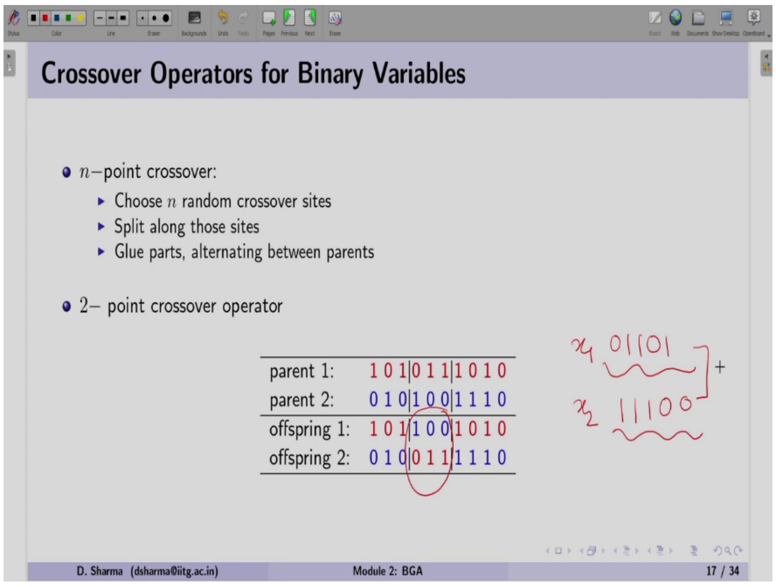
Now, we have discussed various kind of selection operator and we are free to choose any one of them, with the help of their limitations, we can make it out that which selection operator will be useful for us. Now, we are moving to the variation operators. In this class we will start with the crossover operator. Now, generally when we say crossover operator we use more than one parent. So, what is the basic difference between a crossover and a mutation? In crossover, we need more than one solution to create new solutions; however, in mutation we in mutation we need only one solution to generate another new solution.

So, in crossover we have a probability pc, which we referred to as crossover probability. Theoretically, if we are saying 100 times of pc that many strings will take part in the crossover and the rest of the string will be copied as it is. If you remember the case we discussed in the last session, there were a couple of strings that were just copied, because the random number was more than the probability of a crossover.

So, those particular strings were copied directly and rest of the strings based on the crossover and random number, we perform the crossover operator. In this category we have a different kind of crossover operators. So, let us begin with a single point. This crossover operator we already discussed. So, in this case what we do? We have for example, parent 1 and a parent 2, we pick the random site here, I am using the different colour coding, so, that we can understand.

And, we know that once we know the site we are going to swap the tail and with the help of colour coding you can make it out, that since we have red and a blue combination we are generating new solutions. So, those are referred to as offspring 1 and offspring 2.

(Refer Slide Time: 34:37)



Crossover Operators for Binary Variables

- n−point crossover:
  - Choose n random crossover sites
  - Split along those sites
  - Glue parts, alternating between parents
- 2− point crossover operator

parent 1:	1 0 1|0 1 1|1 0 1 0
parent 2:	0 1 0|1 0 0|1 1 1 0
offspring 1:	1 0 1|1 0 0|1 0 1 0
offspring 2:	0 1 0|0 1 1|1 1 1 0

x_1 01101
x_2 11100
+

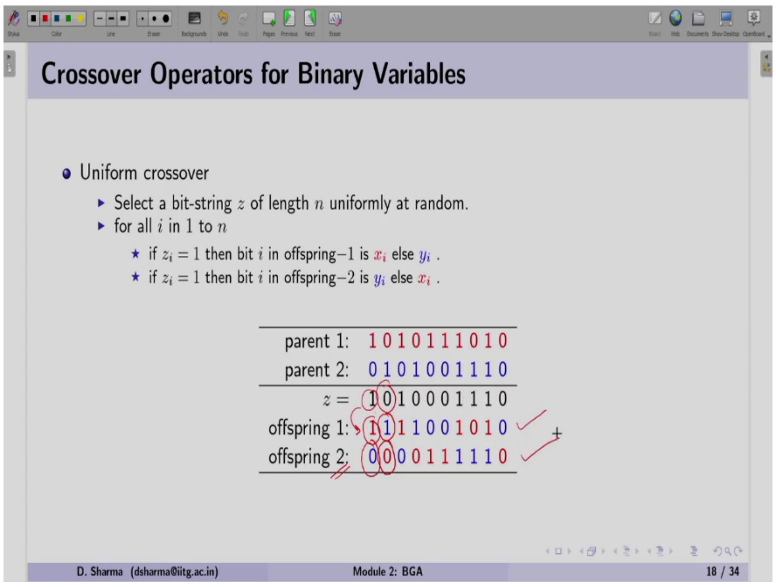
D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 17 / 34

Another kind of crossover operator we have is n-point crossover operator. We choose n random sites we split the site and then we glue some part and we exchange some part. Let us see how it works. So, here we will be discussing 2 point crossover operator. So, in this case we will be selecting two random sites. So, the first random site is given here and another random site is given here.

What we will do is we will glue the corners and the middle part will be swapped. You can see in the offspring 1 and 2 only this middle part is swapped and the tails and the front will remain the same. So, this crossover operator can generate the offspring in this way. Now, you can see a little difference with respect to the 1 point crossover. Now, using 1 point crossover operator, you may be changing 1 variable at a time, but using 2 point or 3 point crossover operator you may be changing multiple variables at a time.

Now, remember that we used suppose we have x 1 and x 2 we use some set of binary string says 0 1 1 0 1 and we have 1 1 1 0 0. So, both of these two strings when we combine them that makes a chromosome. So, we have a chromosome string of length 10. So, when we perform crossover using one. So, single point crossover maybe x 1 or x 2 will change, but with 2 point crossover, both variables can change at a time.

(Refer Slide Time: 36:37)



Crossover Operators for Binary Variables

- Uniform crossover
  - Select a bit-string z of length n uniformly at random.
  - for all i in 1 to n
    - if $z_i = 1$ then bit i in offspring−1 is x_i else y_i .
    - if $z_i = 1$ then bit i in offspring−2 is y_i else x_i .

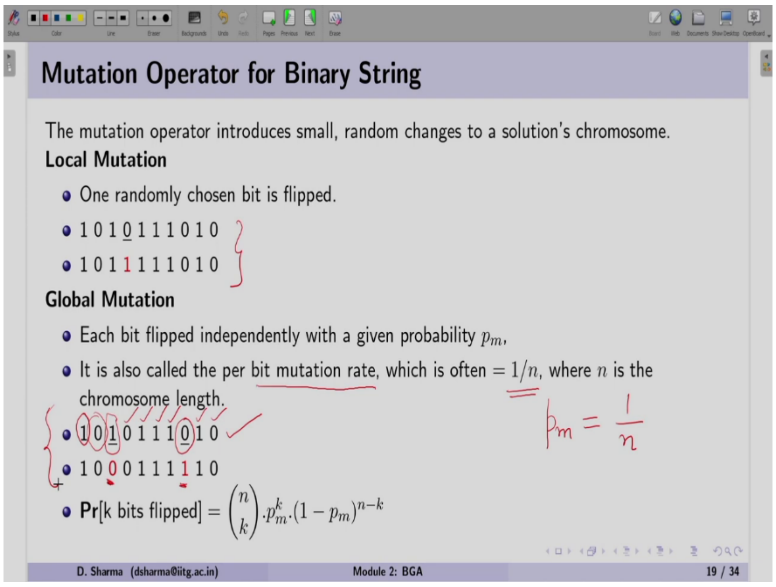
parent 1:	1 0 1 0 1 1 1 0 1 0
parent 2:	0 1 0 1 0 0 1 1 1 0
$z =$	1 0 1 0 0 0 1 1 1 0
offspring 1:	1 1 1 1 0 0 1 0 1 0
offspring 2:	0 0 0 0 1 1 1 1 1 0

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 18 / 34

There is another crossover operator, which is referred to as uniform crossover. In this crossover operator, we select a string z, another binary string generated randomly. Let us see how it works? Say, suppose you have parent 1 and parent 2 and we created this z string randomly. Now, what it says that suppose we have 1 here, then we will be selecting the bit from parent 1 for offspring 1. If it is 0 then we will be selecting a bit from parent 2 for offspring 1. So, let us see how?

So, in this case you can see 1, if it is 1 then the offspring 1 is get the bit from the parent 1, if it is 0, then it is getting from parent 2. Similarly, for offspring 2 it says that if you have 1 then the bit will be taken from parent 2, if it is 0 then the bit will be taken from parent 1 for offspring 2. So, in this way there is a total random swapping of the bits for 2 parents and that is going to create 2 new offspring's for us.

(Refer Slide Time: 38:07)



Mutation Operator for Binary String

The mutation operator introduces small, random changes to a solution's chromosome.

Local Mutation

- One randomly chosen bit is flipped.
- 1 0 1 $\underline{0}$ 1 1 1 0 1 0
- 1 0 1 1 1 1 1 0 1 0

Global Mutation

- Each bit flipped independently with a given probability p_m,
- It is also called the per bit mutation rate, which is often $= 1/n$, where n is the chromosome length.
- 1 0 $\underline{1}$ 0 1 1 1 $\underline{0}$ 1 0
- 1 0 0 0 1 1 1 1 1 0
- **Pr**[k bits flipped] $= \binom{n}{k} . p_m^k . (1 - p_m)^{n-k}$

$p_m = \frac{1}{n}$

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 19 / 34

Once, we perform the crossover operators we have different types of mutation operators. Now, mutation operator introduce small, random changes to a solution chromosome, it is because we perform mutation with a lower probability. Now, let us take the first mutation operator that is the local mutation operator.

