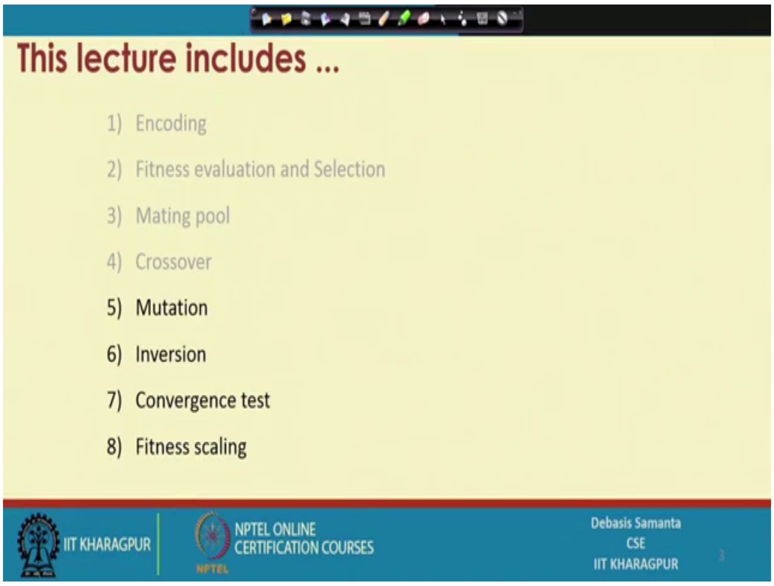
Introduction to Soft Computing
Prof. Debasis Samanta
Department of Computer Science & Engineering
Indian Institute of Science, Kharagpur

Lecture – 23
GA Operator: Mutation and others

We are discussing operations related to genetic algorithm; today we will discuss few more operations. So, this is related to reproduction task the operations are mutation, inversion.

(Refer Slide Time: 00:36)

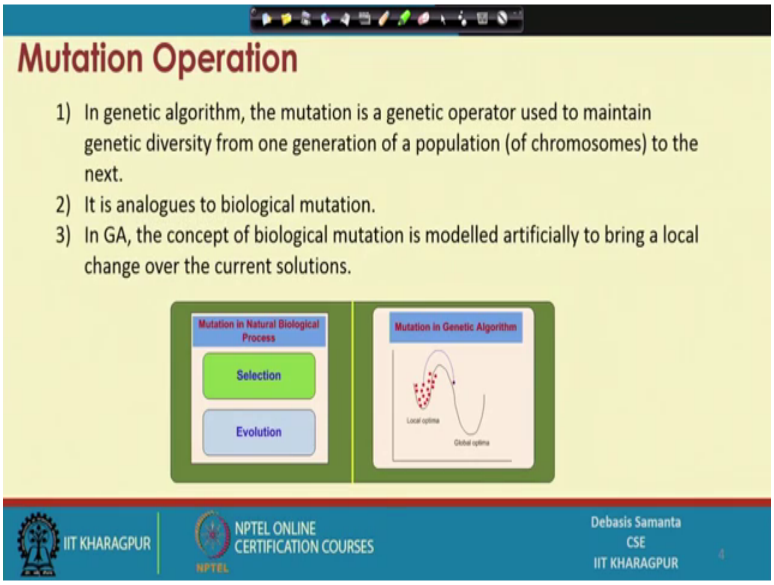


This lecture includes …

1) Encoding
2) Fitness evaluation and Selection
3) Mating pool
4) Crossover
5) Mutation
6) Inversion
7) Convergence test
8) Fitness scaling

So, like the crossover operations that we have discussed and it depends on the different type of encoding scheme; that means, different crossover techniques are to be followed to different type of GA. So, here also the mutation operations are dependent on the type of encoding scheme that we are following.

(Refer Slide Time: 00:56)



Mutation Operation

1) In genetic algorithm, the mutation is a genetic operator used to maintain genetic diversity from one generation of a population (of chromosomes) to the next.
2) It is analogues to biological mutation.
3) In GA, the concept of biological mutation is modelled artificially to bring a local change over the current solutions.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Debasis Samanta CSE IIT KHARAGPUR

So, first let us discuss about why the mutation operation is required? Or in what context the mutation operation in genetic algorithm is significant?

So, basically sometimes whenever the GA is in certain intermediate stage of alteration, it appears that there is not so much desirable level of population diversity; as a result whatever the solutions are obtained, they leads to a solution final solution because there will be no scope of further improvement. Now in order to search bit in a different direction so, what exactly the thing it is required is that, we have to change some chromosomes, some solution in an (Refer Time: 01:42) manner. So, the way it can be solved, I mean it can be changed and this is called the mutation operation.

Now, the mutation operation is very much similar to the biological mutation, and we know in case of biological mutation. So, there is a all of a sudden changes in the gene for example, in your garden if you plant a rose, tree and then the flower of the rose tree it is say it is red, then one day all of a sudden it is quite possible that one flower which appears is basically white colour or there is a tree out of so many red rose tree, there is a one tree which produce say the white flower.

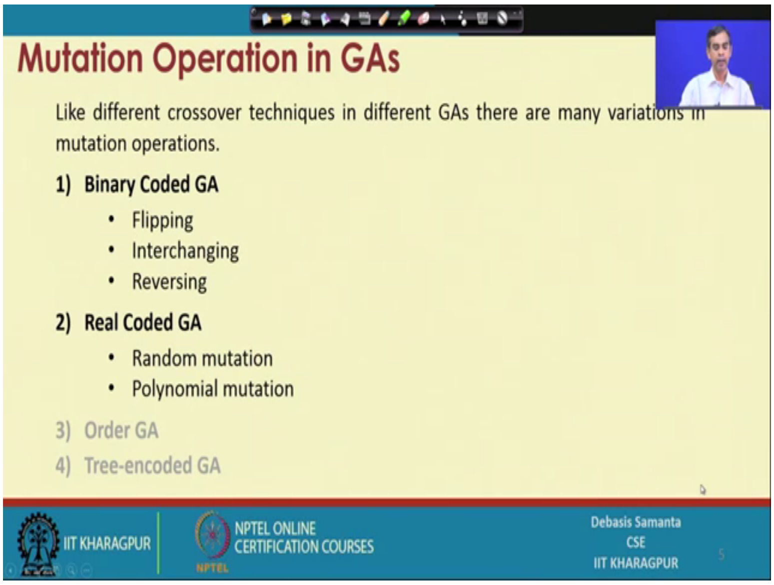
So, this is an example of mutation in our nature. So, similar mutation is followed in case of genetic algorithm also. So, basically it is the idea about that how forcefully we can modify a chromosome. Now why this mutation operation is required? We can understand these things if we look at the figure. So, this figure is basically planned for this purpose.

Now, we are searching for a solution and suppose at any instant the solution that we are here. So, if we do some what is called the reproduction by means of cross over and everything. So, from this two solution; it produce another solution here, so all the solutions are confined into this one, then after some iterations if we do not find any improvement, then we can accept the solution as the optimum solution. In fact, this is the solution that we obtain; it is called the local optimum, this is because we can diversify the such phase from these region to some other region here or some other region here so, that we can find, if some other what is called the optimized possible.

So, if we can change some chromosomes from this region to this region, then we can come to this another optimum maybe the better optimum or the global optimum. So, how these sudden changes in chromosome can be takes place? For example, if we consider this is the one chromosome and if we mutate it, then the mutate chromosome this one, it can leads to I mean to your search to some other optimum value.

So, this way the mutation operation is very much effective, and it is usually applicable when before taking a final decision that weather we will terminate the de iteration or we have to do something. So, in that case we can forcefully call the mutation operations on some chromosomes, and see whether the mutated chromosomes when solution is mutated chromosome, it can gives better solution or not. So, this is the one way that the population diversity can be achieved and the mutation operation is mean for this purpose.

