G.L. Nemhauser et al., Eds., *Handbooks in OR & MS, Vol. 1*

Chapter II

Linear Programming

*Donald Goldfarb**
Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027, U.S.A.

*Michael J. Todd***
School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853, U.S.A.

1. Introduction

Although the origin of linear programming as a mathematical discipline is quite recent, linear programming is now well established as an important and very active branch of applied mathematics. The wide applicability of linear programming models and the rich mathematical theory underlying these models and the methods developed to solve them have been the driving forces behind the rapid and continuing evolution of the subject.

Linear programming problems involve the optimization of a linear function, called the objective function, subject to linear constraints, which may be either equalities or inequalities, in the unknowns. The recognition of the importance of linear programming models, especially in the areas of economic analysis and planning, coincided with the development of both an effective method, the 'simplex method' of G.B. Dantzig, for solving linear programming problems, (Dantzig 1951) and a means, the digital computer, for doing so. A major part of the foundation of linear programming was laid in an amazingly short period of intense research and development between 1947 and 1949, as the above three key factors converged.

Prior to 1947 mathematicians had studied systems of linear inequalities, starting with Fourier (1826), and optimality conditions for systems with inequality constraints within the classical theory of the calculus of variations (Bolza 1914; Valentine 1937). For the finite dimensional case, the first general result of the latter type appeared in a master's thesis by Karush (1939). (See also (John 1948).) Also, as early as 1939, L.V. Kantorovich had proposed linear programming models for production planning and a rudimentary algorithm for their solution (Kantorovich

*This work was partially supported by NSF Grant DMS-85-12277 and ONR Contract N00014-87-K-0214.

**This work was partially supported by NSF Grant ECS-8602534 and ONR Contract N00014-87-K-0212.

1939). However, Kantorovich's work was ignored in the U.S.S.R. and remained unknown in the West until long after linear programming had been well established. For a thorough historical survey of linear programming see (Dantzig 1963) and (Schrijver 1986).

In the last decade, linear programming has again become a major focus of attention and an area of heightened activity. This is a result of two developments, both of which are concerned with linear programming algorithms which differ radically from the simplex method. The first was a proof by Khachian (1979) that the so-called ellipsoid method of Shor (1970) and Yudin and Nemiroviskii (1976) for convex, not necessarily differentiable, programming could solve linear programming problems quickly in a theoretical sense. The second was the development by Karmarkar (1984a, b) of a projective interior-point algorithm which appears to have enormous potential for efficiently solving very large problems.

In this chapter, we present and analyze these new methods for solving linear programs (i.e., linear programming problems) as well as providing a thorough development of the simplex method and the basic theory of linear programming. Our point-of-view is both algorithmic and geometric, and we discuss practical, computational issues and give economic interpretations of several aspects of linear programming. We cover most of the standard topics found in textbooks on linear programming. We do not, however, treat specialized applications of linear programming such as game theory, or extensions of it such as integer or quadratic programming. The latter two subjects are discussed in Chapters 6 and 3.

In the remainder of this section we present three standard examples of linear programming problems and introduce several canonical forms for linear programs. Section 2 presents the geometry of linear programming models and algebraically characterizes relevant geometrical concepts. Basic results concerning the fundamental role played by the vertices of the polyhedron of feasible solutions of a linear program are given. In Section 3 we develop the simplex method from a geometric point of view. This development produces, as a by-product, various basic theorems concerning conditions for optimality and unboundedness, and leads in a natural way to the so-called *revised* (i.e., matrix) version of the simplex method. Degeneracy and cycling are briefly considered as are various approaches for implementing the simplex method. These include the standard 'tableau' approach as well as factorizations for the basis matrix in the revised version of the simplex method. This section concludes with a discussion of the use of artificial variables and the so-called phase I problem for obtaining an initial feasible (vertex) solution for the simplex method.

Duality theory and sensitivity analysis are treated in Section 4. In addition to showing that the 'dual' of a linear program arises naturally from the optimality conditions for the latter problem, we show that the dual and its constraints and variables can be given an economic interpretation. The duals of two of the linear programming examples considered in Section 1 are presented, and an important variant of the simplex method, the dual simplex algorithm, is derived. We conclude Section 4 with a discussion of sensitivity (or postoptimality) analysis.

Sections 5 and 6 consider efficient application of the revised simplex method to large and structured problems. Section 5 describes the approach of using a

compact (partitioned) inverse which is useful when there are special constraints such as generalized or variable upper bounds, and illustrates the method in the case of simple upper bounds. Efficient implementation of the simplex method for solving network flow problems is also discussed. Section 6 addresses column generation and the use of the decomposition principle to reduce very large problems to ones that are of manageable size. The former technique, which allows columns (of which there may be an astronomical number) to be generated as needed, is illustrated on the classic cutting-stock problem.

The final three sections discuss the complexity of linear programming (Section 7), and two new methods, the ellipsoid method (Section 8) and Karmarkar's projective method (Section 9), which are distinguished from the simplex method in that they have the desirable theoretical property of polynomial-time boundedness. For each of these methods, we describe the basic idea of the method, provide a precise statement of the algorithm, give a sketch of the proof that the algorithm requires only polynomial time to solve linear programming problems and discuss some extensions and the theoretical and computational significance of the method.

1.1. Examples of linear programming problems

Very many problems of practical interest can be formulated as linear programs. In this section we present several well-known examples of such problems.

Example 1 (The transportation problem). A company needs to ship a product from m locations (origins) to n destinations. Suppose that a_i units of the product are available at the i-th origin, $i = 1, 2, \ldots, m$, and b_j units are required at the j-th destination, $j = 1, 2, \ldots, n$. Further suppose that the total amount available at the origins equals the total amount required at the destinations, i.e.,

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \, .$$

If the cost of shipping one unit of product from origin i to destination j is c_{ij}, $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$, how many units of product should be shipped between each origin-destination pair so as to minimize the total transportation cost?

Defining x_{ij} to be the number of units of product shipped from origin i to

destination j, we can formulate this problem as the linear program:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{j=1}^{n} x_{ij} = a_i\,, \quad i = 1,2,\ldots,m\,, && (1.1)\\
& \sum_{i=1}^{m} x_{ij} = b_j\,, \quad j = 1,2,\ldots,n\,, && (1.2)\\
& x_{ij} \geqslant 0\,, \quad i = 1,2,\ldots,m\,, \quad j = 1,2,\ldots,n\,.
\end{aligned}$$

The objective function that is being minimized is clearly equal to the total shipping cost. Each of the quality constraints (1.1) represents the requirement that the total amount of product shipped from origin i to all destinations is equal to the amount available at the origin. Similarly the constraints (1.2) express the requirement that the demand b_j at destination j is exactly satisfied by the amounts shipped there from all origins. Notice that the nonnegativity restrictions on the amounts shipped are crucial since otherwise one could save money by shipping negative quantities along some routes.

The transportation problem is a linear programming problem with a rather special structure; all coefficients of the decision variables in the equality constraints (1.1) and (1.2) are either zero or one. As we shall see later, the transportation problem is a special case of a network flow problem. It also illustrates why linear programs that are solved in practice often tend to be quite large. For example, if both m and n are 100, then the above problem contains 200 equations in 10 000 nonnegative variables. Because of their special structure, very large transportation problems can be solved in a reasonable amount of computer time; in fact, the solution of a problem with 63 million variables was reported several years ago.

Example 2 (The diet problem). Consider the problem of determining the most economical diet that satisfies certain minimum daily requirements for calories and such nutrients as proteins, calcium, iron and vitamins. Suppose there are n different foods available and our diet must satisfy m minimum nutritional requirements. We can also have maximum requirements on certain nutrients such as fats and carbohydrates, but we will ignore these in our example. Let c_j be the unit cost of the j-th food, b_i be the minimum daily requirement of the i-th nutrient, and a_{ij} be the amount of nutrient i provided by a unit of food j.

If we let x_j be the number of units of food j included in our diet then a minimum cost diet can be found by solving the linear programming problem:

$$\begin{aligned}
\text{minimize} \quad & \sum_{j=1}^{n} c_j x_j \\
\text{subject to} \quad & \sum_{j=1}^{n} a_{ij}x_j \geqslant b_i\,, \quad i = 1,2,\ldots,m\,. \\
& x_j \geqslant 0\,, \quad j = 1,2,\ldots,n\,.
\end{aligned}$$

Example 3 (Product mix problem). A manufacturer is capable of producing n different products using m different limited resources. These may be hours of labor or operation times for various machines per week, or material availabilities. Let c_j be the profit (revenue minus cost) obtainable from each unit of product j manufactured, b_i be the amount of resource i available and a_{ij} the amount of resource i used in the production of one unit of product j. The problem facing the manufacturer is one of determining the product mix (i.e., production plan) that maximizes total profit.

If we let x_j be the number of units of product j manufactured, then this linear programming problem can be formulated as:

$$\begin{aligned} &\text{maximize} && \sum_{j=1}^{n} c_j x_j \\ &\text{subject to} && \sum_{j=1}^{n} a_{ij} x_j \leqslant b_i\,, \quad i = 1, 2, \ldots, m\,, \\ & && x_j \geqslant 0\,, \quad j = 1, 2, \ldots, n\,. \end{aligned}$$

1.2. Canonical forms

Linear programs are usually expressed in either of two forms (however, see Section 8 for an exception). These are the *inequality form*

$$\begin{aligned} &\text{minimize} && z = c^{\mathrm{T}} x \\ &\text{subject to} && Ax \leqslant b\,, \\ & && x \geqslant 0\,, \end{aligned}$$

and the *standard form*

$$\begin{aligned} &\text{minimize} && z = c^{\mathrm{T}} x \\ &\text{subject to} && Ax = b\,, \\ & && x \geqslant 0\,, \end{aligned}$$

where A in both cases denotes an $(m \times n)$ matrix, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $x \in \mathbb{R}^n$ is the n-vector of variables. These forms are completely equivalent and any linear program can be put into either form using the following simple transformations.

A *free*, or unrestricted variable x_j can be replaced by a pair of non-negative variables, $x'_j \geqslant 0$, $x''_j \geqslant 0$ by writing

$$x_j = x'_j - x''_j\,.$$

The sense of an inequality can be reversed by multiplying both sides of it by minus one. Further, an inequality

$$\sum_{j=1}^{n} a_{ij} x_j \leqslant b_i$$

can by converted to an equality by the addition of a nonnegative *slack* variable

$$x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij}x_j$$

and an equality

$$\sum_{j=1}^{n} a_{ij}x_j = b_i$$

can be replaced by two inequalities

$$\sum_{j=1}^{n} a_{ij}x_j \lessgtr b_i$$

Finally, maximizing the linear function $c^{\mathrm{T}}x$ is equivalent to minimizing $-c^{\mathrm{T}}x$.

Observe that the first example of Section 1.1, the transportation problem, is in standard form, while the other examples, the diet and product mix problems are essentially in inequality form.

Slack variables can usually be given some economic or physical interpretation. For example, if we add a slack variable x_{n+i} to the i-th inequality in the product mix example to make it an equality then x_{n+i} is the amount of resource i not used in the production of the product mix.

2. Geometric interpretation

In order to understand the theory underlying linear programming and the methods used to solve linear programming problems, it is essential to have a geometric understanding of these problems and be able to algebraically characterize relevant geometrical concepts. With this in mind we now give several definitions.

2.1. Definitions

Definition 2.1. A set $C \subseteq \mathbb{R}^n$ is *convex* if for any two points x', $x'' \in C$, all points of the form $x(\lambda) = \lambda x' + (1-\lambda)x''$, where $0 \leqslant \lambda \leqslant 1$, are in C.

That is, for any two points in the set, the line segment between these points must lie in the set for in to be convex (see Figure 2.1).

Definition 2.2. x is a *convex combination* of points $x_1, \ldots, x_N$ if

$$x = \sum_{i=1}^{N} \lambda_i x_i, \quad \lambda_i \geqslant 0, \text{ all } i \quad \text{and} \quad \sum_{i=1}^{N} \lambda_i = 1\,.$$

Definition 2.3. A set $C \subseteq \mathbb{R}^n$ is a *cone* if for any point $x \in C$ and any nonnegative scalar λ, the point λx is in C. The set $\{x \in \mathbb{R}^n : x = A\alpha,\ \alpha \geqslant 0\}$ is a *convex cone generated by the columns of* $A \in \mathbb{R}^{n\times m}$. (Here, $\alpha \in \mathbb{R}^m$.)

Fig. 2.1.

Definition 2.4. An *extreme point* of a convex set C is a point $x \in C$ which cannot be expressed as a convex combination of (two) other points in C.

Definition 2.5. The set $H = \{x \in \mathbb{R}^n : a^{\mathrm{T}}x = \beta\}$ where $a \in \mathbb{R}^n$, $a \neq 0$, and $\beta \in \mathbb{R}$ is a *hyperplane. The set* $\bar{H} = \{x \in \mathbb{R}^n : a^{\mathrm{T}}x \leqslant \beta\}$ is a *closed half-space.* The hyperplane H associated with the half-space $\bar{H}$ is referred to as the *bounding hyperplane* for that half-space.

The vector a in the above definition is orthogonal to the hyperplane H and is called its *normal*, and it is directed towards the exterior of $\bar{H}$. To see this let y, $z \in H$ and $w \in \bar{H}$. Then

$$a^{\mathrm{T}}(y - z) = a^{\mathrm{T}}y - a^{\mathrm{T}}z = \beta - \beta = 0\,,$$

i.e., a is orthogonal to all vectors parallel to H, and

$$a^{\mathrm{T}}(w - z) = a^{\mathrm{T}}w - a^{\mathrm{T}}z \leqslant \beta - \beta = 0\,,$$

i.e., a makes an obtuse angle whit any vector which points towards the interior of $\bar{H}$. A hyperplane in $\mathbb{R}^n$ is just an $(n-1)$-dimensional *affine set* (or *affine subspace* of $\mathbb{R}^n$) which is defined as:

Definition 2.6. A set $S_a \subseteq \mathbb{R}_n$ is an *affine set* if for any two points $x', x'' \in S_a$, all points of the form $x(\lambda) = \lambda x' + (1 - \lambda)x''$ where $-\infty < \lambda < \infty$, are in S_a.

Note that in contrast with the definition of a convex set, given any two points in an affine set we have that the entire line passing through them lies in that set rather than just the line segment between them. We also note that an affine set S_a is simply a linear subspace S translated by a vector y; i.e., $S_a = \{y + x : x \in S\}$. We say S_a is *parallel* to S.

Definition 2.7. A *convex polyhedron* is a set formed by the intersection of a finite number of closed half-spaces (and hyperplanes). If it is non-empty and bounded it is called *convex polytope*, or simply a *polytope.*

It is easy to show that hyperplanes and closed half-spaces are convex and that the intersection of convex sets; hence, a *convex polyhedron* as defined above is convex. Clearly, the set of feasible solutions of a linear programming problem,

$P = \{x \in \mathbb{R}^n : Ax \leqslant b, x \geqslant 0\}$ (or $\bar{P} = \{x \in \mathbb{R}^n : Ax = b, x \geqslant 0\}$) is a convex polyhedron since it is the intersection of the half-spaces defined by the inequalities

$$a_1^{\mathrm{T}} x \leqslant b_1, \ldots, a_m^{\mathrm{T}} x \leqslant b_m \qquad \text{and} \qquad e_1^{\mathrm{T}} x \geqslant 0, \ldots, e_n^{\mathrm{T}} x \geqslant 0 \tag{2.1}$$

where a_i^{T} is the i-th row of A and e_i^{T} is the i-th row of the $n \times n$ identity matrix.

Before we can define certain important features of convex polyhedra we need the following two definitions.

Definition 2.8. The *dimension* of a subspace S, and any affine subspace S_a parallel to it, is equal to the maximum number of linearly independent vectors in S. The *dimension* of any subset of $D \subseteq R^n$ is the smallest dimension of any affine subspace which contains D.

Definition 2.9. A *supporting hyperplane* of a convex set C is a hyperplane H such that $H \cap C \neq \emptyset$ and $C \subseteq \bar{H}$, one of the two closed half-spaces associated with H.

Definition 2.10. Let P be a convex polyhedron and H be any supporting hyperplane of P. The intersection $F = P \cap H$ defines a *face* of P.

There are three special kinds of faces.

Definition 2.11. A *vertex, edge*, and *facet* are faces of a d-dimensional convex polyhedron of dimension zero, one, and $d - 1$, respectively.

If $P = \{x \in \mathbb{R}^n : Ax \leqslant b, x \geqslant 0\}$ then it is fairly obvious that every facet of P corresponds to the intersection of P with a half-space defined by one of the inequalities (2.1). However, not all such intersections necessarily define facets since some of the inequalities may be *redundant*; i.e., deleting them from the definition of P does not change P.

