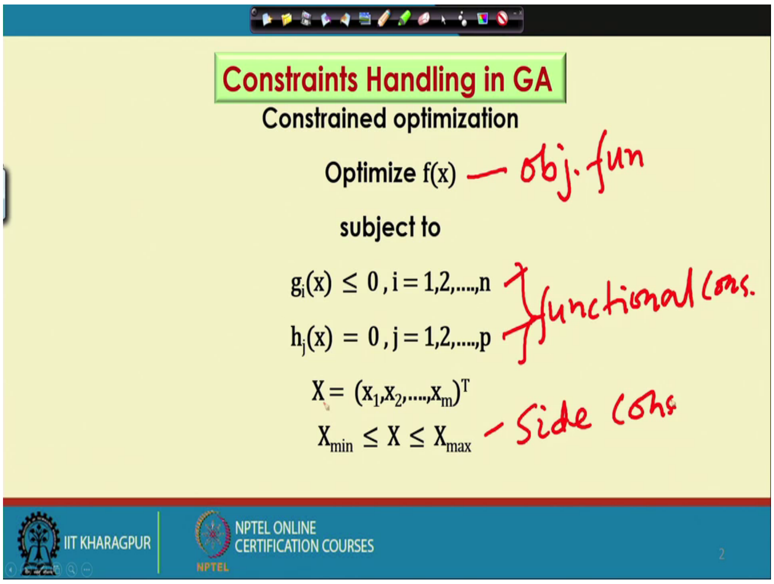
Traditional and Non-Traditional Optimization Tools
Prof. D. K. Pratihar
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture - 11
Constraint Handling

Now, I am going to start with the next topic that is topic 5, Constraints Handling in GA. Now, GA actually used to solve the different types of optimization problems. Now, in most of the real world problems actually we will have to handle some sort of constrained optimization problem; that means, there will be some function of constraints as well as objective function and the side constraints or the geometric constraints. Now, let us take one example of a constrained optimization problem.

(Refer Slide Time: 00:54)

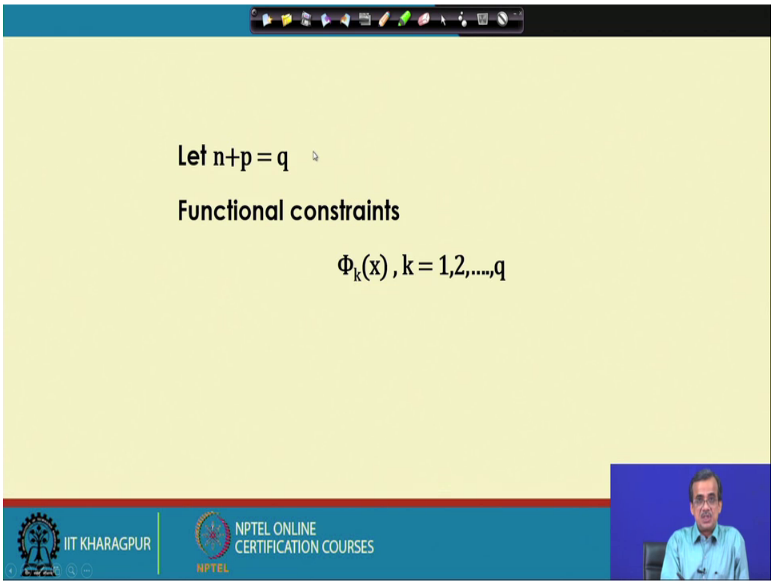


Now, supposing that I have got a function y equals to f x and this I will have to optimize say either maximize or minimize. Now, subject to some functional constraints like g i x is less than equals to 0 or i varies from 1 to up to n. So, this is one functional constraint and this is the inequality constraint.

Next is we have got h j x equals to 0, j varies from 1 2 up to p. Now, this is actually one equality constraint. So, there are nothing but the functional constraints, these are the functional constraints the functional constraint and here, this is the objective function the objective function and we have got the variable bounds that x is lying between x minimum and x maximum. So, this is nothing but the side constrained or the geometric constrained side constrained or the geometric constrained and capital X is nothing, but is a collection of all small x values like x 1 x 2 upto x m. So, this is actually the constrained optimization problem.

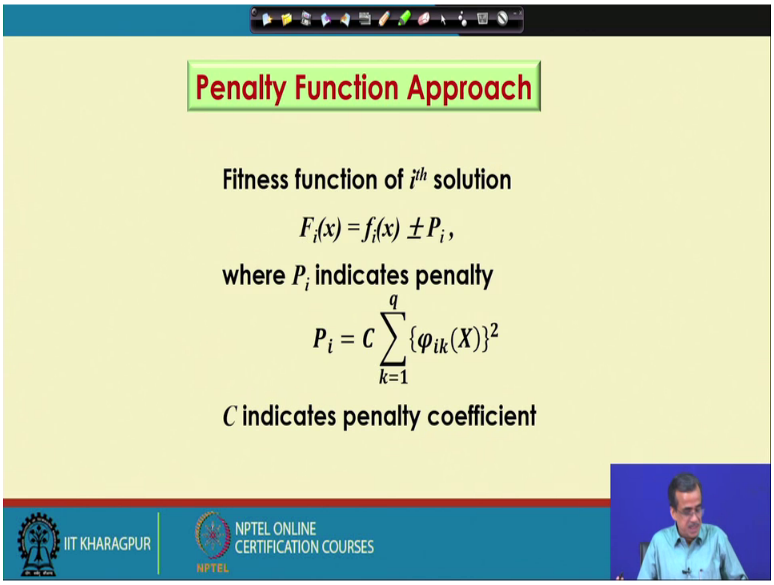
Now, let me discuss how to tackle, how to handle this type of constrained optimization problem.

