Linear Programming

TO MAXIMISE PROFITS, WE NEED TO MINIMISE THE COST OF PRODUCTION

- **Objective Function:** The quantity to be maximised / minimised
- **Linear Constraints:** Set of linear inequalities restricting the possible solutions.

- ***There may be a unique solution, infinitely many solutions or no solution***

$$
\begin{aligned}
\text{Maximise:} \quad & 2x + 3y \\
\text{Subject to:} \quad & 3x + 2y \leq 15 \\
& 2y - x \leq 5 \\
& x + 2y \leq 7 \\
& x, y \geq 0
\end{aligned}
$$

- **Example:** A company creates its smoothies using:

Banana, Strawberry and Mango

Each smoothie must contain at least 15% RDA of Vitamin A, at least 80% RDA of Vitamin C, and at least 300mg of potassium.

- 100g of Banana contains 1% of the RDA of Vitamin A, 14% of the RDA of Vitamin C, 358mg of potassium and costs 15p.
- 100g of Strawberry contains no Vitamin A, 97% of the RDA of Vitamin C, 153mg of potassium and costs 27p.
- 100g of Mango contains 21% of the RDA of Vitamin A, 60% of the RDA of Vitamin C, 168mg of potassium and costs 30p.

What proportion of ingredients would maximise profits?

- We are asked to maximise profit with the proportions given to us

Example (cont.)

Cost of Production

Vitamin A requirement

REQ

Vitamin C requirement

$$
\begin{aligned}
\text{Minimise:} \quad & 15x + 27y + 30z \\
\text{Subject to:} \quad & x + 21z \geq 15 \\
& 14x + 97y + 60z \geq 80 \\
& 358x + 153y + 168z \geq 300 \\
& x,\ y,\ z \geq 0
\end{aligned}
$$

Potassium requirement

Non-negativity constraint

Everywhere

TO FIND A PRICE OF A PRODUCT IN THIS CASE SMOOTHIE, WE HAVE:

TO FIND SOLUTION WE CAN STICK THE CONSTRAINTS INTO THE LP SOLVER AND IT WILL COME UP WITH OPTIMAL SOLUTION THAT SATISFIES THE MINIMISE CONSTRAINT.

WITH THE PRICE OF EACH SMOOTHIE OF 35.59p

Optimal Solution

$x = 0.365, \quad y = 0.341, \quad z = 0.697$

To solve **LPs**, it is helpful to write them in a **Standard Form**:

- This reduces the number of cases we must consider
- We can design algorithms that are highly specialised for a particular input format (**this is why we use CNF as the 'standard form' for SAT problems**)

LP Standard Form

STANDARD FORM FOR LINEAR PROGRAMS

- The criteria must be to **maximise** the objective function
  IT DOES NOT ACCEPT MINIMISE NEED TO CONVERT MINIMISE TO MAXIMISE.

- All **linear constraints** must be of the form *'less-than or-equal-to'* <

$$\boxed{a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq c}$$

(where $a_1, \ldots, a_n$ and c are constants)

- We seek a **non-negative solution** with the additional constraint – we don't want to be searching over the whole space negative and positive numbers so all must be non-negative.

$$\boxed{x_1, \ldots, x_n \geq 0}$$

Converting to STANDARD FORM

Step 1) Change the criteria for the objective function (IF REQUIRED)

So, you basically add a negative '– 'next to all coefficients

$$\boxed{\text{Minimise:} \quad a_1x + a_2y} \Rightarrow \boxed{\text{Maximise:} \quad -a_1x + -a_2y}$$

(minimising F is the same as maximising $-F$)

Step 2) Replace any 'equality constraints'

$$\boxed{\text{Subject to:} \quad b_1x + b_2y = c} \Rightarrow \boxed{\begin{array}{ll}\text{Subject to:} & b_1x + b_2y \leq c \\ & b_1x + b_2y \geq c\end{array}}$$

($A = c$ if and only if $A \leq c$ and $A \geq c$)

Step 3) Negate any *'greater-than-or-equal-to'* constraints

$$\boxed{\text{Subject to:} \quad b_1x + b_2y \geq c} \Rightarrow \boxed{\text{Subject to:} \quad -b_1x + -b_2y \leq -c}$$

