Introduction to Mathematical Programming

What is Mathematical Optimization? [1]

"In a mathematical optimization (or programming) problem, one seeks to minimize or maximize a real function of real or integer variables, subject to constraints on the variables. The term mathematical optimization refers to the study of these problems:

- Their mathematical properties
- The development and implementation of algorithms to solve these problems, and
- The application of these algorithms to real world problems.

(...) In fact, the term mathematical programming was coined before the word programming became closely associated with computer software. This confusion is sometimes avoided by using the term optimization as an approximate synonym for mathematical programming."

The Nature of Mathematical Programming [2]

"A mathematical program is an optimization problem of the form:

$$\text{Maximize } f(\boldsymbol{x}) : \boldsymbol{x} \in \boldsymbol{X}, g(\boldsymbol{x}) \leq 0, h(\boldsymbol{x}) = 0$$

where $\boldsymbol{X}$ is a subset of $\mathbb{R}^n$ and is in the domain of the functions, f, g and h, which map into real spaces. The relations, $\boldsymbol{x} \in \boldsymbol{X}$, $g(\boldsymbol{x}) \leq 0$ and $h(\boldsymbol{x}) = 0$ are called constraints, and f is called the objective function.

There are, however, forms that deviate from this paradigm, and it is typically a modeling issue to find an equivalent standard form. Important examples are as follows:

- Fuzzy mathematical programs, Goal programs, Multiple objective programs
- Randomized programs, Stochastic programs"

The Nature of Mathematical Programming (cont.) [2]

- "A point $\boldsymbol{x}$ is feasible if it is in $\boldsymbol{X}$ and satisfies the constraints: $g(\boldsymbol{x}) \leq 0$ and $h(\boldsymbol{x}) = 0$.
- A point $\boldsymbol{x}^*$ is optimal if it is feasible and if the value of the objective function is not less than that of any other feasible solution: $f(\boldsymbol{x}^*) \geq f(\boldsymbol{x})$ for all feasible $\boldsymbol{x}$.
- The sense of optimization is presented here as maximization, but could just as well be minimization, with the appropriate change in the meaning of optimal solution: $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all feasible $\boldsymbol{x}$. (...)"

Taxonomy of Mathematical Programming

It is difficult to provide a single taxonomy since many subfields are linked in different ways. Important classifications include:

- Convex vs. non-convex optimization
- Continuous vs. discrete optimization
- Deterministic vs. non-deterministic optimization
- Constrained vs. unconstrained optimization
- None, one, or many objectives

