

16 Linear Programming (November 29, December 1)

The maximum flow/minimum cut problem is a special case of a very general class of problems called *linear programming*. Many other optimization problems fall into this class, including minimum spanning trees and shortest paths, as well as several common problems in scheduling, logistics, and economics. Linear programming was used implicitly by Fourier in the early 1800s, but it was first formalized and applied to problems in economics in the 1930s by Leonid Kantorovich. Kantorovich's work was hidden behind the Iron Curtain (where it was largely ignored) and therefore unknown in the West. Linear programming was rediscovered and applied to shipping problems in the early 1940s by Tjalling Koopmans. The first complete algorithm to solve linear programming problems, called the *simplex method*, was published by George Dantzig in 1947. Koopmans first proposed the name "linear programming" in a discussion with Dantzig in 1948. Kantorovich and Koopmans shared the 1975 Nobel Prize in Economics "for their contributions to the theory of optimum allocation of resources". Dantzig did not; his work was apparently too pure. Koopmans wrote to Kantorovich suggesting that they refuse the prize in protest of Dantzig's exclusion, but Kantorovich saw the prize as a vindication of his use of mathematics in economics, which had been written off as "a means for apologists of capitalism".

A linear programming problem asks for a vector $x \in \mathbb{R}^d$ that maximizes (or equivalently, minimizes) a given linear function, among all vectors x that satisfy a given set of linear inequalities. The general form of a linear programming problem is the following:

$$\begin{aligned} & \text{maximize} && \sum_{j=1}^d c_j x_j \\ & \text{subject to} && \sum_{j=1}^d a_{ij} x_j \leq b_i \quad \text{for each } i = 1 \dots p \\ & && \sum_{j=1}^d a_{ij} x_j = b_i \quad \text{for each } i = p + 1 \dots p + q \\ & && \sum_{j=1}^d a_{ij} x_j \geq b_i \quad \text{for each } i = p + q + 1 \dots n \end{aligned}$$

Here, the input consists of a matrix $A = (a_{ij}) \in \mathbb{R}^{n \times d}$, a column vector $b \in \mathbb{R}^n$, and a row vector $c \in \mathbb{R}^d$. Each coordinate of the vector x is called a *variable*. Each of the linear inequalities is called a *constraint*. The function $x \mapsto x \cdot b$ is called the *objective function*. I will always use d to denote the number of variables, also known as the *dimension* of the problem. The number of constraints is usually denoted n .

A linear programming problem is said to be in *canonical form*¹ if it has the following structure:

$$\begin{aligned} & \text{maximize} && \sum_{j=1}^d c_j x_j \\ & \text{subject to} && \sum_{j=1}^d a_{ij} x_j \leq b_i \quad \text{for each } i = 1 \dots n \\ & && x_j \geq 0 \quad \text{for each } j = 1 \dots d \end{aligned}$$

¹Confusingly, some authors call this *standard form*.

We can express this canonical form more compactly as follows. For two vectors $x = (x_1, x_2, \dots, x_d)$ and $y = (y_1, y_2, \dots, y_d)$, the expression $x \geq y$ means that $x_i \geq y_i$ for every index i .

$$\begin{array}{ll} \max & c \cdot x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \end{array}$$

Any linear programming problem can be converted into canonical form as follows:

- For each variable x_j , add the equality constraint $x_j = x_j^+ - x_j^-$ and the inequalities $x_j^+ \geq 0$ and $x_j^- \geq 0$.
- Replace any equality constraint $\sum_j a_{ij}x_j = b_i$ with two inequality constraints $\sum_j a_{ij}x_j \geq b_i$ and $\sum_j a_{ij}x_j \leq b_i$.
- Replace any upper bound $\sum_j a_{ij}x_j \geq b_i$ with the equivalent lower bound $\sum_j -a_{ij}x_j \leq -b_i$.

This conversion potentially triples the number of variables and doubles the number of constraints; fortunately, it is almost never necessary in practice.

Another convenient formulation, especially for describing the simplex algorithm, is *slack form*², in which the only inequalities are of the form $x_j \geq 0$:

$$\begin{array}{ll} \max & c \cdot x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array}$$

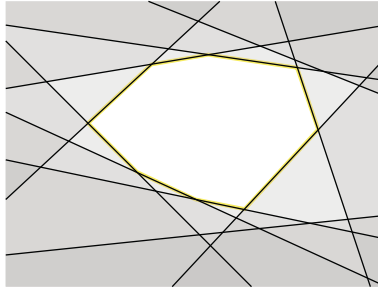
It's fairly easy to convert any linear programming problem into slack form. This form will be especially useful in describing the simplex algorithm.

