Linear Program (LP):

Integer (Linear) Program (IP):

Mixed Integer (Linear) Program (MIP):

Binary Integer (Linear) Program (BIP), or 0-1 program:

Combinatorial Optimization Problem: Given a finite set $[n] := \{1, \ldots, n\}$, a weight c_i for each $i \in [n]$, and a set $\mathcal{F} \subseteq 2^{[n]} := \{S \mid S \subseteq [n]\}$ of feasible solutions (i.e., feasible subsets of $[n]$), the associated combinatorial optimization problem (minimization variant) is:

Usually, $\mathcal{F}$ admits a compact description (e.g., set of all s-t paths, or of all spanning trees).

It's good practice, when formulating IPs, to define:

- indices
- sets
- parameters
- variables
- objective function
- constraints (main and nonnegativity)

The assignment problem. There are n people available to carry out n jobs. Each person should be assigned to carry out exactly one job. Some individuals are better suited to particular jobs than others, so there is an estimated cost c_{ij} if person i carries out job j. The problem is to find a minimum cost assignment of people to jobs.

The 0-1 knapsack problem. There is a budget of b dollars available for investment in projects in the coming year, and n projects are under consideration. Here, a_i is the outlay for project i, and c_i is the expected return. Formulate a 0-1 program to maximize the expected return while not exceeding the budget.

The set cover problem. The problem is to decide where to locate ambulances so as to be able to quickly respond to emergencies. There are n sites under consideration, and an ambulance can service areas that are within 8 minutes of its base location. Formulate a 0-1 program that minimizes the total cost to locate ambulances, where the cost to locate an ambulance in region i is c_i dollars.

The (asymmetric) traveling salesman problem (TSP). A salesman must visit each of n cities exactly once and then return to his starting point. The time taken to travel from city i to city j is c_{ij}. (Do not assume that $c_{ij} = c_{ji}$.) Formulate a 0-1 program that will tell the salesman which order to visit the cities so as to minimize the total travel time.

Uncapacitated facility location (UFL). Consider a set $[n] := \{1, \ldots, n\}$ of potential depots and a set $[m] := \{1, \ldots, m\}$ of clients. There is a fixed cost f_i to open depot i. The cost to fulfill client j's demand from depot i is c_{ij}. Assume that a client can receive demand from multiple depots. For example, a client can get half of its demand from depot 1 and the other half from depot 2—in which case it pays $\frac{1}{2}c_{1j} + \frac{1}{2}c_{2j}$. The task is to determine which depots to open and how to fulfill demand such that the total cost is minimized. The total cost is the sum of transportation costs and depot opening costs.

Uncapacitated lot sizing (ULS). The problem is to decide on a production plan for an n-period horizon for a single product. The basic model has data:

- f_t is the fixed cost of producing in period t;
- p_t is the unit production cost in period t;
- h_t is the unit storage cost in period t;
- d_t is the demand in period t.

Disjunctive constraints. Suppose we want a formulation for the set of all $x \in \mathbb{R}^n,\ 0 \le x \le d$ that satisfy at least k of the following m constraints:

$$a^i x \le b_i,\ \forall i \in [m] := \{1, \ldots, m\}.$$

Note: there is an $M \ge 0$ such that for any x with $\mathbf{0} \le x \le d$ and for each i we have $a^i x \le b_i + M$.

Graphs, set packings, and stable sets. The notion of a graph will be important in this class. We usually consider a simple graph $G = (V, E)$ with vertex set V and edge set $E \subseteq \binom{V}{2}$.

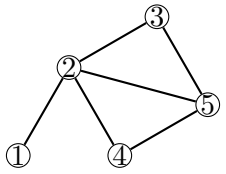

Figure 1: An example graph $G = (V, E)$. What are V and E here?

Definition 1 (stable set)**.** *A subset $S \subseteq V$ of vertices of a graph $G = (V, E)$ is said to be a stable set if, for every edge $\{u, v\} \in E$, at most one of its endpoints (u or v) belongs to S.*

Give a 0-1 programming formulation for maximum stable sets for *arbitrary* $G = (V, E)$. What is a maximum (cardinality) stable set of the graph above?