($A \geq c$ if and only if $-A \leq -c$)

$$\begin{array}{lrcl}\text{Maximise:} & 2x + y & & \\ \text{Subject to:} & 2x + y & \leq & 2 \\ & x - y & \leq & 5 \\ & x, y & \geq & 0\end{array}$$

standard form ✓

$$\begin{array}{lrcl}\text{Minimise:} & 3y - 2x & & \\ \text{Subject to:} & x + y & = & 7 \\ & 2y - x & \geq & -4 \\ & x, y & \geq & 0\end{array}$$

not standard form ✗

But how do we convert to Standard Form?

$$\begin{array}{lrcl}\text{Maximise:} & 2x - 3y & & \\ \text{Subject to:} & x + y & \leq & 7 \\ & -x - y & \leq & -7 \\ & x - 2y & \leq & 4 \\ & x, y & \geq & 0\end{array}$$

Step 4) Ensure all variables are required to be non-negative **(this may require introducing additional variables)**

Case 4.1) If we have the constraint like $\boxed{x \leq 0}$

$$\boxed{\begin{array}{c}\textbf{Replace} \\ x := (-x')\end{array}} \qquad \boxed{\begin{array}{c}\textbf{Add constraint} \\ x' \geq 0\end{array}}$$

Case 4.2) If there is no constraint on a variable x at all

$$\boxed{\begin{array}{c}\textbf{Replace} \\ x := (x' - x'')\end{array}} \qquad \boxed{\begin{array}{c}\textbf{Add constraints} \\ x', x'' \geq 0\end{array}}$$

If we have a constraint that x must be a negative value, we won't throw it away, but we need to re-write/ replace it with new one. 4.1 have restriction you are flipping it, 4.2 you enforce some restriction,

So, case 4.2 must enforce some restriction that we wont search entire search-space and we don't have any constraints for X thus we create X' and X'' which will act like a bound, allowing to getting negative value when x' is lower than x'' or positive value when x'' is lower than x'. This restricts the x to be above negative search space but still allowing to have negative and positive values. In a way that it does not restrict our value X. We claim solution if we want to retrieve X back we just need to take X' – X'' = X

<u>LP Standard Form</u>

SLACK FORM

So, after getting all constraints into the same format we can introduce Slack Form, which introduces the variable which says how far off from that bound I am I. For example, we know that "x – 2y <= 4)" 4 is less by we don't know by how much, and that's why we will introduce SLACK VARIABLE into equation.

Step 1) Replace any '*less-than-or-equal*' constraints

Subject to: $b_1x + b_2y \le c$ $\Rightarrow$ Subject to: $b_1x + b_2y + s = c$

Step 3) Replace any 'greater-than-or-equal' constraints (IF WE DIDN'T GET RID OFF THEM YET)

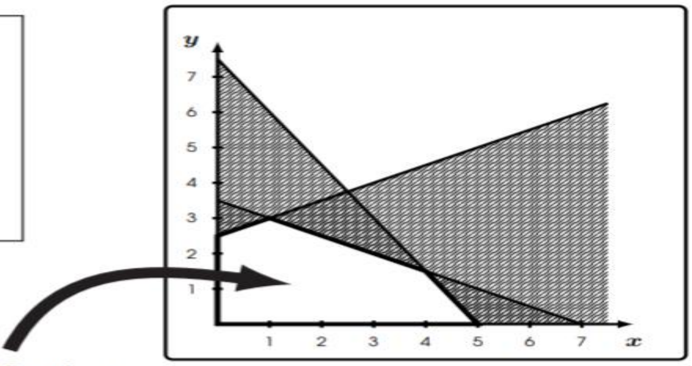
Subject to: $b_1x + b_2y \ge c$ $\Rightarrow$ Subject to: $b_1x + b_2y - s = c$

*Where **s** is a SLACK VARIABLE.*

Visualising a Solution

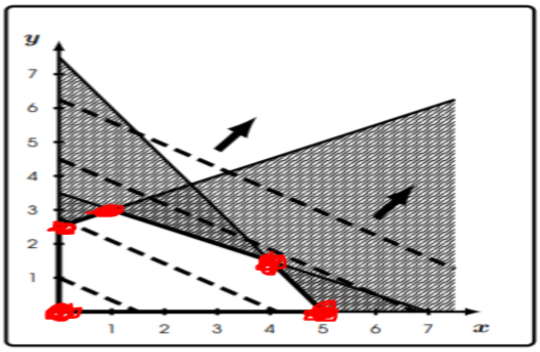
- What does a solution to an LP look like?

