1 Outline

In this lecture, we cover

- introduction to integer programming,
- hardness of integer programming,
- modeling categorical decisions,
- facility location.

2 Integer programming models

An **integer linear program** (or simply **integer program (IP)**) is simply a linear program with additional constraints that some variables must take integer values only:

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x_j \in \mathbb{Z}, \quad j \in D \end{aligned}$$

where $D \subseteq [d]$ is the index set of variables required to be integers.

- When **all variables** are required to be integers, i.e., $D = [d]$, then we say that it is a **pure integer program**.
- When **some variables are integers and others are continuous**, we say that the problem is a **mixed integer program (MIP)**.
- Note that $x_j \in \mathbb{Z}$ and $0 \leq x_j \leq 1$ means $x_j \in \{0, 1\}$.
- When all integer variables are in $\{0, 1\}$, we say that the problem is a **0-1 program** or a **binary program**.

The optimization model without the integrality constraints, given by

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \end{aligned}$$

is called the **linear programming relaxation** or the **LP relaxation** of the integer program.

3 Hardness of integer programming

Let us consider a toy example. First, the following is a simple integer linear program with two variables.

$$\begin{aligned} \max \quad & 4x + 5y \\ \text{s.t.} \quad & x + 3y \leq 10, \\ & 3x + y \leq 10, \\ & x, y \geq 0, \\ & (x, y) \in \mathbb{Z}^2. \end{aligned}$$

We can draw the feasible region of the LP relaxation as in Figure 15.1. The red dots depict the

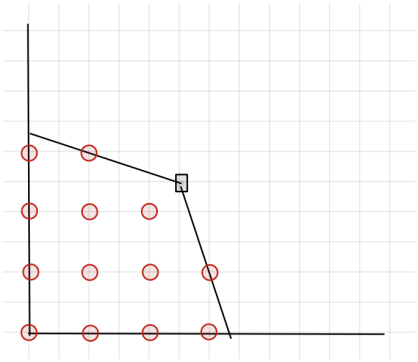

Figure 15.1: Depicting the feasible region of the linear and integer programs

set of solutions that satisfy the linear inequalities and the **integrality condition**, so the set of red dots is the set of feasible solutions to the integer program.

By solving the LP relaxation, we know that the optimal solution is $(x, y) = (5/2, 5/2)$ which is at the intersection of two lines $x + 3y = 10$ and $3x + y = 10$.

What is an optimal solution to the integer program? The first attempt is to look at the optimal solution to the linear program, which is $(5/2, 5/2)$, and round the components to the nearest integers. However, there are a couple of issues with the rounding procedure.

1. There can be many integer solutions obtained from rounding the optimal solution to the linear program. We may round $5/2$ to 2 or 3. Hence, $(2, 2)$, $(2, 3)$, $(3, 2)$, and $(3, 3)$ may be obtained from rounding $(5/2, 5/2)$. When the number of variables is d, the number of solutions from rounding is up to 2^d.

2. We cannot guarantee the feasibility of solutions obtained from rounding. $(2, 2)$ is feasible, but $(2, 3)$, $(3, 2)$, and $(3, 3)$ are all infeasible.

3. More importantly, it is not always the case that there is an optimal solution from the list of solutions obtained from rounding. In fact, the optimal solution to the our integer program is given by $(1, 3)$.

Therefore, linear programming combined with rounding does not necessary solve integer programming. It turns out that integer programming is NP**-hard**, which implies that there would be no polynomial time algorithm unless $P = NP$.

Integer programming includes difficult problems in computer science such as Satisfiability and the Traveling Salesman Problem (TSP). Furthermore, linear programming is a class of convex optimization, while the set of solutions to a integer program is discrete and thus non-convex. These computational challenges require methodologies that deal with the discrete nature of integer programming, which has motivated an extensive research on integer programming both in theory and practice.

There are two main techniques used to solve integer programming models (mostly in combination).

Branch-and-bound (divide and conquer). For the example, as $x \in \mathbb{Z}$, we know that $x \leq 2$ or $x \geq 3$. Then we solve **two** separate linear programs with constraints $x \leq 2$ and $x \geq 3$, respectively.

$$\begin{array}{lll} \max & 4x + 5y & \\ \text{s.t.} & x + 3y \leq 10, & \\ & 3x + y \leq 10, & \\ & x, y \geq 0, & \\ & \boldsymbol{x \leq 2} & \\ & \boldsymbol{y} \in \mathbb{Z}. & \end{array} \quad \text{and} \quad \begin{array}{lll} \max & 4x + 5y & \\ \text{s.t.} & x + 3y \leq 10, & \\ & 3x + y \leq 10, & \\ & x, y \geq 0, & \\ & \boldsymbol{x \geq 3} & \\ & \boldsymbol{y} \in \mathbb{Z}. & \end{array}$$

If these still do not return integer solutions, divide further as necessary, while using past information to eliminate certain linear programs from contention.

Cutting plane methods. Generate constraints in a clever way to force integrality. This also requires repeatedly solving linear programs.

