LECTURE 18: INTEGER PROGRAMMING (I)

Today: Gentle introduction to Integer Programming

1. Motivation

Problem: Often with LP problems, you get non-integer solutions, like $x_1 = \frac{3}{2}$ and $x_2 = \frac{7}{4}$ which might not make sense in context of a problem.

Example: If x_i are number of people, it wouldn't really make sense to have $\frac{3}{2}$ of a person

Example: If x_i are flights, it wouldn't make sense to take $\frac{2}{3}$ of a flight

Example: Some commodities can't be divided: Remember the original motivation for Operations Research was in military operations. There x_1 might represent the number of submarines, and it again wouldn't make sense to produce $\frac{7}{4}$ submarines.

Example: Towards the beginning of the course, we studied an example of assigning classes to students, where $x_{ij} =$ Put student i in class j or not, then $x_{ij} = 0$ or 1, which requires our solutions to be integers

This leads to a special sub-class of LP called **Integer Programming:**

2. Integer Programming

It's just like LP, except we require our variables to be integers:

Date: Thursday, November 10, 2022.

Integer Programming:

$$\begin{aligned} \max c^T x \\ \text{subject to } Ax \leq b \\ x \geq 0 \\ x \in \mathbb{N}^n \end{aligned}$$

Note: The same definition also works for non-linear programs

Note: Sometimes you have **mixed LP** where some variables are integers, but others are not

Example:

$$\begin{aligned} x_1 &= \text{number of cars (integers)} \\ x_2 &= \text{amount of fuel being used (real)} \end{aligned}$$

There's a special sub-class of integer programming called **binary programming** which we'll discuss now:

3. Binary Programming

Binary Variables:

$$\text{If } \; x_i = \begin{cases} 0 & \text{(no)} \\ 1 & \text{(yes)} \end{cases}$$

Then x_i is a **binary variable**

Binary Programming:

$$\begin{aligned} &\max c^T x \\ \text{subject to } &Ax \le b \\ &x \in \{0,1\}^n \end{aligned}$$

Usually good when making decisions

Example: Assigning people to jobs, so if you have n people, then

$$x_i = \begin{cases} 0 & \text{if person } i \text{ doesn't get the job} \\ 1 & \text{if person } i \text{ gets the job} \end{cases}$$

We can play around with this to solve more interesting situations

Example: $x_2 + x_3 + x_5 = 1$ means that exactly one of people $2, 3, 5$ get the job. This is because if no one gets the job, then you get $0+0+0 \neq 1$, and if, say, two of those people get the job, then you get $1 + 1 + 0 \neq 1$

Example: $x_2 + x_3 + x_5 \geq 1$ means that at least one person gets the job

Example: Multiple Choice

Suppose you have 10 questions, with 3 choices $1, 2, 3$ each, then

$$x_{ij} = \begin{cases} 1 & \text{if for question } i \text{ you pick } j \\ 0 & \text{if not} \end{cases}$$

If for Question 5 you choose 3 but not 1 or 2, then $x_{51} = 0, x_{52} = 0, x_{53} = 1$

Then "exactly one answer is correct" would mean $x_{i1} + x_{i2} + x_{i3} = 1$ (for $i = 1, \cdots, 10$)

Whereas "at least one is correct" would mean $x_{i1} + x_{i2} + x_{i3} \geq 1$

Example: Event Conditions

Suppose you have two binary variables x_1 and x_2 where

$$x_1 = \begin{cases} 1 & \text{if there's an earthquake} \\ 0 & \text{if not} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if there's an aftershock} \\ 0 & \text{if not} \end{cases}$$

This is called an **event condition**, x_2 happens only if x_1 happens, $x_2 \Rightarrow x_1$

Then we have the following table of possibilities:

x_1 \ x_2	1	0
1	✓	✓
0	X	✓

Basically everything can happen except for $x_1 = 0$ and $x_2 = 1$ (how can there be an aftershock without an earthquake?)

Can rewrite this as $x_2 \leq x_1$ (because can't have $1 \leq 0$)

So here $x_2 \Rightarrow x_1$ means $x_2 \leq x_1$

More Generally:

If $x_n \Rightarrow x_m$ then $x_n \leq x_m$

Application: Another application of this is prerequisites: If Course 1 is a prequisite to Course 2, then again $x_2 \leq x_1$ meaning that you can't take Course 2 without Course 1, but you could take both of them at the same time.