This is what we studied or understood in the last session. So, here one random bit will be flipped. Flip means that, if it is 0, it is converted to 1 and 1 is converted into 0. Suppose, we have a binary string and randomly we pick the fourth position.

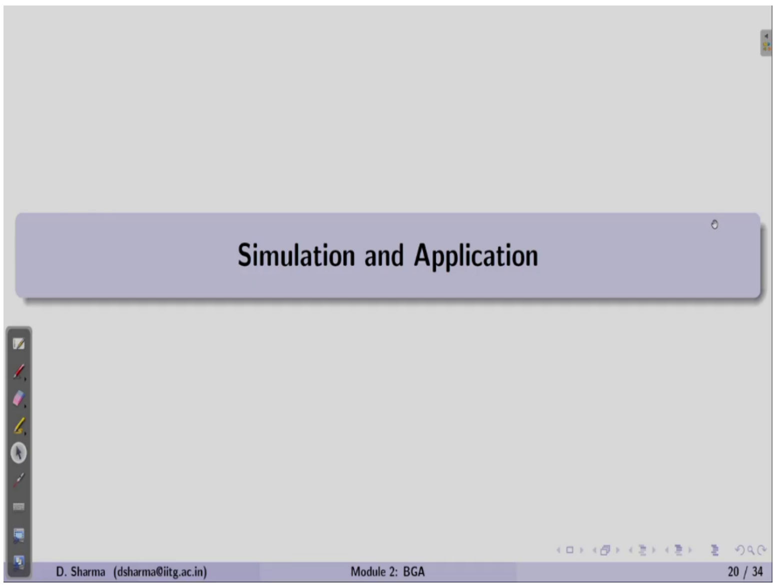
Now, in this case, we will convert this 0 to a 1, which is the local mutation. And, at a 1 time we are changing just 1 bit throughout the string. Another kind of mutation operator is global mutation operator. So, for each bit in this case we are going to take each bit in the binary string. So, each bit will be flipped independently with a given probability p m. It is also called as bit mutation rate and it is often take as a 1 by N.

Now, N is the length of the chromosome length. Suppose we have a binary string and for this particular bit we generated, we generated a random number. If, this random number is smaller than the probability of a mutation so, currently probability of mutation is 1 by N. If, this probability is if the random number generated is less than the probability we will perform the mutation.

So, in the first bit suppose the random number was more similarly for a second, but for the third the random number was smaller than the probability of a mutation. So, this is going to be so, this particular bit will be flipped. Similarly, we checked for all the bit 1 by 1 independently and suppose this is the bit, this is going to change and then the other one.

Now, you will realise that the flipped bits are changed. So, 1 has been changed to 0 and 0 is changed to 1. So, that makes the random change in a binary string. So, in the case local, in the case of local mutation, we are changing just 1 bit per string; however, in the case of global mutation, we are changing the bit as per the random number is smaller than the probability of mutation.

(Refer Slide Time: 41:13)



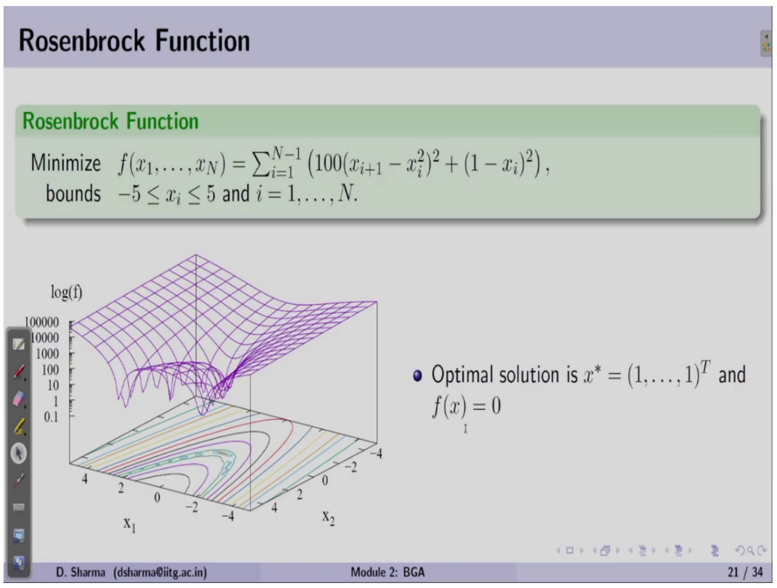
Simulation and Application

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 20 / 34

Now, let us start the simulation and application of binary coded GA. So, till now we have understood how binary coded GA works. And, there are a different kind of operators, which are available.

Now, the operators which we have discussed are limited in number, when you look into the literature there could be more types of crossover operator, mutation operator, and sometimes the selection operator. So, the researchers keep on changing the kind of operators that suits their problem. Now, let us begin with the simulations and application.

(Refer Slide Time: 41:57)



Rosenbrock Function

Rosenbrock Function

Minimize $f(x_1,\ldots,x_N)=\sum_{i=1}^{N-1}\left(100(x_{i+1}-x_i^2)^2+(1-x_i)^2\right)$,

bounds $-5 \le x_i \le 5$ and $i=1,\ldots,N$.

- Optimal solution is $x^*=(1,\ldots,1)^T$ and $f(x)=0$

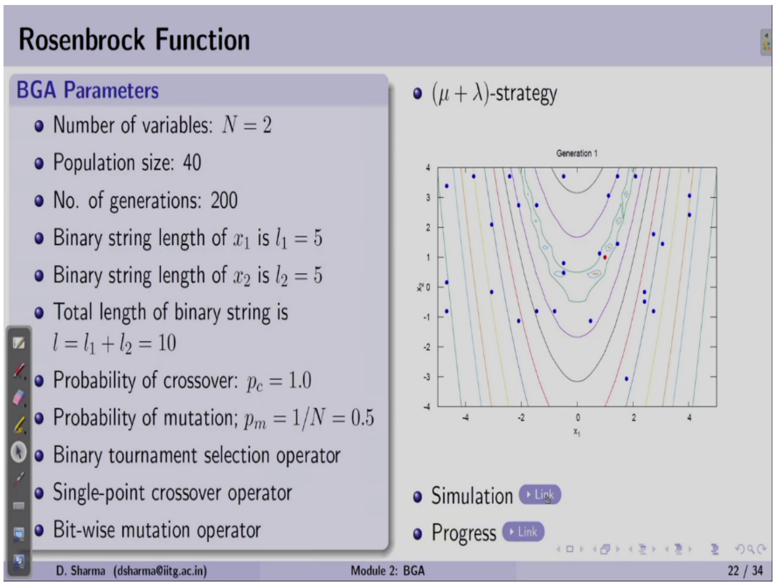
So, the first function we have taken is a Rosenbrock function. You remember that, this is the same function we have used to understand the working principle of binary coded GA. The Rosenbrock function can be written in terms of N number of variables and the formula is given here in the generalised way.

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

$$bounds\ -5 \le x_i \le 5\ \ and\ i = 1,\ \ldots,\ n.$$

All the variables will be lying between minus 5 to plus 5. In the figure which is shown on the left hand side, it is shown only for the 2 variables x 1 and x 2. So, the optimum solution for Rosenbrock function is that the solution will always have a value 1. So, all variables will get a 1 and the function value at the optimum solution is 0.

(Refer Slide Time: 42:55)



Rosenbrock Function

BGA Parameters

- Number of variables: $N = 2$
- Population size: 40
- No. of generations: 200
- Binary string length of x_1 is $l_1 = 5$
- Binary string length of x_2 is $l_2 = 5$
- Total length of binary string is $l = l_1 + l_2 = 10$
- Probability of crossover: $p_c = 1.0$
- Probability of mutation; $p_m = 1/N = 0.5$
- Binary tournament selection operator
- Single-point crossover operator
- Bit-wise mutation operator

- $(\mu + \lambda)$-strategy

- Simulation Link
- Progress Link

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 22 / 34

We have used binary coded GA. So, in the generalised formulation in this step 2 there are a lot of input parameters that we have to set. So, this particular slide will tell how many parameters we have to fix before we start the binary coded GA.

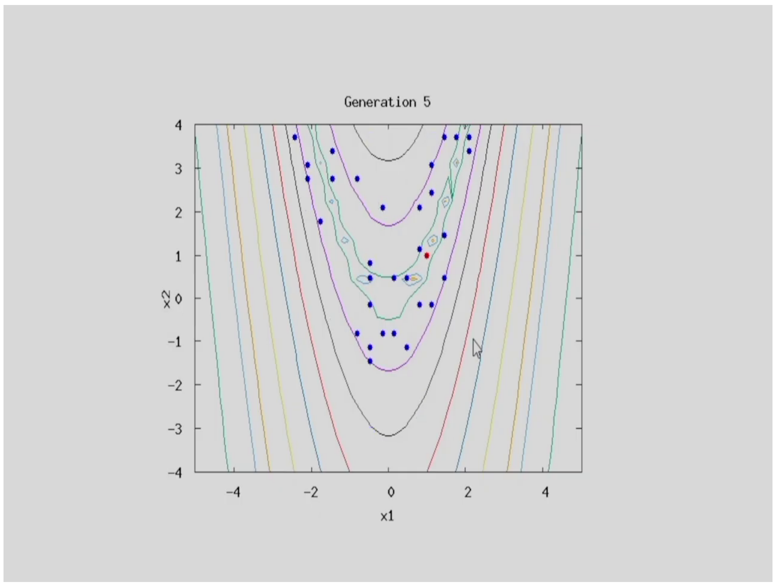
Let us take an example of 2 variable Rosenbrock function, the population size is set as 40, and we allow a large number of a generation to see the progress. So, here we have set the number of maximum the maximum number of generation 200. Currently we pick x 1 so, binary string length for x 1 is 5, binary string length for x 2 is also 5 and the total length of binary string is 10. So, 5 plus 5 will make 10.

