

# Session 6

---

Corbet Elkins

October 23, 2026

Purdue University

## Questions over lecture content?

See title

# Introduction to LPs

Lecture only covered 2D LPs, but most LPs of interest are going to be higher dimensional.

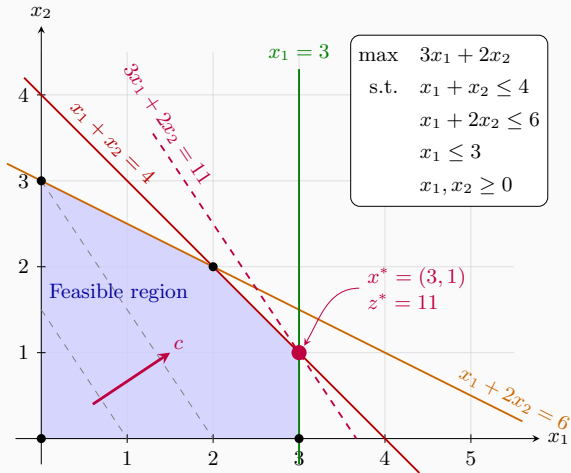
In general, an LP is something of the form:

$$\begin{aligned} &\text{maximize } c^\top x \\ &\text{such that } Mx + b \leq y \end{aligned}$$

Where  $c, M, y$  are fixed, and we want to find  $x$ . Here  $M \in \mathbb{R}^{m \times n}$ ,  $c, x \in \mathbb{R}^n$ ,  $y \in \mathbb{R}^m$ .

Note we can get  $\geq$  requirements by flipping signs and equality requirements by imposing a  $\geq$  and  $\leq$  requirement.

## Example LPs



## Polygons and Polytopes oh my

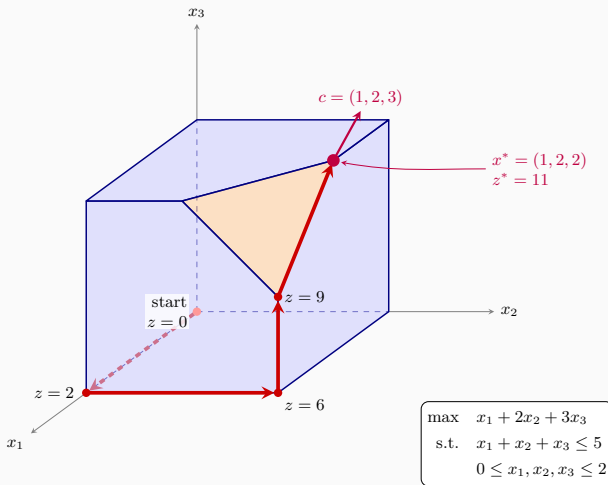
Generalizing from 2D to higher dimensions, each constraint specifies a half space bounded by a hyperplane.

The feasibility region of these half spaces will form a polyhedron (or a polytope if bounded).

Objective level sets become hyperplanes, and if the optimum exists, it is at a vertex.

For simplicity (and to ensure our LP has a max), we will suppose that our feasibility region forms a polytope.

## 3D example



# What are vertices

What does a vertex look like in 2D? 3D?

How do we generalize this to higher dimensions, and how do we find them?

# What are vertices

What does a vertex look like in 2D? 3D?

How do we generalize this to higher dimensions, and how do we find them?

A first thought is that any  $n$  equations in  $n$ -dimensional space will form a vertex, but this isn't quite true since we could have two equations that are equal.

Thoughts on what additional requirement we need to have?

# What are vertices

What does a vertex look like in 2D? 3D?

How do we generalize this to higher dimensions, and how do we find them?

A first thought is that any  $n$  equations in  $n$ -dimensional space will form a vertex, but this isn't quite true since we could have two equations that are equal.

Thoughts on what additional requirement we need to have?

If we instead require  $n$  linearly independent equations, then they will form a vertex.

## Vertices (cont.)

Intuitively, intersecting  $k$   $n$ -dimensional hyper planes, gives an  $n - k$ -dimensional affine subspace.

For example intersecting two 2-dimensional subspaces gives a line.

A vertex is a 0-dimensional affine subspace, thus we need  $n$  linearly independent constraints to fully constrain the resulting equation.

## Finding a vertex

We call a constraint  $c_i$  tight, if  $c_i^\top p = y_i$ , the tight set is the set of constraints which are tight.

