AI505

Optimization

Linear Constrained Optimization

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Problem Formulation

- If an optimization problem has a linear objective and constraints, it is called a **linear programming problem (linear program, LP)**

- The general form of a linear program is:

$$
\begin{aligned}
\underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{c}^T \boldsymbol{x} \\
\text{subject to} \quad & A\boldsymbol{x} \leq \boldsymbol{b} \\
& D\boldsymbol{x} \geq \boldsymbol{e} \\
& F\boldsymbol{x} = \boldsymbol{g} \\
\\
& \boldsymbol{x}, \boldsymbol{c} \in \mathbb{R}^n, \\
& A \in \mathbb{R}^{m \times n}, \boldsymbol{b} \in \mathbb{R}^m \\
& D \in \mathbb{R}^{p \times n}, \boldsymbol{e} \in \mathbb{R}^p \\
& F \in \mathbb{R}^{q \times n}, \boldsymbol{g} \in \mathbb{R}^q
\end{aligned}
$$

Numerical Example

$$
\begin{aligned}
\underset{x_1, x_2, x_3}{\text{minimize}} \quad & 2x_1 - 3x_2 + 7x_3 \\
\text{subject to} \quad & 2x_1 + 3x_2 - 8x_3 \leq 5 \\
& 4x_1 + x_2 + 3x_3 \leq 9 \\
& x_1 - 5x_2 - 3x_3 \geq -4 \\
& x_1 + x_2 + 2x_3 = 1
\end{aligned}
$$

Modelling in Linear Programming

Example

Given a set of items I, each item with a price p_i and a value v_i, i in I, select the subset of items that maximizes the total value collected subject to a total expense that does not exceed a given budget B.

$$\begin{aligned} \max \; & \sum_{i \in I} p_i x_i \\ \text{s.t.} \; & \sum_{i \in I} v_i x_i \leq B \\ & x_i \in \{0, 1\}, \quad \text{for all } i \text{ in } I \end{aligned}$$

Modelling in Linear Programming

Many problems can be converted into linear programs that have the same solution.

Example

$$\text{minimize } L_1 = \|A\boldsymbol{x} - \boldsymbol{b}\|_1$$

$$\begin{aligned} \min\ & 1^T \boldsymbol{s} \\ \text{s.t. } & A\boldsymbol{x} - \boldsymbol{b} \leq \boldsymbol{s} \\ & -(A\boldsymbol{x} - \boldsymbol{b}) \leq \boldsymbol{s} \end{aligned}$$

Example

$$\text{minimize } L_\infty = \|A\boldsymbol{x} - \boldsymbol{b}\|_\infty$$

$$\begin{aligned} \min\ & t \\ \text{s.t. } & A\boldsymbol{x} - \boldsymbol{b} \leq t1 \\ & -(A\boldsymbol{x} - \boldsymbol{b}) \leq t1 \end{aligned}$$

Problem Formulation

Every general form linear program can be rewritten more compactly in **standard form**

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} \quad & A\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \\ & \boldsymbol{x}, \boldsymbol{c} \in \mathbb{R}^n, \\ & A \in \mathbb{R}^{m \times n}, \boldsymbol{b} \in \mathbb{R}^m \end{aligned}$$

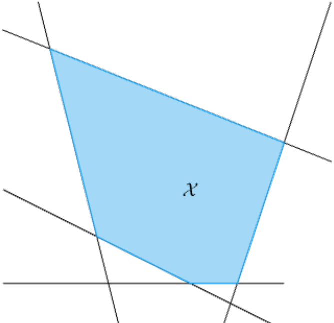
Example

$$\begin{aligned} \text{minimize } & 5x_1 + 4x_2 \\ \text{s.t. } & 2x_1 + 3x_2 \leq 5 \\ & 4x_1 + x_2 \leq 11 \end{aligned}$$