(Refer Slide Time: 04:56)



Mutation Operation in GAs

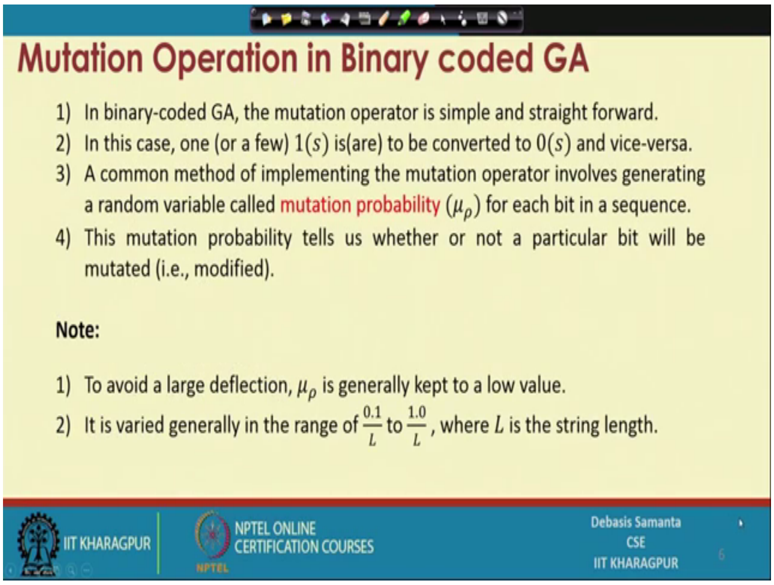
Like different crossover techniques in different GAs there are many variations in mutation operations.

1) **Binary Coded GA**
   - Flipping
   - Interchanging
   - Reversing
2) **Real Coded GA**
   - Random mutation
   - Polynomial mutation
3) Order GA
4) Tree-encoded GA

So, this is the a rational behind mutation operation, and there are different operations related to the difference GAs and in this lecture we will learn two different GA type mainly the binary coded and real coded GA. The other two mutation operation in other 2 GAs like order GA and n t GA is left as an self-study. So, it will not be possible to cover because of the timing constant.

So, we will discuss about 2 GA those are basically more important people usually follow binary coded GA or real coded GA; so, we will discuss our idea about these two. So, in case of binary coded GA. So for the mutation operations are concerned, there are three different versions one is called flipping, interchanging and reversing. Similarly for the real coded GA the mutation operation two different techniques are followed one is random mutation and then polynomial mutation.

(Refer Slide Time: 05:52)



Mutation Operation in Binary coded GA

1) In binary-coded GA, the mutation operator is simple and straight forward.
2) In this case, one (or a few) $1(s)$ is(are) to be converted to $0(s)$ and vice-versa.
3) A common method of implementing the mutation operator involves generating a random variable called mutation probability (μ_p) for each bit in a sequence.
4) This mutation probability tells us whether or not a particular bit will be mutated (i.e., modified).

Note:

1) To avoid a large deflection, μ_p is generally kept to a low value.
2) It is varied generally in the range of $\frac{0.1}{L}$ to $\frac{1.0}{L}$, where L is the string length.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Debasis Samanta CSE IIT KHARAGPUR

So, first let us discuss about the binary coded GA mutation, and we will discuss the flipping operation first. Now before going to take discussion about the flipping operation or other GA operations, mutation operation in binary coded GA; we will tell exactly what is the basic concept that is followed; so far the mutation is concerned.

Now, the mutation operator basically we know in case of binary coded GA, chromosome is represented in terms of 0’s and 1’s. So, if we change some 1’s into 0 and vice versa; then it is called the mutation. So, basically changing some 1’s or 0’s to 0’s or 1’s and it is a mutation.

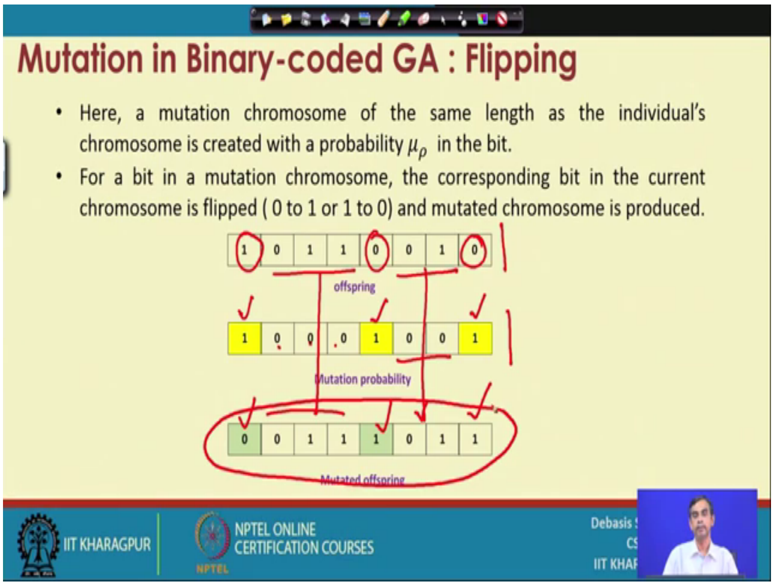
Now the question is that which bits in the binary chromosome should be changes? So, to decide it, we have to follow the mutation probability; that means, for each bit position; we have to decide the mutation probability, and then based on this mutation probability, we basically decide that how many bits are to be fit, or how many bits are to be reversed or interchanged?

So, this is the one parameter that user has to decide the mutation probability. So, we denote this mutation probability by the symbol mu rho, and there is a heuristic; heuristic is that this mutation rho should be a very low value. If we take the high value then your search will be in a random direction, that is some time not desirable and then it will take longer time to terminate the searching process and you may not get the optimal solution at all.

So, there is a good heuristic that is usually followed in genetic algorithm is that, if L is the length of the chromosome then this mu rho should be within the range 0.1 by l to L point 0 l. So, if the value of L is large as we see the, this probability will also reduce into a smaller value like.

Anyway so, this is the heuristic that is followed, and this is the heuristic that is followed to decide the mu rho it is called the mutation probability. So, like different parameters mutation probability is also another parameter it is called the GA parameter to be decided by the user during the execution of the genetic algorithm.

(Refer Slide Time: 08:22)



Mutation in Binary-coded GA : Flipping

- Here, a mutation chromosome of the same length as the individual's chromosome is created with a probability μ_ρ in the bit.
- For a bit in a mutation chromosome, the corresponding bit in the current chromosome is flipped (0 to 1 or 1 to 0) and mutated chromosome is produced.

1 0 1 1 0 0 1 0

offspring

1 0 0 0 1 0 0 1

Mutation probability

0 0 1 1 1 0 1 1

Mutated offspring

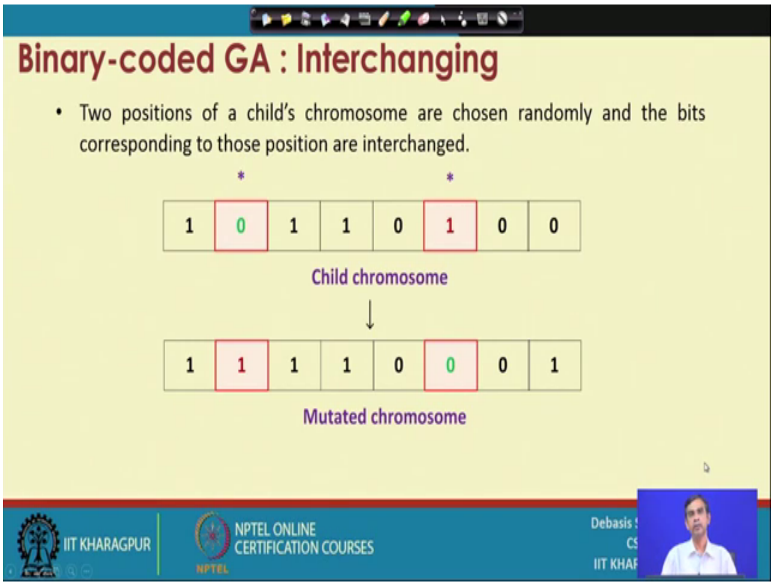
Now, having this is the concept mutation probability, we will discuss about the first operations related to the mutation in case of binary coded GA, the operation is called the flipping and as the name implies flipping means the 1 will be flipped to 0 or 0 will be flipped to 1 it is kind this. Now first I told you the mu rho needs to be decided so in fact, we will toss a coin and we will decide the coin in such a way that, mu rho number of tosses will be 1 and other will be 0.