The algorithm for finding a vertex is as follows, supposing you know a point  $p$  is in the feasible set:

1. Let  $A_I$  be the submatrix of  $A$  with tight constraints. If  $\text{rank}(A_I) = n$ , then  $p$  is a vertex and we stop.
2. Otherwise, we pick a direction  $d$  such that  $A_I d = 0$ , this ensures that the constraints remain tight.
3. We let  $p = p + xd$  where  $x$  is the smallest set to make another constraint tight.
4. If no such constraint is tight, then the polytope has no vertex.
5. Repeat.

# Simplex Method

Suppose we have a vertex  $x$ :

1. Pick a basis  $B$  of tight constraints. Let  $A_B$  be the matrix of these constraints.
2. Solve  $A_B \lambda = c$ .
3.  $\lambda \leq 0$ , then  $x$  is optimal.
4. Otherwise, pick an  $i$  such that  $\lambda_i > 0$  and let  $d = A_B^{-1} e_i$ .
5. Move  $x$  in the direction of  $d$  until we have another tight constraint.
6. Repeat

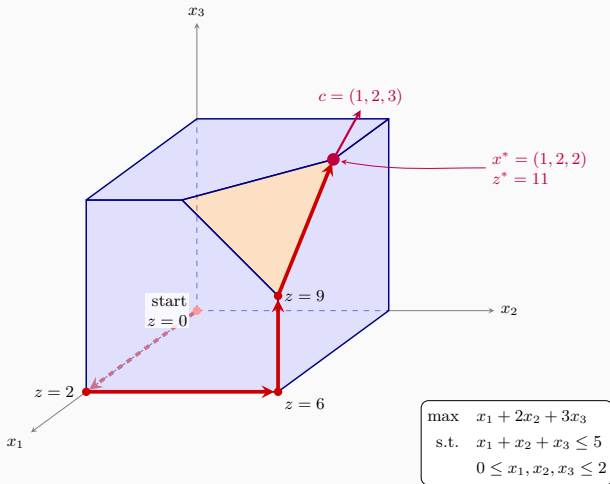
## Correctness of Simplex Method

First we show that the simplex method will always terminate.

Note that after each iteration, as long as  $x$  moves, we are always increasing the objective function and getting a new vertex. As there are only a finite number of vertices, this will terminate.

The issue is if  $x$  does not move, but by always picking the lowest index eligible bases, we can avoid this.

# Simplex Example



## Issue with Simplex

In practice the simplex method is pretty fast. On average inputs, it runs in polynomial time.

However, for certain inputs, it can take exponential time.

For complexity purposes, there is an algorithm (the ellipsoid method) which runs in polynomial time, but it is not practical in real life.

## LP Rounding of Vertex Cover

Suppose you have a graph  $G = (V, E)$ , a vertex cover of the graph is a subset of vertices  $V' \subseteq V$ , such that for every edge, at least one endpoint of the edge is in  $V'$ .

Solving this problem is hard (more on this in a couple of weeks), but using LPs we can get an approximate solution in polynomial time.

## Vertex cover of an ILP

An integer linear program is like a linear program, but we require  $x$  to be an integer vector.

e.g. minimize  $x_1 + x_2 + x_3$  such that  $x_i + x_j \geq 1$ , for  $i \neq j$  with  $x_i \in \{0, 1\}$ .

How can we write the vertex cover problem as an ILP?

## Vertex cover of an ILP

An integer linear program is like a linear program, but we require  $x$  to be an integer vector.

e.g. minimize  $x_1 + x_2 + x_3$  such that  $x_i + x_j \geq 1$ , for  $i \neq j$  with  $x_i \in \{0, 1\}$ .

How can we write the vertex cover problem as an ILP?

Let  $x_i \in \{0, 1\}$  be a variable that indicates if  $v_i$  is in our cover.

Then we want to minimize  $\sum_{i=1}^{|V|} x_i$ , and for each edge  $(v_i, v_j)$  we want  $x_i + x_j \geq 1$ .

integer linear programs are also very hard to solve, but what if we just pretend they are linear programs.

i.e. instead of requiring  $x_i \in \{0, 1\}$ , we have  $0 \leq x_i \leq 1$

Now we have an LP and can solve it using an LP solver.

## Approximating the solution

Let  $OPT$  be the optimum solution to the vertex cover problem, and let  $OPT_{LP}$  be the solution to the LP relaxation.

1. Prove  $OPT \geq OPT_{LP}$ .
2. Prove  $OPT \leq 2OPT_{LP}$ .<sup>1</sup>

This shows that by using an LP relaxation, we can get within double the optimal solution to vertex cover.

---

<sup>1</sup>Consider rounding each  $x_i$  up if  $x_i \geq 1/2$  and down otherwise.