CHAPTER 13

Linear Programming: The Simplex Method

Dantzig's development of the simplex method in the late 1940s marks the start of the modern era in optimization. This method made it possible for economists to formulate large models and analyze them in a systematic and efficient way. Dantzig's discovery coincided with the development of the first electronic computers, and the simplex method became one of the earliest important applications of this new and revolutionary technology. From those days to the present, computer implementations of the simplex method have been continually improved and refined. They have benefited particularly from interactions with numerical analysis, a branch of mathematics that also came into its own with the appearance of electronic computers, and have now reached a high level of sophistication.

Today, linear programming and the simplex method continue to hold sway as the most widely used of all optimization tools. Since 1950, generations of workers in management, economics, finance, and engineering have been trained in the techniques of formulating linear models and solving them with simplex-based software. Often, the situations they model are actually nonlinear, but linear programming is appealing because of the advanced state of the software, guaranteed convergence to a global minimum, and the fact that uncertainty in the model makes a linear model more appropriate than an overly complex nonlinear model. Nonlinear programming may replace linear programming as the method of choice in some applications as the nonlinear software improves, and a new class of methods known as interior-point methods (see Chapter 14) has proved to be faster for some linear programming problems, but the continued importance of the simplex method is assured for the foreseeable future.

LINEAR PROGRAMMING

Linear programs have a linear objective function and linear constraints, which may include both equalities and inequalities. The feasible set is a polytope, a convex, connected set with flat, polygonal faces. The contours of the objective function are planar. Figure 13.1 depicts a linear program in two-dimensional space, in which the contours of the objective function are indicated by dotted lines. The solution in this case is unique—a single vertex. A simple reorientation of the polytope or the objective gradient c could however make the solution non-unique; the optimal value $c^T x$ could take on the same value over an entire edge. In higher dimensions, the set of optimal points can be a single vertex, an edge or face, or even the entire feasible set. The problem has no solution if the feasible set is empty (the *infeasible* case) or if the objective function is unbounded below on the feasible region (the *unbounded* case).

Linear programs are usually stated and analyzed in the following *standard form*:

$$\min c^T x, \quad \text{subject to } Ax = b, x \geq 0, \tag{13.1}$$

where c and x are vectors in $\mathbb{R}^n$, b is a vector in $\mathbb{R}^m$, and A is an $m \times n$ matrix. Simple devices can be used to transform any linear program to this form. For instance, given the problem

$$\min c^T x, \quad \text{subject to } Ax \leq b$$

(without any bounds on x), we can convert the inequality constraints to equalities by introducing a vector of *slack variables* z and writing

$$\min c^T x, \quad \text{subject to } Ax + z = b, z \geq 0. \tag{13.2}$$

This form is still not quite standard, since not all the variables are constrained to be

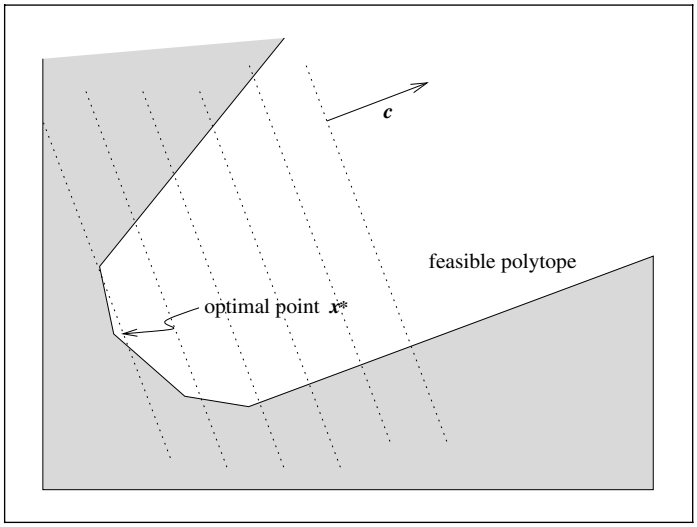

Figure 13.1 A linear program in two dimensions with solution at x^*.

nonnegative. We deal with this by *splitting* x into its nonnegative and nonpositive parts, $x = x^+ - x^-$, where $x^+ = \max(x, 0) \geq 0$ and $x^- = \max(-x, 0) \geq 0$. The problem (13.2) can now be written as

$$\min \begin{bmatrix} c \\ -c \\ 0 \end{bmatrix}^T \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix}, \quad \text{s.t.} \quad \begin{bmatrix} A & -A & I \end{bmatrix} \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix} = b, \quad \begin{bmatrix} x^+ \\ x^- \\ z \end{bmatrix} \geq 0,$$

which clearly has the same form as (13.1).

Inequality constraints of the form $x \leq u$ or $Ax \geq b$ always can be converted to equality constraints by adding or subtracting *slack variables* to make up the difference between the left- and right-hand sides. Hence,

$$x \leq u \Leftrightarrow x + w = u, \ w \geq 0,$$
$$Ax \geq b \Leftrightarrow Ax - y = b, \ y \geq 0.$$

(When we subtract the variables from the left hand side, as in the second case, they are sometimes known as *surplus variables.*) We can also convert a "maximize" objective max $c^T x$ into the "minimize" form of (13.1) by simply negating c to obtain: min $(-c)^T x$.

We say that the linear program (13.1) is *infeasible* if the feasible set is empty. We say that the problem (13.1) is *unbounded* if the objective function is unbounded below on the feasible region, that is, there is a sequence of points x^k feasible for (13.1) such that $c^T x^k \downarrow -\infty$. (Of course, unbounded problems have no solution).

Many linear programs arise from models of transshipment and distribution networks. These problems have additional structure in their constraints; special-purpose simplex algorithms that exploit this structure are highly efficient. We do not discuss such problems further in this book, except to note that the subject is important and complex, and that a number of fine texts on the topic are available (see, for example, Ahuja, Magnanti, and Orlin [1]).

For the standard formulation (13.1), we will assume throughout that $m < n$. Otherwise, the system $Ax = b$ contains redundant rows, or is infeasible, or defines a unique point. When $m \geq n$, factorizations such as the QR or LU factorization (see Appendix A) can be used to transform the system $Ax = b$ to one with a coefficient matrix of full row rank.

13.1 OPTIMALITY AND DUALITY

OPTIMALITY CONDITIONS

Optimality conditions for the problem (13.1) can be derived from the theory of Chapter 12. Only the first-order conditions—the Karush–Kuhn–Tucker (KKT) conditions—are needed. Convexity of the problem ensures that these conditions are sufficient for a global minimum. We do not need to refer to the second-order conditions from Chapter 12, which are not informative in any case because the Hessian of the Lagrangian for (13.1) is zero.

The theory we developed in Chapter 12 make derivation of optimality and duality results for linear programming much easier than in other treatments, where this theory is developed more or less from scratch.

The KKT conditions follow from Theorem 12.1. As stated in Chapter 12, this theorem requires linear independence of the active constraint gradients (LICQ). However, as we noted in Section 12.6, the result continues to hold for *dependent* constraints provided they are linear, as is the case here.

We partition the Lagrange multipliers for the problem (13.1) into two vectors λ and s, where $\lambda \in \mathbb{R}^m$ is the multiplier vector for the equality constraints $Ax = b$, while $s \in \mathbb{R}^n$ is the multiplier vector for the bound constraints $x \geq 0$. Using the definition (12.33), we can write the Lagrangian function for (13.1) as

$$\mathcal{L}(x, \lambda, s) = c^T x - \lambda^T (Ax - b) - s^T x. \tag{13.3}$$

Applying Theorem 12.1, we find that the first-order necessary conditions for x^* to be a solution of (13.1) are that there exist vectors λ and s such that

$$A^T \lambda + s = c, \tag{13.4a}$$
$$Ax = b, \tag{13.4b}$$
$$x \geq 0, \tag{13.4c}$$
$$s \geq 0, \tag{13.4d}$$
$$x_i s_i = 0, \quad i = 1, 2, \ldots, n. \tag{13.4e}$$

The complementarity condition (13.4e), which essentially says that at least one of the components x_i and s_i must be zero for each $i = 1, 2, \ldots, n$, is often written in the alternative form $x^T s = 0$. Because of the nonnegativity conditions (13.4c), (13.4d), the two forms are identical.

Let (x^*, λ^*, s^*) denote a vector triple that satisfies (13.4). By combining the three equalities (13.4a), (13.4d), and (13.4e), we find that

$$c^T x^* = (A^T \lambda^* + s^*)^T x^* = (Ax^*)^T \lambda^* = b^T \lambda^*. \tag{13.5}$$

As we shall see in a moment, $b^T \lambda$ is the objective function for the dual problem to (13.1), so (13.5) indicates that the primal and dual objectives are equal for vector triples (x, λ, s) that satisfy (13.4).

It is easy to show directly that the conditions (13.4) are *sufficient* for x^* to be a global solution of (13.1). Let $\bar{x}$ be any other feasible point, so that $A\bar{x} = b$ and $\bar{x} \geq 0$. Then

$$c^T \bar{x} = (A\lambda^* + s^*)^T \bar{x} = b^T \lambda^* + \bar{x}^T s^* \geq b^T \lambda^* = c^T x^*. \tag{13.6}$$

We have used (13.4) and (13.5) here; the inequality relation follows trivially from $\bar{x} \geq 0$ and $s^* \geq 0$. The inequality (13.6) tells us that no other feasible point can have a lower objective value than $c^T x^*$. We can say more: The feasible point $\bar{x}$ is optimal if and only if

$$\bar{x}^T s^* = 0,$$

since otherwise the inequality in (13.6) is strict. In other words, when $s_i^* > 0$, then we must have $\bar{x}_i = 0$ for *all* solutions $\bar{x}$ of (13.1).

THE DUAL PROBLEM

Given the data c, b, and A, which defines the problem (13.1), we can define another, closely related, problem as follows:

$$\max\ b^T \lambda, \quad \text{subject to } A^T \lambda \leq c. \tag{13.7}$$

This problem is called the *dual problem* for (13.1). In contrast, (13.1) is often referred to as the *primal*. We can restate (13.7) in a slightly different form by introducing a vector of *dual slack variables* s, and writing

$$\max\ b^T \lambda, \quad \text{subject to } A^T \lambda + s = c, \quad s \geq 0. \tag{13.8}$$

The variables (λ, s) in this problem are sometimes jointly referred to collectively as *dual variables*.

The primal and dual problems present two different viewpoints on the same data. Their close relationship becomes evident when we write down the KKT conditions for (13.7). Let us first restate (13.7) in the form

$$\min\ -b^T\lambda \ \text{ subject to } c - A^T\lambda \geq 0,$$

to fit the formulation (12.1) from Chapter 12. By using $x \in \mathbb{R}^n$ to denote the Lagrange multipliers for the constraints $A^T\lambda \leq c$, we see that the Lagrangian function is

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

Using Theorem 12.1 again, we find the first-order necessary conditions for λ to be optimal for (13.7) to be that there exists x such that

$$Ax = b, \tag{13.9a}$$
$$A^T\lambda \leq c, \tag{13.9b}$$
$$x \geq 0, \tag{13.9c}$$
$$x_i(c - A^T\lambda)_i = 0, \quad i = 1, 2, \ldots, n. \tag{13.9d}$$

Defining $s = c - A^T\lambda$ (as in (13.8)), we find that the conditions (13.9) and (13.4) are identical! The optimal Lagrange multipliers λ in the primal problem are the optimal variables in the dual problem, while the optimal Lagrange multipliers x in the dual problem are the optimal variables in the primal problem.

