Math 482: Linear Programming[1] Mikhail Lavrov

Lecture 33: Integer Programming

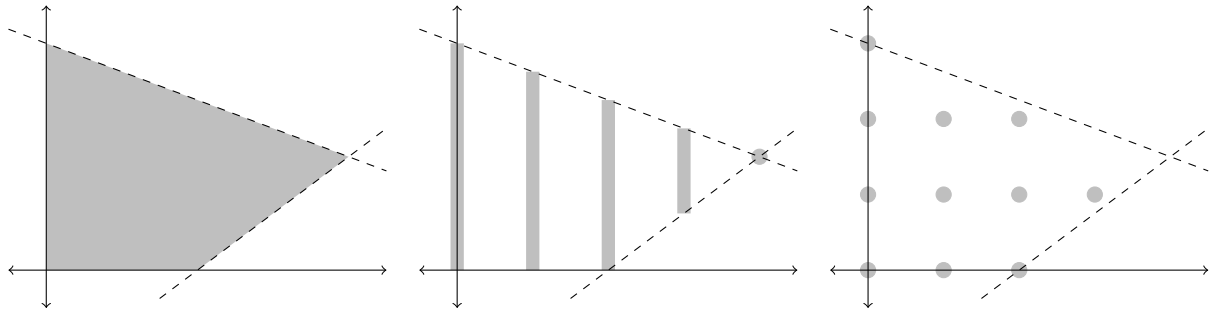
December 2, 2019 *University of Illinois at Urbana-Champaign*

1 Integer linear programming

An integer linear program (often just called an "integer program") is your usual linear program, together with a constraint on some (or all) variables that they must have integer solutions.

We saw these appear earlier in the class when looking at graph theory problems like the bipartite matching problem, but in all of the problems we looked at, we had total unimodularity to save us: we didn't have to think about the integer constraints, because they would hold automatically. This is rare and unusual: typically, the constraints matter.

For example, consider the following three optimization problems:



$$\begin{aligned} \underset{x,y\in\mathbb{R}}{\text{maximize}} \quad & x + y \\ \text{subject to} \quad & 3x + 8y \leq 24 \\ & 3x - 4y \leq 6 \\ & x, y \geq 0 \end{aligned} \qquad \begin{aligned} \underset{x\in\mathbb{Z},y\in\mathbb{R}}{\text{maximize}} \quad & x + y \\ \text{subject to} \quad & 3x + 8y \leq 24 \\ & 3x - 4y \leq 6 \\ & x, y \geq 0 \end{aligned} \qquad \begin{aligned} \underset{x,y\in\mathbb{Z}}{\text{maximize}} \quad & x + y \\ \text{subject to} \quad & 3x + 8y \leq 24 \\ & 3x - 4y \leq 6 \\ & x, y \geq 0 \end{aligned}$$

The first example is an ordinary linear program with optimal solution $(4, \frac{3}{2})$.

The second example is a (mixed) integer program where $(4, \frac{3}{2})$ is still the optimal solution. In fact, here, all vertices of the feasible region have $x \in \mathbb{Z}$; if we know this ahead of time, we can solve the integer program as a linear program.

The last example is an integer program with the same constraints, but the optimal solutions are $(2, 2)$ and $(3, 1)$ instead. Note that we can't even solve the integer program by rounding $(4, \frac{3}{2})$ to the nearest integer; that won't give us a feasible solution.

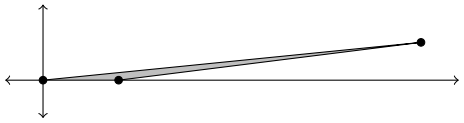
In general, an optimal integer solution can be arbitrarily far from the optimal solution. For example,

[1] This document comes from the Math 482 course webpage: https://faculty.math.illinois.edu/~mlavrov/courses/482-fall-2019.html

consider the region

$$\left\{(x,y) : \frac{x-1}{998} \leq y \leq \frac{x}{1000}, x, y \geq 0\right\}.$$

This region has a vertex at $(x, y) = (500, \frac{1}{2})$, but its only integer points are at $(0, 0)$ and $(1, 0)$. (A version with $998, 1000$ replaced by $8, 10$ is shown below.)



This is just to illustrate that integer programming is hard, and weird things can happen when we add the integer constraint.

2 The power of integer programming

Here are some of the things we can do with integer programming that we couldn't have done before.

2.1 Fixed costs

In an economic setting, in addition to scaling costs per unit, there are often fixed costs that apply only once. Maybe if your business might need paperclips, you have to pay \$1000 to arrange for paperclip delivery, but then each box of paperclips only costs \$1. But if you don't use paperclips at all, you can skip the fixed cost. This behavior cannot be modeled by a linear program.

With integer programs, if $x_1 \geq 0$ is the number of paperclips, we could model this fixed cost with an additional integer variable x_2, which also satisfies $0 \leq x_2 \leq 1$ (so either $x_2 = 0$ or $x_2 = 1$). Adding the constraint

$$x_1 \leq M x_2$$

for some very very large M means that if $x_1 = 0$, x_2 can also be 0; if $x_1 > 0$, x_2 cannot be 0, so it must be 1. Then,

$$x_1 + 1000 x_2$$

is the total cost of x_1 paperclips.