Vertices of a convex polyhedron P are obviously extreme points of P and we shall henceforth use these terms interchangeably. A rigorous proof of this equivalence is left to the reader. Edges are either line segments which connect *neighboring* (or *adjacent*) vertices or are semi-infinite lines emanating from a vertex.

2.2. Extreme points and basic feasible solutions

It is easy to see that if a linear programming problem in two or three variables has a finite optimal solution then it occurs at a vertex (i.e., extreme point) of the polyhedron P of feasible solutions. As we shall prove in the next section, this statement holds in higher dimension as well. For this reason, we now algebraically characterize the vertices of P. For the remainder of this section we shall use P to denote the polyhedron $\{x \in \mathbb{R}^n : Ax = b, x \geqslant 0\}$.

Theorem 2.1. *A point $x \in P = \{x \in \mathbb{R}^n : Ax = b, x \geqslant 0\}$ is a vertex of P if and only if the columns of A corresponding to positive components of x are linearly independent.*

Proof. Without loss of generality, let us assume that the first p components of x are positive and the last $n-p$ components of x are zero. If we partition x so that $x = \binom{\bar{x}}{0}$, $\bar{x} > 0$, and we denote the first p columns of A by $\bar{A}$, then $Ax = \bar{A}\bar{x} = b$.

Suppose that the columns of $\bar{A}$ are *not* linearly independent. Then, there exists a vector $\bar{w} \neq 0$ such that $\bar{A}\bar{w} = 0$. Therefore, $\bar{A}(\bar{x} \pm \varepsilon\bar{w}) = \bar{A}\bar{x} = b$ and for small enough ε, $(\bar{x} \pm \varepsilon\bar{w}) \geqslant 0$. Consequently the points

$$y' = \begin{pmatrix} \bar{x} + \varepsilon\bar{w} \\ 0 \end{pmatrix} \qquad \text{and} \qquad y'' = \begin{pmatrix} \bar{x} - \varepsilon\bar{w} \\ 0 \end{pmatrix}$$

are both in P. Further, since $x = \frac{1}{2}(y' + y'')$, x cannot be a vertex (extreme point) of P. Thus, if x is a vertex, then the columns of $\bar{A}$ are linearly independent.

Now suppose that x is not a vertex. This means that $x = \lambda y' + (1-\lambda)y''$ where y', $y'' \in P$, $y' \neq y''$ and $0 < \lambda < 1$. Since both x and y' are in P, $A(x - y') = Ax - Ay' = b - b = 0$. Further, since both λ and $1 - \lambda$ are strictly positive, the last $n - p$ components of y', and hence $x - y'$, must be zero, since those components of x are zero. Therefore, it follows that the columns of $\bar{A}$ are linearly dependent. Thus, if the columns of $\bar{A}$ are linearly independent, then x is a vertex.

When A has full row rank, an equivalent characterization of the vertices of P involves the concept of a *basic solution*.

Definition 2.12. Let B be any nonsingular $m \times m$ matrix composed of m (linearly independent) columns of A. If all components of x not associated with the columns of B, called *nonbasic variables*, are set equal to zero and the set of linear equations $Ax = b$ is solved for the remaining components of x, called *basic variables*, then the resulting x is said to be a *basic solution* with respect to the *basis* (matrix) B. We shall also use the term *basis* to refer both to the set of basic variables and the set of indices of those variables.

Notice that if we set the nonbasic variables equal to zero, we are left with a system of m equations in m unknowns.

$$Bx_B = b\,,$$

which is uniquely solvable for the basic variables x_B. The reason for the above terminology is that the columns of B form a basis for the column space of A and $Ax = b$ can be thought of as expressing b as a linear combination of the columns of A.

When A does not have full row rank, either the system of linear equations $Ax = b$ has no solution and P is the empty set, or some of the equations in the system are *redundant*. In the latter case, redundant constraints can be removed one by one to give a reduced system of equations and a constraint matrix of full row rank.

If a basic solution x with respect to a basis B is nonnegative then it is called a *basic feasible solution*, and the following corollary to Theorem 2.1 is an immediate consequence of the above definitions.

Corollary 2.1. *A point $x \in P$ is a vertex of P if and only if x is a basic feasible solution corresponding to some basis B.*

Corollary 2.2. *The polyhedron P has only a finite number of vertices.*

Proof. This follows from the previous corollary and the fact that there are only a finite number of ways to choose m linearly independent 'basis' columns from the n columns of A. Clearly an upper bound on the number of vertices of P is $n!/(m!(n-m)!)$.

If the polyhedron P is bounded—i.e., P is a polytope—then any point in P can be represented as a convex combination of the vertices of P. (See Corollary 2.3 below.) When P is unbounded the representation of any point in P is slightly more complicated and requires the following definition.

Definition 2.13. A *direction* of P is a nonzero vector $d \in \mathbb{R}^n$, such that for any point $x_0 \in P$ the ray $\{x \in \mathbb{R}^n : x = x_0 + \lambda d,\ \lambda \geqslant 0\}$ lies entirely in P.

Obviously P is unbounded if and only if P has a direction. It is also easily proved that $d \neq 0$ is a direction of P if and only if

$$Ad = 0 \qquad \text{and} \qquad d \geqslant 0\,.$$

We can now state the following representation theorem.

Theorem 2.2 (Representation theorem). *Any point $x \in P$ can be represented as*

$$x = \sum_{i \in I} \lambda_i v_i + d\,,$$

where $\{v_i : i \in I\}$ is the set of vertices of P,

$$\sum_{i \in I} \lambda_i = 1\,, \quad \lambda_i \geqslant 0 \text{ for all } i \in I,$$

and either d is a direction of P or $d = 0$.

Proof. We prove this theorem by induction on p, the number of positive components of x. It is obviously true for $p = 0$ (x is a vertex). Now assume that it is true for points with fewer than p positive components, and that x has p positive components.

If x is a vertex, then the theorem is obviously true since $x = v_i$ for some $i \in I$. Therefore suppose that x is not a vertex. Then there is a vector $w \neq 0$ with $w_i = 0$ if $x_i = 0$ such that $Aw = 0$. There are three cases to consider.

Case (a): w has components of both signs.

Consider points $x(\theta) = x + \theta w$ on the line through x determined by w, and let θ' and θ'' be, respectively, the smallest positive, and (algebraically) largest negative values of θ at which $x(\theta)$ has at least one more zero component than x.

Clearly the points $x' = x(\theta')$ and $x'' = x(\theta'')$ lie in P and can be represented as in the statement of the theorem by the induction hypothesis. Consequently, we can represent x, which lies on the line between x' and x'', as

$$\begin{aligned} x &= \mu x' + (1-\mu)x'' \\ &= \mu\left(\sum_{i\in I}\lambda_i' v_i + d'\right) + (1-\mu)\left(\sum_{i\in I}\lambda_i'' v_i + d''\right) \\ &= \sum_{i\in I}\left(\mu\lambda_i' + (1-\mu)\lambda_i''\right)v_i + \mu d' + (1-\mu)d'' \,, \end{aligned}$$

where $\mu = -\theta''/(\theta' - \theta'')$.

Since $0 < \mu < 1$,

$$\begin{aligned} &\lambda_i' \geqslant 0 \text{ and } \lambda_i'' \geqslant 0 \quad \text{for all } i \in I, \qquad \sum_{i\in I}\lambda_i' = \sum_{i\in I}\lambda_i'' = 1\,, \\ &Ad' = Ad'' = 0, \qquad d' \geqslant 0 \quad \text{and} \quad d'' \geqslant 0\,, \end{aligned}$$

it follows that

$$\begin{aligned} &\lambda_i \equiv \mu\lambda_i' + (1-\mu)\lambda_i'' \geqslant 0 \quad \text{for all } i \in I, \qquad \sum_{i\in I}\lambda_i = 1\,, \\ &d \equiv \mu d' + (1-\mu)d'' \geqslant 0 \quad \text{and} \quad Ad = 0\,, \end{aligned}$$

and we have proved that x has the desired form.

Case (b): $w \leqslant 0$.

Define x' as in Case (a). Now x can be written as

$$x = x' + \theta'(-w) \quad \text{where} \quad \theta' > 0.$$

Since x' can be represented in the desired form by induction, and $(-w)$ is a direction of P, x clearly has the desired form.

Case (c): $w \geqslant 0$.

The proof for this case is identical to case (b) with x', θ', and $-w$ replaced by x'', $-\theta''$, and w, respectively.

Hence we obtain:

Corollary 2.3. *If P is bounded (i.e., a polytope) then any $x \in P$ can be represented as a convex combination of its vertices.*

2.3. Fundamental theorems of linear programming

In this section we prove two theorems that are of fundamental importance to the development of algorithms (and in particular the simplex algorithm) for solving linear programs. Specifically, these theorems identify the special importance of the vertices of P—i.e., basic feasible solutions—for such methods.

Theorem 2.3. *If P is nonempty, then it has at least one vertex.*

Proof. This follows immediately from Theorem 2.2 and its proof.

Theorem 2.4. *If P is nonempty, then the minimum value of $z = c^{\mathrm{T}}x$ for $x \in P$ is attained at a vertex of P or z has no lower bound on P.*

Proof. There are two cases to consider:

Case (a): P has a direction d such that $c^{\mathrm{T}}d < 0$. In this case P is unbounded and the value of $z \to -\infty$ along the direction d.

Case (b): P has no direction d such that $c^{\mathrm{T}}d < 0$. In this case we need only consider points that can be expressed as convex combination of the vertices v_i of P, since even if P is unbounded any point of the form $x = \hat{x} + d$, where

$$\hat{x} = \sum_{i \in I} \lambda_i v_i, \qquad \sum_{i \in I} \lambda_i = 1, \quad \text{and} \quad \lambda_i \geqslant 0 \text{ for all } i \in I\,,$$

has an objective value that is bounded below by $c^{\mathrm{T}}\hat{x}$. But

$$c^{\mathrm{T}}\hat{x} = c^{\mathrm{T}}\left[\sum_{i \in I} \lambda_i v_i\right] = \sum_{i \in I} \lambda_i c^{\mathrm{T}} v_i \geqslant \min_{i \in I}\{c^{\mathrm{T}}v_i\}\,.$$

Hence, the minimum of z is attained at a vertex.

This theorem is fundamental to solving linear programming problems. It shows that we need only consider vertices of P, i.e., basic feasible solutions, as candidates for the optimal solution. Also it shows that we must be on the lookout for directions along which $z \to -\infty$.

3. The simplex method

We saw in Section 2 that to solve the linear programming problem,

$$\begin{array}{ll} \text{minimize} & z = c^{\mathrm{T}}x \\ \text{subject to} & Ax = b\,, \\ & x \geqslant 0\,, \end{array} \tag{3.1}$$

we need only consider the vertices of the polyhedron $P = \{x : Ax = b, x \geqslant 0\}$ of feasible solutions as candidates for the optimal solution, assuming for the moment that (3.1) has a finite optimal solution. For large m and n, determining all of the vertices of P is impractical; P can have as many as $\binom{n}{m} = n!/m!(n-m)!$ basic solutions. Clearly a more systematic approach such as the simplex method developed by George Dantzig in 1947 is required. In fact, the simplex method has been so successful that it has become one of the best known and, in terms of computer time, one of the most used methods in numerical computing.

3.1. Geometric motivation

The idea of the simplex method is really quite simple. First a vertex of P is found. Then the method proceeds from vertex to vertex along edges of P that are 'downhill' with respect to the objective function $z = c^{\mathrm{T}}x$, generating a sequence of vertices with strictly decreasing objective values. Consequently, once the method

leaves a vertex the method can never return to that vertex. Thus, in a finite number of steps, a vertex will be reached which is optimal, or an edge will be chosen which goes off to infinity and along which z goes to $-\infty$.

Our task here is to convert the above geometric description of the simplex method into an algebraic and, hence, computational form. We will consider first, the so-called *second phase* (*phase II*) of the simplex method which assumes that a vertex of P is given, for as we shall see later, the algorithm for this second phase can itself by used to solve the phase I problem of finding a vertex of P if P is nonempty. Also, we shall assume that the rows of A are linearly independent, i.e., the rank of A is m, and that $m < n$, so that our problem is not trivial. If the rank of A is less than m, then either the equality constraints are inconsistent or some of them are redundant and can be removed. Our assumptions ensure that there is at least one basic solution and that a basis matrix B can be formed from the columns of A.

For simplicity in describing a step of the simplex method, let us assume that the components of the current vertex x are ordered so that the first m are basic. That is, the vertex x of P corresponds to the basic feasible solution

$$x = \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} \tag{3.2}$$

where A is partitioned as $A = [B|N]$. We also partition c^{T} as $c^{\mathrm{T}} = [c_B^{\mathrm{T}}\ c_N^{\mathrm{T}}]$ to conform to the above partition of x into basic and nonbasic parts.

Definition 3.1. If one or more basic variables are zero then such a basic solution is called *degenerate*, otherwise, it is called *nondegenerate*.

If the basic feasible solution (3.2) is nondegenerate then it lies in the intersection in $\mathbb{R}^n$ of the m hyperplanes corresponding to the equality constraints $Ax = b$ and the $n - m$ hyperplanes corresponding to requirement that the $n - m$ nonbasic variables equal zero, i.e., $x_N = 0$. Consider the matrix

$$M = \begin{bmatrix} B & N \\ 0 & I \end{bmatrix} \tag{3.3}$$

whose rows are just the normals to these n hyperplanes. Since B is non-singular, the rows of M are linearly independent, and hence M is non-singular. Thus the vertex (3.2) is determined by the intersection of n linearly independent hyperplanes.

If a basic feasible solution is degenerate, then some of the basic variables x_B are also equal to zero. Consequently, more than $n-m$ of the nonnegativity constraints $x_j \geqslant 0$ are satisfied as equalities and the point x satisfies more than n equations.

A conceptual illustration of degeneracy is given in Figure 3.1. It might appear from this illustration that there are always redundant constraints at a degenerate vertex. However, this is only the case when $n - m \leqslant 2$.

When a basic feasible solution x is degenerate, there can be an enormous number of bases associated with the vertex x. In fact, if x has $p < m$ positive components, there may be as many as

$$\binom{n-p}{n-m} = \frac{(n-p)!}{(n-m)!(m-p)!}$$

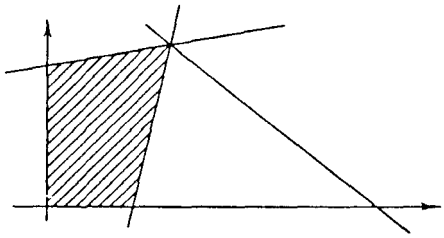

Fig. 3.1. An illustration of degeneracy.

'different' basic feasible solutions corresponding to x. The point x is the same in each, but the sets of variables that we label basic and nonbasic are different. An extreme example of degeneracy is exhibited by the so-called 'assignment' problem. It can be shown that the polytope

$$P_k = \left\{ x_{ij}, 1 \leqslant i,j \leqslant k : \sum_{j=1}^{k} x_{ij} = 1, 1 \leqslant i \leqslant k; \right.$$
$$\left. \sum_{i=1}^{k} x_{ij} = 1, 1 \leqslant j \leqslant k;\ 0 \leqslant x_{ij}, 1 \leqslant i,j \leqslant k \right\}$$

of this rather special 'capacitated transportation' problem has $k!$ vertices, and that there are $2^{k-1}k^{k-2}$ bases corresponding to each of these vertices. Thus, for $k = 8$, each of the 40 320 vertices of P_8 has 33 554 432 different bases associated with it (Balinski and Russakoff 1972).

Let us assume for simplicity that the basic feasible solution (3.2) is nondegenerate. This ensures that there are exactly $n-m$ edges (i.e., one-dimensional faces) of P emanating from the vertex. The directions of these edges are given by the last $n-m$ columns of the inverse of the matrix of active constraint normals M in (3.3). It is easily verified that

$$M^{-1} = \begin{bmatrix} B^{-1} & -B^{-1}N \\ 0 & I \end{bmatrix}. \tag{3.4}$$

Each edge direction corresponds to increasing one of the nonbasic variables while keeping all of the remaining nonbasic variables fixed at zero. To verify the above statements we observe that the q-th column of M^{-1}, $q > m$, is orthogonal to all rows of M other than the q-th, and hence, it is orthogonal to the normals of all of the hyperplanes that intersect at x except the one corresponding to $x_q = 0$. This means that this vector $\eta_q = M^{-1}e_q$ (where e_q is again the q-th column of the $n \times n$ identity matrix) is parallel to the intersection of the $n-1$ linearly independent hyperplanes corresponding to $Ax = b$ and $x_k = 0$, $k > m$, $k \neq q$. Also the edge direction η_q is a feasible direction because, for small enough $\theta > 0$, points of the form

$$x(\theta) = x + \theta\eta_q \tag{3.5}$$

are feasible. In fact $x_k(\theta) = 0$, $k > m$, $k \neq q$, $x_q(\theta) = \theta > 0$ and

$$x_B(\theta) = x_B - \theta B^{-1} a_q \geqslant 0 \tag{3.6}$$

for θ small enough, where a_q denotes the q-th column of A.