Optimization tree

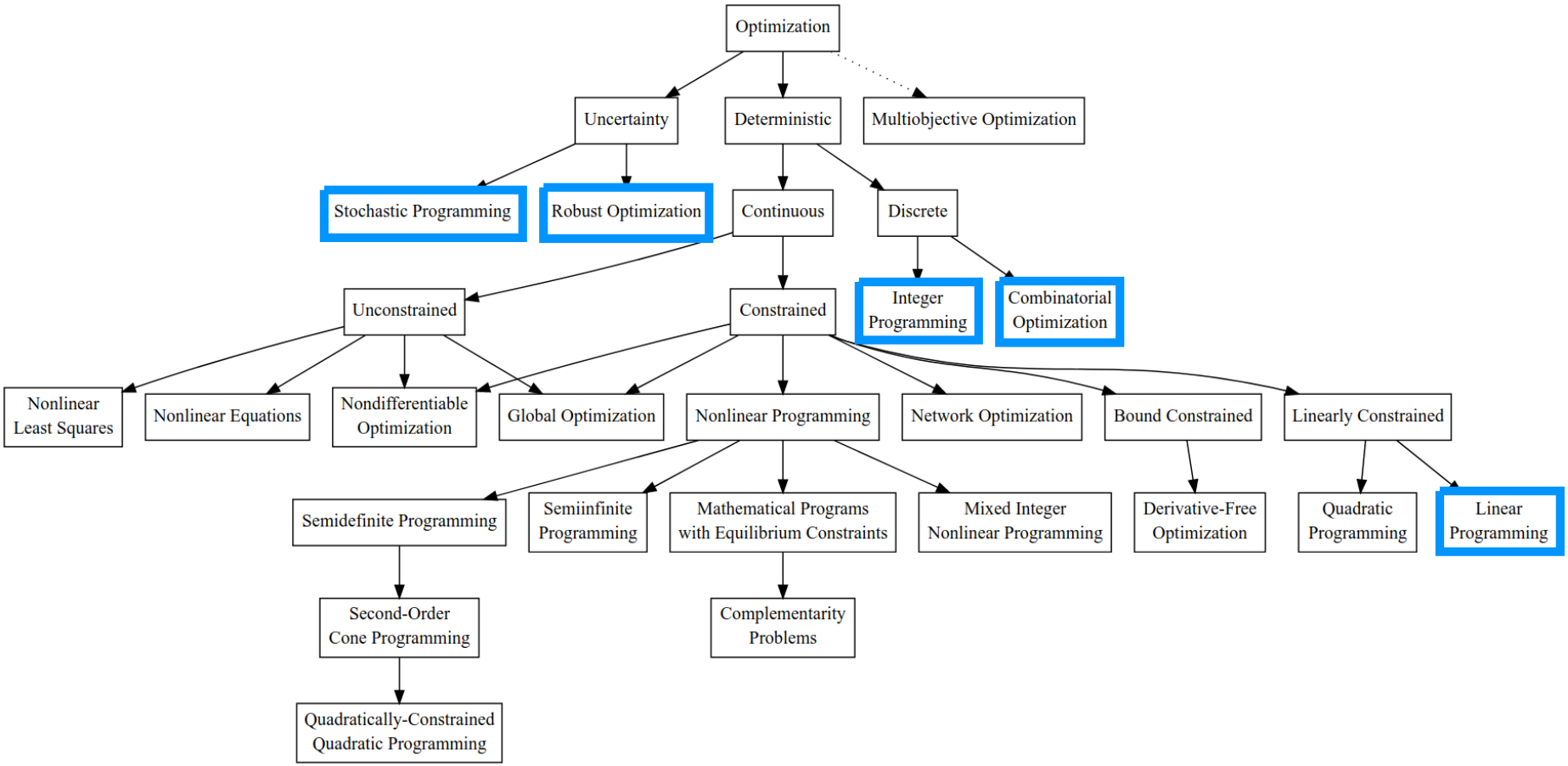


Figure 1: Classification from NEOS Wiki [3]

Repetition, Basics, and some Extensions

Linear Program (LP)

$$\begin{aligned} \min \quad & \boldsymbol{c}'\boldsymbol{x} \\ & \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

where the data are the

- vector $\boldsymbol{c} \in \mathbb{Q}^n$
- matrix $\boldsymbol{A} \in \mathbb{Q}^{m \times n}$, and the
- vector $\boldsymbol{b} \in \mathbb{Q}^m$

and the variables are the

- vector $\boldsymbol{x} \in \mathbb{R}^n_+$

Linear Program (LP)

Set $P \subseteq \mathbb{R}^n_+$ of **feasible solutions** (feasible set) of an LP is

- a convex set ($\Rightarrow$ any local optimum is a global optimum)
- a polyhedron (or polytope if bounded)

Optimal solutions:

- If the feasible set is nonempty and bounded, then there exists an optimal solution which is an extreme point (vertex of polytope)
- If the feasible set is unbounded, two possibilities exist:
  - (i) There exists an optimal solution which is an extreme point
  - (ii) The optimal cost is $-\infty$ (minimization)
- There may be exponentially many (in the number of variables and constraints) extreme points

Algorithms to solve LPs

(Primal) simplex method (G. B. Dantzig, 1947)

- good practical (empirical) performance
- exponential worst-case runtime
- warm start possible (e.g., after adding rows or columns)

Ellipsoid method (L. Khachian, 1979)

- polynomial runtime
- mainly of theoretical value

Interior point (barrier) methods (N. Karmarkar, 1984)

- polynomial runtime, partly competitive (LP)
- applicable for linear and nonlinear convex optimization problems
- M. H. Wright, *The interior-point revolution in optimization: history, recent developments, and lasting consequences*, Bull. Amer. Math. Soc. (42), 39–56, 2005.

The Dual Simplex Method

- "exploits" the strong duality theorem
- dual simplex is a variant that solves the dual problem
- *primal simplex* maintains primal feasibility and works towards dual feasibility
- *dual simplex* starts with a dual feasible solution and works towards primal feasibility

When should we use the dual simplex?

- If a basic feasible solution of the dual problem is available or can be derived easily
- For primal degenerated problems dual simplex often outperforms primal simplex
- More applications on the next slides ...

Sensitivity Analysis

We now consider the standard form problem

$$\begin{aligned} \min \quad & \boldsymbol{c}'\boldsymbol{x} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

and its dual

$$\begin{aligned} \max \quad & \boldsymbol{p}'\boldsymbol{b} \\ & \boldsymbol{p}'\boldsymbol{A} \leq \boldsymbol{c}' \end{aligned}$$

Sensitivity analysis

- studies the **dependence of the optimal cost and the optimal solution** on coefficient matrix $\boldsymbol{A}$, vector $\boldsymbol{b}$, and cost vector $\boldsymbol{c}$,
- predicts the effects of certain parameter changes,
- is important for incomplete knowledge of the problem data,
- studies how to (efficiently) obtain an optimal solution if we add or delete constraints or variables.