Analogously to (13.6), we can show that (13.9) are in fact *sufficient* conditions for a solution of the dual problem (13.7). Given x^* and λ^* satisfying these conditions (so that the triple $(x, \lambda, s) = (x^*, \lambda^*, c - A^T\lambda^*)$ satisfies (13.4)), we have for any other dual feasible point $\bar{\lambda}$ (with $A^T\bar{\lambda} \leq c$) that

$$\begin{aligned} b^T\bar{\lambda} &= (x^*)^T A^T\bar{\lambda} \\ &= (x^*)^T(A^T\bar{\lambda} - c) + c^T x^* \\ &\leq c^T x^* && \text{because } A^T\bar{\lambda} - c \leq 0 \text{ and } x^* \geq 0 \\ &= b^T\lambda^* && \text{from (13.5).} \end{aligned}$$

Hence λ^* achieves the maximum of the dual objective $b^T\lambda$ over the dual feasible region $A^T\lambda \leq c$, so it solves the dual problem (13.7).

The primal–dual relationship is symmetric; by taking the dual of the dual problem (13.7), we recover the primal problem (13.1). We leave the proof of this claim as an exercise.

Given a feasible vector x for the primal (satisfying $Ax = b$ and $x \geq 0$) and a feasible point (λ, s) for the dual (satisfying $A^T\lambda + s = c$, $s \geq 0$), we have as in (13.6) that

$$c^T x - b^T\lambda = (c - A^T\lambda)^T x = s^T x \geq 0. \tag{13.10}$$

Therefore we have $c^T x \geq b^T \lambda$ (that is, the dual objective is a lower bound on the primal objective) when both the primal and dual variables are feasible—a result known as *weak duality*.

The following *strong duality* result is fundamental to the theory of linear programming.

Theorem 13.1 (Strong Duality).

(i) *If either problem (13.1) or (13.7) has a (finite) solution, then so does the other, and the objective values are equal.*

(ii) *If either problem (13.1) or (13.7) is unbounded, then the other problem is infeasible.*

PROOF. For (i), suppose that (13.1) has a finite optimal solution x^*. It follows from Theorem 12.1 that there are vectors λ^* and s^* such that (x^*, λ^*, s^*) satisfies (13.4). We noted above that (13.4) and (13.9) are equivalent, and that (13.9) are sufficient conditions for λ^* to be a solution of the dual problem (13.7). Moreover, it follows from (13.5) that $c^T x^* = b^T \lambda^*$, as claimed.

A symmetric argument holds if we start by assuming that the dual problem (13.7) has a solution.

To prove (ii), suppose that the primal is unbounded, that is, there is a sequence of points x^k, $k = 1, 2, 3, \ldots$ such that

$$c^T x^k \downarrow -\infty, \quad Ax^k = b, \quad x^k \geq 0.$$

Suppose too that the dual (13.7) is feasible, that is, there exists a vector $\bar{\lambda}$ such that $A^T \bar{\lambda} \leq c$. From the latter inequality together with $x^k \geq 0$, we have that $\bar{\lambda}^T A x^k \leq c^T x^k$, and therefore

$$\bar{\lambda}^T b = \bar{\lambda}^T A x^k \leq c^T x^k \downarrow -\infty,$$

yielding a contradiction. Hence, the dual must be infeasible.

A similar argument can be used to show that unboundedness of the dual implies infeasibility of the primal. □

As we showed in the discussion following Theorem 12.1, the multiplier values λ and s for (13.1) indicate the sensitivity of the optimal objective value to perturbations in the constraints. In fact, the process of finding (λ, s) for a given optimal x is often called *sensitivity analysis*. Considering the case of perturbations to the vector b (the right-hand side in (13.1) and objective gradient in (13.7)), we can make an informal argument to illustrate the sensitivity. Suppose that this small change produces small perturbations in the primal and dual solutions, and that the vectors Δs and Δx have zeros in the same locations as s and x, respectively. Since x and s are complementary (see (13.4e)) it follows that

$$0 = x^T s = x^T \Delta s = (\Delta x)^T s = (\Delta x)^T \Delta s.$$

We have from Theorem 13.1 that the optimal objectives of the primal and dual problems are equal, for both the original and perturbed problems, so

$$c^T x = b^T \lambda, \qquad c^T (x + \Delta x) = (b + \Delta b)^T (\lambda + \Delta \lambda).$$

Moreover, by feasibility of the perturbed solutions in the perturbed problems, we have

$$A(x + \Delta x) = b + \Delta b, \qquad A^T \Delta \lambda = -\Delta s.$$

Hence, the change in optimal objective due to the perturbation is as follows:

$$\begin{aligned} c^T \Delta x &= (b + \Delta b)^T (\lambda + \Delta \lambda) - b^T \lambda \\ &= (b + \Delta b)^T \Delta \lambda + (\Delta b)^T \lambda \\ &= (x + \Delta x)^T A^T \Delta \lambda + (\Delta b)^T \lambda \\ &= (x + \Delta x)^T \Delta s + (\Delta b)^T \lambda \\ &= (\Delta b)^T \lambda. \end{aligned}$$

In particular, if $\Delta b = \epsilon e_j$, where e_j is the jth unit vector in $\mathbb{R}^m$, we have for all ϵ sufficiently small that

$$c^T \Delta x = \epsilon \lambda_j. \tag{13.11}$$

That is, the change in optimal objective is λ_j times the size of the perturbation to b_j, if the perturbation is small.

13.2 GEOMETRY OF THE FEASIBLE SET

BASES AND BASIC FEASIBLE POINTS

We assume for the remainder of the chapter that

$$\text{The matrix } A \text{ in (13.1) has full row rank.} \tag{13.12}$$

In practice, a preprocessing phase is applied to the user-supplied data to remove some redundancies from the given constraints and eliminate some of the variables. Reformulation by adding slack, surplus, and artificial variables can also result in A satisfying the property (13.12) .

Each iterate generated by the simplex method is a *basic feasible point* of (13.1). A vector x is a basic feasible point if it is feasible and if there exists a subset $\mathcal{B}$ of the index set $\{1, 2, \ldots, n\}$ such that

- $\mathcal{B}$ contains exactly m indices;
- $i \notin \mathcal{B} \;\Rightarrow\; x_i = 0$ (that is, the bound $x_i \geq 0$ can be inactive only if $i \in \mathcal{B}$);
- The $m \times m$ matrix B defined by

$$B = [A_i]_{i \in \mathcal{B}} \tag{13.13}$$

is nonsingular, where A_i is the ith column of A.

A set $\mathcal{B}$ satisfying these properties is called a *basis* for the problem (13.1). The corresponding matrix B is called the *basis matrix*.

The simplex method's strategy of examining only basic feasible points will converge to a solution of (13.1) only if

(a) the problem *has* basic feasible points; and

(b) at least one such point is a *basic optimal point*, that is, a solution of (13.1) that is also a basic feasible point.

Happily, both (a) and (b) are true under reasonable assumptions, as the following result (sometimes known as the fundamental theorem of linear programming) shows.

Theorem 13.2.

(i) *If (13.1) has a nonempty feasible region, then there is at least one basic feasible point;*

(ii) *If (13.1) has solutions, then at least one such solution is a basic optimal point.*

(iii) *If (13.1) is feasible and bounded, then it has an optimal solution.*

PROOF. Among all feasible vectors x, choose one with the minimal number of nonzero components, and denote this number by p. Without loss of generality, assume that the nonzeros are $x_1, x_2, \ldots, x_p$, so we have

$$\sum_{i=1}^{p} A_i x_i = b.$$

Suppose first that the columns $A_1, A_2, \ldots, A_p$ are linearly dependent. Then we can express one of them (A_p, say) in terms of the others, and write

$$A_p = \sum_{i=1}^{p-1} A_i z_i, \tag{13.14}$$

for some scalars $z_1, z_2, \ldots, z_{p-1}$. It is easy to check that the vector

$$x(\epsilon) = x + \epsilon(z_1, z_2, \ldots, z_{p-1}, -1, 0, 0, \ldots, 0)^T = x + \epsilon z \tag{13.15}$$

satisfies $Ax(\epsilon) = b$ for any scalar ϵ. In addition, since $x_i > 0$ for $i = 1, 2, \ldots, p$, we also have $x_i(\epsilon) > 0$ for the same indices $i = 1, 2, \ldots, p$ and all ϵ sufficiently small in magnitude. However, there is a value $\bar{\epsilon} \in (0, x_p]$ such that $x_i(\bar{\epsilon}) = 0$ for some $i = 1, 2, \ldots, p$. Hence, $x(\bar{\epsilon})$ is feasible and has at most $p - 1$ nonzero components, contradicting our choice of p as the minimal number of nonzeros.

Therefore, columns $A_1, A_2, \ldots, A_p$ must be linearly independent, and so $p \leq m$. If $p = m$, we are done, since then x is a basic feasible point and $\mathcal{B}$ is simply $\{1, 2, \ldots, m\}$. Otherwise $p < m$ and, because A has full row rank, we can choose $m - p$ columns from among $A_{p+1}, A_{p+2}, \ldots, A_n$ to build up a set of m linearly independent vectors. We construct $\mathcal{B}$ by adding the corresponding indices to $\{1, 2, \ldots, p\}$. The proof of (i) is complete.

The proof of (ii) is quite similar. Let x^* be a solution with a minimal number of nonzero components p, and assume again that $x_1^*, x_2^*, \ldots, x_p^*$ are the nonzeros. If the columns $A_1, A_2, \ldots, A_p$ are linearly dependent, we define

$$x^*(\epsilon) = x^* + \epsilon z,$$

where z is chosen exactly as in (13.14), (13.15). It is easy to check that $x^*(\epsilon)$ will be feasible for all ϵ sufficiently small, both positive and negative. Hence, since x^* is optimal, we must have

$$c^T(x^* + \epsilon z) \geq c^T x^* \quad \Rightarrow \quad \epsilon c^T z \geq 0$$

for all ϵ sufficiently small (positive and negative). Therefore, $c^T z = 0$ and so $c^T x^*(\epsilon) = c^T x^*$ for all ϵ. The same logic as in the proof of (i) can be applied to find $\bar{\epsilon} > 0$ such that $x^*(\bar{\epsilon})$ is feasible and optimal, with at most $p - 1$ nonzero components. This contradicts our choice of p as the minimal number of nonzeros, so the columns $A_1, A_2, \ldots, A_p$ must be linearly independent. We can now apply the same reasoning as above to conclude that x^* is already a basic feasible point and therefore a basic optimal point.

