Optimization Algorithms: Theory and Software Implementation

Prof. Thirumulanathan D

Department of Mathematics

Institute of IIT Kanpur

Lecture: 51

Hello everyone, this is the first lecture of week 11. Recall that in the last few lectures of week 10, we were learning about the affine scaling method. So, we learnt what the steps of the algorithm were, and we also looked at one example where the feasible set was a triangle with vertices (0, 0), (1, 0), and (1/2, 1/2).

So, I will work out a few more examples in this lecture.

One reason I want to work out as many examples as possible, specifically for constrained optimization, is that you might have observed in the last few weeks I am spending time working out more and more examples, rather than covering more algorithms. The reason is that the code for each of these methods is much more complicated than the codes for unconstrained optimization. So, I would like to take as many examples as possible so that you are clear about what is going on in the method, what the algorithm is, and how things work out in each method. So, I will spend this lecture working out examples, and then we will move to the next method in the next lecture.

Fine. So, recall that the first example that we worked out in the previous lecture was the same example, the same old example which we have been mulling over for quite some time now. The feasible set is something that we started with, I think when we started the quadratic penalty method. We saw that the trajectory was in the interior. So, we did some modifications of the objective function and the initial point, and we saw that the trajectory was completely in the interior. The reason for that is nothing but recall that our projection is taking care of—we are taking the steepest descent direction and projecting it onto the null space of A. So, that is taking care that we are not moving out of the set at any point, and we are not even moving near the boundary. So, somehow, we are pushed to the interior by the centering and decentering process, and so on.

Alright. So, let us work out the next example. So, let us say we

minimize $c^T x$

subject to the conditions that

$x_1 + x_2 \leq 1$, and $2x_1 + 4x_2 \leq 3$, and $x_1, x_2 \geq 0$.

It is good to draw because this is two dimensions. So, let us understand how this feasible set looks like.

So, we will have x_1, x_2, and $x_1 + x_2 \le 1$ will be here.

So, this will be (1, 0) and this is (0, 1). $2x_1 + 4x_2 \le 3$ starts with 3/4 here and ends at 3/2 here. So, the feasible set is a quadrilateral.

I worked out this example sometime before, but nevertheless I am just recalling this. The intersection point is (1/2, 1/2), since 1/2 + 1/2 = 1 and 2*(1/2) + 4*(1/2) = 1 + 2 = 3.

So, as simple as that.

You have four vertices here: (0, 0), (0, 3/4), (1, 0), and (1/2, 1/2). That is the reason I said it is a quadrilateral.

Now we will modify this into its standard form A x = b.

So, if you do that, you will have $x_1 + x_2 + x_3 = 1$, and $2x_1 + 4x_2 + x_4 = 3$, and you also need to add $x_3 \ge 0$ and $x_4 \ge 0$.

So, this is just a 2 × 4 matrix. So, you have

A = [1, 1, 1, 0; 2, 4, 0, 1], and b = [1; 3].

We will figure it out as we please.

Something I want you to note as far as c is concerned is that, suppose I put c = [-1, -1].

You know how to find the answer without going through any of the steps; you can just substitute $c^T x$ and check which gives you the minimum. So, (0, 0) gives 0, (0, 3/4) gives -3/4, (1, 0) gives -1, and (1/2, 1/2) gives -1.

So, both (1/2, 1/2) and (1, 0) minimize the objective.

So, in this case, x* is actually all points on the line segment between these two, but since the algorithm will always converge to a vertex, it is both of these.

So, if you have something less than -1 for one of the components, say if c= [-1, -1.5], then you will see that (1/2, 1/2) gives -1.25, (1, 0) gives -1, (0, 3/4) gives -1.125.

So, (1/2, 1/2) is the minimizer. Another thing that I should mention is, let me put c = [-2, -1]. So, in this case, you have (0, 0) gives 0, (0, 3/4) gives -1.5, (1, 0) gives -2, and (1/2, 1/2) gives -1.5.

