Math 3272: Linear Programming[1] Mikhail Lavrov

Lecture 1: What is a linear program?

August 16, 2022 *Kennesaw State University*

1 Linear programs and optimization

"Linear programming" suggests telling a computer to do linear stuff. But that's not what the word "programming" means in the title of this class. In this context:

- "programming" means "optimization": finding a point that maximizes (or minimizes) the value of a function.
- "linear" means that the functions we optimize will be linear, and the constraints on our optimization will all be linear equations or inequalities.

Today, we will take an example linear program from formulation all the way to finding a solution, and see some basic ideas of linear programming along the way.

Problem 1. *At a major music company, you are in charge of hiring for the xylophone department and the yodeling department. You want to change the number of employees in the xylophone department by x and in the yodeling department by y. (Both x and y can be positive or negative: you can hire people or you can fire people.)*

Right now, the xylophone department is doing well: each employee is bringing you \$1000 *in profit each day. However, the yodeling department is actually a loss for the company: each employee is losing* \$300 *for the company each day.*

What should you do to maximize profit, given the following constraints?

- $x + y \leq 50$*: you don't have office space to increase the total size of the departments by more than* 50*.*
- $y \geq -20$*: the yodeling department will just stop functioning if it loses more than* 20 *people.*
- *The yodelists' union has forced you to agree to the constraint* $2x - y \leq 40$ *in a recent negotiation.*

Here's the problem written down without words:

$$\begin{aligned} \underset{x,y\in\mathbb{R}}{\text{maximize}}\ & 1000x - 300y \\ \text{subject to}\quad & x + y \leq 50 \\ & y \geq -20 \\ & 2x - y \leq 40 \end{aligned}$$

(A subtle issue: we've written $x, y \in \mathbb{R}$ but actually x and y should be integers: you can't hire $\frac{1}{2}$ of a yodelist. We'll ignore this problem today and return to it near the end of the semester.)

[1] This document comes from an archive of the Math 3272 course webpage: http://misha.fish/archive/3272-fall-2022

1.1 The feasible region

The linear program has a set of **constraints**: $x + y \leq 50$, $y \geq -20$, and $2x - y \leq 40$. In general, we will allow our constraints to be linear inequalities or linear equations. (But don't worry about "equations" for now: we'll limit ourselves to inequalities for the moment.)

In linear algebra, we write down a system of equations as a single matrix equation. In linear programming, we will also write down a system of inequalities as a single matrix inequality. Here's how we do it.

First, we should make sure all our variables are on one side, if we haven't done that already. In this example, that's already been done. Second, let's multiply the second inequality by -1, so that all the inequalities are $\leq$ inequalities:

$$\begin{aligned} x + \quad y &\leq 50 \\ -y &\leq 20 \\ 2x - \quad y &\leq 40 \end{aligned}$$

The column of values on the left-hand side can be written as a matrix multiplication:

$$\begin{bmatrix} x+y \\ -y \\ 2x-y \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & -1 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

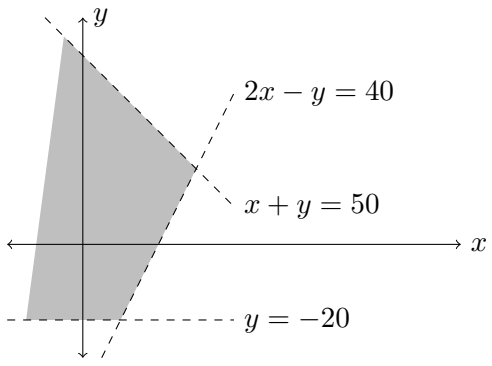
So we will summarize this system of inequalities as the matrix inequality

$$\begin{bmatrix} 1 & 1 \\ 0 & -1 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} 50 \\ 20 \\ 40 \end{bmatrix}.$$

Putting a single "$\leq$" between two vectors is something you might not be used to. What it means is that *every* component of the vector on the left is less than or equal to the corresponding component of the vector on the right.

What happens in general? Suppose our linear program has n variables that, for lack of creativity, we will call $x_1, x_2, \ldots, x_n$. We can put all these variables together into a column vector $\mathbf{x} \in \mathbb{R}^n$. Then any collection of m linear inequalities in $x_1, \ldots, x_n$ can be combined into a matrix inequality $A\mathbf{x} \leq \mathbf{b}$ where A is an $m \times n$ matrix and $\mathbf{b}$ is an $m \times 1$ vector.

The set $\{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} \leq \mathbf{b}\}$ of all $\mathbf{x}$ which satisfy the constraints is called the **feasible region**. In our example, the feasible region is shown below (it extends infinitely far to the left):



A point $\mathbf{x}$ in the feasible region is called a **feasible solution**. You should think of it as follows: a point (x, y) in this region is a feasible decision you could make (even if it loses your company a lot of money), whereas a point (x, y) outside this region is just not an option you could choose.

1.2 The objective function

The linear program also has an **objective function**: we want to maximize $1000x - 300y$. In general, we might be maximizing or minimizing an arbitrary linear function.

What does an "arbitrary linear function" look like? Well, we can write $1000x - 300y$ as a product of a row vector and a column vector:

$$1000x - 300y = \begin{bmatrix} 1000 & -300 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

(Technically, one of these is a scalar and one of these is a 1×1 matrix, but we will often ignore the difference.)