The final statement (iii) is a consequence of finite termination of the simplex method. We comment on the latter property in the next section. □

The terminology we use here is not quite standard, as the following table shows:

our terminology	**terminology used elsewhere**
basic feasible point	basic feasible solution
basic optimal point	optimal basic feasible solution

The standard terms arose because "solution" and "feasible solution" were originally used as synonyms for "feasible point." However, as the discipline of optimization developed,

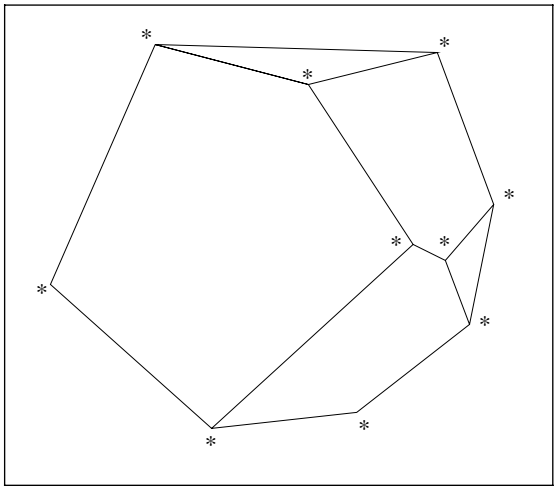

Figure 13.2
Vertices of a three-dimensional polytope (indicated by $*$).

the word "solution" took on a more specific and intuitive meaning (as in "solution to the problem"). We maintain consistency with the rest of the book by following this more modern usage.

VERTICES OF THE FEASIBLE POLYTOPE

The feasible set defined by the linear constraints is a polytope, and the *vertices* of this polytope are the points that do not lie on a straight line between two other points in the set. Geometrically, they are easily recognizable; see Figure 13.2. Algebraically, the vertices are exactly the basic feasible points defined above. We therefore have an important relationship between the algebraic and geometric viewpoints and a useful aid to understanding how the simplex method works.

Theorem 13.3.

All basic feasible points for (13.1) are vertices of the feasible polytope $\{x \mid Ax = b, x \geq 0\}$, *and vice versa.*

PROOF. Let x be a basic feasible point and assume without loss of generality that $\mathcal{B} = \{1, 2, \ldots, m\}$. The matrix $B = [A_i]_{i=1,2,\ldots,m}$ is therefore nonsingular, and

$$x_{m+1} = x_{m+2} = \cdots = x_n = 0. \tag{13.16}$$

Suppose that x lies on a straight line between two other feasible points y and z. Then we can find $\alpha \in (0, 1)$ such that $x = \alpha y + (1 - \alpha)z$. Because of (13.16) and the fact that α and $1 - \alpha$ are both positive, we must have $y_i = z_i = 0$ for $i = m + 1, m + 2, \ldots, n$. Writing $x_{\mathrm{B}} = (x_1, x_2, \ldots, x_m)^T$ and defining y_{B} and z_{B} likewise, we have from $Ax = Ay = Az = b$

that

$$Bx_{\mathcal{B}} = By_{\mathcal{B}} = Bz_{\mathcal{B}} = b,$$

and so, by nonsingularity of B, we have $x_{\mathcal{B}} = y_{\mathcal{B}} = z_{\mathcal{B}}$. Therefore, $x = y = z$, contradicting our assertion that y and z are two feasible points *other* than x. Therefore, x is a vertex.

Conversely, let x be a vertex of the feasible polytope, and suppose that the nonzero components of x are $x_1, x_2, \ldots, x_p$. If the corresponding columns $A_1, A_2, \ldots, A_p$ are linearly dependent, then we can construct the vector $x(\epsilon) = x + \epsilon z$ as in (13.15). Since $x(\epsilon)$ is feasible for all ϵ with sufficiently small magnitude, we can define $\hat{\epsilon} > 0$ such that $x(\hat{\epsilon})$ and $x(-\hat{\epsilon})$ are both feasible. Since $x = x(0)$ obviously lies on a straight line between these two points, it cannot be a vertex. Hence our assertion that $A_1, A_2, \ldots, A_p$ are linearly dependent must be incorrect, so these columns must be linearly independent and $p \leq m$. If $p < m$, and since A has full row rank, we can add $m - p$ indices to $\{1, 2, \ldots, p\}$ to form a basis $\mathcal{B}$, for which x is the corresponding basic feasible point. This completes our proof. □

We conclude this discussion of the geometry of the feasible set with a definition of *degeneracy*. This term has a variety of meanings in optimization, as we discuss in Chapter 16. For the purposes of this chapter, we use the following definition.

Definition 13.1 (Degeneracy).

A basis $\mathcal{B}$ is said to be degenerate *if $x_i = 0$ for some $i \in \mathcal{B}$, where x is the basic feasible solution corresponding to $\mathcal{B}$. A linear program (13.1) is said to be* degenerate *if it has at least one degenerate basis.*

13.3 THE SIMPLEX METHOD

OUTLINE

In this section we give a detailed description of the simplex method for (13.1). There are actually a number of variants the simplex method; the one described here is sometimes known as the *revised simplex method.* (We will describe an alternative known as the *dual simplex method* in Section 13.6.)

As we described above, all iterates of the simplex method are basic feasible points for (13.1) and therefore vertices of the feasible polytope. Most steps consist of a move from one vertex to an adjacent one for which the basis $\mathcal{B}$ differs in exactly one component. On most steps (but not all), the value of the primal objective function $c^T x$ is decreased. Another type of step occurs when the problem is unbounded: The step is an edge along which the objective function is reduced, and along which we can move infinitely far without ever reaching a vertex.

The major issue at each simplex iteration is to decide which index to remove from the basis $\mathcal{B}$. Unless the step is a direction of unboundedness, a single index must be removed from $\mathcal{B}$ and replaced by another from outside $\mathcal{B}$. We can gain some insight into how this decision is made by looking again at the KKT conditions (13.4).

From $\mathcal{B}$ and (13.4), we can derive values for not just the primal variable x but also the dual variables (λ, s), as we now show. First, define the nonbasic index set $\mathcal{N}$ as the complement of $\mathcal{B}$, that is,

$$\mathcal{N} = \{1, 2, \ldots, n\} \backslash \mathcal{B}. \tag{13.17}$$

Just as B is the basic matrix, whose columns are A_i for $i \in \mathcal{B}$, we use N to denote the nonbasic matrix $N = [A_i]_{i \in \mathcal{N}}$. We also partition the n-element vectors x, s, and c according to the index sets $\mathcal{B}$ and $\mathcal{N}$, using the notation

$$\begin{aligned} x_{\mathrm{B}} &= [x_i]_{i \in \mathcal{B}}, \quad & x_{\mathrm{N}} &= [x_i]_{i \in \mathcal{N}}, \\ s_{\mathrm{B}} &= [s_i]_{i \in \mathcal{B}}, \quad & s_{\mathrm{N}} &= [s_i]_{i \in \mathcal{N}}, \\ c_{\mathrm{B}} &= [c_i]_{i \in \mathcal{B}}, \quad & c_{\mathrm{N}} &= [c_i]_{i \in \mathcal{N}}. \end{aligned}$$

From the KKT condition (13.4b), we have that

$$Ax = Bx_{\mathrm{B}} + Nx_{\mathrm{N}} = b.$$

The primal variable x for this simplex iterate is defined as

$$x_{\mathrm{B}} = B^{-1}b, \quad x_{\mathrm{N}} = 0. \tag{13.18}$$

Since we are dealing only with basic feasible points, we know that B is nonsingular and that $x_{\mathrm{B}} \geq 0$, so this choice of x satisfies two of the KKT conditions: the equality constraints (13.4b) and the nonnegativity condition (13.4c).

We choose s to satisfy the complementarity condition (13.4e) by setting $s_{\mathrm{B}} = 0$. The remaining components λ and s_{N} can be found by partitioning this condition into c_{B} and c_{N} components and using $s_{\mathrm{B}} = 0$ to obtain

$$B^T\lambda = c_{\mathrm{B}}, \quad N^T\lambda + s_{\mathrm{N}} = c_{\mathrm{N}}. \tag{13.19}$$

Since B is square and nonsingular, the first equation uniquely defines λ as

$$\lambda = B^{-T}c_{\mathrm{B}}. \tag{13.20}$$

The second equation in (13.19) implies a value for s_{N}:

$$s_{\mathrm{N}} = c_{\mathrm{N}} - N^T\lambda = c_{\mathrm{N}} - (B^{-1}N)^T c_{\mathrm{B}}. \tag{13.21}$$

Computation of the vector s_N is often referred to as *pricing*. The components of s_N are often called the *reduced costs* of the nonbasic variables x_N.

The only KKT condition that we have not enforced explicitly is the nonnegativity condition $s \geq 0$. The basic components s_B certainly satisfy this condition, by our choice $s_B = 0$. If the vector s_N defined by (13.21) also satisfies $s_N \geq 0$, we have found an optimal vector triple (x, λ, s), so the algorithm can terminate and declare success. Usually, however, one or more of the components of s_N are negative. The new index to enter the basis $\mathcal{B}$—the *entering index*—is chosen to be *one of the indices* $q \in \mathcal{N}$ *for which* $s_q < 0$. As we show below, the objective $c^T x$ will decrease when we allow x_q to become positive if and only if (i) $s_q < 0$ and (ii) it is possible to increase x_q away from zero while maintaining feasibility of x. Our procedure for altering $\mathcal{B}$ and changing x and s can be described accordingly as follows:

- allow x_q to increase from zero during the next step;
- fix all other components of x_N at zero, and figure out the effect of increasing x_q on the current basic vector x_B, given that we want to stay feasible with respect to the equality constraints $Ax = b$;
- keep increasing x_q until one of the components of x_B (x_p, say) is driven to zero, or determining that no such component exists (the unbounded case);
- remove index p (known as the *leaving index*) from $\mathcal{B}$ and replace it with the entering index q.

This process of selecting entering and leaving indices, and performing the algebraic operations necessary to keep track of the values of the variables x, λ, and s, is sometimes known as *pivoting*.

We now formalize the pivoting procedure in algebraic terms. Since both the new iterate x^+ and the current iterate x should satisfy $Ax = b$, and since $x_N = 0$ and $x_i^+ = 0$ for $i \in \mathcal{N}\backslash\{q\}$, we have

$$Ax^+ = Bx_B^+ + A_q x_q^+ = Bx_B = Ax.$$

By multiplying this expression by B^{-1} and rearranging, we obtain

$$x_B^+ = x_B - B^{-1} A_q x_q^+. \tag{13.22}$$

Geometrically speaking, (13.22) is usually a move along an edge of the feasible polytope that decreases $c^T x$. We continue to move along this edge until a new vertex is encountered. At this vertex, a new constraint $x_p \geq 0$ must have become active, that is, one of the components x_p, $p \in \mathcal{B}$, has decreased to zero. We then remove this index p from the basis $\mathcal{B}$ and replace it by q.