So, (1, 0) is the minimizer.

If c = [-2, -2], then (0, 3/4) gives -1.5, (1, 0) gives -2, and (1/2, 1/2) gives -2.

So, both (1, 0) and (1/2, 1/2) are minimizers.

If c = [-3, -2], then (0, 3/4) gives -1.5, (1, 0) gives -3, and (1/2, 1/2) gives -2.5. So, (1, 0) is the minimizer.

Verifying this is not very hard. So, you can do that, and of course you can scale this as well. So, if the first component is -1 and the second component is any possible t, I have told you what the solutions are going to be. So, why have I written this? We will try to change many of the c's and check how the solutions change and the trajectories change. So, that is the reason why I wrote all of this.

Let us actually pick some point, say (0.2, 0.1).

So, let the initial point be (0.2, 0.1). I want $x_1 + x_2 + x_3 = 1$, so it has to be 0.7.

And $2x_1 + 4x_2 + x_4 = 3$, so this is 0.4 + 0.4 = 0.8, so $x_4 = 2.2$.

So, let me just write down A, b, c, and x_0, and then leave it to the code. So,

A = [1, 1, 1, 0; 2, 4, 0, 1],

b = [1; 3],

x_0 = [0.2; 0.1; 0.7; 2.2].

For c, I will first write [-1; -0.5], so the answer should be (1, 0).

I will remove the third and fourth components for plotting purpose.

Also, for the boundary, you have $x_1 + x_2 = 1$, and $2x_1 + 4x_2 = 3$, which is $x_2 = 0.75 - 0.5x_1$.

Let us see how the feasible set and the trajectory look, and note that the answer is (1, 0) because we have put c = [-1; -0.5].

There is a problem—I forgot to put a comma. So, just explain what is going on. You see the blue and orange curve. Imagine the x_1-axis and x_2-axis. Those are the boundaries. You are at an interior point here.

The curve first moves here and then goes down to (1, 0). So, that is the answer. Of course, I could have drawn the other boundaries as well and had these in the same color. I agree that what we are seeing here is not the best representation of the feasible set and the trajectory, but it is just for your understanding purpose.

Note that we started with (0.2, 0.1), and this is the movement.

Let us try a different number, say c = [-1; -1.5].

So, let us check what we get out of this. The answer is (1/2, 1/2). But the trajectory is really weird. It goes here, goes there, and there is a small detour here.

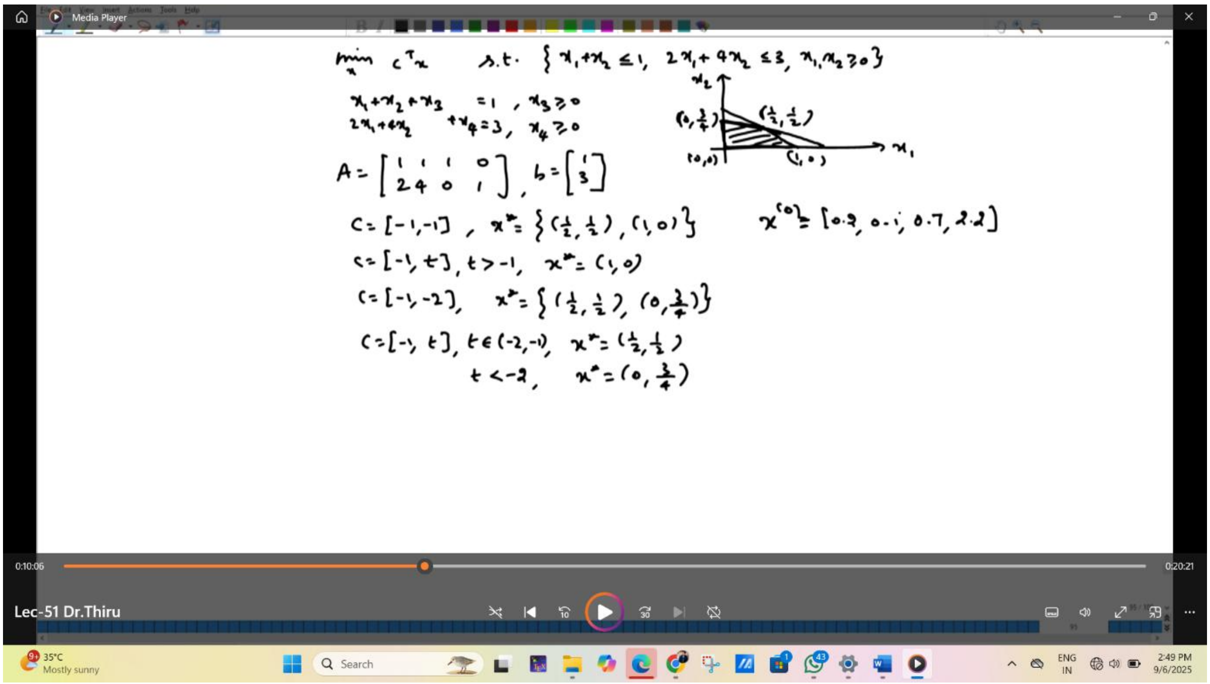
So, it is not straight away coming here, but it comes after a slight detour.