Now, the first task in an iteration of the simplex method is to find a 'downhill' edge. This involves computing the so-called *reduced* or *relative costs*

$$\bar{c}_j = c^{\mathrm{T}} \eta_j = c_j - c_B^{\mathrm{T}} B^{-1} a_j, \quad j > m\,.$$

If $\bar{c}_j < 0$, then the gradient c of the objective function $z = c^{\mathrm{T}} x$ makes an obtuse angle with the edge direction η_j and z decreases as one moves along that direction, i.e., as θ is increased. The terminology reduced cost comes from the fact that $\bar{c}_j$ represents the change in the objective function z per unit change in the nonbasic variable x_j, keeping all other nonbasic variable fixed, since from (3.5) with $q = j$, it follows that

$$z(x(\theta)) = c^{\mathrm{T}} x(\theta) = c^{\mathrm{T}} x + \theta c^{\mathrm{T}} \eta_j = z(x) + \theta \bar{c}_j\,.$$

Clearly the reduced cost $\bar{c}_j$ is the directional derivative of $z = c^{\mathrm{T}} x$ with respect to the edge direction η_j.

Although any downhill edge will do for the simplex method, the usual rule used in textbooks is to choose the edge corresponding to the most negative reduced cost. (This is not the 'steepest-edge' with respect to the objective function z. Such a choice η_q, corresponds to

$$\frac{c^{\mathrm{T}} \eta_q}{\|\eta_q\|} = \min_{j>m} \left\{ \frac{c^{\mathrm{T}} \eta_j}{\|\eta_j\|} \right\}\,.$$

That is, the steepest edge is the one which makes the most obtuse angle ϕ_q with c, where

$$\phi_q = \cos^{-1} \left(\frac{c^{\mathrm{T}} \eta_q}{\|c\| \cdot \|\eta_q\|} \right)\,.$$

This pivot rule can be implemented in a practicable manner for large problems if the quantities $\|\eta_j\|^2 = \eta_j^{\mathrm{T}} \eta_j$ for all of the nonbasic variables are stored and updated from one iteration to the next. (See (Goldfarb and Reid 1977).))

Note that the reduced costs corresponding to basic variables are zero and that those corresponding to nonbasic variables can be obtained by first computing the vector of *simplex multipliers* $\pi^{\mathrm{T}} = c_B^{\mathrm{T}} B^{-1}$ followed by 'pricing-out' all nonbasic columns, i.e.,

$$\bar{c}_j = c_j - \pi^{\mathrm{T}} a_j, \quad j > m\,.$$

The terminology used above is derived from the interpretation of the components of π both as Lagrange multipliers and as equilibrium prices at optimality.

We now show that every point $y \in P$ lies within the convex polyhedral cone generated by a given basic feasible solution x and the 'edge directions' η_j emanating from x determined by the basis. In the nondegenerate case these directions are true edge directions. In the degenerate case some of them are infeasible.

Lemma 3.1. *Given the basic feasible solution x in (3.2), every $y \in P$ can be expressed as*

$$y = x + \sum_{j=m+1}^{n} y_j \eta_j, \qquad y_j \geqslant 0, \quad j = m+1, \ldots, n$$

where η_j is the j-th column of M^{-1} in (3.4).

Proof. Since $y \in P$, $Ay = b$ and $y = \begin{pmatrix} y_B \\ y_N \end{pmatrix} \geqslant 0$. Moreover, since $Ax = b$ and $x_N = 0$, it follows that

$$M(y-x) = \begin{bmatrix} B & N \\ 0 & I \end{bmatrix} (y-x) = \begin{pmatrix} 0 \\ y_N \end{pmatrix},$$

and hence that

$$(y-x) = M^{-1} \begin{pmatrix} 0 \\ y_N \end{pmatrix} = \begin{bmatrix} -B^{-1}N \\ I \end{bmatrix} y_N$$

where $y_N \geqslant 0$.

From this lemma we have that

$$z(y) - z(x) = c^{\mathrm{T}}(y-x) = \sum_{j=m+1}^{n} (c^{\mathrm{T}}\eta_j) y_j = \sum_{j=m+1}^{n} \bar{c}_j y_j \tag{3.7}$$

for all $y \in P$. Since y is nonnegative, if the reduced costs $\bar{c}_j$ are nonnegative, it follows that $z(y) \geqslant z(x)$ for all $y \in P$. Thus we have proved:

Theorem 3.1. *A basic feasible solution is an optimal solution to the linear programming problem (3.1) if all* reduced costs (*relative to the given basis*) *are nonnegative.*

In the nondegenerate case the converse of this theorem is true. However a degenerate basic feasible solution can be optimal even if some reduced costs are negative, since the corresponding downhill edge directions may not be feasible. A direction is infeasible at a point x where $x_j = 0$ if its j-th component is negative. The following corollaries are also immediate consequences of (3.7).

Corollary 3.1. *A basic feasible solution x is the unique optimal solution to (3.1) if all nonbasic reduced costs are strictly positive.*

Corollary 3.2. *If x given by (3.2) is an optimal basic feasible solution, with nonbasic reduced costs $\bar{c}_{j_1} = \bar{c}_{j_2} \ldots = \bar{c}_{j_k} = 0$, then any point $y \in P$ of the form*

$$y = x + \sum_{i=1}^{k} y_{j_i} \eta_{j_i} \tag{3.8}$$

is also optimal.

If an optimal basic feasible solution is degenerate and the reduced costs corresponding to some of the nonbasic variables are zero, it does not follow from Corollary 3.2 that the optimal solution is nonunique. This is because in the degenerate case x may be the only point of the form (3.8) that is actually in P, due to the infeasibility of the edge directions η_{j_i} in (3.8).

Once a downhill edge η_d has been chosen the next task in an iteration of the simplex method involves moving along that edge to the vertex adjacent to x. This is accomplished by increasing the nonbasic variable x_q—i.e., increasing θ in (3.5)—until one of the basic variables becomes zero.

Letting

$$w = B^{-1}a_q \tag{3.9}$$

it follows from (3.5)and (3.6) that $x(\theta) \geqslant 0$ if and only if $x_B - \theta w \geqslant 0$ and $\theta \geqslant 0$. Hence we obtain:

Theorem 3.2. *If $\bar{c}_q$ is negative and w in (3.9) is nonpositive, then the linear programming problem (3.1) is unbounded; $x(\theta)$ is feasible for all $\theta \geqslant 0$ and $z(x(\theta)) \to -\infty$ as $\theta \to \infty$. In this case, $d = \eta_q$ is a direction with $c^{\mathrm{T}}d = \bar{c}_q < 0$.*

If w has a positive component, the largest step θ that we can take while still keeping $x(\theta) \geqslant 0$, and the basic variable, say x_p, which first becomes zero as we increase θ are determined by the so-called 'minimum ratio test'

$$\theta = \bar{x}_q = \min\left\{\frac{x_i}{w_i} : w_i > 0, 1 \leqslant i \leqslant m\right\} = \frac{x_p}{w_p}\,. \tag{3.10}$$

We have use an overbar to indicate that $\bar{x}_q$ is the value of the q-th variable at the new vertex. All that remains to be done in a simplex iteration is to change the basis, making the q-th variable basic and the p-th variable nonbasic. As far as the basic matrix B is concerned, one of its columns, a_p, is replaced by the column a_q, i.e.

$$\bar{B} = B + (a_q - a_p)e_p^{\mathrm{T}}\,.$$

From (3.5)–(3.10) it follows that

$$\begin{aligned}\bar{x}_q &= x_p/w_p\,,\\ \bar{x}_i &= x_i - w_i\bar{x}_q\,, \quad i = 1,\ldots,m\,.\end{aligned}$$

Summarizing the above we obtain:

Theorem 3.3. *If $\bar{c}_q$ is negative and w in (3.9) has a positive component, then $\bar{x}$ given above is another basic feasible solution with $c^{\mathrm{T}}\bar{x} = c^{\mathrm{T}}x + \theta\bar{c}_q$ strictly less than $c^{\mathrm{T}}x$ if θ in (3.10) is positive, i.e., if the basic feasible solution x is nondegenerate.*

3.2. The revised simplex method

We are now ready to formally state the simplex method in algorithmic form.

Simplex method

(0) Let the basic feasible solution, x_B, to the linear program (3.1), corresponding to the basis matrix $B = [a_{j_i}, \ldots, a_{j_m}]$, be given. Let $\boldsymbol{B} = \{j_1, \ldots, j_m\}$ denote the index set of basic variables; hence x_{j_i} denotes the i-th basic variable.

(1) Compute *simplex multipliers* by solving

$$B^{\mathrm{T}}\pi = c_B$$

for π, and compute the *reduced costs*

$$\bar{c}_j = c_j - \pi^{\mathrm{T}} a_j, \quad \text{for all } j \notin \boldsymbol{B}.$$

(2) Check for optimality: If $\bar{c}_j \geqslant 0$, for all $j \notin \boldsymbol{B}$, STOP; the current solution is optimal.

(3) Determine the nonbasic variable x_q to enter the basis; i.e., find a downhill edge: Choose

$$q \in V \equiv \{j \notin \boldsymbol{B} : \bar{c}_j < 0\}\,.$$

(4) Check for unbounded ray: Compute w by solving

$$Bw = a_q\,.$$

If $w \leqslant 0$, STOP; there is a feasible ray of solutions along which $z \to -\infty$.

(5) Determine the basic variable x_{j_p} to leave the basis: Compute

$$\frac{x_{j_p}}{w_p} = \min_{1 \leqslant i \leqslant m} \left\{ \frac{x_{j_i}}{w_i} : w_i > 0 \right\}.$$

(6) Update the solution and the basis matrix B: Set

$$\begin{aligned}
&x_q \leftarrow \theta = x_{j_p}/w_p\,,\\
&x_{j_i} \leftarrow x_{j_i} - \theta w_i, \quad 1 \leqslant i \leqslant m\,,\\
&B \leftarrow B + (a_q - a_{j_p})e_p^{\mathrm{T}}\,,\\
&\boldsymbol{B} \leftarrow \boldsymbol{B} \cup \{q\} \setminus \{j_p\}\,,\\
&j_p \leftarrow q\,,
\end{aligned}$$

and go to step (1).

The above form of the simplex method is usually referred to as the *revised simplex method*.

3.3. Degeneracy and cycling

Although we assumed earlier that x nondegenerate, degeneracy does not usually cause any real difficulty for the above algorithm. All that may happen is that in step (5) $x_{j_p} = 0$, which results in a step being taken without any actual change in x. This occurs because the 'edge direction' η_q immediately runs into the constraint $x_{j_p} \geqslant 0$. Although x does not change the basis does. Because x and hence, z, do not change it is theoretically possible for the simplex method to 'cycle' indefinitely through a sequence of bases and corresponding basic feasible solutions, all associated with the same vertex. In practice this is not a problem and there are pivot rules (i.e., rules for choosing the entering and leaving basic variable) which prevent cycling. For example, if one always chooses the entering and leaving basic variables when there is more than one candidate as the one with the smallest subscript, then it can be shown that the simplex method terminates in a finite number of iterations (Bland 1977). No matter what pivot rule is used, if every pivot is nondegenerate—i.e., θ in step (6) is strictly positive—then z decreases on each iteration; consequently, the simplex method must terminate in a finite number of iterations since there are only a finite number of basic feasible solutions to problem (3.1).

3.4. Implementations

For large m it is just not practicable to solve the $m \times m$ systems of equations $B^{\mathrm{T}}\pi = c_B$ and $Bw = a_q$ to compute π and w at each simplex step. In most textbooks the simplex method is described by a set of procedures for manipulating a tableau of the form (actually a column permutation of the form):

1	0	$\bar{c}_N^{\mathrm{T}}$	$-z_0$
0	I	$B^{-1}N$	$B^{-1}b$

(3.11)

where $\bar{c}_N^{\mathrm{T}} = c_N^{\mathrm{T}} - c_B^{\mathrm{T}} B^{-1} N$ and $z_0 = c_B^{\mathrm{T}} B^{-1} b$.

If T is the matrix of numbers in tableau (3.11) then this tableau actually represents the system

$$T \begin{bmatrix} -z \\ x_B \\ x_N \\ -1 \end{bmatrix} = 0$$

of $m + 1$ linear equations in $n + 1$ unknowns x and z.

To implement the tableau version of the simplex method one follows the simplex algorithm presented above except that the computation of π and $\bar{c}_N$ in step (1) and w in step (4) are eliminated and step (6) is replaced by a 'pivot operation'. If we assume that the entering leaving basic variable on a simplex pivot step are q and p, respectively, and that the rows and columns of the current tableau T given by (3.11) are numbered starting from zero, this pivot operation

(i) divides the p-th row of the T by $t_{p,q}$, the (p, q)-th element of T, and

(ii) for $0 \leqslant i \leqslant m$, $i \neq p$, subtracts t_{iq} times this new p-th row from row i to zero out the element in column q of that row.

This operation maintains the form of (3.11) with respect to the new basis. Moreover, the reduced costs, $\bar{c}_N$, basic components of the edge direction η_q, $-B^{-1}a_q$, and the vector of basic variables, $B^{-1}b$, required by the simplex method are all available directly from the tableau. The tableau version of the simplex method is often referred to simply as the simplex method since it was the form in which the method was originally described. Although this approach is acceptable for small 'textbook' problems, it is not suitable for solving the large and typically sparse problems that arise in practice. This is because pivoting in the tableau usually destroys any sparsity that is present in A, and hence in the 'initial' tableau

1	c_B^{T}	c_N^{T}	0
0	B	N	b

Furthermore, it generates all columns of $B^{-1}N$ on each iteration when only $B^{-1}a_j$ is needed.

In the revised simplex method given in the previous section only the information required on each iteration is generated directly from the original data. The initial implementations of the revised simplex method maintained an 'explicit inverse', B^{-1}, of the basis matrix, updating it after each pivot (i.e., basis change). The required update can be expressed as

$$\bar{B}^{-1} = EB^{-1}$$

where

$$E = I - \frac{(w - e_p)e_p^{\mathrm{T}}}{w_p} = \begin{bmatrix} 1 & & & -w_1/w_p & & \\ & \ddots & & \vdots & & \\ & & 1 & -w_{p-1}/w_p & & \\ & & & 1/w_p & & \\ & & & -w_{p+1}/w_p & 1 & \\ & & & \vdots & & \ddots \\ & & & -w_m/w_p & & 1 \end{bmatrix} \quad (3.12)$$

$$\uparrow$$

$$\text{column } p$$

and $w = B^{-1}a_q$.

This follows from the fact that

$$\bar{B} = BE^{-1},$$

where

$$E^{-1} = \begin{bmatrix} 1 & & & w_1 & & & \\ & \ddots & & \vdots & & & \\ & & 1 & w_{p-1} & & & \\ & & & w_p & & & \\ & & & w_{p+1} & 1 & & \\ & & & \vdots & & \ddots & \\ & & & w_m & & & 1 \end{bmatrix}$$

is the inverse of the matrix E in (3.12). Notice that postmultiplication of B by E^{-1} leaves all columns of B unchanged except for the p-th, which is transformed into $Bw = a_q$, as required.

Since B itself can be formed by replacing the columns of I, one at a time, by the appropriate columns of B, it follows that we can express any inverse basis matrix in *product form* as

$$B^{-1} = E_k E_{k-1} \cdots E_1 \tag{3.13}$$

where each E_i has the form (3.12). Clearly only the column of E_i that differs from a column of the identity matrix, and its place in E_i, need to be stored. Every now and then it is also advisable to refactorize B^{-1} to reduce a very long string of elementary elimination matrices, E_i, to one of only m matrices. This saves computational time on subsequent iterations, saves storage, and reduces the effects of roundoff errors.

When B is refactorized it is usually worthwhile to permute its rows and columns so that the resulting factorization is as sparse as possible. Several schemes for doing this have been proposed, the most popular of these being the 'preassigned pivot procedures' P^3 and P^4 of Hellerman and Rarick (1971,1972) and those which use the Markowitz criterion (see Section 9). The procedures P^3 and P^4 permute B into a block lower triangular matrix with nonzeros above the diagonal confined to a relatively small number of spikes within each diagonal block when B is sparse. The advantage of doing this is that when Gauss-Jordan elimination is applied to B to produce the product form representation (3.13) or the LU factors of B (see below), fill-in occurs only in spike columns.