16.1 The Geometry of Linear Programming

A point $x \in \mathbb{R}^d$ is *feasible* with respect to some linear programming problem if it satisfies all the linear constraints. The set of all feasible points is called the *feasible region* for that linear program. The feasible region has a particularly nice geometric structure that lends some useful intuition to later linear programming algorithms.

Any linear equation in d variables defines a *hyperplane* in \mathbb{R}^d ; think of a line when $d = 2$, or a plane when $d = 3$. This hyperplane divides \mathbb{R}^d into two *halfspaces*; each halfspace is the set of points that satisfy some linear inequality. Thus, the set of feasible points is the intersection of several hyperplanes (one for each equality constraint) and halfspaces (one for each inequality constraint). The intersection of a finite number of hyperplanes and halfspaces is called a *polyhedron*. It's not hard to verify that any halfspace, and therefore any polyhedron, is *convex*—if a polyhedron contains two points x and y , then it contains the entire line segment \overline{xy} .

²Confusingly, some authors call this *standard form*.



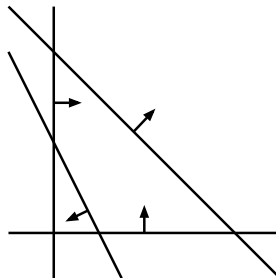
A two-dimensional polyhedron (white) defined by 10 linear inequalities.

By rotating \mathbb{R}^d so that the objective function points downward, we can express *any* linear programming problem in the following geometric form:

Find the lowest point in a given polyhedron.

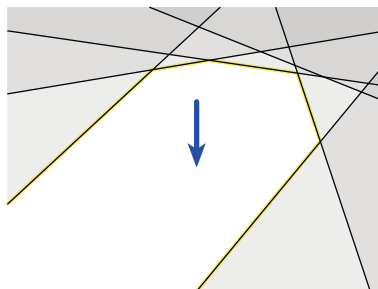
With this geometry in hand, we can easily picture two pathological cases where a given linear programming problem has no solution. The first possibility is that there are no feasible points; in this case the problem is called *infeasible*. For example, the following LP problem is infeasible:

$$\begin{aligned} & \text{maximize } x - y \\ & \text{subject to } 2x + y \leq 1 \\ & \quad \quad \quad x + y \geq 2 \\ & \quad \quad \quad x, y \geq 0 \end{aligned}$$



An infeasible linear programming problem; arrows indicate the constraints.

The second possibility is that there are feasible points at which the objective function is arbitrarily large; in this case, we call the problem *unbounded*. The same polyhedron could be unbounded for some objective functions but not others, or it could be unbounded for every objective function.



A two-dimensional polyhedron (white) that is unbounded downward but bounded upward.

16.2 Example 1: Shortest Paths

We can compute the length of the shortest path from s to t in a weighted directed graph by solving the following very simple linear programming problem.

$$\begin{aligned} & \text{maximize} && d_t \\ & \text{subject to} && d_s = 0 \\ & && d_v - d_u \leq \ell_{u \rightarrow v} \quad \text{for every edge } u \rightarrow v \end{aligned}$$

Here, $w_{u \rightarrow v}$ is the length of the edge $u \rightarrow v$. Each variable d_v represents a tentative shortest-path distance from s to v . The constraints mirror the requirement that every edge in the graph must be relaxed. These relaxation constraints imply that in any feasible solution, d_v is *at most* the shortest path distance from s to v . Thus, somewhat counterintuitively, we are correctly *maximizing* the objective function to compute the *shortest* path! In the optimal solution, the objective function d_t is the actual shortest-path distance from s to t , but for any vertex v that is not on the shortest path from s to t , d_v may be an underestimate of the true distance from s to v . However, we can obtain the true distances from s to every other vertex by modifying the objective function:

$$\begin{aligned} & \text{maximize} && \sum_v d_v \\ & \text{subject to} && d_s = 0 \\ & && d_v - d_u \leq \ell_{u \rightarrow v} \quad \text{for every edge } u \rightarrow v \end{aligned}$$

There is another formulation of shortest paths as an LP minimization problem using indicator variables.

