LESSON: LP1: Linear Programming

***

(Slide 1) Linear Programming

Linear programming is a very general technique for solving optimization problems. It handles any problem that can be formulated as an optimization over a set of variables with a goal known as the objective function and the constraints can all be expressed as linear functions of the variables.

We'll see that **Max-flow** can be formulated as a linear program.

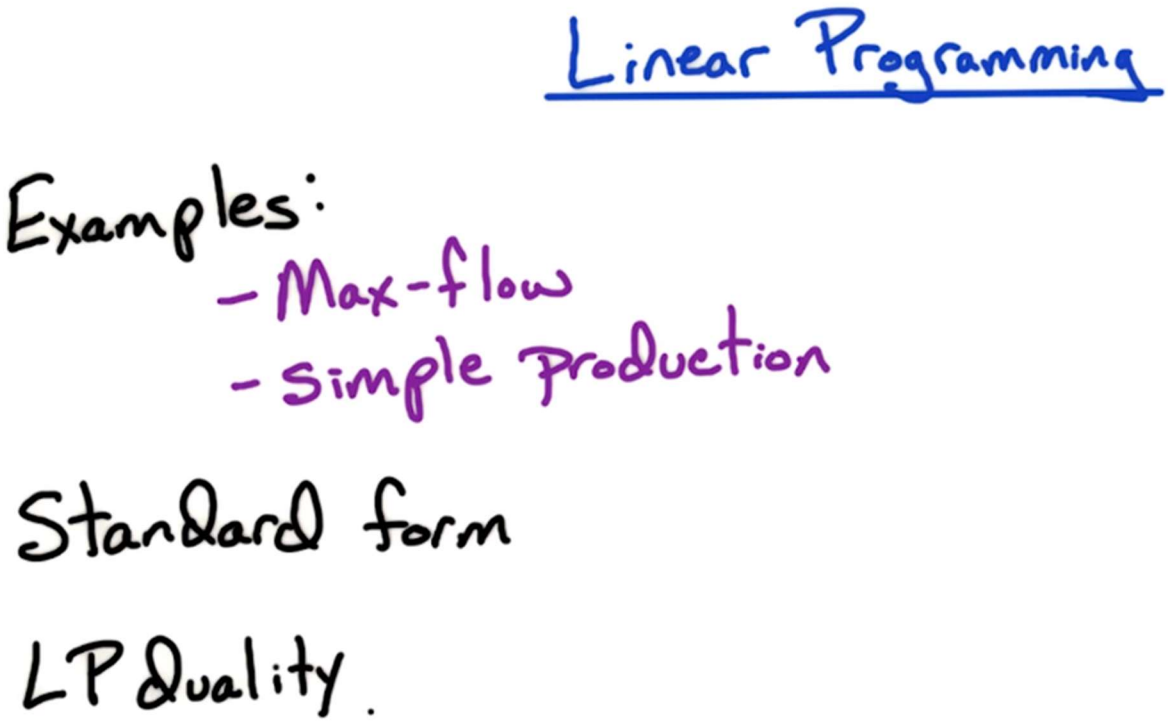
Also, we'll see that many problems from operations research can be expressed as LPs. We'll describe the general idea of the **simplex** algorithm which is widely used for solving LPs.

We'll take a closer look at one of the beautiful aspects of linear programming, **LP duality**. We'll explain what LP duality is and some of the useful consequences for us.

Finally,we'll look at an application of LP to **approximation algorithms**. We'll use Linear programming to design an approximation algorithm to the classic SAT problem.

***

(Slide 2) Linear Programming Lecture Outline



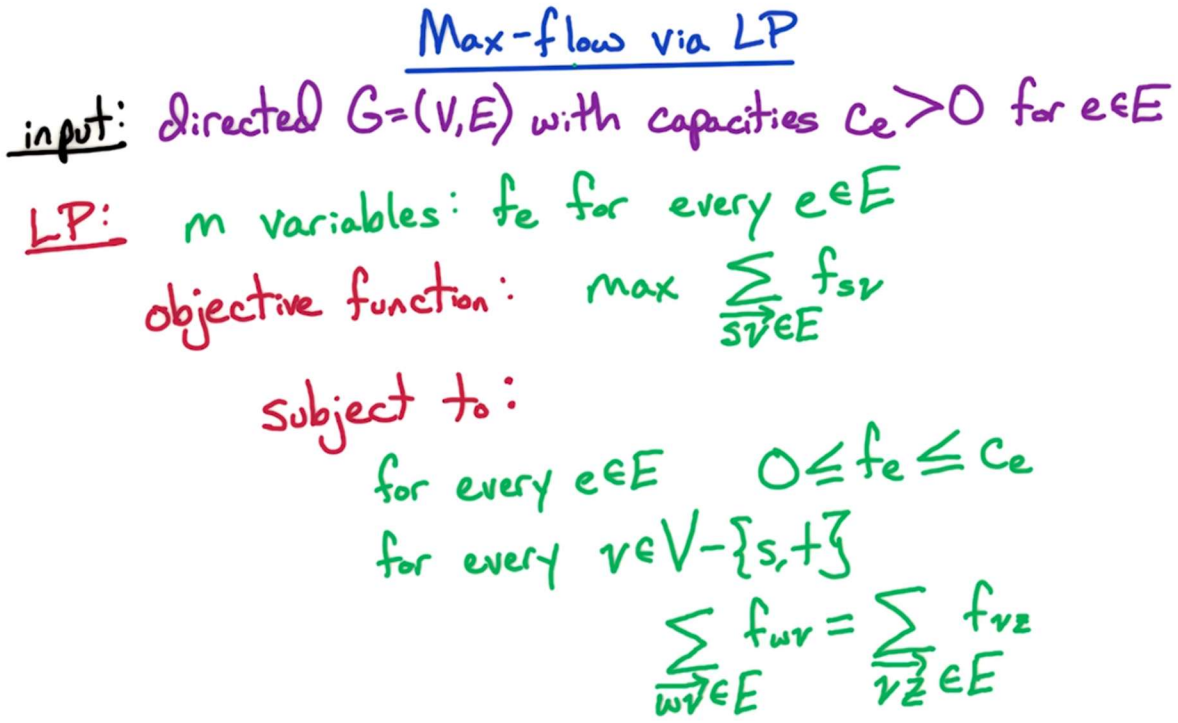
In this section, we'll explore **Linear Programming**. This is a powerful technique. We'll see how to solve a variety of problems using linear programming.

We'll start by seeing how we solve **Max-flow** using linear programming, then we'll see a few simple examples of linear programs for production problems. After looking at these examples, we'll give the general formulation of Linear Programming, and look at the standard form for LPs.

Finally, we'll look at **LP duality**, which is a beautiful and important aspect of this topic. Let's dive in and look at how we express Max-flow as a linear program.

***

(Slide 3) Max-Flow as LP



The input to the Max-flow problem is a directed graph G along with positive capacities for every edge. Now our linear program is going to have m variables, one for every edge. Now, we're going to have an objective function - this is the function that we're trying to optimize. This should be a linear function of the variables.

Now, for Max-flow, we're trying to maximize the size of the flow. We measure the size of flow by the flow out of the source vertex or the flow into the sink vertex. We'll state it in terms of the flow out of the source vertex, and thus we're maximizing the total flow out of the source vortex. So we're summing over edges out of the source vertex S, and we're looking at the flow along that edge.

Now we have some constraints on our variables:

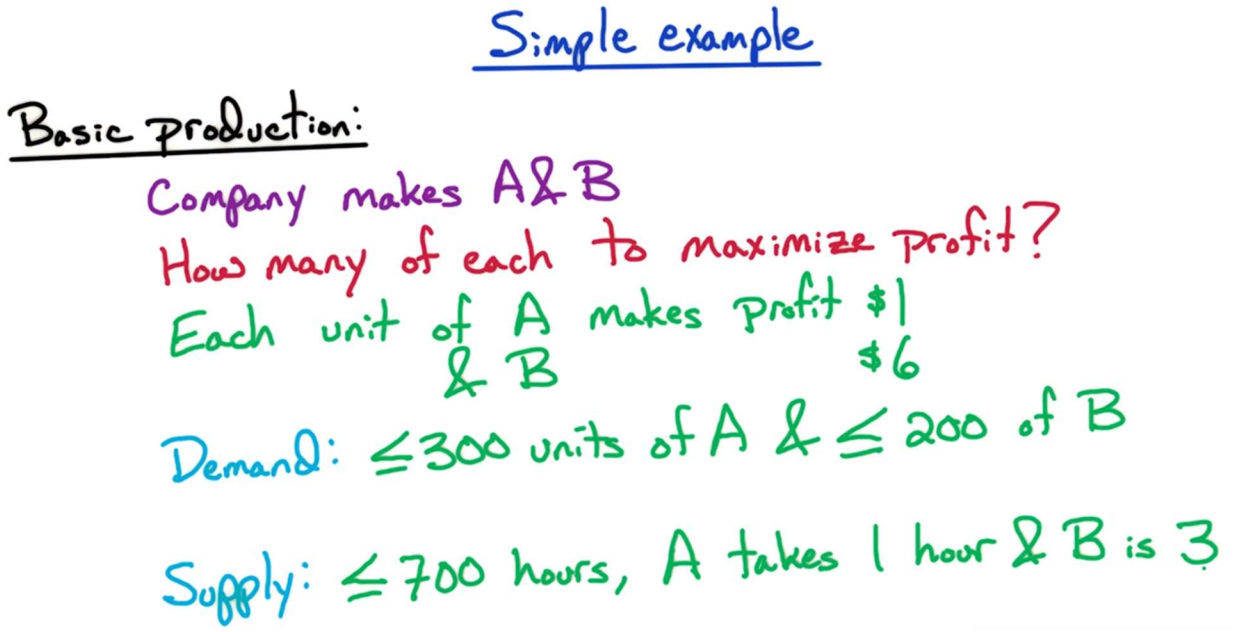
- For every edge, the flow along that edge has to be non-negative and it needs to satisfy the capacity constraint. So the flow must be at most the capacity.
- We also have a constraint for every internal vertex. So, for every vertex, except for s and t, we need that the flow into vertex v must equal the flow out of vertex v. So the flow is conserved. So if we look at the total flow into v, that needs to match the total flow out of v.