(Refer Slide Time: 02:31)



Now, here we have got small n number of inequality constraint and small p number of equality constraint. So, the total number of constraint is n plus p that is equals to q and the functional constraints, these are represented as phi k x k varies from 1 2 up to q. So, this phi k x is nothing but the collection of all functional constraints.

(Refer Slide Time: 03:04)



Now, to handle this constraint optimization problem we take the help of an approach which is called the penalty function approach. Now, let us see how to use this to tackle this type of constrained optimization problem. Now, supposing that I have got 1 objective function that is small f i x and the variable are such that the functional constraint or some of the functional constraint are going to be violated. So, if there is any such violation of the functional constraint, so I will have to penalize so that particular the solution by the penalty term that is nothing, but p i.

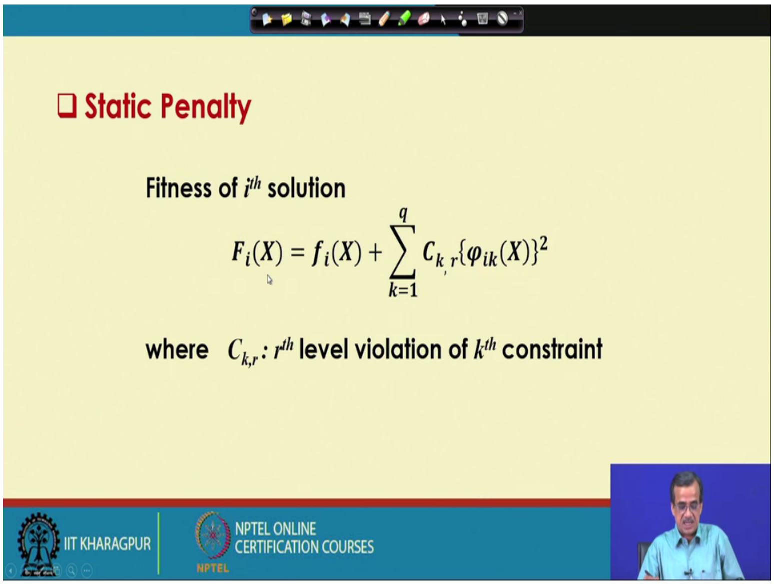
So, I am just going to concentrate on the ith solution or the ith GA string and corresponding to the ith solution the fitness value is say f i x and the penalty is P i and here I am putting, plus minus the reason I am going to tell you and the modified fitness is nothing, but capital f i x is nothing but smaller f i x plus minus P i and P i indicates the penalty. Now, this particular P i is having positive value and supposing that I am solving 1 maximization problem. Now, for solving the maximization problem, this penalty term has to be subtracted on the other hand if I want to solve 1 minimization problem. So, I will have to add this particular the penalty term.

Moreover if a particular functional constraint is not going to violative, not going to be violated or if it satisfies a particular functional constraint for that the penalty term should be equal to 0. So, if there is any violation of the functional constraint I will have to add or subtract the penalty depending on whether it is the maximization or minimization problem and if there is no such violation of the functional constraint, we will have to put this particular penalty term equal to 0.

Now, let us see how to calculate, this particular the penalty term. Now, here I am just going to write down one expression to calculate the penalty that is P i that is the penalty for the ith string, the ith solution is nothing but C multiplied by summation k equals to 1 to q phi i k x square. Now, here, this particular C is actually a constant and the numerical value for this constant has to be decided by the user and small q indicates the total number of functional constraints and here the phi i k x, this phi i k x actually it indicates how much is the violation of this particular the functional constraint.

So, what I do is we are going to find out the left hand side value and the right hand side value for a particular functional constraint and by subtracting we try to find out how much is this particular the violation. Now, this particular phi i k x it could be either positive or negative and that is why to make it positive we use this particular the square term and sometimes we also use the mod value to get this particular positive value in place of this particular the square term. Now, let us see how to calculate how to find out this particular penalty value using different approaches. So, I am going to discuss a few approaches like how to calculate this particular the P i value.

(Refer Slide Time: 07:28)



Now, here actually what I do is we are going to use the concept of the static penalty first. Now, here in static penalty what I do, we try to use this particular expression to find out what should be the modified fitness. So, corresponding to ith solution the modified fitness capital F i x is nothing, but small f i x plus summation k equals to 1 to q, C k r that is the constant term multiplied by phi i k x square.

Now, here actually the C k r is the rth level violation of kth constraint. Now, this particular numerical value will have to define will have to supply before we start with the calculation of this particular the modified fitness that is capital F i x. Now, how to assign this particular value that is C k r that I am going to discuss.

Now, to discuss this let me take one very practical example. Now, this particular example is as follows. Now, supposing that I am going to design shaft for a particular fan say I am considering 2 fans, one is the ceiling fan another is the that table fan sort of thing. Now, supposing that the blades of this particular fan will be supported by a shaft. Now, I am just going to design the shaft for the ceiling fan and I am also going to design the shaft for the table fan.