(The large number M will also be an upper bound on the number of paperclips we can produce, so it needs to be large enough to avoid imposing any false constraints. However, making M too large will later make it harder for us to solve the integer program, so we should make M as small as possible if we can.)

2.2 Unions of regions

We can use integer programming to optimize over a disjoint union of several regions: this is, in general, not convex, so linear constraints can't describe such a disjoint union.

Here, we'll consider bounded regions: suppose that we have the two regions

$$\begin{cases} A\mathbf{x} \leq \mathbf{b} \\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{m} \end{cases} \quad \text{and} \quad \begin{cases} C\mathbf{x} \leq \mathbf{d} \\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{n} \end{cases}$$

Here, $A, \mathbf{b}, C, \mathbf{d}, \mathbf{m}, \mathbf{n}$ are constant matrices and vectors, only $\mathbf{x}$ is a variable vector. The variables $\mathbf{x}$ are still real.

We model the union of these two regions as follows. We write $\mathbf{x} = \mathbf{x}^{(1)} + \mathbf{x}^{(2)}$ with the idea that $\mathbf{x}^{(1)}$ will be equal to $\mathbf{x}$ in the first region, and $\mathbf{x}^{(2)}$ will be equal to $\mathbf{x}$ in the second region. We also add a single integer variable $t \in \mathbb{Z}$, with $0 \leq t \leq 1$. Then, we write the constraints:

$$\begin{cases} A\mathbf{x}^{(1)} \leq t\mathbf{b} \\ \mathbf{0} \leq \mathbf{x}^{(1)} \leq t\mathbf{m} \\ C\mathbf{x}^{(2)} \leq (1-t)\mathbf{d} \\ \mathbf{0} \leq \mathbf{x}^{(2)} \leq (1-t)\mathbf{n} \end{cases}$$

The variable t can be either 0 or 1. Suppose that $t = 1$. Then, the first two constraints are just $A\mathbf{x}^{(1)} \leq \mathbf{b}$ and $0 \leq \mathbf{x}^{(1)} \leq \mathbf{m}$, the constraints on the first region. Then, the next constraints simplify to $C\mathbf{x}^{(2)} \leq \mathbf{0}$ and $\mathbf{0} \leq \mathbf{x}^{(2)} \leq \mathbf{0}$, so $\mathbf{x}^{(2)}$ is forced to be $\mathbf{0}$. We conclude that $\mathbf{x} = \mathbf{x}^{(1)}$ is in the first region.

When $t = 0$, the roles are swapped: $\mathbf{x}^{(1)}$ is forced to be 0 and $\mathbf{x} = \mathbf{x}^{(2)}$ is in the second region.

An interesting thing happens with this problem, however. Although linear programs can't model the union of two regions like this, they can still optimize linear functions over such a union. If we forget the domain constraint $t \in \mathbb{Z}$, we get the convex hull of the union of the two regions, and this convex hull will not introduce any new extreme points. So the optimal solution to an optimization problem can be found by treating this system as a linear program. This doesn't always happen—so treasure it when it does!

2.3 Logical constraints

More generally, one of the reasons that integer programming is so powerful is that it can be used to encode arbitrary logical constraints.

Here, we work with variables in the set $\{0, 1\}$ (integer variables x_i with the constraint $0 \leq x_i \leq 1$). You may have noticed a pattern by now: this is a very common way to encounter integer variables. We interpret 1 as meaning "true", and 0 as meaning "false".

We already had an implicit "AND" operation: if we write down several inequalities, all of them have to be true.

Now we can also express an "OR" of many logical variables: if we write down the inequality

$$x_1 + x_2 + \cdots + x_k \geq 1$$

then at least one of $x_1, x_2, \ldots, x_k$ has to be 1 (or true), meaning that this inequality is equivalent to the logical condition "x_1 OR x_2 OR ... OR x_k".

Finally, if we replace x_i by $1 - x_i$, that gives the logical NOT of x_i: it turns 1 to 0 and 0 to 1.

It turns out that any logical expression can be put into "conjuctive normal form": an AND of many OR statements, each one of which includes only a variable or its negation. Using the ideas

above, any such logical expression can be written down as constraints on a linear program. For example,

$$(x_1 \text{ OR NOT } x_2) \text{ AND } (x_2 \text{ OR NOT } x_3 \text{ OR } x_4)$$

can be encoded as the system

$$\begin{cases} x_1 + (1 - x_2) \geq 1 \\ x_2 + (1 - x_3) + x_4 \geq 1 \\ 0 \leq x_1, x_2, x_3, x_4 \leq 1 \\ x_1, x_2, x_3, x_4 \in \mathbb{Z} \end{cases}$$

If you haven't taken a CS class, take my word for it: if you can solve problems with logical expressions, you can do anything.

Statements such as "some two large prime numbers multiply together to this constant" form the basis of modern cryptography. Statements such as "some thousand lines of mathematical argument prove this theorem" form the basis of my job. They can all be expressed this way, so they're all (very complicated) integer programs.

What this means in practice is that we don't know of a way to solve integer programs quickly, or the world would look very different.

But we can do our best.