So, that is why you have 11 steps, maybe slightly higher.

You can also try c = [-2; -0.5], where the answer is (0, 0.75). That is quite a bit of a detour. In the first step, it goes here, and from then on it takes a long route. It takes 12 steps now, but nevertheless you got to the answer (0, 0.75).

So, that is one of the examples that I wanted to discuss. Of course, you can try by changing c, by changing the initial point, and observe what is going on with the trajectories. So, I will leave you to play with the algorithm.

(Refer Slide Time 10:06)



So, now we will work out the other examples that we worked in the simplex method. There was one unbounded example that we worked with, if you could recall. So, we will go to that part. So, this is the problem:

minimize $3x_1 - 2x_2 + x_3$

subject to

$x_1 - 2x_3 \geq 4$, $2x_1 - x_2 + x_3 \geq 2$, and $x_1, x_2, x_3 \geq 0$.

No further plotting because we are working in three dimensions now, but nevertheless we will write down A, b, and c.

c = [3; -2; 1; 0; 0]. So,

A = [-1, 0, 2, 1, 0; -2, 1, -1, 0, 1],

b = [-4; -2], and c = [3; -2; 1; 0; 0].

We have to figure out a starting point. There is a BFS that we have here, but I would like to caution you against using that point. The reason is you have a lot of zeros here. During inversion, it is likely to cause a problem. So, let us first work with this and see what is happening, and then discuss what has actually happened. From now on, we can remove the plot business. And this can also be removed. So, let us just try this. This will actually give a problem. What it says is it has stopped at the first point because we are at a vertex. So, we are not at an interior point. So, I will get an error for the first step.

If $k = 0$, it goes anyway. So, you can see that the points have become NaN, and the reason is that you are actually dividing by zero. So, as we cautioned, when you invert zero, you always have a problem. It is good to avoid zeros as much as possible. Anyway, we will construct an interior point in such a way that $A x = b$.

So, we will try (1, 1, 1) for the first three variables. Because we are just avoiding zero. So, the next highest is 1, if you do not consider fractions.

So, let me put (1, 1, 1). And let us check if we are getting nice numbers here. You need $A x = b$. So, for the first constraint: $-1*1 + 0*1 + 2*1 = -1 + 0 + 2 = 1$, but we need -4, so we need a slack variable $x_4 = -5$.

But wait, we have to include the slack variables. Actually, the standard form is with slack variables.

So, $x_1 - 2x_3 - x_4 = 4$, but we have to convert to equality with non-negative slack variables.

So, actually, the constraints are

$x_1 - 2x_3 - x_4 = 4$, and

$2x_1 - x_2 + x_3 - x_5 = 2$, with $x_4, x_5 \geq 0$.