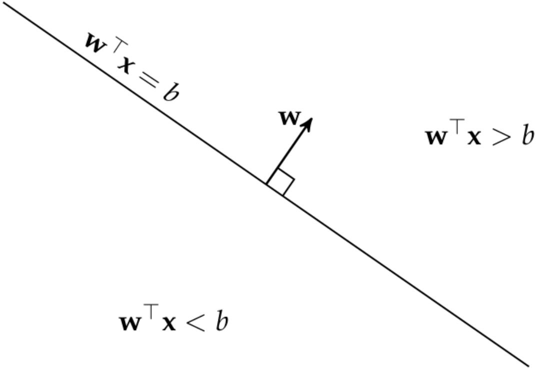
Problem Formulation

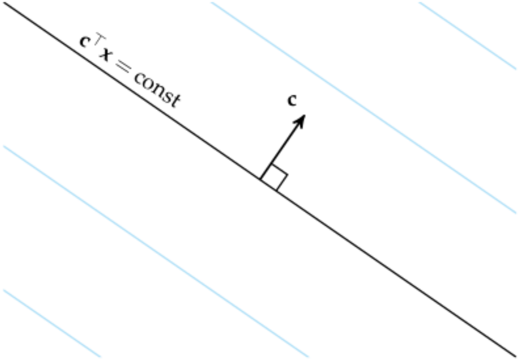
- Each inequality constraint defines a planar boundary of the feasible set called a **half-space**
- The set of inequality constraints define the intersection of multiple half-spaces forming a **convex set**
- Convexity of the feasible set, along with convexity of the objective function, implies that if we find a local feasible minimum, it is also a global feasible minimum.

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} \quad & A\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \end{aligned}$$



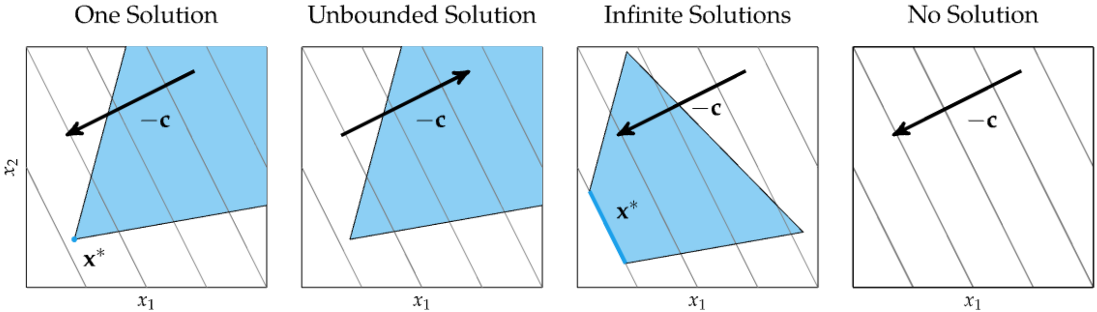
Half-Spaces and Supporting Hyperplanes





Problem Formulation

- How many solutions are there?

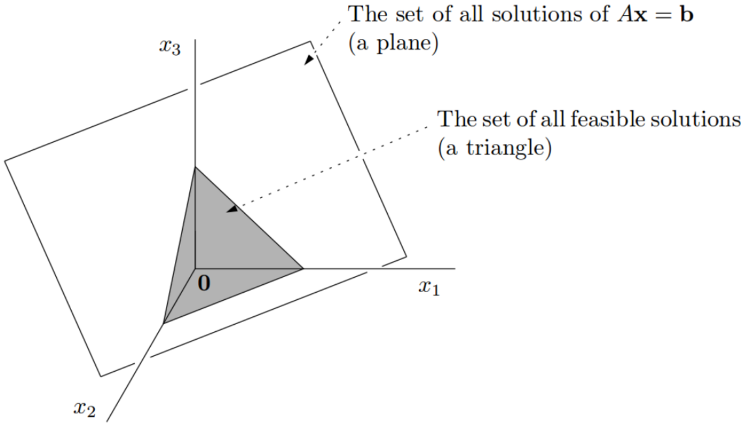


Problem Formulation

Linear programs are often solved in **equality form**

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} \quad & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \end{aligned}$$

$$\boldsymbol{x}, \boldsymbol{c} \in \mathbb{R}^{2n+m}, \quad A \in \mathbb{R}^{m \times 2n+m}, \quad \boldsymbol{b} \in \mathbb{R}^m$$



Simplex Algorithm

- Guaranteed to solve any feasible and bounded linear program
- Works on the equality form
- Assumes that rows of A are linearly independent and $m \leq n'$ ($n' \leq 2n + m$)
- The feasible set of a linear program forms a **polytope** (polyhedra bounded by faces of $n - 1$ dimension)
- The simplex algorithm moves between **vertices** of the polytope until it finds an optimal **vertex**
- Points on faces not perpendicular to $\boldsymbol{c}$ can be improved by sliding along the face in the direction of the projection of $-c$ onto the face.