Maximise:	$2x + 3y$		
Subject to:	$3x + 2y$	$\le$	15
	$2y - x$	$\le$	5
	$x + 2y$	$\le$	7
	x, y	$\ge$	0



Feasible Region

- The **Feasible Region** is the set of all possible solutions satisfying the linear constraints.



- Since the objective function is **linear**, its maximum / minimum value must occur along the **boundary** of the feasible region.
- In fact, it is enough to examine only the **corners** of the feasible region.
- If the objective function is **parallel** to some contraint, there may be **infinitely many** solutions along some edge.

Evaluate each corner and see which one gives us the highest value when evaluating the objective function and pick up the best. In a bigger criterion there will be many intersections and that can be tricky.

(4, 3/2)

Week 8

<u>The Simplex Method</u>

Simplex Method is used where they are many boundaries to check.

The **Simplex Method** greedily explores the boundary of the feasible region to find an optimal solution.

The Simplex Method (concept)

- Start at the origin, with all variables set to zero.
- Select the variable that leads to the greatest increase in the objective function.
- Increase until you hit a constraint.
- Move along the constraint if doing so leads to an increase in the objective function.
- Repeat, moving around the boundary of the feasible region.

Introducing Tableaux

- A **Tableau** *(plural **Tableaux**)* is a matrix representation of a system of linear equations:

Linear Constraints

$$3x + 2y + s_1 = 15$$
$$-x + 2y + s_2 = 5$$
$$x + 2y + s_3 = 7$$

matrix of coefficients

$$\begin{array}{cccccccc} x & y & s_1 & s_2 & s_3 & C & & \\ \end{array}$$
$$\left[\begin{array}{cccccc:c} 3 & 2 & 1 & 0 & 0 & 0 & 15 \\ -1 & 2 & 0 & 1 & 0 & 0 & 5 \\ 1 & 2 & 0 & 0 & 1 & 0 & 7 \\ \hdashline -2 & -3 & 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Objective Function

$$C = 2x + 3y$$

rewriting as

$$-2x - 3y + C = 0$$

SIMPLEX METHOD

STEP 1) Construct the **initial tableau** from the slack form of the linear program. We are rewriting objective function by adding C constraint.

STEP 2) Identify the column with the **most negative** coefficient in the final row – FIND MOST NEGATIVE VALUE IN THE BOTTOM ROW IN TABLE.

STEP 3) Calculate the **row quotients** by dividing each of the entries in the final column by the entries in the pivot column. BASICALLY, DIVIDE ALL ENTRIES IN THE RIGHTMOST COLUMN BY ALL ENTRIES in the PIVOT COLUMN.

STEP 4) The row with the **smallest positive** row quotient is the first constraint that is violated when increasing the pivot variable. SELECT SMALLES ONE, THIS ONE IS A PIVOT ROW IT CROSSES WITH PIVOT ROW.

STEP 5) Apply the **row transformation** that

- Leaves a 1 in the **pivot row** of the pivot column
- Leaves a 0 in all other rows of the pivot column INCLUDING THE FINAL ROW.
- WE LOOK FOR OPERATION THAT CAN BE PERFORMED TO BRING THE VALUE TO EITHER **1** or **2** and THEN WE PERFORM THE SAME OPERATION ON ENTIRE ROW, THEN GOING TO NEXT ROW

Week 8

Simplex Method Example

Step 1) Construct the **initial tableau** from the slack form of the linear program.

$$\begin{array}{c} \begin{array}{ccccccc} x & y & s_1 & s_2 & s_3 & C & \end{array} \\ \left[\begin{array}{cccccc|c} 3 & 2 & 1 & 0 & 0 & 0 & 15 \\ -1 & 2 & 0 & 1 & 0 & 0 & 5 \\ 1 & 2 & 0 & 0 & 1 & 0 & 7 \\ \hline -2 & -3 & 0 & 0 & 0 & 1 & 0 \end{array}\right] \end{array}$$

'Pivot' column

Step 2) Identify the column with the **most negative** coefficient in the final row.