So, somehow this toss can be planned or some program can be written. So, for example, this is the toss; so, this toss is decided in such a way that only few of the bits to be flipped, the bits which are to be fit it is marked as a yellow colour; so, this bits, this bits and this bits. So, if it is like this then suppose this is the one chromosome that is given to you we can say the child chromosome are offspring chromosome.

And so, based on this mutation probability we can bleep for example, it is a 1. So, this bleep will be flopped. So, it is 0. Now all the 0 0 and 0; so, this will remain unchanged. Now again 1, so this 0 will be flipped to 1 and there is a 0. So, this means they will not be changed and here this is 1. So, it is 0; so, this will be 1; so, this way this is the mutated chromosome can be obtained after the operation of shipping. So, idea is pretty straightforward and simple; in fact, and it is also not so time consuming operations.

And generally the child chromosome we have selected at random; that means, which chromosome needs to be mutated. So, you can take the child chromosome at random. And not all the child chromosome to be mutated, there are again few I mean child chromosome to be mutated. So, that is also very lesser number of things to be mutated, and it depends on. If you want to have a very high diversity then you can go for a large number of child chromosomes and for a smaller diversity it is less number of child chromosome can be considered for the mutation.

(Refer Slide Time: 10:40)



Binary-coded GA : Interchanging

- Two positions of a child's chromosome are chosen randomly and the bits corresponding to those position are interchanged.

	*				*		
1	0	1	1	0	1	0	0

Child chromosome

↓

1	1	1	1	0	0	0	1

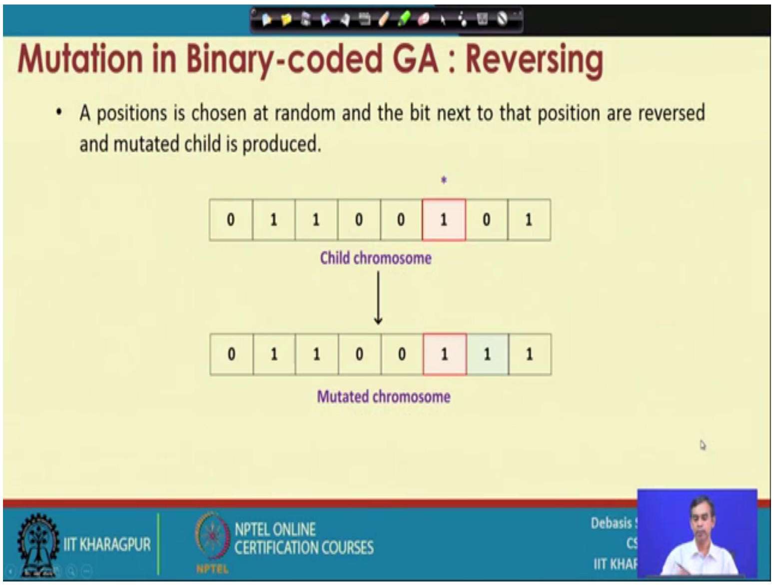
Mutated chromosome

So, this is the flipping operations, and I will discuss about the next operations mutation operations it is called the interchanging. So, in this case again 2 bit position are to be selected at random. So, randomly we select 2 bit position for example, this is the child chromosome and then these are the 2 bits are selected at random in between the inter chromosomes, there and then interchanging means this bit 0, if it is 0; it will be interchanged to 1 and if it is 1; this will be 0 it is just like a flipping of course, but in case of flipping operation, there are certain tossing required.

But here we do not have to do any tossing only the thing is that we have to select two bit position random. Sometimes instead of 2 bit positions we can take 3 bit positions or more number of positions, and then accordingly all those bit positions can be mutated.

So, it is almost similar as the flipping operation also, but let it be in a different way. So, user can try with first 2 bits to be mutated then 3 bits, and then less number of bits whatever it is there and this one. So, it is more controllable than the previous one in fact.

(Refer Slide Time: 11:56)



Mutation in Binary-coded GA : Reversing

- A positions is chosen at random and the bit next to that position are reversed and mutated child is produced.

0	1	1	0	0	1	0	1

Child chromosome

0	1	1	0	0	1	1	1

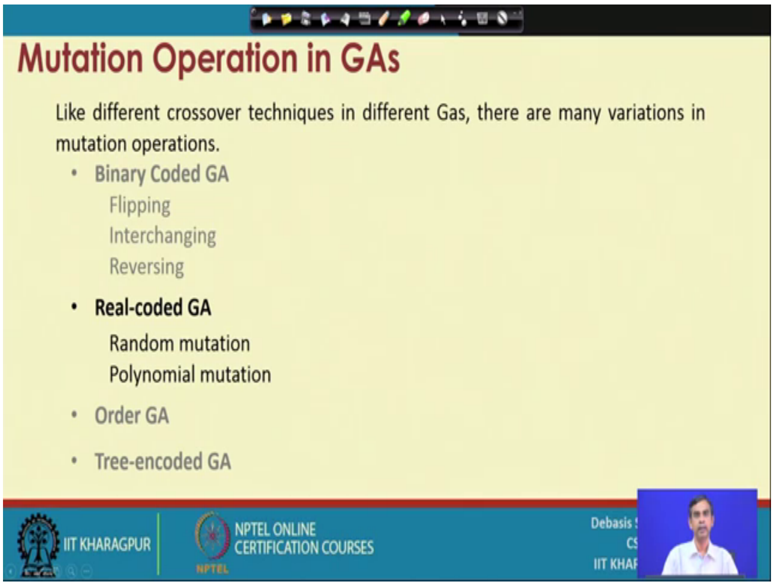
Mutated chromosome

So, this is a interchanging and the next operation is called the reversing. So, idea it is basically the idea it is that, here in the previous interchanging operations what we have to do is we have to select a k number of this position, but here you do not have to select k number of bit positions, this is basically the mutation operation is required whenever you need a very slight modification in the chromosomes.

So, it is basically the idea it is that you have to fist generate a random number or we can say the decided a bit position at random for example, this is the child chromosome, and we decide 1 bit position at random this one. Then what is the procedure here is that either the previous bits or next bit whatever it is there. So, we can fit it. So, the previous bit for example, the next bit suppose if you consider the next bit 0 1 up to the selected bit will flip it. So, if we flip it then it will give this one 1. So, this is a mechanism that is followed here in case of reversing, and this is again most simple method compared to the previous two methods that we have discussed.

So, these are the 3 methods that is followed in case of binary coded GA and for the mutation operation.

(Refer Slide Time: 13:10)



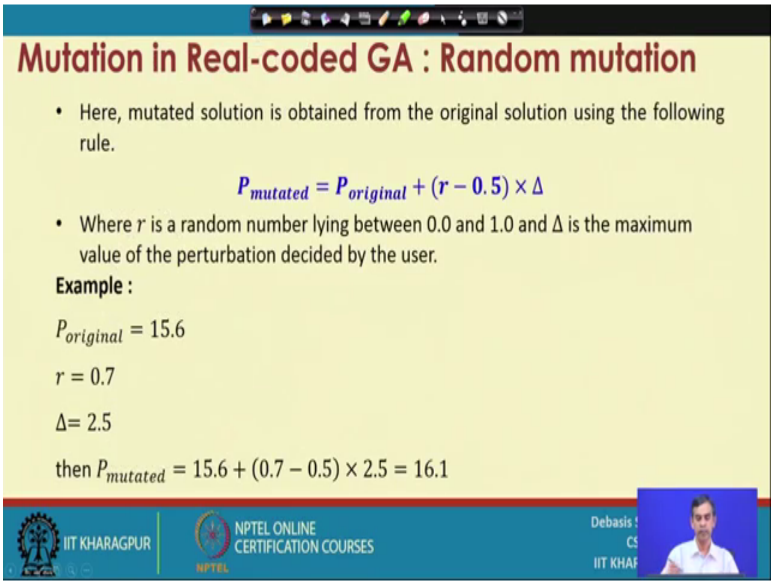
Mutation Operation in GAs

Like different crossover techniques in different Gas, there are many variations in mutation operations.

