The greatest flood has the soonest ebb; the sorest tempest the most sudden calm; the hottest love the coldest end; and from the deepest desire oftentimes ensues the deadliest hate.

— Socrates

Th' extremes of glory and of shame, Like east and west, become the same.

— Samuel Butler, *Hudibras* Part II, Canto I (c. 1670)

Everything is dual; everything has poles; everything has its pair of opposites; like and unlike are the same; opposites are identical in nature, but different in degree; extremes meet; all truths are but half-truths; all paradoxes may be reconciled.

— *The Kybalion: A Study of The Hermetic Philosophy of Ancient Egypt and Greece* (1908)

Oh, that!

— John von Neumann to George Dantzig (May 1947)

CHAPTER **H**

Linear Programming

[Read Chapters F and G first.]

H.1 Introduction

The maximum flow and minimum cut problems are examples of a 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 and Jacobi in the early 1800s, but it was first formalized and applied to problems in economics in the 1930s by Leonid Kantorovich. Kantorivich'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 late 1930s 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 his Soviet colleagues had written off as "a means for apologists of capitalism".

A linear programming problem—or more simply, a ***linear program***—asks for a vector $x \in \mathbb{R}^d$ that maximizes or 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:

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

Here, the input consists of a *constraint matrix* $A = (a_{ij}) \in \mathbb{R}^{n \times d}$, an *offset vector* $b \in \mathbb{R}^n$, and an *objective 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 $x \mapsto c \cdot x$ is called the *objective function*.

A linear program is said to be in ***canonical form***[1] if it has the following structure:

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

A canonical linear program is completely determined by its constraint matrix A, its offset vector b, and its objective vector c. In addition to the n matrix constraints $Ax \ge b$, every canonical linear program includes d *sign constraints* of the form $x_j \ge 0$.

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

[1] Confusingly, some authors call this *standard form*. Linear programs in this form are also sometimes called *packing* LPs.

for every index i.

$$\begin{array}{rl} \max & c \cdot x \\ \text{s.t.} & Ax \le b \\ & x \ge 0 \end{array}$$

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

- For each variable x_j, add two new variables x_i^+ and x_i^-, an equality constraint $x_j = x_j^+ - x_j^-$, and inequalities $x_j^+ \ge 0$ and $x_j^- \ge 0$.
- Replace any equality constraint $\sum_j a_{ij}x_j = b_i$ with two inequality constraints $\sum_j a_{ij}x_j \ge b_i$ and $\sum_j a_{ij}x_j \le b_i$.
- Finally, replace any upper bound constraint $\sum_j a_{ij}x_j \ge b_i$ with the equivalent lower bound $\sum_j -a_{ij}x_j \le -b_i$.

This conversion potentially doubles the number of variables and the number of constraints; fortunately, it is rarely necessary in practice.

Another useful format for linear programs is ***slack form***[2], in which every inequality is of the form $x_j \ge 0$:

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

It's fairly easy to convert any linear programming problem into slack form; I'll leave the details as an easy exercise (hint, hint). Slack form is especially useful in executing the simplex algorithm, which we'll see in the next chapter.

H.2 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 the linear programming algorithms we'll see later.

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 $\overline{xy}$.

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

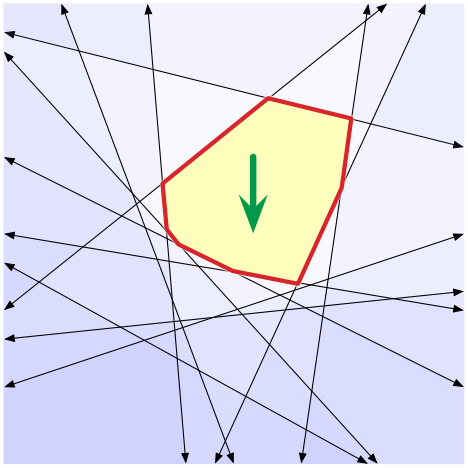


Figure H.1. A two-dimensional polyhedron defined by 11 linear inequalities.

By rotating $\mathbb{R}^d$ (or choosing an appropriate coordinate frame) 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***.

maximize	$x - y$
subject to	$2x + y \leq 1$
	$x + y \geq 2$
	$x, y \geq 0$

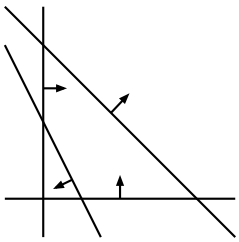


Figure H.2. An infeasible linear programming problem; arrows indicate the constraints.

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.

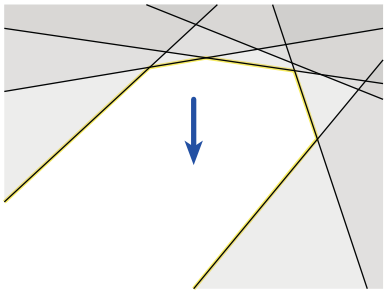


Figure H.3. A two-dimensional polyhedron (white) that is unbounded downward but bounded upward.