Now, why is this a linear program? Because the objective function is a max or a min of a linear function of the variables, and all the constraints are linear functions of the variables.

What we'll see is that this formulation of the Max-flow problem is quite powerful. We can easily solve several variants of the max flow problem with simple modifications to this LP.

***

(Slide 4) Simple 2D Example



Let's take a look at a simple example of a linear program. This is motivated by a basic production problem.

Consider a company that makes two products - let's call them A and B. We want to figure out how many of each product to produce in order to maximize our profit.

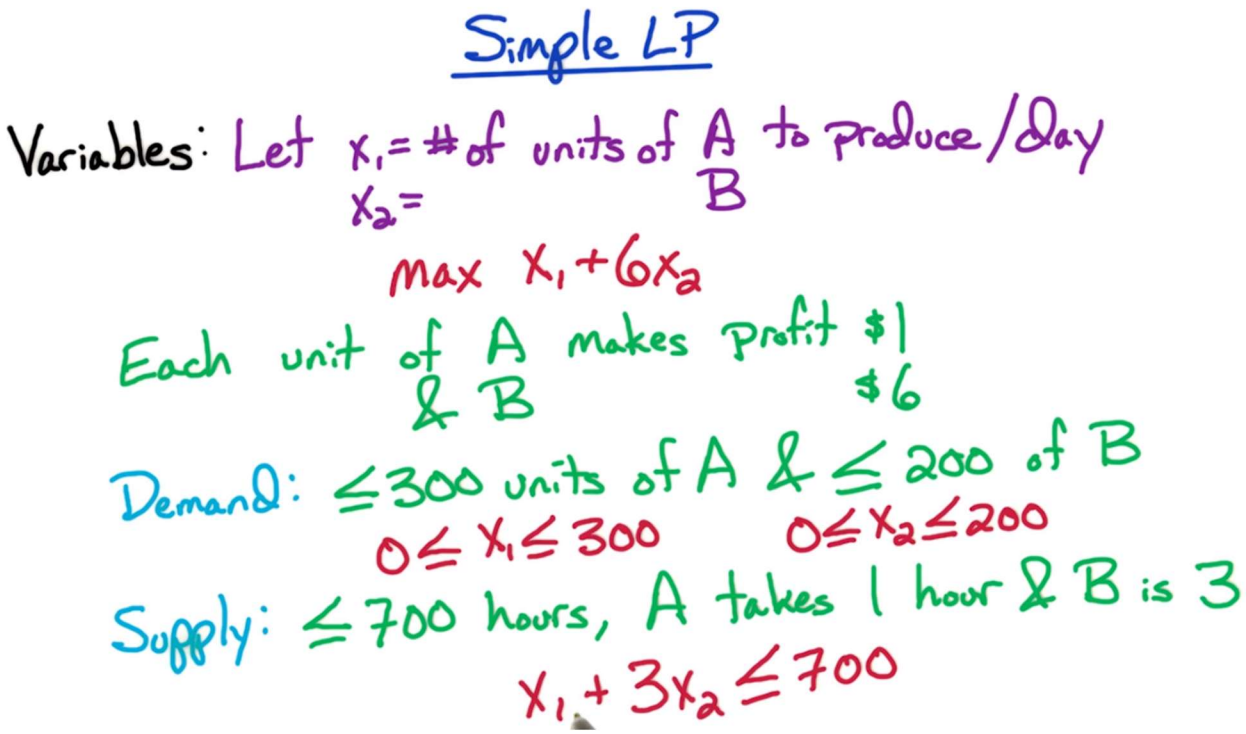
Now let's make some modeling assumptions:

- Each unit of A that we sell makes a profit of one dollar and for each unit of B, we'll assume that it makes a profit of six dollars.
- Now let's make some assumptions about the demand for each product. We'll assume that the demand for A is at most 300 units per day and at most 200 units per day for B.
- We'll also assumes some supply constraints. We'll assume our employees can work at most 700 hours per day and each unit of A that we produce takes one hour, whereas each unit of B takes three hours.

Now, let's express this is as a linear program.

***

(Slide 5) 2D LP Formulation



First off, what are the variables in our linear program?

What we're trying to determine is how many units of A and B to produce each day. Hence, let x1 be the number of units of A that we should produce per day and x2 is the number of units of B to produce per day.

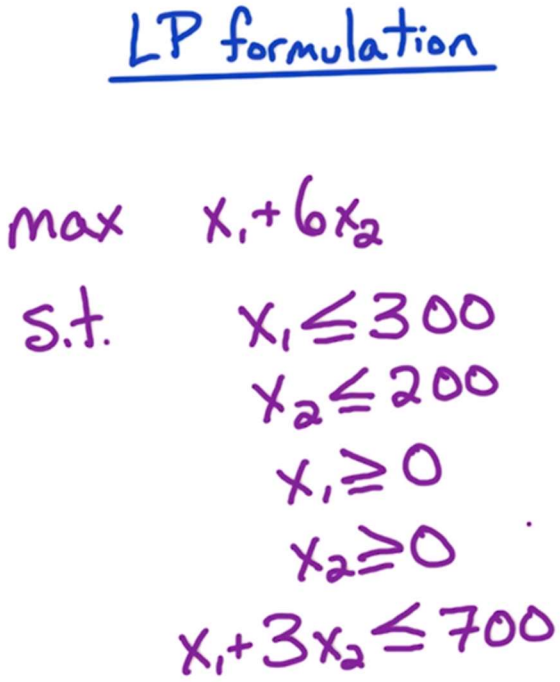
Our goal is to maximize the profit. So, we have a maximization problem. Now, how do we express the profit in terms of x1 and x2? Well, we make a profit of $1 per unit of A, so that's x1 and $6 per unit of B that we sell - so, that's 6 x2.

Now what are the constraints?

- First off, the demand constraint. Well, we want to make at most 300 units of A. So, this means that x1 is at most 300 and x2 is at most 200.
- Now, also, we want x1 and x2 to be non-negative. We can't produce a negative amount of any of the products. So, let's throw in these non-negative constraints.
- Now let's look at the supply constraint. The total supply, the total number of man hours available is 700 and it takes one hour per unit of A and three hours per unit of B. So, we want that x1 + 3 x2 is at most 700.

***

(Slide 6) 2D LP Recap



Now, let's go ahead and restate the LP we just formulated. Our objective function was to maximize x1 + 6 x2. Recall, this was the profit we obtained.

Now,this was subject to the following constraints.

- First off, the demand for x1 product A was at most 300 units.
- For Product B the demand was at most 200 units, so x2 is at most 200.
- We need it to ensure that these variables are non-negative.
- Finally, we have the supply constraint.

Now let's look at this problem geometrically. For now, let's ignore the objective function and let's look at the five constraints. Now each of these five constraints is a half plane. If it was equality, it would be a line, but we're doing one side of the line. So, this gives us a half plane.

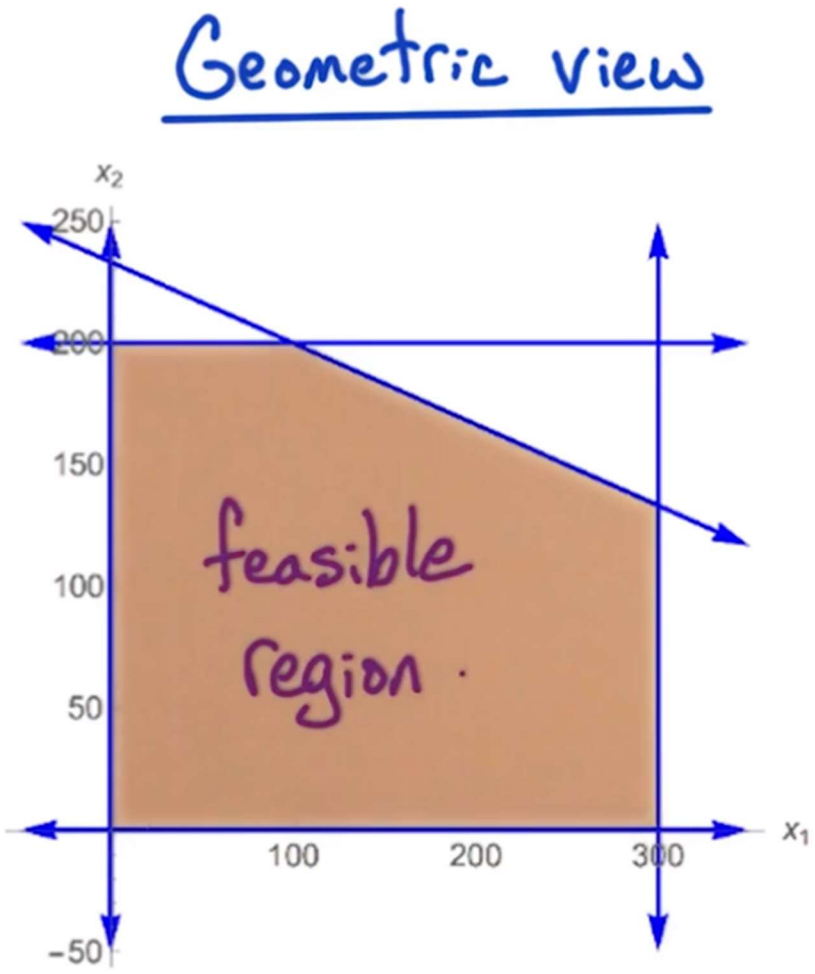
Now, note, we're in two dimensions because we have two variables. Now, we want to satisfy all five constraints, so we want to look at the intersection of these five half planes.