- Binary Coded GA
  - Flipping
  - Interchanging
  - Reversing
- **Real-coded GA**
  - Random mutation
  - Polynomial mutation
- Order GA
- Tree-encoded GA

Now, we will discuss about the real coded GA, as I told you there are 2 techniques the random mutation and polynomial mutation. Now let us discuss the 2 techniques one by one so, first we will discuss about random mutation.

(Refer Slide Time: 13:25)



Mutation in Real-coded GA : Random mutation

- Here, mutated solution is obtained from the original solution using the following rule.

$$P_{mutated} = P_{original} + (r - 0.5) \times \Delta$$

- Where r is a random number lying between 0.0 and 1.0 and Δ is the maximum value of the perturbation decided by the user.

Example :

$P_{original} = 15.6$

$r = 0.7$

$\Delta = 2.5$

then $P_{mutated} = 15.6 + (0.7 - 0.5) \times 2.5 = 16.1$

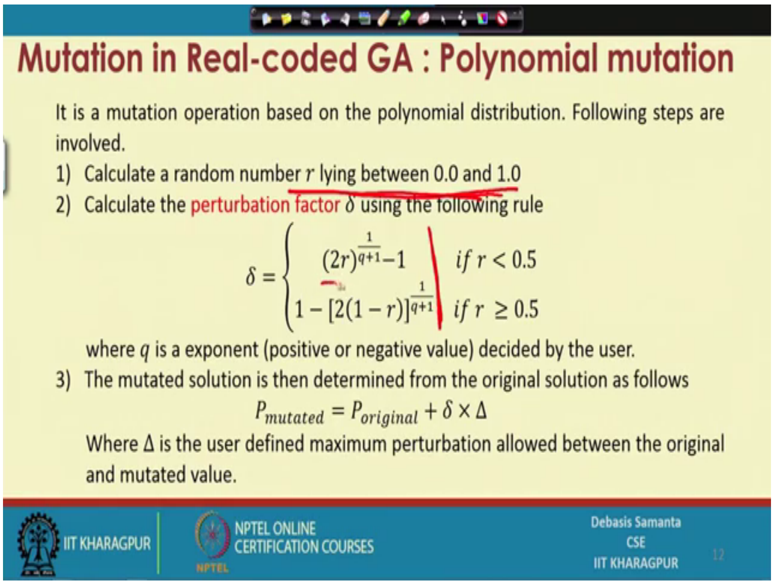
So, here mutation solution is obtained by means of some what is called the formula, this formula is basically given empirically, the formula takes the form which is shown here. So, this is the formula. So, it basically the parent values of the original it is basically child chromosome, and then it computes some calculation like this here the r is basically the random number random number in the range between 0 0 and 1.0, and delta is also another constant it is decided by the user. So, this constant it is basically called the perturbation factor.

So, if we compute this equation, then it will produce a mutated chromosome that is denoted as P mutated. So, in this operation the ask that is required you have to generate a random number first, and then delta is already predefined constant decided by the user and use this calculation this calculation is also simple calculation only in terms of some division addition and multiplication not so, costly calculation here and we will be able to obtain the another value from a given value.

So, idea it is like this let us illustrate the concept of this random mutation with an example, say suppose this is the parent value I mean chromosome value of a child chromosome and this is the random number, which is generated at that instant and this is the fit factor that is decided for this process. Then the mutated chromosome can be represented which can obtain this value, if you follow the expression which is already stated there. So, this way from one value that is for the chromosome value belongs to a particular child will be mutated to another value. So, there is a slight changes basically we have now; obviously, much how much deflection, how much diversity you want that depends on these factor. So, we can control these values, and then accordingly some values can be obtained which is higher than the value that is required. So, these are the basically perturbation factor, if we control this value then the different chromosome mutated chromosome can be obtained.

So, this is the idea about that random mutation.

(Refer Slide Time: 15:52)



Mutation in Real-coded GA : Polynomial mutation

It is a mutation operation based on the polynomial distribution. Following steps are involved.

1) Calculate a random number r lying between 0.0 and 1.0
2) Calculate the perturbation factor δ using the following rule

$$\delta = \begin{cases} (2r)^{\frac{1}{q+1}} - 1 & if\ r < 0.5 \\ 1 - [2(1-r)]^{\frac{1}{q+1}} & if\ r \geq 0.5 \end{cases}$$

where q is a exponent (positive or negative value) decided by the user.

3) The mutated solution is then determined from the original solution as follows

$$P_{mutated} = P_{original} + \delta \times \Delta$$

Where Δ is the user defined maximum perturbation allowed between the original and mutated value.

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

Debasis Samanta
CSE
IIT KHARAGPUR

Now, we will discuss about a little bit computationally expensive, but gives better result usually it is called the polynomial mutation.

Now, in this mutation like in case of random mutation we have to follow a random number. So, this is a random number r in the same range as in between 0.0 to 1.0 and then we have to calculate another factor here, it is called the perturbation factor delta and. In fact, in the previous method it is a users responsibility to decide the delta.

But in this case it is an idea it is given that the delta can be calculated more statistically or more probabilistically then, and this calculation is based on some statistical function distribution function which is there. So, one distribution function is follows there, if the random number r is less than 0.5 and another function that is followed here if the number less than the, this one. So, these are the 2 expression given by the developer or the designer itself and we can consider this as empirical formula and then.

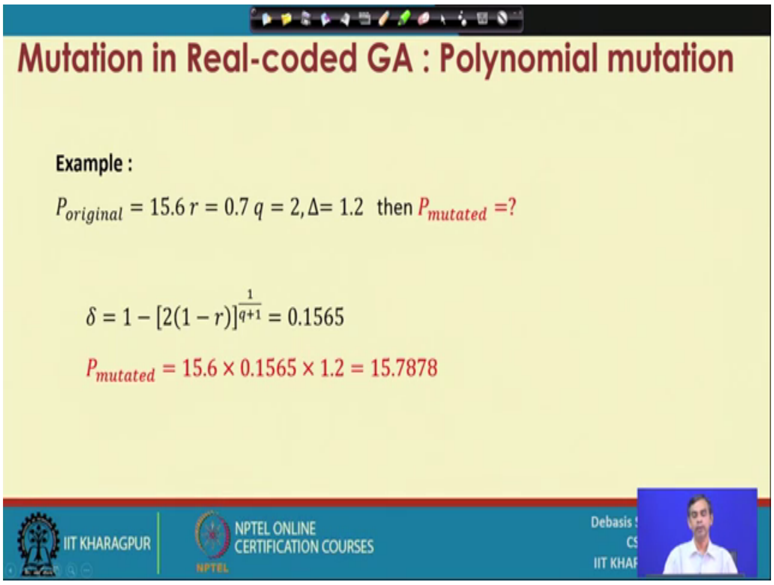
So, following this formula and based on the values of r we will be able to delta. So, here we can see in the previous case delta is fixed for any operations, but here the delta is not fixed rather delta is decided by the r value always. So, here because r is there and r is there and accordingly this one. So, delta is basically dependent on r. So, delta is not truly a fix for all mutation operations for any other chromosome. So, it is basically varies from one operation to another operation as r varies.

And here in this operation another constant to be decided by the user q, like delta that is there we have to also decide one constant and that constant can be based on the designer experience or users experience.

So, once you know the value of q, and then r can be decided a random and accordingly delta can be computed, and then we will be able to use this formula here. So, the mutated chromosome that can be obtained using this formula. So, p original and delta and this is again another perturbation factor, that we have followed there in case of random there also you have to concerned it or sometimes only delta something into this one some other in terms of values also can be constant. So, anyway. So, if we fix we consider an fixed deviation that is allowed. So, this is the delta and then based on this things the mutated chromosome can be obtained.