More generally, when we have a vector of variables $\mathbf{x} \in \mathbb{R}^n$, we can write the objective function as $\mathbf{c}^\mathsf{T}\mathbf{x}$ for some constant vector $\mathbf{c} \in \mathbb{R}^n$.

Putting together these ideas, any linear program can be written as

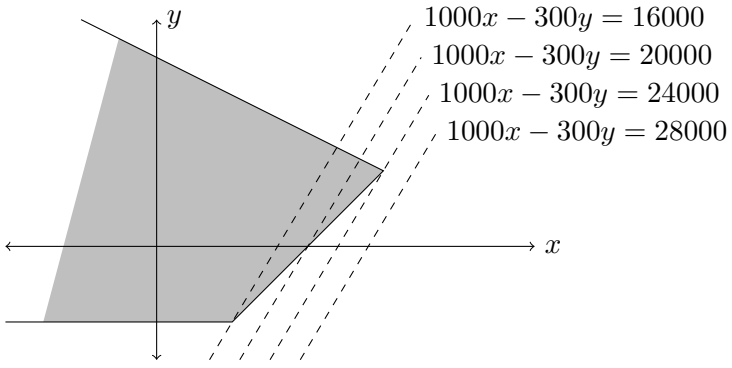
$$\begin{aligned} \underset{\mathbf{x} \in \mathbb{R}^n}{\text{maximize}} \quad & \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b}. \end{aligned}$$

What about minimizing? Well, minimizing $\mathbf{c}^\mathsf{T}\mathbf{x}$ would be the same as maximizing its negative $(-\mathbf{c})^\mathsf{T}\mathbf{x}$. We will encounter both kinds of linear programs in class, but we don't lose any generality by focusing on one kind whenever it's convenient.

Whether we're minimizing or maximizing, a point $\mathbf{x} \in \mathbb{R}^n$ with the best value of the objective function is called an **optimal solution**. In our example, the point $(x, y) = (30, 20)$ is the unique optimal solution, as we'll see in a moment.

2 The naive approach to solving linear programs

Let's go back to the original linear program. Here are some possible values of the objective function $1000x - 300y$, and the places where they occur:



Pick a small value of $1000x - 300y$ (such as 16000) and the feasible points with that value of $x - y$ are a line segment. Pick a large value of $1000x - 300y$ (such as 28000) and there are no feasible points with that value of $1000x - 300y$. But when $1000x - 300y = 2400$, just before the value becomes impossible, the segment shrinks to a single point: a corner of the feasible region.

Without drawing this picture and lots of carefully measured parallel lines, all we know is that this happens a *some* corner. Where are the corners? Well, at each corner, two of our boundary lines intersect. So we can try taking our boundaries two at a time, and seeing where they intersect:

- $x + y = 50$ and $2x - y = 40$ when $(x, y) = (30, 20)$. This is one of our corners: the top one.
- $2x - y = 40$ and $y = -20$ when $(x, y) = (10, -20)$. This is the lower of the two corners.
- $x+y = 50$ and $y = -20$ when $(x, y) = (70, -20)$. This is not actually a corner: two boundaries intersect here, but the inequality $2x - y \leq 40$ does not hold.

Now we can compare the values of $1000x-300y$ at $(30, 20)$ and $(10, -20)$. The first corner turns out to be better than the second, so that's our optimal solution. (There's actually one more important thing to check, which we'll get to in a bit, but in this case it doesn't affect the answer.)

This is the "naive" approach to solving linear programs. It's quick to explain, and for small examples, especially ones you can draw in the plane, it may be the easiest thing to do.

Imagine, however, that you have a linear program with 50 variables and 100 inequalities. (This is a "tiny" linear program: my computer solves one of these in approximately 0.03 seconds.) With the naive approach, there are $\binom{100}{50} = 100\,891\,344\,545\,564\,193\,334\,812\,497\,256$ combinations of 50 of the equations bounding the region. Each of these combinations (in general) intersects at a single point, so we need to compare that many points to find the best one.

Our goal in this class is going to be to try to do less work than this. Ahead of us is the simplex method, which starts at one vertex of a linear program, and moves from vertex to vertex until it finds the best one: hopefully long before it visits all the vertices. We won't solve problems with 100 inequalities, but computers solve such problems in a very similar way.

2.1 Misbehaving linear programs

In the previous example, our linear program had a unique optimal solution, but this is not always guaranteed to be true. What can go wrong?

1. **The optimal solution might not be unique.** Imagine that the lines we are drawing are parallel to a boundary of the region. Then several corners are equally good! This is a minor problem, as these things go, but it will complicate our life a little.
2. **The optimal solution might not exist, because the objective function can be arbitrarily large.** Imagine that we are minimizing $1000x - 300y$, not maximizing. Then we can keep moving the dashed line farther and farther left, and getting lower and lower values.

   This means one of two things: either we found a hack to get infinite profits, or (more likely) there is another constraint we didn't model because it's "obvious": to us, but not to the linear program. Maybe we can't actually hire 1000 yodelists and fire 1000 xylophonists, because we don't have that many xylophonists to fire!

Either way, even the naive approach needs to worry about this: we should check that in the direction that our region extends forever, the solutions keep getting worse and not better. We won't go into detail about how to check this, because we won't be using the naive approach.

3. **The optimal solution might not exist, because there are no feasible solutions.** Imagine that the constraints we have contradict each other: there is no way to satisfy all of them. This is a time to rethink our model and see if we can relax some constraints. (Maybe union negotiations have actually forced us to acquire more office space before they can be satisfied, for example.)