The intersection of these five half planes is going to give us a set of feasible x – the set of x which satisfy all five constraints and, then, we're going to look in that feasible set to find the x which maximizes this objective function.

So, let's first look at this feasible region.

***

(Slide 7) 2D Geometric View



Let's take a look pictorially at these five constraints.

Now, once again we have two variables. So we're in two-dimensional space.

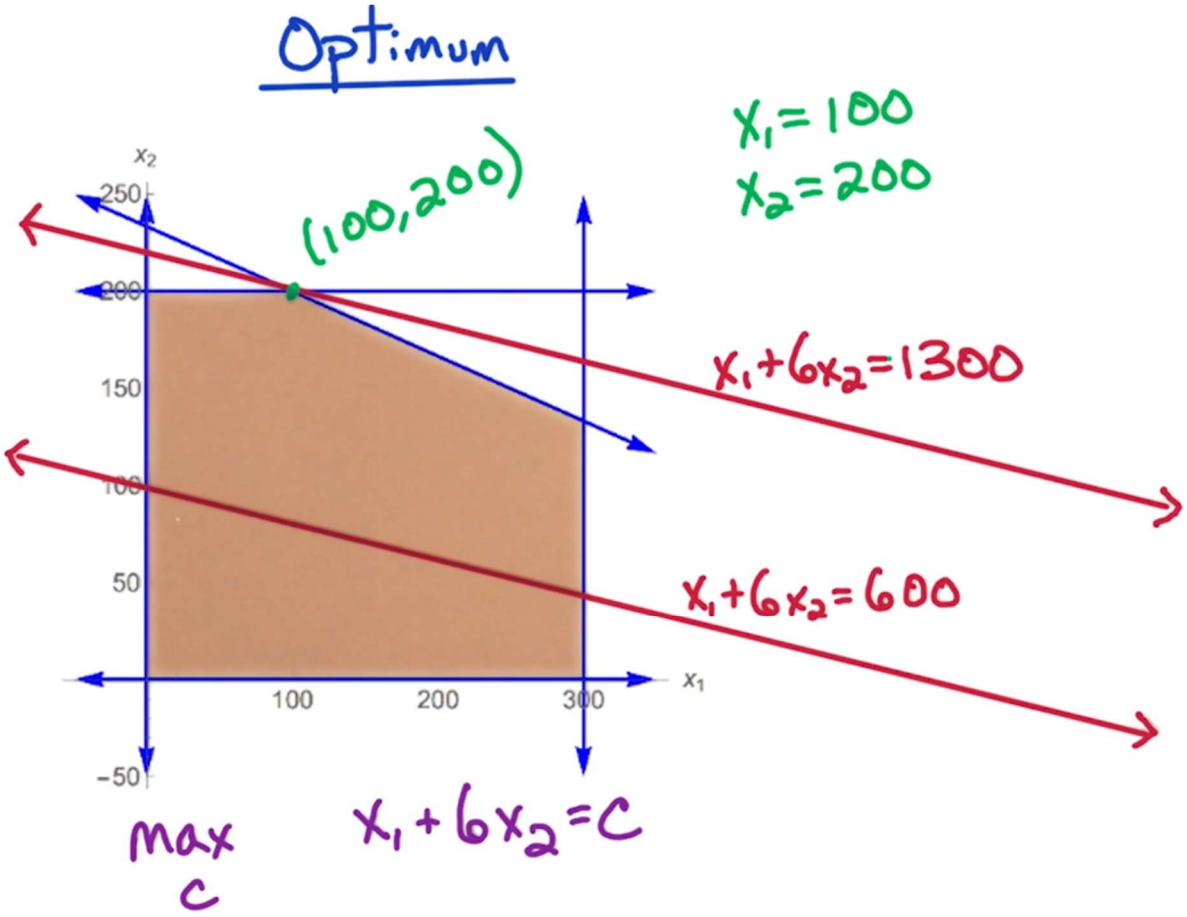
- Now, our first constraint says that x1 is at most 300. So, we have this line x1=300 and we're looking at the left side – points on the left side of this line.
- The second constraint is that x2 <= 200, so we're looking at all points below this line and we want points satisfying both constraints. So, we're looking at the points in the bottom left quadrant of these two vectors.
- Next, we have the non-negative constraints - putting in both non-negative constraints.

Now we lie in this rectangle.

- And, then we have our final, our fifth constraint which says that we lie on this side of this line. Hence, this tan region is our feasible region – all the points in this tan region … in this convex.

***

(Slide 8) Optimum



Now, how do we find the optimal point? Our goal is to maximize this function x1 + 6x2.

So, let's look at this line x1 + 6x2 = C. We're trying to maximize the C, which intersects this feasible region.

So, let's find the maximum value of this C, where this linear function intersects the feasible region. So, let's take a look at this function, this linear function for different values of C. Now, here's the line x1+6x2 = 600. Now, all these points on the line which intersect the feasible region, so these points, all satisfy the five constraints and achieve profit of 600. Now, clearly these are not optimal because we can move up this line. Now, we can think of moving up this line into the last point when we intersect the feasible region.

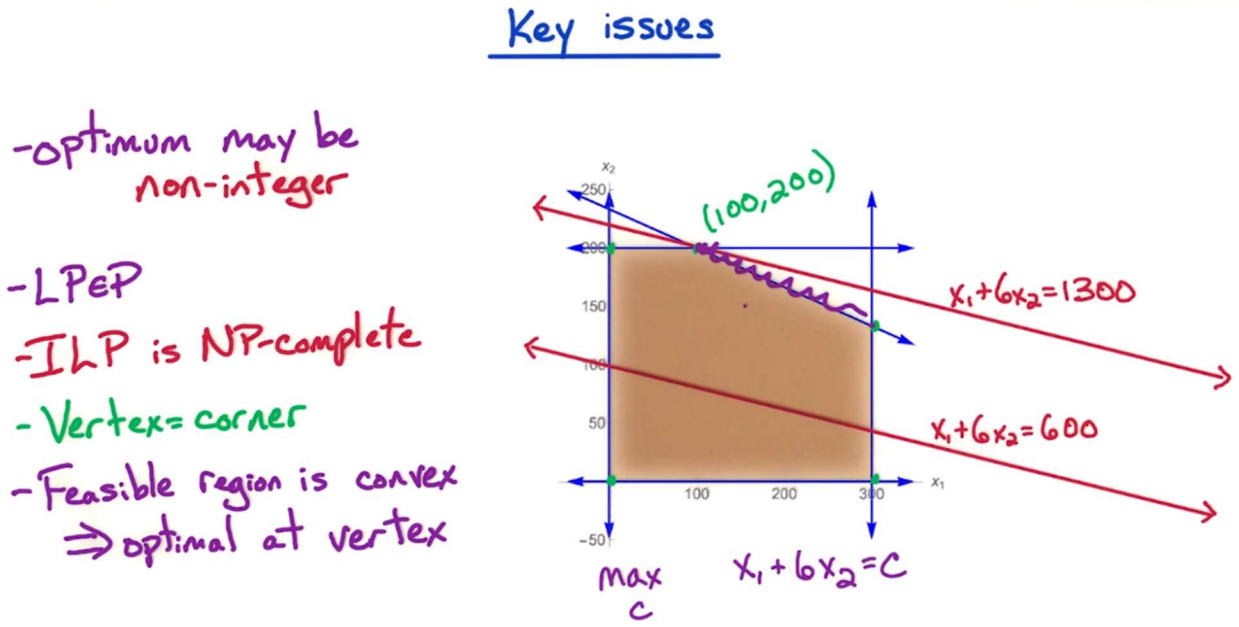
Now, it turns out the maximum value of C for which this line intersects the feasible region is C = 1300. This line x1 + 6x2 equals 1300 intersects the feasible region at this corner of the polygon -

this vertex of this feasible region. This is the point (100, 200). So, we know the optimum is at the point x1 = 100 and x2 = 200 and that achieves a profit of 1300.

Now, this was a simple 2-dimensional example. Let's look at a simple 3-dimensional example and then we'll get some idea of how it generalizes to higher dimensions.

***

(Slide 9) Key Issues



Before moving on, let's use this simple example to identify a few important issues.

- The first and perhaps the most important issue is that we're optimizing over this entire feasible region - this tan set here. Now, in this example, the optimum happened be at an integer point, this point (100, 200). So x1 and x2 are integer values, but we have no way to only consider solutions lying on integer points. We can only find the best point in this whole region.

  For example, for a different objective function, the optimum may lie at this vertex. At this vertex. x1 is an integer value, it's 300, but x2 is a fractional value. In such a scenario, what do we do if we only want to consider integer points such as in our example for the production situation? Well, if we end up with a fractional point, which is the optimal point, we could try to round it to an integer point. That's something we'll do in the lecture LP4 on the Max-SAT Approximation Algorithm. The key point is that Linear Programming optimizes over this entire set and when we optimize over this entire set, then the problem is polynomial time solvable.