So, this is the idea about the polynomial mutation in case of GA real coded G A.

(Refer Slide Time: 18:34)



Mutation in Real-coded GA : Polynomial mutation

Example :

$P_{original} = 15.6\ r = 0.7\ q = 2, \Delta = 1.2$ then $P_{mutated} = ?$

$$\delta = 1 - [2(1-r)]^{\frac{1}{q+1}} = 0.1565$$

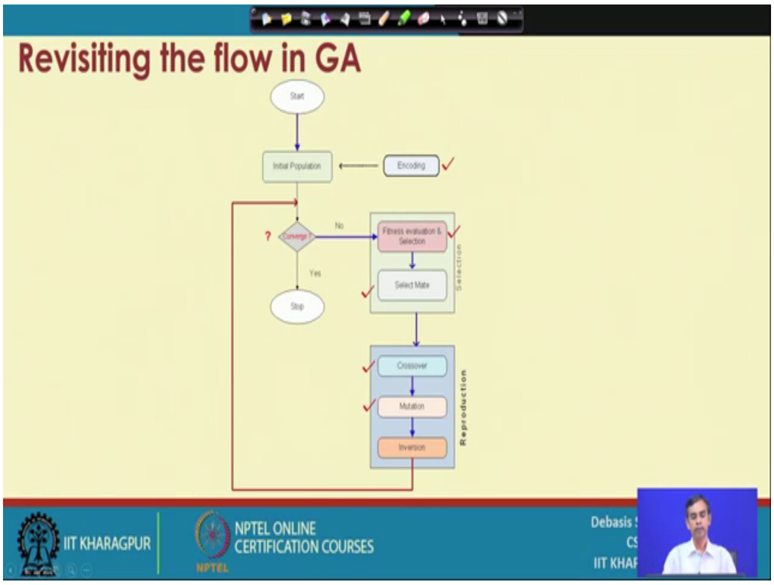
$$P_{mutated} = 15.6 \times 0.1565 \times 1.2 = 15.7878$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Debasis | CS | IIT KHAR

Here is an example in this example we consider the child chromosome the value is 15.6 r is decided a random 0 point c q is the 2 standard constant data is another constant remain throughout this one. And then now we have to calculate P mutated. First we have to calculate delta in this case r is 0.7. So, the second formula needs to be followed and this formula gives the value of delta is this one and once the delta value is known we will be able to calculate using the same formula and then this one. So, if this is the child chromosome then the mutated chromosome is like this one. So, this way the mutated chromosome value can be obtained.

So, these are the 2 I mean techniques that is there for the real coded GA.

(Refer Slide Time: 19:24)



Revisiting the flow in GA

Now, we have discussed many operations and the GA regarding the GA cycles, we have discussed about the how to create the population by means of the encoding scheme, and then we have also discuss about how to evaluate the fitness of each solution and then the selection can be carried out, and then we have to create a meeting pool. So, this completes the selection operation.

And then comes to the reproduction operation. So, for the reproduction operation the crossover and mutation that you have been discussed in details. Now there is another operation it is basically called the inversion it is part of the mutation operation it is part of the reproduction task.

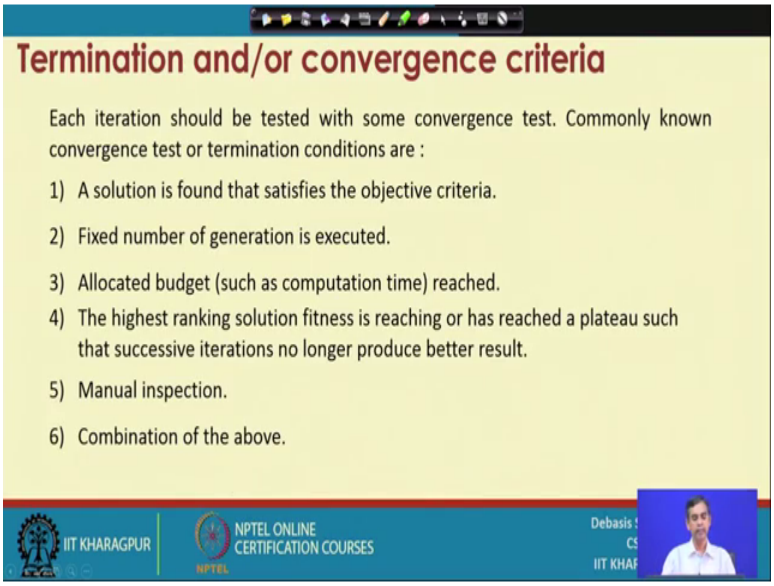
So, in case of inversion operation is a very dusty corn operation usually occurs very little time in the entire GA cycles maybe out of the 100 cycles we have to follow 1 or 2, and that is to not to all chromosome for some chromosomes. So, the inversion operation is basically select some chromosomes in the current population at random, say out of thousand we can select may be say 20. So, 0.2 percent like this one or this one, then out of this a selected chromosome in the current population we have to follow inversion before going to either crossover or mutation or cycle basically.

So, inversion operation basically it will changes in case of binary coded, it will change basically all zeros to all ones and all ones to all zeros. So, this way a drastic changes can take place on the other hand in case of real coded GA if the value is very low then we can change this to a very high value. So, low to high value or high to low value is the inversion operation that is in case of real coded GA.

And so, these are the operation that is there we have discussed all these operations now here we have to discuss about the convergence; that means, how to terminate this cycle or how long we have to continue this searching for the optimum solution.

So, in the next few slides will discuss the convergence operations.

(Refer Slide Time: 21:40)



Termination and/or convergence criteria

Each iteration should be tested with some convergence test. Commonly known convergence test or termination conditions are :

1) A solution is found that satisfies the objective criteria.
2) Fixed number of generation is executed.
3) Allocated budget (such as computation time) reached.
4) The highest ranking solution fitness is reaching or has reached a plateau such that successive iterations no longer produce better result.
5) Manual inspection.
6) Combination of the above.

Usually the convergence criteria we have to follow, I have listed few important criteria usually the GA programmer follows. So, the first criteria says that the first criteria is basically if we find a solution, which is basically our expected one solution then we can terminate because if we know that this is the expected results to be then we can stop it there and then what are the solution we got it design values design parameters result we can take it as a solution.

So, this is basically whenever you expect the desired result satisfying the objective criteria then we can stop it there. This is a first criteria that we follow, in the second criteria is basically we can defined the maximum number of cycles that we should execute. So, is basically how many cycles that needs to be executed if we decide, say maybe it is 50 sometime 100 depending on your I mean computation affordability how much computation time that you can afford we can decide the fix number of cycles that needs to be alternated and based on this things second criteria is followed.

Now, another idea is that the budget allocation, budget allocation in the sense that I will allow maximum 3 hours, to run once GA algorithm. So, 3 hours in the 3 hours if it is terminate before this thing it is fine if it does not we have to continue the search till the 3 hours is over. So, depending again based on the programmers time available.

So, they can fix the budgets that mean computation time budget or sometimes the memory budget. So, within this memory we have to solve it, then how many iterations whatever you want you do it. So, it is whatever the budget it is their time budget or memory budget we have to follow it and then so long time budget permits we can cycle the GA operation.