We now show how the step defined by (13.22) affects the value of $c^T x$. From (13.22), we have

$$c^T x^+ = c_{\text{B}}^T x_{\text{B}}^+ + c_q x_q^+ = c_{\text{B}}^T x_{\text{B}} - c_{\text{B}}^T B^{-1} A_q x_q^+ + c_q x_q^+. \tag{13.23}$$

From (13.20) we have $c_{\text{B}}^T B^{-1} = \lambda^T$, while from the second equation in (13.19), since $q \in \mathcal{N}$, we have $A_q^T \lambda = c_q - s_q$. Therefore,

$$c_{\text{B}}^T B^{-1} A_q x_q^+ = \lambda^T A_q x_q^+ = (c_q - s_q) x_q^+,$$

so by substituting in (13.23) we obtain

$$c^T x^+ = c_{\text{B}}^T x_{\text{B}} - (c_q - s_q) x_q^+ + c_q x_q^+ = c^T x + s_q x_q^+. \tag{13.24}$$

Since q was chosen to have $s_q < 0$, it follows that the step (13.22) produces a decrease in the primal objective function $c^T x$ whenever $x_q^+ > 0$.

It is possible that we can increase x_q^+ to ∞ without ever encountering a new vertex. In other words, the constraint $x_{\text{B}}^+ = x_{\text{B}} - B^{-1} A_q x_q^+ \geq 0$ holds for all positive values of x_q^+. When this happens, the linear program is *unbounded*; the simplex method has identified a ray that lies entirely within the feasible polytope along which the objective $c^T x$ decreases to $-\infty$.

Figure 13.3 shows a path traversed by the simplex method for a problem in $\mathbb{R}^2$. In this example, the optimal vertex x^* is found in three steps.

If the basis $\mathcal{B}$ is *nondegenerate* (see Definition 13.1), then we are guaranteed that $x_q^+ > 0$, so we can be assured of a strict decrease in the objective function $c^T x$ at this step. If

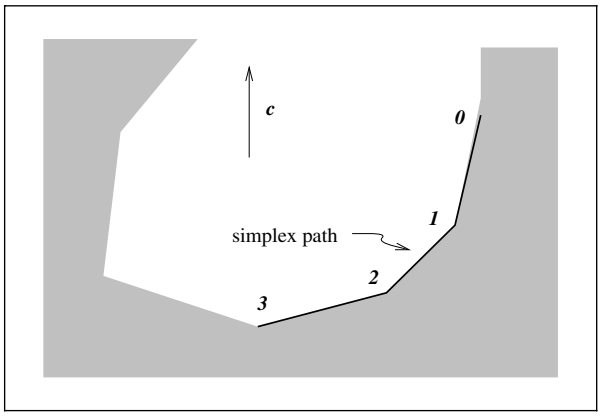

Figure 13.3
Simplex iterates for a two-dimensional problem.

the *problem* (13.1) is nondegenerate, we can ensure a decrease in $c^T x$ at *every* step, and can therefore prove the following result concerning termination of the simplex method.

Theorem 13.4.

Provided that the linear program (13.1) is nondegenerate and bounded, the simplex method terminates at a basic optimal point.

PROOF. The simplex method cannot visit the same basic feasible point x at two different iterations, because it attains a strict decrease at each iteration. Since the number of possible bases $\mathcal{B}$ is finite (there are only a finite number of ways to choose a subset of m indices from $\{1, 2, \ldots, n\}$), and since each basis defines a single basic feasible point, there are only a finite number of basic feasible points. Hence, the number of iterations is finite. Moreover, since the method is always able to take a step away from a nonoptimal basic feasible point, and since the problem is not unbounded, the method must terminate at a basic optimal point. □

This result gives us a proof of Theorem 13.2 (iii) in the case in which the linear program is nondegenerate. The proof of finite termination is considerably more complex when nondegeneracy of (13.1) is not assumed, as we discuss at the end of Section 13.5.

A SINGLE STEP OF THE METHOD

We have covered most of the mechanics of taking a single step of the simplex method. To make subsequent discussions easier to follow, we summarize our description.

Procedure 13.1 (One Step of Simplex).
Given $\mathcal{B}, \mathcal{N}, x_B = B^{-1}b \geq 0, x_N = 0$;
Solve $B^T\lambda = c_B$ for λ,
Compute $s_N = c_N - N^T\lambda$; (* pricing *)
if $s_N \geq 0$
    stop; (* optimal point found *)
Select $q \in \mathcal{N}$ with $s_q < 0$ as the entering index;
Solve $Bd = A_q$ for d;
if $d \leq 0$
    stop; (* problem is unbounded *)
Calculate $x_q^+ = \min_{i \mid d_i > 0} (x_B)_i / d_i$, and use p to denote the minimizing i;
Update $x_B^+ = x_B - dx_q^+, x_N^+ = (0, \ldots, 0, x_q^+, 0, \ldots, 0)^T$;
Change $\mathcal{B}$ by adding q and removing the basic variable corresponding to column p of B.

We illustrate this procedure with a simple example.

❑ EXAMPLE 13.1

Consider the problem

$$\begin{aligned} \min\ -4x_1 - 2x_2 \quad & \text{subject to} \\ x_1 + x_2 + x_3 &= 5, \\ 2x_1 + (1/2)x_2 + x_4 &= 8, \\ x &\geq 0. \end{aligned}$$

Suppose we start with the basis $\mathcal{B} = \{3, 4\}$, for which we have

$$x_{\text{B}} = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix}, \quad \lambda = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad s_{\text{N}} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} -3 \\ -2 \end{bmatrix},$$

and an objective value of $c^T x = 0$. Since both elements of s_{N} are negative, we could choose either 1 or 2 to be the entering variable. Suppose we choose $q = 1$. We obtain $d = (1, 2)^T$, so we cannot (yet) conclude that the problem is unbounded. By performing the ratio calculation, we find that $p = 2$ (corresponding to the index 4) and $x_1^+ = 4$. We update the basic and nonbasic index sets to $\mathcal{B} = \{3, 1\}$ and $\mathcal{N} = \{4, 2\}$, and move to the next iteration.

At the second iteration, we have

$$x_{\text{B}} = \begin{bmatrix} x_3 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \quad \lambda = \begin{bmatrix} 0 \\ -3/2 \end{bmatrix}, \quad s_{\text{N}} = \begin{bmatrix} s_4 \\ s_2 \end{bmatrix} = \begin{bmatrix} 3/2 \\ -5/4 \end{bmatrix},$$

with an objective value of -12. We see that s_{N} has one negative component, corresponding to the index $q = 2$, so we select this index to enter the basis. We obtain $d = (3/2, -1/2)^T$, so again we do not detect unboundedness. Continuing, we find that the maximum value of x_2^+ is $4/3$, and that $p = 1$, which indicates that index 3 will leave the basis $\mathcal{B}$. We update the index sets to $\mathcal{B} = \{2, 1\}$ and $\mathcal{N} = \{4, 3\}$ and continue.

At the start of the third iteration, we have

$$x_{\text{B}} = \begin{bmatrix} x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} 4/3 \\ 11/3 \end{bmatrix}, \quad \lambda = \begin{bmatrix} -5/3 \\ -2/3 \end{bmatrix}, \quad s_{\text{N}} = \begin{bmatrix} s_4 \\ s_3 \end{bmatrix} = \begin{bmatrix} 7/3 \\ 5/3 \end{bmatrix},$$

with an objective value of $c^T x = -41/3$. We see that $s_{\text{N}} \geq 0$, so the optimality test is satisfied, and we terminate.

❑

We need to flesh out Procedure 13.1 with specifics of three important aspects of the implementation:

- Linear algebra issues—maintaining an LU factorization of B that can be used to solve for λ and d.
- Selection of the entering index q from among the negative components of s_N. (In general, there are many such components.)
- Handling of degenerate bases and degenerate steps, in which it is not possible to choose a positive value of x_q^+ without violating feasibility.

Proper handling of these issues is crucial to the efficiency of a simplex implementation. We give some details in the next three sections.

13.4 LINEAR ALGEBRA IN THE SIMPLEX METHOD

We have to solve two linear systems involving the matrix B at each step; namely,

$$B^T\lambda = c_B, \quad Bd = A_q. \tag{13.25}$$

We never calculate the inverse basis matrix B^{-1} explicitly just to solve these systems. Instead, we calculate or maintain some factorization of B—usually an LU factorization—and use triangular substitutions with the factors to recover λ and d. It is less expensive to update the factorization than to calculate it afresh at each iteration because the basis matrix B changes by just a single column between iterations.

The standard factorization/updating procedures start with an LU factorization of B at the first iteration of the simplex algorithm. Since in practical applications B is large and sparse, its rows and columns are rearranged during the factorization to maintain both numerical stability and sparsity of the L and U factors. One successful pivot strategy that trades off between these two aims was proposed by Markowitz in 1957 [202]; it is still used as the basis of many practical sparse LU algorithms. Other considerations may also enter into our choice of row and column reordering of B. For example, it may help to improve the efficiency of the updating procedure if as many as possible of the leading columns of U contain just a single nonzero, on the diagonal. Many heuristics have been devised for choosing row and column permutations that produce this and other desirable structural features.

Let us assume for simplicity that row and column permutations are already incorporated in B, so that we write the initial LU factorization as

$$LU = B, \tag{13.26}$$

(L is unit lower triangular, U is upper triangular). The system $Bd = A_q$ can then be solved by the following two-step procedure:

$$L\bar{d} = A_q, \quad Ud = \bar{d}. \tag{13.27}$$

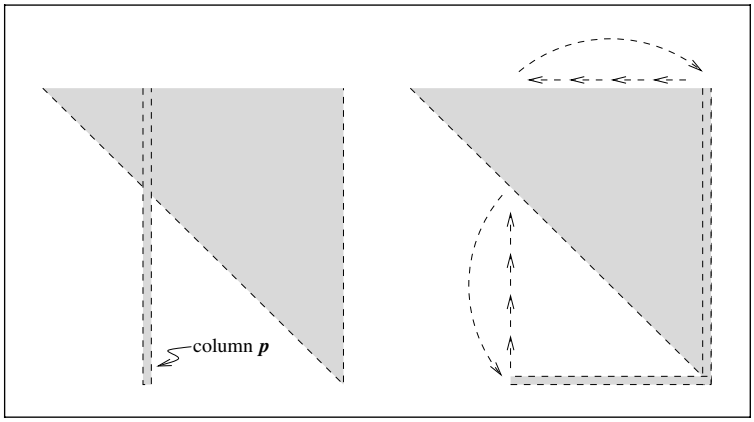

Figure 13.4 Left: $L^{-1}B^{+}$, which is upper triangular except for the column occupied by A_p. Right: After cyclic row and column permutation P_1, the non–upper triangular part of $P_1L^{-1}B^{+}P_1^T$ appears in the last row.

