Course Name: Optimization Theory and Algorithms

Professor Name: Dr. Uday K. Khankhoje

Department Name: Electrical Engineering

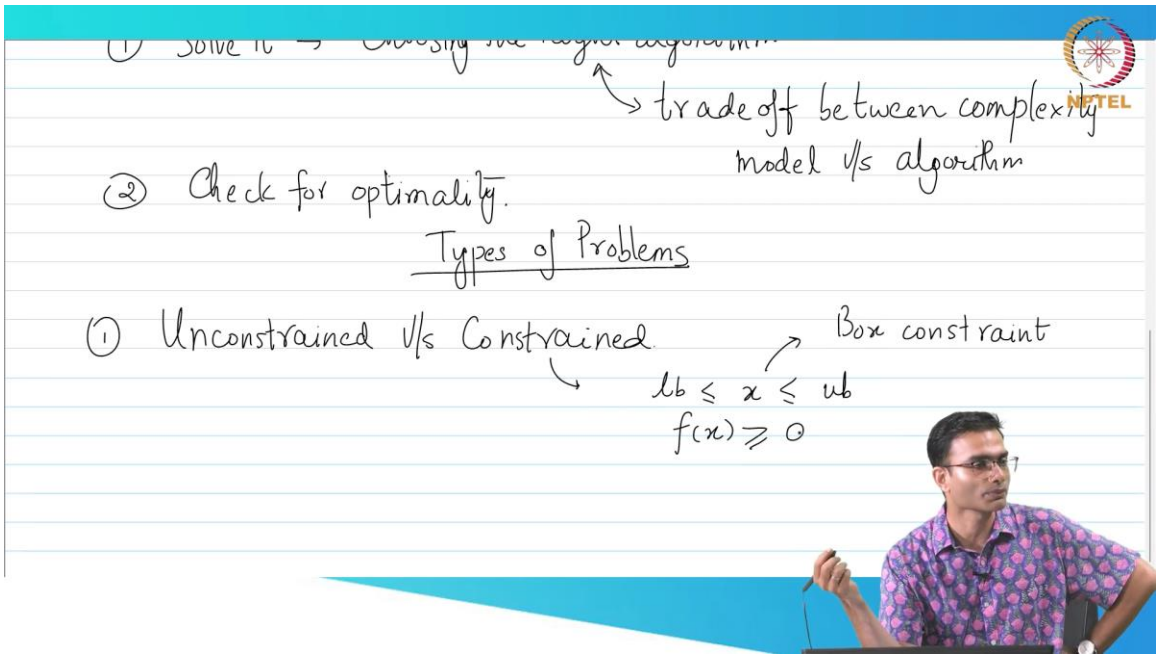
Institute Name: Indian Institute of Technology Madras

Week – 01

Lecture - 02

Introduction to the course - 2 - Types of problems

So, first one we have already looked at which is constrained versus unconstrained. Let us just note that down ok. Now, example of constraints we said was the simplest example we gave was for example, lower bound on the variable, right, this is what is called. So, the variable is bound between a lower bound and a upper bound, ok. So, this is in engineering words, this is called a box constraint because literally it is like constraining x to be within a box, okay? Variations of this are for example, positivity is a box constraint with the upper bound being right and you can think of various such things. They can be more complicated inequalities also.



For example, they can be

$$f(x) \geq 0,$$

f is some function which the user gives me and I say $f(x)$ is greater than equal to 0, ok, right. So, we will deal with these as they come. So, this is constrained versus unconstrained. Does it automatically start the next? Yeah, okay.

Can someone, can someone think of another type of problem? Very good, right. So, we can have continuous versus discrete. Can someone give me an example of a discrete optimization problem? Sounds very fancy discrete optimization problem, but we have all played discrete optimization problems since childhood. Give me an example. Number of parcels to pack in a truck.

Number of parcels that can be taken in a truck is going to be discrete quantity. I can take two parcels or three parcels. I cannot take 2.76 parcels. that is a good example, but I did not play this as a child.

Something simpler? You had a gifted childhood closest number to a perfect square that is also, chess. Are the moves continuous? Can your horse move 3.75 steps? Can your pawn move one and a half steps? No. So, it is actually a discrete optimization problem because the variable of optimization is not one variable. How many variables are there? Supposing you are a player, how many variables do you have? 16 variables and they can take only discrete values.