Recently, the product form representation for B^{-1} has been replaced in large-scale linear programming codes by a numerically stable *LU factorization* of B. In this approach L^{-1} is stored as a sequence of elementary elimination matrices which differ from the identity by just one nonzero below the diagonal, and permutation matrices (for numerical stability). U is stored as a permuted upper triangular matrix. Bartels and Golub (1969) first showed how such a factorization could be efficiently and stably updated when a column of B was replaced by a new column. Variants of their algorithm and the product form algorithm which take advantage of sparsity, make it practicable today to solve truly large linear programming problems. (For example, see (Forrest and Tomlin 1972), (Saunders 1976) and (Reid 1982).)

3.5. Artificial variables and Phase I

One nice feature of the simplex method is that it can be used to find a basic feasible solution to problem (3.1). The application of the simplex method to this feasibility problem is called phase I, while its application to finding the optimal solution to (3.1) is called phase II. The feasibility problem for (3.1) can be defined as the (artificial) minimization problem

$$\begin{array}{ll} \text{Minimize} & \displaystyle\sum_{i=1}^{m} y_i \\ \text{subject to} & Ax + y = b \quad (b \geqslant 0)\,, \\ & x \geqslant 0\,, \quad y \geqslant 0\,. \end{array}$$

As this problem has the obvious basic feasible solution $x = 0$, $y = b$, with basis I, the simplex method can be immediately applied to it. The y_i, $i = 1, \ldots, m$, are called *artificial variables* and the purpose of the above minimization problem is to drive them to zero. If the original problem has a feasible solution, then this will be possible. In such a case the simplex method will terminate with a basic feasible solution with all $y_i = 0$. If this solution is degenerate, any artificial variables remaining in the basis can be either exchanged for nonbasic x variables or eliminated along with *redundant* equations so that a basic feasible solution involving only x variables is available for the start of phase II. Specifically, if $y_i = 0$ is the k-th basic variable and $e_k^{\mathrm{T}} B^{-1} a_j \neq 0$, y_i can be replaced by the nonbasic variable x_j. If $e_k^{\mathrm{T}} B^{-1} a_j = 0$ for all $j \notin \boldsymbol{B}$, the original system of equations $Ax = b$ is *redundant*, and the k-th row and column can be removed from B and the k-th row eliminated from A. If the original problem has no feasible solutions, then phase I will terminate with a positive artificial objective function value.

Another approach is to combine phases I and II into one phase by minimizing the objective function

$$z = \sum_{j=1}^{n} c_j x_j + M \sum_{i=1}^{m} y_i$$

where M is chosen so large that eventually all of the artificial variables will be driven to zero if the original problem has a feasible solution.

4. Duality and sensitivity analysis

In this section we introduce the very important and powerful concept of duality. In particular we show that every linear program has associated with it another linear program called the dual that is intimately related to optimality conditions for the original problem. We provide an economic interpretation for the variables of the dual and give several illustrations of how the entire dual problem can be interpreted. We also develop a variant of the simplex method known as the dual simplex method and discuss how to deal with changes that are made to a linear program after it has been solved.

4.1. Duality and optimality

If a basic feasible solution x is nondegenerate then the conditions given in Theorem 3.1 (i.e., all $\bar{c}_j \geqslant 0$) for x to be an optimal solution are both necessary and sufficient, as mentioned below the statement of that theorem. This follows from the fact that the simplex method can be used in this case to compute another basic feasible solution with a lower value of z if and only if some relative cost $\bar{c}_j$ is < 0. When x is a nondegenerate basic feasible solution, we can also state necessary and sufficient conditions for x to be optimal in a different way.

Theorem 4.1. *The basic nondegenerate feasible solution*

$$x = \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} \tag{4.1}$$

to the linear programming problem

$$\begin{array}{ll} \textit{minimize} & z = c^{\mathrm{T}}x \\ \textit{subject to} & Ax = b\,, \quad x \geqslant 0\,, \end{array} \tag{4.2}$$

is optimal if, and only if,

$$c^{\mathrm{T}} = (y^{\mathrm{T}}, \bar{w}^{\mathrm{T}}) \begin{bmatrix} B & N \\ 0 & I \end{bmatrix}, \tag{4.3}$$

where $\bar{w} \geqslant 0$.

Proof. Recall that the rows of

$$M = \begin{bmatrix} B & N \\ 0 & I \end{bmatrix}$$

are linearly independent. Hence they form a basis for $\mathbb{R}^n$ implying that there is a unique vector $(y^{\mathrm{T}}, \bar{w}^{\mathrm{T}})$ that satisfies (4.3). To complete the proof we need only observe that $\bar{w}$ is the vector of nonbasic reduced costs, $\bar{c}_N$, since from (4.3) we have

$$\begin{aligned} \left(y^{\mathrm{T}}, \bar{w}^{\mathrm{T}}\right) &= c^{\mathrm{T}}M^{-1} = (c_B^{\mathrm{T}}, c_N^{\mathrm{T}}) \begin{bmatrix} B^{-1} & -B^{-1}N \\ 0 & I \end{bmatrix} \\ &= \left(c_B^{\mathrm{T}}B^{-1}, c_N^{\mathrm{T}} - c_B^{\mathrm{T}}B^{-1}N\right). \end{aligned}$$

Note that y is the vector of simplex multipliers π computed by the revised simplex method at optimality, and that the 'if' part of this theorem is true even if the basic feasible solution (4.1) is degenerate.

Geometrically, this theorem states that at an optimal nondegenerate vertex x, the gradient of the objective function can be expressed as a linear combination of the normals to all equality constraints plus a nonnegative linear combination of the inward normals to all nonnegativity constraints satisfied as equalities at x.

Let us now consider a linear programming problem which is related in very important ways to the linear program (4.2):

$$\begin{array}{ll} \text{maximize} & v = b^{\mathrm{T}} y \\ \text{subject to} & A^{\mathrm{T}} y \leqslant c\,. \end{array} \tag{4.4}$$

This problem is called the *dual* of problem (4.2), which is now referred to as the *primal*. Note that the dual problem makes use of the same data, A, b, and c, as the primal, and that the dual is, in a sense, a transposed version of the primal, with minimization replaced by maximization. Further, by putting (4.4) into standard form using the techniques of Section 1.2, we can easily prove the following.

Lemma 4.1. *The dual of the problem* (4.4) *is the primal problem* (4.2).

We now show that the dual problem (4.4) arises quite naturally from the optimality conditions of Theorem 4.1. Observe that these conditions can be written as $c^{\mathrm{T}} = y^{\mathrm{T}} A + w^{\mathrm{T}}$, where $w^{\mathrm{T}} \equiv (0, \bar{w}^{\mathrm{T}}) \geqslant 0$. Relaxing the requirement that the first m components of w equal zero yields

$$A^{\mathrm{T}} y + w = c\,, \quad w \geqslant 0\,, \tag{4.5}$$

which are just the constraints of the dual problem (4.4) put into equality form by the introduction of nonnegative slack variables $w \geqslant 0$. Moreover we have:

Lemma 4.2 (Weak duality). *If x is primal feasible and y is dual feasible, then $b^{\mathrm{T}} y \leqslant c^{\mathrm{T}} x$.*

Proof. Since $Ax = b$, $y^{\mathrm{T}} Ax = y^{\mathrm{T}} b$ for any $y \in \mathbb{R}^m$, and since $A^{\mathrm{T}} y \leqslant c$ and $x \geqslant 0$, $y^{\mathrm{T}} Ax \leqslant c^{\mathrm{T}} x$. Combining these results concludes the proof.

This lemma states that the objective value corresponding to a primal (dual) feasible solution provides an upper (a lower) bound for the objective value for any feasible solution including an optimal solution for the other problem. An immediate consequence of this lemma is:

Corollary 4.1. *If x is primal feasible and y is dual feasible and $c^{\mathrm{T}} x = b^{\mathrm{T}} y$, then x and y are optimal solutions.*

But are there feasible solutions x and y that satisfy the hypotheses of this corollary? The answer to this question is provided by:

Theorem 4.2 (Duality theorem of linear programming). (a) *If either the* primal *problem or the* dual *problem has a finite optimal solution, then so does the other and* $\min c^{\mathrm{T}} x = \max b^{\mathrm{T}} y$.

(b) *If either problem has an unbounded objective function value then the other has no feasible solution.*

Proof. Because of Lemma 4.1 and Corollary 4.1, to prove part (a) we need only exhibit a (finite) primal optimal solution x and a dual feasible solution y that satisfy $c^{\mathrm{T}}x = b^{\mathrm{T}}y$. Let x be an optimal basic feasible solution, say (4.1), obtained by the simplex method and let y be the corresponding vector of simplex multipliers $\pi = B^{-1}c_B$. Now y is dual feasible, since

$$c - A^{\mathrm{T}}y = \begin{bmatrix} c_B \\ c_N \end{bmatrix} - \begin{bmatrix} B^T \\ N^T \end{bmatrix} \pi = \begin{bmatrix} 0 \\ \bar{c}_N \end{bmatrix} \geqslant 0$$

and

$$c^{\mathrm{T}}x = c_B^{\mathrm{T}} B^{-1} b = y^{\mathrm{T}} b\,,$$

which concludes the proof of part (a).

Part (b) of the theorem follows directly from weak duality (Lemma 4.2).

The proof shows that the vector of simplex multipliers corresponding to the primal optimal solution x is a dual optimal solution y. Indeed, at any iteration of the simplex method, the simplex multipliers form a vector y with $c^{\mathrm{T}}x = b^{\mathrm{T}}y$, but unless all reduced costs are nonnegative, y is not dual feasible. So the algorithm maintains primal feasibility and $c^{\mathrm{T}}x = b^{\mathrm{T}}y$, while trying to attain dual feasibility.

We note that the converse of part (b) is not necessarily true. That is, if either the primal or dual is infeasible then it does *not* follow that the other problem is unbounded; both can be infeasible.

The following well-known and useful *alternative* theorem for systems of equalities and inequalities is easily derived from part (b) of the Duality theorem.

Theorem 4.3 (Farkas' Lemma). *The system*

$$\text{(I)} \qquad Ax = b\,, \quad x \geqslant 0\,,$$

is unsolvable if and only if the system

$$\text{(II)} \qquad y^{\mathrm{T}}A \leqslant 0, \quad b^{\mathrm{T}}y > 0,$$

is solvable.

Proof. Consider the primal-dual pair of linear programs

$$\begin{array}{llll} \text{(P)} & \text{minimize} & 0^{\mathrm{T}}x & \\ & \text{subject to} & Ax = b\,, & \\ & & x \geqslant 0\,, & \end{array} \qquad \begin{array}{llll} \text{(D)} & \text{maximize} & b^{\mathrm{T}}y & \\ & \text{subject to} & y^{\mathrm{T}}A \leqslant 0\,. & \end{array}$$

Since (D) is feasible ($y = 0$ is a solution), it follows from Theorem 4.2 that (P) is infeasible, or equivalently, (I) is unsolvable, if and only if (D) is unbounded. But clearly (D) is unbounded if and only if (II) is solvable, completing the proof.

Farkas' Lemma (1902) predates the development of linear programming and is often used to prove the Duality theorem rather than the other way round.

Geometrically it states that exactly one of the following is true: (I) b is in the convex cone C generated by the columns of A; or (II) there is a vector y that makes an acute angle with b but not with any vector in C.

Before presenting other consequences of the Duality theorem, we note that corresponding to any linear program, there is a dual linear program, and that the Weak duality lemma, its corollary, and the Duality theorem apply to such primal-dual pairs. For example, the linear programs

$$\begin{array}{llll} \text{(P)} & \text{minimize} & c^{\mathrm{T}}x & \\ & \text{subject to} & Ax \geqslant b\,, & \\ & & x \geqslant 0\,, & \end{array} \qquad \begin{array}{llll} \text{(D)} & \text{maximize} & b^{\mathrm{T}}y & \\ & \text{subject to} & A^{\mathrm{T}}y \leqslant c\,, & \\ & & y \geqslant 0\,, & \end{array} \tag{4.6}$$

are a primal-dual pair. The dual (D) in (4.6) above can be derived by first converting (P) into standard form, then writing down the latter problem's dual and simplifying. The above primal-dual pair is often referred to when discussing duality, since the pair is nicely 'symmetric', in that both problems involve inequalities, in nonnegative variables with the inequalities being '$\geqslant$' in the minimization problem and '$\leqslant$' in the maximization problem. We now state two more theorems that characterize the optimal solutions of this primal-dual pair of problems.

Theorem 4.4 (Complementary slackness). *Let x and y be primal and dual feasible solutions, respectively, of the primal-dual pair (4.6). Necessary and sufficient conditions that they be optimal solutions for their respective problems are*

$$(c^{\mathrm{T}} - y^{\mathrm{T}}A)x = 0 \tag{4.7}$$

and

$$y^{\mathrm{T}}(Ax - b) = 0\,. \tag{4.8}$$

Proof. For primal feasible x and dual feasible y we have

$$s \equiv Ax - b \geqslant 0, \quad x \geqslant 0, \text{ and } w^{\mathrm{T}} \equiv c^{\mathrm{T}} - y^{\mathrm{T}}A \geqslant 0,\ y \geqslant 0\,, \tag{4.9}$$

and hence

$$c^{\mathrm{T}}x \geqslant y^{\mathrm{T}}Ax \geqslant y^{\mathrm{T}}b\,. \tag{4.10}$$

If the conditions (4.7) and (4.8) hold then equality holds throughout (4.10), and the optimality of x and y follows from Corollary 4.1. Conversely, by the Duality theorem if x and y are optimal then $c^{\mathrm{T}}x = y^{\mathrm{T}}b$, which implies that equality holds throughout (4.10) and hence that conditions (4.7) and (4.8) are satisfied.

For the primal-dual pair of linear programs (4.2) and (4.4) only condition (4.7) is meaningful, as (4.8) is true for all primal feasible x. Because of the nonnegativity of the primal and dual variables x and y and slack vectors s and w (see (4.9)) conditions (4.7) and (4.8) can be stated in the following more useful form.

Complementary slackness conditions

$$\begin{aligned} w_j &\equiv (c - A^{\mathrm{T}}y)_j = 0 \text{ or } x_j = 0\,, \quad \text{for all } j = 1, \ldots, n\,, \\ s_i &\equiv (Ax - b)_i = 0 \text{ or } y_i = 0\,, \quad \text{for all } i = 1, \ldots, m\,. \end{aligned} \tag{4.11}$$

Using these conditions Theorem 4.4 states that feasible solutions to the primal and dual problems (4.6) are optimal if and only if (i) a variable is zero in one of the problems whenever the corresponding slack variable is strictly positive (i.e., the corresponding inequality constraint is strictly satisfied) in the other problem, and (ii) a slack variable is zero (i.e., the corresponding inequality constraint is satisfied as an equality) in one of the problems whenever the corresponding variable is positive in the other problem.

The so-called Kuhn-Tucker necessary conditions (Kuhn and Tucker 1951) (developed independently by Karush (1939) and John (1948)) for a solution to be optimal to a nonlinear programming problem are easily derived from Theorem 4.4 for the special case of linear programming. Here they are also sufficient conditions. We state them now for the standard form linear program (4.2).

Theorem 4.5 (Kuhn-Tucker conditions). *x is an optimal solution to the linear program (4.2) if and only if there exist vectors y and w such that*

(i) $Ax = b$, $x \geqslant 0$,
(ii) $A^{\mathrm{T}}y + w = c$, $w \geqslant 0$, *and*
(iii) $w^{\mathrm{T}}x = 0$.

A proof based upon Theorem 4.4 is obvious since condition (i) is primal feasibility, (ii) is dual feasibility, and (iii) is complementary slackness. The standard development of the Kuhn-Tucker conditions for nonlinear programs extends the use of the so-called Lagrangian function of classical equality-constrained nonlinear optimization to the inequality constrained case. In the case of Theorem 4.5 the dual variables y are classical Lagrange multipliers. The dual slacks w are also Lagrange multipliers; however they are not classical as they correspond to inequality constraints and consequently are restricted to be nonnegative. Moreover, the complementary slackness condition (iii) in Theorem 4.5 requires those multipliers that correspond to inactive constraints (inequalities satisfied strictly) to be zero, which makes sense since inactive constraints should not play any part is deciding the optimality of a point.