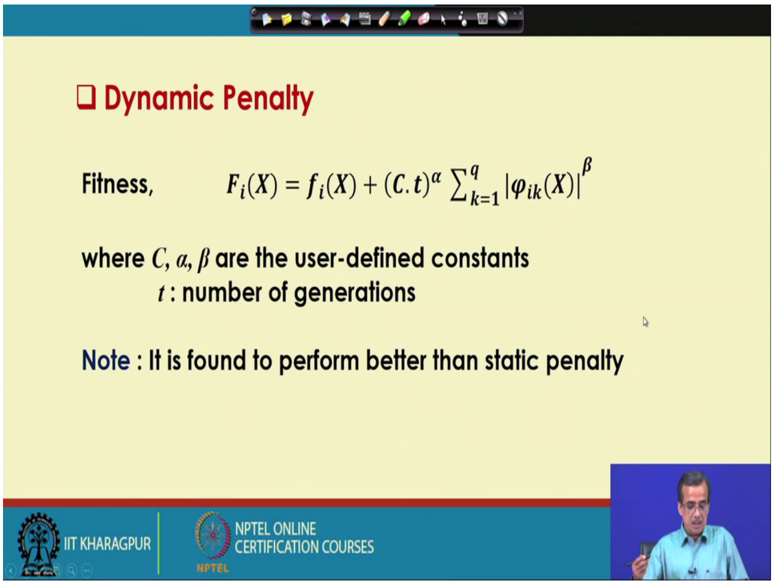
Now, while designing this will have to consider one fact that if there is any failure in the shaft of the ceiling fan there could be fatal accident the person we sitting below may die. On the other hand if there is any failure on the shaft of this table fan there would be an accident, but it may not be so much dangerous so the person who is sitting in front of the fan may get hurt, but he or she may not die.

Now, if I want to design these 2 shafts for 2 different fans; I will have to consider 2 different values for this particular the C k r. Now, this is the way depending on the practical situations or the requirement the user will have to select what should be the suitable value for this particular the C k r; that means, supposing that I have got 2 functional constraints out of these 2 functional constraints we will have to find out which one is more important whose violation cannot be tolerated so easily and if this is the situation.

So, if one is more important another is less important we can assign 2 different numerical values for this particular C k r. So, once we have selected the numerical values for this particular C k r. Now, very easily you can find out what should be this particular the amount; that means, we can find out this particular the numerical value and I can also calculate what should be this particular the modified fitness.

Now, after sometime I am just going to discuss one numerical example just show you how to calculate this penalty term numerically.

(Refer Slide Time: 11:19)

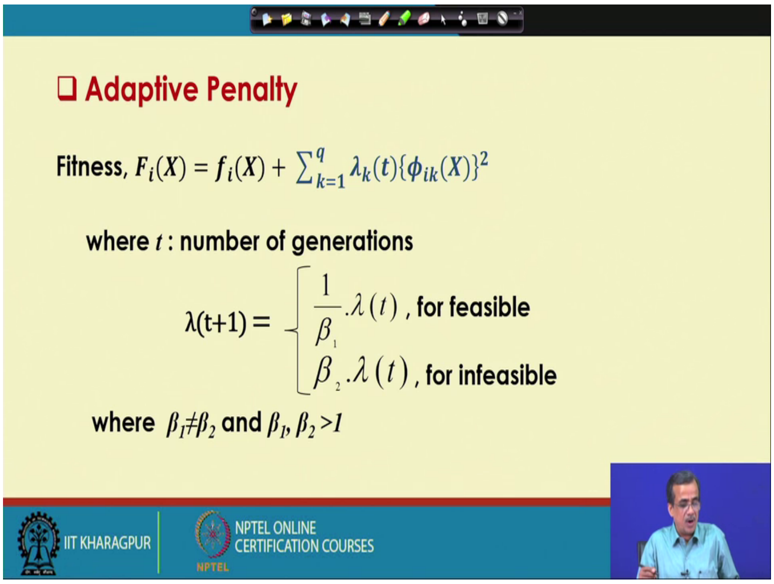


Now, then comes the concept of the dynamic penalty. Now, here actually what you do is. So, this penalty term; that means, this particular term is going to vary with the number of iterations. Now, let us see how does it work. So, capital F i x is the modified fitness that is nothing, but small f i x is the original fitness plus C t raise to the power alpha multiplied by summation k equals to 1 to q mod value phi i k x raise to the power beta. Now, here I consider the mod value just to consider the positive value for this particular the deviation and this C is actually a constant user defined constant alpha is another constant beta is another constant and t indicates the number of iteration.

Now, here as t increases what will happen to this particular penalty term? The penalty term is going to increase the numerical value of this penalty term is going to increase. It means that at the beginning we put less amount of this penalty if there is a violation, but with the number of iterations the amount of penalty which we are going to put that is increased; that means, initially we may allow a few in feasible solutions, but with time with iteration we should be more strict and will have to put more penalty for the infeasible solution so that in ultimately we should not get any in feasible solution in the population of the GA.

Now, once again I will be solving a numerical example to show the importance of this particular the penalty calculation. Now, here I have put one note. So, this dynamic penalty is found to perform better than the static penalty as usual.

(Refer Slide Time: 13:36)



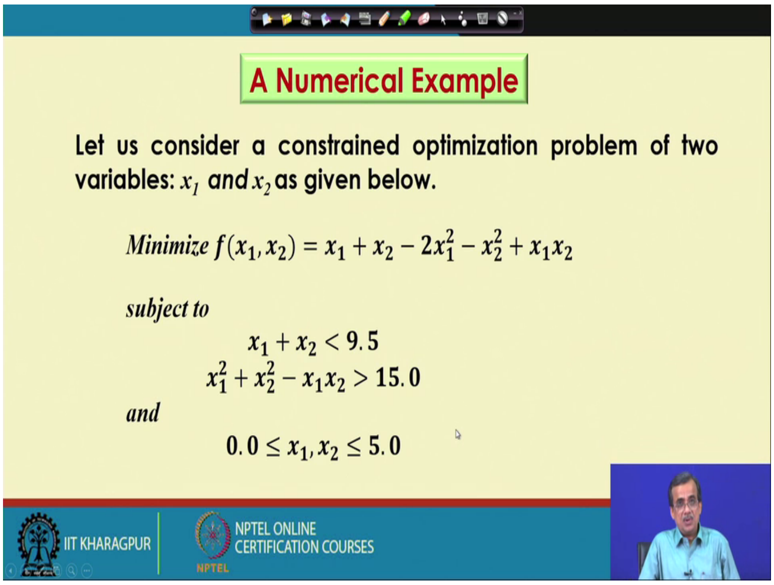
Now, then comes the concept of this adaptive penalty. Now, here actually what I do is we calculate the modified fitness capital F i x that is nothing, but small f i x plus summation k equals to 1 to q. Then comes lambda k t multiplied by phi i k x square. So, this is nothing, but the amount of violation square of that and this lambda k t we will have to define, using this conditions. Now, let us say how to define this particular lambda k t. So, t is nothing but the number of iterations.

