CME 307: Optimization CME 307 / MS&E 311 / OIT 676

Lecture 3: Solving and modeling LPs

Fall 2025 *Prof. Udell*

1 Solving linear programs

We now turn to algorithms for solving linear programs. The main idea, motivated by the geometry we have developed, is that if an LP has a solution, then some *corner point* (BFS / vertex / extreme point) is optimal. Thus algorithms can search among vertices rather than the entire feasible polyhedron.

1.1 Algorithms for LPs

- *Enumerate vertices*: generate all BFS and evaluate the objective.
- *Fourier–Motzkin elimination*: eliminate variables to project down to a feasible interval.
- *Simplex method*: walk along edges from BFS to BFS, improving the objective.
- *Ellipsoid method*: first polynomial-time algorithm, theoretical but not practical.
- *Interior point methods*: polynomial-time and practically efficient.
- *First-order methods*: scalable to very large problems, using gradient information only.

Remark 1.1. Enumeration and elimination are conceptually simple but not practical. The simplex algorithm remains a workhorse in practice. Interior point methods (IPMs) provide an alternative with strong complexity guarantees. First-order methods are attractive for large-scale machine learning problems.

1.2 Fourier–Motzkin elimination*

Fourier–Motzkin elimination is a procedure for projecting a polyhedron onto a subspace by eliminating variables. It is a conceptually simple algorithm, and one of the earliest algorithms for linear programming, but it is not practical for large problems. The algorithm is named after Joseph Fourier, who proposed the method in 1826, and Theodore Motzkin who re-discovered it in 1936. As a side benefit, it shows that the projection of a polyhedron is again a (closed) polyhedron.

We illustrate the method with an example.

Example 1.2 (Fourier–Motzkin elimination)**.** Consider the system

$$\begin{aligned} x_1 + 2x_2 &\leq 4, \\ -x_1 + x_2 &\leq 1, \\ x_1, x_2 &\geq 0. \end{aligned}$$

We eliminate x_1. Collect inequalities bounding x_1:

$$\{0,\ x_2 - 1\} \ \le\ x_1 \ \le\ 4 - 2x_2.$$

Hence feasible x_2 must satisfy

$$0 \le 4 - 2x_2, \qquad x_2 - 1 \le 4 - 2x_2, \qquad x_2 \ge 0.$$

Simplifying, $x_2 \in [0, 5/3]$. Thus the projection of the feasible region onto x_2 is the interval $[0, 5/3]$. This method shows that the projection of a polyhedron is again a (closed) polyhedron.

1.3 Enumerating vertices

One of the main benefits of the basic feasible solution (BFS) concept is that it allows us to enumerate all vertices of a polyhedron.

Proposition 1.3. *Let $A \in \mathbb{R}^{m\times n}$ with full row rank m. Then every BFS corresponds to some choice of m columns A_S with $|S| = m$ and A_S invertible. The BFS is*

$$x_S = A_S^{-1} b, \qquad x_{\bar{S}} = 0.$$

If $x_S \ge 0$, this point is feasible.

Proof. Direct from the definition of a BFS: pick m basic variables, solve the equality, and set the rest to zero. Feasibility requires nonnegativity. Thus every BFS arises in this way. □

Remark 1.4. Enumerating all BFS requires considering $\binom{n}{m}$ bases, which is exponential in m. Hence naive enumeration is not a practical algorithm.

1.4 Simplex algorithm

The simplex algorithm is a local search method on the set of BFS. It starts at some BFS and repeatedly moves to a neighboring BFS with strictly better objective value, until no improving neighbor exists. Historically, the simplex algorithm was the first practical method for solving LPs, developed by George Dantzig in the 1940s.

Definition 1.5 (Simplex algorithm)**.** The simplex algorithm is a *local search* on BFS:

1. Start at some BFS x.
2. Move to a neighboring BFS x' with strictly better objective $c^T x'$.
3. Repeat until no improving neighbor exists.

The simplex algorithm raises several key questions:

- How to find an initial BFS?

- How to choose a neighboring BFS that improves the objective?
- How to prove optimality when no improving neighbor exists?