Since, Rosenbrock function has many local optimum solution. So, we are keeping the probability of crossover high which is 1. The probability of a mutation we have taken 1 by N should be small n because the small n tells the number of variables. So, in this case since the number of variables is 2.

So, this will become 0.5. Binary tournament selection operator is used for selection before variation operator, thereafter we perform single point crossover operator, for a mutation we perform bitwise mutation. And, finally, for selecting good solutions from parent and offspring population, we are taking the strategy mu plus lambda strategy.

So, let us see how this binary coded GA work here. So, here in the picture you can see this is generation 1. So, all the blue dots are the randomly generated solution in the plane of x 1 and x 2. So, we make sure that, the solution should generate within the range of x 1 and x 2. Now, let us see, how this simulation work.

(Refer Slide Time: 45:17)

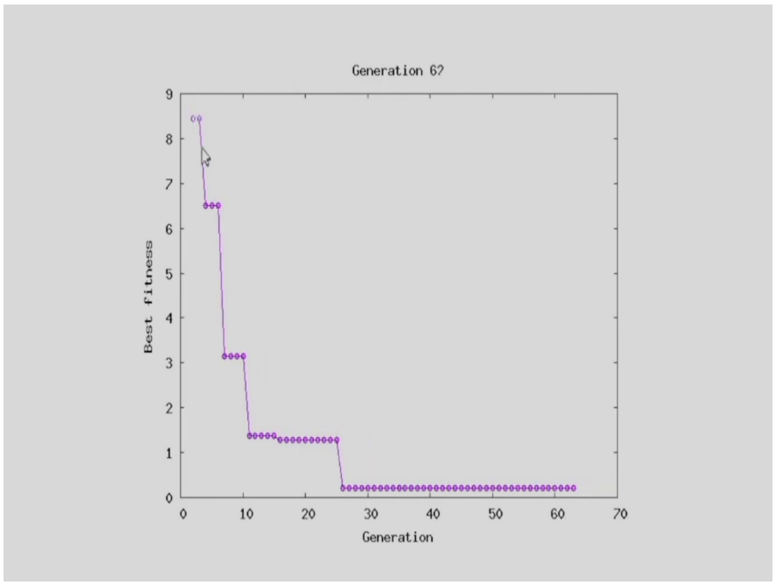


Here, you can see that solutions were randomly distributed and with the number of generations these solutions now moving towards the optima. So, the red dot here is the optimum solution, and there are many local optimum solutions. Now, at this stage you can see that the solutions are not improving much. It is because we have taken a small binary string length for x 1 and x 2.

Now, since these solutions are here and there are small changes, you can observe no change. It is because crossover is not generating any good or any new solution. It's only the mutation operator that will help us to generate a solution close to the red dot.

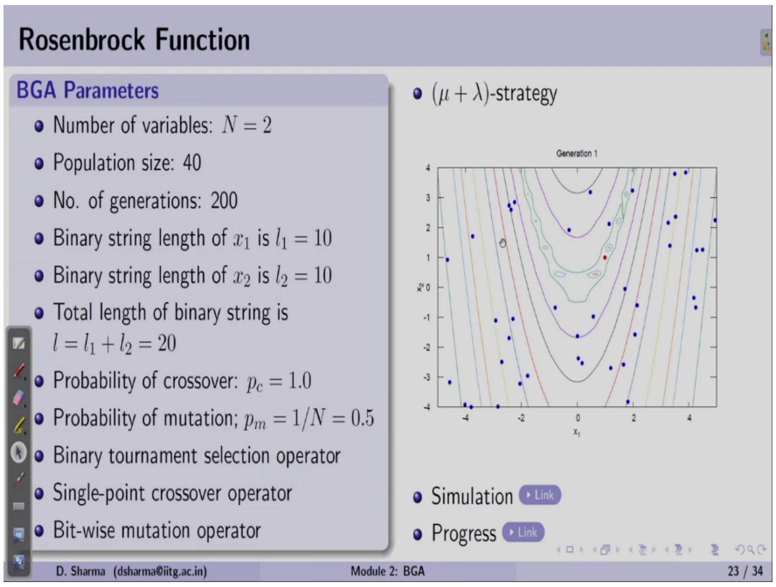
So, let us wait for the generation number 200 and see how close we can reach. So, this is the blue point that is closest and that remains the same. Now, let us look at the progress. So, the progress means that the x axis will be the number of generation and y axis is the best fitness.

(Refer Slide Time: 46:37)



Now, in this case you can see the improvement in the solution, it is drastic, drastically it reduces in the early generations and after that say after say 23 generation, the improvement there is no improvement in the fitness, so, the best fitness. We have observed this and the same situation when we shown the convergence plot for one dimensional landscape. The GA will improve its fitness very quickly; however, it will take some time to converge to the optimum solution.

(Refer Slide Time: 47:21)



Rosenbrock Function

BGA Parameters

- Number of variables: $N = 2$
- Population size: 40
- No. of generations: 200
- Binary string length of x_1 is $l_1 = 10$
- Binary string length of x_2 is $l_2 = 10$
- Total length of binary string is $l = l_1 + l_2 = 20$
- Probability of crossover: $p_c = 1.0$
- Probability of mutation; $p_m = 1/N = 0.5$
- Binary tournament selection operator
- Single-point crossover operator
- Bit-wise mutation operator

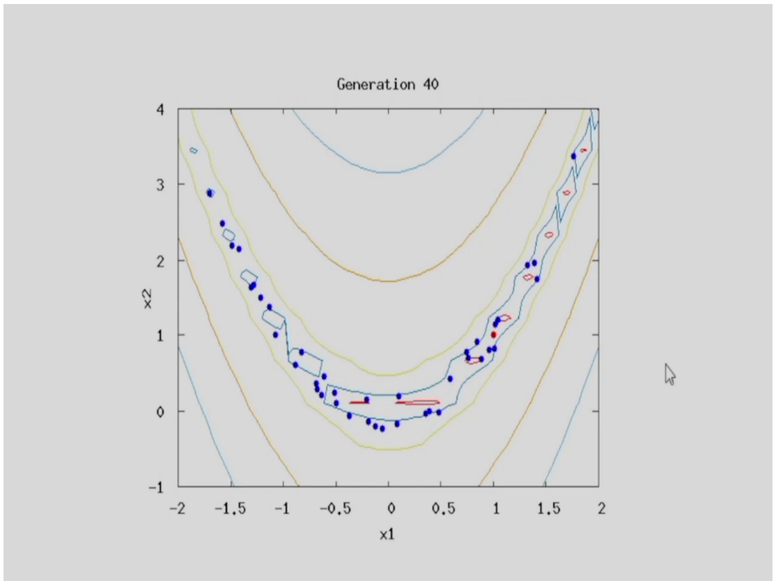
- $(\mu + \lambda)$-strategy
- Simulation Link
- Progress Link

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 23 / 34

Let us take another case now here the parent the number of variables remains 2, population size remains 40, number of a generation 200, binary string has been changed now. Now, you remember that if we are going to take a larger size of a binary string, then the precision of x 1 will be higher. So, for x 1 we are taking binary string length of 10, for x 2 the binary string of length 10 is taken so, total is 20.

The crossover probability is kept the same: 1, mutation probability 0.5 and we use the same set of operators, such as binary tournament selection operator, single point crossover, bitwise mutation, and mu plus lambda strategy. For this case we generated a random population. This population of these members are different from the previous case. So, these are generated in a plane of x 1 and x 2, let us see whether changing the l 1 and l 2 the binary string will help us to improve this solution or not.

(Refer Slide Time: 48:39)

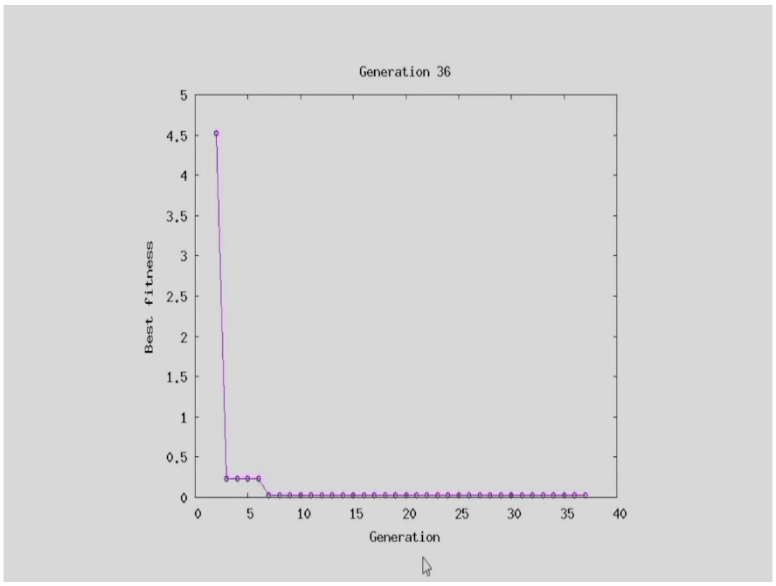


Now, these are the simulations. So, the solutions were generated randomly throughout the range of x 1 and x 2 in few iteration, that is less than 25 generations, the solutions are already close to the optimum. Now, the distribution of the solutions you can see, because there are lot of local minima’s, but there are certain solution as you can see that these solutions are quite close to the red dot which is the optimum solution.