$$\begin{aligned} & \text{minimize} && \sum_{u \rightarrow v} \ell_{u \rightarrow v} \cdot x_{u \rightarrow v} \\ & \text{subject to} && \sum_u x_{u \rightarrow s} - \sum_w x_{s \rightarrow w} = 1 \\ & && \sum_u x_{u \rightarrow t} - \sum_w x_{t \rightarrow w} = -1 \\ & && \sum_u x_{u \rightarrow v} - \sum_w x_{v \rightarrow w} = 0 \quad \text{for every vertex } v \neq s, t \\ & && x_{u \rightarrow v} \geq 0 \quad \text{for every edge } u \rightarrow v \end{aligned}$$

Intuitively, $x_{u \rightarrow v}$ equals 1 if $u \rightarrow v$ is in the shortest path from s to t , and equals 0 otherwise. The constraints merely state that the path should start at s , end at t , and either pass through or avoid every other vertex v . Any path from s to t —in particular, the shortest path—clearly implies a feasible point for this linear program, but there are other feasible solutions with non-integral values that do not represent paths.

Nevertheless, there is an optimal solution in which every x_e is either 0 or 1 and the edges e with $x_e = 1$ comprise the shortest path. Moreover, in any optimal solution, the objective function gives the shortest path distance, even if not every x_e is an integer!

16.3 Example 2: Maximum Flows and Minimum Cuts

Recall that the input to the maximum (s, t) -flow problem consists of a weighted directed graph $G = (V, E)$, two special vertices s and t , and a function assigning a non-negative *capacity* c_e to

each edge e . Our task is to choose the flow f_e across each edge e , as follows:

$$\begin{aligned} & \text{maximize} && \sum_w f_{s \rightarrow w} - \sum_u f_{u \rightarrow s} \\ & \text{subject to} && \sum_w f_{v \rightarrow w} - \sum_u f_{u \rightarrow v} = 0 && \text{for every vertex } v \neq s, t \\ & && f_{u \rightarrow v} \leq c_{u \rightarrow v} && \text{for every edge } u \rightarrow v \\ & && f_{u \rightarrow v} \geq 0 && \text{for every edge } u \rightarrow v \end{aligned}$$

Similarly, the minimum cut problem can be formulated using ‘indicator’ variables similarly to the shortest path problem. We have a variable S_v for each vertex v , indicating whether $v \in S$ or $v \in T$, and a variable $X_{u \rightarrow v}$ for each edge $u \rightarrow v$, indicating whether $u \in S$ and $v \in T$, where (S, T) is some (s, t) -cut.³

$$\begin{aligned} & \text{minimize} && \sum_{u \rightarrow v} c_{u \rightarrow v} \cdot X_{u \rightarrow v} \\ & \text{subject to} && X_{u \rightarrow v} + S_v - S_u \geq 0 && \text{for every edge } u \rightarrow v \\ & && X_{u \rightarrow v} \geq 0 && \text{for every edge } u \rightarrow v \\ & && S_s = 1 \\ & && S_t = 0 \end{aligned}$$

Like the minimization LP for shortest paths, there can be optimal solutions that assign fractional values to the variables. Nevertheless, the minimum value for the objective function is the cost of the minimum cut, and there is an optimal solution for which every variable is either 0 or 1, representing an actual minimum cut. No, this is not obvious; in particular, my claim is not a proof!

16.4 Linear Programming Duality

Each of these pairs of linear programming problems is related by a transformation called *duality*. For any linear programming problem, there is a corresponding dual linear program that can be obtained by a mechanical translation, essentially by swapping the constraints and the variables. The translation is simplest when the LP is in canonical form:

$$\begin{array}{ccc} \text{Primal (II)} & \iff & \text{Dual (II)} \\ \boxed{\begin{array}{l} \max \quad c \cdot x \\ \text{s.t. } Ax \leq b \\ x \geq 0 \end{array}} & & \boxed{\begin{array}{l} \min \quad y \cdot b \\ \text{s.t. } yA \geq c \\ y \geq 0 \end{array}} \end{array}$$

We can also write the dual linear program in exactly the same canonical form as the primal, by swapping the coefficient vector c and the objective vector b , negating both vectors, and replacing the constraint matrix A with its negative transpose.⁴

$$\begin{array}{ccc} \text{Primal (II)} & \iff & \text{Dual (II)} \\ \boxed{\begin{array}{l} \max \quad c \cdot x \\ \text{s.t. } Ax \leq b \\ x \geq 0 \end{array}} & & \boxed{\begin{array}{l} \max \quad -b^\top \cdot y^\top \\ \text{s.t. } -A^\top y^\top \leq -c \\ y^\top \geq 0 \end{array}} \end{array}$$

³These two linear programs are not quite *syntactic* duals; I’ve added two redundant variables S_s and S_t to the min-cut program increase readability.

