Lesson 8: Logical If/Then and Either/Or Constraints

1   A Binary IP

USNA is considering purchasing some new generators to ensure power in the case of storms. The table below gives the fixed cost and MW of power generated by each potential generator.

	Fixed cost ($ million)	Power Generated (MW)
Generator 1	1.20	3,000
Generator 2	0.75	4,000
Generator 3	0.50	5,000
Generator 4	1.00	3,500
Generator 5	1.10	3,800

USNA wants to ensure that at least 8500 MW of power are available if there is an outage.

1. Formulate a concrete IP that would tell USNA the optimal generators to purchase while satisfying power demands at minimum cost.

Variables

→ Which generators does USNA purchase

let $x_i = 1$ if USNA purchases generator i for $i = \{1, 2, 3, 4, 5\}$ and 0 otherwise.

Objective

Minimize cost: $1.2 x_1 + 0.75 x_2 + 0.5 x_3 + 1 x_4 + 1.1 x_5$

Constraints

$$x_1, x_2, x_3, x_4, x_5 \in \{0, 1\}$$

$$3000 x_1 + 4000 x_2 + 5000 x_3 + 3500 x_4 + 3800 x_5 \geq 8500$$

2   Types of Binary Variable Constraints

In general, there are several types of binary constraints we'd like to enforce:

- Fixed Charge (Lesson 6) $\rightarrow$ *strong and weak forcing*

- Mutually exclusive and multiple choice constraints

- If/then constraints ⎤
- Either/or constraints ⎦ *Hard / tricky constraints*

3   Mutually Exclusive and Multiple Choice Constraints

The two simplest type of logical constraints deal with mutual exclusion or multiple choice.

- **Mutual Exclusive:** *, You can't choose every variable* You are only allowed to choose a specific subset of variables to be 1.

   ○ Suppose in the generator problem above, you are only allowed to choose at most 1 of generators 2, 3, and 4. Write a logical constraint that enforces this.

Choose at most 1 of generators 2, 3, and 4

$$X_2 + X_3 + X_4 \quad \rightarrow \text{How many of the 3 are chosen}$$

$$X_2 + X_3 + X_4 \le 1 \quad \rightarrow \text{Allows USNA to choose 0 or 1 of the three.}$$

- **Multiple Choice:** You must choose a certain subset of variables to be equal to 1.

   ○ Suppose that USNA knows that facilities will only approve the purchase of exactly 2 of generators 1, 2, 3, 4, and 5. Write a logical constraint that enforces this.

Must choose exactly 2 of gen 1, 2, 3, 4, 5

$$X_1 + X_2 + X_3 + X_4 + X_5 \quad \rightarrow \text{Total number chosen}$$

$$X_1 + X_2 + X_3 + X_4 + X_5 = 2 \quad \rightarrow \text{Choose exactly 2 of the 5}$$

4 Introducing If/Then Constraints on Two Variables

Often, it is the case that we want to enforce if/then logic on binary variables. For example, perhaps Generator 1 and 3 are made by competing companies, so if Generator 1 is purchased, we can not purchase Generator 3.

x_1, x_3 binary   if $x_1 = 1$ then $x_3 = 0$

2. Try to write a constraint to capture this logic. That is write an inequality which says if $x_1 = 1$ then $x_3 = 0$.

$x_1 + x_3 = 1$

$x_1 = 1 \Rightarrow x_3 = 0$ ✓

$x_1 = 0 \Rightarrow x_3 = 1$ ✗

Satisfies $x_1 = 1 \to x_3 = 0$

but if $x_1 = 0$
x_3 should be free

$x_1 \cdot x_3 = 0$

$x_1 = 1 \Rightarrow x_3 = 0$ ✓

$x_1 = 0 \Rightarrow x_3$ free ✓

works mathematically
but is nonlinear

$x_1 + x_3 \leq 1$

$x_1 = 1 \Rightarrow x_3 = 0$

$x_1 = 0 \Rightarrow x_3 \leq 1 \Rightarrow$ free

Works

$x_1 \leq 1 - x_3$

↗
works

Next class
why does
this work

It can seem like a lot of guess and check in order to get a constraint that actually works. Fortunately, there's a process you can follow to make this significantly easier. Namely, the following steps can make writing these logical constraints much easier.

For if/then Constraints on binary Variables. If $x_1 = 1$ then $x_3 = 0$

- Write the constraint as a conditional statement and convert all clauses to 1s (why?)