  So, Linear Programming LP is in the complexity class P. Now, the corresponding problem where we look for the best integer valued solution is called Integer Linear Programming and is denoted as ILP. The ILP problem is NP-complete. So it's unlikely that we can solve it in polynomial time. We'll see this fact that ILP is NP-complete in

the Max-set Approximation Algorithm lecture.

- Another key idea is that the optimal point for this linear program lies at a vertex of this polygon. What does a vertex mean? A vertex means a corner of this polygon. So, there are five vertices of this polygon: one, two, three, four, and five. Why is it the case that the optimum must lie at a vertex of this polygon of this feasible region?

  Now, it may be the case that other points are also optimal. We're just saying that vertices are at least as good as any other point. Look at this line for the max C achievable. In this case, C equals 1300. Suppose this optimal line intersects a point such as this point z which is not a vertex of this polygon. Well, this point z lies on on edge of the polygon. Therefore, it must be the case that one of these two endpoints of this edge must be better than this point z, or it has to be the case that this optimal line intersects this edge. If this optimal line intersects this edge, then all the points on this edge are optimal. Therefore, the two endpoints are both optimal and therefore an optimal point lies at a vertex.

- The last important concept is convexity. This feasible region, this tan region is convex. What exactly does convexity mean? Well, if we take any two points in this set and we look at the line connecting them, then that entire line - that entire edge is contained in this set. This means that we don't have a shape like this. The feasible region can't go down and then back up because if we take a point over here and a point over here and we look at the line - the edge connecting them - then it goes outside the set. So this is a non-convex set.

  Because the feasible region is defined by the intersection of half spaces, it must be a convex set. We can't get these nasty shapes. Since it's convex, if we have a vertex such as this point,(100, 200), which is better than its neighbors, these two points, well, the feasible region can't go down and then back up. So, if this point is better than its neighbors, then the entire convex region - the entire feasible region - is below these two lines. Therefore, this point is optimal because if a point is better than its neighbors, then that vertex is an optimal point.

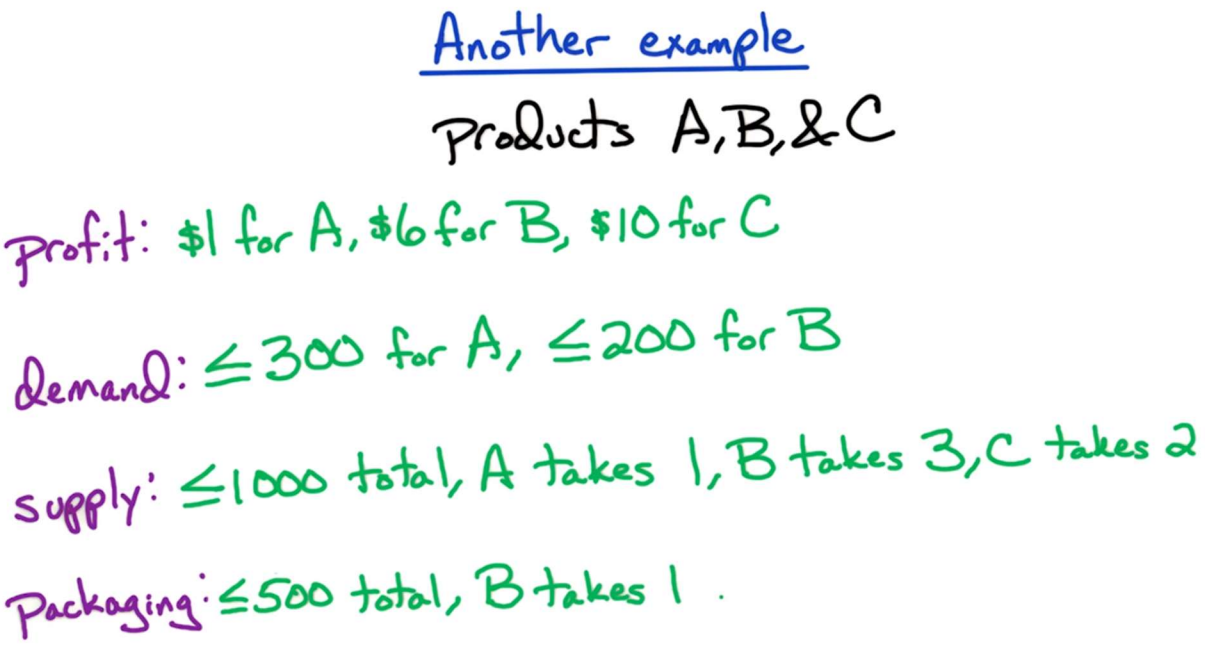
  That's the key point of convexity, the optimal point always lies at a vertex of this feasible region. There may be other points which are optimal, for example, this entire line might be optimal, but there always is a vertex of this feasible region. In this case, these two points which will be optimal. This concept that an optimal point lies at a vertex of the polygon and the key fact that if a vertex is better than all of its neighbors, then it's, therefore, the global optima.

This leads to the **simplex algorithm** which we'll detail later, but let's give a brief sketch of it. It's sort of a local greedy approach. We'll start at some vertex of the polygon. We'll check its neighbors and we'll look if any of its neighbors are better. Then we'll move to a neighbor which has a better objective function and we continue until we find a vertex of the polygon in the feasible region which is better than all of its neighbors and therefore it's optimal. That's the basic idea of the simplex algorithm.

Now let's move on to another 3-dimensional example.

***

(Slide 10) 3D Example



Let's take a look at another example before giving the general formulation of Linear Programming. Now this will be a slight twist on the previous example.

Now we have three products A, B and C. We'll look at the profit obtained from each - the demand for each - the supply just as before, and we'll add one additional constraint compared to before - this will be a packaging constraint.

- profit: We'll say the profit is one dollar for each unit of A that we sell, six dollars for each unit of B and ten dollars for C.

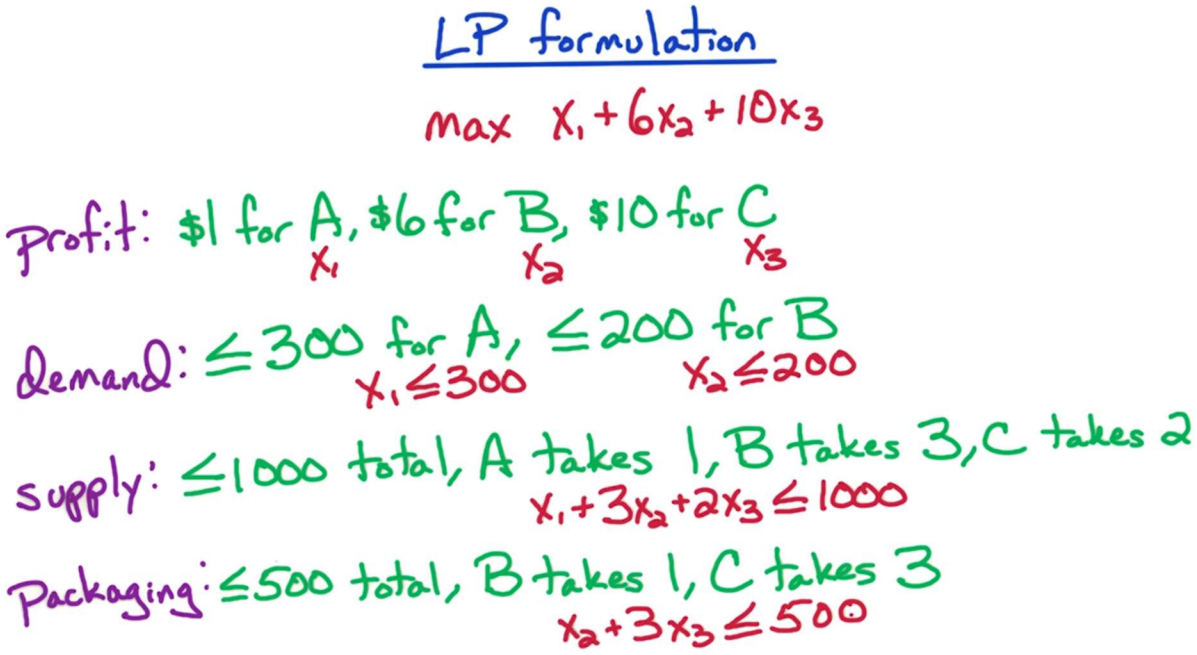
- demand: The demand is at most 300 units per day for A and at most C.

- supply: The supply that we can produce is at most 1000 hours per day. Each unit of A takes one hour, each unit of B takes three hours to produce and each unit of C takes two hours to produce.

- Finally, the packaging. Now we can produce at most 500 units in total per day. We'll say that A doesn't require any packaging, B uses one unit of packaging per unit and C takes three units of the packaging per unit.