4.2. Economic interpretation of duality

In the previous section we showed that the dual of a linear program arises naturally from the optimality conditions for the primal problem. In this section we shall show that, typically, if the primal problem has an economic interpretation, so does its dual and the optimal values of the dual variables can be interpreted as prices.

To demonstrate the latter, suppose that

$$x^* = \begin{pmatrix} x_B^* \\ 0 \end{pmatrix} = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$$

is a nondegenerate optimal basic feasible solution to the standard form linear program (4.2). Since, by assumption $x_B^* > 0$, making a small change Δb to b will not cause the optimal basis B to change. Hence, if b is replaced by $b + \Delta b$, the new optimal solution becomes

$$\hat{x}^* = \begin{pmatrix} \hat{x}_B^* \\ 0 \end{pmatrix} = \begin{bmatrix} B^{-1}(b + \Delta b) \\ 0 \end{bmatrix}$$

and the optimal value of the objective function changes by

$$\Delta z = c_B^{\mathrm{T}} B^{-1} \Delta b = \pi^{*\mathrm{T}} \Delta b,$$

where $\pi^* = B^{-\mathrm{T}} c_B$ is the vector of simplex multipliers for the primal problem (4.2) at optimality. As shown in the proof of Theorem 4.2, π^* is the optimal solution to the dual problem (4.4). Clearly, π_i^* can be viewed as the *marginal* price (or *value*) of the i-th resource (i.e., right hand side b_i) in (4.2), since it gives the change in the optimal objective value per unit increase in that resource. This economic interpretation can be very useful since it indicates the maximum amount that one should be willing to pay to increase the amount of the i-th resource. Note that the complementary slackness conditions (4.11) imply that the marginal price for a resource is zero if that resource is not fully utilized at optimality. Other names for these 'prices at optimality' are *shadow prices* and *equilibrium prices*.

These shadow or marginal prices are also useful in determining whether or not to engage in a new activity. For example in the diet problem of Section 1.1 suppose that a previously unavailable food can be purchased. Having obtained a minimal cost diet without this food, should we consider adding it to our diet? To answer this question let the amount of nutrient i provided by the new food be a_{ik} and let the unit cost of the food be c_k. Since the optimal value y_i, of the i-th dual variable can be interpreted as the marginal price of a unit of the i-th nutrient, the nutrients provided by the new food have a value of $\sum_{i=1}^m y_i a_{ik}$. Consequently if c_k is less than this value, the new food is worth purchasing and should be considered for inclusion in the optimal diet (y is not feasible in the new constraint); otherwise, the current optimal diet remains optimal (y remains feasible). In the former case, if the simplex method is invoked to reoptimize, the activity of purchasing the new food is immediately selected to enter the basis. The above operation corresponds to the computation of the reduced cost of an activity in the revised simplex method. The economic interpretation given above accounts for the terminology 'pricing-out' used to describe the operation.

Let us consider the first two linear programming examples presented in Section 1.1. We now show that their duals can be given economic interpretations.

Example 1 (The transportation problem). The dual of the transportation problem is:

$$\text{maximize} \quad \sum_{i=1}^{m} a_i u_i + \sum_{j=1}^{n} b_j v_j \tag{4.12}$$

$$\text{subject to} \quad u_i + v_j \leqslant c_{ij}, \quad i = 1, \ldots, m;\ j = 1, \ldots, n\,. \tag{4.13}$$

According to the discussion above, the dual variables u_i and v_j correspond to the marginal value of increasing the supply at the i-th origin and increasing the demand at the j-th destination by one unit, respectively. This interpretation makes sense from the point of view of the company that is trying to determine an optimal shipping schedule. An interpretation which gives economic meaning to the dual problem, and not just the variables of that problem, is the following:

Suppose that a 'shipping' company proposes to the producer (i.e., the company facing the primal problem) to remove a unit of product from origin i for a price of u_i per unit and to deliver a unit of product at destination j for a price of v_j per unit. By imposing the inequality constraints (4.13) of the dual, the shipping company ensures that its prices are 'competitive' since the producer will always do better to have its product removed and delivered by the shipping company than to ship it directly. Assuming that the amounts, a_i and b_j, of a product available at origin i and required at destination j, respectively, are known to the shipping company, its problem is to set the prices $u_1, \ldots, u_m$ and $v_1, \ldots, v_n$ so as to satisfy (4.13) and maximize its total return (4.12).

Because of the Duality theorem, the producer will not save money by using the shipping company instead of shipping directly. However, by having someone else formulate and solve the dual problem, the producer is saved the task of solving the primal problem.

Example 2 (The diet problem). The diet problem has the form of the primal (P) in (4.6). Consequently, its dual has the form of the dual (D) in (4.6). As in the transportation problem let us interpret the dual of the diet problem as one which is faced by a competitor of the solver of the primal problem. Let this competitor be a pill salesman who sells pure nutrient pills—e.g., pills containing only iron, or only protein. In order to sell such pills to the dietician of the primal problem, this salesman must price these pills competitively. This requires that the nonnegative prices $y_1, \ldots, y_m$ satisfy

$$\sum_{i=1}^{m} y_i a_{ij} \leqslant c_j, \quad j = 1, \ldots, n\,.$$

Recall that a_{ij} is the amount of nutrient i provided by a unit of food j and c_j is the unit cost of food j. Since the minimum daily requirement of nutrient i is b_i, the pill salesman will attempt to maximize $\sum_{i=1}^{m} b_i y_i$; i.e., solve the dual problem (D) in (4.6).

The dual of the product mix problem (Example 3 in Section 1.1) can also be given an economic interpretation as an optimization problem faced by a competitor of the manufacturer that wishes to solve the primal problem. For this and other examples of the use of linear programming, and duality theory in particular, in economic analysis, the reader is referred to (Gale 1960) and (Dorfman, Samuelson, and Solow 1958).

4.3. The dual simplex algorithm

Suppose that one has an infeasible basic solution to a linear programming problem which prices out optimally; i.e., whose simplex multipliers are dual feasible.

Such a situation arises, for example, when an inequality constraint is added to a linear programming problem after that problem has already been solved. If the new inequality is satisfied by the current optimal solution, nothing needs to be done. If the inequality is not satisfied, it can be converted to an equality by the addition of a nonnegative slack variable and added to the constraints of the linear program. Clearly, the optimal basis for the original problem and the new slack variable provide a basis for the expanded problem. This basis prices out optimally but is infeasible because the value of the new basic slack variable equals the negative of the amount by which the current solution fails to satisfy the new inequality.

The dual simplex method (Lemke 1954; Beale 1954) is designed to deal with just such a situation. As in the (primal) simplex method, the method proceeds from basic solution to neighboring basic solution. However, instead of maintaining primal feasibility at each step, dual feasibility is maintained. When a dual feasible basis is obtained that is also primal feasible, the algorithm terminates with the optimal solution of the linear program. In this section, we will abuse nomenclature somewhat by calling a basic (not necessarily feasible) solution of (4.2) 'optimal' if its basis is dual feasible.

To derive the method, let us assume that we are solving the linear program (4.2), and that the current basis consists of the first m variables. Hence, $x_B = B^{-1}b$, $\pi^{\mathrm{T}} = c_B^{\mathrm{T}}B$, and $\bar{c}_N^{\mathrm{T}} = c_N^{\mathrm{T}} - \pi^{\mathrm{T}}N \geqslant 0$. If $x_B \not\geqslant 0$, the point $x^{\mathrm{T}} = (x_B^{\mathrm{T}}, 0)$ corresponds to an optimal but infeasible vertex of the polyhedron of feasible solutions of (4.2); i.e., x would be an optimal vertex if we could ignore the nonnegativity constraints to the basic variables that are negative in the current solution.

Suppose that $x_p < 0$. Clearly, it makes sense to move to a neighboring basic solution (feasible or infeasible vertex) that has $x_p = 0$, by replacing x_p in the basis by a nonbasic variable x_q. The selection of x_q is governed by the requirement that dual feasibility be maintained. To determine which of the $n - m$ neighboring vertices of the current vertex are optimal (i.e., dual feasible) we shall make use of the following:

Lemma 4.3 (Sherman-Morrison-Woodbury modification formula). (a) *If M is an $n \times n$ nonsingular matrix and u and v are any two vectors in $\mathbb{R}^n$ then $M + uv^{\mathrm{T}}$ is nonsingular if and only if $w \equiv 1 + v^{\mathrm{T}}M^{-1}u \neq 0$.*

(b) *Moreover, in this case, $(M + uv^{\mathrm{T}})^{-1} = M^{-1} - (1/w)M^{-1}uv^{\mathrm{T}}M^{-1}$.*

Proof. Since $M + uv^{\mathrm{T}} = (I + uv^{\mathrm{T}}M^{-1})M$, (a) follows from the fact that $I + uv^{\mathrm{T}}M^{-1}$ has $n - 1$ eigenvalues equal to one and one eigenvalue equal to $1 + v^{\mathrm{T}}M^{-1}u$. The updating formula (b) is easily verified by multiplication by $M+uv^{\mathrm{T}}$.

From the proof of Theorem 4.1, we see that the simplex multipliers and nonbasic reduced costs can be computed as

$$(\pi^{\mathrm{T}}, \bar{c}_N^{\mathrm{T}}) = c^{\mathrm{T}}M^{-1},$$

where M is the matrix of active constraint normals (3.3), and M^{-1} is given by (3.4). If on a simplex pivot the p-th basic variable (i.e., x_p) is replaced by x_q, this is equivalent to replacing the q-th row of M (currently e_q^{T}) by e_p^{T}; i.e., M becomes $\bar{M} = M + e_q(e_p - e_q)^{\mathrm{T}}$. Now using Lemma 4.3 and the fact that $e_q^{\mathrm{T}} M^{-1} = e_q^{\mathrm{T}}$ we have that

$$\bar{M}^{-1} = M^{-1} - \frac{M^{-1}e_q\left(e_p^{\mathrm{T}}M^{-1} - e_q^{\mathrm{T}}\right)}{e_p^{\mathrm{T}}M^{-1}e_q}.$$

Premultiplying both sides of this expression by c^{T} we obtain the following formulas for computing the updated simplex multipliers $\bar{\pi}$ and reduced costs $\bar{\bar{c}}_N$:

$$\bar{\pi} = \pi + \gamma u\,,$$

$$\bar{\bar{c}}_j = \bar{c}_j - \gamma\alpha_j\,, \quad j > m\,,\ j \neq q\,,$$

and

$$\bar{\bar{c}}_p = -\gamma\,,$$

where

$$u^{\mathrm{T}} = e_p^{\mathrm{T}}B^{-1}\,, \quad \alpha_j = u^{\mathrm{T}}a_j\,, \text{ and } \gamma = \bar{c}_q/\alpha_q\,.$$

Note that u^{T} is the p-th row of B^{-1} and α_q is the so-called pivot element w_p in the primal simplex algorithm presented in Section 3. It follows from the recurrence relations for the reduced costs, that in order for $\bar{\bar{c}}$, the reduced cost vector at the new basic solution, to be nonnegative we must choose q so that

$$0 \leqslant -\gamma = -\bar{c}_q/\alpha_q \leqslant -\bar{c}_j/\alpha_j\,, \quad \text{for all } \alpha_j < 0,\ j > m\,.$$

If $\alpha_j \geqslant 0$ for all nonbasic j, then $u^{\mathrm{T}}A$ is a nonnegative vector; hence $u^{\mathrm{T}}Ax = u^{\mathrm{T}}b$ cannot have a nonnegative solution since $u^{\mathrm{T}}b = x_p < 0$. This implies that the linear program (4.2) is infeasible. We can now give a 'revised' version of the dual simplex method.

Dual simplex method

(0) Let the dual feasible basic solution x_B to the linear program (4.2), corresponding to the basis matrix $B = [a_{j_1}, \ldots, a_{j_m}]$, be given. Let $\boldsymbol{B} = \{j_1, \ldots, j_m\}$ denote the index set of basic variables. Compute an initial vector of feasible dual variables (simplex multipliers) by solving $B^{\mathrm{T}}\pi = c_B$, and compute $\bar{c}_j = c_j - \pi^{\mathrm{T}}a_j$, for all $j \notin \boldsymbol{B}$.

(1) Check for primal feasibility: If $x_B \geqslant 0$, STOP; the current solution is feasible, and hence, optimal. Otherwise, continue.

(2) Determine the basic variable x_{j_p} to leave the basis: Choose

$$j_p \in V \equiv \{j_i \in \boldsymbol{B} : x_{j_i} < 0\}\ .$$

(3) Check for infeasibility: Compute u by solving $B^{\mathrm{T}}u = e_p$ for u and compute $\alpha_j = u^{\mathrm{T}}a_j$, for all $j \notin \boldsymbol{B}$. If $\alpha_j \geqslant 0$ for all $j \notin \boldsymbol{B}$, STOP; the problem is infeasible.

(4) Determine the nonbasic variable x_q to enter the basis: Choose

$$-\bar{c}_q/\alpha_q = \min\{-\bar{c}_j/\alpha_j : \alpha_j < 0,\ j \notin \boldsymbol{B}\} = -\gamma\,.$$

(5) Update the reduced costs: Set

$$\bar{c}_j \leftarrow \bar{c}_j - \gamma\alpha_j\,, \quad j \notin \boldsymbol{B}\,,\ j \neq q\,,$$

$$\bar{c}_{j_p} \leftarrow -\gamma\,.$$

(6) Update the solution and the basis matrix B: Compute w by solving

$$Bw = a_q\,.$$

Set

$$x_q \leftarrow \theta = x_{j_p}/\alpha_q\,,$$

$$x_{j_i} \leftarrow x_{j_i} - \theta w_i\,, \quad \text{for } 1 \leqslant i \leqslant m,\ i \neq p$$

$$B \leftarrow B + (a_q - a_{j_p})e_p^{\mathrm{T}}\,,$$

$$\boldsymbol{B} \leftarrow \boldsymbol{B} \cup \{q\} \setminus \{j_p\}\,,$$

$$j_p \leftarrow q\,,$$

and go to step (1).

By updating the reduced costs at each iteration, the above version of the dual simplex method requires essentially the same amount of work per iteration as a similarly implemented revised version of the primal simplex method. The principal effort in both cases comes from one B^{-1} and one $B^{-\mathrm{T}}$ operation, the computation of inner products of a vector with all nonbasic columns of A, and the updating of the representation of B^{-1}. If we update π instead of $\bar{c}_N$, additional inner products are required to compute $\bar{c}_j$ for all $j \notin \boldsymbol{B}$ such that $\alpha_j < 0$. We can also compute π directly at each iteration but this requires an extra $B^{-\mathrm{T}}$ operation. One practical disadvantage of the dual simplex method is that all of the $n - m$ inner products $\alpha_j = u^{\mathrm{T}}a_j$, $j \notin \boldsymbol{B}$, must be performed, while in the primal method, one need only compute inner products $\pi^{\mathrm{T}}a_j$ until some specified number of columns price out negatively or all columns have been priced out. This strategy is called *partial pricing* and is commonly used in practice.

Solving the linear program (4.2) by the dual simplex method is mathematically equivalent to solving the dual of that problem by the primal simplex method. This is not surprising since both approaches generate basic feasible solutions to the dual problem and maintain complementary slackness. Applying the simplex method directly to the dual involves working with an $n \times n$ basis matrix $\hat{B}$, in contrast with the $m \times m$ basis matrix B used by the dual simplex method. This seems to indicate that the methods are different. However, it is easy to see that $\hat{B}$ equals M^{T}, where M is the matrix (3.3). For simplicity we are assuming that we are using a variant of the primal simplex method that keeps all free variables in the basis at every iteration. Because of the special form of M and M^{-1} (see

(3.4)), we only need a representation of B^{-1} to implement this method, and it is easily verified that such an implementation is essentially equivalent to the dual simplex method. This corresponds to a 'compact inverse' implementation of the simplex method as described in Section 5 using the 'working basis' B.

Before ending our discussion of the dual simplex method we should point out that it is extensively used in solving integer and mixed integer linear programs using either branch-and-bound or cutting-plane approaches. (See Chapter 6.)

4.4. Sensitivity analysis

In the previous two sections we showed how to obtain the optimal solution of a linear program after the addition of new activities and new constraints, given the optimal solution of the original problem, without resolving the resulting modified problems from scratch. We also explained, in Section 4.2, that the optimal simplex multipliers (i.e., dual variables) give the changes in the optimal objective value for small changes in the right hand sides of the constraints, in the case of a nondegenerate optimal basic feasible solution.

We shall now investigate how more general changes in the right hand sides or in the objective function coefficients effect a previously obtained optimal solution. Such studies are referred to as *sensitivity* or *post-optimality analyses*.