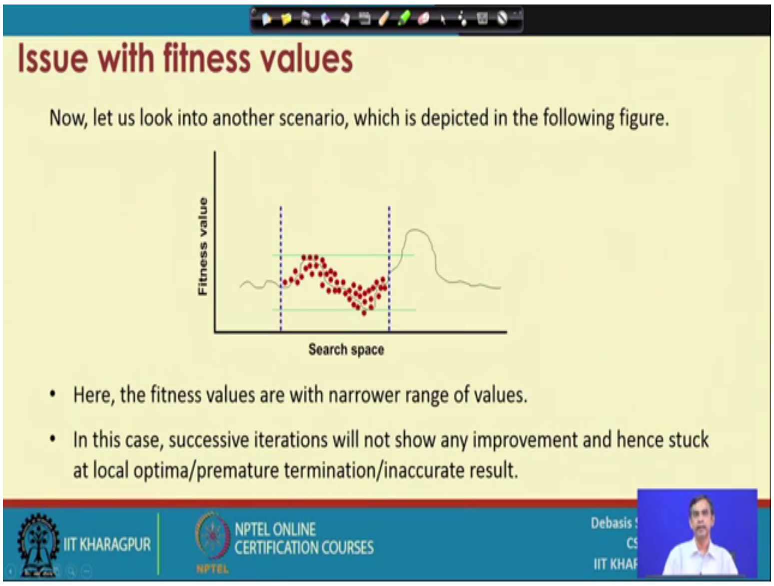
The next is. So, another criteria is that. So, fine sometimes we have to find the best solution, best solution in terms of say ranking of the fitness if we find one highest ranking fitness solution right that has reached to a basically after a successful number of iterations. So, after suppose 10 conjugative iterations we are getting the highest ranking solution all the time; this means that we have already come to a global optima and then we can stop the search criteria there. So, this is the one criteria and then manual inspection it is a little bit tedious and very difficult, if it is very small number of solutions are there then we can do it. So, is basically.

So, check the solutions 1 by 1 if you plot the solution graphically also sometimes it works, and then we can decide whether we should terminate this one. So, in this case you have to run on cycle check the solution manually and then decide whether you should continue the iteration or stop it. So, this is; obviously, not a desirable operations many programmer do not like it, and another criteria is basically combination of anyone or any 2 or any combinations out of the 5 criteria that we have decided. So, this is obviously, at the cost of time because we have to check lot of things in after every iteration, because whether these are the criteria is satisfied or not.

So, this is basically the rule of thumb. So, for the convergence criteria is concerned and usually we follow this kind of method.

Now, so, we have learned about the different operations in particular operations related to reproduction. Now there are few issues and this issues are related to the fine tuning the g operations. One issue in this case is the fitness scaling and we will discuss about the fitness scaling and their different techniques which are there in the fitness scaling approaches.

(Refer Slide Time: 25:31)



Issue with fitness values

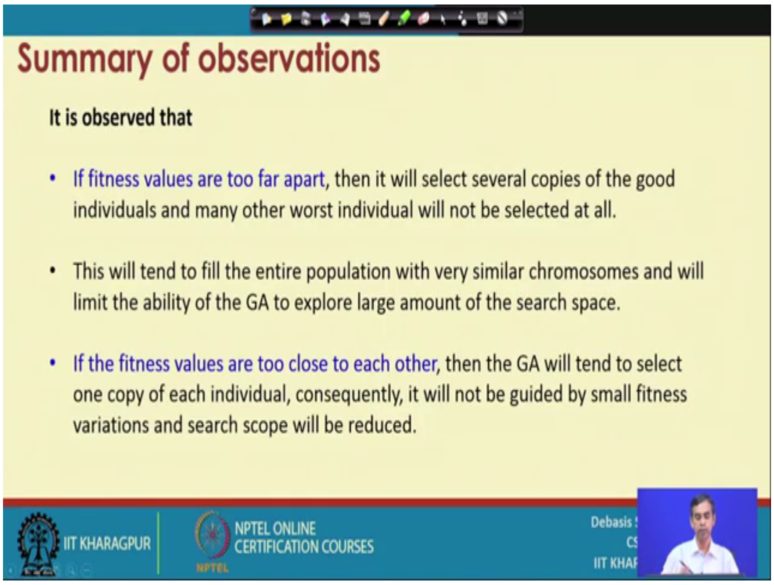
Now, let us look into another scenario, which is depicted in the following figure.

- Here, the fitness values are with narrower range of values.
- In this case, successive iterations will not show any improvement and hence stuck at local optima/premature termination/inaccurate result.

Now, idea of the fitness scaling obtained from on what is called the situations the situation can be explained like this one, suppose at any instant of the searching. So, these are the solution it is available. Now if we check the range of the fitness values of the solutions. So, we see that these are the range; that means, the lowest fitness value to the highest fitness value this range.

Now, sometime this range matters a lot. So, this range in fact, signify whether there will be premature convergence or inaccurate result and everything. Now let us consider few situations how this gap between the lowest to highest fitness value matters; that means, if the high gap how it works if the gap is narrow then how it works or whatever the gaps it is require.

(Refer Slide Time: 26:27)



Summary of observations

It is observed that

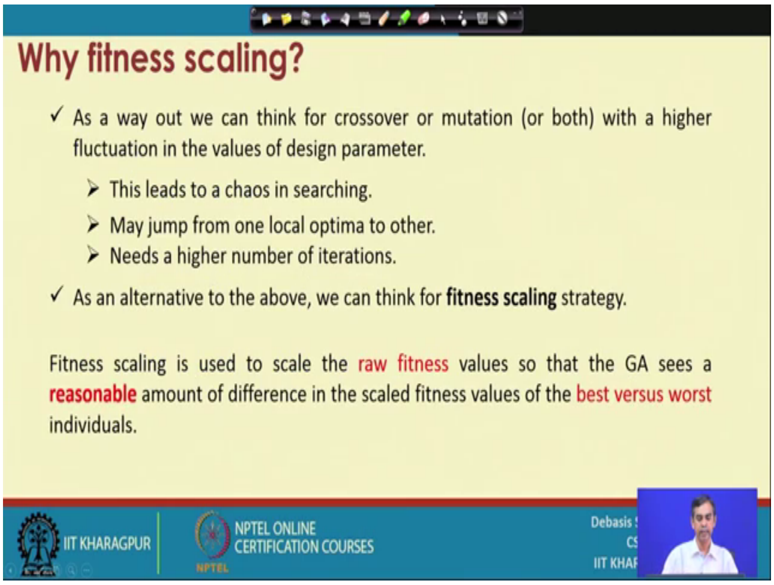
- If fitness values are too far apart, then it will select several copies of the good individuals and many other worst individual will not be selected at all.
- This will tend to fill the entire population with very similar chromosomes and will limit the ability of the GA to explore large amount of the search space.
- If the fitness values are too close to each other, then the GA will tend to select one copy of each individual, consequently, it will not be guided by small fitness variations and search scope will be reduced.

So, it is basically to trade of cities there if fitness values are too far apart mean; that means, they are having very wide gap then it will select several copies of the good individuals. So, all is and many other worst individuals may not be selected at all. So, this is the one issues are there.

So, this will basically tend to fill the entire population with very similar chromosomes, and eventually it will terminate to a local optimum possibly. On the other hand if the fitness values are too close to each other; that means, the gap is very narrow, then the GA will tend to select 1 copy of individual and conjugately it will not be guided by the small finance variation, and such scope will be terminated.

So, it is basically now reduce such scope will be explode. So, both the techniques both the consequence situations have their own limitation. So, this means that we should have the fitness values of the individuals in such a way that the gap between the highest to lowest should not be narrow neither or it should not be the wider again. So, there some reasonable gap how this reasonable gap can be ensured, we will discuss and the different techniques are there.

(Refer Slide Time: 27:40)



Why fitness scaling?

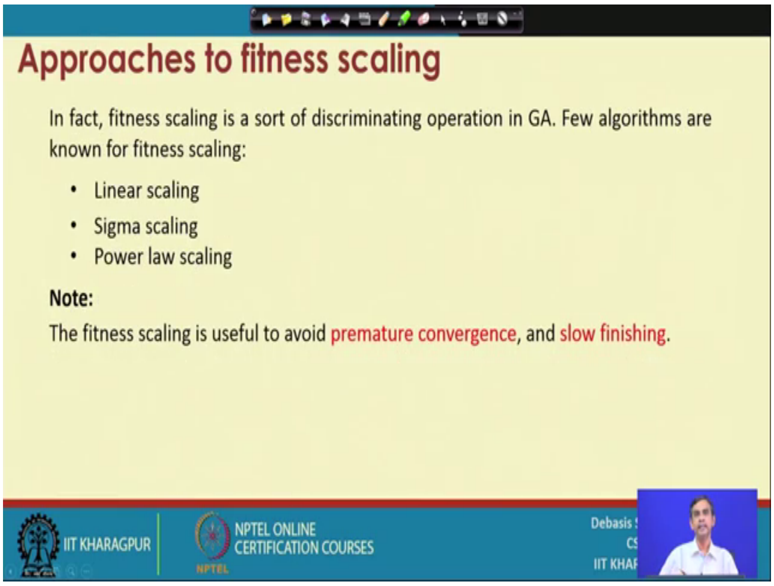
- ✓ As a way out we can think for crossover or mutation (or both) with a higher fluctuation in the values of design parameter.
  - ➤ This leads to a chaos in searching.
  - ➤ May jump from one local optima to other.
  - ➤ Needs a higher number of iterations.
- ✓ As an alternative to the above, we can think for **fitness scaling** strategy.