***

(Slide 11) 3D LP Formulation

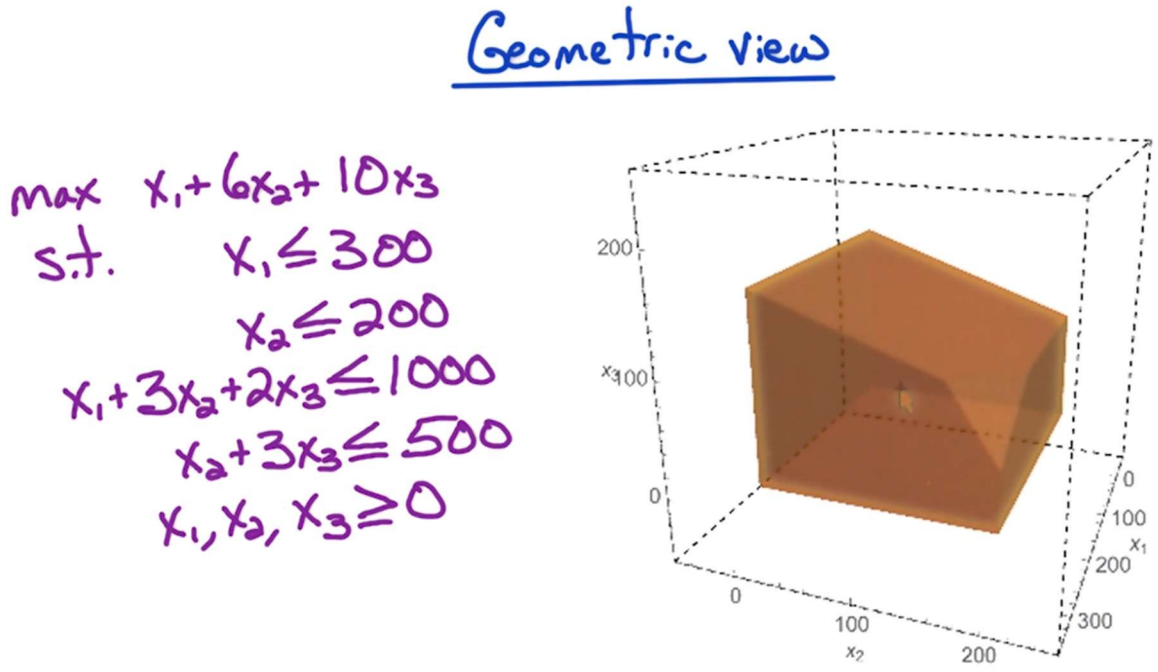


Now, let's formulate this problem as a linear program.

- Now, we're going to have three variables x1, x2, x3, corresponding to how much we produce of A, and B, and C, respectively, per day.
- Now, our goal is to maximize the profit. So, our objective function is going to mean maximize x1 + 6 x2 + 10 x3 based on our profit assumption.
- Now, we have two demand constraints. x1 is at most 300 and x2 is at most 200.
- Now, we have one supply constraint, x1 + 3 x2 + 2 x3 is at most 1000.
- And, finally, the packaging constraint says that x2 + 3 x3 is at most 500.
- And, let's not forget the non-negativity constraints. x1, x2, x3 all have to be non-negative.

***

(Slide 12) 3D Geometric View



Let's recap our LP once again.

- Our objective function is to maximize x1 + 6 x2 + 10 x3.
- Subject to the following seven constraints.
    - We have the two demand constraints.
    - We have the supply constraint,
    - the packaging constraint
    - and, finally, the non-negativity constraint.

Now, let's take a look at this geometrically. Now, we have three variables, so we're going to lie in 3-dimensional space. Now, if we had equality for one of these constraints, there would be a hyperplane, a 3-dimensional hyperplane.

Now, with an inequality, we're taking one side of that hyperplane. So, that's a half space. So we're looking at the intersection of these seven half spaces. This is a geometric view of the feasible region.

The points in this ten polyhedron are the feasible axes. These are the points which satisfy all seven constraints.

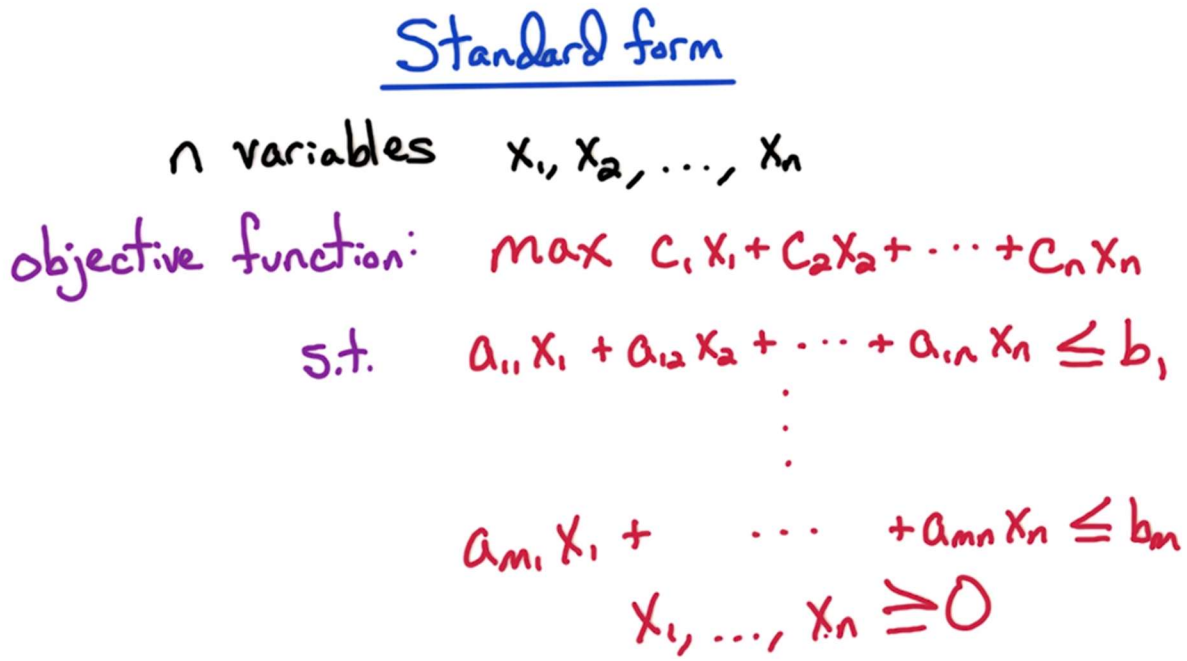
Now, notice this is a convex set, so, it can't go down and then back up. Convexity is an

important property for the algorithms to solve linear programs.

Now, let's take a look at the general formulation of linear programs and then we'll come back to this example and we'll figure out the optimum for this example.

***

(Slide 13) Standard Form



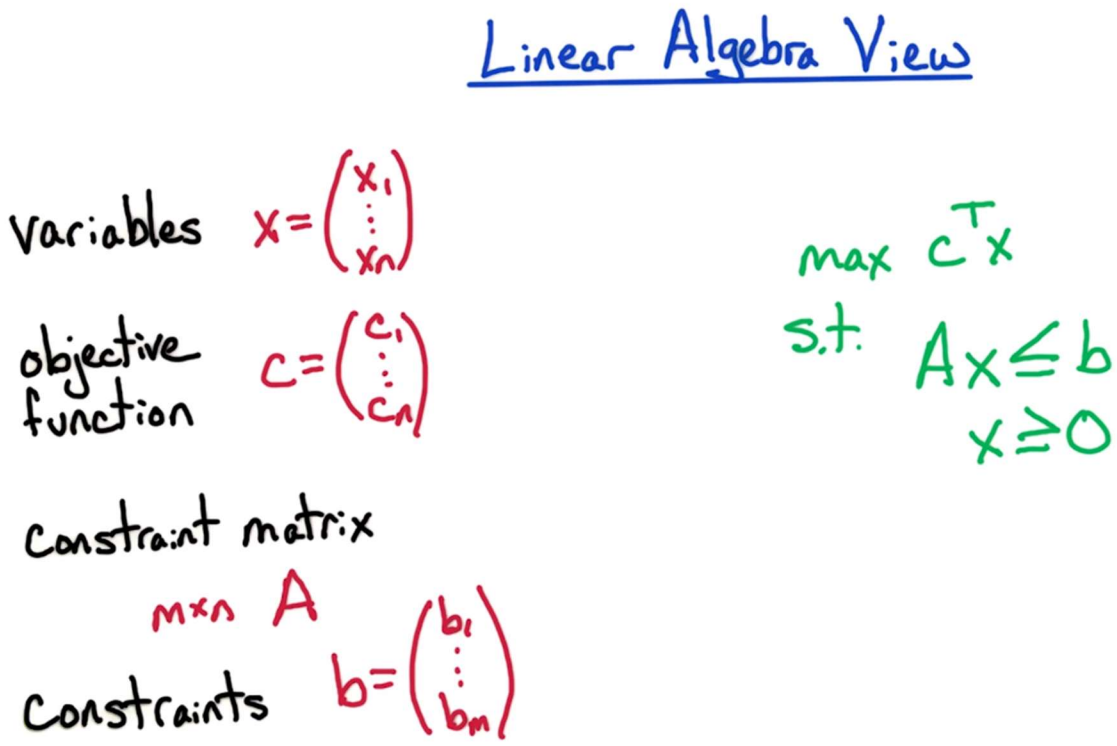
Let's look at the standard form for linear programs.

- We have n variables x1 through xn.
- Our objective function is to **maximize** a linear function of these n variables.
- We have these coefficients, c1 through cn, and we're maximizing this linear function of the n variables.
- We're optimizing subject to the m constraints.
    - The first constraint is specified by a11 through a1n. These are the coefficients for x1 through xn. We want that this linear function is at most b1 for a given b1. This is the first constraint.
    - There are m constraints. The mth constraint is specified by am1 through amn, that specifies the left hand side and we want this to be at most bm.
    - And finally, we also have the non-negativity constraints. We want that these n variables are non-negative.