Now, one particular solution has already reached very close to the optimum solution somewhere in the 90 generation. And, let us see that in 200 generation can we get the optimum solution. Now, you can see another solution is generated close to the optimum and rest of the solutions are also improving. And, slowly and slowly these solutions are moving, it is because we have selection operators.

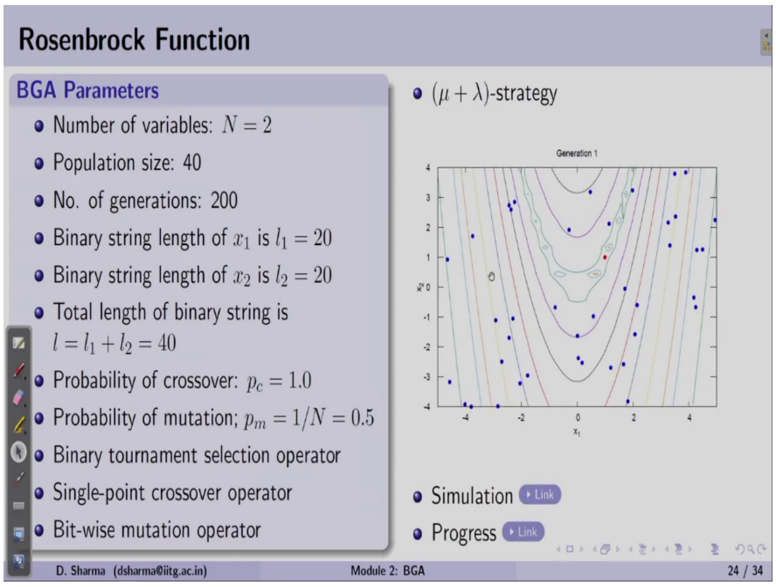
Now, the solution since these are moving, we expect that at the end 1 of the solution should generate. So, now, you can see that we have increased the string length for x 1 and x 2 from 5 to 10, the solution has no solution that has converged to the optimum solution.

(Refer Slide Time: 50:05)



Let us see the progress, in this progress the number of generation versus best fitness. You can see, how quickly GA binary coded GA has reached the value close to 0. And, then there is a and then the best fitness remains the same, the other solutions are improving. Somewhere at 150 generation there was a improvement in the fitness. So, let us see one more time. So, as close to 150, we will see little improvement yes. So, here somewhere close to 150, we have a little improvement in the fitness.

(Refer Slide Time: 50:43)



Rosenbrock Function

BGA Parameters

- Number of variables: $N = 2$
- Population size: 40
- No. of generations: 200
- Binary string length of x_1 is $l_1 = 20$
- Binary string length of x_2 is $l_2 = 20$
- Total length of binary string is $l = l_1 + l_2 = 40$
- Probability of crossover: $p_c = 1.0$
- Probability of mutation; $p_m = 1/N = 0.5$
- Binary tournament selection operator
- Single-point crossover operator
- Bit-wise mutation operator

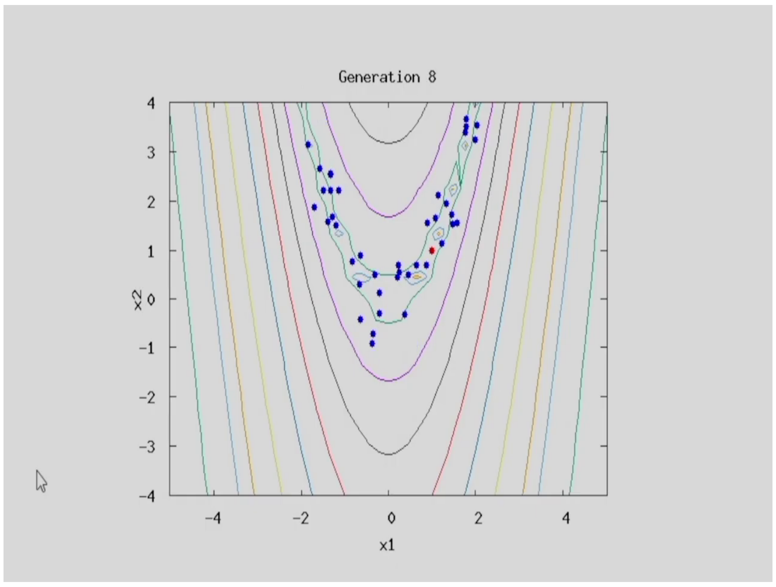
- $(\mu + \lambda)$-strategy
- Simulation Link
- Progress Link

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 24 / 34

Since we did not get the optimum solution yet, we increase the length of the binary string for x 1 and x 2. In this case the binary string length for x 1 is 20 and binary string length for x 2 is 20.

So, total chromosome string length is 40 now. We have taken the same 2 number of variables population size number of generations are kept same, similarly the probabilities and the operators were also kept the same as before. So, this is the random population generated, and we will see the simulation for the larger binary string.

(Refer Slide Time: 51:27)

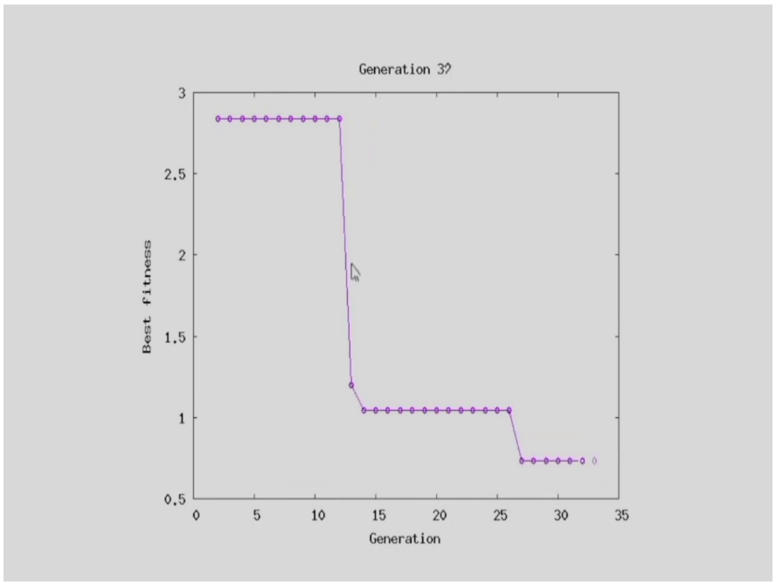


Now, the solutions are moving quickly close to the optimum solutions and then in 25 generations, we are already close to the optimum solution. Now, we have to see whether binary coded j can generate a solution on the red dot meaning that, binary coded j can find an optimum solution for this problem when length of the string is increased.

Now, this blue point as you can see which is quite close to the red dot which we have identified or the binary GA has found in say 80th generation. Other solutions are also moving now the there are two solutions, which are close to the red dot and we have to see whether B GA can find a solution for us.

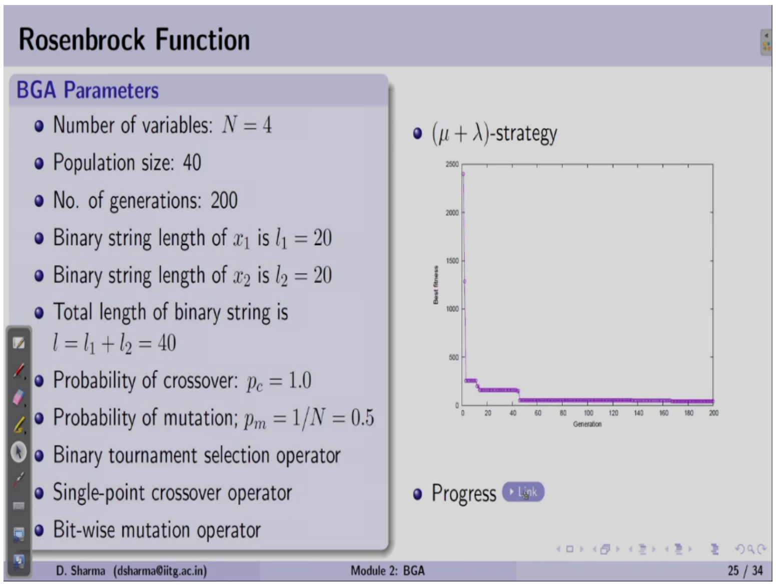
Now, slowly the other solutions are also moving from other directions, now you have more solutions so, many solutions surround this red dot. So, from this simulation, we have identified that since we are increasing the binary string length in the binary string, increasing the binary string length means we are increasing the precision for x 1 and x 2. The solutions are going closer to the red dot, but it has still not reached the optimum solution.

(Refer Slide Time: 52:59)



Let us see, the progress now. So, we have a drastic improvement here and with the number of a regeneration the best fitness in the population is keep on improving. So, around just before 70 iterations, we have found that the best fitness remains the same and after few more generations let see one more time here. So, somewhere close to 100 and 50, we generated. So, B GA generated a solution quite close to the optimum solution.