Fitness scaling is used to scale the raw fitness values so that the GA sees a **reasonable** amount of difference in the scaled fitness values of the best versus worst individuals.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Debasis
CS
IIT KHAR

(Refer Slide Time: 27:44)



Approaches to fitness scaling

In fact, fitness scaling is a sort of discriminating operation in GA. Few algorithms are known for fitness scaling:

- Linear scaling
- Sigma scaling
- Power law scaling

Note:

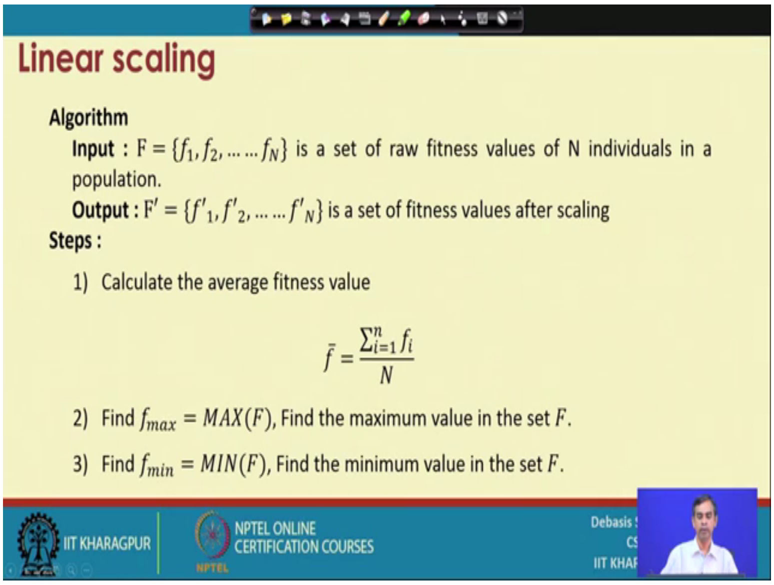
The fitness scaling is useful to avoid premature convergence, and slow finishing.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Debasis
CS
IIT KHAR

So, basically idea is that from the law fitness value we have to evaluate the better fitness value, and there are 3 techniques usually followed it is basically linear scaling, sigma scaling and power law scaling.

(Refer Slide Time: 27:56)



Linear scaling

Algorithm

Input : F = $\{f_1, f_2, \dots \dots f_N\}$ is a set of raw fitness values of N individuals in a population.

Output : F′ = $\{f'_1, f'_2, \dots \dots f'_N\}$ is a set of fitness values after scaling

Steps :

1) Calculate the average fitness value

$$\bar{f} = \frac{\sum_{i=1}^{n} f_i}{N}$$

2) Find $f_{max} = MAX(F)$, Find the maximum value in the set F.
3) Find $f_{min} = MIN(F)$, Find the minimum value in the set F.

IIT KHARAGPUR

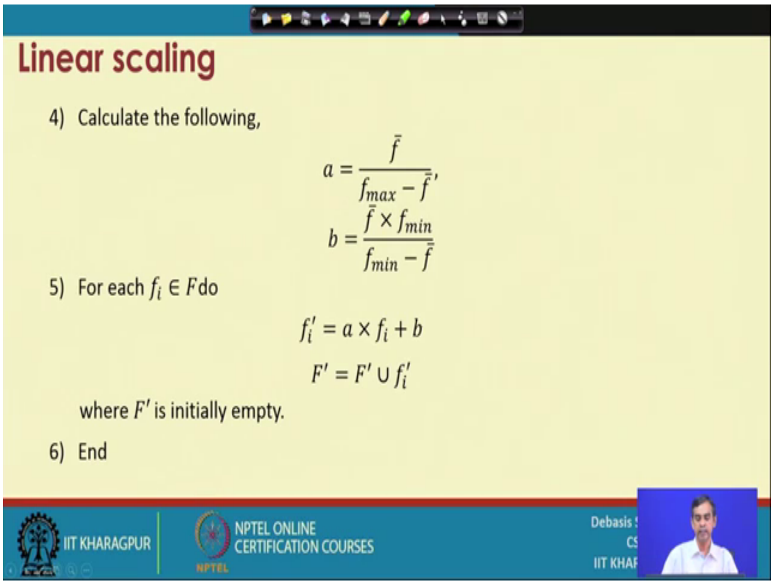
NPTEL ONLINE CERTIFICATION COURSES

Debasis
CS
IIT KHAR

Will quickly covered the 3 techniques here; the idea of the linear scaling is discussed first. So, here basically these are the law of fitness values of the current population where the n number of solutions are there, and this algorithm linear scaling will produce the scaled fitness values; that means, fitness value should be changed so that the gap between the lowest and highest is reasonable.

So, idea it is there; so, in this process we have to calculate the average fitness value using this formula. So, it is basically average of all fitness values and then it calculates f max and f min; that means, that has the highest fitness value and this is the lowest fitness values one these two values are obtained then it basically follows the decision how to change it.

(Refer Slide Time: 28:39)



Linear scaling

4) Calculate the following,

$$a = \frac{\bar{f}}{f_{max} - \bar{f}},$$
$$b = \frac{\bar{f} \times f_{min}}{f_{min} - \bar{f}}$$

5) For each $f_i \in F$ do

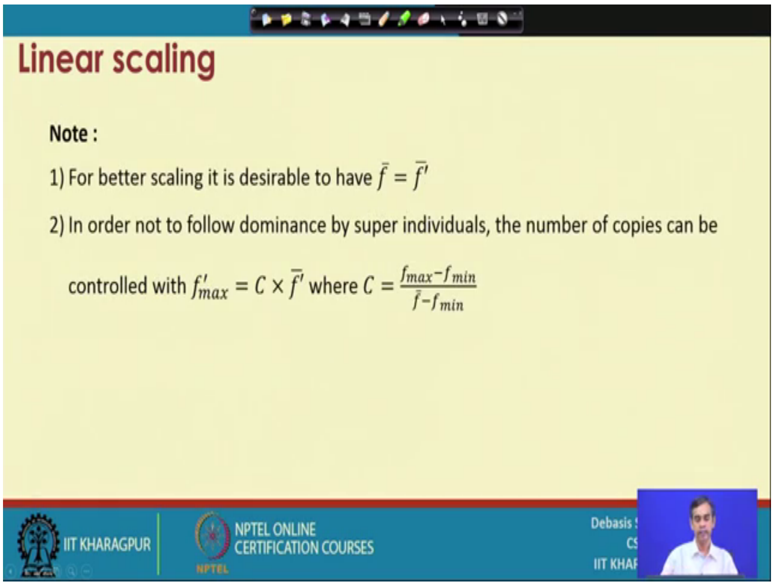
$$f_i' = a \times f_i + b$$
$$F' = F' \cup f_i'$$

where F' is initially empty.

6) End

So, basically in this method, linear scaling it computes a and b the 2 value using this formula. Once this two values a and b known we will be able to obtain the scale fitness value using this formula, and this is the fitness value it needs to be added into the f dash where f dash is basically, the set of all the scale fitness value and f dash is initially empty. So, this is a method that is followed there in case of linear scaling.