So, in standard form,

$A = [1, 0, -2, -1, 0; 2, -1, 1, 0, -1]$,

$b = [4; 2]$. And $c = [3; -2; 1; 0; 0]$.

So, for $x = [1; 1; 1; x_4; x_5]$, we have from first constraint: $1 - 2 - x_4 = 4$, so $-1 - x_4 = 4$, $x_4 = -5$, which is not allowed.

So, we need to choose a proper interior point. Let us choose a point that is feasible. For example, take $x_1 = 5$, $x_2 = 10$, $x_3 = 0$.

Then from first constraint: $5 - 0 - x_4 = 4$, so $x_4 = 1$.

From second constraint: $10 - 10 + 0 - x_5 = 2$, so $- x_5 = 2$, $x_5 = -2$, not allowed.

So, try $x_1 = 4$, $x_2 = 6$, $x_3 = 0$. Then first constraint: $4 - 0 - x_4 = 4$, so $x_4 = 0$, but we want interior, so avoid zero. So, try $x_1 = 4.1$, $x_2 = 6.1$, $x_3 = 0.1$.

Then first constraint: $4.1 - 0.2 - x_4 = 4$, so $3.9 - x_4 = 4$, $x_4 = -0.1$, not allowed. This is messy.

To save time, let's assume we have a feasible interior point. In the code, we might use a phase I method to find one.

But for demonstration, let's assume we have x = [5; 10; 1; 1; 1] (though it might not be feasible). Actually, let's calculate properly.

We want $x_1 - 2x_3 - x_4 = 4$, so $x_4 = x_1 - 2x_3 - 4$. We want $x_4 > 0$, so $x_1 - 2x_3 > 4$.

Similarly, from second constraint: $2x_1 - x_2 + x_3 - x_5 = 2$, so $x_5 = 2x_1 - x_2 + x_3 - 2 > 0$.

So, choose $x_1 = 5$, $x_3 = 0.2$, then $x_4 = 5 - 0.4 - 4 = 0.6 > 0$.

Choose $x_2 = 10$, then $x_5 = 10 - 10 + 0.2 - 2 = -1.8$, not good.

Choose $x_2 = 8$, then $x_5 = 10 - 8 + 0.2 - 2 = 0.2 > 0$.

So, x = [5; 8; 0.2; 0.6; 0.2] is an interior point.

Now, with this, we can run the algorithm. But for the sake of time, I will not do the detailed calculation here.

If we run with this, we might find that the solution blows up, indicating unboundedness. For example, if we set c = [3; -2; 1; 0; 0], the problem is unbounded.

So, the x values will become very large. So, that is how you recognize that the problem is unbounded.