1.5 Finding an initial BFS

One approach is to solve an auxiliary problem for which a BFS is obvious:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m} z_i \\ \text{subject to} & Ax + Dz = b, \\ & x, z \geq 0, \end{array}$$

where $D \in \mathbb{R}^{m \times m}$ is diagonal with $D_{ii} = \text{sign}(b_i)$. The BFS $x = 0$, $z = |b|$ is always feasible. If the optimal solution has $z = 0$, then x is a BFS for the original problem.

Exercise. Show that the auxiliary problem has the following properties:

1. If the original problem is feasible, then the auxiliary problem has optimal value zero.
2. If the original problem is infeasible, then the auxiliary problem has optimal value greater than zero.
3. If the original problem is unbounded, then the auxiliary problem is also unbounded.

1.6 Optimization via reduced costs

Let us now address the other two questions: how to choose an improving neighboring BFS, and how to prove optimality when no improving neighbor exists. We will use the concept of *reduced cost* to answer both questions: the reduced cost tells us how the objective changes when we move from one BFS to a neighboring BFS.

Suppose x is a nondegenerate BFS with active set S. Consider entering variable $j \notin S$. We will increase x_j from zero, and adjust the basic variables x_S to maintain feasibility. Define the *basic direction* d^j:

$$d^j_j = 1, \qquad d^j_S = -A_S^{-1} A_j, \qquad d^j_i = 0 \text{ for } i \notin S \cup \{j\}.$$

Then $Ad^j = 0$, so feasibility is preserved. For sufficiently small $\theta > 0$, $x + \theta d^j \geq 0$. The change in the objective is

$$c^T(x + \theta d^j) - c^T x = \theta\, \bar{c}_j,$$

where

$$\bar{c}_j := c_j - c_S^T A_S^{-1} A_j$$

is the *reduced cost* of variable j.

An improving neighbor. To find a neighboring BFS with better objective, we can compute the reduced costs $\bar{c}_j$ for all $j \notin S$. If any $\bar{c}_j < 0$, then moving in direction d^j improves the objective. We can increase θ until some basic variable x_i hits zero, which gives a new BFS with active set $S \cup \{j\} \backslash \{i\}$. This is called a *pivot* operation.

Remark 1.6. The choice of entering variable j and leaving variable i can affect the number of iterations required for convergence. There are many pivot rules; a common choice is to pick the entering variable with most negative reduced cost.

An optimality criterion. The reduced cost also gives a simple optimality test:

Proposition 1.7 (Reduced cost optimality test)**.** *If all reduced costs $\bar{c}_j \geq 0$ for $j \notin S$, then the BFS x is optimal.*

Proof. Any feasible direction d at x is a conic combination of $\{d^j : j \notin S\}$. (See Proposition 1.8.) Thus for any feasible $x' = x + \sum_{j \notin S} \alpha_j d^j$ with $\alpha \geq 0$,

$$c^T x' = c^T x + \sum_{j \notin S} \alpha_j \bar{c}_j \quad \geq \quad c^T x,$$

so no feasible point has smaller objective value. Hence x is optimal. □

Proposition 1.8. *The feasible set of the LP lies in the cone generated by the basic directions $x + \text{cone}(\{d^j : j \notin S\})$ for any BFS x.*

Proof. The basic directions d^j for $j \notin S$ form a basis for the nullspace of A. To see this, note that $Ad^j = 0$ by construction. Moreover, the $n-m$ vectors d^j are linearly independent: if $\sum_{j \notin S} \alpha_j d^j = 0$, then looking at the components in $\bar{S}$ shows $\alpha_j = 0$ for all $j \notin S$. Since $\dim(\text{null}(A)) = n - m$, the d^j form a basis.