Similarly, the system $B^T\lambda = c_B$ is solved by performing the following two triangular substitutions:

$$U^T\bar{\lambda} = c_B, \qquad L^T\lambda = \bar{\lambda}.$$

We now discuss a procedure for updating the factors L and U after one step of the simplex method, when the index p is removed from the basis $\mathcal{B}$ and replaced by the index q. The corresponding change to the basis matrix B is that the column B_p is removed from B and replaced by A_q. We call the resulting matrix B^+ and note that if we rewrite (13.26) as $U = L^{-1}B$, the modified matrix $L^{-1}B^+$ will be upper triangular except in column p. That is, $L^{-1}B^+$ has the form shown on the left in Figure 13.4.

We now perform a cyclic permutation that moves column p to the last column position m and moves columns $p+1, p+2, \ldots, m$ one position to the left to make room for it. If we apply the same permutation to rows p through m, the net effect is to move the non-upper triangular part to the last row of the matrix, as shown in Figure 13.4. If we denote the permutation matrix by P_1, the matrix illustrated at right in Figure 13.4 is $P_1L^{-1}B^{+}P_1^T$.

Finally, we perform sparse Gaussian elimination on the matrix $P_1L^{-1}B^{+}P_1^T$ to restore upper triangular form. That is, we find L_1 and U_1 (lower and upper triangular, respectively) such that

$$P_1L^{-1}B^{+}P_1^T = L_1U_1. \tag{13.28}$$

It is easy to show that L_1 and U_1 have a simple form. The lower triangular matrix L_1 differs from the identity only in the last row, while U_1 is identical to $P_1L^{-1}B^{+}P_1^T$ except that the (m, m) element is changed and the off-diagonal elements in the last row are eliminated.

We give details of this process for the case of $m = 5$. Using the notation

$$L^{-1}B = U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ & u_{22} & u_{23} & u_{24} & u_{25} \\ & & u_{33} & u_{34} & u_{35} \\ & & & u_{44} & u_{45} \\ & & & & u_{55} \end{bmatrix}, \qquad L^{-1}A_q = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix},$$

and supposing that $p = 2$ (so that the second column is replaced by $L^{-1}A_q$), we have

$$L^{-1}B^+ = \begin{bmatrix} u_{11} & w_1 & u_{13} & u_{14} & u_{15} \\ & w_2 & u_{23} & u_{24} & u_{25} \\ & w_3 & u_{33} & u_{34} & u_{35} \\ & w_4 & & u_{44} & u_{45} \\ & w_5 & & & u_{55} \end{bmatrix}.$$

After the cyclic permutation P_1, we have

$$P_1 L^{-1} B^+ P_1{}^T = \begin{bmatrix} u_{11} & u_{13} & u_{14} & u_{15} & w_1 \\ & u_{33} & u_{34} & u_{35} & w_3 \\ & & u_{44} & u_{45} & w_4 \\ & & & u_{55} & w_5 \\ & u_{23} & u_{24} & u_{25} & w_2 \end{bmatrix}. \tag{13.29}$$

The factors L_1 and U_1 are now as follows:

$$L_1 = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ 0 & l_{52} & l_{53} & l_{54} & 1 \end{bmatrix}, \quad U_1 = \begin{bmatrix} u_{11} & u_{13} & u_{14} & u_{15} & w_1 \\ & u_{33} & u_{34} & u_{35} & w_3 \\ & & u_{44} & u_{45} & w_4 \\ & & & u_{55} & w_5 \\ & & & & \hat{w}_2 \end{bmatrix}, \tag{13.30}$$

for certain values of l_{52}, l_{53}, l_{54}, and $\hat{w}_2$ (see Exercise 13.10).

The result of this updating process is the factorization (13.28), which we can rewrite as follows:

$$B^+ = L^+ U^+, \quad \text{where } L^+ = L P_1^T L_1, \quad U^+ = U_1 P_1. \tag{13.31}$$

There is no need to calculate L^+ and U^+ explicitly. Rather, the nonzero elements in L_1 and the last column of U_1, and the permutation information in P_1, can be stored in compact form, so that triangular substitutions involving L^+ and U^+ can be performed by applying a number of permutations and sparse triangular substitutions involving these factors. The factorization updates from subsequent simplex steps are stored and applied in a similar fashion.

The procedure we have just outlined is due to Forrest and Tomlin [110]. It is quite efficient, because it requires the storage of little data at each update and does not require much movement of data in memory. Its major disadvantage is possible numerical instability. Large elements in the factors of a matrix are a sure indicator of instability, and the multipliers in the L_1 factor (l_{52} in (13.30), for example) may be very large. An earlier scheme of Bartels and Golub [12] allowed swapping of rows to avoid these problems. For instance, if $|u_{33}| < |u_{23}|$ in (13.29), we could swap rows 2 and 5 to ensure that the subsequent multiplier l_{52} in the L_1 factor does not exceed 1 in magnitude. This improved stability comes at a price: The lower right corner of the upper triangular factor may become more dense during each update.

Although the update information for each iteration (the permutation matrices and the sparse triangular factors) can often be stored in a highly compact form, the total amount of space may build up to unreasonable levels after many such updates have been performed. As the number of updates builds up, so does the time needed to solve for the vectors d and λ in Procedure 13.1. If an unstable updating procedure is used, numerical errors may also come into play, blocking further progress by the simplex algorithm. For all these reasons, most simplex implementations periodically calculate a fresh LU factorization of the current basis matrix B and discard the accumulated updates. The new factorization uses the same permutation strategies that we apply to the very first factorization, which balance the requirements of stability, sparsity, and structure.

13.5 OTHER IMPORTANT DETAILS

PRICING AND SELECTION OF THE ENTERING INDEX

There are usually many negative components of s_N at each step. How do we choose one of these to become the index that enters the basis? Ideally, we would like to choose the sequence of entering indices q that gets us to the solution x^* in the fewest possible steps, but we rarely have the global perspective needed to implement this strategy. Instead, we use more shortsighted but practical strategies that obtain a significant decrease in $c^T x$ on just the present iteration. There is usually a tradeoff between the effort spent on finding a good entering index and the amount of decrease in $c^T x$ resulting from this choice. Different pivot strategies resolve this tradeoff in different ways.

Dantzig's original selection rule is one of the simplest. It chooses q such that s_q is the most negative component of $s_N = N^T \lambda$. This rule, which is motivated by (13.24), gives the maximum improvement in $c^T x$ per unit increase in the entering variable x_q. A large

reduction in $c^T x$ is not guaranteed, however. It could be that we can increase x_q^+ only a tiny amount from zero (or not at all) before reaching the next vertex.

Calculation of the entire vector s_N from (13.21) requires a multiplication by N^T, which can be expensive when the matrix N is very large. *Partial pricing* strategies calculate only a subvector of s_N and make the choice of entering variable from among the negative entries in this subvector. To give all the indices in $\mathcal{N}$ a chance to enter the basis, these strategies cycle through the nonbasic elements, periodically changing the subvector of s_N they evaluate so that no nonbasic index is ignored for too long.

Neither of these strategies guarantees that we can make a substantial move along the chosen edge before reaching a new vertex. *Multiple pricing* strategies are more thorough: For a small subset of indices $q \in \mathcal{N}$, they evaluate s_q and, if $s_q < 0$, the maximum value of x_q^+ that maintains feasibility of x^+ and the consequent change $s_q x_q^+$ in the objective function (see (13.24)). Calculation of x_q^+ requires evaluation of $d = B^{-1}A_q$ as in Procedure 13.1, which is not cheap. Subsequent iterations deal with this same index subset until we reach an iteration at which all s_q are nonnegative for q in the subset. At this point, the full vector s_N is computed, a new subset of nonbasic indices is chosen, and the cycle begins again. This approach has the advantage that the columns of the matrix N outside the current subset of priced components need not be accessed at all, so memory access in the implementation is quite localized.

Naturally, it is possible to devise heuristics that combine partial and multiple pricing in various imaginative ways.

A sophisticated rule known as *steepest edge* chooses the "most downhill" direction from among all the candidates—the one that produces the largest decrease in $c^T x$ per unit distance moved along the edge. (By contrast, Dantzig's rule maximizes the decrease in $c^T x$ per unit change in x_q^+, which is not the same thing, as a small change in x_q^+ can correspond to a large distance moved along the edge.) During the pivoting step, the overall change in x is

$$x^+ = \begin{bmatrix} x_B^+ \\ x_N^+ \end{bmatrix} = \begin{bmatrix} x_B \\ x_N \end{bmatrix} + \begin{bmatrix} -B^{-1}A_q \\ e_q \end{bmatrix} x_q^+ = x + \eta_q x_q^+, \tag{13.32}$$

where e_q is the unit vector with a 1 in the position corresponding to the index $q \in \mathcal{N}$ and zeros elsewhere, and the vector η_q is defined as

$$\eta_q = \begin{bmatrix} -B^{-1}A_q \\ e_q \end{bmatrix} = \begin{bmatrix} -d \\ e_q \end{bmatrix}; \tag{13.33}$$

see (13.25). The change in $c^T x$ per unit step along η_q is given by

$$\frac{c^T \eta_q}{\|\eta_q\|}. \tag{13.34}$$

The steepest-edge rule chooses $q \in \mathcal{N}$ to minimize this quantity.

If we had to compute each η_i by solving $Bd = A_i$ for each $i \in \mathcal{N}$, the steepest-edge strategy would be prohibitively expensive. Goldfarb and Reid [134] showed that the measure (13.34) of edge steepness for all indices $i \in \mathcal{N}$ can, in fact, be updated quite economically at each iteration. We outline their steepest-edge procedure by showing how each $c^T\eta_i$ and $\|\eta_i\|$ can be updated at the current iteration.

First, note that we already know the numerator $c^T\eta_i$ in (13.34) without calculating η_i, because by taking the inner product of (13.32) with c and using (13.24), we have that $c^T\eta_i = s_i$. To investigate the change in denominator $\|\eta_i\|$ at this step, we define $\gamma_i = \|\eta_i\|^2$, where this quantity is defined before and after the update as follows:

$$\gamma_i = \|\eta_i\|^2 = \|B^{-1}A_i\|^2 + 1, \tag{13.35a}$$

$$\gamma_i^+ = \|\eta_i^+\|^2 = \|(B^+)^{-1}A_i\|^2 + 1. \tag{13.35b}$$

Assume without loss of generality that the entering column A_q replaces the first column of the basis matrix B (that is, $p = 1$), and that this column corresponds to the index t. We can then express the update to B as follows:

$$B^+ = B + (A_q - A_t)e_1^T = B + (A_q - Be_1)e_1^T, \tag{13.36}$$

where $e_1 = (1, 0, 0, \ldots, 0)^T$. By applying the Sherman–Morrison formula (A.27) to the rank-one update formula in (13.36), we obtain

$$(B^+)^{-1} = B^{-1} - \frac{(B^{-1}A_q - e_1)e_1^T B^{-1}}{1 + e_1^T(B^{-1}A_q - e_1)} = B^{-1} - \frac{(d - e_1)e_1^T B^{-1}}{e_1^T d},$$

where again we have used the fact that $d = B^{-1}A_q$ (see (13.25)). Therefore, we have that

$$(B^+)^{-1}A_i = B^{-1}A_i - \frac{e_1^T B^{-1}A_i}{e_1^T d}(d - e_1).$$

By substituting for $(B^+)^{-1}A_i$ in (13.35) and performing some simple manipulation, we obtain

$$\gamma_i^+ = \gamma_i - 2\left(\frac{e_1^T B^{-1}A_i}{e_1^T d}\right)A_i^T B^{-T}d + \left(\frac{e_1^T B^{-1}A_i}{e_1^T d}\right)^2 \gamma_q. \tag{13.37}$$

Once we solve the following two linear systems to obtain $\hat{d}$ and r:

$$B^T\hat{d} = d, \qquad B^T r = e_1. \tag{13.38}$$

The formula (13.37) then becomes

$$\gamma_i^+ = \gamma_i - 2\left(\frac{r^T A_i}{r^T A_q}\right)\hat{d}^T A_i + \left(\frac{r^T A_i}{r^T A_q}\right)^2 \gamma_q. \tag{13.39}$$

Hence, the entire set of γ_i values, for $i \in \mathcal{N}$ with $i \neq q$, can be calculated by solving the two systems (13.38) and then evaluating the inner products $r^T A_i$ and $\hat{d}^T A_i$, for each i.