Fundamental Theorem of LP

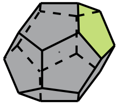
Theorem (Fundamental Theorem of Linear Programming)

Given:

$$\min\{\boldsymbol{c}^T\boldsymbol{x} \mid \boldsymbol{x} \in P\} \text{ where } P = \{\boldsymbol{x} \in \mathbb{R}^n \mid A\boldsymbol{x} \le \boldsymbol{b}\}$$

If P is a bounded polyhedron and not empty and x^* is an optimal solution to the problem, then:

- $\boldsymbol{x}^*$ *is an extreme point (vertex) of P, or*
- $\boldsymbol{x}^*$ *lies on a face $F \subset P$ of optimal solution*



Proof:

- assume x^* not a vertex of P then $\exists$ a ball around it still in P. Show that a point in the ball has better cost
- if x^* is not a vertex then it is a convex combination of vertices. Show that all points are also optimal.

Simplex Algorithm

- Every vertex for a linear program in equality form can be uniquely defined by $n - m$ components of $\boldsymbol{x}$ that equal zero.
- choosing m design variables and setting the remaining variables to zero effectively removes $n - m$ columns of A, yielding an $m \times m$ constraint matrix
- the m selected columns of the matrix A are called **basis** and denoted by B: $\quad x_i \geq 0$ for $i \in B$
- the $n - m$ columns not in B are called **not in basis** and are denoted by V: $\quad x_i = 0$ for $i \in V$.

$$A\boldsymbol{x} = A_B \boldsymbol{x}_B = \boldsymbol{b} \quad \Longrightarrow \quad x_B = A_B^{-1}\boldsymbol{b}$$

Simplex Algorithm

- every vertex has an associated partition (B, V),
- not every partition corresponds to a vertex.
  A_B might be not invertible or the point $\boldsymbol{x}_B$ might not be ≥ 0.
- identifying partitions that correspond to vertices corresponds to solving an LP problem as well!

Two phases of the algorithm

1. **Initialization Phase**: finding a feasible starting vertex
2. **Optimization Phase**: finding the optimal vertex

Simplex Algorithm: FONCs

Lagrangian function:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \boldsymbol{c}^T\boldsymbol{x} - \mu^T\boldsymbol{x} - \lambda^T(A\boldsymbol{x} - \boldsymbol{b})$$

Conditions for Optimality for linear programs: KKT are also sufficient:

- feasibility: $A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq 0$
- dual feasibility: $\boldsymbol{\mu} \geq 0$
- complementary slackness: $\boldsymbol{\mu} \cdot \boldsymbol{x} = 0$
- stationarity: $A^T\boldsymbol{\lambda} + \boldsymbol{\mu} = \boldsymbol{c}$

$$A^T\boldsymbol{\lambda} + \boldsymbol{\mu} = \boldsymbol{c} \quad \Longrightarrow \quad \begin{cases} A_B^T\boldsymbol{\lambda} + \boldsymbol{\mu}_B = \boldsymbol{c}_B \\ A_V^T\boldsymbol{\lambda} + \boldsymbol{\mu}_V = \boldsymbol{c}_V \end{cases}$$

- We can choose $\mu_B = 0$ to satisfy complementry slackness (because $\boldsymbol{x}_B \geq 0$)

$$\boldsymbol{\mu}_V = \boldsymbol{c}_V - \left(A_B^{-1}A_V\right)^T \boldsymbol{c}_B$$

- Knowing $\boldsymbol{\mu}_V$ allows us to assess the optimality of the vertices. If $\boldsymbol{\mu}_B$ contains negative components, then dual feasibility is not satisfied and the vertex is sub-optimal.

- maintain a partition (B, V), which corresponds to a vertex of the feasible set polytope.

- The partition can be updated by swapping indices between B and V. Such a swap equates to moving from one vertex along an edge of the feasible set polytope to another vertex.