Let us first consider changes in the objective function. In particular, consider the one-parameter family of linear programs

$$\begin{array}{ll} \text{minimize} & z(\theta) = (c + \theta d)^{\mathrm{T}} x \\ \text{subject to} & Ax = b, \quad x \geqslant 0\,. \end{array} \tag{4.14}$$

Suppose that we have an optimal basic feasible solution for $\theta = \theta_0$, and we wish to determine the interval $\underline{\theta} \leqslant \theta \leqslant \bar{\theta}$ for the parameter θ for which the current basis remains optimal. Let this basis be B and let c and d be partitioned into basic and nonbasic parts c_B, d_B and c_N and d_N, respectively. B will be optimal as long as the nonbasic reduced costs remain nonnegative; i.e., $(c_N + \theta d_N)^{\mathrm{T}} - (c_B + \theta d_B)^{\mathrm{T}} B^{-1} N \geqslant 0$. Defining reduced costs $\bar{c}_N^{\mathrm{T}} = c_N^{\mathrm{T}} - c_B^{\mathrm{T}} B^{-1} N$ and $\bar{d}_N^{\mathrm{T}} = d_N^{\mathrm{T}} - d_B^{\mathrm{T}} B^{-1} N$, in terms of c and d alone, the above condition becomes $\theta \bar{d}_N^{\mathrm{T}} \geqslant -\bar{c}_N^{\mathrm{T}}$. Consequently, the range is

$$\begin{array}{rcl} \underline{\theta} & = & \max\{\max\{-\bar{c}_j/\bar{d}_j : \bar{d}_j > 0,\ j \notin \boldsymbol{B}\},\ -\infty\} \leqslant \theta \\ & \leqslant & \min\{\min\{-\bar{c}_j/\bar{d}_j : \bar{d}_j < 0,\ j \notin \boldsymbol{B}\},\ \infty\} = \bar{\theta}\,. \end{array} \tag{4.15}$$

And, for $\underline{\theta} \leqslant \theta \leqslant \bar{\theta}$, the optimal objective value is a linear function of θ; i.e.,

$$z^*(\theta) = (c_B^{\mathrm{T}} + \theta d_B^{\mathrm{T}}) B^{-1} b = z^*(\theta_0) + (\theta - \theta_0) d_B^{\mathrm{T}} x_B\,.$$

If $\theta_0 = 0$ and we choose $d = e_j$, then $[c_j + \underline{\theta}, c_j + \bar{\theta}]$ gives the *range* for the j-th cost coefficient for which the optimal solution, corresponding to $\theta = 0$, remains optimal as long as all other problem data remain fixed.

The optimal solution of the parametric linear program (4.14) can be determined for all values of the parameter θ. Given a range $[\underline{\theta}, \bar{\theta}]$ of θ for a particular optimal basic feasible solution, either there is a neighboring basic feasible solution that is

optimal for values of θ in an interval $[\underline{\theta}, \underline{\underline{\theta}}]$, with $-\infty < \underline{\underline{\theta}}$, or $z^*(\theta)$ is unbounded below for all θ in $(-\infty, \underline{\theta})$. This new solution and basis is obtained by performing a simplex pivot which introduces into the basis the variable x_j that yields $\underline{\theta} = -\bar{c}_j/\bar{d}_j$; in (4.15), and $\underline{\underline{\theta}}$ is determined using the new basis. If an unbounded ray is detected while trying to execute a simplex pivot, $z^*(\theta)$ is unbounded below for all $\theta < \underline{\theta}$. An analogous procedure gives a neighboring optimal basic feasible solution, if one exists, and range $[\bar{\theta}, \bar{\bar{\theta}}]$ for $\theta \geqslant \bar{\theta}$.

As in the simplex method, degenerate pivots can occur; however, the number of nontrivial ranges is clearly finite, and it can be easily shown that $z^*(\theta)$ is a piecewise linear and concave function of θ. Although the number of ranges can be as large as 2^n in pathological cases (Murty 1980), the probabilistic analysis of variants of the simplex method based upon solving (4.14) has yielded bounds on the expected number of iterations which are quadratic in the problem size (see Section 7).

Consider now the right hand side parametric linear program

$$\begin{array}{ll} \text{minimize} & z(\theta) = c^{\mathrm{T}}x \\ \text{subject to} & Ax = b + \theta d\,, \quad x \geqslant 0\,. \end{array}$$

If B is an optimal basis for some value of $\theta = \theta_0$, then the interval $[\underline{\theta}, \bar{\theta}]$ for which this basis remains feasible, and hence yields an optimal solution $x^{\mathrm{T}} = (x_B^{\mathrm{T}}, x_N^{\mathrm{T}}) = (\bar{b}^{\mathrm{T}} + \theta\bar{d}^{\mathrm{T}}, 0)$ where $\bar{b} = B^{-1}b$ and $\bar{d} = B^{-1}d$, is clearly given by

$$\begin{aligned} \underline{\theta} &= \max\{\max_{1\leqslant i\leqslant m}\{-\bar{b}_i/\bar{d}_i : \bar{d}_i > 0\},\ -\infty\} \leqslant \theta \\ &\leqslant \min\{\min_{1\leqslant i\leqslant m}\{-\bar{b}_i/\bar{d}_i : \bar{d}_i < 0\},\ \infty\} = \bar{\theta}\,. \end{aligned}$$

In this interval, although the optimal primal solution varies linearly with θ, the basis and optimal dual solution remain fixed. Neighboring bases and ranges are determined by dual simplex pivots if infeasibility is not detected in contrast with the case of cost function parametrics, which involves primal simplex pivots or the detection of unboundedness.

5.2. Network problems

Here we will outline how the simplex method can be efficiently implemented when the problem is to find a minimum cost network flow subject to capacity and flow conservation constraints.

Let $G = (V, E)$ be a directed graph. Thus V is a finite set of *nodes*, and E a finite set of ordered pairs of distinct nodes called *arcs*. If $e = (i, j) \in E$, we say e *joins* its *tail* $i = t(e)$ to its *head* $j = h(e)$. A *path* P is an alternating sequence $(i_0, e_1, i_1, \ldots, e_l, i_l)$ of distinct nodes and distinct arcs with $e_k = (i_{k-1}, i_k)$ (a *forward* arc) or $e_k = (i_k, i_{k-1})$ (a *reverse* arc) for $1 \leqslant k \leqslant l$. It is *from* i_0 *to* i_l, and of *length* l. A sequence satisfying all these requirements except that $i_0 = i_l$, is a *cycle*. A graph is *connected* if there is a path from any node to any other node, and *acyclic* if it contains no cycle. A graph $H = (W, F)$ is a *subgraph of* G if $W \subseteq V$ and $F \subseteq E$; it is a *spanning* subgraph if $W = V$.

Suppose $G = (V, E)$ is a connected directed graph, $\hat{b} = (b_i)_{i \in V}$ is a vector of net supplies satisfying $\sum_{i \in V} b_i = 0$, and $c = (c_e)_{e \in E}$ is a vector of costs. Then the *transshipment problem*

$$\begin{array}{lll} \text{(P)} & \text{minimize} & \displaystyle\sum_{e \in E} c_e x_e \\ & \text{subject to} & \displaystyle\sum_{e:t(e)=i} x_e - \sum_{e:h(e)=i} x_e = b_i\,, \quad i \in V, \\ & & x_e \geqslant 0\,, \quad e \in E\,, \end{array}$$

is that of shipping a good at minimum cost to satisfy given demands (at nodes i with $b_i < 0$) from given supplies (at nodes i with $b_i > 0$). The minimum cost network flow problem adds upper bounds (capacities) on the flows x_e, but since such problems can be solved by easy extensions of methods for transshipment problems (corresponding to extensions of the simplex method for handling upper bounds), we will keep the discussion simple by ignoring capacities.

Let us denote by $\hat{A}$ the node-arc incidence matrix of G. Thus $\hat{A}$ has a row for each node and a column for each arc with

$$\hat{a}_{ie} = \begin{cases} +1 & \text{if } t(e) = i\,, \\ -1 & \text{if } h(e) = i\,, \\ 0 & \text{otherwise.} \end{cases}$$

Then we can write (P) as

$$\begin{array}{ll} \text{minimize} & c^{\mathrm{T}} x \\ \text{subject to} & \hat{A}x = \hat{b}\,, \\ & x \geqslant 0\,. \end{array}$$

The reason for the carets is that this representation violates our usual assumption that the constraint matrix has full row rank. Indeed it is clear that each column contains one $+1$ and one -1 so that the sum of the rows of $\hat{A}$ is the zero vector. (This is why we required that the sum of the components of $\hat{b}$ be zero, i.e. that the total net supply be zero.) We shall now show that omitting any row from $\hat{A}$ and $\hat{b}$ remedies this difficulty.

Let r be an arbitrary node of G, which we call the root. Let A and b be $\hat{A}$ and $\hat{b}$ with the row corresponding to r deleted; then (P) is equivalent to the standard form problem

$$\begin{array}{ll} \text{minimize} & c^{\mathrm{T}} x \\ \text{subject to} & Ax = b\,, \\ & x \geqslant 0\,. \end{array}$$

We will show that A has rank $n - 1$, where $n = |V|$. At the same time we will characterize the bases or nonsingular submatrices of A of order $n - 1$. We require the notion of a *spanning tree*.

Lemma 5.2. *Let $H = (V, F)$ be a subgraph of the connected directed graph $G = (V, E)$ where $|V| = n$. Then the following are equivalent:*

(i) *$|F| = n - 1$ and H is connected;*
(ii) *$|F| = n - 1$ and H is acyclic;*
(iii) *H is connected and acyclic;*
(iv) *H is minimally connected—the removal of any arc disconnects it; and*
(v) *H is maximally acyclic—the addition of any arc creates a cycle.*

We omit the proof of this standard result in graph theory. If any of these equivalent conditions hold, we say H (or just F) is a spanning tree of G, and we will often omit the adjective spanning.

We can now prove:

Theorem 5.1. *Let $G = (V, E)$ be a connected directed graph with $|V| = n$, let $\hat{A}$ be its node-arc incidence matrix, let $r \in V$ be arbitrary, and let A be $\hat{A}$ with the row indexed by r deleted. Then A has full row rank $n - 1$, and if B is a square submatrix of A of order $n - 1$, then B is nonsingular iff its columns are indexed by the arcs of a spanning tree of G.*

Proof. We first note that any connected graph has a spanning tree by Lemma 5.2(iv): just keep removing arcs until the resulting subgraph is minimally connected.

Next we show that the columns of A indexing the arcs of a cycle of G arc linearly dependent. Indeed, if $P(Q)$ indexes the forward (reverse) arcs of the cycle, then it is easy to see that

$$\sum_{e \in P} a_e - \sum_{e \in Q} a_e = 0 \,.$$

It therefore suffices to show that any submatrix B of A whose columns index the arcs of a spanning tree is nonsingular. This follows from:

Lemma 5.3. *Let $H = (V, F)$ be a spanning tree of G, and let B be the corresponding submatrix of A. Then there is an ordering of the rows and columns of B that makes it upper triangular with nonzero diagonal entries.*

Proof. By induction on n. For $n = 1$, B is the null matrix of order 0, which trivially satisfies the conclusions. (If the reader objects to null matrices, the case $n = 2$ is just as easy: B is the 1×1 matrix whose entry is ± 1.) Suppose the lemma is true for $n < k$, and consider the case $n = k$. Since $2n - 2 = 2|F|$ equals the sum of the degrees of the nodes in H (i.e., the sum over $i \in V$ of the number of arcs e of H with $h(e)$ or $t(e)$ equal to i), and each node has degree at least 1 (H is connected), we deduce that there are at least two nodes of degree 1 (these are called *leaves*). Pick a leaf $i \in V$ other than r and let $e \in F$ be its incident arc. Consider the graph $H' = (V \setminus \{i\}, F \setminus \{e\})$. By Lemma 5.2(ii) it is a spanning tree of $G' = (V \setminus \{i\}, E \setminus \{e\})$, and by the inductive hypothesis we can therefore order the nodes and arcs of H' so that the corresponding matrix, B', is upper

triangular with nonzero diagonal entries. Now add node i as the last row and arc e as the last column, to find

$$B = \begin{pmatrix} B' & u \\ 0 & \pm 1 \end{pmatrix}$$

for some vector u. Hence B has been ordered to have the desired form, proving the inductive step.

Lemma 5.3 shows that every submatrix of A of order $n-1$ has determinant 0, $+1$, or -1 (since the determinant of a triangular matrix is the product of its diagonal entries). In fact, the proof of the lemma can be easily extended to show that every square submatrix of A has determinant in $\{0, +1, -1\}$, i.e. A is *totally unimodular*. Such matrices are of great interest in combinatorial optimization, since the inverse of any nonsingular square submatrix is integer-valued. Hence for any integer-valued b, the basic solutions of $Ax = b$, $x \geqslant 0$, are integer-valued. We record this formally in:

Corollary 5.1. *The transshipment problem (P) has the integrality property that if the net supplies b_i are all integer, every basic solution is integer-valued. Moreover, every basic solution x has x_e nonzero only for $e \in F$ for some spanning tree $F \subseteq E$ of G.*

We now discuss the implementation of the primal simplex algorithm for problem (P). We ignore problems of getting an initial basic feasible solution and resolving degeneracy, for both of which there are special network techniques available—see for instance Chapter 19 of Chvátal (1983). The key idea is to represent the basis B or the corresponding tree $H = (V, F)$ to allow the efficient solution of the linear systems $B^{\mathrm{T}}\pi = c_B$ and $Bw = a_e$. Since B can be reordered to make it triangular, these systems can be solved very fast.

Let $e = (u, v) \in E \setminus F$. Our proof of Theorem 5.1 shows that the solution to $Bw = a_e$ can be obtained by finding the path from u to v in H. If P denotes the set of forward and Q the reverse arcs in this path, then $w_f = +1$ if $f \in P$, -1 if $f \in Q$, and 0 otherwise. To quickly determine such a path we store the *predecessor* $p(i)$ of i for each $i \in V$ other than the root r, i.e. the next node to i in the unique path from i to r, as well as the $d(i)$ of i, the length of this path. We can also use signs on $p(i)$ to indicate whether the arc joining i and $p(i)$ is a forward or reverse arc of this path.

The system $B^{\mathrm{T}}\pi = c_B$ can be written as

$$\pi_i - \pi_j = c_f \quad \text{for } f = (i, j) \in F,$$

where $\pi_r \equiv 0$. We can solve for the π's successively by traversing the nodes of the tree starting from the root r so that $p(i)$ is always visited before i. Such an order is called a *preorder* of the tree; we let $s(i)$ denote the successor of i in this order.

To illustrate these ideas, consider the following:

Example 5.2. Let $V = \{1, 2, \ldots, 9\}$ with $r = 1$, where H is the tree shown in Figure 5.1. Let $x_{35} = x_{94} = 1$, $x_{37} = x_{18} = 2$, $x_{23} = x_{48} = 3$, and $x_{31} = x_{64} = 4$. The tree is represented by the three vectors p, d and s of length $|V|$ as in Table 5.1.

Table 5.1

i	1	2	3	4	5	6	7	8	9
$p(i)$	–	+3	+1	+8	−3	+4	−3	−1	+4
$d(i)$	0	2	1	2	2	3	2	1	3
$s(i)$	3	7	5	9	2	–	8	4	6

The steps of each iteration are then performed efficiently as follows:

(1) Solve $B^{\mathrm{T}}\pi = c_B$, or $\pi_i - \pi_j = c_f$ for $f = (i,j) \in F$, where $\pi_r \equiv 0$. To do this, we calculate the π's in preorder so that $\pi_{|p(i)|}$ is available when we compute π_i. In our example, we calculate π_3, π_5, π_2, π_7, π_8, π_4, π_9, and then π_6. (As we shall see, it is cheaper to update π.)

(2) Check whether $\pi_i - \pi_j \leqslant c_e$ for all $e = (i,j) \in E$. This is a search as in the revised simplex method, except that we need only look up $i = h(e)$ and $j = t(e)$ instead of the column a_e. If all inequalities are satisfied, the current solution is optimal and we stop.

(3) Otherwise, choose some $e = (u,v) \in E \setminus F$ with $\bar{c}_e = c_e - \pi_u + \pi_v < 0$.

(4) Solve $Bw = a_e$ and check for unboundedness. We must find the path from u to v in H, using p and d. If the path has no forward arcs, there is an unbounded ray. In our example, suppose $u = 7$, $v = 9$. Since $d(9) > d(7)$, we find $p(9) = +4$; thus $(9,4)$ is a reverse arc of the path. Now $d(7) = d(4)$, but $7 \neq 4$. So we find $p(7) = -3$, $p(4) = +8$, and $(3,7)$ and $(4,8)$ are reverse arcs of the path. Since $3 \neq 8$, we find $p(3) = +1$, $p(8) = -1$, so $(3,1)$ and $(1,8)$ are forward arcs of the path, and since the two subpaths from u and v have now coalesced, we have found the complete path from u to v.