Now consider any feasible point x' with $Ax' = b$ and $x' \geq 0$. Then $d = x' - x$ satisfies $Ad = 0$, so d lies in the nullspace of A. Thus $d = \sum_{j \notin S} \alpha_j d^j$ for some $\alpha_j \in \mathbb{R}$. Since $x' = x + d \geq 0$ and $x_j = 0$ for all $j \notin S$, we must have $\alpha_j \geq 0$ for all $j \notin S$. Thus $x' \in x + \text{cone}(\{d^j : j \notin S\})$. □

Gotcha 1.9. The reduced cost optimality test requires a *nondegenerate* BFS. We say that a BFS x is *degenerate* if some basic variables are zero. If x is degenerate, some basic directions d^j may not be feasible directions, and the argument above fails. Hence the optimality test Proposition 1.7 provides a sufficient but not necessary condition for optimality.

1.7 Worked example

Example 1.10 (Two-variable LP)**.** Consider

$$\begin{array}{ll} \text{minimize} & -x_1 - 2x_2 \\ \text{subject to} & x_1 + x_2 \leq 4, \\ & x_1 + 3x_2 \leq 6, \\ & x_1, x_2 \geq 0. \end{array}$$

The feasible region is a polygon in $\mathbb{R}^2$ with BFS at $(0,0)$, $(0,2)$, $(3,1)$, $(4,0)$. Evaluating the objective shows the minimum occurs at $(3,1)$. Reduced costs confirm optimality: no improving direction exists at this BFS.

1.8 Exercises

Exercise. Consider the LP

$$\begin{aligned} \text{minimize} \quad & x_1 - x_2 \\ \text{subject to} \quad & x_1 + 2x_2 \leq 4, \\ & 2x_1 + x_2 \leq 5, \\ & x_1, x_2 \geq 0. \end{aligned}$$

(a) List all BFS by choosing pairs of active constraints.

(b) For each BFS, compute the objective value.

(c) Identify the optimal BFS.

(d) Verify optimality using reduced costs.

2 Modeling linear programs

We now turn to *modeling*: the art of expressing practical decision problems as linear programs. Many common constraints and objectives can be represented in LP form, often using additional variables and transformations.

2.1 Inequality constraints

Definition 2.1 (Slack variables)**.** An inequality constraint $Ax \leq b$ can be written in *standard form* using a slack variable $s \geq 0$:

$$Ax \leq b \quad \Longleftrightarrow \quad Ax + s = b, \;\; s \geq 0.$$

Example 2.2 (Standard form with slack variables)**.** Consider

$$\begin{aligned} \text{minimize} \quad & x_1 + 2x_2 \\ \text{subject to} \quad & x_1 + x_2 \leq 3, \\ & x_1, x_2 \geq 0. \end{aligned}$$

Introducing a slack $s \geq 0$ yields

$$x_1 + x_2 + s = 3, \quad x_1, x_2, s \geq 0.$$

Exercise. Rewrite the problem

$$\begin{aligned} \text{minimize} \quad & -2x_1 + x_2 \\ \text{subject to} \quad & 2x_1 + 3x_2 \leq 5, \\ & x_1, x_2 \geq 0 \end{aligned}$$

in standard form using slack variables.

2.2 Free variables

Definition 2.3 (Variable splitting)**.** A free variable $x_j \in \mathbb{R}$ can be expressed as the difference of two nonnegative variables:

$$x_j = x_j^+ - x_j^-, \qquad x_j^+, x_j^- \geq 0.$$

Example 2.4 (Standard form with free variable)**.** Suppose we want to solve

$$\text{minimize} \quad -x \quad \text{subject to} \quad 2x = 1, \quad x \in \mathbb{R}.$$

Introduce $x^+, x^- \geq 0$ with $x = x^+ - x^-$. Then

$$\text{minimize} \quad -x^+ + x^- \quad \text{subject to} \quad 2x^+ - 2x^- = 1, \quad x^+, x^- \geq 0.$$

2.3 Absolute values

Definition 2.5 (Epigraph trick)**.** An absolute value constraint $|x| \leq 10$ can be modeled using inequalities:

$$-10 \leq x \leq 10.$$

More generally, to handle $\|x\|_1 = \sum_{i=1}^n |x_i|$ in the objective, introduce variables $t_i \geq 0$ and enforce

$$-t_i \leq x_i \leq t_i.$$

Then $\|x\|_1 = \mathbf{1}^T t$.

