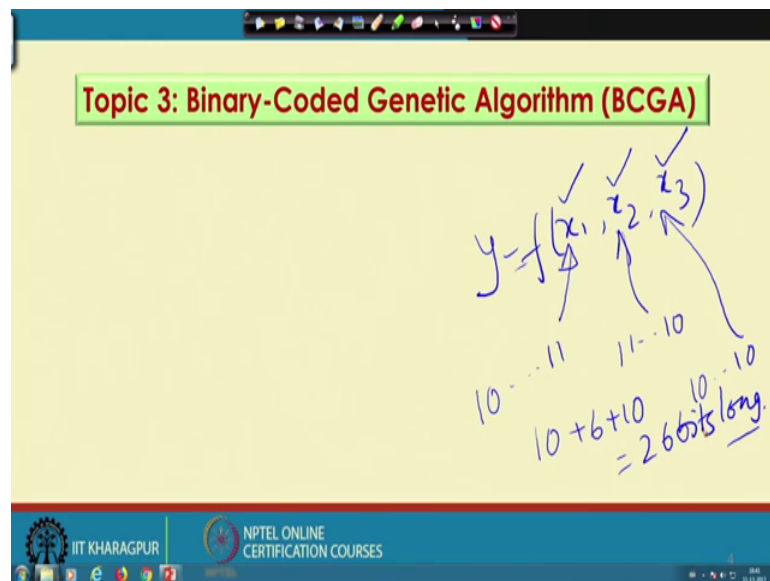


Traditional and Non-Traditional Optimization Tools
Prof. D. K. Pratihar
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture - 44
Summary – II

Now, the third topic that is on binary coded genetic algorithm.

(Refer Slide Time: 00:20)



So, here actually what we do is the design variables those are expressed in terms of the binary codes. Like if I have got a function like y is a function of say 3 variable like x_1 , x_2 , and x_3 . And let me take that out of these 3 variable say x_1 is a real variable, x_2 is an integer variable, and x_3 is once again a real variable. So, each of this particular variables are having their own rangers.

Now, within this particular range so, what we do is we will have to explain we will have to represent. So, this particular variable and to represent each of the variable we use some binary codes. For example, say depending on the accuracy limit or the precision limit. So, we will have to assign some number of bit to express this particular x_1 , supposing that say I am representing I am using say 10 bits to represent x_1 then I am using say 6 bits to represent say x_2 and I am using. So, once again that 10 bits to represent say x_3 .

So, the GA string will be 10 plus 6 plus 10. So, it is nothing, but 26 bits long. So, I will have to use that 26 bits long GA string and here in binary coded GA actually as I told the variables are coded in the form of the binary number.

Now, once we have coded this particular in the form of binary. So, let us see how does it work?

(Refer Slide Time: 02:09)

Topic 3: Binary-Coded Genetic Algorithm (BCGA)

- Introduction to GAs
- Working Cycle of a GA
- Working Principle of a BCGA

Handwritten annotations on the slide include:

- A circle containing N and n .
- The text "Reproduction of Mating Pool" written in blue ink.
- A diagram showing a matrix of binary strings (10, 01, 10) with arrows pointing to variables x_1, x_2, x_3 .
- The function definition $y = f(x_1, x_2, x_3)$.

The slide footer includes the IIT KHARAGPUR logo and the text "NPTEL ONLINE CERTIFICATION COURSES".

Now, the working cycle of GA, now here actually what I do is we start with a population of solution. Now if it is binary coded GA we start with a large number of binary codes binary strings for example, say if the population size is 100. So, we use 100 GA string and this particular GA string those are generated at random using the random number generator.

Now, once you have got the number of bits to be assign to each of the variable. Now I can find out the decoded value for each of these particular the substance used to represent the design variable. And once I have got the decoded value and knowing the respective rangers and using the linear mapping rule.

So, we can find out what should be the real values for this particular the real variables and those decoded value will be the values for this particular the integer variables. And once I have got the numerical values for this particular so, this the numerical values for the design variables. So, I can find out what should be the value for this particular the

objective function, that is y and for the whole population of this particular binary coded GA.