Note: If you want to say "You have to take Course 1 and then 2, in that order" (but you could take none of them), then you would say $x_2 \leq x_1$ and $x_1 + x_2 \leq 1$

4. Solving Integer Programming Problems

In the next couple of lectures, we learn some algorithms for solving Integer Programming problems, but they all have one main issue:

Problem: They're all super slow!

There are some ways to get around this:

Solution 1: Instead of exact solutions, find approximate solutions

Example: A possible (albeit silly) way is to solve the same LP problem, but without the requirement that $x \in \mathbb{N}^n$. This gives you for example $x_1 = 2.3$ and $x_2 = 4.6$, and just round them to $x_1 = 2$ and $x_2 = 5$

Approximate solutions make sense in practice, I mean who really cares if your optimal x_1 is 9000 or 9001? ☺

Of course we will need a way to measure a way of saying how good your approximation is, which is called giving a **bound** on your solution.

It is also possible that your approximate solution falls outside your feasible region, which might sometimes be ok, and sometimes not.

Solution 2: Rounding Techniques

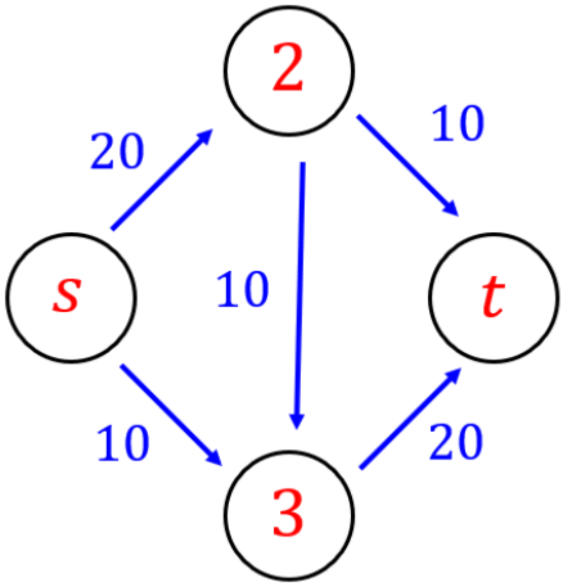
Suppose your LP requires that $x_i \in \{0, 1\}$

Instead, solve your LP but with $x_i \in [0, 1]$ instead, that is $x_i \geq 0$ and $x_i \leq 1$. This gives you (possibly) non-integer values x_i, say $x_i = 0.6$ and then just round them up to the nearest integer, so $x_i = 1$

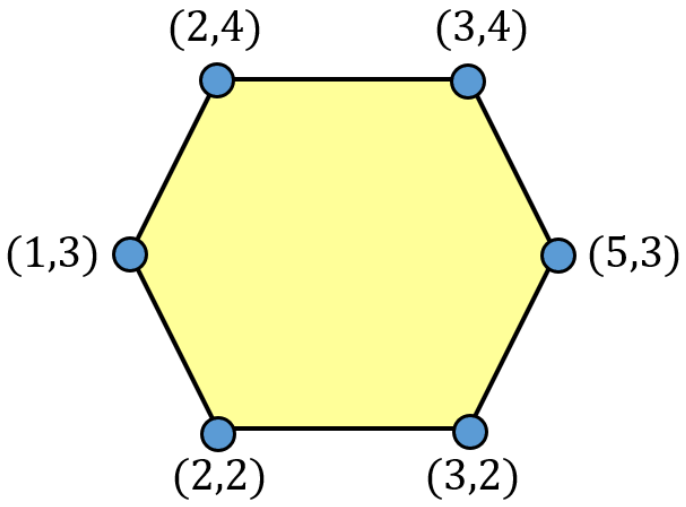
Solution 3: Luck! Of course it could happen that by chance, the solutions to your LP are already integers, in which case you don't have to do anything

Note: There are actually two instances where solving LP automatically gives you integer solutions

One is for **network problems/max flow problems**, provided all the constraints are integers.



Another is if (by luck) all the corners of your feasible region have integer coordinates, like in the following picture



5. Airline Example

Here's a neat application of binary programming.