The steepest-edge strategy does not guarantee that we can take a long step before reaching another vertex, but it has proved to be highly effective in practice.

STARTING THE SIMPLEX METHOD

The simplex method requires a basic feasible starting point x and a corresponding initial basis $\mathcal{B} \subset \{1, 2, \ldots, n\}$ with $|\mathcal{B}| = m$ such that the basis matrix B defined by (13.13) is nonsingular and $x_B = B^{-1}b \geq 0$ and $x_N = 0$. The problem of finding this initial point and basis may itself be nontrivial—in fact, its difficulty is equivalent to that of actually solving a linear program. We describe here the *two-phase approach* that is commonly used to deal with this difficulty in practical implementations.

In Phase I of this approach we set up an auxiliary linear program based on the data of (13.1), and solve it with the simplex method. The Phase-I problem is designed so that an initial basis and initial basic feasible point is trivial to find, and so that its solution gives a basic feasible initial point for the second phase. In Phase II, a second linear program similar to the original problem (13.1) is solved, with the Phase-I solution as a starting point. The solution of the original problem (13.1) can be extracted easily from the solution of the Phase-II problem.

In Phase I we introduce artificial variables z into (13.1) and redefine the objective function to be the sum of these artificial variables, as follows:

$$\min e^T z, \quad \text{subject to } Ax + Ez = b, (x, z) \geq 0, \tag{13.40}$$

where $z \in \mathbb{R}^m$, $e = (1, 1, \ldots, 1)^T$, and E is a diagonal matrix whose diagonal elements are

$$E_{jj} = +1 \ \text{ if } b_j \geq 0, \qquad E_{jj} = -1 \ \text{ if } b_j < 0.$$

It is easy to see that the point (x, z) defined by

$$x = 0, \qquad z_j = |b_j|, \qquad j = 1, 2, \ldots, m, \tag{13.41}$$

is a basic feasible point for (13.40). Obviously, this point satisfies the constraints in (13.40), while the initial basis matrix B is simply the diagonal matrix E, which is clearly nonsingular.

At any feasible point for (13.40), the artificial variables z represent the amounts by which the constraints $Ax = b$ are violated by the x component. The objective function is

simply the sum of these violations, so by minimizing this sum we are forcing x to become feasible for the original problem (13.1). It is not difficult to see that the Phase-I problem (13.40) has an optimal objective value of zero if and only if the original problem (13.1) is feasible, by using the following argument: If there exists a vector $(\tilde{x}, \tilde{z})$ that is feasible for (13.40) such that $e^T\tilde{z} = 0$, we must have $\tilde{z} = 0$, and therefore $A\tilde{x} = b$ and $\tilde{x} \geq 0$, so $\tilde{x}$ is feasible for (13.1). Conversely, if $\tilde{x}$ is feasible for (13.1), then the point $(\tilde{x}, 0)$ is feasible for (13.40) with an objective value of 0. Since the objective in (13.40) is obviously nonnegative at all feasible points, then $(\tilde{x}, 0)$ must be optimal for (13.40), verifying our claim.

In Phase I, we apply the simplex method to (13.40) from the initial point (13.41). This linear program cannot be unbounded, because its objective function is bounded below by 0, so the simplex method will terminate at an optimal point (assuming that it does not cycle; see below). If the objective $e^T z$ is positive at this solution, we conclude by the argument above that the original problem (13.1) is infeasible. Otherwise, the simplex method identifies a point $(\tilde{x}, \tilde{z})$ with $e^T\tilde{z} = 0$, which is also a basic feasible point for the following *Phase-II problem*:

$$\min c^T x \quad \text{subject to } Ax + z = b, \quad x \geq 0, \quad 0 \geq z \geq 0. \tag{13.42}$$

Note that this problem differs from (13.40) in that the objective function is replaced by the original objective $c^T x$, while upper bounds of 0 have been imposed on z. In fact, (13.42) is equivalent to (13.1), because any solution (and indeed any feasible point) must have $z = 0$. We need to retain the artificial variables z in Phase II, however, since some components of z may still be present in the optimal basis from Phase I that we are using as the initial basis for (13.42), though of course the values $\tilde{z}_j$ of these components must be zero. In fact, we can modify (13.42) to include *only* those components of z that are present in the optimal basis for (13.40).

The problem (13.42) is not quite in standard form because of the two-sided bounds on z. However, it is easy to modify the simplex method described above to handle upper and lower bounds on the variables (we omit the details). We can customize the simplex algorithm slightly by deleting each component of z from the problem (13.42) as soon as it is swapped out of the basis. This strategy ensures that components of z do not repeatedly enter and leave the basis, thereby avoiding unnecessary simplex iterations.

If (x^*, z^*) is a basic solution of (13.42), it must have $z^* = 0$, and so x^* is a solution of (13.1). In fact, x^* is a basic feasible point for (13.1), though this claim is not completely obvious because the final basis $\mathcal{B}$ for the Phase-II problem may still contain components of z^*, making it unsuitable as an optimal basis for (13.1). Since A has full row rank, however, we can construct an optimal basis for (13.1) in a postprocessing phase: Extract from $\mathcal{B}$ any components of z that are present, and replace them with nonbasic components of x in a way that maintains nonsingularity of the submatrix B defined by (13.13).

A final point to note is that in many problems we do not need to add a complete set of m artificial variables to form the Phase-I problem. This observation is particularly relevant when slack and surplus variables have already been added to the problem formulation, as

in (13.2), to obtain a linear program with inequality constraints in standard form (13.1). Some of these slack/surplus variables can play the roles of artificial variables, making it unnecessary to include such variables explicitly.

We illustrate this point with the following example.

❑ EXAMPLE 13.2

Consider the inequality-constrained linear program defined by

$$\begin{aligned} \min\ & 3x_1 + x_2 + x_3 \quad \text{subject to} \\ & 2x_1 + x_2 + x_3 \leq 2, \\ & x_1 - x_2 - x_3 \leq -1, \\ & x \geq 0. \end{aligned}$$

By adding slack variables to both inequality constraints, we obtain the following equivalent problem in standard form:

$$\begin{aligned} \min\ & 3x_1 + x_2 + x_3 \quad \text{subject to} \\ & 2x_1 + x_2 + x_3 + x_4 = 2, \\ & x_1 - x_2 - x_3 + x_5 = -1, \\ & x \geq 0. \end{aligned}$$

By inspection, it is easy to see that the vector $x = (0, 0, 0, 2, 0)$ is feasible with respect to the first linear constraint and the lower bound $x \geq 0$, though it does not satisfy the second constraint. Hence, in forming the Phase-I problem, we add just a single artificial variable z_2 to the second constraint and obtain

$$\min\ z_2 \quad \text{subject to} \tag{13.43}$$

$$2x_1 + x_2 + x_3 + x_4 = 2, \tag{13.44}$$

$$x_1 - x_2 - x_3 + x_5 - z_2 = -1, \tag{13.45}$$

$$(x, z_2) \geq 0. \tag{13.46}$$

It is easy to see that the vector $(x, z_2) = ((0, 0, 0, 2, 0), 1)$ is feasible with respect to (13.43). In fact, it is a basic feasible point, since the corresponding basis matrix B is

$$B = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

which is clearly nonsingular. In this example, the variable x_4 plays the role of artificial variable for the first constraint. There was no need to add an explicit artificial variable z_1. ❑

DEGENERATE STEPS AND CYCLING

As noted above, the simplex method may encounter situations in which for the entering index q, we cannot set x_q^+ any greater than zero in (13.22) without violating the nonnegativity condition $x^+ \geq 0$. By referring to Procedure 13.1, we see that these situations arise when there is i with $(x_B)_i = 0$ and $d_i < 0$, where d is defined by (13.25). Steps of this type are called *degenerate steps*. On such steps, the components of x do not change and, therefore, the objective function $c^T x$ does not decrease. However, the steps may still be useful because they change the basis $\mathcal{B}$ (by replacing one index), and the updated $\mathcal{B}$ may be closer to the optimal basis. In other words, the degenerate step may be laying the groundwork for reductions in $c^T x$ on later steps.

Sometimes, however, a phenomenon known as *cycling* can occur. After a number of successive degenerate steps, we may return to an earlier basis $\mathcal{B}$. If we continue to apply the algorithm from this point using the same rules for selecting entering and leaving indices, we will repeat the same cycle ad infinitum, never converging.

Cycling was once thought to be a rare phenomenon, but in recent times it has been observed frequently in the large linear programs that arise as relaxations of integer programming problems. Since integer programs are an important source of linear programs, practical simplex codes usually incorporate a cycling avoidance strategy.

In the remainder of this section, we describe a *perturbation strategy* and its close relative, the *lexicographic strategy*.

Suppose that a degenerate basis is encountered at some simplex iteration, at which the basis is $\hat{\mathcal{B}}$ and the basis matrix is $\hat{B}$, say. We consider a modified linear program in which we add a small perturbation to the right-hand side of the constraints in (13.1), as follows:

$$b(\epsilon) = b + \hat{B} \begin{bmatrix} \epsilon \\ \epsilon^2 \\ \\ \vdots \\ \epsilon^m \end{bmatrix},$$

where ϵ is a very small positive number. This perturbation in b induces a perturbation in the components of the basic solution vector; we have

$$x_{\hat{B}}(\epsilon) = x_{\hat{B}} + \begin{bmatrix} \epsilon \\ \epsilon^2 \\ \\ \vdots \\ \epsilon^m \end{bmatrix}. \tag{13.47}$$

Retaining the perturbation for subsequent iterations, we see that subsequent basic solutions have the form

$$x_{\mathrm{B}}(\epsilon) = x_{\mathrm{B}} + B^{-1}\hat{B}\begin{bmatrix} \epsilon \\ \epsilon^2 \\ \vdots \\ \epsilon^m \end{bmatrix} = x_{\mathrm{B}} + \sum_{k=1}^{m} (B^{-1}\hat{B})_{\cdot k}\epsilon^k, \tag{13.48}$$

where $(B^{-1}\hat{B})_{\cdot k}$ denotes the kth column of $B^{-1}\hat{B}$ and x_{B} represents the basic solution for the unperturbed right-hand side b.