(Refer Slide Time: 53:39)


Rosenbrock Function

BGA Parameters

- Number of variables: $N = 4$
- Population size: 40
- No. of generations: 200
- Binary string length of x_1 is $l_1 = 20$
- Binary string length of x_2 is $l_2 = 20$
- Total length of binary string is $l = l_1 + l_2 = 40$
- Probability of crossover: $p_c = 1.0$
- Probability of mutation; $p_m = 1/N = 0.5$
- Binary tournament selection operator
- Single-point crossover operator
- Bit-wise mutation operator

- $(\mu + \lambda)$-strategy
- Progress

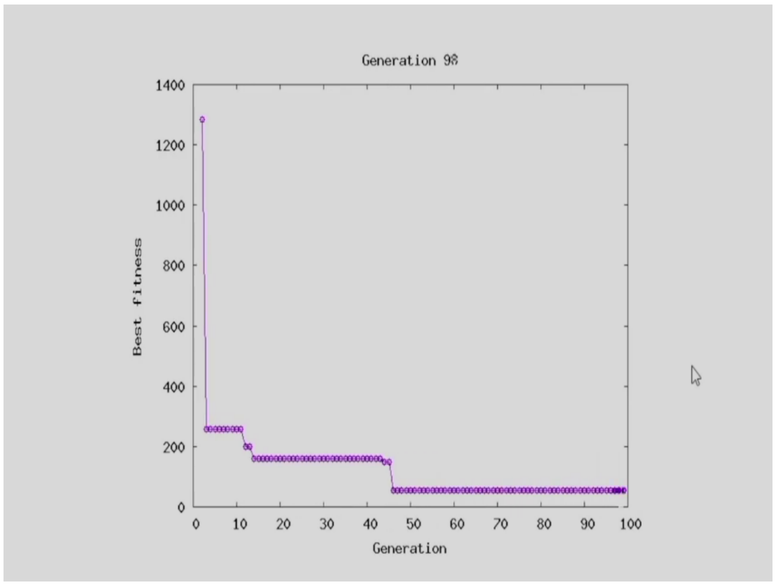
D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 25 / 34

Till now, what we have solved for Rosenbrock function is 2 variable problem. If, we increase the number of a variable, we will see how the performance of binary coded GA will change. In this case the population size number of a generations are kept same. Since, we want a more precise binary string length of 20 and 20 is used. So, please note that here, we have shown for x 1 and x 2. Similarly, for x 3 and x 4 we have used 20 and 20.

So, overall binary string length or the chromosome string length is 80 not 40. So, this is it is 80 for this particular problem. So, probability is kept 1, probability of a mutation is again we kept it 0.5. The operators are binary tournament selection operator, single point crossover, bitwise mutation and mu plus lambda strategy. So, since it is a 4 variable problem, so, we cannot show the simulation how the solutions are moving towards the optimum, but we can see the progress.

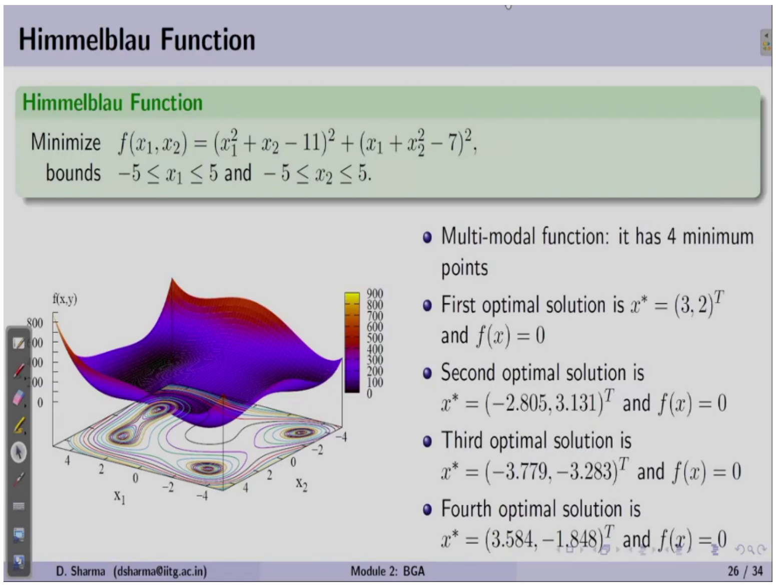
So, this particular plot shows that, there is a drastic change in the fitness value and then of close to say 45 generation, the improvement is very less. So, we will see this particular progress now.

(Refer Slide Time: 55:15)



So, here the improvement can be seen in the early generation of binary coded GA. And, once it is reaches 45 the improvement is there is no improvement. Then, somewhere close to 140 there is a little improvement close to 170 also we have a little improvement. Important point to note that, that is still we have run this binary coded GA for 200 generation, this algorithm does not find the optimum solution.

(Refer Slide Time: 55:53)



Himmelblau Function

Himmelblau Function

Minimize $f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$,
bounds $-5 \le x_1 \le 5$ and $-5 \le x_2 \le 5$.

- Multi-modal function: it has 4 minimum points
- First optimal solution is $x^* = (3, 2)^T$ and $f(x) = 0$
- Second optimal solution is $x^* = (-2.805, 3.131)^T$ and $f(x) = 0$
- Third optimal solution is $x^* = (-3.779, -3.283)^T$ and $f(x) = 0$
- Fourth optimal solution is $x^* = (3.584, -1.848)^T$ and $f(x) = 0$

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 26 / 34

Let us move to the second problem which is Himmelblau function, this Himmelblau function is made of 2 variables.

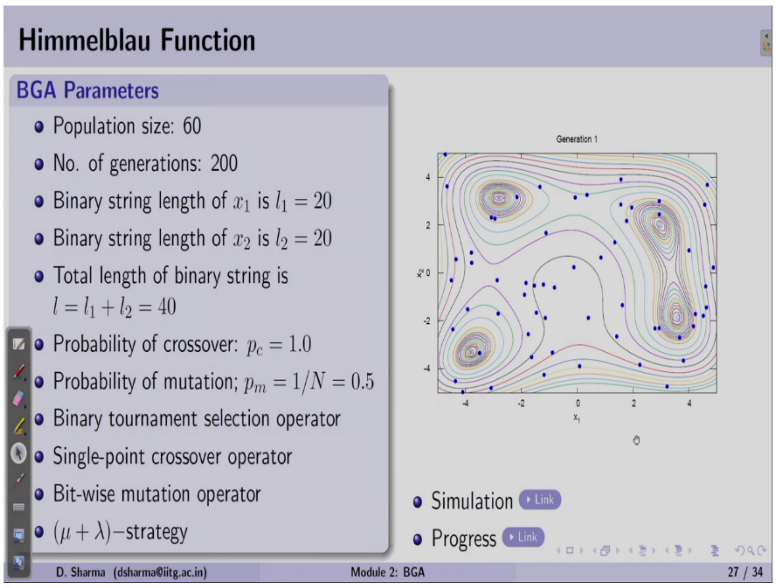
$$Minimize\ f(x_1, x_2) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2$$

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

Now, from the equations you can find that this is a; that this is a non-linear function. The value of x 1 and x 2 are lying between minus 5 to plus 5. So, let us look at the Himmelblau function. So, the surface you can see and when we draw the contour of this Himmelblau function, you can see there are 4 areas where the minimise line. So, the Himmelblau function is a multi modal function as I have mentioned on the right hand side, it has 4 optimum solutions.

So, all the 4 optimum solutions are also mentioned. So, if one is at 3, 2, another is at minus 2.8, 3.1, then another solution is at minus 3.78, and minus 3.28, and the last solution is at 3.58, and minus 1.848. And, for all these 4 solution the function value is 0.

(Refer Slide Time: 57:07)


Himmelblau Function

BGA Parameters

- Population size: 60
- No. of generations: 200
- Binary string length of x_1 is $l_1 = 20$
- Binary string length of x_2 is $l_2 = 20$
- Total length of binary string is $l = l_1 + l_2 = 40$
- Probability of crossover: $p_c = 1.0$
- Probability of mutation; $p_m = 1/N = 0.5$
- Binary tournament selection operator
- Single-point crossover operator
- Bit-wise mutation operator
- $(\mu + \lambda)$–strategy

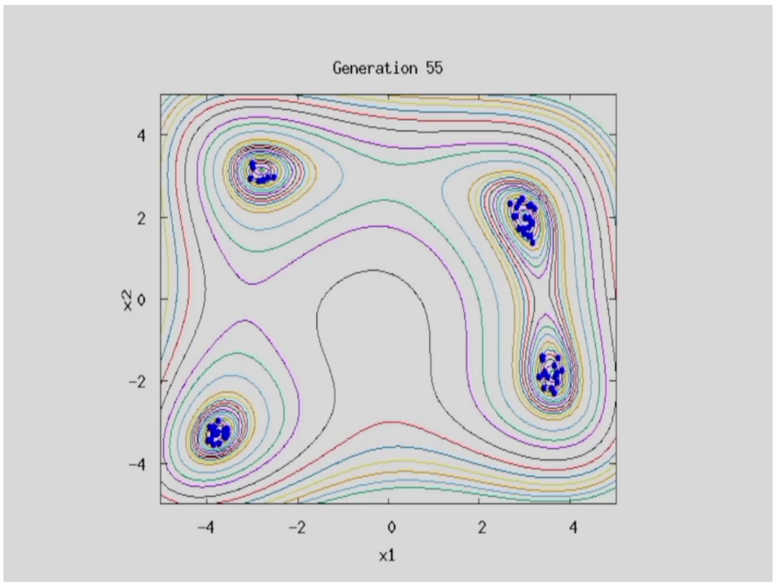
- Simulation Link
- Progress Link

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 27 / 34

Now, we are using the binary coded GA, since it is a multi modal problem, we are keeping the population size 60. The number of generation is kept little high which is 200, to have more precision we are taking x 1 binary string length for x 1, 20 and binary string length for x 2 is also 20. So, overall chromosome length is 40, probability of a crossover is 1 and probability of a mutation is 0.5.