So, if this is the initial population of the binary coded GA; so I can find out for the whole population of size N . So, I can find out the fitness information. So, these are nothing, but the fitness information for the whole population. And once I have got the fitness information for the whole population. Now we are in a position to decide out of these n solutions, which are good and which are bad. Now to decide which are good which are bad we take the help of one operator that is called the reproduction operator. Now the purpose of using this reproduction operator is to form a mating pool.

Now, as this initial population is generated at random. So, there is no guarantee that all the solutions will be equally good there could be a few good solutions; there could be a few bad solutions also. Now if you find out some good solution. So, in the mating pool actually we are going to take a multiple copies of this particular good solution.

And if you find the bad solution in initial population those solutions we are going to delete from this particular the mating pool. The philosophy behind a going for this mating pool that this particular the mating pool will consists of good solution probabilistically.

Now, if it is a maximization problem. So, if I take the average fitness of this particular mating pool, the average fitness of the mating pool is expected to be more compared to that of this particular the initial population.

Now, if you see the literature we have got a few reproduction scheme for example, the oldest one is nothing, but the proportionate selection or the rule (Refer Time: 05:34) selection or the probability of selecting a particular solution is proportional to the fitness. So, the higher the fitness the more will be the probability of being selecting of being selected in this particular the mating pool; that means, if you find out some solution whose fitness is not good there is a possibility it will be deleted from this particular the mating pool.

Now, this particular the proportionate selection that is the oldest method of reproduction has got one drawback that there is a chance of premature convergence and to remove that particular problem. So, another reproduction scheme was proposed and that is nothing,

but the ranking solution. Now in ranking solution actually what is do is in ranking selection what is do is we do the proportionate selection based on the ranks, but not based the fitness information.

So, based on the rank we do this particular proportionate selection and by doing that we can remove the chance of this premature convergence, but now it is actually we use some sort of tournament selection. And in tournament selection we select a tournament size and those solutions are selected at random from this particular initial population and we try to find out the best and the best one is selected in the mating pool and once we have selected the best one in the mating pool.

So, all the solutions consider in the tournament will be returned back to the initial population and if the population size is capital N . So, we will have to play the tournament for capital N size N times and each time we are going to select a particular the solution.

Now, if you see the computational complexity the computational complexity of this particular your the tournament selection is much less compared to the proportionate selection. And that is why nowadays we use only the tournament selection.

Now, once we have got this particular mating pool next we try to form the mating pair and this particular mating pair selection is done at random.

Now, supposing that we have got n population size. So, there will be N by 2 mating pairs and an each pair there will be 2 such solution and these particular mating pair selection will be at random. Now supposing that, we have got N by 2 mating pairs. Now for each of the mating pair whether it is going to participate in the next operation that is the crossover. So, we try to take the help of one probability that is called the probability of crossover.

Now, depending on the probability of crossover we take the decision whether this particular mating pair is going to participate in crossover or not. Now if it participating crossover there will be exchange of properties and due to this particular crossover operation. So, starting from the 2 parents in fact, we are going to get we are going to get like 2 children.

(Refer Slide Time: 08:54)

The slide is titled "Topic 3: Binary-Coded Genetic Algorithm (BCGA)" in a red box. Below the title, there is a bulleted list of topics:

- Introduction to GAs
- Working Cycle of a GA
- Working Principle of a BCGA

Handwritten in blue ink on the right side of the slide are the following terms: "Single-pt.", "Two-pt.", "Multi-pt.", and "Uniform Mutation". The word "Mutation" is circled in blue.

At the bottom of the slide, there are logos for "IIT KHARAGPUR" and "NPTEL ONLINE CERTIFICATION COURSES".

Now, if you see this particular literature we have got the different types of crossover operators like for example, say we have got the single point crossover.

So, we have got the single point crossover then comes we have got the 2 point crossover, then comes we have got the multi-point crossover, and we have got your the uniform crossover. So, this particular crossover operators we generally use just to find out the children's say solutions from the parents.

And there will be exchange of properties at due to this particular crossover and there is a possibility some new properties are going to come in the children solution.

Now, once I have got this particular children solution next we go for the mutation operator. Now in mutation operator there will be actually sudden change of parameter and this particular mutation operator is going to help us to come out of the local minima if any. And using this particular this mutation operator generally we go for the bitwise mutation and using this particular bitwise mutation. So, there is a possibility that will be able to come out from this particular local minima problem and GA is going to hit the globally optimal solution.