⁴For the notational purists: In these formulations, x and b are column vectors, and y and c are row vectors. This is a somewhat nonstandard choice. Yes, that means the dot in $c \cdot x$ is redundant. Sue me.

Written in this form, it should be immediately clear that duality is an *involution*: The dual of the dual linear program Π is identical to the primal linear program Π . The choice of which LP to call the ‘primal’ and which to call the ‘dual’ is totally arbitrary.⁵

The Fundamental Theorem of Linear Programming. *A linear program Π has an optimal solution x^* if and only if the dual linear program Π has an optimal solution y^* where $c \cdot x^* = y^* Ax^* = y^* \cdot b$.*

The weak form of this theorem is trivial to prove.

Weak Duality Theorem. *If x is a feasible solution for a canonical linear program Π and y is a feasible solution for its dual Π , then $c \cdot x \leq yAx \leq y \cdot b$.*

Proof: Because x is feasible for Π , we have $Ax \leq b$. Since y is positive, we can multiply both sides of the inequality to obtain $yAx \leq y \cdot b$. Conversely, y is feasible for Π and x is positive, so $yAx \geq c \cdot x$. \square

It immediately follows that if $c \cdot x = y \cdot b$, then x and y are optimal solutions to their respective linear programs. This is in fact a fairly common way to prove that we have the optimal value for a linear program.

16.5 Duality Example

Before I prove the stronger duality theorem, let me first provide some intuition about where this duality thing comes from in the first place.⁶ Consider the following linear programming problem:

$$\begin{aligned} &\text{maximize} && 4x_1 + x_2 + 3x_3 \\ &\text{subject to} && x_1 + 4x_2 \leq 1 \\ &&& 3x_1 - x_2 + x_3 \leq 3 \\ &&& x_1, x_2, x_3 \geq 0 \end{aligned}$$

Let σ^* denote the optimum objective value for this LP. The feasible solution $x = (1, 0, 0)$ gives us a lower bound $\sigma^* \geq 4$. A different feasible solution $x = (0, 0, 3)$ gives us a better lower bound $\sigma^* \geq 9$. We could play this game all day, finding different feasible solutions and getting ever larger lower bounds. How do we know when we’re done? Is there a way to prove an *upper* bound on σ^* ?

In fact, there is. Let’s multiply each of the constraints in our LP by a new non-negative scalar value y_i :

$$\begin{aligned} &\text{maximize} && 4x_1 + x_2 + 3x_3 \\ &\text{subject to} && y_1(x_1 + 4x_2) \leq y_1 \\ &&& y_2(3x_1 - x_2 + x_3) \leq 3y_2 \\ &&& x_1, x_2, x_3 \geq 0 \end{aligned}$$

⁵For historical reasons, maximization LPs tend to be called ‘primal’ and minimization LPs tend to be called ‘dual’, which is really really stupid, since the only difference is a sign change.

⁶This example is taken from Robert Vanderbei’s excellent textbook *Linear Programming: Foundations and Extensions* [Springer, 2001], but the idea appears earlier in Jens Clausen’s 1997 paper ‘Teaching Duality in Linear Programming: The Multiplier Approach’.

Because each y_i is non-negative, we do not reverse any of the inequalities. Any feasible solution (x_1, x_2, x_3) must satisfy both of these inequalities, so it must also satisfy their sum:

$$(y_1 + 3y_2)x_1 + (4y_1 - y_2)x_2 + y_2x_3 \leq y_1 + 3y_2.$$

Now suppose that each y_i is larger than the i th coefficient of the objective function:

$$y_1 + 3y_2 \geq 4, \quad 4y_1 - y_2 \geq 1, \quad y_2 \geq 3.$$

This assumption lets us derive an upper bound on the objective value of *any* feasible solution:

$$4x_1 + x_2 + 3x_3 \leq (y_1 + 3y_2)x_1 + (4y_1 - y_2)x_2 + y_2x_3 \leq y_1 + 3y_2.$$

We have just proved that $\sigma^* \leq y_1 + 3y_2$.