Local Sensitivity Analysis

Given

- LP (minimization)
- optimal basis $\boldsymbol{B}$ (= all columns of $\boldsymbol{A}$ corresponding to variables in optimal basic feasible solution)
- associated optimal LP solution $\boldsymbol{x}^*$

Since $\boldsymbol{B}$ is an optimal basis it satisfies

- $\boldsymbol{x}_B^* = \boldsymbol{B}^{-1}\boldsymbol{b} \geq \boldsymbol{0}$ (feasibility)
- reduced costs $\boldsymbol{c}' - \boldsymbol{c}_B'\boldsymbol{B}^{-1}\boldsymbol{A} \geq \boldsymbol{0}'$ (optimality)

What happens if

- a new variable is added
- a new constraint is added
- some entry of $\boldsymbol{A}$, $\boldsymbol{b}$, or $\boldsymbol{c}$ is changed

Add a Variable

New Problem

$$\begin{aligned} \min \quad & \boldsymbol{c}'\boldsymbol{x} + c_{n+1}x_{n+1} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}_{n+1}x_{n+1} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0},\ x_{n+1} \geq 0 \end{aligned}$$

- $(\boldsymbol{x}, x_{n+1}) = (\boldsymbol{x}^*, 0)$ is a basic feasible solution
- optimality: $\overline{c}_{n+1} = c_{n+1} - \boldsymbol{c}_B'\boldsymbol{B}^{-1}\boldsymbol{A}_{n+1} \geq 0$?
  - yes: $(\boldsymbol{x}^*, 0)$ is an optimal solution
  - no: $(\boldsymbol{x}^*, 0)$ is not necessarily optimal
    $\rightarrow$ add column associated with new variable to simplex tableau and apply primal simplex starting from current basis $\boldsymbol{B}$

Add a Constraint

New Problem

$$\begin{aligned} \min \quad & \boldsymbol{c}'\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{a}'_{\boldsymbol{m+1}}\boldsymbol{x} = b_{m+1} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

- if $\boldsymbol{x}^*$ satisfies the new constraint, then $\boldsymbol{x}^*$ is optimal
- otherwise, we can (easily) derive a feasible dual basis and re-optimize using dual simplex algorithm

Parameter Changes

Changes of an entry of

- non-basic column of $\boldsymbol{A}$: affects (primal) optimality
  $\rightarrow$ re-optimize using primal simplex
- basic column of $\boldsymbol{A}$: affects (primal) optimality and (primal) feasibility
- vector $\boldsymbol{b}$: affects (primal) feasibility
  $\rightarrow$ re-optimize using dual simplex
- cost vector $\boldsymbol{c}$: affects (primal) optimality
  $\rightarrow$ re-optimize using primal simplex

Note: One can obtain intervals on changes δ for which the original basis remains optimal.

Integer Linear Program (IP, ILP)

$$\begin{aligned} \max \quad & \boldsymbol{c}'\boldsymbol{x} \\ & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \text{ and integral} \end{aligned}$$

where the data are the

- vector $\boldsymbol{c} \in \mathbb{Q}^n$
- matrix $\boldsymbol{A} \in \mathbb{Q}^{m \times n}$, and the
- vector $\boldsymbol{b} \in \mathbb{Q}^m$

and the variables are the

- vector $\boldsymbol{x} \in \mathbb{Z}_+^n$

Binary Integer Linear Program (BIP, BILP)

$$\begin{aligned} \max \quad & \boldsymbol{c}'\boldsymbol{x} \\ & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \in \{0,1\}^n \end{aligned}$$

where the data are the

- vector $\boldsymbol{c} \in \mathbb{Q}^n$
- matrix $\boldsymbol{A} \in \mathbb{Q}^{m \times n}$, and the
- vector $\boldsymbol{b} \in \mathbb{Q}^m$

and the variables are the

- vector $\boldsymbol{x} \in \{0,1\}^n$

Mixed Integer Linear Program (MIP, MILP)

$$
\begin{aligned}
\max \quad & \boldsymbol{c}'\boldsymbol{x} + \boldsymbol{h}'\boldsymbol{y} \\
& \boldsymbol{A}\boldsymbol{x} + \boldsymbol{G}\boldsymbol{y} \leq \boldsymbol{b} \\
& \boldsymbol{x} \geq \boldsymbol{0} \text{ and integral} \\
& \boldsymbol{y} \geq \boldsymbol{0}
\end{aligned}
$$

where the data are the

- vectors $\boldsymbol{c} \in \mathbb{Q}^n$, $\boldsymbol{h} \in \mathbb{Q}^p$,
- matrices $\boldsymbol{A} \in \mathbb{Q}^{m \times n}$, $\boldsymbol{G} \in \mathbb{Q}^{m \times p}$, and the
- vector $\boldsymbol{b} \in \mathbb{Q}^m$

and the variables are the

- vectors $\boldsymbol{x} \in \mathbb{Z}^n_+$ and $\boldsymbol{y} \in \mathbb{R}^p_+$