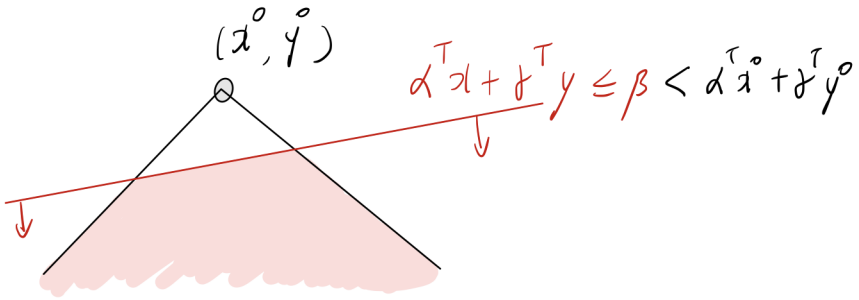

Figure 15.2: Applying a cutting-plane

These techniques are outside the scope of this course but are important in optimization and operations research.

4 Modeling categorical decisions

An important setting where linear programming comes up short is when we have **categorical decisions**. We may use integer programming to deal with situations where we choose among a finite set of alternatives, e.g. Yes/No, a set of locations, a set of finite actions. Note that the alternatives **need not even be quantitative**.

- Non-quantitative decision: **which** brand should I stock?
- Quantitative decision: **how many items** should I stock?

At the same time, we still need to be able to **quantify the effects** of the choices.

- If I choose this brand, then I will make on average $2000 more profit.
- If I go with this other brand, I have to use two more shelves of storage space.

Suppose that we have to select between d different alternatives. We define d variables $x_1, \ldots, x_d$ for the alternatives. Here, we impose that $x_j \in \{0, 1\}$ for $j = 1, \ldots, d$. Then we may model the situation where

$$x_j = \begin{cases} 1, & \text{we choose option } j, \\ 0, & \text{we do not choose option } j. \end{cases}$$

Next, we add constraint $\sum_{j \in [d]} x_j = 1$. Then this requires that exactly one of the d alternatives is chosen. Instead of this constraint, we may change the right-hand side as follows.

- $\sum_{j \in [d]} x_j = k$ means we select **exactly** k of the alternatives.
- $\sum_{j \in [d]} x_j \leq k$ means we select **at most** k of the alternatives.
- $\sum_{j \in [d]} x_j \geq k$ means we select **at least** k of the alternatives.

5 Facility location

Suppose that we have d different suburbs. We want to select k of these to be locations for fire stations.

Categorical decisions: x_j for each $j \in [d]$ binary variables,

$$x_j = \begin{cases} 1, & \text{a fire station is located in suburb } j, \\ 0, & \text{no fire station in suburb } j. \end{cases}$$

Constraints: choose exactly k suburbs

$$\sum_{j \in [d]} x_j = k.$$

Objective: minimize the **largest distance** between a suburb and its closest first station. Here, f_{ij} is the distance between suburbs i and j. Note that the distance between j and its closest first station is

$$\min_{i:\ x_i = 1} f_{ij}$$

Then the largest distance between a suburb and its closest first station is

$$\max_{j \in [d]} \quad \min_{i:\ x_i = 1} f_{ij}$$

Therefore, the objective is

$$\min \quad \max_{j \in [d]} \quad \underbrace{\min_{i:\ x_i = 1} f_{ij}}_{\substack{\text{distance between } j \text{ and} \\ \text{its closest fire station}}}$$

We can rewrite

$$\max_{j \in [d]} \quad \min_{i:\ x_i = 1} f_{ij} = \min \left\{ t :\ t \geq \min_{i:\ x_i = 1} f_{ij}, \quad \forall j \in [d] \right\}$$

Hence, the optimization problem is

$$\begin{aligned} \min \quad & t \\ \text{s.t.} \quad & t \geq \min_{i:\ x_i=1} f_{ij}, \quad \forall j \in [d] \end{aligned}$$

However, constraint $t \geq \min_{i:\ x_i=1} f_{ij}$ is not linear. Here, the trick is to think of $\min_{i:\ x_i=1} f_{ij}$ as another selection: for each $j \in [d]$, we select the station with the smallest distance. More precisely, we add binary variable $y_{ij} \in \{0, 1\}$ to model the selection. For each $i, j \in [d]$,

$$y_{ij} = \begin{cases} 1, & \text{a fire station in suburb } i \text{ is selected by suburb } j, \\ 0, & \text{no fire station in } i \text{ is selected by } j. \end{cases}$$

We make sure that only one fire station is assigned to each suburb.

$$\sum_{i \in [d]} y_{ij} = 1.$$

Moreover, we add constraints to ensure that only the suburbs with fire stations can be selected. In other words, if $x_i = 0$, then $y_{ij} = 0$. This can be modeled by

$$y_{ij} \leq x_{ij}.$$

Then the objective is now

$$\max_{j \in [d]} \sum_{i \in [d]} f_{ij} y_{ij} = \min \left\{ t : \ t \geq \sum_{i \in [d]} f_{ij} y_{ij}, \quad \forall j \in [d] \right\}.$$

Finally, we deduce the following model.

$$\begin{aligned} \min \quad & t \\ \text{s.t.} \quad & x \in \{0,1\}^d, \ y \in \{0,1\}^{d \times d}, \ t \in \mathbb{R} \\ & \sum_{j \in [d]} x_j = k \\ & \sum_{i \in [d]} y_{ij} = 1, \quad j \in [d] \\ & y_{ij} \leq x_i, \quad i, j \in [d] \\ & t \geq \sum_{i \in [d]} f_{ij} y_{ij}, \quad j \in [d] \end{aligned}$$

Note that we are not explicitly enforcing that $y_{ij} = 1$ when i gives rise to the smallest distance d_{ij} among those with $x_i = 1$. Nevertheles, the minimizing objective will select the best one for us.