Example 2.6 (ℓ_1 minimization)**.** The problem

$$\min \|x\|_1 \quad \text{subject to } Ax = b$$

can be written in standard form as

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T t \\ \text{subject to} & Ax = b, \\ & -t \leq x \leq t, \\ & t \geq 0. \end{array}$$

Exercise. Show that the reformulation above is equivalent to the original ℓ_1 minimization problem.

2.4 Piecewise linear objectives

Definition 2.7 (Epigraph of maximum)**.** The problem $\min \max(x_1, \ldots, x_n)$ can be modeled by introducing an auxiliary variable t such that

$$x_i \le t \quad \text{for all } i.$$

Then the problem is equivalent to

$$\text{minimize } \ t \quad \text{subject to } \ x_i \le t \ \ \forall i.$$

Example 2.8 (Minimum of maximum)**.** Consider $\min \max(x_1, x_2)$ subject to $x_1 + x_2 = 1$, $x \ge 0$. Introduce t with $x_1 \le t$, $x_2 \le t$. The LP becomes

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & x_1 + x_2 = 1, \\ & x_1 \le t, \ x_2 \le t, \\ & x_1, x_2, t \ge 0. \end{array}$$

2.5 Assignment constraints

Definition 2.9 (Assignment)**.** In an assignment problem, each task must be assigned to exactly one agent. Let $X_{ij} \in \{0, 1\}$ be a binary variable indicating assignment of task i to agent j. Constraints:

$$\sum_j X_{ij} = 1 \ \ \forall i, \qquad \sum_i X_{ij} \le 1 \ \ \forall j.$$

Example 2.10 (Classroom assignment)**.** Suppose each class must be assigned to one room. The cost of assigning class i to room j is C_{ij}.

$$\begin{array}{ll} \text{minimize} & \sum_{i,j} C_{ij} X_{ij} \\ \text{subject to} & \sum_j X_{ij} = 1, \ \ \forall i, \\ & \sum_i X_{ij} \le 1, \ \ \forall j, \\ & X_{ij} \in \{0, 1\}. \end{array}$$

This is a mixed-integer linear program (MILP).

2.6 Logical constraints

Definition 2.11 (Big-M method)**.** A logical implication can often be modeled by introducing a large constant M. For example, to enforce "$Ax \le b$ if $y = 1$" where $y \in \{0, 1\}$, we can write

$$Ax \le b + (1 - y)M.$$

Example 2.12 (Capacity constraint with logic)**.** Suppose class i is assigned to room j with binary variable X_{ij}. Let p_i be enrollment of class i and c_j the capacity of room j. The constraint

$$p_i \leq c_j \quad \text{if } X_{ij} = 1$$

can be modeled as

$$p_i \leq c_j + (1 - X_{ij})M.$$

2.7 Flow constraints

Definition 2.13 (Flow conservation)**.** In a flow network with nodes V and edges E, variables f_{ij} denote flow on edge (i, j). Conservation at node k requires

$$\sum_{j:(k,j)\in E} f_{kj} - \sum_{i:(i,k)\in E} f_{ik} = s_k,$$

where s_k is supply (positive), demand (negative), or zero.

Example 2.14 (Minimum cost flow)**.** Find the cheapest way to ship one unit from source s to sink t:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{(i,j)\in E} c_{ij} f_{ij} \\ \text{subject to} \quad & \textstyle\sum_j f_{sj} - \sum_i f_{is} = 1, \\ & \textstyle\sum_j f_{tj} - \sum_i f_{it} = -1, \\ & \textstyle\sum_j f_{kj} - \sum_i f_{ik} = 0 \quad \forall k \notin \{s, t\}, \\ & f_{ij} \geq 0. \end{aligned}$$

2.8 Exercises

Exercise. Model the following constraints in standard form LP:

(a) $|x_1 - x_2| \leq 5$.

(b) $\max(x_1, x_2, x_3) \leq 2$.

(c) A task must be assigned to exactly one of three machines, each of which can handle at most one task.