(5) Find the leaving arc f; this is the forward arc in the path ($w_f = +1$) with minimum flow. We calculate f as we determine the path in step 4 by comparing the flow on each forward arc found with the current minimum flow for such arcs. In our example $f = (1,8)$.

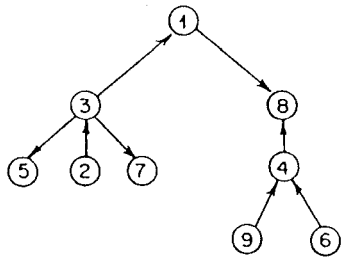

Fig. 5.1.

(6) Update:

$$F \to (F \cup \{e\}) \setminus \{f\}\,.$$

Update x:

$$x_e \leftarrow x_f\,,$$
$$x_g \leftarrow \begin{cases} x_g - x_e & g \text{ forward arc of path}\,, \\ x_g + x_e & g \text{ reverse arc of path}\,, \\ x_g & \text{otherwise}\,, \end{cases}$$

In our example, $x_{79} \leftarrow 2$, $x_{37} \leftarrow 4$, $x_{31} \leftarrow 2$, $x_{18} \leftarrow 0$, $x_{48} \leftarrow 5$ and $x_{94} \leftarrow 3$, with others unchanged.

Update π: let $f = (i, j)$, and assume $d(i) < d(j)$—analogous rules hold if $d(i) > d(j)$.

$$\pi_j \leftarrow \pi_j + \bar{c}_e, \quad k \leftarrow s(j)$$

While $d(k) > d(j)$,

$$\pi_k \leftarrow \pi_k + \bar{c}_e\,,$$
$$k \leftarrow s(k).$$

In our example, we add $\bar{c}_e$ to π_8, then to π_4, π_9 and π_6. (It is easy to see that $\bar{c}_g$ remains zero for arcs in the subtree hanging from node j. Also, π_v increases by $\bar{c}_e$ so that $\bar{c}_e$ becomes zero as required for tree arcs.)

Finally, we must update our representation of the tree, i.e., p, d, and s. While this can be done efficiently, the rules are somewhat complicated and we refer the reader to, e.g., Chvátal (1983). Basically, if $d(i) < d(j)$, the subtree below j now hangs down from the arc (u, v); a crucial role is played by the path from v to j, called the *backpath*.

In our example, the result is the tree shown in Figure 5.2, with representation given in Table 5.2.

The specialization of the primal simplex algorithm for network problems as above is called the network simplex algorithm. While there are examples of problems for which a very large (exponential in n) number of iterations are required using standard choices of entering arc (see (Zadeh 1973)), the method is usually

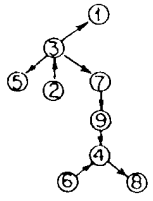

Fig. 5.2.

Table 5.2

i	1	2	3	4	5	6	7	8	9
$p(i)$	–	+3	+1	−9	−3	+4	−3	−4	−7
$d(i)$	0	2	1	4	2	5	2	5	3
$s(i)$	3	7	5	6	2	8	9	–	4

very efficient in practice. For this special class of linear programming problems, there are combinatorial algorithms available which are guaranteed to terminate in a polynomial number of steps—see Chapter 4. Comparisons of different algorithms can be found in the papers in (Gallo and Sandi, 1986).

6. Column generation and the decomposition principle

In this section we continue to consider the efficient application of the revised simplex method to large-scale problems. The first problem we address is the standard form problem

$$\begin{array}{ll} \text{minimize} & z = c^{\mathrm{T}}x \\ \text{subject to} & Ax = b\,, \\ & x \geqslant 0\,, \end{array} \tag{6.1}$$

where A is $m \times n$ and the columns of A are only known implicity (there may be an astronomical number of them). However, the *form* of such columns is known, and they can be generated as needed during the course of the algorithm—hence the name column generation. We will discuss a classic example of such a problem, the cutting-stock problem analyzed by Gilmore and Gomory (1961, 1963) in the first subsection.

Next, suppose we wish to solve the linear programming problem

$$\begin{array}{ll} \text{minimize} & z = c^{\mathrm{T}}x \\ \text{subject to} & A_0x = b_0\,, \\ & A_1x = b_1\,, \\ & x \geqslant 0\,, \end{array} \tag{6.2}$$

where the constraints have been partitioned into 'general' constraints $A_0x = b_0$ and 'special' constraints $A_1x = b_1$, $x \geqslant 0$. For example, the special constraints could define a network problem, or could separate into several groups of constraints involving disjoint sets of variables. We wish to solve (6.2) efficiently, using the fact that linear programming problems involving only the special constraints can be solved much more easily. We shall see that this leads again to the column generation idea; the resulting method is known as the decomposition principle of Dantzig and Wolfe (1960). We discuss this in Subsection 6.2—note that we are reversing chronological order for expository reasons.

6.1. The cutting-stock problem

Suppose that a paper company has a supply of large rolls of paper of width W. However, customer demand is for smaller widths of paper; suppose b_i rolls of width w_i, $i = 1, 2, \ldots, m$, need to be produced. We obtain smaller rolls by slicing a large roll using a particular *pattern*; for example, a large roll of width $w = 70''$ can be cut into three rolls of width $w_1 = 17''$ and one roll of width $w_2 = 15''$, with a waste of $4''$. We can (conceptually at least) consider all such patterns and thus form a matrix A, with a_{ij} indicating how many rolls of width w_i are produced by the j-th pattern, $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$. For example, the pattern described above yields a column $(3, 1, 0, \ldots, 0)^{\mathrm{T}}$ of A. If we let c_j equal 1 for all j, then (6.1) is the problem of determining the minimum number of large rolls ($z = \sum_j x_j$) to cut to satisfy the demand for b_i rolls of width w_i for all i. Actually, we would like to solve the corresponding integer programming problem (see Chapter 6) where x_j, the number of large rolls cut according to pattern j, is also restricted to be integer-valued; however, a solution to the linear programming problem (6.1) often provides a sufficiently accurate solution by rounding and ad hoc procedures, at least if the demands b_i are reasonably large.

Solving (6.1) itself is already a considerable computational task; even if m is comparatively small the number of possible patterns n can be huge, so that just forming the coefficient matrix A in full is impractical. However, as shown by (Gilmore and Gomory 1961, 1963), the problem can be solved efficiently by the revised simplex method, by generating columns of A as required rather than in advance.

Finding an initial basic feasible solution is easy. Indeed, by letting pattern i consist of $[W/w_i]$ rolls of width w_i and none of any other width ($[\lambda]$ denotes the largest integer not exceeding λ), the first m columns of A yield a feasible and diagonal basis matrix. Suppose therefore that at some iteration we have a basic feasible solution and we wish to continue the revised simplex algorithm.

We compute the simplex multipliers π by solving $B^{\mathrm{T}}\pi = e$ as usual. (Here e denotes a vector of all ones of appropriate dimension. Recall that all c_j's are one, so $c_B = e$.) The next step is to compute the reduced costs

$$\bar{c}_j = 1 - \pi^{\mathrm{T}} a_j \tag{6.3}$$

for all j, in order to either determine that we are currently optimal or choose a nonbasic variable x_q, with $\bar{c}_q < 0$, to enter the basis. This appears difficult, since we do not know all columns a_j of A. However, because of the structure of the problem, we can perform this step implicitly.

A vector $a = (\alpha_1, \alpha_2, \ldots, \alpha_m)^{\mathrm{T}} \in Z_+^m$ (each α_i is a nonnegative integer) will be a column of A if it corresponds to a feasible pattern, i.e. if $w^{\mathrm{T}} a \leqslant W$, where $w = (w_1, w_2, \ldots, w_m)^{\mathrm{T}}$. We wish to know whether the reduced cost $1 - \pi^{\mathrm{T}} a$ is nonnegative for all such a, and, if not, find a feasible vector a with $1 - \pi^{\mathrm{T}} a$ negative. Hence we solve the subproblem

$$\begin{array}{ll} \underset{a}{\text{maximize}} & \pi^{\mathrm{T}} a \\ \text{subject to} & w^{\mathrm{T}} a \leqslant W\,, \\ & a \in Z_+^m\,. \end{array} \tag{6.4}$$

This is an instance of the so-called *knapsack* problem. (Think of π_i, as the value and w_i the weight of the i-th item; we seek the most valuable knapsack of weight at most W.) If the optimal value of (6.4) is at most 1, then all reduced costs are nonnegative and our current basic feasible solution is optimal. Otherwise, an optimal solution to (6.4) provides the column $a_q = a$ for a nonbasic variable to enter the basis, and the iteration proceeds, as usual.

We have therefore demonstrated how the revised simplex method can be applied to the cutting-stock problem (6.1) without knowing all the columns of A in advance, by generating them as needed from the subproblem (6.4). We indicate briefly how (6.4) can be solved using a dynamic programming recursion when, as is reasonable, W and all w_i's are integers. Let $f(v)$ denote the optimal value of (6.4) when the right-hand side W is replaced by v. Then $f(v) = 0$ for $0 \leqslant v < w_{\min}$, where $w_{\min} = \min w_i$. We obtain $f(W)$ by using the recursion

$$f(v) = \max_{\substack{1 \leqslant i \leqslant m \\ w_i \leqslant v}} \{f(v - w_i) + \pi_i\}$$

for $v = w_{\min}, \ldots, W$. The optimal solution yielding $f(W)$ can easily be obtained by backtracking if a record is kept of a maximizing index at each step.

Similar column generation methods, with more complicated subproblems to be solved at each iteration, can be used for generalized problems, for instance in 2-dimensional cutting-stock problems or where there is a limit on the number of knives (and hence on the number of nonzero components in a column) in a 1-dimensional problem.

6.2. Dantzig-Wolfe decomposition

We now turn to (6.2), which we rewrite as

$$\begin{array}{ll} \text{minimize} & z = c^{\mathrm{T}} x \\ \text{subject to} & A_0 x = b_0\,, \\ & x \in X\,, \end{array} \tag{6.5}$$

where

$$X = \{x \in \mathbb{R}^n : A_1 x = b_1,\ x \geqslant 0\}. \tag{6.6}$$

We suppose A_0 is $m_0 \times n$ and A_1 is $m_1 \times n$, with the vectors c, x, b_0 and b_1 of conforming dimensions. For example, (6.5) could represent a large-scale production-distribution model, where $A_0 x = b_0$ includes the scarce resource constraints from the production side while $A_1 x = b_1$ involves network constraints from the distribution system. A classical example also arises from multi-divisional problems, as we now briefly outline. Suppose a corporation consists of k divisions, indexed 1 through k we will also refer to the corporation itself as division 0. Each division j has a vector x_j of decision variables (so that x_0 refers to corporate variables, for example corresponding to ways of financing debt); it also has a vector b_j of amounts of resources available to it. Let c_j denote the vector of costs corresponding to x_j, and let A_{ij} be the matrix representing the use of resources of division

i by the variables of division j. We typically assume that $A_{ij} = 0$ if $i \neq j$ and $i > 0$; thus division j uses the resources of the corporation and its own resources, but not those of any other division. The resulting problem is

$$\begin{array}{lllllllll} \text{minimize} & c_0^{\mathrm{T}} x_0 + c_1^{\mathrm{T}} x_1 + \cdots + c_k^{\mathrm{T}} x_k & & & & & & & \\ \text{subject to} & A_{00}x_0 & + & A_{01}x_1 & + & \cdots & + & A_{0k}x_k & = & b_0\,, \\ & & & A_{11}x_1 & & & & & = & b_1\,, \\ & & & & & \ddots & & & \vdots & \\ & & & & & & & A_{kk}x_k & = & b_k\,, \\ & x_0, x_1, \ldots, x_k \geqslant 0\,, & & & & & & & & \end{array} \tag{6.7}$$

whose structure is said to be primal block-angular. Similarly, the problem is called dual block-angular if $A_{ij} = 0$ for $i \neq j$ and $j > 0$ (so the division j does not use the corporate resources, but the corporate variables x_0 may impinge on each division since A_{i0} can be nonzero). In this case, we say x_0 is a vector of linking variables, while the first constraints in (6.7) are called linking constraints. Finally, we can allow both linking variables and linking constraints.

Clearly, problem (6.7) can be viewed as an instance of the general form (6.5), by setting $c^{\mathrm{T}} = (c_0^{\mathrm{T}}, c_1^{\mathrm{T}}, \ldots, c_k^{\mathrm{T}})$ and $A_0 = [A_{00}, A_{01}, \ldots, A_{0k}]$, and

$$\begin{aligned} X \quad = \quad & \{x = (x_0^{\mathrm{T}}, x_1^{\mathrm{T}}, \ldots, x_k^{\mathrm{T}})^{\mathrm{T}} : x_0 \geqslant 0,\ A_{jj}x_j = b_j,\ x_j \geqslant 0, \\ & j = 1, \ldots, k\}. \end{aligned}$$

We will discuss (6.7) later, but for now it will be simpler to assume that there are no corporate variables. Then the problem becomes

$$\begin{array}{lllllll} \text{minimize} & c_1^{\mathrm{T}} x_1 + \cdots + c_k^{\mathrm{T}} x_k & & & & & \\ \text{subject to} & A_{01}x_1 & + & \cdots & + & A_{0k}x_k & = & b_0\,, \\ & A_{11}x_1 & & & & & = & b_1\,, \\ & & & \ddots & & & \vdots & \\ & & & & & A_{kk}x_k & = & b_k\,, \\ & x_1, \ldots, x_k \geqslant 0\,. & & & & & & \end{array} \tag{6.8}$$

Again, it is easy to put (6.8) into the form (6.5), with

$$X = \{x = (x_1^{\mathrm{T}}, \ldots, x_k^{\mathrm{T}})^{\mathrm{T}} : A_{jj}x_j = b_j,\ x_j \geqslant 0,\ j = 1, \ldots, k\}.$$

Alternatively, we can consider each division separately and write (6.8) as

$$\begin{array}{ll} \text{minimize} & c_1^{\mathrm{T}} x_1 + \cdots + c_k^{\mathrm{T}} x_k \\ \text{subject to} & A_{01}x_1 + \cdots + A_{0k}x_k = b_0\,, \\ & x_1 \in X_1, \ldots, x_k \in X_k\,, \end{array} \tag{6.9}$$

with $X_j = \{x_j : A_{jj}x_j = b_j,\ x_j \geqslant 0\}$.

There are several ways to motivate the decomposition idea. One which has a strong economic interpretation is to consider the corporation as trying to decentralize its decision-making by announcing prices for the corporate resources. If no limitations are placed on division j's use of corporate resources, but it must

buy them at prices given by the vector $-\pi_0$, then division j will seek to solve the subproblem

$$\begin{array}{lll} \mathrm{SP}_j(\pi_0) & \text{minimize} & (c_j^{\mathrm{T}} - \pi_0^{\mathrm{T}} A_{0j}) x_j \\ & \text{subject to} & x_j \in X_j . \end{array} \tag{6.10}$$

Of course, the corporation would like to set the prices $-\pi_0$ so that, when each division solves its corresponding subproblem $\mathrm{SP}_j(\pi_0)$ to get an optimal solution $\bar{x}_j$, $\sum_j A_{0j}\bar{x}_j = b_0$. This is akin to a classical economic scenario: we wish to choose prices so that the resulting demands (of the independently utility-maximizing agents) sum to the total supply. We will call $(\bar{\pi}_0, \bar{x}_1, \ldots, \bar{x}_k)$ an *equilibrium* if $\bar{x}_j$ solves $\mathrm{SP}_j(\bar{\pi}_0)$ for each j and $\sum_j A_{0j}\bar{x}_j = b_0$.