H.3 Examples

One Shortest Path

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

$$\begin{aligned} \text{maximize} \quad & dist(t) \\ \text{subject to} \quad & dist(s) = 0 \\ & dist(v) - dist(u) \le \ell(u \to v) \quad \text{for every edge } u \to v \end{aligned}$$

Here the input consists of a graph $G = (V, E)$ and a vector $\ell \in \mathbb{R}^E$ (or equivalently, a function $\ell : E \to \mathbb{R}$) representing the lengths of edges in G, and our goal is to compute a vector $dist \in \mathbb{R}^V$ (or equivalently, a function $dist : V \to \mathbb{R}$) such that $dist(t)$ is the shortest path distance from s to t. The constraints in the linear program encode Ford's generic requirement that every edge in the graph must be relaxed. These relaxation constraints imply that in any feasible solution, $dist(v)$ is *at most* the shortest path distance from s to v, for *every* vertex v. Thus, somewhat counterintuitively, we are correctly *maximizing* $dist(t)$ to compute the *shortest*-path distance from s to t!

- This linear program is feasible if and only if the graph has no negative cycles. On the one hand, if the graph has no negative cycles, then the true shortest-path distances give a feasible solution. On the other hand, if the graph has a negative cycle, then for any distance values $dist(v)$, at least one edge in that cycle will be tense.
- This linear program is bounded if and only if the graph contains a path from s to t. On the one hand, if there is a path from s to t, the length of that path is an upper bound on the optimal objective value. On the other hand, if there is no path from s to t, we can obtain a feasible solution by setting $dist(v) = 0$ for all vertices v that are reachable from s, and $dist(v) = \infty$ for all vertices v that are not reachable from s.
- The optimal solution to this linear program is not necessarily unique. For example, if the graph contains a vertex v that cannot reach t, we can transform any optimal solution into another optimal solution by arbitrarily decreasing $dist(v)$. More

generally, this LP has a unique optimal solution if and only if (1) every vertex lies on a path from s to t and (2) every path from s to t has the same length.

One Shortest Path, Take Two

Alternatively, and perhaps more intuitively, we can formulate the shortest-path problem as a minimization problem. However, instead of using variables that represent distances, this linear program uses variables $x(u{\to}v)$ that *intuitively* indicate which edges $u{\to}v$ lie on the shortest path from s to t.

$$\begin{aligned}
\text{minimize} \quad & \sum_{u\to v} \ell(u{\to}v) \cdot x(u{\to}v) \\
\text{subject to} \quad & \sum_{u} x(u{\to}t) - \sum_{w} x(t{\to}w) = 1 \\
& \sum_{u} x(u{\to}v) - \sum_{w} x(v{\to}w) = 0 \quad \text{for every vertex } v \neq s, t \\
& x(u{\to}v) \geq 0 \quad \text{for every edge } u{\to}v
\end{aligned}$$

Any path from s to t—in particular, the *shortest* from s to t path—yields a feasible point x for this linear program, where $x(u{\to}v) = 1$ for each edge $u{\to}v$ in the path and $x(u{\to}v) = 0$ for every other edge. More generally, any *walk* from s to t implies a feasible solution, where $x(u{\to}v)$ is the number of times $u{\to}v$ appears in the walk.

Sharp-eyed readers might recognize this linear program as a *minimum-cost flow* problem, where each edge $u{\to}v$ has cost $\ell(u{\to}v)$ and infinite capacity, the source vertex s has balance -1 (one unit of supply), the target vertex t has balance 1 (one unit of demand), and all other vertices have balance 0. (The missing balance constraint for s is implied by the other $V-1$ balance constraints.) More concisely, the feasible points for this linear program are flows from s to t with value 1, and the objective function $\ell \cdot x$ is the *cost* of this flow.

Because all balances and (nonexistent) capacities in this network are integers, we know that there is an integral minimum-cost flow from s to t. (The existence of an integral optimal solution a feature of this particular linear program! It is possible for a linear program with only integral input data A, b, and c to have only non-integral optimal solutions.) A simple flow decomposition argument now implies that the minimum-cost flow from s to t is actually a simple path from s to t.

- This linear program is feasible if and only if the underlying graph contains a path from s to t. If G contains a path from s to t, then sending one unit of flow along that path gives us a feasible solution. On the other hand, any (s,t)-flow is a weighted sum of (s,t)-paths and possibly cycles.
- This linear program is bounded if and only if the underlying graph has no negative cycles. If the G contains a negative cycle, we can arbitrarily decrease the cost of any solution by pushing more flow around that cycle. On the other hand, suppose G has

no negative cycles, and let x be any feasible solution. If x contains any cycles, we can decrease the cost of x by removing them. Then the maxflow-mincut theorem implies that $x(u \to v) \le 1$ for every edge $u \to v$. We conclude that the cost of any feasible solution is at least the sum of all negative edge lengths in G.

- The optimal solution to this linear program is not necessarily unique, because shortest paths are not necessarily unique. Moreover, if there is more than one optimal solution, then there are infinitely many optimal solutions. Specifically, if x and x' are optimal solutions, then for any constant $0 < \alpha < 1$, the convex combination $\alpha x + (1-\alpha)x'$ is also an optimal solution.

Single-Source Shortest Paths

In the optimal solution for the first shortest-path linear program, the objective function $dist(t)$ 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, the corresponding value $dist(v)$ may underestimate the true distance from s to v. We can obtain the true distances from s to *every* other vertex by modifying the objective function as follows:

$$\begin{aligned} \text{maximize} \quad & \sum_v dist(v) \\ \text{subject to} \quad & dist(s) = 0 \\ & dist(v) - dist(u) \le \ell(u \to v) \quad \text{for every edge } u \to v \end{aligned}$$

Again, we are counterintuitively faced with a *maximization* problem to find *shortest* paths.