So, lambda t plus 1 that; that means, the value of lambda in the iteration is nothing, but 1 divided by beta 1 multiplied by lambda t for feasible solution; that means, if there is no violation of the functional constraint, so this is the expression which I will have to use and if there is a violation; that means, for infeasible solution. So, I will have to use lambda t plus 1 is nothing, but beta 2 multiplied by lambda t, where this beta 1 beta 2, where this beta 1 beta 2 are greater than 1 and beta 1 is not equals to beta 2.

Now, here I just want to mention one thing according to this adaptive penalty supposing that a particular functional constraint is going to be violated, now for this particular infeasible solution we put high penalty value and even if for feasible solution when there is no such violation of the functional constraint we put a small penalty value instead of putting 2 equal to 0. So, in place of feasible solution, actually instead of putting penalty equals to 0 we put some small value. So, this matter is slightly different from the dynamic penalty approach and static penalty approach.

Now, I am just going to concentrate on a numerical example just to explain the method of this particular the penalty function approach.

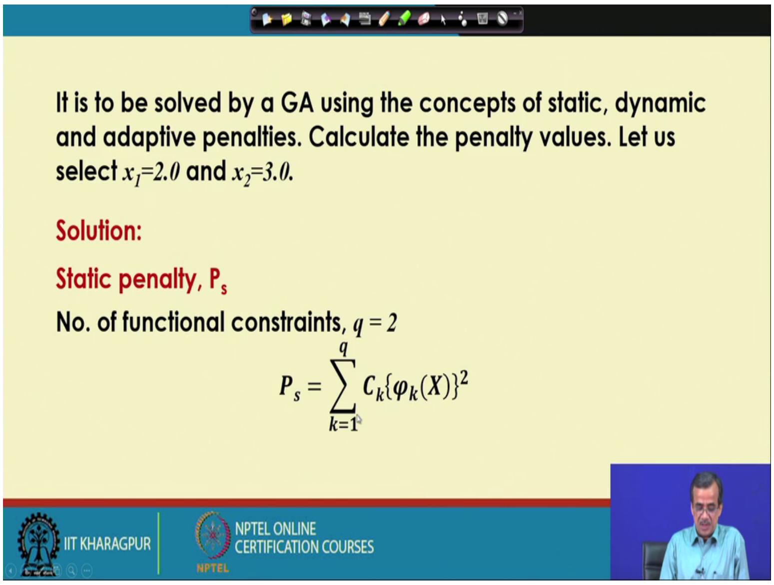
(Refer Slide Time: 16:17)



Now, I am just going to consider a constrained optimization problem of 2 variables x 1 and x 2. Now, the objective function is as follows minimize f of x 1 comma x 2 is nothing, but x 1 plus x 2 minus 2 x 1 square minus x 2 square plus x 1 x 2. So, this is the objective function subject to x 1 plus x 2 less than 9.5 and x 1 square plus x 2 square minus x 1 x 2 is greater than 15.0 and x 1 x 2 align between 0 and 5. So, this is the objective function, these 2 are the functional constraint and this is nothing but the side constraint or the variable bounds.

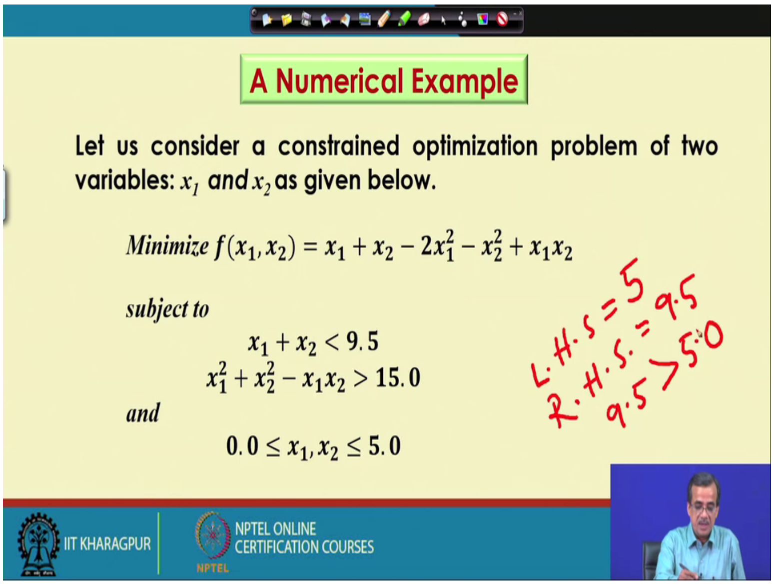
Now, let us see how to calculate the penalty terms using the concept of static penalty approach, then comes dynamic penalty approach and adaptive penalty approach.