Solving (Mixed) Integer Linear Programs

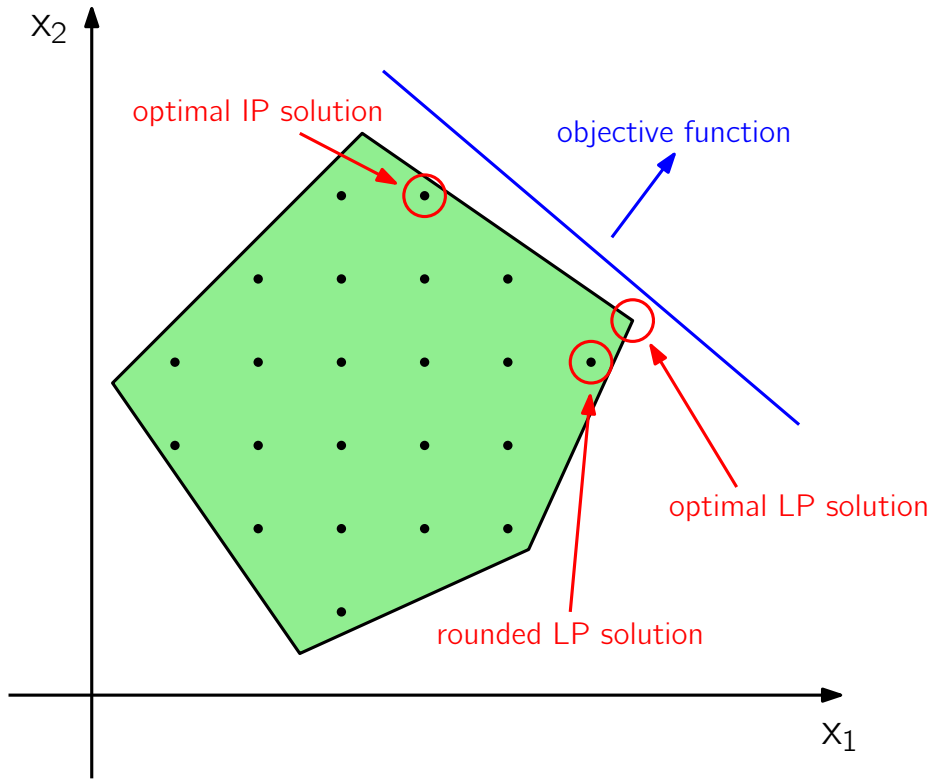


Figure 2: Rounding optimal LP solution? $\Rightarrow$ Does not work!

LP-based Branch-and-Bound

- Most common way to solve (M)IPs
- **LP relaxation** used as dual (upper) bound (maximization)
- **Binary branching** by splitting set S on fractional variables: if x_j is integer and $x_j^{LP} \notin \mathbb{Z}$ then

$$S_1 = S \cap \{\boldsymbol{x} : x_j \leq \lfloor x_j^{LP} \rfloor\},$$
$$S_2 = S \cap \{\boldsymbol{x} : x_j \geq \lceil x_j^{LP} \rceil\}$$

- Resolve LPs with new constraints by **dual simplex algorithm**: former optimal basis stays dual feasible, typically only a few pivots needed to find a new optimal solution ("warm-start")
- **Updating the incumbent solution** (best primal bound): if $\boldsymbol{x}^{t,LP} \in S$ (feasible for original problem) check if this is a new best solution

Algorithm 1: LP-based Branch-and-Bound

problem list $L : \max\{\boldsymbol{c}'\boldsymbol{x} : \boldsymbol{x} \in S\}$
$\underline{z} = -\infty$, incumbent $\boldsymbol{x}^* = NULL$
while $L \neq \emptyset$ **do**
    choose set S_i and remove it from L
    solve $\overline{z}^i = LP(S_i)$
    $\boldsymbol{x}^{i,LP} =$ optimal LP solution
    if $S_i = \emptyset$ **then** prune S_i by infeasibility
    else if $\overline{z}^i \leq \underline{z}$ **then** prune S_i by bound
    else if $\boldsymbol{x}^{i,LP} \in S$ **then** /* $\overline{z}^i = \underline{z}^i = z^i$ */
        if $z^i \geq \underline{z}$ **then**
            update primal bound $\underline{z} = z^i$
            update incumbent $\boldsymbol{x}^* = \boldsymbol{x}^{i,LP}$
        prune S_i by optimality
    else $L = L \cup \{S_{i,1}, S_{i,2}\}$

Definition 1

A polyhedron $P \subset \mathbb{R}^n$ is a *formulation* for a set $X \subseteq \mathbb{Z}^n_+$ if and only if $X = P \cap \mathbb{Z}^n_+$.

$\Rightarrow$ different formulations for a set exist!