We perform the same set of operators such as binary tournament selection operator, single point crossover operator, bitwise mutation and mu plus lambda strategy. Now, this is the first generation in which the solutions are generated randomly throughout the range of x 1 and x 2. So, let us look at the simulation now.

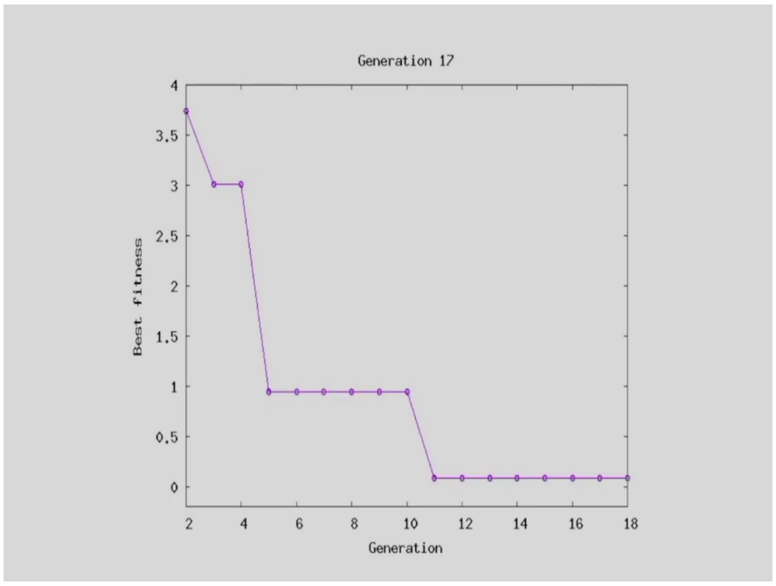
(Refer Slide Time: 58:07)



So, the simulation shows the solutions are divided into all the 4 optimum solutions, around the all 4 optimum solution and slowly and slowly they are converging to the optimum solution. So, in the in during the iterations, the solutions are distributed so that they got all 4 optimum solutions in case of binary coded GA.

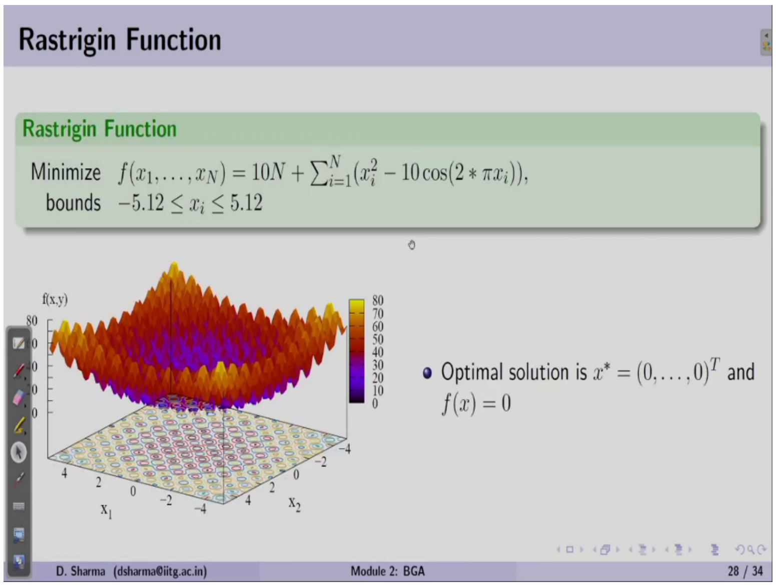
Please note that we got the 4 optimum solution with this simulation, but it is not necessary, because to have the solutions distributed among the peaks, we should use some methods called fitness sharing.

(Refer Slide Time: 59:01)



Let us see, the progress of the simulation on Himmelblau function. There is a drastic improvement and close to 25 generations, we have reached the optimum solution. And, then since we have reached it so, there is no improvement. So, it can be seen that, even though it was a multimodal problem, the binary coded GA was able to solve this problem quickly and generated the optimum solution in very few generations.

(Refer Slide Time: 59:31)



Rastrigin Function

Rastrigin Function

Minimize $f(x_1, \ldots, x_N) = 10N + \sum_{i=1}^{N}(x_i^2 - 10\cos(2 * \pi x_i))$,
bounds $-5.12 \leq x_i \leq 5.12$

- Optimal solution is $x^* = (0, \ldots, 0)^T$ and $f(x) = 0$

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 28 / 34

The third example which we have taken is the Rastrigin function. This function can be represented in N number of variables.

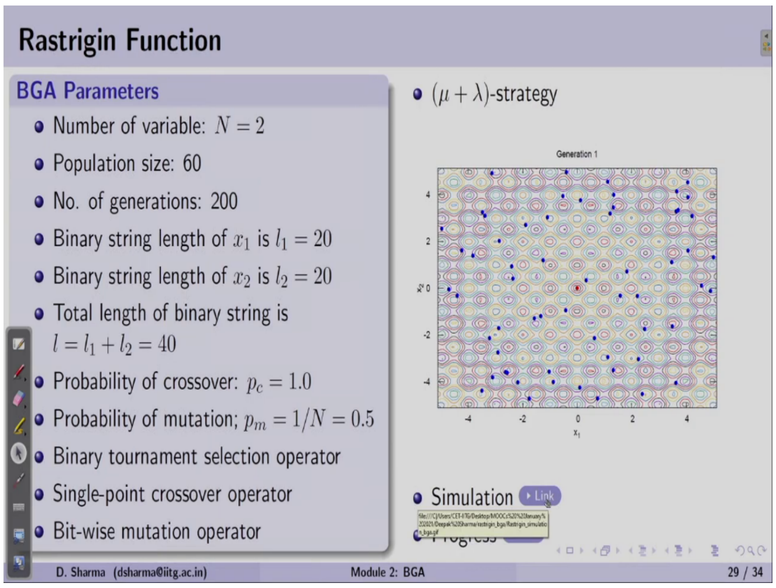
$$Minimize\ f(x_1, ..., x_n) = 10N + \sum_{i=1}^{n} \left(x_i^2 - 10cos(2 * \pi x_i)\right)$$

$$bounds\ -5.12 \le x_i \le 5.12$$

Now, since these terms are added into the objective function, this will create many local minima's. All the variables are lying between minus 1 point minus 5.12 to plus 5.12. Now, for 2 variable Rastrigin problem, you can see the surface with many ups and downs or peaks or valleys. And, if you see the contours at the bottom, you can see many local optimum solutions.

Now, the optimum solution for this problem is lying at the origin and the function value is 0.

(Refer Slide Time: 60:19)



Rastrigin Function

BGA Parameters

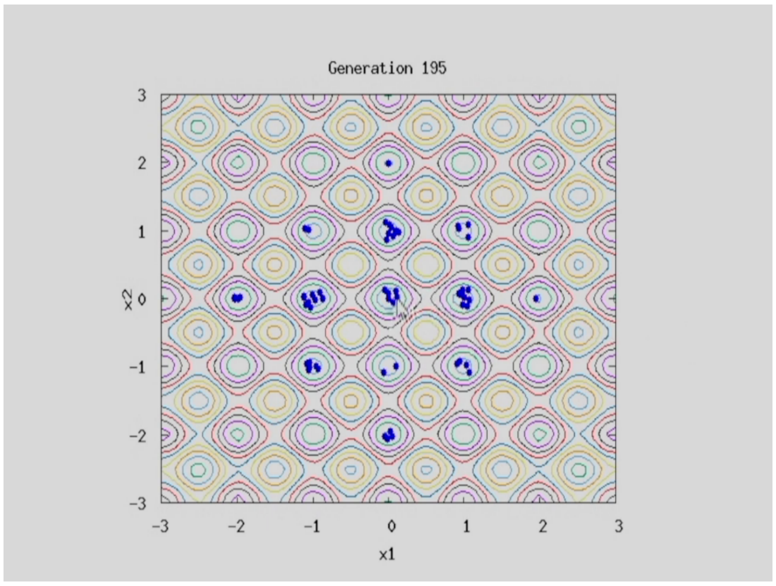
- Number of variable: $N = 2$
- Population size: 60
- No. of generations: 200
- Binary string length of x_1 is $l_1 = 20$
- Binary string length of x_2 is $l_2 = 20$
- Total length of binary string is $l = l_1 + l_2 = 40$
- Probability of crossover: $p_c = 1.0$
- Probability of mutation; $p_m = 1/N = 0.5$
- Binary tournament selection operator
- Single-point crossover operator
- Bit-wise mutation operator