We can similarly modify the second shortest-path linear program to compute a spanning tree of shortest paths.

$$\begin{aligned} \text{minimize} \quad & \sum_{u \to v} \ell(u \to v) \cdot x(u \to v) \\ \text{subject to} \quad & \sum_u x(u \to v) - \sum_w x(v \to w) = 1 \quad \text{for every vertex } v \ne s \\ & x(u \to v) \ge 0 \quad \text{for every edge } u \to v \end{aligned}$$

Let T be any shortest-path tree rooted at s, and for any vertex v, let D_v denote set of all descendants of v in T. (In particular, $D_s = V$, and $D_v = |v|$ if v is a leaf of T.) Then we can obtain an optimal solution to this second LP by setting

$$x(u \to v) = \begin{cases} |D_v| & \text{if } u \to v \in T \\ 0 & \text{otherwise} \end{cases}$$

Again, sharp-eyed readers may recognize this linear program as an uncapacitated minimum-cost flow problem.

Maximum Flows and Minimum Cuts

Recall that the input to the maximum (s,t)-flow problem consists of a 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 compute a flow function $f \in E^{\mathbb{R}}$ subject to the usual capacity and balance constraints. The maximum-flow problem can be encoded as a linear program as follows:

$$\begin{aligned}
\text{maximize} \quad & \sum_w f(s{\to}w) - \sum_u f(u{\to}s) \\
\text{subject to} \quad & \sum_w f(v{\to}w) - \sum_u f(u{\to}v) = 0 && \text{for every vertex } v \neq s,t \\
& f(u{\to}v) \le c(u{\to}v) && \text{for every edge } u{\to}v \\
& f(u{\to}v) \ge 0 && \text{for every edge } u{\to}v
\end{aligned}$$

Similarly, the minimum cut problem can be formulated using "indicator" variables similarly to the shortest-path problem. We define a variable $S(v)$ for each vertex v, indicating whether $v \in S$ or $v \in T$, and a variable $x(u{\to}v)$ for each edge $u{\to}v$, indicating whether $u \in S$ and $v \in T$, where (S,T) is some (s,t)-cut.

$$\begin{aligned}
\text{minimize} \quad & \sum_{u\to v} c(u{\to}v) \cdot x(u{\to}v) \\
\text{subject to} \quad & x(u{\to}v) + S(v) - S(u) \ge 0 && \text{for every edge } u{\to}v \\
& x(u{\to}v) \ge 0 && \text{for every edge } u{\to}v \\
& S(s) = 1 \\
& S(t) = 0
\end{aligned}$$

Like the minimization linear programs 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.

H.4 Linear Programming Duality

Each of the preceding pairs of linear programs is an example of a general relationship 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 original LP is in canonical form:

Primal (Π)		**Dual (Ⅱ)**
$\max \quad c \cdot x$	$\Longleftrightarrow$	$\min \quad y \cdot b$
$\text{s.t.} \quad Ax \le b$		$\text{s.t.} \quad yA \ge c$
$x \ge 0$		$y \ge 0$

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.[3]

Primal (Π)

$$\begin{array}{rl} \max & c \cdot x \\ \text{s.t.} & Ax \le b \\ & x \ge 0 \end{array}$$

$$\Longleftrightarrow$$

Dual (Ш)

$$\begin{array}{rl} \max & -b^\top \cdot y^\top \\ \text{s.t.} & -A^\top y^\top \le -c^\top \\ & y^\top \ge 0 \end{array}$$

Written in this form, it should be immediately clear that duality is an *involution*: The dual of the dual linear program Ш is identical to the primal linear program Π. The choice of which linear program to call the "primal" and which to call the "dual" is totally arbitrary.[4]

Dualizing Arbitrary LPs

Given a linear program Π that is not in canonical form, we can mechanically derive its dual linear program by first converting Π into canonical form and then applying the pattern shown above. If we push the symbols around long enough, we eventually find the following general pattern for constructing the dual of any ***maximization*** linear program. (The rules for dualizing minimization programs are the reverse of these.)

- For each constraint in the primal LP *except* sign constraints of the form $x_i \gtrless 0$, there is a corresponding variable in the dual LP.
  - If the primal constraint is an upper bound, the dual variable must be non-negative.
  - If the primal constraint is a lower bound, the dual variable must be non-positive.
  - If the primal constraint is an equation, the sign of dual variable is not constrained.
  - The right side of the primal constraint becomes the dual variable's coefficient in the objective function.
  - The dual linear program *minimizes* its objective value.
- Conversely, for each variable in the primal LP, there is a corresponding constraint in the dual LP.
  - If the primal variable is non-negative, the dual constraint is a lower bound.
  - If the primal variable is non-positive, the dual constraint is an upper bound.
  - If the sign of the primal variable is not constrained, the dual constraint is an equation.

[3] For 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.

[4] For historical reasons, maximization linear programs tend to be called "primal" and minimization linear programs tend to be called "dual". (Or is it the other way around?) This habit is nothing but a pointless and distracting religious tradition.

- Finally, if the primal variable is equal to zero, the dual constraint doesn't actually exist. (If something can't vary, it's not really a variable!)
- The primal variable's coefficient in the objective function becomes the right side of the dual constraint.