So, this complete actually one iteration of the GA and this particular process will continue for a large number of generation or the large number of iteration, then GA

through a large number of iteration will try to find out what should be the globally optimal solution.

So, this is the way actually one binary coded GA works.

(Refer Slide Time: 10:54)

Topic 3: Binary-Coded Genetic Algorithm (BCGA)

- Introduction to GAs
- Working Cycle of a GA
- Working Principle of a BCGA
- Hand Calculation of BCGA
- GA-parameters setting

Selection Pr.
Pop. diversity
Pc, Pm, N, G

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now if you see the merits and the demerits so, this particular binary coded GA has got a few merits and demerits that I am going to discuss after sometime, but before that let me tell you that we actually carried out some hand calculation just to show you the working principle of a binary coded GA. And by doing that particular hand calculation in fact, we could prove indirectly the GA can optimise the function and GA can find out the optimal solution through a large number of iteration.

Now, if you want to ensure a very efficient search for the GA. So, we will have to select the parameters in an optimal sense and that is why will have to go for this particular the GA parametrics GA parameters setting. Now the performance of a GA depends on a balance between the population diversity and the selection pressure.

So, there are 2 things one is called the selection pressure and another is called the population diversity. Now this population diversity and selection pressure there should be a proper balance if you want to ensure a very efficient search for this particular the GA.

Now, let me see what happens if the selection pressure is too high, if the selection pressure is too high there could be a chance of premature convergent and if the population diversity is too large. So, there is a possibility that there will be enough search, but it may take long time to reach that particular optimal solution.

So, there should be a proper balance between selection pressure and this particular the population diversity in GA. And to ensure that we generally go for some set of parameters setting like, we try to set what should be the optimal parameters like probability of crossover, then probability of mutation, then population size and the maximum number of generation.

Now, we use a particular method of GA parameters setting here one parameter is valued at a time keeping the other parameters constant and this is an approximate way of doing. So, using this approximate method of GA parameter setting we can find out what should be the newer optimal GA parameter.

Now, if you run this particular GA using that particular newer optimal parameters there is a possibility that it will be able to maintain the balance between the selection pressure and the population diversity, and GA will be able to hit that globally optimal solution.

So, this is actually the purpose of the GA parametric study which I have already discussed in much more details.

(Refer Slide Time: 13:55)

The slide features a title box at the top center: **Topic 4: Schema Theorem of BCGA**. Handwritten notes in blue ink are scattered across the slide. On the left, 'Building Block' is written vertically, with 'Schema Template' and 'Schema Growth' written below it. On the right, 'Short defining length' and 'Low order' are written vertically. Below these, 'N Schema fitness' is written. A diagram in the center shows a 2x2 grid of bits with values 10, 01, and -10. The bottom right corner of the slide shows a small video feed of a man in a suit. The footer contains the IIT KHARAGPUR logo and the text 'NPTEL ONLINE CERTIFICATION COURSES'.

Now the next topic is the schema theorem of this particular the binary coded GA. Now actually the purpose of this particular schema theorem was to prove indirectly the convergence of this particular the binary coded GA.

Now, here as there is no such direct mathematical proof till today of this particular the GA; so we took the help of the schema theorem just to indirectly prove that a GA can maximize or minimize GA can optimize that particular the objective function.

Now, here actually what we do is now let me let me just tell you in short in details have already discuss supposing that we have got one population of solution of binary coded GA of size say n and we have got a few binary strings here say large number of binary strings here. And we will try to find out the similarity among this particular the binary string and after watching this particular similarity.

So, we try to design one schema and this schema is nothing, but a template. So, we try to find out a template which is followed by the large number of your the GA string the binary coded GA string. And this particular schema or the template we try to see the growth of the decade through a large number of iteration and mathematically we can actually derive one expression. And that is that expression is known as the schema growth expression the schema growth theorem.

So, this schema growth theorem we can find out one mathematical expression and once I have got that particular mathematical expression. Now, we can take the decision whether a particular schema is going to survive or not and whether this particular schema or the template is going to get a large number of your solutions or not.