(Refer Slide Time: 17:37)



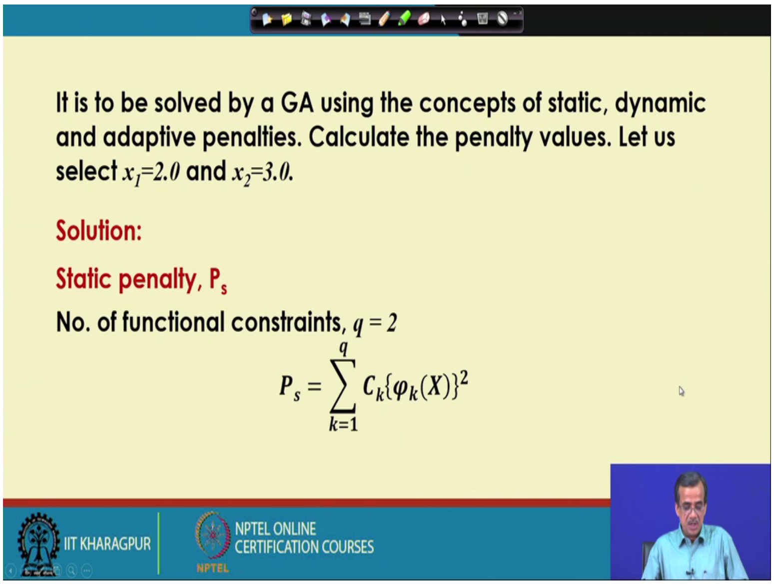
Now, let me consider one set of values of the variables like x 1 equals to 2.0 and x 2 equals to 3.0. Now, what I do is in static penalty, here actually we have got 2 functional constraint, q is equals to 2 and the static penalty which is denoted by p s is nothing but summation k equals to 1 to q C k phi k x square.

(Refer Slide Time: 18:10)



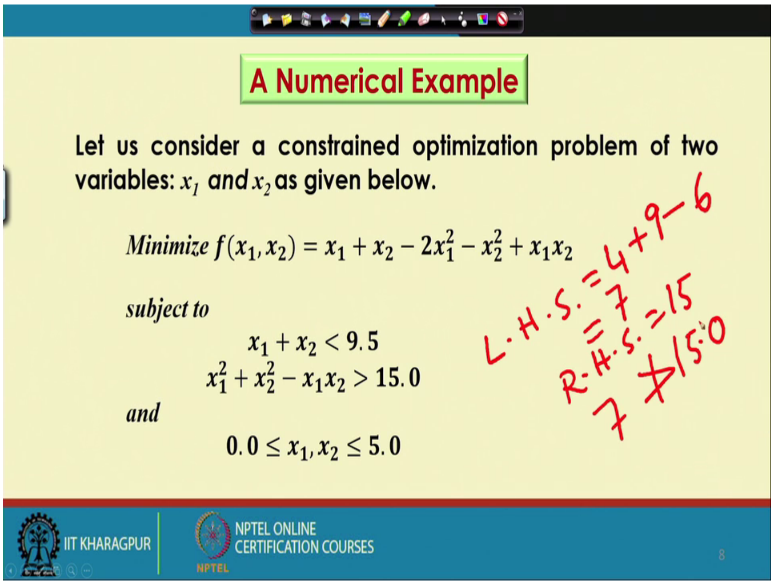
Now, let me assume that C 1 equals to C 2 equals to 10 here the first functional constraint is not violated. Now, let us check the first functional constraint, the first functional constraint that is nothing, but x 1 plus x 2 is less than 9.5. So, here if I calculate the value for this particular left hand side corresponding the first functional constraint that is left hand side is nothing but x 1 plus x 2 that is 2 plus 3, 5 and the right hand side that is your 9.5 and 9.5 is greater than 5.0. So, there is no violation of this particular the functional constraint.

(Refer Slide Time: 19:18)



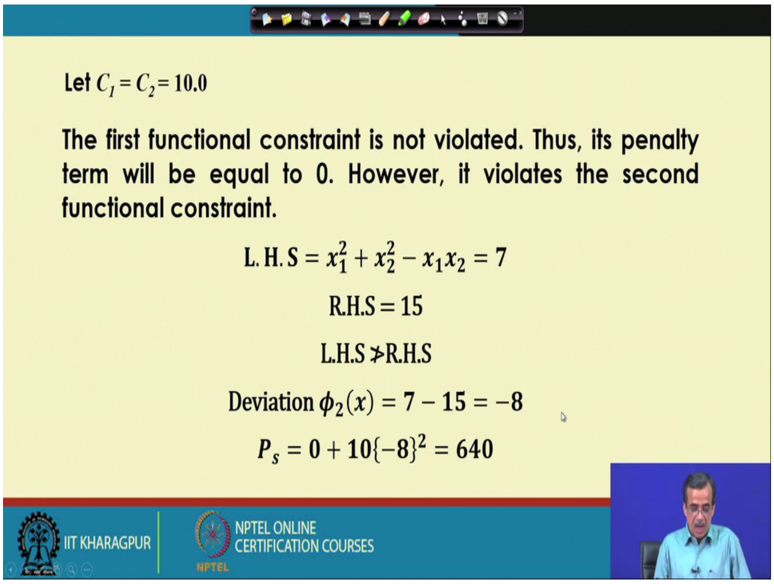
And as I mentioned that if there is no such violation of the functional constraint so we put the penalty term equals to 0.