Simplex Algorithm: Optimization Phase

Pivoting

- $q \in V$ to enter in B

$$A\boldsymbol{x}' = A_B\boldsymbol{x}'_B + A_{\{q\}}x'_q = A_B\boldsymbol{x}_B = A\boldsymbol{x} = \boldsymbol{b}$$

- $p \in B$ to leave B becomes zero during the transition.

$$\boldsymbol{x}'_B = \boldsymbol{x}_B - A_B^{-1}A_{\{q\}}x'_q \quad \Longrightarrow \quad (\boldsymbol{x}'_B)_p = 0 = (\boldsymbol{x}_B)_p - \left(A_B^{-1}A_{\{q\}}\right)_p x'_q$$

- leaving index is obtained using the **minimum ratio test**: compute x'_q for each potential leaving index p and select the leaving index p that yields the smallest x'_q.

- Choosing an entering index q decreases the objective function value by

$$\boldsymbol{c}^T\boldsymbol{x}' = \boldsymbol{c}_B^T\boldsymbol{x}'_B + c_q x'_q = \boldsymbol{c}^T\boldsymbol{x} + \mu_q \boldsymbol{x}'_q$$

- The objective function decreases only if μ_q is negative.

Simplex Algorithm: Optimization Phase

- In order to progress toward optimality, we must choose an index q in V such that μ_q is negative. If all components of μ_V are non-negative, we have found a global optimum.
- Since there can be multiple negative entries in μ_V, Several possible heuristics to search for optimal vertex (choose next q)
  - **Dantzig's rule**: choose most negative entry in μ; easy to calculate
  - **Greedy heuristic (largest decrease)**: maximally reduces objective at each step
  - **Bland's rule**: chooses first vertex found with negative μ; useful for preventing or breaking out of cycles

Simplex Algorithm: Initialization Phase

- The starting vertex of the optimization phase is found by solving an additional **auxiliary linear program** that has a known feasible starting vertex

$$\begin{aligned}\underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} \quad & \begin{bmatrix} 0^T & 1^T \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} \\ & \begin{bmatrix} A & Z \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} = \boldsymbol{b} \\ & \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} \geq 0\end{aligned}$$

- The solution is a feasible vertex in the original linear program

Dual Certificates

- Verification that the solution returned by the algorithm is actually the correct solution
- Recall that the solution to the dual problem, d^* provides a lower bound to the solution of the primal problem, p^*
- If $d^* = p^*$ then p^* is guaranteed to be the unique optimal value because the duality gap is zero
- What happens if one of the two is unbounded or infeasible?

Dual Certificates

Linear programs have a simple dual form:

Primal form (equality)

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} \quad & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \end{aligned}$$

Dual form

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \quad & \boldsymbol{b}^T \boldsymbol{\lambda} \\ \text{subject to} \quad & A^T \boldsymbol{\lambda} \leq \boldsymbol{c} \end{aligned}$$

Strong Duality Theorem

Due to Von Neumann and Dantzig 1947 and Gale, Kuhn and Tucker 1951.

Theorem (Strong Duality Theorem)

Given:

$$(P)\ \min\{\boldsymbol{c}^T\boldsymbol{x} \mid A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq 0\}$$

$$(D)\ \max\{\boldsymbol{b}^T\boldsymbol{\lambda} \mid A^T\boldsymbol{\lambda} \geq \boldsymbol{c}\}$$

exactly one of the following occurs:

1. *(P) and (D) are both infeasible*
2. *(P) is unbounded and (D) is infeasible*
3. *(P) is infeasible and (D) is unbounded*
4. *(P) has feasible solution, then let an optimal be:* $\boldsymbol{x}^* = [x_1^*, \ldots, x_n^*]$
   (D) has feasible solution, then let an optimal be: $\boldsymbol{\lambda}^* = [\lambda_1^*, \ldots, \lambda_m^*]$*, then:*

$$p^* = \boldsymbol{c}^T\boldsymbol{x}^* = \boldsymbol{b}^T\boldsymbol{\lambda}^* = d^*$$

Summary

- Linear programs are problems consisting of a linear objective function and linear constraints
- The simplex algorithm can optimize linear programs globally in an efficient manner
- Dual certificates allow us to verify that a candidate primal-dual solution pair is optimal