Primal	Dual	Primal	Dual
$\max c \cdot x$	$\min y \cdot b$	$\min c \cdot x$	$\max y \cdot b$
$\sum_j a_{ij}x_j \le b_i$	$y_i \ge 0$	$\sum_j a_{ij}x_j \le b_i$	$y_i \le 0$
$\sum_j a_{ij}x_j \ge b_i$	$y_i \le 0$	$\sum_j a_{ij}x_j \ge b_i$	$y_i \ge 0$
$\sum_j a_{ij}x_j = b_i$	–	$\sum_j a_{ij}x_j = b_i$	–
$x_j \ge 0$	$\sum_i y_i a_{ij} \ge c_j$	$x_j \le 0$	$\sum_i y_i a_{ij} \ge c_j$
$x_j \le 0$	$\sum_i y_i a_{ij} \le c_j$	$x_j \ge 0$	$\sum_i y_i a_{ij} \le c_j$
–	$\sum_i y_i a_{ij} = c_j$	–	$\sum_i y_i a_{ij} = c_j$
$x_j = 0$	–	$x_j = 0$	–

Figure H.4. Constructing the dual of an arbitrary linear program.

At first glance, these rules appear to be asymmetric: The inequality directions for primal constraints and dual variables are opposites, but the inequality directions for primal variables and dual constraints are identical. In fact, this asymmetry reflects the switch from a primal *maximization* LP to a dual *minimization* LP. The dual of the dual of any linear program is always (equivalent to) the original linear program.

Shortest Path Example

As an example, let's derive the dual of our very first linear program for shortest paths:

$$\begin{aligned} &\text{maximize} && dist(t) \\ &\text{subject to} && dist(s) = 0 \\ &&& dist(v) - dist(u) \le \ell(u \to v) \quad \text{for every edge } u \to v \end{aligned}$$

This linear program has a variable $dist(v)$ for every vertex v and a constraint for every edge. Thus, the underlying constraint matrix A has a row (constraint) for every edge and a column (variable) for every vertex. Specifically, we have

$$A(u \to v, w) = \begin{cases} -1 & \text{if } u = w, \\ +1 & \text{if } v = w, \\ 0 & \text{otherwise.} \end{cases}$$

(A is the *signed incidence matrix* of the underlying directed graph G.) The coefficients of the objective vector are $+1$ for t and 0 for every other vertex, and the coefficients of the offset vector are the edge lengths.

So, the dual linear program has a constraint for every vertex and a variable for every edge. Let's call the edge variables $x(u{\to}v)$, because why not? Because $dist(s) = 0$, there is no dual constraint corresponding to $dist(s)$ after all. Because the other primal variables $dist(v)$ are not sign-constrained, the dual constraints are all equations. The primal constraints are upper bounds, so the dual variables $x(u{\to}v)$ must be non-negative. Finally, the coefficients of the dual objective vector are the edge lengths, and the coefficients of the dual offset vector are $+1$ for t and 0 for every other vertex.

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

Hey, this looks familiar!

H.5 The Fundamental Theorem

The most important structural observation about linear programming is the following broad generalization of the maxflow-mincut theorem.

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

The Fundamental Theorem (in a slightly different form) was first conjectured by John von Neumann in May 1947, during a fateful meeting with George Dantzig. Dantzig met with von Neumann to explain his general theory of linear programming and his simplex algorithm (which we'll see in the next chapter). According to Dantzig, after about half an hour into his presentation, von Neumann told him "Get to the point!" When Dantzig then explained his algorithm more concisely, von Neumann replied dismissively "Oh, *that!*" and proceeded to lecture Dantzig for over an hour on convexity, fixed points, and his theory of two-player games, while Dantzig sat (as he later put it) dumbfounded. During his lecture, von Neumann defined the negative-transpose dual of an arbitrary linear program—based on a similar duality in two-player zero-sum games that trivially results from swapping the players—and conjectured correctly that the Fundamental Theorem could be derived from his famous min-max theorem, which he proved in 1928:

Min-Max Theorem. *For any matrix* A*, we have* $\min_x \max_y xAy = \max_y \min_x xAy$.

Von Neumann's min-max theorem is itself a consequence of several equivalent "Theorems of the Alternative" proved by Paul Gordan, Hermann Minkowski, Gyula Farkas, Erich Stiemke, Theodore Motzkin, and others in the late 1800s and early 1900s. The earliest example of such a theorem (that I know of) was proved by Gordan in 1873.

Gordan's Theorem of the Alternative. *For any matrix A, either $Ax = 0$ and $x \geq 0$ for some non-zero vector x, or $yA > 0$ for some vector y, but not both.*

Soon after their 1947 meeting, von Neumann and Dantzig independently wrote up proofs of the Fundamental Theorem, which they each distributed privately for many years. Unfortunately, von Neumann's proof was flawed, although his errors were finally corrected when he finally published the proof in 1963, and Dantzig never published his proof. The first *published* proof was written by Albert Tucker, David Gale, and Harold Kuhn in 1951.