Suppose you're the CEO of PeyAmerican Airlines and you're offering 12 possible sequences of flights from San Francisco (SFO) back to SFO, given by the following table

Flight \ Sequence	1	2	3	4	5	6	7	8	9	10	11	12
1. SFO → LAX	1			1			1			1		
2. SFO → DEN		1			1			1			1	
3. SFO → SEA			1			1			1			1
4. LAX → ORD				2			2		3	2		3
5. LAX → SFO	2					3				5	5	
6. ORD → DEN				3	3				4			
7. ORD → SEA							3	3		3	3	4
8. DEN → SFO		2		4	4				5			
9. DEN → ORD					2			2			2	
10. SEA → SFO			2				4	4				5
11. SEA → LAX						2			2	4	4	2
Cost ($1000's)	2	3	4	6	7	5	7	8	9	9	8	9

The nonzero numbers in this table indicate the order of the flights

How to read this table:

Let's look at Column 9 for example: This says Sequence # 9 first goes from SFO to SEA (1 is in row 3) then from SEA to LAX (2 is in row 11) then from LAX to ORD (3 is in row 4) then from ORD to DEN (4 is in row 6) then from DEN to SFO (5 is in row 5).

So Flight Sequence # 9 is

$$\text{SFO} \to \text{SEA} \to \text{LAX} \to \text{ORD} \to \text{DEN} \to \text{SFO}$$

And the total cost (see below) is \$9000 (last row)

Our Goal: Assign 3 crews based in SFO to the sequences in such a way that every flight (like SFO $\to$ LAX or ORD $\to$ DEN) has a crew. The number in the last row is the cost of assigning a crew to sequence.

Our variables: The trick here is to assign sequences to crews (instead of crews to sequences), so

$$x_j = \begin{cases} 1 & \text{if sequence } j \text{ is assigned to a crew} \\ 0 & \text{otherwise} \end{cases}$$

Objective Function:

The quantities are given by x_j (0 or 1) and the costs are given by the last row, so

$$z = 2x_1+3x_2+4x_3+6x_4+7x_5+5x_6+7x_7+8x_8+9x_9+9x_{10}+8x_{11}+9x_{12}$$

Constraints:

We need to assign 3 crews, so

$$x_1 + x_2 + \cdots + x_{12} = 3$$

Moreover, each flight needs to have at least one crew.

For example, let's look at SFO $\to$ LAX, which needs ≥ 1 crew. The sequences that use SFO $\to$ LAX are sequences # $1, 4, 7, 10$, and so

$$x_1 + x_4 + x_7 + x_{10} \geq 1$$

Same thing for the other flights as well.

Binary LP Problem

$$\begin{aligned}
\min z = {} & 2x_1 + 3x_2 + 4x_3 + 6x_4 + 7x_5 + 5x_6 \\
& + 7x_7 + 8x_8 + 9x_9 + 9x_{10} + 8x_{11} + 9x_{12} \\
\text{subject to } & x_1 + \cdots + x_{12} = 3 \\
& x_1 + x_4 + x_7 + x_{10} \geq 1 \\
& x_2 + x_5 + x_8 + x_{11} \geq 1 \\
& x_3 + x_6 + x_9 + x_{12} \geq 1 \\
& x_4 + x_7 + x_9 + x_{10} + x_{12} \geq 1 \\
& x_1 + x_6 + x_{10} + x_{11} \geq 1 \\
& x_4 + x_5 + x_9 \geq 1 \\
& x_7 + x_8 + x_{10} + x_{11} + x_{12} \geq 1 \\
& x_2 + x_4 + x_5 + x_9 \geq 1 \\
& x_5 + x_8 + x_{11} \geq 1 \\
& x_3 + x_7 + x_8 + x_{12} \geq 1 \\
& x_6 + x_9 + x_{10} + x_{11} + x_{12} \geq 1 \\
& x_j \in \{0, 1\}
\end{aligned}$$

Solution: It turns out that an optimal solution is $x_3 = 1, x_4 = 1, x_{11} = 1$ and all other $x_j = 0$. That is, assign a crew to sequences # $3, 4, 11$. In that case the cost is $4 + 6 + 8 = 18$ thousand dollars.

Another optimal solution would be $x_1 = 1, x_5 = 1, x_{12} = 1$

Notice it's not very obvious how we got those optimal solutions!