$x_3 = 0$
↳ $1 - x_3 = 1$

If $x_1 = 1$ then $1 - x_3 = 1$

- Write the constraint from left to right.

$x_1 \leq 1 - x_3$

  ○ The left side of the **if** statement is the LHS of the constraint] everything before then
  ○ The inequality is always $\leq$] then $\Rightarrow \leq$
  ○ If it is converted to 1s, everything following **then** is the RHS of the constraint. That is the high value on the left "pushes" the RHS to its high value.

- **ALWAYS** check every logical constraint for **explicit** and **implicit** satisfaction.

  ○ **Explicit** satisfaction: Does it do what the constraint says?

  $x_1 = 1 \Rightarrow 1 \leq 1 - x_3 \Rightarrow x_3 = 0$ ✓

  ○ **Implicit** satisfaction: If the statement is not true is the RHS free

  $x_1 = 0 \Rightarrow 0 \leq 1 - x_3$ ↗ $x_3 = 0$ $0 \leq 1$ ✓ x_3 free
  ↘ $x_3 = 1 \Rightarrow 0 \leq 0$ ✓

3

Using the steps above, capture the following if/then constraint logic:

3. If we purchase generator 2 then we must purchase generator 4.

① If $x_2=1$, then $x_4=1$
    LHS   ≤   RHS

② $x_2 \leq x_4$

③ Explicit Check
   $x_2=1 \Rightarrow 1 \leq x_4 \Rightarrow x_4=1$ ✓

④ Implicit
   $x_2=0 \Rightarrow 0 \leq x_4$ ⟶ $x_4=0$ $0 \leq 0$
                               ⟶ $x_4=1$ $0 \leq 1$ ✓
   x_4 free

4. If we don't purchase generator 1 then we must purchase generator 5.

① If $x_1=0$ then $x_5=1$
   If $(1-x_1)=1$ then $x_5=1$
       LHS   ≤   RHS

② $\boxed{1-x_1 \leq x_5}$

③ $x_1=0$
   $1 \leq x_5 \Rightarrow x_5=1$ ✓

④ $x_1=1$
   $0 \leq x_5 \Rightarrow x_5$ free ✓

5. If we don't purchase generator 3 then we can't purchase generator 1. **and we must purchase gen 2**

① If $x_3=0$ then $x_1=0$ and $x_2=1$
   If $(1-x_3)=1$ then $1-x_1=1$ and $x_2=1$
       LHS   ≤   RHS

② $1-x_3 \leq (1-x_1) + x_2$
   $\boxed{2(1-x_3) \leq (1-x_1) + x_2}$

③ $x_3=0$
   want → $1 \leq (1-x_1) + x_2$
   2
   Doesn't work
   $2 \leq (1-x_1) + x_2$ ⟶ $x_1=0$ ✓
                          ⟶ $x_2=1$

④ $x_3=1$
   $0 \leq (1-x_1) + x_2$ ⟶ x_1 free ⟶ x_3 free ✓

The same idea will also work with 3 or more variables, just the LHS or RHS of the constraints may be off by a constant factor. That's why it's important to always check explicit and implicity satisfaction.

5   Either/Or Constraints: Motivating Example

The remainder of this lesson will focus on either/or constraints.

Quality Cabinets used an integer-program to determine how many of each type of cabinet to make in order to maximize profit. They used decision variables x_s, x_d, and x_e to represent the number of standard, deluxe, and enhanced cabinets, respectively, to produce each week. Each cabinet requires a certain number of hours of painting time and there is a limit on the number of painting hours available. A small part of the model is:

> Maximize $\quad 25x_a + 45x_d + 60x_e$
>
> subject to $\quad 2x_a + 4x_d + 5x_e \leq 700$ $\qquad\qquad$ (painting time)

$x_a \to$ Standard $\qquad\qquad$ $\underline{2x_a} + 4x_d + 5x_e \leq 700$

$x_d \to$ deluxe

$x_e \to$ enhanced $\qquad\qquad$ Time to paint each Cabinet

6   Model Update Requested

Now Quality Cabinets is considering renting better painting equipment and has asked us to update our model to help with this decision. The equipment will cost $300 per week, but will reduce the time required to to do the painting by 15 minutes for a standard cabinet, by 30 minutes for a deluxe cabinet, and by 1 hour for an enhanced cabinet.

We decide to add a binary variable z. In the solution, if $z = 1$, then Quality Cabinets should rent the equipment. If $z = 0$, they should not.