(Refer Slide Time: 19:22)



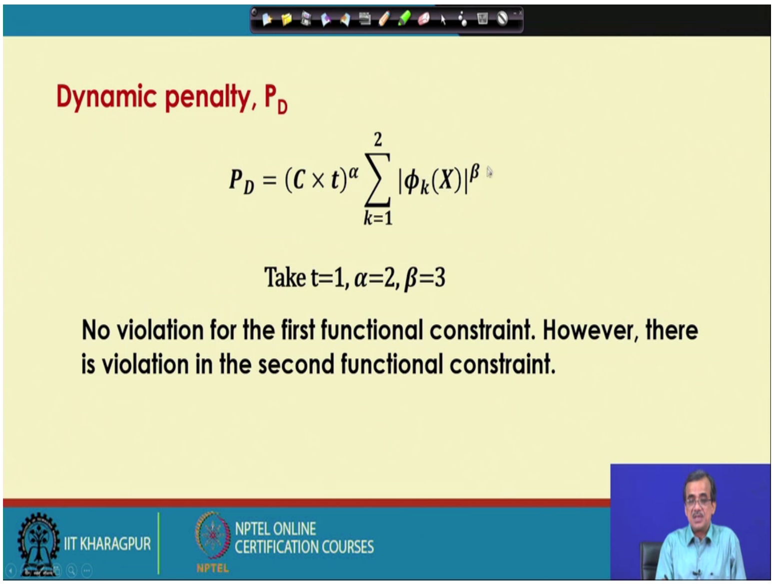
But for the second functional constraint there is a violation just to check that once again let me concentrate on the second, your this functional constraint. There is a left hand side that is the left hand side for the fun second functional constraint is nothing but x 1 square that is 4 plus x 2 square that is 9 minus 6, so this is nothing, but 13 minus 6 that is 7 and the right hand side is equal to 15 and 7 cannot be greater than 15. So, it is going to violate the second constraint, second functional constraint. Now, for this particular violation actually what will have to do is, so we will have to find out how much will be the penalty value.

(Refer Slide Time: 20:21)



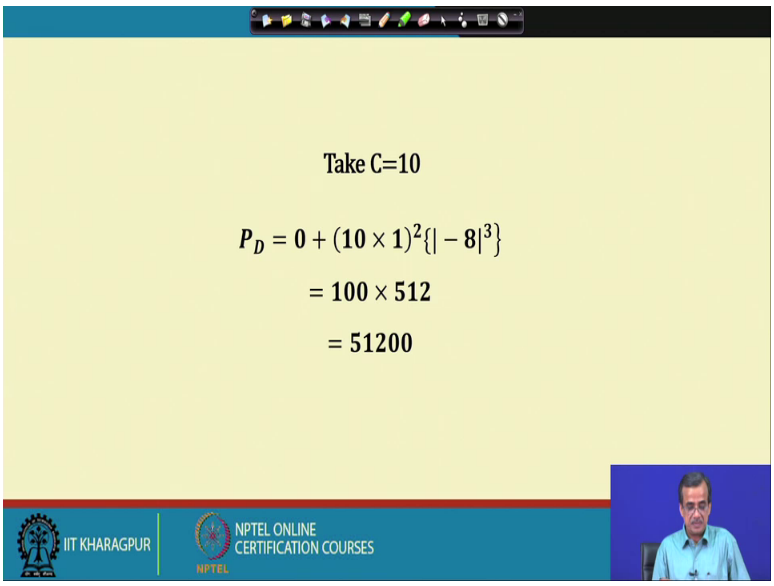
Now, penalty value for this particular the second functional constraint is something like this here we can find out the deviation that is 7 minus 15 is the deviation because left hand side was 7 right hand side was 15 so deviation is minus 8. So, I can find out the amount of static penalty that is equals to 0 for the first functional constraint because there is no violation there and for the second functional constraint this is 10 multiplied by minus 8 square and we will be getting 640. Now this is actually the amount of the static penalty. So, this is the way actually will have to calculate how much will be the penalty value using the concept of the static penalty approach.

(Refer Slide Time: 21:17)



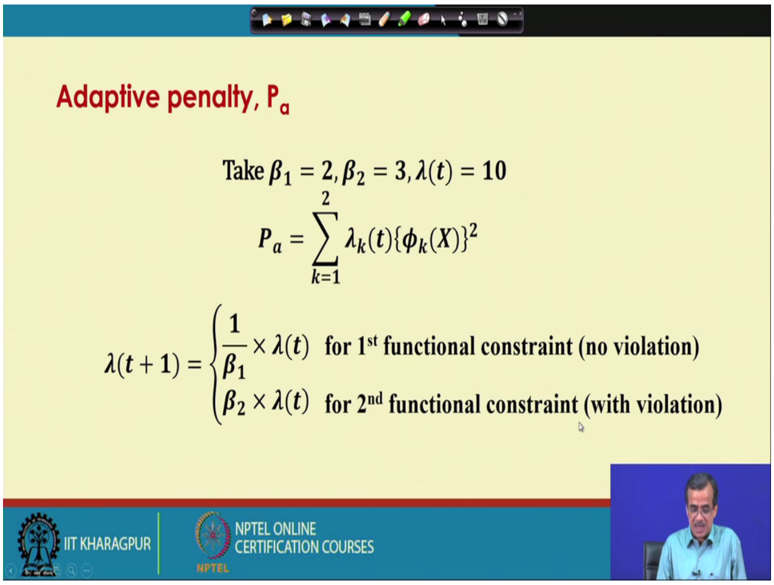
Now, I am going to discuss the concept of the dynamic penalty. The dynamic penalty is calculated as follows like P D is nothing, but C multiplied by t raised to the power alpha summation k equals to 1 to 2 mod value phi k x raise to the power beta. Now, let me assume that t equals to 1 that is the next generation, alpha equals to 2, beta equals to 3. Now, using this actually we can find out what should be the penalty term for the second functional constraint. And for the first functional constraint there is no violation so the penalty term will become equal to 0.

(Refer Slide Time: 22:09)



So, we take C equals to 10 and now I am just going to put the numerical values for all these symbols. Now, if I put the numerical value like C is equals to 10 I am putting and t equals to 1, alpha is equals to 2, beta is equals to 3 and the amount of violation also I know that is minus 8. So, if I put all such numerical values, I will be able to find out what should be the value for this dynamic penalty and which is coming to be equal to 51200. So, this is the amount of the dynamic penalty.