I can code them in some way. I can map them to real number, I mean to whole numbers, whatever, right. They are discrete valued. and in fact, they also have constraints, right? a pawn can move one sometimes two it cannot move three, right? So, chess is therefore very complicated problem, but I believe it has been solved right by one of deep mind or whatever right I mean at least beats all humans not deep mind deep blue, okay. So, discrete for example, games, puzzles, chess and continuous is most of engineering you can say.

Which do you think is easier? Continuous. Continuous is easier. We are of course saying this intuitively. What is your intuition? It is correct. Continuous are generally easier than discrete.

What is our intuition? you can use calculus that is the correct answer. Calculus tells us that if I calculate a small change in the input variable, I can calculate what is the corresponding change in the in the language of optimization what would that be? Objective function. Objective function. So, a small change in variable I can calculate what is the change in the objective function. Now, if my problem is to minimize an objective function, I can see my d x is it leading to the objective function reducing or increasing and the whole I mean the whole first part of this course on unconstrained optimization we are going to assume continuous variables and it is going to be built on the plank of calculus.

and this calculus you are already familiar with from class 12 right first derivative second derivative. The only difference I put it in quotes only difference is that we are going to generalize from scalars to vectors. So, that is going to be a new kind of calculus which I do not think you have ever looked at before ah, but do not be afraid it is all built on scalar stuff putting it together into vectors ok. So, the fact that I can do calculus means that I can find out some trends changing this variable this way what is happening to the objective function it is is it going in the direction that I want or not. this is very helpful on the other hand think of a chess game right there are two pawns next to each other right one pawn moving one step up or the adjacent pawn moving one step up can lead to totally different game outcomes one outcome can be you win the other out the other step could be you are in checkmate so there is no continuity So this makes discrete optimization problems difficult and so we are not going to talk about it in this course ok.

But a lot of engineering problems do need discrete optimization problems. There are some tricks which you can do to convert a continuous optimization problem to discrete. These tricks do not always work but so, I will give you an example. Supposing my variable is x which is discrete valued and it can take only values 0 and 1. So, let us say a bit.

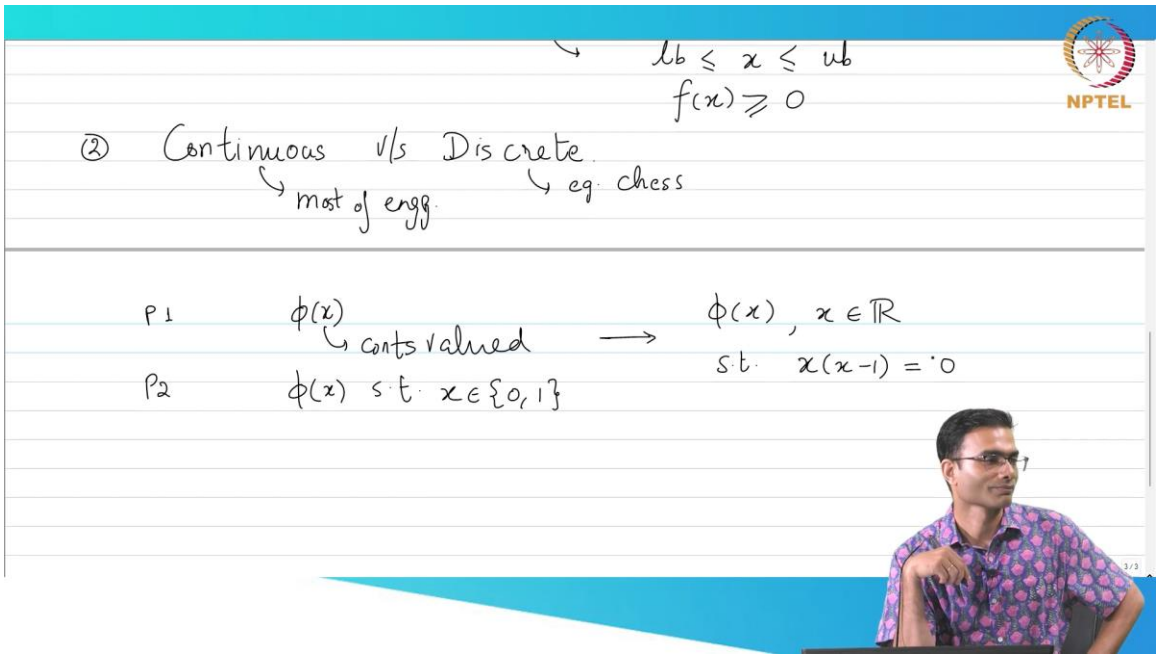
How do you think I can trick a continuous optimization problem into solving my discrete optimization problem? So, one solution is to threshold the solution, if something is above half I say 1, something is below half I say it is 0 right, it is back on ok. The problem with this is this is an intuitive hack, there is no proof that it will work. Can anyone think of another way to trick a continuous problem into solving my discrete optimization problem? So, multiplying a given solution with a number so as to make it this to to to discard it something simpler. Closest. That is what closest thresholding that was also already suggested.