Now, this actually mathematical we can find out one expression and from this particular express and we can come to one conclusion that if I have got a schema, which is having short defining length short defining length short defining length then no order and if the average fitness of this particular schema is found to be more than the average fitness for this particular the whole population and that is called the scheme of fitness. So, if the schema fitness is found to be more than the average fitness of this particular the whole population.

So, there is a possibility that I will be able to find out. So, this particular schema will receive the more and more number of copies in the as the iteration proceed. So, if I have

got a schema which is having short defining length low order and fitness is higher than the average fitness of the population that particular schema is known as the building block.

So, this is nothing, but the building block for this particular your GA search and this building block is going to gain a more and more number, if it is a good building block it is going to gain in number through a large number of iteration and that is going to win at the end and this particular building block is going to indicate what should be the globally optimal solution.

So, using this schema theorem or the building block hypothesis so, we can say that if there is a good schema. So, that particular good schema is going to gain in number through a large number of iteration and that is going to dictate the whole population and that is going to indicate are going to give the globally optimal solution. Now this particular schema theorem is actually an indirect proof or the convergence of this particular the binary coded GA.

Now, here this binary coded GA has got the schema theorem this is a plus point because many people did not initially believe the binary coded GA, but after the schema theorem was proposed people started believing so, this particular the binary coded GA.

(Refer Slide Time: 18:27)

Topic 4: Schema Theorem of BCGA

- Building-Block Hypothesis
- Numerical Example
- Limitations of BCGA

Hamming
Stiff.

Gray coded

$y = f(x_1) \dots x_{10}$

$20 \times 10 = 200$

$N = 1000$

\log_2

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now here this building block hypothesis I discuss in details I also solved one numerical example like how to find out the defining length? How to find out the order of a particular schema?

How to find out the schema fitness and how to find out whether a particular schema will receive more number of copy or it is going to decay? So, that type of decision we can take; so just to explain that I took the help of one numerical example in this particular the course.

Now, I am just going to concentrate little bit on the limitation of these particular the binary coded GA. Now we have already discussed that the accuracy depends on accuracy depends on the number of bits assigned to represent a particular the variable now if you need more accuracy.

So, we will have to assign a large number of bits and supposing that I have got a function which is to be optimised. So, y is a function of say 10 variables x_1 x_2 up to say x_{10} . So, I have got 10 variables and I want a very good precision in each of this particular say 10 variables.

That means we will have to assign a large number of bits might be say 20 30 or 40 bits. Now if I assign say 20 bits to represent each of the variable. So, that 10 such variable. So, 10 multiplied by 20. So, your GA string will be 200 bits long. Now if the GA string is longer and for it is effective processing so, I will have to use a large population size. So, might be if it is 200 I will have to take n equals to say 1000.

So, there will be huge competition and GA will become very complex computationally very expensive and it will become slower and slower. And that is why because this type of algorithm the binary coded GA the computational complexity is actually $L \log L$.

So, L multiplied by $\log L$ is the computational complexity or this capital L is nothing, but the total string length and if L is large. So, the computational complexity is going to increase and just to overcome; so this type of problem. So, actually we try to take the help of your the real coded GA. So, the binary coded GA is not going to help us to achieve the arbitrary precision which we need in the values of the variable.

So, this is one of the drawbacks of this particular the binary coded GA. And another very important drawback is nothing, but the hamming clip problem now this hamming clip problem is actually a very difficult problem faced by this binary coded GA because in binary codes from one number if I want to move to the next number like or the previous number the number of change which I will have to consider in the values of the bits is not kept the same.

For example say let me take a very simple example say 15 to 16 if I want to go and 15 to 14 if I want to go. The number of change to be incorporated in the binary representation is not exactly the same. So, I will have to go for the different number of changes.

If I want to move from a particular number to the previous number or to the next number so, actually so, this particular the problem is known as the hamming clip problem of the binary codes. Now this hamming clip problem is going to create some sort of artificial hindrance to the gradual search of this particular the genetic algorithm.

Now, if the GA search is proper. So, this particular search has to be gradual and there should not be any abrupt search and that is why so, this particular hamming clip problem of the binary coded GA is not desirable. And to solve this particular hamming clip problem actually we take the help of another type of GA that is called the Gray code GA. So, it take the help of this gray coded GA now this particular gray coded GA is going to help us just to remove this particular the hamming clip problem of the binary coded GA.