- $(\mu + \lambda)$-strategy
- Simulation Link
- Progress

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 29 / 34

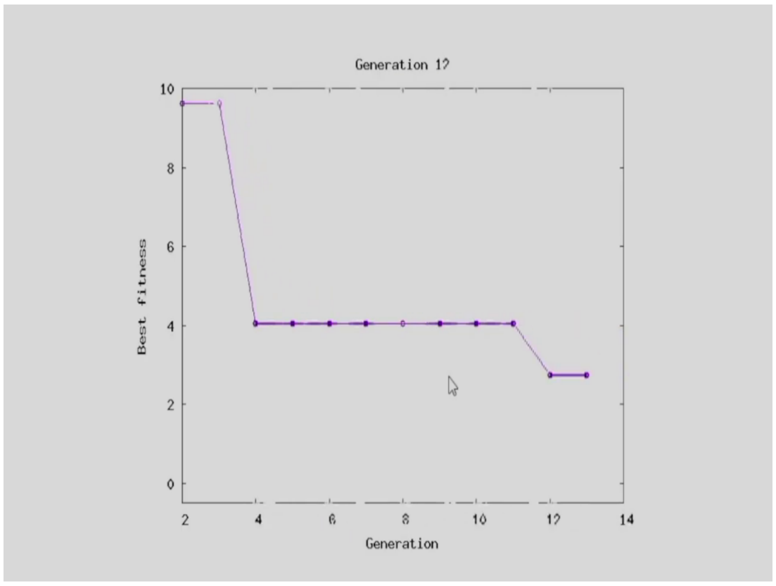
Now, the Rastrigin function here, since we have a 2 variable, we are taking the 2 variable problem, we keep the population size 60, number of generation 200, the length for x 1 and x 2 20 and a 20. So, the chromosome string is 40, probability of crossover is 1, probability of mutation is 0.5 and the same set of operators we have used it. This figure shows that the solutions are randomly distributed in x 1 and x 2 plane. So, let us see the simulation for this problem.

(Refer Slide Time: 61:01)



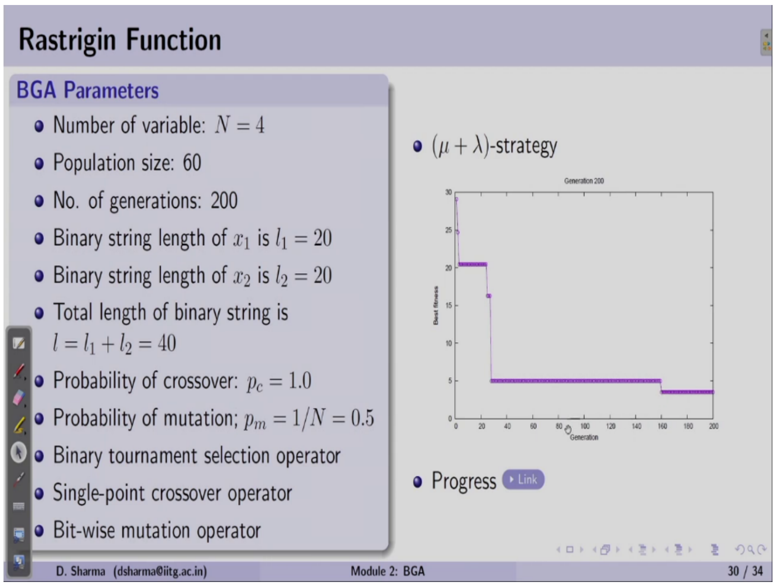
Now, you can see the solutions that are distributed now moving towards the optimum, not only optimum to the local optimum solutions. Now, in the focused view you see that the solution, one solution was created close to the optimum and eventually, the other solutions are also moving to the optimum. Now, we cannot see the red dot, it means that, the optimum solution for the Rastrigin function is found by the binary coded GA.

(Refer Slide Time: 61:35)



Let us see the progress. Now in this case again there is a improvement in the fitness value with the number of a generations and in 60, less than 60 generation we are quite close to the optimum and then we are searching. And, finally, at just before 100 and 80 generation, we got the solution, which converge to the optimum solution for the Rastrigin function.

(Refer Slide Time: 62:03)



Rastrigin Function

BGA Parameters

- Number of variable: $N = 4$
- Population size: 60
- No. of generations: 200
- Binary string length of x_1 is $l_1 = 20$
- Binary string length of x_2 is $l_2 = 20$
- Total length of binary string is $l = l_1 + l_2 = 40$
- Probability of crossover: $p_c = 1.0$
- Probability of mutation; $p_m = 1/N = 0.5$
- Binary tournament selection operator
- Single-point crossover operator
- Bit-wise mutation operator

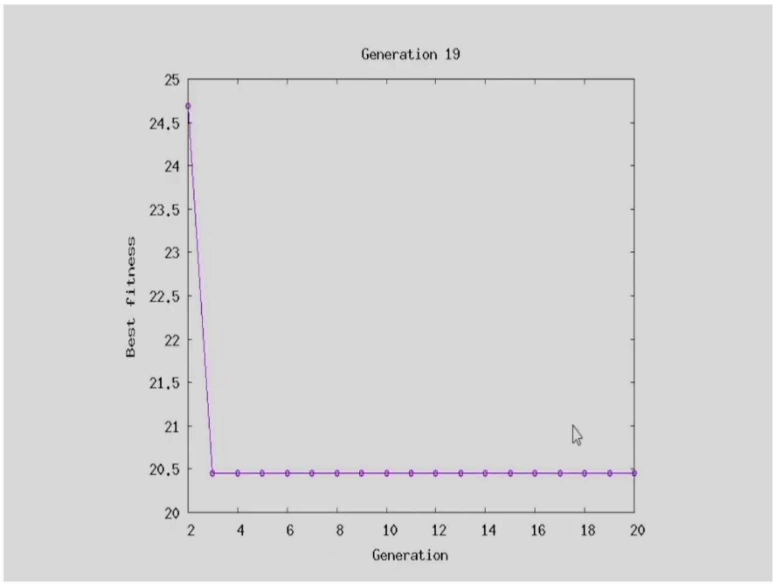
- $(\mu + \lambda)$-strategy
- Progress Link

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 30 / 34

We will take another case of Rastrigin function, where the number of variable is 4 we take all the parameters here please note that, we have written the binary string length of x 1 and x 2, since there are 4 variable. So, we are taking 20 binary string length for all. So, the chromosome string, it is not 40. So, the chromosome string will become 80 here. Probability of crossover and mutation are given and other operators are the same as before.

Since, it's a 4 variable problem we will be showing the progress. Now, from this particular figure you can see, there is a drastic improvement less than 30 generations and then the fitness remains the same and then there is a last improvement somewhere close to 160 generation.

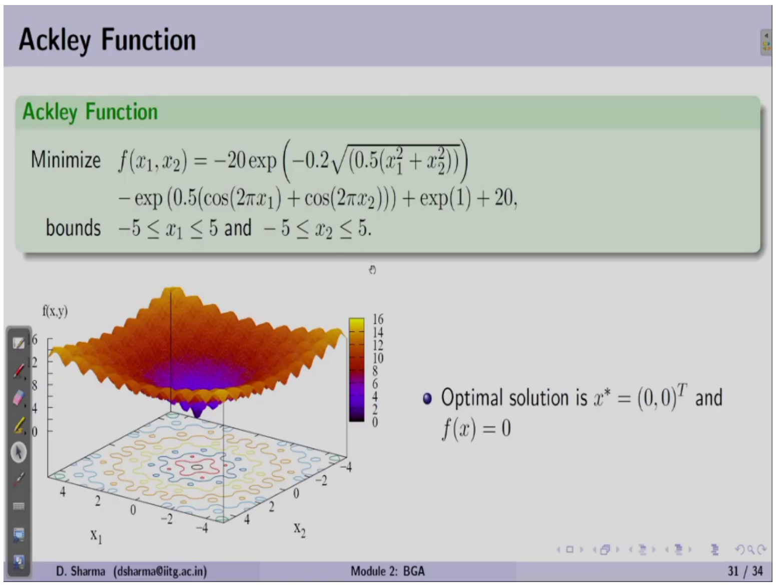
(Refer Slide Time: 62:59)



So, let us see the progress, while solving this problem, Rastrigin problem with 4 variables. So, the improvement can be seen once it is reaches 30, there is no improvement and at around close to 160 we get another improvement.

Note that, the optimum solution for this problem is 0 0 and the function value should be 0 and after 200 generation is still binary coded GA is unable to find the optimum solution in 200 generation. So, we can try, if suppose we increase the number of generation and population size we may solve this problem using binary coded GA.

(Refer Slide Time: 63:47)



Ackley Function

Ackley Function

Minimize $f(x_1,x_2) = -20\exp\left(-0.2\sqrt{(0.5(x_1^2+x_2^2))}\right)$
$- \exp\left(0.5(\cos(2\pi x_1)+\cos(2\pi x_2))\right) + \exp(1) + 20,$
bounds $-5 \le x_1 \le 5$ and $-5 \le x_2 \le 5$.