Sampling? In the sense? So, you want to sample it, but there is no guarantee you will land up at a discrete solution. If it is 0 or 1 and I land up at 0.1, it does not help me. So, let me give you a very simple example. Supposing my objective function is

$$\phi(x), x \in R,$$

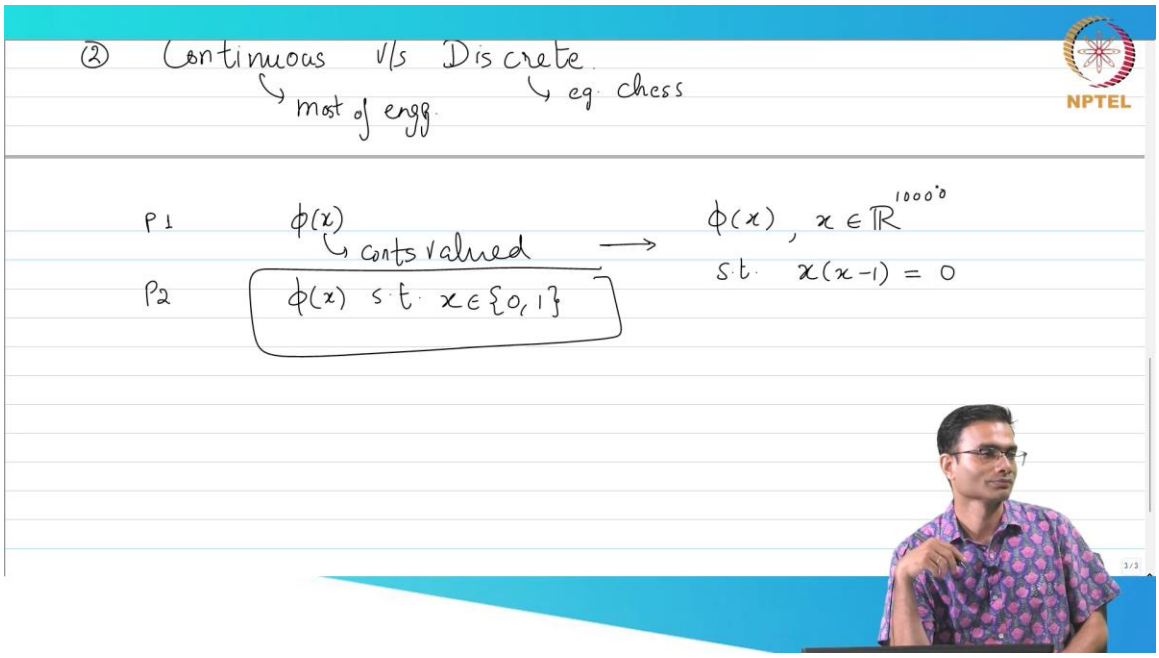
Okay? This is problem 1. Now problem 2 is

$$\phi(x), x \in \{0, 1\}$$



So, what I can do is I can take problem 1 and add a constraint to it. So, I say $\phi(x)$, x belongs to real numbers such that here is my trick right. So, this is the constraint will be satisfied only at 0 or 1.

So, I am kind of tricking my continuous optimization problem into solving a discrete optimization problem. So, this is why you know optimization in many ways is more like an art. There is no like grunge based do this, this, this and you come up with a solution. Now, I wrote this, this is now a quadratic constraint, right? And there are an infinite number of equations with solution as 0 and 1 right. I could multiply this by sin (x) it would still work right.



So, it is really up to you to use your creativity to come up with these kinds of tricks ok. So, this is just an example of how you can navigate between these worlds. In this case it is. Right. So, that is a good point he is mentioning that if my if this is my if this is my problem that optimize

$$\phi(x), x \in \{0,1\}$$

the common-sense thing to do is substitute 0 substitute 1 and pick the best one.