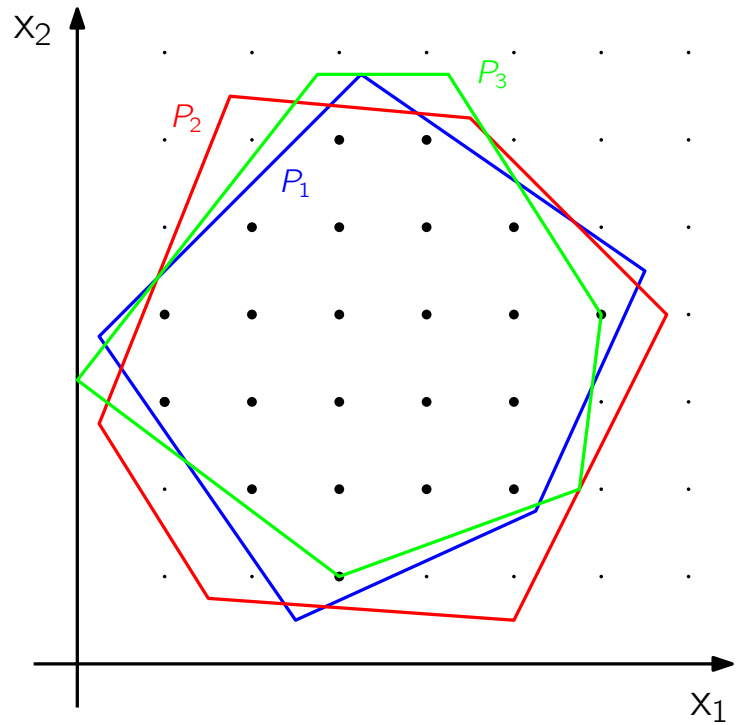
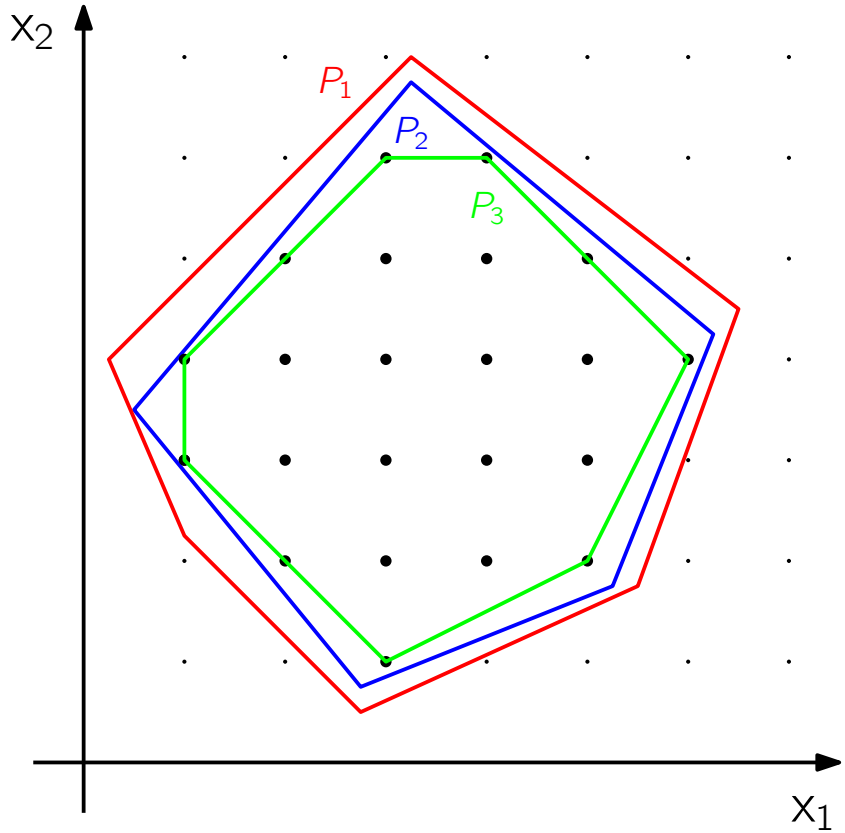


Figure 3: Set X of feasible solutions is defined by the big black dots.



Definition 2

A formulation P for a set $X \subseteq \mathbb{R}^n$ is *ideal* if and only if $P = \text{conv}(X)$.

$\Rightarrow$ If P is an ideal formulation, we can solve the IP $\{\max \ \boldsymbol{cx} : \boldsymbol{x} \in X\}$ by solving the LP $\{\max \ \boldsymbol{cx} : \boldsymbol{x} \in \text{conv}(X)\}$.

Comparing formulations

Given a set $X \subseteq \mathbb{R}^n$ and formulations P_1 and P_2 for X, the following cases are possible:

- P_1 and P_2 are equivalent, i.e. $P_1 = P_2$.
- P_1 is stronger (better) than P_2, i.e. $P_1 \subseteq P_2$ and $\exists \boldsymbol{x} \in P_2 : \boldsymbol{x} \notin P_1$.
- P_2 is stronger (better) than P_1, i.e. $P_2 \subseteq P_1$ and $\exists \boldsymbol{x} \in P_1 : \boldsymbol{x} \notin P_2$.
- P_1 and P_2 are incomparable, i.e. $\exists \boldsymbol{x} \in P_1 : \boldsymbol{x} \notin P_2$ and $\exists \boldsymbol{x} \in P_2 : \boldsymbol{x} \notin P_1$.

Note: If P_1 and P_2 are defined on different variables, we first need to project their polyhedra onto a common space.