I will defer a proof of the full Fundamental Theorem until after we develop some more intuition, but the following weak form 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 $\amalg$, 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 non-negative, we can multiply both sides of the inequality to obtain $yAx \leq y \cdot b$. Conversely, y is feasible for $\amalg$ and x is non-negative, so $yAx \geq c \cdot x$. □

The Weak Duality Theorem has two important immediate consequences:

- Let x and y be arbitrary *feasible* solutions to a canonical linear program and its dual, respectively. If $c \cdot x = y \cdot b$, then x and y are *optimal* primal and dual solutions, respectively, and $c \cdot x = yAx = y \cdot b$.
- If a linear program is unbounded, then its dual is infeasible; equivalently, if a linear program is feasible, then its dual is bounded. (It is possible for a linear program and its dual to both be infeasible, but only if both linear programs are degenerate.)

The full version of the Fundamental Theorem implies that that all these implications are actually equivalences.

H.6 Another Duality Example

Before I prove the stronger duality theorem, let me try to provide some intuition about how this duality thing actually works.[5] Consider the following canonical linear

[5]This 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".

programming problem:

$$\begin{array}{lrcl} \text{maximize} & 4x_1 + & x_2 + 3x_3 & \\ \text{subject to} & x_1 + 4x_2 & & \le 2 \\ & 3x_1 - & x_2 + x_3 & \le 4 \\ & & x_1, x_2, x_3 & \ge 0 \end{array}$$

Let σ^* denote the optimum objective value for this LP. The feasible solution $x = (1, 0, 0)$ gives us a lower bound $\sigma^* \ge 4$. A different feasible solution $x = (0, 0, 3)$ gives us a better lower bound $\sigma^* \ge 9$. 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:

$$\begin{array}{lrcl} \text{maximize} & 4x_1 + & x_2 + 3x_3 & \\ \text{subject to} & y_1(\ x_1 + 4x_2 & &) \le 2y_1 \\ & y_2(3x_1 - & x_2 + x_3) & \le 4y_2 \\ & & x_1, x_2, x_3 & \ge 0 \end{array}$$

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:

$$(y_1 + 3y_2)x_1 + (4y_1 - y_2)x_2 + y_2x_3 \le 2y_1 + 4y_2.$$

Now suppose that the coefficient of each variable x_i in the expression on the left side of this inequality is larger than the corresponding coefficient of the objective function:

$$y_1 + 3y_2 \ge 4, \qquad 4y_1 - y_2 \ge 1, \qquad y_2 \ge 3.$$

This assumption implies an upper bound on the objective value of *any* feasible solution:

$$4x_1 + x_2 + 3x_3 \ \le\ (y_1 + 3y_2)x_1 + (4y_1 - y_2)x_2 + y_2x_3 \ \le\ 2y_1 + 4y_2. \qquad (*)$$

In particular, by plugging in the optimal solution (x_1^*, x_2^*, x_3^*) for the original LP, we obtain the following upper bound on σ^*:

$$\sigma^* \ =\ 4x_1^* + x_2^* + 3x_3^* \ \le\ 2y_1 + 4y_2.$$

Now it is natural to ask how tight this upper bound can be. That is, how small can we make the expression $2y_1 + 4y_2$ without violating any of the inequalities we used to

prove the upper bound? This is just another linear programming problem.

$$\begin{aligned} \text{minimize} \quad & 2y_1 + 4y_2 \\ \text{subject to} \quad & y_1 + 3y_2 \geq 4 \\ & 4y_1 - y_2 \geq 1 \\ & y_2 \geq 3 \\ & y_1, y_2 \geq 0 \end{aligned}$$

In fact, the resulting linear program is precisely the dual of our original linear program! Moreover, inequality $(*)$ is just an instantiation of the Weak Duality Theorem.

H.7 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 $\amalg$, 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 objective vector is not normal to any constraint hyperplane, and (b) at most d constraint hyperplanes pass through any point. The first assumption implies that the optimal solution to the LP—if it exists—is unique and is therefore a vertex of the feasible region. 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 assume that $n \geq d$; the argument for under-constrained LPs is similar (if not simpler).

To develop some intuition, let's first consider the *very* special case where $x^* = (0, 0, \ldots, 0)$. Let e_i denote the ith standard basis vector, whose ith coordinate is 1 and all other coordinates are 0. Because $x_i^* = 0$ for all i, our non-degeneracy assumption implies the *strict* inequality $a_i \cdot x^* < b_i$ for all i. Thus, any sufficiently small ball around the origin does not intersect any other constraint hyperplane $a_i \cdot x = b_i$. Thus, for all i, and for any sufficiently small $\delta > 0$, the vector δe_i is feasible. Because x^* is the unique optimum, we must have $\delta c_i = c \cdot (\delta e_i) < c \cdot x^* = 0$. We conclude that $c_i < 0$ for all i.

Now let $y = (0, 0, \ldots, 0)$ as well. We immediately observe that $yA \geq c$ and $y \geq 0$; in other words, y is a *feasible* solution for the dual linear program $\amalg$. But $y \cdot b = 0 = c \cdot x^*$, so the weak duality theorem implies that y is an *optimal* solution to $\amalg$, and the proof is complete for this very special case!

Now let's consider the more general case. Let x^* be the optimal solution for the linear program Π; our non-degeneracy assumption implies that this solution is unique, and that it satisfies exactly d of the n linear constraints with equality. Without loss of

generality (by permuting the constraints and possibly changing coordinates), we can assume that these are the first d constraints. Thus, we have

$$\begin{aligned} a_i \cdot x^* &= b_i \qquad \text{for all } i \le d, \\ a_i \cdot x^* &< b_i \qquad \text{for all } i \ge d+1, \end{aligned}$$

where a_i denotes the ith row of A. Let $A_\bullet$ denote the $d \times d$ matrix containing the first d rows of A. Our non-degeneracy assumption implies that $A_\bullet$ has full rank, and thus has a well-defined inverse $V = A_\bullet^{-1}$.