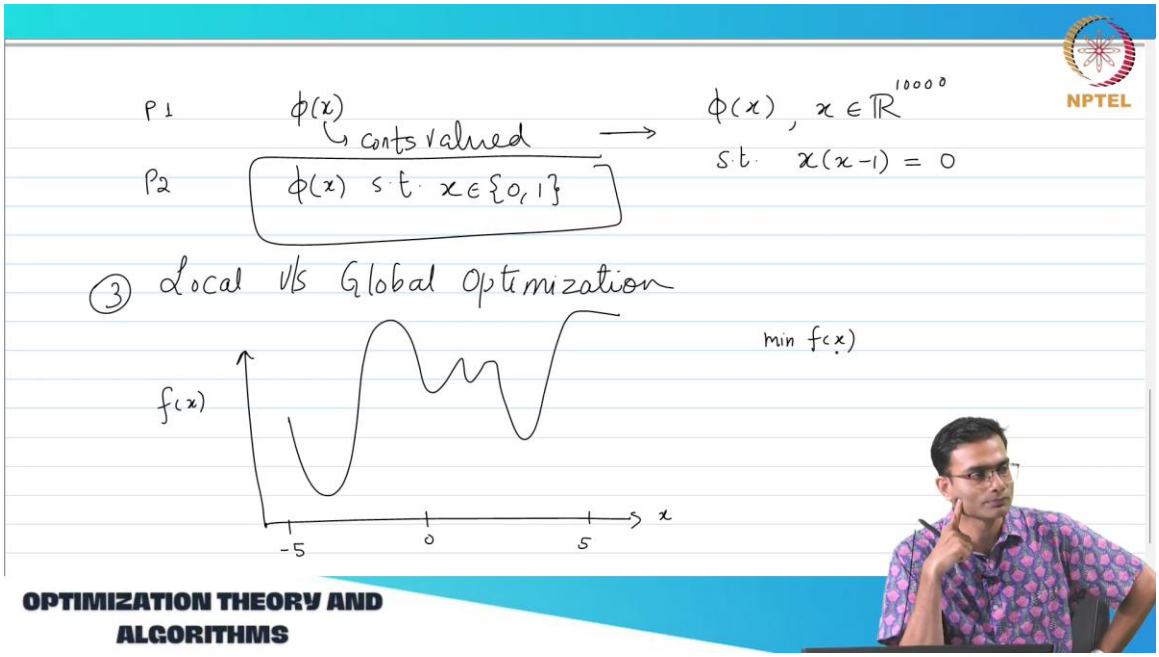
However, the way I have written x supposing I have written over here x belongs to R now supposing I say x belongs to. Right, ten thousand variables! Are you going to go manually one by one, one by one? No. And in fact, this is what happens. For example, most your favorite machine learning problems will have a million variables. So, what are you going to do? So, that is why you need some elegant tricks to solve these and sometimes it is not possible such as like this.

Okay, so, we have looked at continuous versus discrete, we have looked at constraint, unconstraint. The other type of optimization that we come across is local versus global optimization. So, the best way to describe this is via a figure. So, let me try to draw a figure. So, let us say this is zero, this is minus five and this is.

Now, if my problem is

$$\min f(x)$$

right, it is obvious that how many solutions are there to this problem? It depends on constraints. No, I have just said minimize f x, there is one solution. Now intuitively what have you answered? You have answered which you have given me the global solution to this problem, right. So, the global solution is over here. So, let us call this this is my global, but notice that the value of x at the global solution is let us say minus three, something okay.



Now, let us say that this was a real-life engineering problem with positivity constraints on x. then if you would look at this global solution and discard it. So, if you are looking at positivity constraints, is there another solution? Yes, there is a solution close to three. So, this is, we would call it a local solution because it is locally minimized. Local means in the neighborhood of that solution, it is the lowest possible thing.

So, what is the neighborhood over here? Roughly if I take this neighborhood, this is the local solution, okay? Now you can you can also automatically see that if I constraint myself if my constraints are different you know for example if my constraint is in this region, then I see one solution over here, one solution over here. So, as they say that the devil is in the details. So, depending on what your constraints are, you may come up with local solutions which are different from global solutions. And depending on your

problem, you may be happy with a local solution. A global solution may be infeasible, may be impractical, unphysical, whatever.