Basic Modeling

Linking Variables

Forcing constraints

- Decision A can only be made if decision B has also been made
- Introduce binary variables x_A and x_B equal to one if A (respectively, B) is chosen, and zero otherwise:

$$x_A \leq x_B$$

Relations between variables

- Given n alternatives; at most (exactly) one can (must) be chosen
- Introduce n binary variables x_j, $j = 1, \ldots, n$:

$$\sum_{j=1}^{n} x_j \leq 1 \quad \left(\sum_{j=1}^{n} x_j = 1 \right)$$

Disjunctive Constraints

- Given nonnegative decision vector $\boldsymbol{x} \geq \mathbf{0}$, two constraints $\boldsymbol{a}'\boldsymbol{x} \geq b$ and $\boldsymbol{c}'\boldsymbol{x} \geq d$ with $\boldsymbol{a} \geq \mathbf{0}$ and $\boldsymbol{c} \geq \mathbf{0}$
- **At least one** of the two constraints needs to be satisfied!
- Define binary variable $y \in \{0, 1\}$ and impose constraints:

$$\boldsymbol{a}'\boldsymbol{x} \geq yb$$
$$\boldsymbol{c}'\boldsymbol{x} \geq (1 - y)d$$

- More generally, given m constraints $\boldsymbol{a}_i'\boldsymbol{x} \geq b_i$ with $\boldsymbol{a}_i \geq \mathbf{0}$, $\forall i = 1, \ldots, m$, we require that at least k of them are satisfied
- Introduce binary (indicator) variables $y_i \in \{0, 1\}$, $i = 1, \ldots, m$, and constraints

$$\boldsymbol{a}_i'\boldsymbol{x} \geq b_i y_i \qquad i = 1, \ldots, m$$
$$\sum_{i=1}^{m} y_i \geq k$$

Restricted Value Range

- How to restrict a variable x to take values in a set $\{a_1, \ldots, a_m\}$?
- Introduce binary variables y_j, $j = 1, \ldots, m$, and constraints

$$\sum_{j=1}^{m} a_j y_j = x$$

$$\sum_{j=1}^{m} y_j = 1$$

$$y_j \in \{0, 1\} \qquad j = 1, \ldots, m$$

Note: The set of variables $\{y_1, y_2, \ldots, y_m\}$ is usually called a special ordered set (SOS) of variables.

Piecewise Linear Cost Functions

We want to build a model minimizing a piecewise linear (not necessarily convex) cost function.

- Suppose $a_1 < a_2 < \cdots < a_k$ and
- $f(x)$ is a continuous, piecewise linear function specified by points $(a_i, f(a_i))$, for $i = 1, \ldots, k$, defined on the interval $[a_1, a_k]$, see Figure 4.

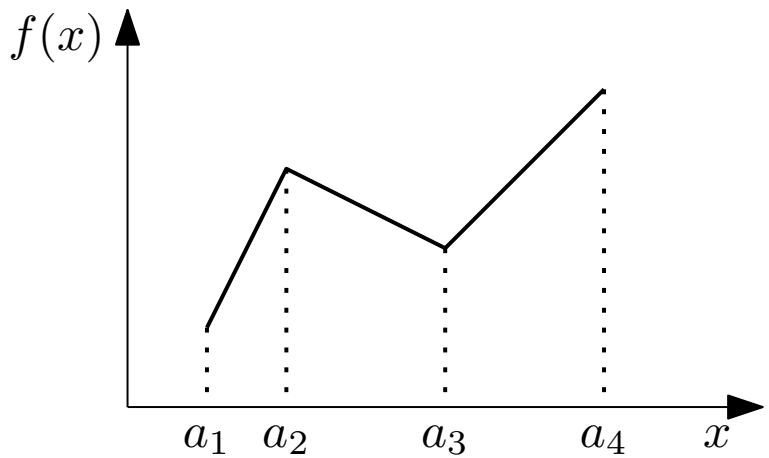


Figure 4: A continuous piecewise linear cost function.

Piecewise Linear Cost Functions – Modeling

- Any $x \in [a_1, a_k]$ and corresponding $f(x)$ can be expressed as

$$x = \sum_{i=1}^{k} \lambda_i a_i, \quad f(x) = \sum_{i=1}^{k} \lambda_i f(a_i), \quad \sum_{i=1}^{k} \lambda_i = 1, \quad \lambda_1, \ldots, \lambda_k \geq 0.$$

- Choice of coefficients $\lambda_1, \ldots, \lambda_k$ is, however, only feasible if we require that at most two consecutive coefficients λ_i can be nonzero
- Then $x \in [a_i, a_{i+1}]$ is represented as $x = \lambda_i a_i + \lambda_{i+1} a_{i+1}$ with $\lambda_i + \lambda_{i+1} = 1$
- We introduce binary variables y_i, $i = 1, \ldots, k-1$, and y_i is equal to one if $a_i \leq x < a_{i+1}$ and zero otherwise

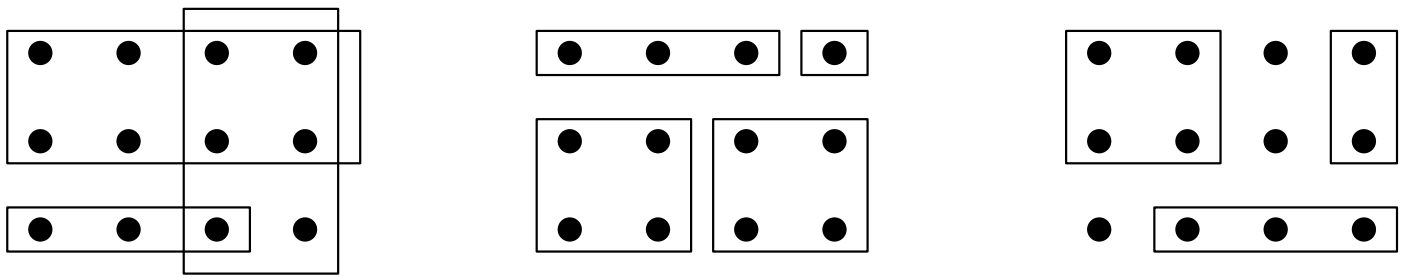
Piecewise Linear Cost Functions – Formulation