Definition 2 (set packing)**.** *Given a finite set U and a family $\mathcal{S} := \{S_1, \ldots, S_n\}$ of subsets of U, a set packing is a subfamily of $\mathcal{S}$ in which every two of its members are pairwise disjoint.*

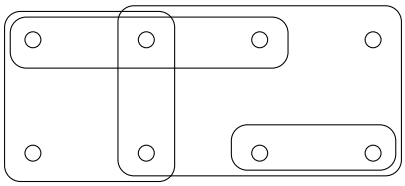

Figure 2: This set packing instance has $n = 4$ and $|U| = 8$.

Formulate a 0-1 program for finding a maximum set packing of the instance above. Then, construct a graph whose stable sets represent the set packings of the instance above. Write out the corresponding 0-1 formulation for maximum stable sets. Which formulation is "better"? Why?

The combinatorial explosion. How many (feasible) solutions are there to:

- the assignment problem?

- the 0-1 knapsack problem? (say, where $b = \frac{1}{2}\sum_i a_i$)

- the (asymmetric) traveling salesman problem?

This implies that brute force will not work well. But, it does not imply the problems are hard. Why?

Point of reference: an iPhone 6 can perform approximately 3.36×10^9 instructions per second.

n	$\lg n$	$\sqrt{n}$	n^2	2^n	$n!$
10	3.32	3.16	10^2	1.02×10^3	3.6×10^6
100	6.64	10.00	10^4	1.27×10^{30}	9.33×10^{157}
1000	9.97	31.62	10^6	1.07×10^{301}	4.02×10^{2567}

Table 1: How rapid do functions grow?

Thus, brute force for 0-1 knapsack (with $n = 100$) would take longer than the universe has existed. But, such instances are routinely solved in seconds with commercial MIP solvers.

Why can't we just solve IPs by solving the LP relaxation and rounding it?

Example 1.1 Consider the integer program:

$$\max 1.00x_1 + 0.64x_2$$
$$50x_1 + 31x_2 \leq 250$$
$$3x_1 - 2x_2 \geq -4$$
$$x_1, x_2 \geq 0 \text{ and integer.}$$

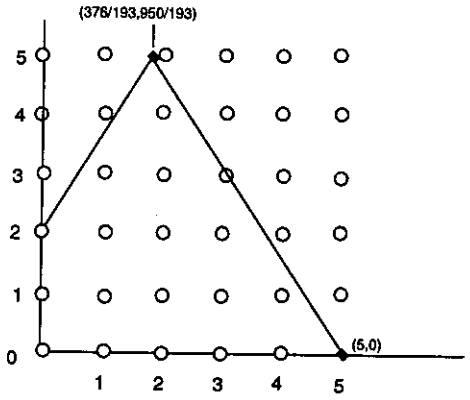

Fig. 1.1 Rounding the LP

As we see from Figure 1.1, the linear programming solution $(376/193, 950/193)$ is a long way from the optimal integer solution $(5, 0)$. ∎

There are many different IP formulations. Which are best?

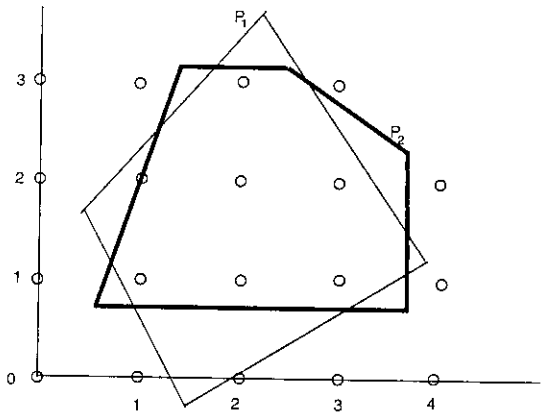

Fig. 1.5 Two different formulations of an IP