Now define a vector $y \in \mathbb{R}^n$ by setting

$$\begin{aligned} y_j &:= c \cdot v^j \qquad \text{for all } j \le d, \\ y_j &:= 0 \qquad\quad\ \text{for all } j \ge d+1, \end{aligned}$$

where v^j denotes the jth column of V. The definition of matrix inverse implies that $a_i \cdot v^i = 1$ for all i, and $a_i \cdot v^j = 0$ for all $i \neq j$.

To simplify notation, let $y_\bullet = (y_1, y_2, \ldots, y_d)$ and let $b_\bullet = (b_1, b_2, \ldots, b_d) = A_\bullet x^*$. Because $y_i = 0$ for all $i \ge d+1$, we immediately have

$$y \cdot b = y_\bullet \cdot b_\bullet = cV b_\bullet = cA_\bullet^{-1} b_\bullet = c \cdot x^*$$

and

$$yA = y_\bullet A_\bullet = cVA_\bullet = cA_\bullet^{-1} A_\bullet = c.$$

The point x^* lies on exactly d constraint hyperplanes; moreover, any sufficiently small ball around x^* intersects *only* those d constraint hyperplanes. Consider the point $\tilde{x} = x^* - \varepsilon v^j$, for some index $1 \le j \le d$ and some sufficiently small $\varepsilon > 0$. We have $a_i \cdot \tilde{x} = a_i \cdot x^* - \varepsilon(a_i \cdot v^j) = b_i$ for all $i \neq j$, and $a_j \cdot \tilde{x} = a_j \cdot x^* - \varepsilon(a_j \cdot v^j) = b_j - \varepsilon < b_j$. Thus, $\tilde{x}$ is a feasible point for Π. Because x^* is the *unique* optimum for Π, we must have $c \cdot \tilde{x} = c \cdot x^* - \varepsilon(c \cdot v^j) < c \cdot x^*$. We conclude that $y_j = c \cdot v^j > 0$ for all j.

We have shown that $yA \ge c$ and $y \ge 0$, so y is a *feasible* solution for the dual linear program Π. We have also shown that $y \cdot b = c \cdot x^*$, so the Weak Duality Theorem implies that y is an *optimal* solution for Π, and the proof is complete! □

We can also give a useful geometric interpretation to the dual vector $y_\bullet \in \mathbb{R}^d$. Each linear equation $a_i \cdot x = b_i$ defines a hyperplane in $\mathbb{R}^d$ with normal vector a_i. The normal vectors $a_1, \ldots, a_d$ are linearly independent (by non-degeneracy) and therefore describe a coordinate frame for the vector space $\mathbb{R}^d$. The definition of $y_\bullet$ implies that $c = y_\bullet A_\bullet = \sum_{i=1}^d y_i a_i$. In other words, ***the non-zero dual variables $y_1, \ldots, y_d$ are the coefficients of the objective vector c in the coordinate frame $a_1, \ldots, a_d$.***

In more physical terms, imagine that the objective vector c points downward, and that x^* is the lowest point in the feasible region, which is found by tossing a marble into the feasible region and letting it fall as far as possible without crossing any constraint hyperplane. Even after the marble reaches the lowest feasible point, gravity is still

pulling it downward, but the marble isn't moving, so the constraint hyperplanes must be pushing it upward. Each constraint hyperplane can only push in the direction of its normal vector. ***Each dual variable y_i is the amount of force applied by the ith constraint hyperplane.***

H.8 Complementary Slackness

The Strong Duality Theorem implies a relationship between the optimal solutions of a linear program and its dual that is stronger than just the equality of their objective values. In physical terms, a constraint hyperplane applies a non-zero force to the marble only if it touches the marble; that is, if a dual variable is positive, then the corresponding constraint must be satisfied with equality.

Recall that the theorem states that if x^* is the optimal solution to a canonical linear program Π, then the dual program $\amalg$ has an optimal solution y^* such that $c \cdot x^* = y^* A x^* = y^* \cdot b$. These equations have two immediate consequences for any optimal solution vectors x^* and y^*.

- For any row index i, either $y_i^* = 0$ or $a_i \cdot x^* = b$ (or both).
- For any column index j, either $x_j^* = 0$ or $y^* \cdot a^j = c$ (or both).

Here, a_i and a^j respectively denote the ith row and jth column of A. These consequences of strong duality are called the ***complementary slackness*** conditions.

Call a constraint *tight* if it is satisfied with equality and *loose* otherwise. The complementary slackness conditions can be phrased more colloquially as follows:

- If a primal variable is positive, the corresponding dual constraint is tight.
- If a dual constraint is loose, the corresponding primal variable is zero.
- If a dual variable is positive, the corresponding primal constraint is tight.
- If a primal constraint is loose, the corresponding dual variable is zero.