So, these are the two different kinds of situations that can arise. So, remember I mentioned checking of optimality earlier right. So, this is where checking of optimality can help, that once I got my solution, you know supposing I got a global solution and my optimality checking will tell me yes this is the global solution good. If I know that my solution is globally optimum, what should I do? I should go to the beach because life is done. I've got my solution, right? On the other hand, supposing I do my check and I find out that this is not the global solution, which means I have more work to do, right? So that's why it's very, very important.

So, this could be, for example, this could be your Uber solution and this could be Ola. for a particular routing problem to get to a certain customer. So, you know, if I were Ola, I would look at this, I would be particularly interested in knowing why is the competition doing better than me. Maybe my algorithms are giving me solutions that in the neighborhood are the best, but they may not be globally best.

Right. So, this is and checking optimality is not always very easy ok. Mathematically as we will go in this course, we will see that checking optimality may not always be very easy and it is an active area of research. So, these are places where you can make contributions to as well ok. Now, can someone point out some kind of difficulties that you can see if the landscape is like this? So, I have given you a one-dimensional example because there is only one variable of optimization. So, we will see this as we go later in the course, but one of the problems, so what is the characteristic of this kind of a problem? This is what would be called a non-linear problem.

You can see clearly that $f(x)$ is not linear in x, it is highly non-linear. Right, so, we can have saddle points when we do optimization saddle points are points where in one dimension it is a maxima in the other direction it is a minima, right? Those are saddle points those are relatively easy to deal with, the problem is if you have lots of local solutions right. Now, here I have drawn you a solution in one I mean I have drawn you a problem in one dimension. Now, let us imagine this x to the ten thousand, right? Am I going to be able to go and check every single local solution and say oh no local, local, local right this becomes very, very hard and this is one of the deepest challenges in

machine learning, deep learning that I have a large number of variables, my objective function is not linear, it is highly non-linear and so I got a solution, but is this the best solution? We do not know. So, this is a very real-life real-world problem and what happens is that we I already gave you a hint that we are going to use calculus in a continuous optimization problem.

Now, what happens when we do calculus is I will take some starting point let us say I take this point and calculus means let us say take small changes in x and see what happens to the objective function is it increasing is it decreasing. So, if I start from here I may end up at this solution which is good. On the other hand, if I standard started over here, then will I go in this direction? No, because the objective function is increasing. that means I will go in the reverse direction. So, supposing I start walking down over here, the objective function is reducing as I go in that direction very good you get happy, I will land up over here and I find out that in the local neighborhood the solution is not improving.

So, I declare success right. So, this is what in technical words this is called we are getting trapped in a local optima or local minima. So, in a non-linear optimization problem this is what happens very very easily you get stuck in local minima. So, there is a full branch of optimization called global optimization which tries to deal with these problems which we will not cover in this course, but I will just mention a few names they are called for example, simulated annealing. the analogies with heating a metal or some object which has some kind of defects in the crystal structure. So, you heat it slowly slowly and the defects kind of go away and then.

So, the point is the heat is analogous to heating this guy. So, that this guy can fall into an adjacent value. So, you keep restarting heating, restarting heating. So, it is this is one way. Then there are very very popular things called as evolutionary algorithm, algorithms, which again we will not study in this course, but they are very important.

You have all heard these crazy names. For example, can anyone mention a evolutionary algorithm? genetic algorithms. You model your problems in terms of your optimization variables become genes, your solutions become I think chromosomes and then one solution gets crossed with another solution, you can have mutation and lead to new solutions which may be good or not. And in many very very complicated problems

people do this. In fact, in the engineering literature, you will find a lot of these kinds of things.

So, you have genetic algorithms. Then it's very interesting. I mean, if you can read a review paper on this, there's no calculus involved. So, it's just, you can just read it like a novel. Okay.

There's something called ant colony optimization. People studied how ants walk around and form a colony and they got some intuition from that. Or again, inspired from nature, you've seen, these birds flying in a flock right it is not random they have some kind of coordination going on. So, you will find particle swarm optimization, the latest I read a few months ago was gray wolf optimization. So, people have come up with all sorts of funny names their inspiration is nature and they are trying to model it. But they do not come with proof of proof of convergence or those kinds of things, but if your problem is so complicated then you know you do not have much choice.

So, the question is that we can can we test these algorithms against standard problems and if it works on the standard problems, we can kind of get some faith that they will work. Yes and no. because if it works for some template or a benchmark problem there is no guarantee it will work just as well on your problem. So, you again you have to take it with a pinch of salt, but in a practical engineering problem you may have no choice, but to try it out.