Step 3) Calculate the **row quotients** by dividing each of the entries in the final column by the entries in the pivot column

$$\begin{array}{c} \begin{array}{ccccccc} x & y & s_1 & s_2 & s_3 & C & \end{array} \\ \left[\begin{array}{cccccc|c} 3 & 2 & 1 & 0 & 0 & 0 & 15 \\ -1 & 2 & 0 & 1 & 0 & 0 & 5 \\ 1 & 2 & 0 & 0 & 1 & 0 & 7 \\ \hline -2 & -3 & 0 & 0 & 0 & 1 & 0 \end{array}\right] \end{array}$$

$$\frac{15}{2} = 7.5 \qquad \frac{5}{2} = 2.5 \qquad \frac{7}{2} = 3.5$$

Step 4) The row with the **smallest *positive*** row quoteint is the first constraint that is violated when increasing the pivot variable.

Step 5) Apply the **row transformation** that

- leaves a 1 in the pivot row of the pivot column,
- leaves a 0 in all other rows of the pivot column

$$\begin{array}{c} \begin{array}{ccccccc} x & y & s_1 & s_2 & s_3 & C & \end{array} \\ \left[\begin{array}{cccccc|c} 3 & 2 & 1 & 0 & 0 & 0 & 15 \\ -1 & 2 & 0 & 1 & 0 & 0 & 5 \\ 1 & 2 & 0 & 0 & 1 & 0 & 7 \\ \hline -2 & -3 & 0 & 0 & 0 & 1 & 0 \end{array}\right] \end{array}$$

$$R_1 \leftarrow (R_1 - R_2) \quad , \quad R_2 \leftarrow \tfrac{1}{2}R_2 \quad , \quad R_3 \leftarrow (R_3 - R_2) \quad , \quad R_4 \leftarrow (R_4 + \tfrac{3}{2}R_2)$$

Step 5) Repeat until the final row contains only positive coefficients.

$$\begin{array}{c} \begin{array}{ccccccc} x & y & s_1 & s_2 & s_3 & C & \end{array} \\ \left[\begin{array}{cccccc|c} 4 & 0 & 1 & -1 & 0 & 0 & 10 \\ -0.5 & 1 & 0 & 0.5 & 0 & 0 & 2.5 \\ 2 & 0 & 0 & -1 & 1 & 0 & 2 \\ \hline -3.5 & 0 & 0 & 1.5 & 0 & 1 & 7.5 \end{array}\right] \end{array}$$

$$R_1 \leftarrow (R_1 - 2R_3) \quad , \quad R_2 \leftarrow (R_2 + \tfrac{1}{4}R_3) \quad , \quad R_3 \leftarrow \tfrac{1}{2}R_3 \quad , \quad R_4 \leftarrow (R_4 + \tfrac{7}{4}R_3)$$

Step 5) Repeat until the final row contains only positive coefficients.

$$\begin{array}{c} \begin{array}{ccccccc} x & y & s_1 & s_2 & s_3 & C & \end{array} \\ \left[\begin{array}{cccccc|c} 0 & 0 & 1 & 1 & -2 & 0 & 6 \\ 0 & 1 & 0 & 0.25 & 0.25 & 0 & 3 \\ 1 & 0 & 0 & -0.5 & 0.5 & 0 & 1 \\ \hline 0 & 0 & 0 & -0.25 & 1.75 & 1 & 11 \end{array}\right] \end{array}$$

$$R_1 \leftarrow (R_1 - 2R_3) \quad , \quad R_2 \leftarrow (R_2 + \tfrac{1}{4}R_3) \quad , \quad R_3 \leftarrow \tfrac{1}{2}R_3 \quad , \quad R_4 \leftarrow (R_4 + \tfrac{7}{4}R_3)$$

Step 5) Repeat until the final row contains only positive coefficients.

$$\begin{array}{c} \begin{array}{ccccccc} x & y & s_1 & s_2 & s_3 & C & \end{array} \\ \left[\begin{array}{cccccc|c} 0 & 0 & 1 & 1 & -2 & 0 & 6 \\ 0 & 1 & 0 & 0.25 & 0.25 & 0 & 3 \\ 1 & 0 & 0 & -0.5 & 0.5 & 0 & 1 \\ \hline 0 & 0 & 0 & -0.25 & 1.75 & 1 & 11 \end{array}\right] \end{array}$$

$$R_1 \leftarrow R_1 \quad , \quad R_2 \leftarrow (R_1 - \tfrac{1}{4}R_1) \quad , \quad R_3 \leftarrow (R_3 + \tfrac{1}{2}R_1) \quad , \quad R_4 \leftarrow (R_4 + \tfrac{1}{4}R_1)$$