Complementary Slackness Theorem. *Let x be an arbitrary feasible solution to a canonical linear program Π, and let y be an arbitrary feasible solution to the dual linear program $\amalg$. The following conditions are equivalent:*

- *x and y are optimal solutions to their respective linear programs.*
- $c \cdot x = y \cdot b$
- $c \cdot x = yAx$
- $yAx = y \cdot b$
- *x and y satisfy all complementary slackness conditions.*

The complementary slackness conditions are not necessarily exclusive; it is possible for a constraint to be tight *and* for the corresponding dual variable to be zero. Nevertheless, every bounded and feasible linear program has optimal primal and dual solutions x^* and y^* that satisfy *strict* complementary slackness conditions:

- For any row index i, either $y_i^* > 0$ or $a_i \cdot x^* < b$ (but not both).
- For any column index j, either $x_j^* > 0$ or $y^* \cdot a^j > c$ (but not both).

However, if the linear program is non-degenerate, the primal and dual optimal solutions are both unique and therefore satisfy these stricter conditions.

Exercises

Need more!

★★★

1. A matrix $A = (a_{ij})$ is *skew-symmetric* if and only if $a_{ji} = -a_{ij}$ for all indices $i \neq j$; in particular, every skew-symmetric matrix is square. A canonical linear program $\max\{c \cdot x \mid Ax \leq b; x \geq 0\}$ is *self-dual* if the matrix A is skew-symmetric and the objective vector c is *equal* to the constraint vector b.
   (a) Prove that any self-dual linear program Π is syntactically equivalent to its dual program $\amalg$.
   (b) Show that any linear program Π with d variables and n constraints can be transformed into a self-dual linear program with $n + d$ variables and $n + d$ constraints. The optimal solution to the self-dual program should include both the optimal solution for Π (in d of the variables) and the optimal solution for the dual program $\amalg$ (in the other n variables).

2. (a) Give a linear-programming formulation of the ***bipartite maximum matching*** problem. The input is a bipartite graph $G = (U \cup V; E)$, where $E \subseteq U \times V$; the output is the largest matching in G. Your linear program should have one variable for each edge.
   (b) Now dualize the linear program from part (a). What do the dual variables represent? What does the objective function represent? What problem is this!?

3. (a) Give a linear-programming formulation of the ***minimum-cost circulation*** problem. You are given a flow network whose edges have both capacities and costs, and your goal is to find a feasible circulation (flow with value 0) whose total cost is as small as possible.
   (b) Derive the dual of your linear program from part (a).

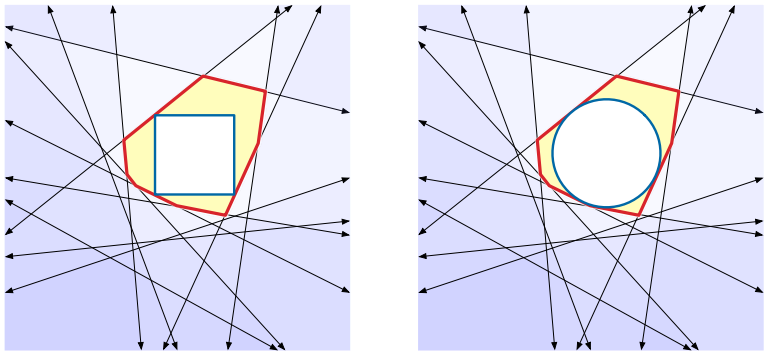
4. Suppose we are given a sequence of n linear inequalities of the form $a_i x + b_i y \leq c_i$. Collectively, these n inequalities describe a convex polygon P in the plane.
   (a) Describe a linear program whose solution describes the largest axis-aligned square that lies entirely inside P.
   (b) Describe a linear program whose solution describes the maximum-perimeter axis-aligned rectangle that lies entirely inside P.

(c) Describe a linear program whose solution describes the largest circle that lies entirely inside P.

(d) Describe a polynomial-time algorithm to compute two interior-disjoint axis-aligned squares with maximum total perimeter that lie entirely inside P. *[Hint: There are exactly two interesting cases to consider; for each case, formulate a corresponding linear program.]*

(e) Describe a polynomial-time algorithm to compute two interior-disjoint axis-aligned rectangles with maximum total perimeter that lie entirely inside P. *[Hint: Again, there are only two interesting cases to consider.]*

In all subproblems, "axis-aligned" means that the edges of the square(s) or rectangles(s) are horizontal and vertical.



5. Given points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ in the plane, the ***linear regression problem*** asks for real numbers a and b such that the line $y = ax + b$ fits the points as closely as possible, according to some criterion. The most common fit criterion is minimizing the ***L_2 error***, defined as follows:[6]

$$\varepsilon_2(a, b) = \sum_{i=1}^{n} (y_i - ax_i - b)^2.$$

But there are many other ways of measuring a line's fit to a set of points, some of which can be optimized via linear programming.

(a) The ***L_1 error*** (or *total absolute deviation*) of the line $y = ax + b$ is defined as follows:

$$\varepsilon_1(a, b) = \sum_{i=1}^{n} \left| y_i - ax_i - b \right|.$$

Describe a linear program whose solution (a, b) describes the line with minimum L_1 error.

[6] This measure is also known as *sum of squared residuals*, and the algorithm to compute the best fit is normally called *(ordinary/linear) least squares*.

(b) The ***L_∞ error*** (or *maximum absolute deviation*) of the line $y = ax + b$ is defined as follows:

$$\varepsilon_\infty(a, b) = \max_{i=1}^{n} \left| y_i - ax_i - b \right|.$$

Describe a linear program whose solution (a, b) describes the line with minimum L_∞ error.