(Refer Slide Time: 22:54)

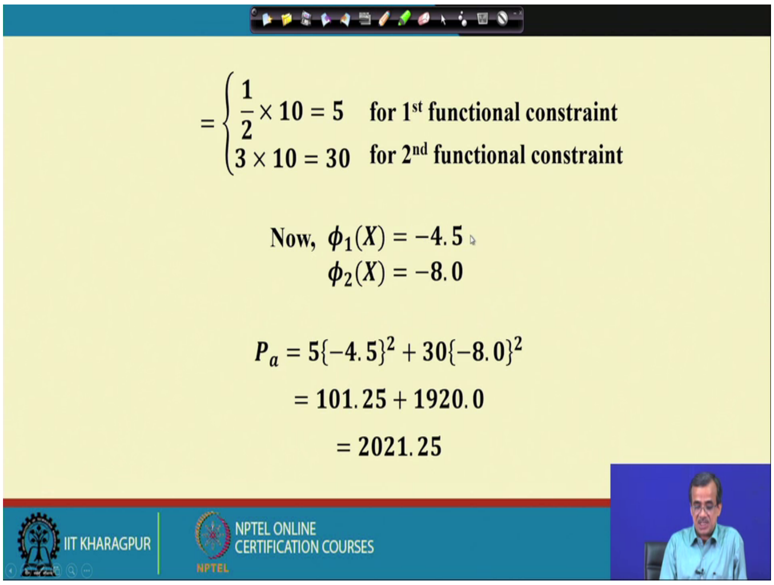


Now, let us see how to calculate the adoptive penalty for the same constrained optimization problem. Now, we take beta 1 equals to 2, beta 2 equals to 3 and let we consider that lambda t is equals to 10 and using this particular expression. So, we will have to calculate this amount of adaptive penalty that is P a is nothing but summation k equals to 1 to 2 lambda k t multiplied by phi k x square.

Now, here as I told that if there is violation definitely I will have to find out the penalty term, but whenever there is no such violation then also I will have to calculate this particular the penalty term. And according to this particular formula like lambda t plus 1 is nothing, but 1 by beta 1 into lambda t for the first functional constraint because here there is no violation of the constraint and I will have to use for the second functional constraint, this particular expression that is beta 2 multiplied by lambda t because here there is a violation of functional constraint.

Now, if I put the numerical values for the first functional constraint, I will be getting half multiplied by 10. Now, from where I am getting half, beta 1 equals to 2, so 1 divided by 2 multiplied by 10 and here for the second functional constraint, this is 3 multiplied by 10, so 30.

(Refer Slide Time: 24:33)



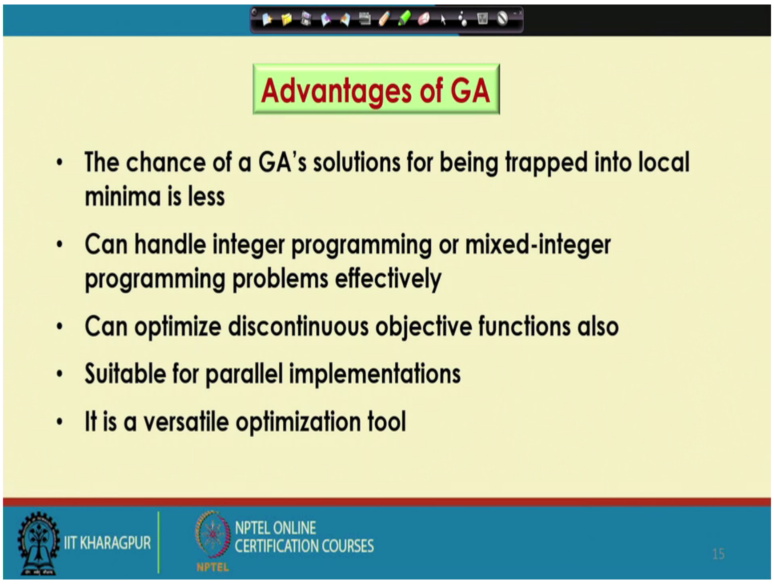
Now, using this I can find out what should be the penalty term, but before that I will have to find out the amount of violation. Now, if I just compare the left hand side and the right hand side, for the first functional constraint although there is no violation of the functional constraint the difference between the left hand side and the right hand side is found to be minus 4.5 and for the second functional constraint the difference between the left hand side and the right hand side is minus 8.0.

Now, using this concept of adaptive penalty I can find out that is 5 that is this lambda multiplied by minus 4.5 square plus thirty is this particular lambda into minus 8 square and if you calculate, for the first functional constraint, I will be getting the penalty term something like this and for the second functional constraint I will be getting this is the penalty term.

Now, if I compare these 2 penalty term. So, for the functional constraint where there is violation the amount a penalty will be higher compared to the other where there is no violation. Now, if I just add them up I will be getting the amount of total penalty like this. So, this is the way actually will have to calculate the penalty term.

Now, once I have got this particular the penalty term. So, I will have to calculate the modified fitness and based on the modified fitness. Now, I will have to carry out the reproduction scheme or the selection scheme in genetic algorithm. So, that depending on the modified fitness the reproduction scheme selects good solutions for the mating pool. We have discussed the working principle of a binary coded GA and now, I am just going to concentrate on the merits and demerits of genetic algorithm. Now, one good advantage of this particular genetic algorithm is it is a robust optimization tool it can solve a variety of optimization problems.

