You are trying to schedule training meetings for a large group of people, each with different availability times. How can you schedule the minimum number of meetings so that everyone can attend at least one?

Need to first collect data of availability times.

Also need a way to store these data mathematically.

Let's say for simplicity all the possible meeting times don't overlap.

Possible Meeting times

People

$$\begin{array}{cc} & \begin{matrix}1 & 2 & 3 & 4 & 5\end{matrix} \\ \begin{matrix}1\\2\\3\\4\end{matrix} & \begin{pmatrix}0 & 1 & 0 & 1 & 1\\ & & & & \\ & & & & \\ & & & & \end{pmatrix}\end{array} \qquad x_2+x_4+x_5 \geq 1$$

person 1 can attend meeting 2

person 1 can't attend meeting 2

Let a_{ij} be our data, where

$$a_{ij} = \begin{cases} 1 & \text{if person } i \text{ can attend meeting } j \\ 0 & \text{otherwise} \end{cases}$$

These are constant, not variables.

Variables of the IP:

$$x_j = \begin{cases} 1 & \text{if we choose to hold meeting } j \\ 0 & \text{otherwise} \end{cases}$$

Minimize $\sum_j x_j$

s.t. $x_j = 0$ or 1 for all i

For all people i,

$$\sum_{j \text{ meeting}} a_{ij} x_j \geq 1$$

Suppose you can only schedule four meetings. How can you do so while maximizing the number of people that can attend?

For each meeting j, let

$$x_j = \begin{cases} 1 & \text{if meeting } j \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

For each person i, let

$$y_i = \begin{cases} 1 & \text{if person } i \text{ can attend any chosen meeting} \\ 0 & \text{otherwise} \end{cases}$$

Maximize $\sum_i y_i$

s.t. $x_j = 0$ or 1 for all j

$y_i = 0$ or 1 for all i

$$\sum x_j \leq 4$$ At most four meetings

For all people i,

$$y_i \leq \sum_{j \text{ meeting}} a_{ij} x_j$$ y_i can be 1 only if i can attend a chosen meeting

Solving integer programs

Linear programs can be solved efficiently, both in theory and practice. (It is in complexity class P.)

Integer and mixed integer programs are theoretically difficult to solve. (It is NP-complete.)

In practice, there are algorithms for solving IP's which are fast enough on real-world problems.

Branch-and-bound is a general paradigm for optimization where one repeatedly branches into subproblems, while eliminating branches that cannot contain the optimum. (NOT ON EXAM)

To solve an IP with this method, we first solve it as a linear program. If we get an integer solution, we are done.

Otherwise, we use the solution to branch the original feasible region into two feasible regions. We repeat the process on each branch.

Each integer solution we obtain in the process gives a bound for the actual solution. If a branch cannot beat this solution, the entire branch can be eliminated.

Max $3x+4y$

s.t. $x+y \leq 10$

$2x+5y \leq 30$

$x, y \geq 0$

$x, y \in \mathbb{Z}$

$2x+5y=30$

$x+y=10$

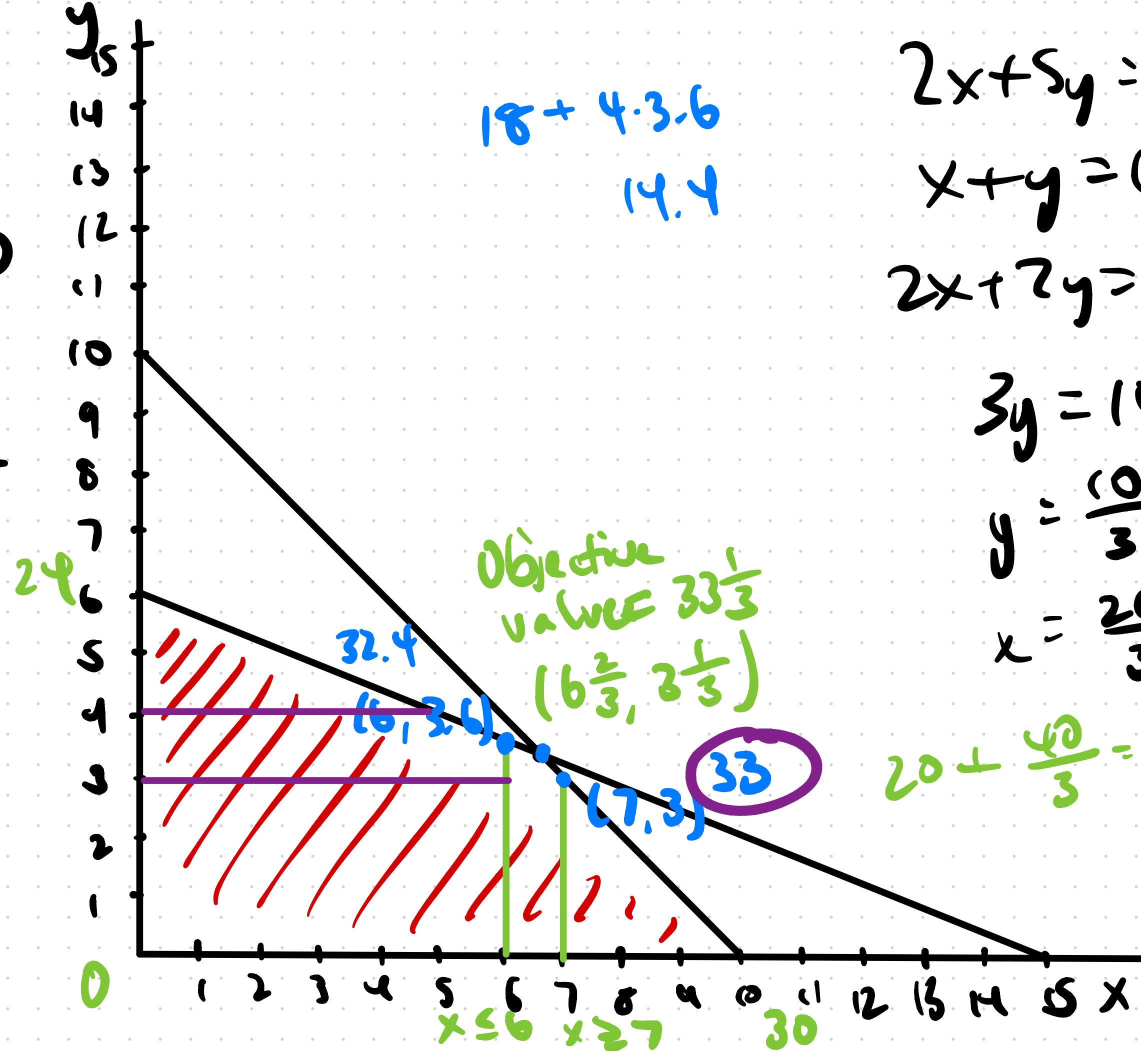
$2x+2y=20$

$3y=10$

$y=\frac{10}{3}$

$x=\frac{20}{3}$

$20+\frac{40}{3}=\frac{100}{3}$



In the worst-case scenario, the branch-and-bound method yields exponentially many branches. In practice, many branches may be eliminated early.

In practice, integer solutions are usually vertex solutions of branches. Therefore simplex method is generally preferred for solving IP's.