Now, here are actually this real coded GA, I have discussed in much more detail, but the gray coded GA most probably I did not discuss in detail, but it is very simple only thing the design variable should be expressed in terms of gray codes in place of the binary code; other things will remain the same.

(Refer Slide Time: 24:05)

The slide is titled "Topic 5: Constraints Handling in GA" in a green box. It contains two bullet points: "Penalty Function Approach (Static, Dynamic and Adaptive penalties)" and "Numerical Example". Handwritten blue ink notes include "Minimize", "Maximize", and "Minimize" with arrows pointing to the objective function $y = f(x_1, x_2)$. A constraint equation $x_1 + 2x_2 - x_2 \geq 50$ is written in blue ink. The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES at the bottom.

Now, the next is actually we discussed little bit on the constant handling in genetic algorithm.

Now, genetic algorithm can solve actually both the unconstrained as well as the constrained optimisation problem. Now here if you want to solve the constrained optimisation problem. So, will have to take care of this particular the functional constrained.

Now, this functional constrained actually it could be non-linear function of the design variable, it could be the linear function of the design variable, and it could be either or equality some or it could be inequality functional constrained.

So, all the functional constrained actually will have to consider and GA will have to find out one optimal solution, which is visible in the sense that it is not going to violate any of this particular the functional constrained.

Now, to ensure that actually what we do is we take the help of one approach and that is called the penalty function approach. Now this penalty function approach in fact, I have discussed in much more detail and I have solved one numerical example. Now in sort let me tell you that in sort let me tell you like how to implement this particular the penalty function approach.

Now, supposing that say I have got a function say y is a function of say 2 variable for simplicity let me consider the this is a function of only 2 variable. Now using these 2 variables I have got some say objective functions say this is the objective function and I have got a few functional constrained.

For example, so, let me take a very hypothetical functional constrained like $x_1^2 + x_1 x_2 - x_2^2$ should be greater than or equals to 5.0. So, this is a functional constraint non-linear functional constraint inequality functional constraint.

Now, if I want to solve with the help of a genetic algorithm. So, genetic algorithm will have to find out those values of x_1 and x_2 . So, that is satisfied this particular the functional constraint and if it does not satisfy. So, this particular functional constrained. So, that will be declared as the invalid solution and that is not be will not be considered as valid optimal solution.

Now, how to how to ensure and how to penalize? If there is any such violation of this particular the functional constraint. Now supposing that I am solving say one say minimization problem say, and I have already got the objective function the value of the objective function is a small f and I am just going to solve one minimization problem; that means, lower the value of the objective function. So, better will be the solution.

Now, here if there is any such violation of the functional constrain. So, I calculate the modified fitness capital F and that is nothing, but small f plus a penalty term P , if it is a minimization problem, and if it is a maximization problem. So, this particular the penalty term has to be. So, this is for minimization problem this is for minimization problem and F is nothing, but small f minus P it is for the maximization problem.

Now, how to calculate this particular penalty? So, this penalty is calculated using the different approaches. Now, one is called the static penalty, another is called the dynamic penalty and we have got the concept of adaptive penalty.

Now, in static penalty actually what we do is we use some constant penalty value, generally we considered some high value and this particular high value is calculated with the help of some user defined the constant terms the in details those mathematical expression I have already discussed.

So, I am not going for that and this dynamic penalty and this adaptive penalty are going to vary it is going to vary proper example if you see this dynamic penalty. So, it is going to vary from iteration to iteration and to calculate the penalty term in dynamic penalty approach.