Now it's natural to ask how tight we can make this upper bound. How small can we make the expression $y_1 + 3y_2$ without violating any of the inequalities we used to prove the upper bound? This is just another linear programming problem!

$$\begin{aligned} &\text{minimize} && y_1 + 3y_2 \\ &\text{subject to} && y_1 + 3y_2 \geq 4 \\ &&& 4y_1 - y_2 \geq 1 \\ &&& y_2 \geq 3 \\ &&& y_1, y_2 \geq 0 \end{aligned}$$

This is precisely the dual of our original linear program!

16.6 Strong Duality

The Fundamental Theorem can be rephrased in the following form:

Strong Duality Theorem. *If x^* is an optimal solution for a canonical linear program Π , then there is an optimal solution y^* for its dual Π , such that $c \cdot x^* = y^* A x^* = y^* \cdot b$.*

Proof (Sketch): I'll prove the theorem only for *non-degenerate* linear programs, in which (a) the optimal solution (if one exists) is a unique vertex of the feasible region, and (b) at most d constraint planes pass through any point. These non-degeneracy assumptions are relatively easy to enforce in practice and can be removed from the proof at the expense of some technical detail. I will also prove the theorem only for the case $n \geq d$; the argument for under-constrained LPs is similar (if not simpler).

Let x^* be the optimal solution for the linear program Π ; non-degeneracy implies that this solution is unique, and that exactly d of the n linear constraints are satisfied with equality. Without loss of generality (by permuting the rows of A), we can assume that these are the first d constraints.

So let A_\bullet be the $d \times d$ matrix containing the first d rows of A , and let A_\circ denote the other $n - d$ rows. Similarly, partition b into its first d coordinates b_\bullet and everything else b_\circ . Thus, we have partitioned the inequality $Ax^* \leq b$ into a system of equations $A_\bullet x^* = b_\bullet$ and a system of strict inequalities $A_\circ x^* < b_\circ$.

Now let $y^* = (y_\bullet^*, y_\circ^*)$ where $y_\bullet^* = cA_\bullet^{-1}$ and $y_\circ^* = 0$. We easily verify that $y^* \cdot b = c \cdot x^*$:

$$y^* \cdot b = y_\bullet^* \cdot b_\bullet = (cA_\bullet^{-1})b_\bullet = c(A_\bullet^{-1}b_\bullet) = c \cdot x^*.$$

Similarly, it's trivial to verify that $y^*A \geq c$:

$$y^*A = y_{\bullet}^*A_{\bullet} = c.$$

Once we prove that y^* is non-negative, and therefore feasible, the Weak Duality Theorem implies the result. Clearly $y_{\circ}^* \geq 0$. As we will see below, the inequality $y_{\bullet}^* \geq 0$ follows from the fact that x^* is optimal—we had to use that fact somewhere! This is the hardest part of the proof.

The key insight is to give a geometric interpretation to the vector $y_{\bullet}^* = cA_{\bullet}^{-1}$. Each row of the linear system $A_{\bullet}x^* = b_{\bullet}$ describes a hyperplane $a_i \cdot c^* = b_i$ in \mathbb{R}^d . The vector a_i is normal to this hyperplane and points *out* of the feasible region. The vectors a_1, \dots, a_d are linearly independent (by non-degeneracy) and thus describe a coordinate frame for the vector space \mathbb{R}^d . The definition of y_{\bullet}^* can be rewritten as follows:

$$c = y_{\bullet}^*A_{\bullet} = \sum_{i=1}^d y_i^*a_i.$$

We are expressing the objective vector c as a linear combination of the constraint normals a_1, \dots, a_d .

Now consider any vertex z of the feasible region that is adjacent to x^* . The vector $z - x^*$ is normal to all but one of the vectors a_i . Thus, we have

$$A_{\bullet}(z - x^*) = (0, \dots, \nu, \dots, 0)^{\top}$$

where the constant ν is in the i th coordinate. The vector $z - x^*$ points *into* the feasible region, so $\nu \leq 0$. It follows that

$$c \cdot (z - x^*) = y_{\bullet}^*A_{\bullet}(z - x^*) = \nu y_i^*.$$

The optimality of x^* implies that $c \cdot x^* \geq c \cdot z$, so we must have $y_i^* \geq 0$. We're done! \square