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{k} \lambda_i f(a_i) \\
\text{subject to} \quad & \lambda_1 \leq y_1 \\
& \lambda_k \leq y_{k-1} \\
& \lambda_i \leq y_{i-1} + y_i && i = 2, \ldots, k-1 \\
& \sum_{i=1}^{k} \lambda_i = 1 \\
& \sum_{i=1}^{k-1} y_i = 1 \\
& \lambda_i \geq 0 && i = 1, \ldots, k \\
& y_i \in \{0, 1\} && i = 1, \ldots, k-1
\end{aligned}
$$

Set Covering, Set Packing, and Set Partitioning

Given

- set of elements $M = \{1, \ldots, m\}$
- element subsets $M_1, M_2, \ldots, M_n \subseteq M$
- weight w_j for each subset M_j

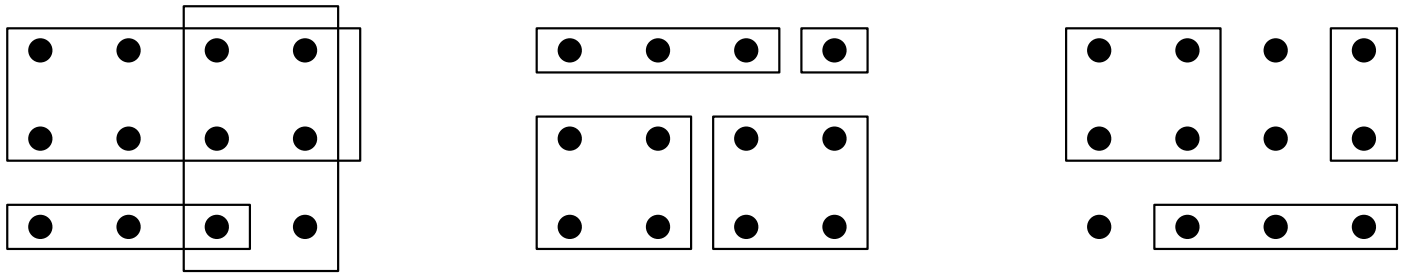


A subset $F \subseteq \{1, \ldots, n\}$ of element subsets $M_1, \ldots, M_n$ is a

- **cover** of M if $\bigcup_{j \in F} M_j = M$
- **packing** of M if $M_j \cap M_k = \emptyset$, for all $j, k \in F$, $j \neq k$
- **partition** of M if it is both a cover and a packing of M

The *total weight* of a set F is defined as $\sum_{j \in F} w_j$.

Set Covering, Set Packing, and Set Partitioning



In the

- **set covering** problem we want to find a minimum weight **cover** F
- **set packing** problem we want to find a maximum weight **packing** F
- **set partitioning** problem minimization and maximization is possible

Note: Many important problems (e.g., crew scheduling, vehicle routing) can be modeled using set covering, packing, or partitioning formulations!

Set Covering, Set Packing, and Set Partitioning

To formulate these problems as MIPs we introduce

- the $m \times n$ incidence matrix $\boldsymbol{A}$ for element subsets $\{M_j \mid j = 1, \ldots, n\}$, whose entries are given by

$$a_{ij} = \begin{cases} 1, & \text{if } i \in M_j \\ 0, & \text{otherwise.} \end{cases}$$

- decision variables x_j, $j = 1, \ldots, n$, equal to one if $j \in F$ and zero otherwise.

Let $\boldsymbol{x} = (x_1, \ldots, x_n)$. Set F is a cover, packing, partition if and only if

$$\boldsymbol{Ax} \geq \mathbf{1}, \quad \boldsymbol{Ax} \leq \mathbf{1}, \quad \boldsymbol{Ax} = \mathbf{1},$$

respectively, where $\mathbf{1}$ is an m-dimensional vector of ones.

The pigeonhole principle

A central proof method in combinatorics which states that

- we cannot place $n+1$ pigeons into n holes
- such that no two pigeons share the same hole.

Formulation for the feasible set:

- Variables: $x_{ij} \in \{0,1\}$, $\forall i = 1,\ldots,n+1$, $\forall j = 1,\ldots,n$, x_{ij} is one if pigeon i occupies hole j, and zero otherwise
- Obviously, the feasible set will be empty!