We have m+n constraints. We can specify this more compactly using linear algebra. Let's go ahead and do that.

***

(Slide 14) Linear Algebra View



Now, the variables are specified by this column vector x. The objective function is specified by these coefficients - this column vector c, and hence our objective function is to maximize $c^T x$ (c transpose times x).

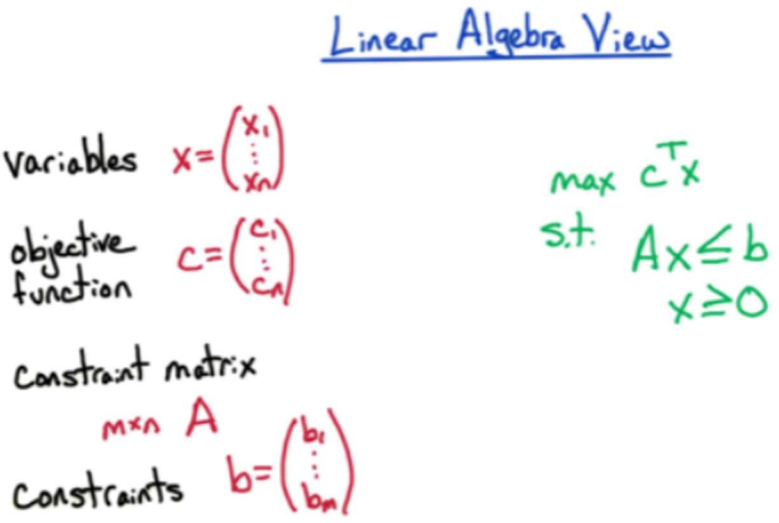
Then we have our constraint matrix. This is an m x n matrix A. Finally, we have the constraints for the right hand side. This is specified by this column vector, b1 through bm. There are m constraints, so this vector has size m.

Going back to our LP formulation, we have our objective function. This is subject to the following constraints: We want those x where $Ax \leq b$, and we have the non-negativity constraint, x is non-negative.

Now the last constraint is important, the non-negativity constraint. Because, if the feasible region is non-empty – so, there are some feasible x - then we know that the zero vector is a feasible point. So we can trivially find a feasible point or determine that the feasible region is empty, and therefore this LP is infeasible.

***

LP1: Linear Programming: Linear Algebra View



Question 1: What is the Objective Function (vector)?

Question 2: What are the Constraints?

- Objective function: **maximize** $x_1 + 6x_2 + 10x_3$
  maximize x1+6x2+10x3
- Constraints:

$$x_1 \leq 300$$

$$x_2 \leq 200$$

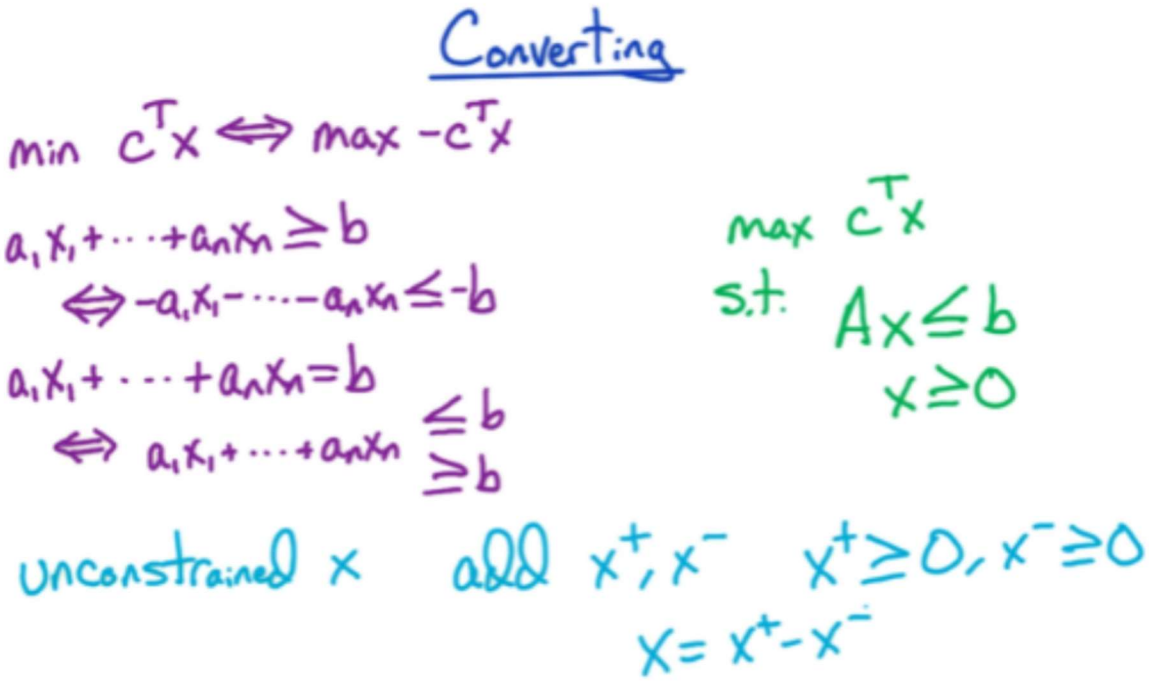
$$x_1 + 3x_2 + 2x_3 \leq 1000$$

$$x_2 + 3x_3 \leq 500$$

$$x_1, x_2, x_3 \geq 0$$

***

(Slide 15) Converting to Standard Form



Now, how can we convert an arbitrary linear program into this standard form?

- For instance, suppose that we want to minimize this linear function instead of maximize this linear function. Well, we simply multiply this whole thing by negative one. And this is equivalent to maximizing $-c^T x$. So, we can replace this vector c by -c and then we can maximize and it's equivalent in LP.

- Now, what if we have a constraint that says a1 x1 through an xn is at least b, instead of at most b. Once again we can multiply this by -1. And this is equivalent to -a1 x1 through -an xn is at most -b.

- Now, what if we have an equality constraint? So we want to set it equal to b, where we can replace this by two inequality constraints. We can say that a1x1 through anxn is at most b and at least b. And the second of these – the at least b constraint - we can replace in this manner by an inequality at most -b.

- Now, what if we have a strict inequality? So we say that x is strictly less than 100. How do we convert this into a non-strict inequality?

  Well, suppose the LP just said maximize x subject to the constraint that x is strictly less than 100. What's the solution, what's the optimal value of this function, this linear

program? Well, it's not clear what the optimal is. So this is an ill-defined linear program. So strict inequalities are not allowed in Linear Programming.

Now, geometrically the way to view these strict inequalities. Well, first off look at these non-strict inequalities. In this case ($Ax <= b$), the feasible region corresponds to this closed convex polyhedron. So the optimal points will lie on the boundary of this polyhedron. But if we have strict inequalities, then the points on the boundary of the polyhedron do not lie in the feasible region. So, it's an open set.

And the problem is ill-defined because we don't know what the optimal is. So strict inequalities are not allowed.

- Finally, what do we do if we have an unconstrained variable x? So, x can be positive or negative. We don't have this non-negativity constraint ($x >= 0$). How do we add in this non-negativity constraint?

  Well, we create two new variables, the positive magnitude of X and the negative magnitude of x, x+ and x-. We constrain x+ to be non-negative and x- to be non-negative. So, these correspond to the magnitudes and we replace x by x+ - x- ($x = x+ - x-$).

  And, in this manner, we can consider all variables to be non-negative.

***

LP1: Linear Programming: Converting to Standard Form

- Objective function: $\textbf{minimize } 3x_1 - 2.5x_2 + x_3$
  - If you have trouble viewing LaTeX, here is a plain text version of the Objective function: minimize $3x_1-2.5x_2+x_3$
- Constraints:

$$x_2 + x_3 \geq 300$$

$$0.5x_1 + 7x_2 - 2x_3 = 4$$

$$x_1, x_2, x_3 \geq 0$$

  - Plain Text Version of Constraints:

    $x_2 + x_3 \geq 300$

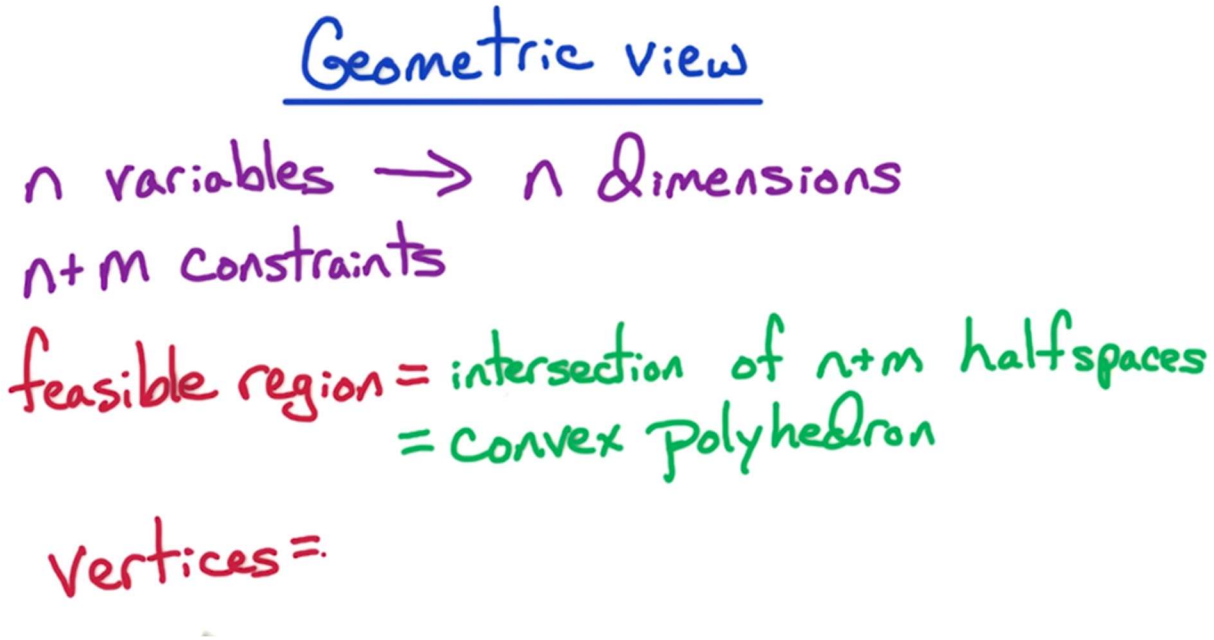
    $0.5x_1 + 7x_2 - 2x_3 = 4$

    $x_1, x_2, x_3 \geq 0$

Convert this LP into standard form, and then express the LP in the Linear Algebra view below.