From (13.47), we have that for all ϵ sufficiently small (but positive), $(x_{\hat{B}}(\epsilon))_i > 0$ for all i. Hence, the basis is nondegenerate for the perturbed problem, and we can perform a step of the simplex method that produces a nonzero (but tiny) decrease in the objective.

Indeed, if we retain the perturbation over all subsequent iterations, and provided that the initial choice of ϵ was small enough, we claim that *all* subsequent bases visited by the algorithm are nondegenerate. We prove this claim by contradiction, by assuming that there is some basis matrix B such that $(x_{\mathrm{B}}(\epsilon))_i = 0$ for some i and all ϵ sufficiently small. From (13.48), we see that this can happen only when $(x_{\mathrm{B}})_i = 0$ and $(B^{-1}\bar{B})_{ik} = 0$ for $k = 1, 2, \ldots, m$. The latter relation implies that the ith row of $B^{-1}\bar{B}$ is zero, which cannot occur, because both B and $\bar{B}$ are nonsingular.

We conclude that, provided the initial choice of ϵ is sufficiently small to ensure nondegeneracy of all subsequent bases, no basis is visited more than once by the simplex method and therefore, by the same logic as in the proof of Theorem 13.4, the method terminates finitely at a solution of the perturbed problem. The perturbation can be removed in a postprocessing phase, by resetting $x_{\mathrm{B}} = B^{-1}b$ for the final basis B and the original right-hand side b.

The question remains of how to choose ϵ small enough at the point at which the original degenerate basis $\hat{\mathcal{B}}$ is encountered. The *lexicographic strategy* finesses this issue by not making an explicit choice of ϵ, but rather keeping track of the dependence of each basic variable on each power of ϵ. When it comes to selecting the leaving variable, it chooses the index p that minimizes $(x_{\mathrm{B}}(\epsilon))_i/d_i$ over all variables in the basis, for all sufficiently small ϵ. (The choice of p is uniquely defined by this procedure, as we can show by an argument similar to the one above concerning nondenegeracy of each basis.) We can extend the pivot procedure slightly to update the dependence of each basic variable on the powers of ϵ at each iteration, including the variable x_q that has just entered the basis.

13.6 THE DUAL SIMPLEX METHOD

Here we describe another variant of the simplex method that is useful in a variety of situations and is often faster on many practical problems than the variant described above. This *dual*

simplex method uses many of the same concepts and methodology described above, such as the splitting of the matrix A into column submatrices B and N and the generation of iterates (x, λ, s) that satisfy the complementarity condition $x^T s = 0$. The method of Section 13.3 starts with a feasible x (with $x_B \geq 0$ and $x_N = 0$) and a corresponding dual iterate (λ, s) for which $s_B = 0$ but s_N is not necessarily nonnegative. After making systematic column interchanges between B and N, it finally reaches a feasible dual point (λ, s) at which $s_N \geq 0$, thus yielding a solution of both the primal problem (13.1) and the dual (13.8). By contrast, the dual simplex method starts with a point (λ, s) feasible for (13.8), at which $s_N \geq 0$ and $s_B = 0$, and a corresponding primal feasible point x for which $x_N = 0$ but x_B is not necessarily nonnegative. By making systematic column interchanges between B and N, it finally reaches a feasible primal point x for which $x_B \geq 0$, signifying optimality. Note that although the matrix B used in this algorithm is a nonsingular column submatrix of A, it is no longer correct to refer to it as a basis matrix, since it does not satisfy the feasibility condition $x_B = B^{-1}b \geq 0$.

We now describe a single step of this method in a similar fashion to Section 13.3, though the details are a little more complicated here. As mentioned above, we commence each step with submatrices B and N of A, and corresponding sets $\mathcal{B}$ and $\mathcal{N}$. The primal and dual variables corresponding to these sets are defined as follows (cf. (13.18), (13.20), and (13.21)):

$$x_B = B^{-1}b, \qquad x_N = 0, \tag{13.49a}$$

$$\lambda = B^{-T}c_B, \tag{13.49b}$$

$$s_B = c_B - B^T\lambda = 0, \qquad s_N = c_N - N^T\lambda \geq 0, \tag{13.49c}$$

If $x_B \geq 0$, the current point (x, λ, s) satisfies the optimality conditions (13.4), and we are done. Otherwise, we select a *leaving index* $q \in \mathcal{B}$ such that $x_q < 0$. Our aim is to move x_q to zero (thereby ensuring that nonnegativity holds for this component), while allowing s_q to increase away from zero. We will also identify an *entering index* $r \in \mathcal{N}$, such that s_r becomes zero on this step while x_r increases away from zero. Hence, the index q will move from $\mathcal{B}$ to $\mathcal{N}$, while r will move from $\mathcal{N}$ to $\mathcal{B}$. How do we choose r, and how are x, λ, and s changed on this step? The description below provides the answer. We use (x^+, λ^+, s^+) to denote the updated values of our variables, after this step is taken.

First, let e_q the vector of length m that contains all zeros except for a 1 in the position occupied by index q in the set $\mathcal{B}$. Since we increase s_q away from zero while fixing the remaining components of s_B at zero, the updated value s_B^+ will have the form

$$s_B^+ = s_B + \alpha e_q \tag{13.50}$$

for some positive scalar α to be determined. We write the corresponding update to λ as

$$\lambda^+ = \lambda + \alpha v, \tag{13.51}$$

for some vector v. In fact, since $s_{\mathcal{B}}^+$ and λ^+ must satisfy the first equation in (13.49c), we must have

$$\begin{aligned} s_{\mathcal{B}}^+ &= c_{\mathcal{B}} - B^T \lambda^+ \\ \Rightarrow s_{\mathcal{B}} + \alpha e_q &= c_{\mathcal{B}} - B^T(\lambda + \alpha v) \\ \Rightarrow e_q &= -B^T v, \end{aligned} \tag{13.52}$$

which is a system of equations that we can solve to obtain v.

To see how the dual objective value $b^T\lambda$ changes as a result of this step, we use (13.52) and the fact that $x_q = x_{\mathcal{B}}^T e_q$ to obtain

$$\begin{aligned} b^T\lambda^+ &= b^T\lambda + \alpha b^T v \\ &= b^T\lambda - \alpha b^T B^{-T} e_q && \text{from (13.52)} \\ &= b^T\lambda - \alpha x_{\mathcal{B}}^T e_q && \text{from (13.49a)} \\ &= b^T\lambda - \alpha x_q && \text{by definition of } e_q. \end{aligned}$$

Since $x_q < 0$ and since our aim is to maximize the dual objective, we would like to choose α as large as possible. The upper bound on α is provided by the constraint $s_{\mathcal{N}}^+ \geq 0$. Similarly to (13.49c), we have

$$s_{\mathcal{N}}^+ = c_{\mathcal{N}} - N^T\lambda^+ = s_{\mathcal{N}} - \alpha N^T v = s_{\mathcal{N}} - \alpha w,$$

where we have defined

$$w = N^T v = -N^T B^{-T} e_q.$$

The largest α for which $s_{\mathcal{N}}^+ \geq 0$ is given by the formula

$$\alpha = \min_{j \in \mathcal{N},\, w_j > 0} \frac{s_j}{w_j}.$$

We define the entering index r to be the index at which the minimum in this expression is achieved. Note that

$$s_r^+ = 0 \quad \text{and} \quad w_r = A_r^T v > 0, \tag{13.53}$$

where, as usual, A_r denotes the rth column of A.

Having now identified how λ and s are updated on this step, we need to figure out how x changes. For the leaving index q, we need to set $x_q^+ = 0$, while for the entering index r we can allow x_r^+ to be nonzero. We denote the direction of change for $x_{\mathcal{B}}$ to be the vector

d, defined by the following linear system:

$$Bd = \sum_{i \in \mathcal{B}} A_i d_i = A_r. \tag{13.54}$$

Since from (13.49a), we have

$$\sum_{i \in \mathcal{B}} A_i x_i = b,$$

we have that

$$\sum_{i \in \mathcal{B}} A_i (x_i - \gamma d_i) + A_r \gamma = b, \tag{13.55}$$

for any scalar γ. To ensure that $x_q^+ = 0$, we set

$$\gamma = \frac{x_q}{d_q}, \tag{13.56}$$

which is well defined only if d_q is nonzero. In fact, we have that $d_q < 0$, since

$$d_q = d^T e_q = A_r^T B^{-T} e_q = -A_r^T v = -w_r < 0,$$

where we have used the definition of e_q along with (13.54), (13.52), and (13.53) to derive these relationships. Since $x_q < 0$, it follows from (13.56) that $\gamma > 0$. Following (13.55) we can define the updated vector x^+ as follows:

$$x_i^+ = \begin{cases} x_i - \gamma d_i, & \text{for } i \in \mathcal{B} \text{ with } i \neq q, \\ 0, & \text{for } i = q, \\ 0, & \text{for } i \in \mathcal{N} \text{ with } i \neq r, \\ \gamma, & \text{for } i = r. \end{cases}$$

13.7 PRESOLVING

Presolving (also known as preprocessing) is carried out in practical linear programming codes to reduce the size of the user-defined linear programming problem before passing it to the solver. A variety of techniques—some obvious, some ingenious—are used to eliminate certain variables, constraints, and bounds from the problem. Often the reduction in problem size is quite dramatic, and the linear programming algorithm takes much less time when applied to the presolved problem than when applied to the original problem. Presolving is

beneficial regardless of what algorithm is used to solve the linear program; it is used both in simplex and interior-point codes. Infeasibility may also be detected by the presolver, eliminating the need to call the linear programming algorithm at all.

We mention just a few of the more straightforward preprocessing techniques here, referring the interested reader to Andersen and Andersen [4] for a more comprehensive list. For the purpose of this discussion, we assume that the linear program is formulated with both lower and upper bounds on x, that is,

$$\min c^T x, \quad \text{subject to } Ax = b, l \leq x \leq u, \tag{13.57}$$

where some components l_i of the lower bound vector may be $-\infty$ and some upper bounds u_i may be $+\infty$.

Consider first a *row singleton*, which happens when one of the equality constraints involves just one of the variables. Specifically, if constraint k involves only variable j (that is, $A_{kj} \neq 0$, but $A_{ki} = 0$ for all $i \neq j$), we can immediately set $x_j = b_k / A_{kj}$ and eliminate x_j from the problem. Note that if this value of x_j violates its bounds (that is, $x_j < l_j$ or $x_j > u_j$), we can declare the problem to be infeasible, and terminate.