(Refer Slide Time: 20:07)



Linear scaling

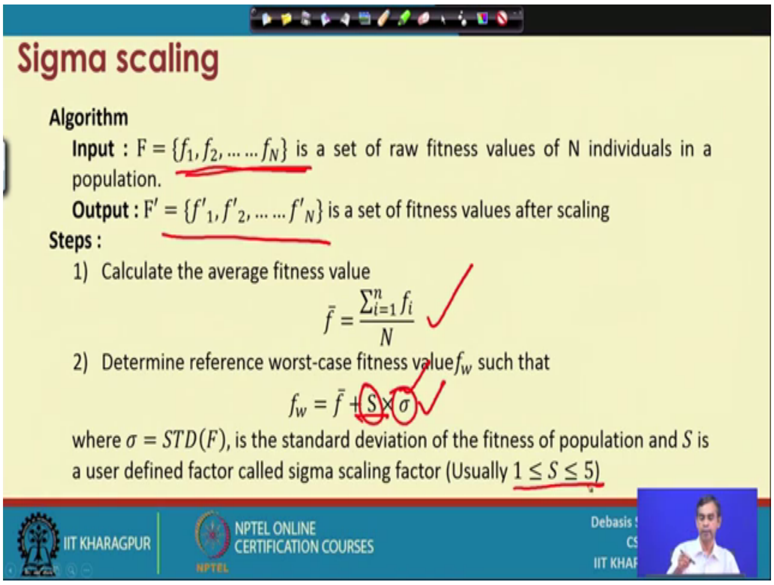
Note :

1) For better scaling it is desirable to have $\bar{f} = \bar{f}'$

2) In order not to follow dominance by super individuals, the number of copies can be controlled with $f'_{max} = C \times \bar{f}'$ where $C = \frac{f_{max} - f_{min}}{\bar{f} - f_{min}}$

And then there is another idea about this is called the sigma scaling another technique on case of sigma scaling.

(Refer Slide Time: 29:11)



Sigma scaling

Algorithm

Input : F = $\{f_1, f_2, \dots\dots f_N\}$ is a set of raw fitness values of N individuals in a population.

Output : F′ = $\{f'_1, f'_2, \dots\dots f'_N\}$ is a set of fitness values after scaling

Steps :

1) Calculate the average fitness value

$$\bar{f} = \frac{\sum_{i=1}^{n} f_i}{N}$$

2) Determine reference worst-case fitness value f_w such that

$$f_w = \bar{f} + S \times \sigma$$

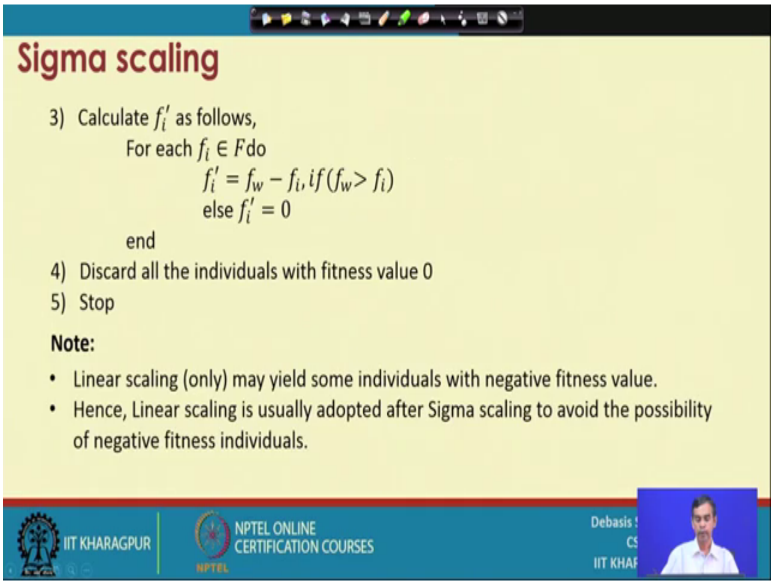
where $\sigma = STD(F)$, is the standard deviation of the fitness of population and S is a user defined factor called sigma scaling factor (Usually $1 \leq S \leq 5$)

So, input and output are the same as the previous one, it calculates the average fitness value it is there. But the different between the previous linear scaling and sigma scaling is rise here. So, in this method we have to decide 2 parameters, the S and the sigma where the sigma is basically the standard deviation of all the fitness value it is there.

So, it is basically standard deviation and S is the one factor it is also called the sigma scaling factor and usually this value is in between 1 and 5, it is a standard procedure that is followed value the lowest value of the S as lowest as small as 1 and then highest value is as 5.

So, one these values are known to us we will be able to calculate f w one calculation for the entire population. So, f w is calculated based on this formula.

(Refer Slide Time: 30:07)



Sigma scaling

```
3) Calculate $f_i'$ as follows,
       For each $f_i \in F$ do
           $f_i' = f_w - f_i, if (f_w > f_i)$
           else $f_i' = 0$
       end
4) Discard all the individuals with fitness value 0
5) Stop
```

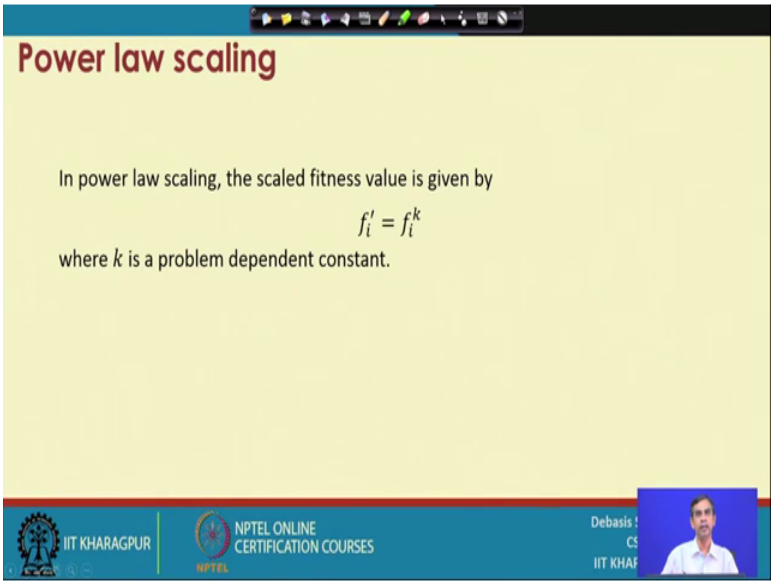
Note:

- Linear scaling (only) may yield some individuals with negative fitness value.
- Hence, Linear scaling is usually adopted after Sigma scaling to avoid the possibility of negative fitness individuals.

Once this p w is known we can use this f w to calculate the raw fitness and the scale fitness value. So, far each f i in f that is a given population, we have to calculate the scale fitness value f i dash which is basically f w minus f i. If f w is greater than f i and it is 0 in if this is 0.

So, this way the scale fitness value for the entire population can be obtained. So, this concept is followed there in case of sigma scaling, and usually the sigma scaling is followed the linear is followed by the linear scaling because some time linear scaling can results some raw scale fitness value is a negative which is not acceptable. So, we can follow the sigma scaling after the linear scaling so that the more refined scaled feature fitness value can be obtained.

(Refer Slide Time: 30:59)



Power law scaling

In power law scaling, the scaled fitness value is given by

$$f_i' = f_i^k$$

where k is a problem dependent constant.

So, this is the idea about the sigma scaling and the simplest scaling is called the power loss scaling, it is very simple idea is a neat approach we can say, if a f i is the current fitness value user has to decide a k value, k it usually some constant when including the real value also 1.5, 1.2, 2.5 5 or 2 whatever it is there means how much you have to have a variation.

So, there then the fitness scale fitness value can be obtained by means of this exponential calculation and then this is the idea, such this is very simple and straightforward method sometimes it is we followed there in order to have a very good gap between the lowest and highest value. So, this is the method.

So, we have disused about the different scaling operations and the fitness scaling basically and. So, this includes the operation that is there. So, for the GA reproduction is concerned, we have learn about the GA reproduction which includes a crossover then mutation and then scaling operation convergence criteria. And we will discuss about the next new topics in the next class it is basically multi objective optimisation.

Thank you.