***

(Slide 16) General Geometric View



Now, in general, we have n variables. So we're going to lie in n-dimensional space. Now n is large, so we're in high dimensional space.

Now we have m constraints plus the n non-negativity constraints. So we have a total n+m constraints.

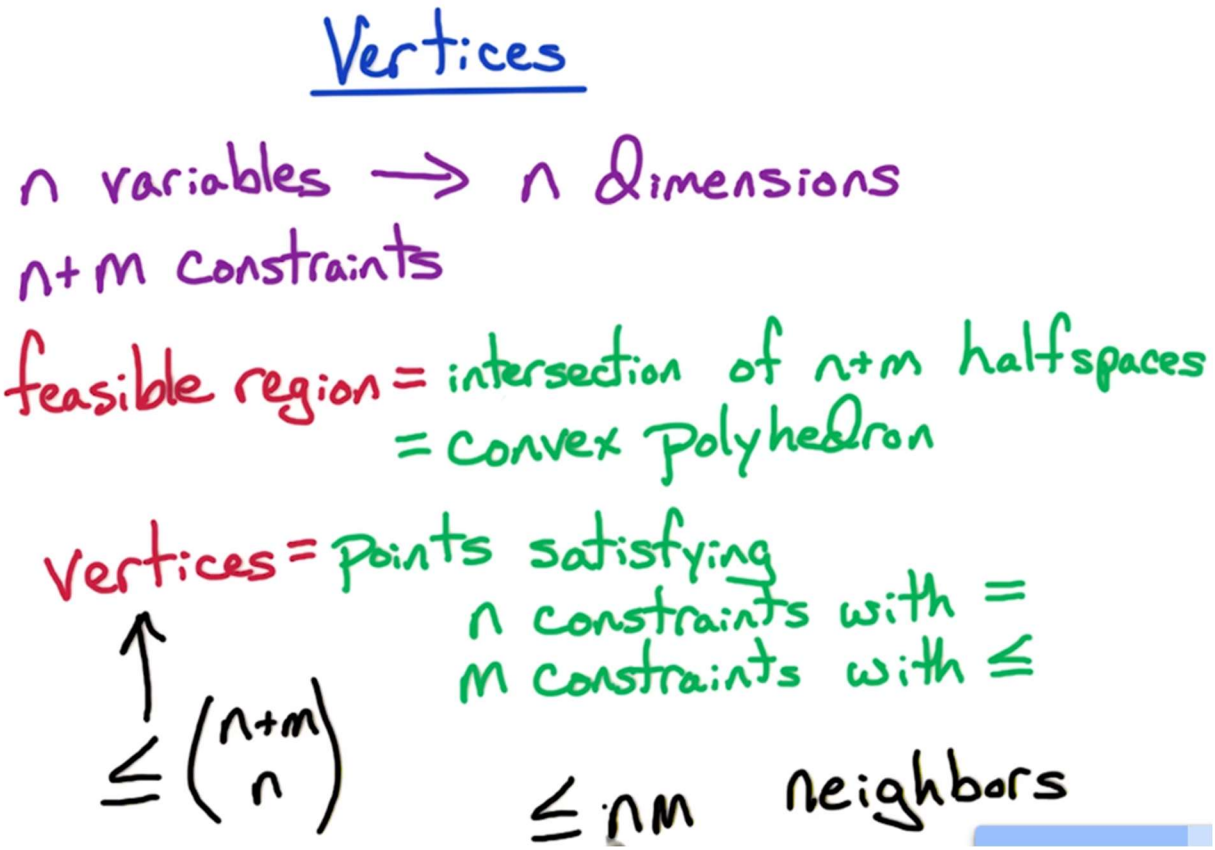
Now what's our feasible region? Now, each constraint corresponds to a half space in n dimensions. Now the feasible region are those points which satisfy all n+m constraints. So we want to lie in all n+m half spaces. So, the feasible region corresponds to the intersection of these n+m half spaces.

Now this is going to correspond to a convex polyhedron in n dimensional space. Now the vertices of this polyhedron are the corners of it.

Now how do you get a vertex of this polyhedron?

***

(Slide 17) Vertices



Now, how do you specify the vertices of the feasible region?

Well, let's take a look back at our simple 2-dimensional example. How do I specify this vertex of this polygon? It satisfies these two lines with equality. The first line is x1 = 300 and the second line is x1 + 3 x2 = 700.

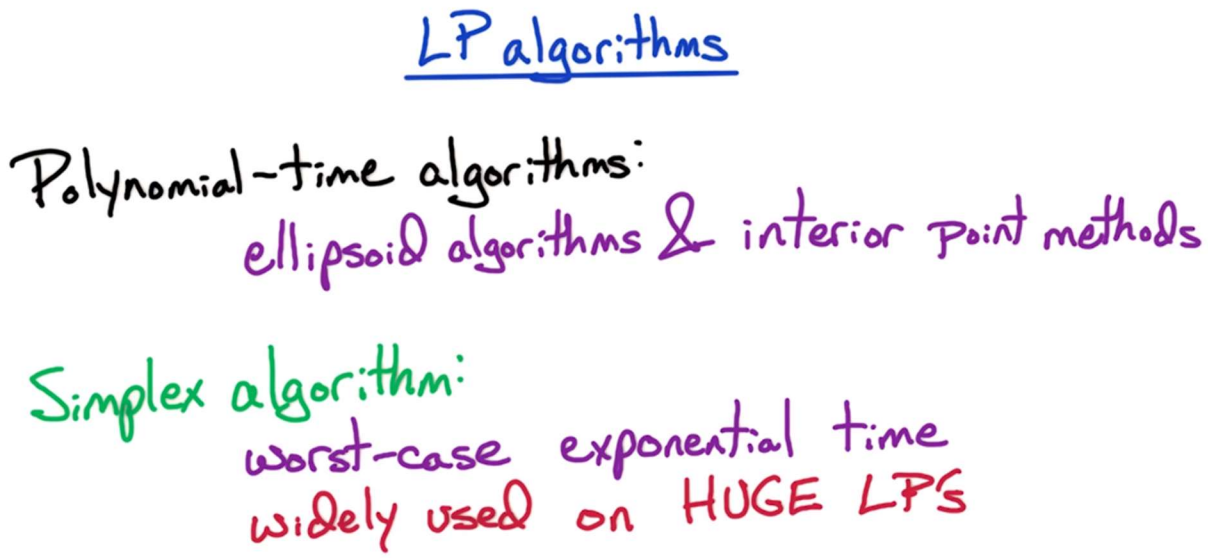
Now, satisfying these two constraints with equality specifies this point; but, then I have to check that this point satisfies the other inequalities. So, in general, in n-dimensional space, a vertex of this convex polyhedron is specified by specifying the end constraints that you satisfy with equality, and then we have to check that the remaining m constraints are still satisfied.

Now, we can get a trivial upperbound on the number of vertices of this convex polyhedron. We have to specify the constraints that are satisfied with equality. There are n+m constraints to choose from. So the number of vertices is most n+m. Choose n - this is exponential in n. So, in the worst case, there are a huge number of vertices in this convex polyhedron, in this feasible region.

Now for a particular vertex, how many neighbors does it have? Can we get an upper bound on the number of neighbors for a specific vertex? Well, "neighbor" would correspond to swapping out one constraint with equality with a different constraint. So, we have n choices for which one we swap out, and then we have m choices for which when we swap in. So the number of neighbors for a specific vertex is at most nm.

***

(Slide 18) LP Algorithms



Now, let's give a quick overview of the type of algorithms for solving linear programs.