(Refer Slide Time: 27:18)



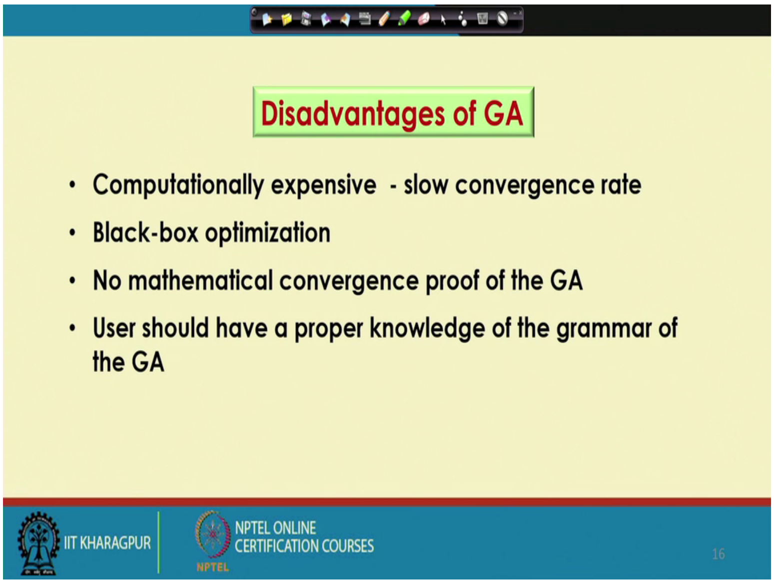
Now, here the chance of the GA solution for being trapped into the local minima is less because this is the population based approach. We start with a number of solution selected at random and moreover actually what I do here we use the operator like mutation which also help us just to come out of the local minima problem. So, here in GA there is a chance that will be getting the globally optimal solution.

Now, genetic algorithm can handle the integer programming problems and mixed integer programming problems; that means, when the variables could be of integer and real and sometimes we will find that there could be a good combination of the real and integer variables. Now, for handling that type of optimization problem, GA is found to be very efficient. Now, for the discontinuous objective function GA can perform optimization because GA does not require the gradient information of the objective function. And as I have already mentioned that we face difficulty with the traditional tools for optimization for their parallel implementation.

Now, as GA starts with a population of solution there is a possibility we can implement genetic algorithm in the parallel platform the whole population is divided into say either 3 or say more than 3 groups and those are assigned 2 3 or more than 3 CPUs and there will be competition and there will be the message passing instruction codes running there and it helps just to do the interaction between the 2 pieces or among, more than 2 pieces just to collect the information.

And consequently the effective competition time to solve a complex optimization problem will be reduced and GA has got the advantage here we can do this parallel implementation very easily because this is a population based search and optimization tool. And as I told that this is very robust GA is very robust and it can solve a variety of problems.

(Refer Slide Time: 30:07)



However, this particular GA has got a few demerits. Now, GA is computationally expensive and consequently there will be slow convergence rate. Now, here we are going to face one problem supposing that we want to carry out some sort optimization online, but this GA is going to take a long time to give that particular optimal solution. So, there is a problem we cannot implement or we cannot use this particular GA for online optimization and GA is actually a black box optimization.

Now, let us try to recapitulate the way we run GA to solve a particular problem. So, once I have got a problem once the mathematical formulation of the problem is with us. We know what is the objective function, what are the functional constraint, what are the variable bounds and accordingly what I do is in the code of GA we write down the expression of the objective function we set the range for the variables and we select the GA parameters like what should be the population size, what should be the probability of crossover, what should be the probability of mutation and so on. And once I have selected the GA parameters now we allow the GA to run, the GA will run for a large number of iteration and ultimately you will be getting that particular optimal solution.

And truly speaking we do not know what is happening inside the GA, but if I wait for a few minutes or few hours ultimately you will be getting that optimal solution; that means, GA works like a black box, the user actually does not know what is happening inside the GA. But here I just want to put one comment. Nowadays actually we generally do not consider GA as a black box optimization. Now, I will be discussing in details after sometime and I will show you that we can also investigate what is happening inside the GA during this particular optimization that I will be discussing after sometime.

Now, here actually in GA there is no mathematical convergence proof and that is why many people actually do not believe GA, but there are some indirect proof like the proof using the schema theorem which I have already discussed or with the help of some hand calculation I have shown that GA can improve the solution and it can find out the optimal solution. So, these are all indirect proof that GA can find out the optimal solution, but till now unfortunately there is no concrete mathematical convergence proof for this particular the genetic algorithm.

And another thing I just want to put which is very important the user should know the grammar of GA and unless he or she knows the grammar. So, he or she will not be able to set the GA parameters properly and if the parameters are not set properly there is no guarantee the GA will be able to hit the globally optimal solution. So, these are all disadvantages or the demerits of the genetic algorithm.

Now, actually comes the concept of efficient optimization tool. Now, supposing that we have got a very complex real world optimization problem and we will have to solve that. Now, to solve that if I take the help of a genetic algorithm genetic algorithm is a very powerful tool for global optimization, but its local search capability is not so good. So, if I use GA to solve that particular problem.

So, it will carry out a huge search, but it may take long time to hit the globally optimal solution. So, what I do is we first use GA to carry out the global search and we try to find out a region that is the region of interest where there could be a possibility that the globally optimal solution is going to lie. So, what I do is we take the help of another very good local search technique for example, say steepest descent method.

Now, once I have got some good region with the help of a GA we try to carry out further search with the help of a steepest descent method which is a very efficient local search method. So, to solve this complex optimization problem actually we consider both the things together say one tool for global search that is genetic algorithm.

And another tool for efficient local search that is the steepest descent algorithm that is the gradient based algorithm and combining these 2 algorithm we can develop a more efficient optimization tool and we have seen to solve a variety of difficult real world optimization problem. So, we will have to go for this type of efficient optimization tool.

Thank you.