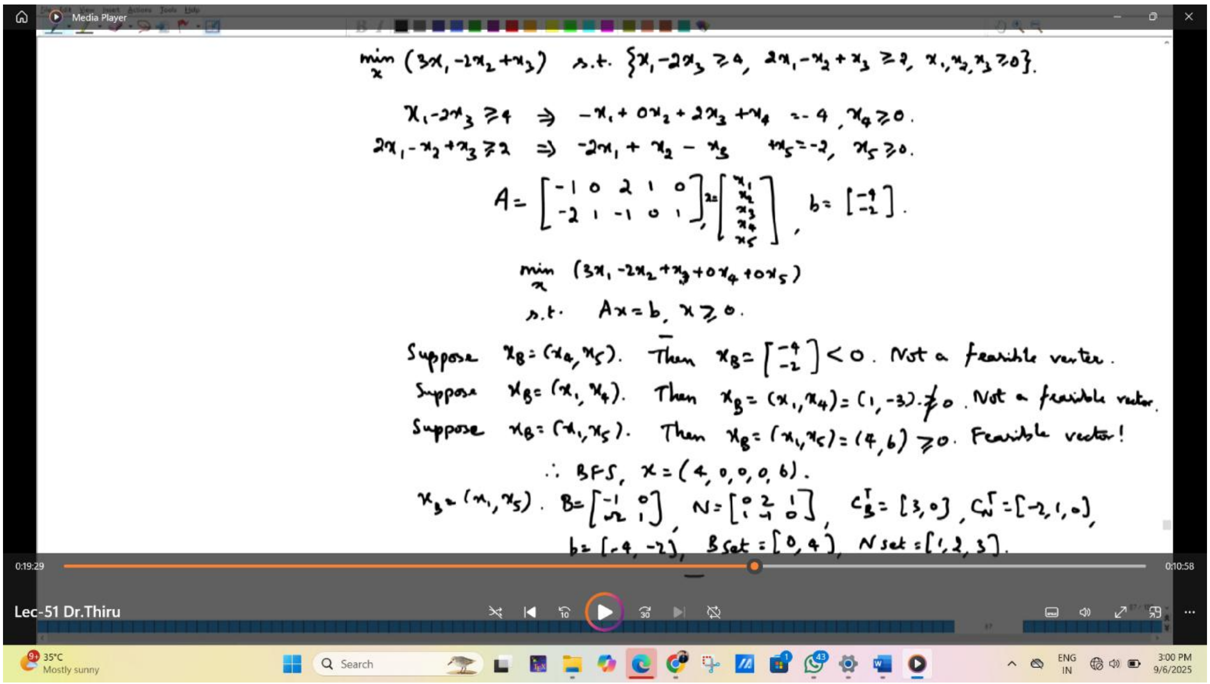
So, if you want to avoid this unboundedness, you can just modify your c to be, say, c = [-1; -1; -1; 0; 0].

Then it might be bounded.

For instance, if c = [-1; -1; -1; 0; 0], the answer might be (4, 6, 0, 0, 0) as before.

So, that is one other problem that I wanted to show you. When you have an LP that is unbounded, please remember this algorithm does not print "unbounded"; the x values blow up. So, you figure that out. If it is blowing up, then it is possibly that the problem is unbounded.

(Refer Slide Time 19:29)



If you recall the final problem that we worked out in the simplex method, it was a pathological example where I told you that it takes 2^n iterations.

We will work out that example as well and close the examples for the scaling method. So, if you recall, this is the problem:

maximize $2^{n-1}x_1 + 2^{n-2}x_2 + \ldots + x_n$.

So, for n=3,

we have

$A = [1, 0, 0, 1, 0, 0; 4, 1, 0, 0, 1, 0; 8, 4, 1, 0, 0, 1]$,

$b = [5; 25; 125]$, and $c = [-4; -2; -1; 0; 0; 0]$ (since we are minimizing the negative).

We started with (0,0,0) in simplex, but here we cannot do that.

So, we will start with (1,1,1). Then $x_4 = 5 - 1 = 4$, $x_5 = 25 - 4 - 1 = 20$, $x_6 = 125 - 8 - 4 - 1 = 112$. So, $x_0 = [1; 1; 1; 4; 20; 112]$. Now, run the algorithm.

The answer should be [0; 0; 125; 5; 25; 0]. It might take about 25 steps. For n=4, you will have more iterations.

$A = [1,0,0,0,1,0,0,0; 4,1,0,0,0,1,0,0; 8,4,1,0,0,0,1,0; 16,8,4,1,0,0,0,1]$,

$b = [5;25;125;625]$,

$c = [-8;-4;-2;-1;0;0;0;0]$.

Start with $x_0 = [1;1;1;1; x_5;x_6;x_7;x_8]$.