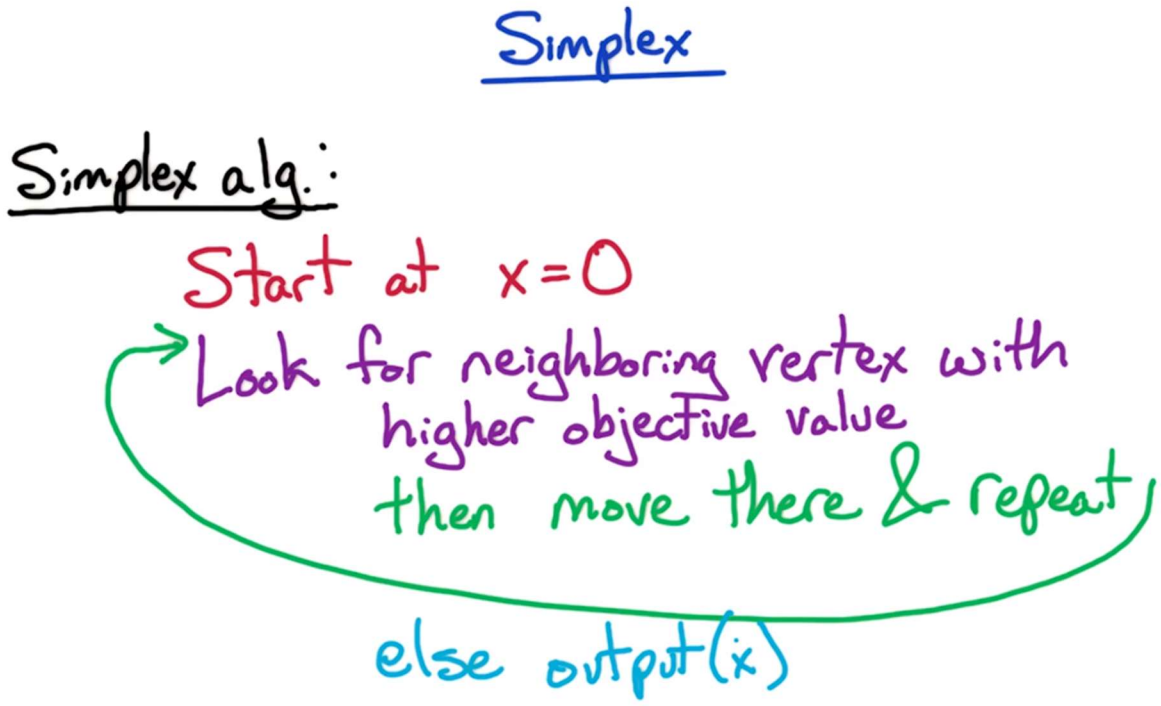
Now there are two types of methods which are guaranteed to solve linear programs in polynomial time in the worst case. These are the **ellipsoid algorithm** which was the first example of a polynomial time algorithm for Linear Programming and interior point methods. Now ellipsoids algorithms are more of theoretical interest, but interior point methods are used quite extensively right now.

Now an important algorithm is the **simplex algorithm**. As a disclaimer for the simplex algorithm is that in the worst case it takes exponential time. Despite that worst case bound, simplex algorithm is widely used because the output of the simplex algorithm first off is guaranteed to be an optimal. So when the simplex algorithm completes, it's guaranteed to give an optimal solution to the linear program and it works quite efficiently, very fast on a humongous LPs. There are very fast LP solvers using the simplex algorithm.

We're going to take a look at the high level idea of the simplex algorithm.

***

(Slide 19) Simplex Algorithm



Let's take a look at the basic idea of this simplex algorithm.

- We want to start at some feasible point. Well, we start at the zero vector. This is the vertex of the feasible region, because it satisfies the n non-negativity constrains with equality.

  Now, of course we still have to check that this point satisfies the other m constraints. Now, if it does not satisfy any of the m constraints, then we know that this point is not in the feasible region and therefore the feasible region is empty. So this LP is infeasible. There are no x which satisfy all of the constraints.

- Now, this starting point is feasible. Then we're going to do a sortof local search.

  So, we're going to look at the neighboring vertices and we're going to try to find one with higher objective value. What do we mean? We mean that the value of the objective function is higher and we want one which is strictly higher.

  Now recall there are most N times M neighboring vertices. So, in the worst case we can check all the neighboring vertices.

- Now, if we find a neighboring vertex with higher objective value then we move there and then we repeat. So, for this new point, we look for a neighboring vertex which has higher objective value. And we continue, keep repeating.

Now, what do we do if there is multiple neighboring vertices with higher objective value? Well, that's one of the heuristics of the variance of simplex algorithm. We could choose one of them at randomly. We could take the neighboring vertex with highest object of value. There's various heuristics that we can use.

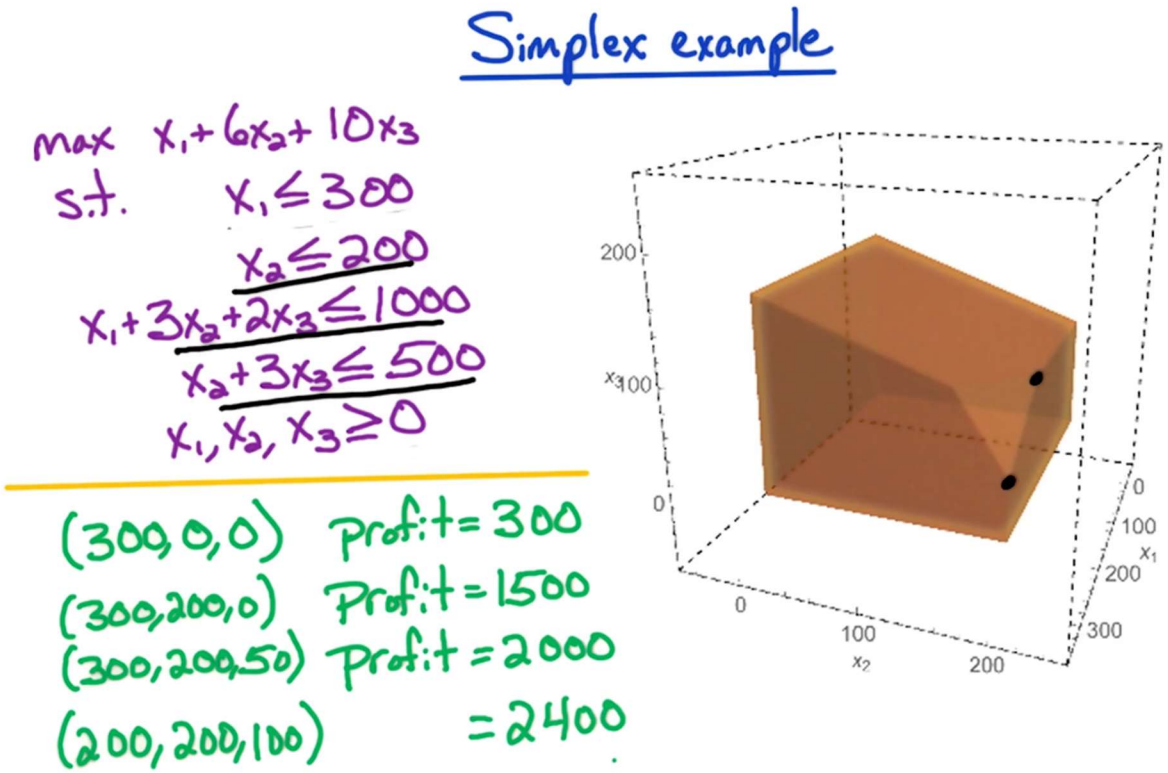
But what happens in the case that this current point is better than all of the neighboring vertices? Well, recall our feasible region is a convex polyhedron and we're walking on vertices of this convex polyhedron. So we're at some vertex and all the neighbors are smaller than it. Since it's convex, it can't go down and then back up. So, if all the neighbors are smaller than it, then the whole convex, the whole feasible region is smaller than it. So, the current point must be the optimum.

So, we simply output the current point and this is guaranteed to be the optimal, the global optimal of the LP.

To solidify the idea of the simplex algorithm, we're going to illustrate it on one of our examples. So, let's go back to our three dimensional example, our example with three variables and implement the simplex algorithm on that example.

***

(Slide 20) Simplex Example



Now, here's our earlier example with three variables, and let's illustrate the simplex algorithm on this example.

The feasible region is this 3-dimensional convex polyhedron. How does simplex start?

Simplex Method:

- Well, we start at the point (0, 0, 0) - it satisfies all of the constraints, and the value of the objective function for this point is zero. So, we have profit zero for this point. Now, where is this point in the feasible region? It's this point in the back. Now, this point is defined by these three inequalities.

  We satisfy these three constraints with equality, and the remaining constraints, the remaining four constraints, we satisfy with inequalities.

  Let's look at a neighboring vertex. Let's look at this vertex. This vertex is defined by this constraint with equality, and these two non-negativity constraints but we dropped that one. So we're looking at the vertex defined by this first constraint with equality, and

these last two non-negativity constraints with equality. This corresponds to this vertex of the feasible region, and this is the point (300, 0, 0).

- So when we consider it from the zero vector, then we're going to move here because it has higher profit.

  Next, let's consider this neighboring vertex of the current point. This new vertex satisfies these first two constraints with equality, and the last constraint. It corresponds to the point (300, 200, 0). And if you plug it into the objective function, it has profit 1500.

- The profit is higher, so we move to this new point.

  Now let's see if there's a neighbor of this new point which is better. Let's consider this neighbor. This is at the intersection of these three faces, these three faces are the first three constraints. This is the point (300, 200, 50).

- The value of the objective function for this point is 2000, so we move there.

  From the current point, let's try a new neighbor. Consider this neighboring vertex - this vertex is defined by equality by these three constraints, and it satisfies the other constraints, so it lies in the feasible region. It corresponds to the point (200, 200, 100). You can see that because x2 = 200. And then once x2 = 200, and this constraint says that x3 = 100. And then finally, this specifies that x1 = 200.

- And then if you check the value of the objective function, we get that it's 2400, so we move to this new point.

  Finally, we check all of the neighbors of this vertex, and we see that all of the neighbors are worse.

So we know that we're at the optimal point and we successfully solved this LP.