Another obvious technique is the *free column singleton*, in which there is a variable x_j that occurs in only one of the equality constraints, and is free (that is, its lower bound is $-\infty$ and its upper bound is $+\infty$). In this case, we have for some k that $A_{kj} \neq 0$ while $A_{lj} = 0$ for all $l \neq k$. Here we can simply use constraint k to eliminate x_j from the problem, setting

$$x_j = \frac{b_k - \sum_{p \neq j} A_{kp} x_p}{A_{kj}}.$$

Once the values of x_p for $p \neq j$ have been obtained by solving a reduced linear program, we can substitute into this formula to recover x_j prior to returning the result to the user. This substitution does not require us to modify any other constraints, but it will change the cost vector c in general, whenever $c_j \neq 0$. We will need to make the replacement

$$c_p \leftarrow c_p - c_j A_{kp} / A_{kj}, \quad \text{for all } p \neq j.$$

In this case, we can also determine the dual variable associated with constraint k. Since x_j is a free variable, there is no dual slack associated with it, so the jth dual constraint becomes

$$\sum_{l=1}^{m} A_{lj} \lambda_l = c_j \quad \Rightarrow \quad A_{kj} \lambda_k = c_j,$$

from which we deduce that $\lambda_k = c_j / A_{kj}$.

Perhaps the simplest preprocessing check is for the presence of *zero rows and columns* in A. If $A_{ki} = 0$ for all $i = 1, 2, \ldots, n$, then provided that the right-hand side is also zero ($b_k = 0$), we can simply delete this row from the problem and set the corresponding

Lagrange multiplier λ_k to an arbitrary value. For a zero column—say, $A_{kj} = 0$ for all $k = 1, 2, \ldots, m$—we can determing the optimal value of x_j by inspecting its cost coefficient c_j and its bounds l_j and u_j. If $c_j < 0$, we set $x_j = u_j$ to minimize the product $c_j x_j$. (We are free to do so because x_j is not restricted by any of the equality constraints.) If $c_j < 0$ and $u_j = +\infty$, then the problem is unbounded. Similarly, if $c_j > 0$, we set $x_j = l_j$, or else declare unboundedness if $l_j = -\infty$.

A somewhat more subtle presolving technique is to check for *forcing or dominated constraints*. Rather than give a general specification, we illustrate this case with a simple example. Suppose that one of the equality constraints is as follows:

$$5x_1 - x_4 + 2x_5 = 10,$$

where the variables in question have the following bounds:

$$0 \leq x_1 \leq 1, \quad -1 \leq x_4 \leq 5, \quad 0 \leq x_5 \leq 2.$$

It is not hard to see that the equality constraint can only be satisfied if x_1 and x_5 are at their upper bounds and x_4 is at its lower bound. Any other feasible values of these variables would result in the left-hand side of the equality constraint being strictly less than 10. Hence, we can set $x_1 = 1$, $x_4 = -1$, $x_5 = 2$ and eliminate these variables, and the equality constraint, from the problem.

We use a similar example to illustrate dominated constraints. Suppose that we have the following constraint involving three variables:

$$2x_2 + x_6 - 3x_7 = 8,$$

where the variables in question have the following bounds:

$$-10 \leq x_2 \leq 10, \quad 0 \leq x_6 \leq 1, \quad 0 \leq x_7 \leq 2.$$

By rearranging the constraint and using the bounds on x_6 and x_7, we find that

$$x_2 = 4 - (1/2)x_6 + (3/2)x_7 \leq 4 - 0 + (3/2)2 = 7.$$

and similarly, using the opposite bounds on x_6 and x_7 we obtain $x_2 \geq 7/2$. We conclude that the stated bounds of -10 and 10 on x_2 are redundant, since x_2 is implicitly confined to an even smaller interval by the combination of the equality constraint and the bounds on x_6 and x_7. Hence, we can drop the bounds on x_2 from the formulation and treat it as a free variable.

Presolving techniques are applied recursively, because the elimination of certain variables or constraints may create situations that allow further eliminations. As a trivial example,

suppose that the following two equality constraints are present in the problem:

$$3x_2 = 6, \quad x_2 + 4x_5 = 10.$$

The first of these constraints is a row singleton, which we can use to set $x_2 = 2$ and eliminate this variable and constraint. After substitution, the second constraint becomes $4x_5 = 10 - x_2 = 8$, which is again a row singleton. We can therefore set $x_5 = 2$ and eliminate this variable and constraint as well.

Relatively little information about presolving techniques has appeared in the literature, in part because they have commercial value as an important component of linear programming software.

13.8 WHERE DOES THE SIMPLEX METHOD FIT?

In linear programming, as in all optimization problems in which inequality constraints are present, the fundamental task of the algorithm is to determine which of these constraints are active at the solution (see Definition 12.1 and which are inactive. The simplex method belongs to a general class of algorithms for constrained optimization known as *active set methods*, which explicitly maintain estimates of the active and inactive index sets that are updated at each step of the algorithm. (At each iteration, the basis $\mathcal{B}$ is our current estimate of the inactive set, that is, the set of indices i for which we suspect that $x_i > 0$ at the solution of the linear program.) Like most active set methods, the simplex method makes only modest changes to these index sets at each step; a single index is exchanged between $\mathcal{B}$ into $\mathcal{N}$.

Active set algorithms for quadratic programming, bound-constrained optimization, and nonlinear programming use the same basic strategy as simplex of making an explicit estimate of the active set and taking a step toward the solution of a reduced problem in which the constraints in this estimated active set are satisfied as equalities. When nonlinearity enters the problem, many of the features that make the simplex method so effective no longer apply. For example, it is no longer true in general that at least $n - m$ of the bounds $x \geq 0$ are active at the solution, and the specialized linear algebra techniques described in Section 13.5 no longer apply. Nevertheless, the simplex method is rightly viewed as the antecedent of the active set class of methods for constrained optimization.

One undesirable feature of the simplex method attracted attention from its earliest days. Though highly efficient on almost all practical problems (the method generally requires at most $2m$ to $3m$ iterations, where m is the row dimension of the constraint matrix in (13.1)), there are pathological problems on which the algorithm performs very poorly. Klee and Minty [182] presented an n-dimensional problem whose feasible polytope has 2^n vertices, for which the simplex method visits every single vertex before reaching the optimal point! This example verified that the complexity of the simplex method is *exponential*;

roughly speaking, its running time may be an exponential function of the dimension of the problem. For many years, theoreticians searched for a linear programming algorithm that has *polynomial complexity*, that is, an algorithm in which the running time is bounded by a polynomial function of the amount of storage required to define the problem. In the late 1970s, Khachiyan [180] described an *ellipsoid method* that indeed has polynomial complexity but turned out to be impractical. In the mid-1980s, Karmarkar [175] described a polynomial algorithm that approaches the solution through the interior of the feasible polytope rather than working its way around the boundary as the simplex method does. Karmarkar's announcement marked the start of intense research in the field of *interior-point methods*, which are the subject of the next chapter.

NOTES AND REFERENCES

The standard reference for the simplex method is Dantzig's book [86]. Later excellent texts include Chvátal [61] and Vanderbei [293].

Further information on steepest-edge pivoting can be found in Goldfarb and Reid [134] and Goldfarb and Forrest [133].

An alternative procedure for performing the Phase-I calculation of an initial basis was described by Wolfe [310]. This technique does not require artificial variables to be introduced in the problem formulation, but rather starts at any point x that satisfies $Ax = b$ with at most m nonzero components in x. (Note that we do not require the basic part x_B to consist of all positive components.) Phase I then consists in solving the problem

$$\min_x \sum_{x_i<0} -x_i \quad \text{subject to } Ax = b,$$

and terminating when an objective value of 0 is attained. This problem is *not* a linear program—its objective is only piecewise linear—but it can be solved by the simplex method nonetheless. The key is to *redefine* the cost vector f at each iteration x such that $f_i = -1$ for $x_i < 0$ and $f_i = 0$ otherwise.

EXERCISES

13.1 Convert the following linear program to standard form:

$$\max_{x,y} c^T x + d^T y \;\text{ subject to } A_1 x = b_1,\; A_2 x + B_2 y \le b_2,\; l \le y \le u,$$

where there are no explicit bounds on x.

13.2 Verify that the dual of (13.8) is the original primal problem (13.1).

✎ **13.3** Complete the proof of Theorem 13.1 by showing that if the dual (13.7) is unbounded above, the primal (13.1) must be infeasible.

✎ **13.4** Theorem 13.1 does not exclude the possibility that both primal and dual are infeasible. Give a simple linear program for which such is the case.

✎ **13.5** Show that the dual of the linear program

$$\min\ c^T x \ \text{ subject to } Ax \geq b,\ x \geq 0,$$

is

$$\max\ b^T \lambda \ \text{ subject to } A^T \lambda \leq c,\ \lambda \geq 0.$$

✎ **13.6** Show that when $m \leq n$ and the rows of A are linearly dependent in (13.1), then the matrix B in (13.13) is singular, and therefore there are no basic feasible points.

✎ **13.7** Consider the overdetermined linear system $Ax = b$ with m rows and n columns ($m > n$). When we apply Gaussian elimination with complete pivoting to A, we obtain

$$PAQ = L \begin{bmatrix} U_{11} & U_{12} \\ 0 & 0 \end{bmatrix},$$

where P and Q are permutation matrices, L is $m \times m$ lower triangular, U_{11} is $\bar{m} \times \bar{m}$ upper triangular and nonsingular, U_{12} is $\bar{m} \times (n - \bar{m})$, and $\bar{m} \leq n$ is the rank of A.

(a) Show that the system $Ax = b$ is feasible if the last $m - \bar{m}$ components of $L^{-1}Pb$ are zero, and infeasible otherwise.

(b) When $\bar{m} = n$, find the unique solution of $Ax = b$.

(c) Show that the reduced system formed from the first $\bar{m}$ rows of PA and the first $\bar{m}$ components of Pb is equivalent to $Ax = b$ (i.e., a solution of one system also solves the other).

✎ **13.8** Verify formula (13.37).

✎ **13.9** Consider the following linear program:

$$\begin{aligned} \min\ -5x_1 - x_2 \quad & \text{subject to} \\ x_1 + x_2 &\leq 5, \\ 2x_1 + (1/2)x_2 &\leq 8, \\ x &\geq 0. \end{aligned}$$

(a) Add slack variables x_3 and x_4 to convert this problem to standard form.

(b) Using Procedure 13.1, solve this problem using the simplex method, showing at each step the basis and the vectors λ, s_N, and x_B, and the value of the objective function. (The initial choice of $\mathcal{B}$ for which $x_B \geq 0$ should be obvious once you have added the slacks in part (a).)

✎ **13.10** Calculate the values of l_{52}, l_{53}, l_{54}, and $\hat{w}_2$ in (13.30), by equating the last row of L_1U_1 to the last row of the matrix in (13.29).

✎ **13.11** By extending the procedure (13.27) appropriately, show how the factorization (13.31) can be used to solve linear systems with coefficient matrix B^+ efficiently.