The pigeonhole principle: Formulations P_1 and P_2

- Place each pigeon in one hole (both formulations)

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n+1$$

- P_1: No two pigeons may be placed in the same hole

$$x_{ij} + x_{kj} \leq 1, \quad i, k = 1, \ldots, n+1, \; i \neq k, \; j = 1, \ldots, n$$

- P_2: At most one pigeon in each hole

$$\sum_{i=1}^{n+1} x_{ij} \leq 1, \quad j = 1, \ldots, n$$

- Is there a significant difference between P_1 and P_2?
- Which formulation is better?

The pigeonhole principle: Strength of P_1 and P_2

LP relaxation of P_2: Summing over all indices, yields

$$\sum_{i=1}^{n+1}\sum_{j=1}^{n} x_{ij} = n+1 \quad \text{and} \quad \sum_{j=1}^{n}\sum_{i=1}^{n+1} x_{ij} \leq n$$

which is a contradiction and hence, the LP relaxation of P_2 is infeasible.

LP relaxation of P_1: Here, the solution $\boldsymbol{x}$ with $x_{ij} = 1/n,\ \forall i, j$, satisfies all constraints!

Conclusions:

- P_2 is stronger than P_1
- P_2 detects infeasibility immediately after solving the LP relaxation
- For P_1 it turns out that almost all (infeasible) integer solutions must be enumerated first

The Traveling Salesman Problem (TSP)

Definition

Given:

- set $N = \{1, ..., n\}$ of cities
- costs c_{ij} for traveling from city i to city j

The problem is to find a tour with minimum traveling costs. Each city has to be visited exactly once and the salesman has to return to his starting city at the end.

Basic Model for TSP

1. **Variables:** $x_{ij} \in \{0,1\},\ \forall i,j \in N,\ i \neq j$
   $x_{ij} = 1$ if the salesman travels directly from city i to j, $x_{ij} = 0$ otherwise
2. **Constraints:**
   - The salesman leaves each city i exactly once:

$$\sum_{j \in N \setminus \{i\}} x_{ij} = 1 \qquad \forall i \in N$$

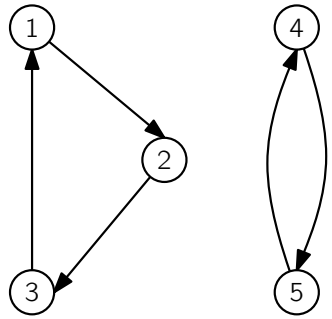
   - The salesman arrives at each city j exactly once:

$$\sum_{i \in N \setminus \{j\}} x_{ij} = 1 \qquad \forall j \in N$$

3. **Objective function:** minimize the total traveling costs

$$\min \quad \sum_{i \in N} \sum_{j \in N \setminus \{i\}} c_{ij} x_{ij}$$

Subtour Elimination



- Problem: subtours are allowed!
- Additional constraints to eliminate these subtours, e.g.:
  - Cut-set constraints:

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1, \quad \forall S \subset N,\ S \neq \emptyset$$

  - Subtour elimination constraints:

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \quad \forall S \subset N,\ |S| \in \{2, ..., n-1\}$$

- Are there other ways to prevent subtours?

Sequential Formulation

- introduced by Miller, Tucker and Zemlin (1960)
- additional (continuous) variables u_i are used to indicate the order in which the cities are visited
- additional constraints:

$$u_i + x_{ij} \leq u_j + M \cdot (1 - x_{ij}) \qquad \forall i, j \in N \setminus \{1\},\ i \neq j$$
$$1 \leq u_i \leq n - 1 \qquad \forall i \in N \setminus \{1\}$$

- M has to inactivate a constraint if $x_{ij} = 0 \Rightarrow M = n - 2$

Single Commodity Flow Formulation

- introduced by Gavish and Graves (1978)
- additional (continuous) variables f_{ij} represent the amount of "flow" on arc (i, j)
- additional constraints: node 1 sends out $n-1$ units of the same commodity and any other node consumes exactly one

$$\sum_{j \in N \setminus \{1\}} f_{1j} = n - 1$$

$$\sum_{i \in N \setminus \{j\}} f_{ij} - \sum_{k \in N \setminus \{j\}} f_{jk} = 1 \qquad \forall j \in N \setminus \{1\}$$

$$0 \leq f_{ij} \leq (n-1) \cdot x_{ij} \qquad \forall i, j \in N,\ i \neq j$$

Multi Commodity Flow Formulation

- introduced by Wong (1980)
- additional (continuous) variables f_{ij}^k represent the amount of "flow" on arc (i, j) for commodity k
- additional constraints: node 1 sends out $n-1$ different commodities each one dedicated to a specific node

$$\sum_{j \in N \setminus \{1\}} f_{1j}^k = 1 \qquad \forall k \in N \setminus \{1\}$$

$$\sum_{i \in N \setminus \{k\}} f_{ik}^k = 1 \qquad \forall k \in N \setminus \{1\}$$

$$\sum_{i \in N \setminus \{j\}} f_{ij}^k - \sum_{i \in N \setminus \{j\}} f_{ji}^k = 0 \qquad \forall j, k \in N \setminus \{1\},\ j \neq k$$

$$0 \leq f_{ij}^k \leq x_{ij} \qquad \forall i, j \in N,\ i \neq j,\ \forall k \in N \setminus \{1\}$$