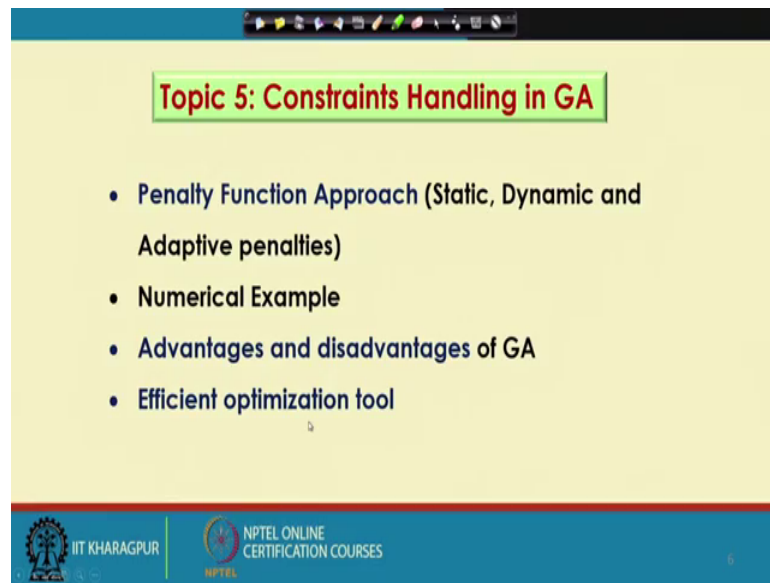
So, we will have to include the iteration number also. So, this particular penalty term is going to be updated with the number of iterations. Now if I consider this adaptive penalty. So, here the concept is slightly different. Now in calculating the adaptive penalty term; so it will depend on the nature of this particular your your this thing nature of this functional constraint also.

So, here if there is no such violation. So, there will be one penalty term which is having some low value, but if there is a violation. So, there will be a penalty term which is having some high value. On the other hand if I go for the static penalty and the dynamic penalty. So, if there is no violation the penalty term will be put equal to 0, now using this particular penalty term. So, I can find out what is this modified fitness that is capital F.

Now, we will have to go for reproduction based on this particular the modified penalty this modified fitness. And based on the modified fitness we go for the reproduction scheme just to get this particular the mating pool. And once you got this particular mating pool, then we will go for your the crossover mutation and others and that will complete the iterations of the GA and by falling this particular method through a large number of iteration.

So, we will be able to find out the GA will be able to find out what should be the globally optimal solution. Now this is the way actually we can use the penalty function approach to handle the functional constraint in this particular the genetic algorithm.

(Refer Slide Time: 30:52)



The slide is titled "Topic 5: Constraints Handling in GA" in a red box. Below the title is a bulleted list of four items: "Penalty Function Approach (Static, Dynamic and Adaptive penalties)", "Numerical Example", "Advantages and disadvantages of GA", and "Efficient optimization tool". At the bottom of the slide, there are logos for IIT Kharagpur and NPTEL Online Certification Courses.

Now, this genetic algorithm has got a few advantages and disadvantages.

Now, it has got an advantages for example, so, it can tackle a variety of problem. So, it is much robust compared to the traditional tools for optimisation. And so, if there is any discontinued of the objective function so, it can handle because it does not required the gradient information of this particular your objective function. And there is another big advantage that we can go for parallel computing as this is a population based approach.

So, easily we can implied implement that parallel computing with the help of this MPL algorithm. Now it has got a few drawbacks for example, the genetic algorithm is computationally very expensive it is low. And in fact, it is bit difficult to implement online if you want to get the optimal solution within a fraction of second. So, the conventional genetic algorithm cannot be used.

And this genetic algorithm is actually a black box set of thing for example; say if you want to solve one maximization problem or minimization problem.

Now, on the computer program of the genetic algorithm, we just a write down the expression of the objective function, we give the range of the variables and we also decide all the GA parameter and along GA then after a large number of iteration there is a possibility the GA will be able to get that globally optimal solution, but in fact, the user

may not know actually what is happening inside the GA and that is why we called the genetic algorithm is nothing, but a black box.

Now, these are all disadvantages so, there are some merits and demerits and these are all disadvantages of this particular the genetic algorithm. Now considering the weakness of this particular genetic algorithm that is the genetic algorithm cannot handle the local optimisation very efficiently, we take the help of on hybrid optimisation tool and that is actually efficient optimisation tool.

Now, sometimes we will have to use this type of efficient optimisation tool which is nothing, but a hybrid optimisation tool. Now as we note the genetic algorithm is actually a very powerful search technique or global optimisation, but it is local search capabilities not so, good and that is why we combine the global search capability of genetic algorithm along with the local search capability of a local search technique likes a steepest descent algorithm, just to efficiently solve that optimisation problem. And that is why the concept of this efficient hybrid optimisation tool came into the picture.

Thank you.