Step 5) Repeat until the final row contains only positive coefficients.

$$\begin{array}{c} \begin{array}{ccccccc} x & y & s_1 & s_2 & s_3 & C & \end{array} \\ \left[\begin{array}{cccccc|c} 0 & 0 & 1 & 1 & -2 & 0 & 6 \\ 0 & 1 & -0.25 & 0 & 0.75 & 0 & 1.5 \\ 1 & 0 & 0.5 & 0 & -0.5 & 0 & 4 \\ \hline 0 & 0 & 0.25 & 0 & 1.25 & 1 & 12.5 \end{array}\right] \end{array}$$

$$\boxed{0.25s_1 + 1.25s_3 + C = 12.5}$$

$$R_1 \leftarrow R_1 \quad , \quad R_2 \leftarrow (R_1 - \tfrac{1}{4}R_1) \quad , \quad R_3 \leftarrow (R_3 + \tfrac{1}{2}R_1) \quad , \quad R_4 \leftarrow (R_4 + \tfrac{1}{4}R_1)$$

- **The Simplex Method (cont.)**

Step 6) Read off the optimal solution

this row tells us the value of s_2

this row tells us the value of y

$$\begin{array}{c} \begin{array}{ccccccc} x & y & s_1 & s_2 & s_3 & C & \end{array} \\ \left[\begin{array}{cccccc|c} 0 & 0 & 1 & 1 & -2 & 0 & 6 \\ 0 & 1 & -0.25 & 0 & 0.75 & 0 & 1.5 \\ 1 & 0 & 0.5 & 0 & -0.5 & 0 & 4 \\ \hline 0 & 0 & 0.25 & 0 & 1.25 & 1 & 12.5 \end{array}\right] \end{array}$$

this row tells us the value of x

$$\boxed{C = 12.5 - 0.25s_1 - 1.25s_3}$$

We can maximise C by setting s_1 and s_3 to be zero

Optimal Solution

$x = 4 \quad y = 1.5 \quad s_1 = 0 \quad s_2 = 6 \quad s_3 = 0 \quad C = 12.5$

`http://www.zweigmedia.com/RealWorld/simplex.html`

Week 8

<u>Branch-and-Bound for Integer Programming</u>

Integer Program

Linear Program + Require *Integer* solution

- **Example:**

$$
\begin{aligned}
\text{Maximise:} \quad & 2x + 3y \\
\text{Subject to:} \quad & 3x + 2y \leq 15 \\
& 2y - x \leq 5 \\
& x + 2y \leq 7 \\
& x,\ y \geq 0 \\
& x,\ y \in \mathbb{Z}
\end{aligned}
$$

(this is the same example as earlier, but with the additional integral requirement)

1.5 is not good for example because we want number of students, that's why we require the x, y to be integers Z.

<u>Integer Programming is NP-complete</u>

Theorem Integer Programming in NP-complete.

Proof:

Step 1) Consider an instance of the **Knapsack problem**

Item 1	Item 2	Item 3
Weight : 10	Weight : 20	Weight : 30
Value : £ 60	Value : £ 100	Value : £ 120

with Knapsack size : 100

We will express this instance of the Knapsack problem as an Integer Program

Step 2) We can express this as an **Integer Program**

Value of Items

Weight of Knapsack

$$
\begin{aligned}
\text{Maximise:} \quad & 60x + 100y + 120z \\
\text{Subject to:} \quad & 10x + 20y + 30z \leq 100 \\
& x,\ y,\ z \geq 0 \\
& x,\ y,\ z \in \mathbb{Z}
\end{aligned}
$$

Non-negativity constraint

Integral constraint

Since Knapsack is NP-complete, so too must be Integer Programming.