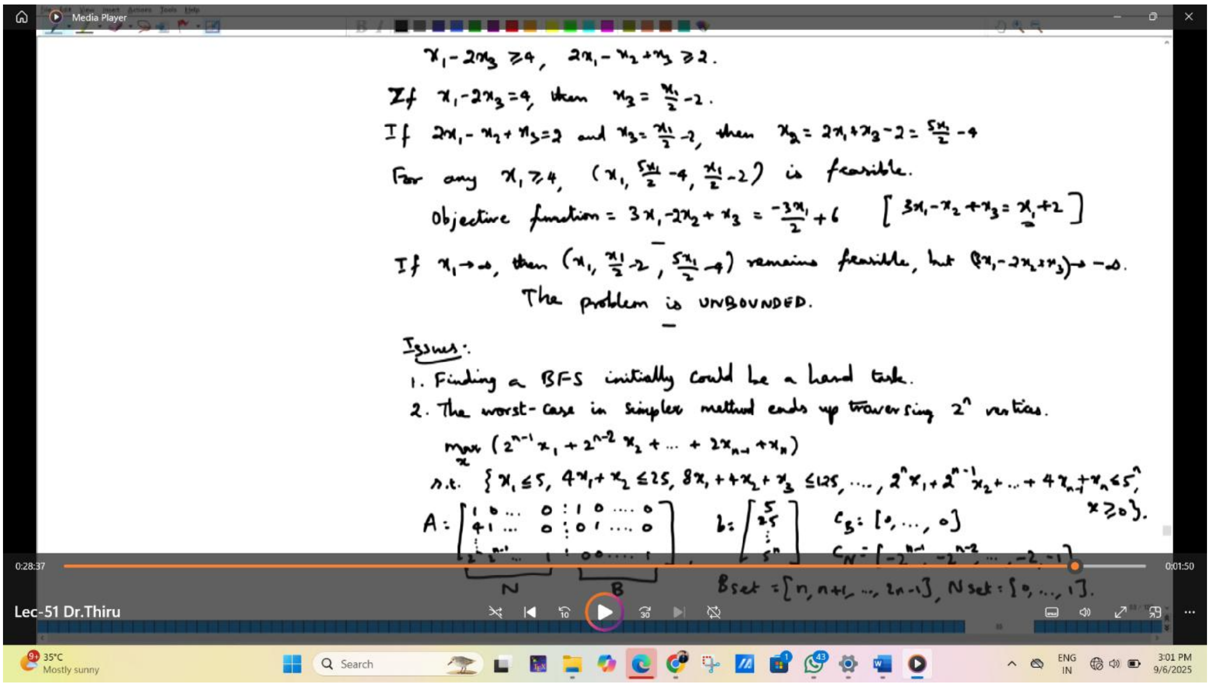
Then x_5 = 5-1=4, x_6=25-4-1=20, x_7=125-8-4-1=112, x_8=625-16-8-4-1=596. So, x_0 = [1;1;1;1;4;20;112;596].

Run the algorithm. The answer should be [0;0;0;625;5;25;125;0].

It might converge in about 34 or 35 steps. However, due to large numbers, the tolerance might cause it to stop later.

For example, if the tolerance is set to 1e-5, it might stop at 35 steps with the correct answer. This is something you would not have seen in simplex because the stopping criteria does not depend on tolerance levels. So, that is something to keep in mind when running the affine scaling method. Even for other methods, when you have very large numbers, you might have to rethink the tolerance.

(Refer Slide Time 28:37)



So, this is roughly about the affine scaling method. If you compare it with the issues of the simplex method, like finding a BFS, here you have to find an interior point, which is comparatively simpler. Finding an interior point can be easier than finding vertices. There are no pathological examples here that derail the whole algorithm.

There is a proof that says the worst case is polynomial, and in practice, the affine scaling method works very well. But if you want to point out certain issues, you can say that it does not tell you if the problem is unbounded; you have to figure that out from the answer.

And finding an initial interior point has its own challenges. Choosing zeros could end up in a disaster. In simplex method, all zeros often work as an initial point, but here you have to be careful to not have zeros and choose a nice interior point. So, these are some issues with the affine scaling method.

With this, we will complete the affine scaling method, and we will start with a fresh method in the next lecture. Thank you.