- Optimal solution is $x^* = (0,0)^T$ and $f(x) = 0$

f(x,y)
x_1
x_2
16
14
12
10
8
6
4
2
0

Come to the last problem which is Ackley function, it is written in the 2 variables as x 1 and x 2.

$$Minimize\ f\left(x_1, x_2\right) = -20\ exp\left(-0.2\sqrt{\left(0.5\left(x_1^2 + x_2^2\right)\right)}\right)$$

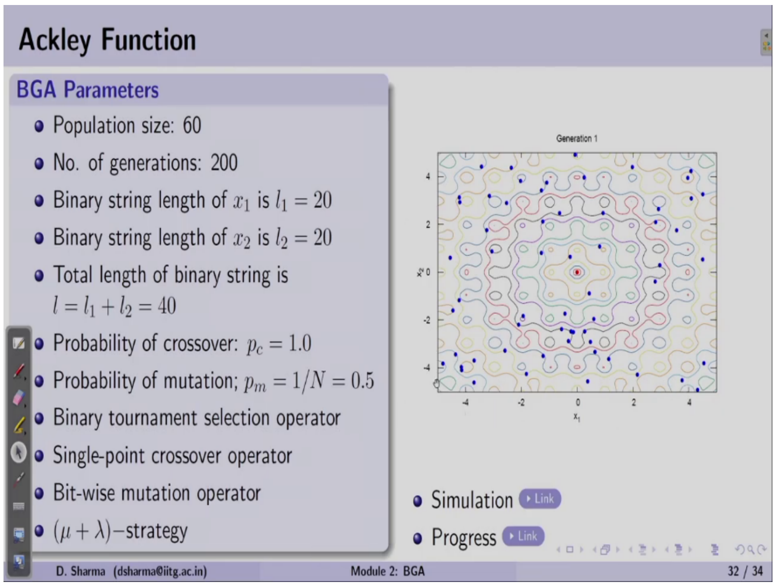
$$-\ exp\left(0.5\left(cos\left(2\pi x_1\right) + cos\left(2\pi x_2\right)\right)\right) + exp(1) + 20$$

$$bounds\quad -5 \le x_1 \le 5\ \ and\ \ -5 \le x_2 \le 5$$

And, its a little complex function and the value the upper and lower limit of x 1 and x 2 is lying from minus 5 to plus 5. Looking at the surface you can make it out, that this wavy kind of a surface will create a lot of local optimum solutions.

Now, in this contours we on the x in the contour of x 1 and x 2, you can make various local optimum solutions. For the Ackley's problem the optimum solution lies at the origin and the function value is 0.

(Refer Slide Time: 64:29)



Ackley Function

BGA Parameters

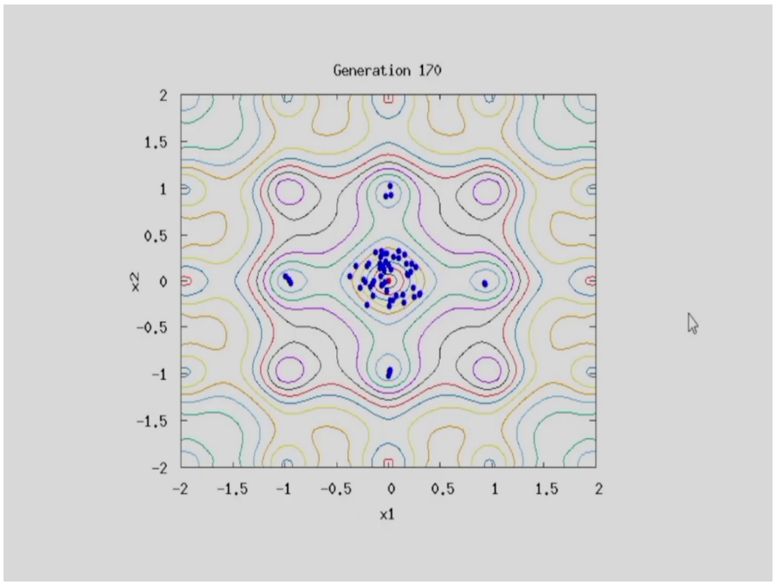
- Population size: 60
- No. of generations: 200
- Binary string length of x_1 is $l_1 = 20$
- Binary string length of x_2 is $l_2 = 20$
- Total length of binary string is $l = l_1 + l_2 = 40$
- Probability of crossover: $p_c = 1.0$
- Probability of mutation; $p_m = 1/N = 0.5$
- Binary tournament selection operator
- Single-point crossover operator
- Bit-wise mutation operator
- $(\mu + \lambda)$–strategy

- Simulation Link
- Progress Link

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 32 / 34

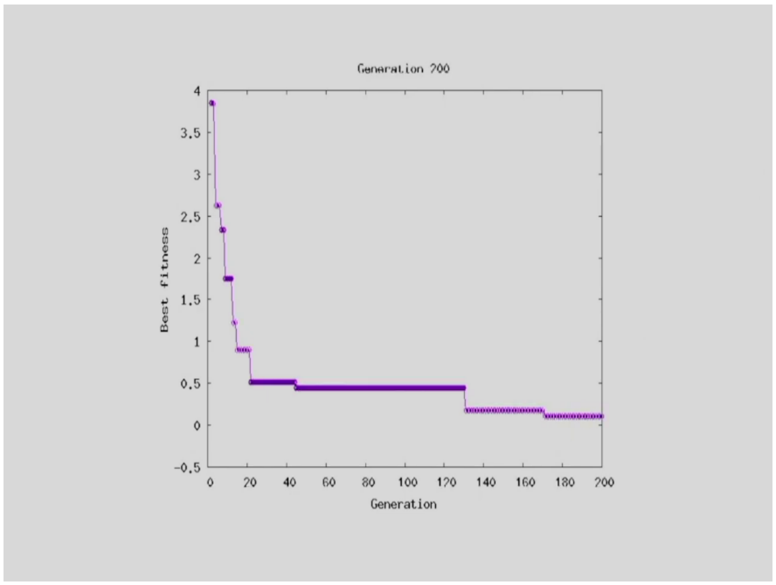
Solving this problem the parameters of BGA is parameters remain the same. So, we kept population size 60, number of generation 200, binary string length for both the variables 20. So, the chromosome string will become 40, probability of crossover and mutation are mentioned and the same set of operators are used. So, in generation 1 the solutions are distributed as you can see in the x 1 and x 2 plane.

(Refer Slide Time: 65:05)



So, let us look at the simulation now. Now, the solutions are moving towards the optima they are also stuck at the local minima. Now, in 20 generation itself the solution have reached to the reach to the reach closer to the optimum solution. And, we have to see whether binary coded GA can solve Ackley's problem or not. Now, see that the blue point is already on the red dot.

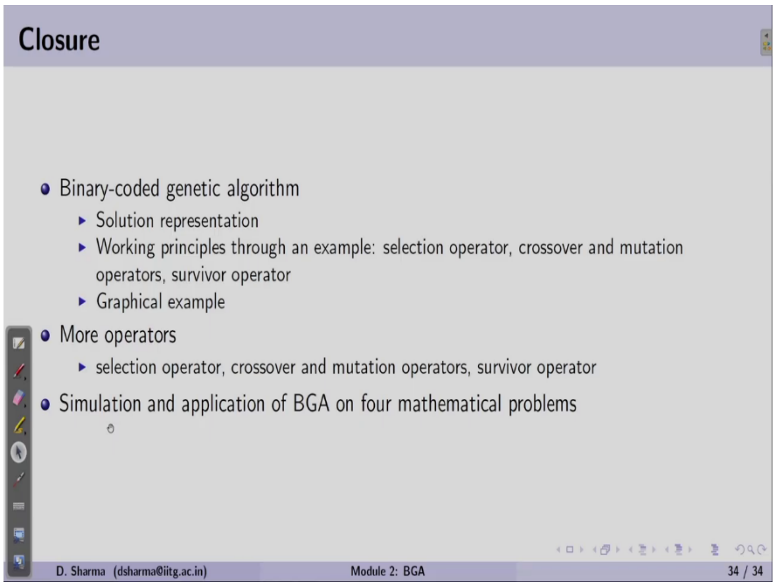
(Refer Slide Time: 65:33)



Let us see the progress in this case. So, there is continuous improvement in the best fitness in the population and that is improving and after 40 generation, there is no improvement in the fitness. Because, the solutions are generating the selection is performing crossover and the mutation.

And, somewhere close to 130, there was improvement in the population, which is why the best fitness was improved. So, let us see one more time. So, closed at 100 and yes 130, the solution has improved and there is a little improvement after 170 generation.

(Refer Slide Time: 66:15)



Closure

- Binary-coded genetic algorithm
  - Solution representation
  - Working principles through an example: selection operator, crossover and mutation operators, survivor operator
  - Graphical example
- More operators
  - selection operator, crossover and mutation operators, survivor operator
- Simulation and application of BGA on four mathematical problems

D. Sharma (dsharma@iitg.ac.in) Module 2: BGA 34 / 34

So, we have come to the closure of module 2, which is based on the binary coded GA. So, in this module we have understood the algorithm and we have gone through the example by following the generalised framework. So, we started this binary coded GA, while showing the solution representation in that one the binary string was used to represent the real variable.

The working principle of binary coded GA was shown with an example of Rosenbrock function and in that example we have evaluated the function. So, first the binary string is decoded and x 1 and x 2 values were calculated, then fit after calculating the fitness selection was made crossover and mutation were performed. And, finally, the survival stage was used to select the best solution.

So, the same hand calculations were shown in the graphical example. We discussed more operators in this session, especially we target selection operator, crossover operator, and mutation operator. Since, they are they work differently, their performance also will be different when we solve a problem.

Finally, the application of binary GA on mathematical problems, we have taken four mathematical problems. We have shown that performance and found that as we increase the binary string length, the optimum solution which we which is generated by BGA, it is closer to the optimum solution.

However, when we increase the number of variable for say Rastrigin function and Rosenbrock function, binary coded GA is still little far from the optimum solution we may require a larger population size, as well as more number of a generation to get an optimum solution. So, with these remarks I conclude module 2 on binary coded genetic algorithm.

Thank you.