6. Let $G = (V, E)$ be an arbitrary directed graph with weighted vertices; vertex weights may be positive, negative, or zero. A ***prefix*** of G is a subset $P \subseteq V$, such that there is no edge $u \to v$ where $u \notin P$ but $v \in P$. A ***suffix*** of G is the complement of a prefix. Finally, an ***interval*** of G is the intersection of a prefix of G and a suffix of G. The weight of a prefix, suffix, or interval is the sum of the weights of its vertices.

   (a) Describe a linear program whose solution describes the maximum-weight prefix of G. Your linear program should have one variable per vertex, indicating whether that vertex is or is not in the chosen prefix.

   (b) Describe an algorithm that computes the maximum-weight prefix of G, by reducing to a standard maximum-flow problem in a related graph. *[Hint: Consider the special case where G is a dag. Haven't we seen this problem before?]*

   (c) Describe an efficient algorithm that computes the maximum-weight prefix of G, when G is a rooted tree with all edges pointing away from the root. *[Hint: This is really easy if you* **don't** *think about linear programming.]*

   (d) Describe a linear program whose solution describes the maximum-weight interval of G.

   (e) Describe an efficient algorithm that computes the maximum-weight interval of G, when G is a rooted tree with all edges pointing away from the root. *[Hint: Again, this is easy if you* **don't** *think about linear programming.]*

   [Hint: Don't worry about the solutions to your linear programs being integral; they will be (essentially by part (b)). If all vertex weights are negative, the maximum-weight interval is empty; if all vertex weights are positive, the maximum-weight interval contains every vertex.]

7. Suppose you are given an arbitrary directed graph $G = (V, E)$ with arbitrary edge weights $\ell : E \to \mathbb{R}$, and two special vertices. Each edge in G is colored either red, white, or blue to indicate how you are permitted to modify its weight:

   - You may increase, but not decrease, the length of any red edge.
   - You may decrease, but not increase, the length of any blue edge.
   - You may not change the length of any black edge.

   Your task is to modify the edge weights—subject to the color constraints—so that *every* path from s to t has exactly the same length. Both the given weights and the

new weights of the edges can be positive, negative, or zero. To keep the following problems simple, assume every edge in G lies on at least one path from s to t, and that G has no isolated vertices.

(a) Describe a linear program that is feasible if and only if it is possible to make every path from s to t the same length. *[Hint: Let* $\text{dist}(v)$ *denote the length of every path from* s *to* v*.]*

(b) Construct the dual of the linear program from part (a). *[Hint: Choose a convenient objective function for your primal LP.]*

♥(c) Give a self-contained description of the combinatorial problem encoded by the dual linear program from part (b), and prove *directly* that it is equivalent to the original path-equalization problem. Do not use the words "linear", "program", or "dual". Yes, you've seen this problem before. *[Hint: The proof is the hard part.]*

(d) Describe and analyze an algorithm to determine *in* $O(EV)$ *time* whether it is possible to make every path from s to t the same length. Do not use the words "linear", "program", or "dual".

(e) Finally, suppose we want to equalize path lengths while modifying the edge weights as little as possible. Specifically, we want to compute new edge weights $\ell'(e)$ that give every path from s to t the same length, such that the sum $\sum_e |\ell'(e) - \ell(e)|$ is as small as possible. Describe and analyze an efficient algorithm to solve this optimization problem. *[Hint: Modify your solutions to parts (a)–(c).]*

♦8. An ***integer program*** is a linear program with the additional constraint that the variables must take only integer values.

(a) Prove that deciding whether an integer program has a feasible solution is NP-complete.

(b) Prove that finding the optimal feasible solution to an integer program is NP-hard.

[Hint: Almost any NP-hard decision problem can be formulated as an integer program. Pick your favorite.]

9. A *convex combination* of n points $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ is weighted average of those points:

$$p = \sum_{i=1}^{n} \lambda_i x_i$$

for some coefficients $\lambda_1, \lambda_2, \ldots, \lambda_n$ such that $\lambda_i \geq 0$ for all i and $\sum_{i=1}^{n} \lambda_i = 1$. The *convex hull* of a set of points X is the set of all convex combinations of points in X. For example, the convex hull of two points is the line segment between them, the convex hull of three points is the (filled) triangle with those three vertices, and so on.

Carathéodory's theorem states that for any set of points $X \subset \mathbb{R}^d$, every point in the convex hull of X lies in the convex hull of $d + 1$ points of X. For example, when

$d = 2$, the theorem states every point inside a convex polygon P is also inside the triangle spanned by three vertices of P.

Prove Carathéodory's theorem. *[Hint: Express "p lies in the convex hull of X" as a system of linear inequalities over the variables λ_i.]*

10. *Helly's theorem* states that for any collection of convex bodies in $\mathbb{R}^d$, if every $d + 1$ of them intersect, then there is a point lying in the intersection of all of them. Prove Helly's theorem for the special case where the convex bodies are halfspaces. Equivalently, show that if a system of linear inequalities $Ax \le b$ is infeasible, then we can select a subset of $d + 1$ inequalities such that the resulting subsystem infeasible. *[Hint: Construct a linear program from the system by imposing a 0 objective vector, and consider the dual LP.]*