IF YOU CAN SOLVE INTEGER PROBLEM QUICKLY YOU CAN SOLVE ALL NP-COMPLETE PRBLEMS. AND THIS CAN BE THE PROOF BECAUSE IT IS POLYNOMIALY REDUICBLE TO KNAP SACK PROBLEM WHICH IS A NP-COMPLETE

Branch-and-Bound

- **Branch-and-Bound Algorithm**

Step 1) Solve the **'continuous/linear relaxation'** using the Simplex Method

(remove the requirement that solution must be integers)

$$\begin{aligned} \text{Maximise:} \quad & 2x + 3y \\ \text{Subject to:} \quad & 3x + 2y \leq 15 \\ & 2y - x \leq 5 \\ & x + 2y \leq 7 \\ & x,\ y \geq 0 \\ & \cancel{x,\ y \in \mathbb{Z}} \end{aligned}$$

Solution

$x = 4$ and $y = 1.5$

Step 2) If all values are integral then we are done!

We can just **return the solution** that we have found!

- **Branch-and-Bound Algorithm (cont.)**

Step 3) Else, **branch** to two sub-problems with the variable **bounded** above and below by the floor and ceiling of the previous solution

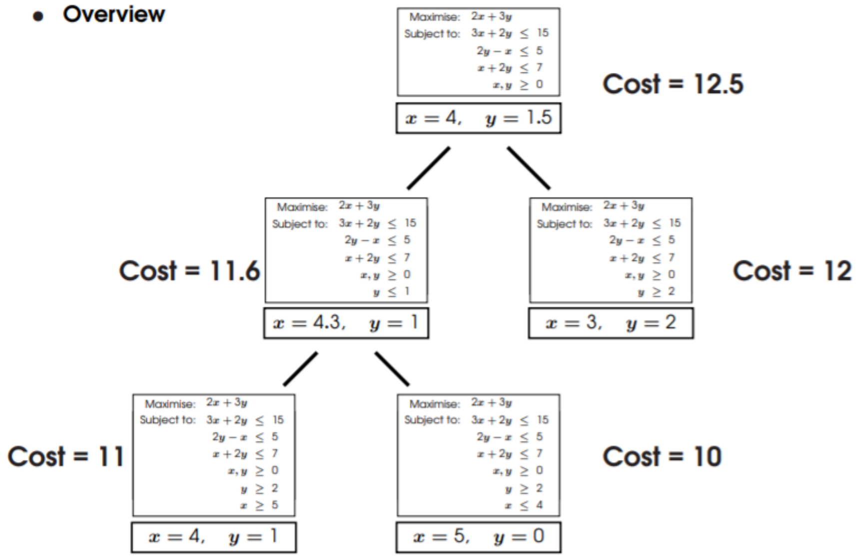
$$\begin{aligned} \text{Maximise:} \quad & 2x + 3y \\ \text{Subject to:} \quad & 3x + 2y \leq 15 \\ & 2y - x \leq 5 \\ & x + 2y \leq 7 \\ & x, y \geq 0 \\ & y \leq 1 \end{aligned} \qquad \begin{aligned} \text{Maximise:} \quad & 2x + 3y \\ \text{Subject to:} \quad & 3x + 2y \leq 15 \\ & 2y - x \leq 5 \\ & x + 2y \leq 7 \\ & x, y \geq 0 \\ & y \geq 2 \end{aligned}$$

Additional Constraints

Step 4) Recursively apply Branch-and-Bound to both sub-problems and return solution which **maximises** the objective function.

(this is yet another example of *Divide-and-Conquer*)

- **Overview**



- Hence the solution to our **Integer Program** is

Solution

$x = 3$ and $y = 2$

$\rightsquigarrow$ **Cost = 12**

- The solution to the Integer Program is always **less optimal** than the solution to its **continuous/linear relaxation**.

- We can **cut corners** by only branching on those children whose are **no worse** than the best integer solution found so far!

(if we had evaluated the right-child before the left-child, we could have stopped early since 11.6 < 12)

That is why Linear programming is in P class and Integer Programming is in NP because following simplex method it would constantly spit out fractional numbers not satisfying our constraints, we can use Branch-and-Bound for this and in some cases it will find a solution in relatively quick time, however there are still cases where it would have to explore the entire tree that is why it is in NP class.

Other Variants of Linear Programming

- **Mixed Integer Linear Progamming (MILP)**
  - Hybrid of an **Integer Program** and a classical **Linear Program**,
  - Some variable are required to be **integers**,
  - Others are allowed to take **non-integral** values.

- **Zero-one Integer Programming**
  - A restriction of **Integer Progamming**,
  - All variables can be either **zero** or **one**,