    6. Should the objective function change? If not, explain. If so, write the updated objective function.

Fixed cost of $300

Max $\quad 25x_a + 45x_d + 60x_e - 300z$

    7. If the equipment is not rented, what should the painting constraint be? Label it (A).

If not rented

$z=0 \Rightarrow$ Dont

improve painting

time and

Constraint stays

same

(A) $\quad 2x_a + 4x_d + 5x_e \leq 700$

8. If the equipment is rented, what should the painting constraint be? Label it (B).

If rented

$z = 1$ and reduce time

   x_a reduced 15 min
   x_d reduced 30 min
   x_e reduced 1 hr

(B) $1.75 x_a + 3.5 x_d + 4 x_e \leq 700$

Goal: $z = 0$ get constraint (A)
      $z = 1$ get constraint (B)

7 Logical (Either/Or) Constraints

There is no place for "if – then" statements in a math programming model, so we have to enforce this logic indirectly using linear constraints and binary variables.

> A constraint is *relaxed* if it has no impact. We can relax a "$\leq$" constraint by making the value on the right so large that it will never restrict the values of the variables on the left.

9. Rewrite constraint (A) above *using z* so that it is enforced if $z = 0$ (the equipment is not rented), but *relaxed* if $z = 1$ (the equipment is rented).

(2)

(1)

(3)
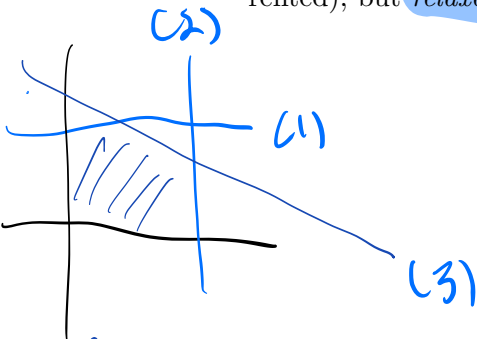
(A): $2 x_a + 4 x_d + 5 x_e \leq 700 + M z$

↗ Big number

$z = 0$: ≤ 700 enforced

$z = 1$: $\leq 700 + M$ relaxed

Relaxing a constraint means you push it out of the feasible region.

10. Rewrite constraint (B) above *using z* so that it is enforced if $z = 1$ (the equipment is rented), but *relaxed* if $z = 0$ (the equipment is not rented).

(B): $1.75 x_a + 3.5 x_d + 4 x_e \leq 700 + M(1-z)$

$z = 0$: $\leq 700 + M$ Relaxed

$z = 1$: ≤ 700 Enforced

6

11. Write the updated version of the partial Quality Cabinets model here.

$$\text{Max} \quad 25x_a + 45x_d + 60x_e - 300z$$

$$2x_a + 4x_d + 5x_e \leq 700 + M \cdot z \qquad (A)$$

$$1.75x_a + 3.5x_d + 4x_e \leq 700 + M(1-z) \qquad (B)$$

$z = 1$ Buy machine pay \$300
  (A) is Relaxed
  (B) is enforced

$z = 0$ Don't buy
  (A) is enforced
  (B) is relaxed

8 Logical Constraints Summary

Suppose $a^T x \leq b$ is a linear constraint, M is a number that is bigger than $a^T x$ could ever be (but not TOO big! choose M wisely), and z is a binary variable.

A constraint that is enforces $a^T x \leq b$ if $z = 0$ and relaxes $a^T x \leq b$ if $z = 1$:

$$a^T x \leq b + M \cdot z$$

A constraint that is enforces $a^T x \leq b$ if $z = 1$ and relaxes $a^T x \leq b$ if $z = 0$:

$$a^T x \leq b + M(1-z)$$

9 Many more possibilities

There are many ways to creatively use linear constraints to enforce modeling requirements; this lesson contains only a few examples. Often this process takes some trial and error. **Always be sure to test the logic with various values of the decision variables to make sure the constraint is doing what you want it to do.**
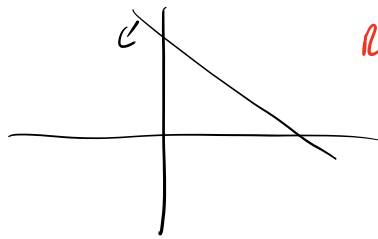
Relax by pushing constraint up
so add M
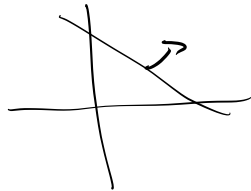
Relax by pushing constraint down
so subtract M