A major problem in economic theory is to determine conditions under which an equilibrium exists. Here it is easy and nicely illustrates linear programming duality:

Theorem 6.1. *If $(\bar{x}_1, \ldots, \bar{x}_k)$ and $(\bar{\pi}_0, \bar{\pi}_1, \ldots, \bar{\pi}_k)$ are optimal primal and dual solutions to (6.8), then $(\bar{\pi}_0, \bar{x}_1, \ldots, \bar{x}_k)$ is an equilibrium. Conversely, if $(\bar{\pi}_0, \bar{x}_1, \ldots, \bar{x}_k)$ is an equilibrium, then $(\bar{x}_1, \ldots, \bar{x}_k)$ is an optimal primal (and $\bar{\pi}_0$ part of an optimal dual) solution to (6.8).*

Proof. Consider the first part. Clearly $\sum_j A_{0j}\bar{x}_j = b_0$, so we only need to establish that $\bar{x}_j$ is optimal in $\mathrm{SP}_j(\bar{\pi}_0)$. It is obviously feasible. Now dual feasibility and complementary slackness in (6.8) show that

$$A_{0j}^{\mathrm{T}}\bar{\pi}_0 + A_{jj}^{\mathrm{T}}\bar{\pi}_j \leqslant c_j\,, \qquad (A_{0j}^{\mathrm{T}}\bar{\pi}_0 + A_{jj}^{\mathrm{T}}\bar{\pi}_j - c_j)^{\mathrm{T}}\bar{x}_j = 0\,. \tag{6.11a}$$

But then

$$A_{jj}^{\mathrm{T}}\bar{\pi}_j \leqslant (c_j - A_{0j}^{\mathrm{T}}\bar{\pi}_0)\,, \qquad (A_{jj}^{\mathrm{T}}\bar{\pi}_j - (c_j - A_{0j}^{\mathrm{T}}\bar{\pi}_0)^{\mathrm{T}}\bar{x}_j = 0\,, \tag{6.11b}$$

which shows again by duality that $\bar{x}_j$ is primal optimal (and $\bar{\pi}_j$ dual optimal) in $\mathrm{SP}_j(\bar{\pi}_0)$. For the converse, let $\bar{\pi}_j$ be an optimal dual solution to $\mathrm{SP}_j(\bar{\pi}_0)$. By duality, we have (6.11b), and hence (6.11a), which implies that $(\bar{x}_1, \ldots, \bar{x}_k)$ and $(\bar{\pi}_0, \bar{\pi}_1, \ldots, \bar{\pi}_k)$ are optimal primal and dual solutions to (6.8).

As well as establishing the existence of equilibrium (if (6.8) has a solution), Theorem 6.1 offers the possibility of efficient computation. If we only knew the appropriate prices $-\bar{\pi}_0$, then it appears that we could solve (6.8) by solving k small linear programming problems (the $\mathrm{SP}_j(\bar{\pi}_0)$) instead of one large one.

Unfortunately, two difficulties now arise. First, it is far from clear how a suitable vector $\bar{\pi}_0$ can be found. Second, even if an appropriate $\bar{\pi}_0$ were known, obtaining $\bar{x}_1, \ldots, \bar{x}_k$ would not be easy. Indeed, given nondegeneracy, an optimal solution to (6.8) will have $m_0 + \sum_{j \geqslant 1} m_j$ positive variables, where b_j is an m_j-vector for each j. On the other hand, a basic optimal solution to $\mathrm{SP}_j(\bar{\pi}_0)$ will have only m_j positive variables, so that putting these together will yield only $\sum_{j \geqslant 1} m_j$ positive variables. The conclusion is that any equilibrium $\bar{\pi}_0$ will make at least one of the subproblems have alternate optimal solutions, and that these may have to be chosen suitably to clear the market of the corporate resources.

These difficulties are eliminated by taking another viewpoint. We will construct a new linear programming problem so that applying the revised simplex method will automatically generate a (finite) sequence of trial vectors π_0: and we will explicitly consider convex combinations of the vertices of X_j for the subproblems.

For notational simplicity we return now to the general problem (6.5) with a single polyhedron X; we will consider the case of several X_j's, as in (6.9), later. The key step is to present the polyhedron X in (6.6) in terms of its vertices and (certain of its) directions, as in Theorem 2.2. We need a slight extension:

Definition 6.1. A direction d of X is *extreme* if it cannot be written as a non-negative combination of two different (i.e., not proportional) directions. That is

$$d = \mu_1 d_1 + \mu_2 d_2, \qquad \mu_1 \geqslant 0,\ \mu_2 \geqslant 0,$$

with d_1 and d_2 also directions of X, implies $d_1 = \alpha_1 d$ and $d_2 = \alpha_2 d$ for some $\alpha_1, \alpha_2 \geqslant 0$.

Theorem 6.2. *Any point $x \in X$ can be presented as*

$$x = \sum_{i \in I} \lambda_i v_i + \sum_{j \in J} \mu_j d_j$$

where $\{v_i : i \in I\}$ is the set of vertices, and $\{d_j : j \in J\}$ is the set of extreme directions, of X, and $\sum_{i \in I} \lambda_i = 1$, $\lambda_i \geqslant 0$ for all $i \in I$, $\mu_j \geqslant 0$ for all $j \in J$. Conversely, any such x lies in X. Moreover, X has only finitely many extreme directions.

We will not prove this result; its proof is similar to that of Theorem 2.2. An important consequence is:

Corollary 6.1. *Let the columns of V and D be all vertices and extreme directions, respectively, of X. Then*

$$X = \{V\lambda + D\mu : e^{\mathrm{T}}\lambda = 1,\ \lambda \geqslant 0,\ \mu \geqslant 0\}.$$

(*Recall that e denotes a vector of ones of appropriate dimension.*)

By substituting for x using this representation, we see that our original problem (6.5) is equivalent to the so-called *master problem*

$$\begin{array}{lll} \text{minimize} & (c^{\mathrm{T}}V)\lambda + (c^{\mathrm{T}}D)\mu & \\ \text{subject to} & (A_0V)\lambda + (A_0D)\mu = b_0, & \\ & e^{\mathrm{T}}\lambda = 1, & \\ & \lambda \geqslant 0, \quad \mu \geqslant 0. & \end{array} \tag{6.12}$$

The decomposition principle of Dantzig and Wolfe is to apply the revised simplex method to the master problem (6.12). Note that, in contrast to (6.2), (6.12) has

only μ_0+1 constraints; to compensate, it has an astronomical number of columns, one for each vertex and each extreme direction of X, and those are known only implicitly. Thus we will use a column generation technique as in Subsection 6.1.

Suppose that we have a basic feasible solution to (6.12), $(\bar\lambda, \bar\mu)$, with associated simplex multipliers $\bar\pi_0$ (for the first m_0 constraints) and $\bar\sigma$. Here $\bar\lambda_i > 0$ implies that we know the corresponding vertex v_i of X, and similarly for $\bar\mu_j$ and the extreme direction d_j. However, the set of all vertices and extreme directions is unknown to us, so that we shall have to generate them (and the associated columns in (6.12)) as needed.

An iteration of the revised simplex method demands that we first seek a vertex v_i with reduced cost

$$c^{\mathrm{T}} v_i - \bar\pi_0^{\mathrm{T}} A_0 v_i - \bar\sigma < 0 \tag{6.13}$$

or an extreme direction d_j with reduced cost

$$c^{\mathrm{T}} d_j - \bar\pi_0^{\mathrm{T}} A_0 d_j < 0\,. \tag{6.14}$$

If there are none, we can conclude that the current solution $(\bar\lambda, \bar\mu)$ is optimal in (6.12), and hence $\bar x = V\bar\lambda + D\bar\mu$ optimal in (6.5).

Consider first (6.13). We wish to find a vertex v_i of X so that the linear function $(c^{\mathrm{T}} - \bar\pi_0^{\mathrm{T}} A_0)v_i$ is smaller than $\bar\sigma$. Since a linear function is minimized over a polyhedron at a vertex (if it is not unbounded below), it is natural to consider the subproblem

$$\mathrm{SP}(\bar\pi_0) \qquad \min\{(c^{\mathrm{T}} - \bar\pi_0^{\mathrm{T}} A_0)x : x \in X\} \tag{6.15}$$

(cf. (6.10)).

Let us discuss each possible outcome of solving $\mathrm{SP}(\bar\pi_0)$. First, if it is infeasible, then X is empty and our original problem (6.5) is also infeasible; in this case, we could not have a current basic feasible solution to (6.12).

Second, $\mathrm{SP}(\bar\pi_0)$ may be unbounded. In this case, application of the revised simplex algorithm will generate an edge vector η_q from some vertex v of X with $v + \theta\eta_q$ in X for all $\theta \geqslant 0$ and $(c^{\mathrm{T}} - \bar\pi_0^{\mathrm{T}} A_0)\eta_q < 0$. In fact η_q is of the form

$$\eta_q = \begin{pmatrix} -w \\ e_{q-m_1} \end{pmatrix}, \quad \text{where } w = B_1^{-1} a_{1q}\,,$$

if the current basis matrix B_1 consists of the first m_1, columns of A_1, and a_{1q} is the entering q-th column of A_1. (See Sections 2 and 3.) It is not too hard to see that η_q is an extreme direction of X, so that setting $d_j = \eta_q$ yields (6.14). Of course η_q is not necessarily a direction of the polyhedron of feasible solutions to (6.2), since we have ignored the constraints $A_0 x = b_0$.

Finally, $\mathrm{SP}(\bar\pi_0)$ may have a finite optimal solution $\bar x$. Then, by the fundamental theorems of linear programming (see in particular Theorem 2.4 and its proof), $(c^{\mathrm{T}} - \bar\pi_0^{\mathrm{T}} A_0)d \geqslant 0$ for all directions d and $\bar x$ can be taken to be a vertex of X (and the revised simplex method finds such a vertex). Hence (6.14) fails for j; no column arising from an extreme direction is a candidate for entering the

basis. If $(c^{\mathrm{T}} - \bar{\pi}_0^{\mathrm{T}} A_0)\bar{x} \geqslant \bar{\sigma}$ then, since we minimized over all X and hence all its vertices, (6.13) fails for each i, thus no column arising from a vertex is a candidate for entering the basis, and we conclude that $(\bar{\lambda}, \bar{\mu})$ is optimal in (6.12) and $x^* = V\bar{\lambda} + D\bar{\mu}$ optimal in (6.5). On the other hand, if $(c^{\mathrm{T}} - \bar{\pi}_0 A_0)\bar{x} < \bar{\sigma}$, then setting $v_i = \bar{x}$ we have (6.13).

Thus in any case, we either prove optimality or generate a column for (6.12) to introduce into the basis, in which case the iteration of the revised simplex method can continue as usual. Summarizing, we have:

Theorem 6.3. (a) *If SP($\bar{\pi}_0$) is unbounded, the revised simplex method applied to it yields an extreme direction d_j satisfying* (6.14)*, so that the column*

$$\begin{pmatrix} A_0 d_j \\ 0 \end{pmatrix} \quad \textit{with cost } c^{\mathrm{T}} d_j$$

is eligible to enter the current basis for the master problem (6.12)*.*

(b) *If SP($\bar{\pi}_0$) has optimal solution v_i with optimal value less than $\bar{\sigma}$, then the column*

$$\begin{pmatrix} A_0 v_i \\ 1 \end{pmatrix} \quad \textit{with cost } c^{\mathrm{T}} v_i$$

is eligible to enter the current basis for the master problem (6.12)*.*

(c) *Finally, if SP($\bar{\pi}_0$) has optimal value at least $\bar{\sigma}$, with optimal dual solution $\bar{\pi}_1$, then the current basic feasible solution $(\bar{\lambda}, \bar{\mu})$ is optimal in* (6.12) *with optimal dual solution $(\bar{\pi}_0, \bar{\sigma})$ and $x^* = V\bar{\lambda} + D\bar{\mu}$ is optimal in* (6.2)*, with optimal dual solution $(\bar{\pi}_0, \bar{\pi}_1)$.*

Proof. We only need to show the last part. Clearly, since (6.13) and (6.14) fail for all $i \in I$, $j \in J$, $(\bar{\pi}_0, \bar{\sigma})$ is feasible in the dual of (6.12), and hence $(\bar{\lambda}, \bar{\mu})$ and $(\bar{\pi}_0, \bar{\sigma})$ are respectively primal and dual optimal so that $c^{\mathrm{T}} V\bar{\lambda} + c^{\mathrm{T}} D\bar{\mu} = \bar{\pi}_0^{\mathrm{T}} b_0 + \bar{\sigma}$. Now x^* is feasible in (6.2), since it satisfies $A_0 x^* = b_0$ and lies in X by Theorem 6.2. It has value $c^{\mathrm{T}} x^* = c^{\mathrm{T}} V\bar{\lambda} + c^{\mathrm{T}} D\bar{\mu}$, the optimal value of (6.12). Now since $\bar{\pi}_1$ is dual optimal in SP($\bar{\pi}_0$), it is dual feasible:

$$\bar{\pi}_1^{\mathrm{T}} A_1 \leqslant c^{\mathrm{T}} - \bar{\pi}_0^{\mathrm{T}} A_0 \tag{6.16}$$

and has value

$$\begin{aligned} \bar{\pi}_1^{\mathrm{T}} b_1 &\geqslant \bar{\sigma} = (\bar{\pi}_0^{\mathrm{T}} b_0 + \bar{\sigma}) - \bar{\pi}_0^{\mathrm{T}} b_0 \\ &= (c^{\mathrm{T}} V\bar{\lambda} + c^{\mathrm{T}} D\bar{\mu}) - \bar{\pi}_0^{\mathrm{T}} b_0 \\ &= c^{\mathrm{T}} x^* - \bar{\pi}_0^{\mathrm{T}} b_0. \end{aligned} \tag{6.17}$$

Hence $(\bar{\pi}_0, \bar{\pi}_1)$ is feasible in the dual of (6.2) by (6.16) and has value at least that of x^* in the primal. Thus weak duality implies that x^* is primal and $(\bar{\pi}_0, \bar{\pi}_1)$ dual optimal in (6.2) as desired.

The theorem shows that we can solve (6.5) by solving instead the master problem (6.12); finite convergence is assured since we are applying the revised simplex

method to a finite problem, even though its coefficients are not all known in advance. The algorithm terminates either with an optimal solution of (6.12), and hence one for (6.2), or with an indication of unboundedness; in the latter case, it is easy to see that (6.2) is also unbounded. The proof also shows that, when the algorithm terminates with an optimal solution, the optimal value of $\mathrm{SP}(\bar{\pi}_0)$ is precisely $\bar{\sigma}$, and, by complementary slackness, all v_i, with $\bar{\lambda}_i$ positive will be alternate optimal solutions to $\mathrm{SP}(\bar{\pi}_0)$. Hence we have resolved the two difficulties discussed below Theorem 6.1. Applying the revised simplex method to the master problem automatically generates a sequence of vectors $-\bar{\pi}_0$ which converges to an 'equilibrium price vector' $-\pi_0^*$; and the master problem explicitly considers how to combine the optimal solutions to $\mathrm{SP}(\pi_0^*)$ to get an optimal solution to (6.5), which is likely not to be a vertex of X. Dantzig (1963) includes a discussion that motivates from an economic viewpoint the proposed equilibrium price vectors $-\bar{\pi}_0$ that are generated in this way.

A natural question concerns the computational behavior of the decomposition algorithm; we may hope that only a small multiple of m_0 iterations are required in (6.12) (see the next section), even though this problem has a huge number of columns. We defer discussion of this point until we have addressed a number of issues we have skirted so far.

First, suppose there are variables that occur only in the 'general' constraints $A_0x = b_0$, but not in the 'special' constraints $A_1x = b_1$, like the corporate variables x_0 in (6.7). It is then more natural to omit these variables from X and carry them over unchanged into the master problem (6.12). To apply the revised simplex algorithm to (6.12), we first check at each iteration whether any of these x-variables is a candidate to enter the basis, and, if so, perform the pivot. Only if no such x-variable is eligible do we form the subproblem $\mathrm{SP}(\bar{\pi}_0)$ and proceed as above. The analysis follows the argument we have already made.

Second, we need to describe how an initial basic feasible solution to (6.12) is found. We first find a vertex v_1, of X by any method—if we discover X is empty, (6.5) is infeasible and we stop. Then we introduce m_0 artificial variables into (6.12) so that λ_1, together with these variables gives a basic feasible solution to the modified (6.12). We now apply a phase I revised simplex method to this problem to minimize the sum of the artificial variables, using again the decomposition idea. If all artificial variables are eliminated, we have a basic feasible solution to (6.12) from which we initiate phase II. We can view this phase I procedure as applying the decomposition algorithm to a phase I version of (6.2), where the artificial variables (in the constraints $A_0x = b_0$ only) are carried over into the master problem as in the previous paragraph.

Third, let us reconsider the multi-divisional problem (6.7). We know now that we can treat the variables x_0 separately, but can we separate the variables $x_1, x_2, \ldots, x_k$ as in (6.9) rather than considering them together? The answer is yes;

we must apply the same idea to each polyhedron X_j. Thus the master problem is

$$\begin{array}{lll}
\text{minimize} & c_0^{\mathrm{T}}x_0 + (c_1^{\mathrm{T}}V_1)\lambda_1 + (c_1^{\mathrm{T}}D_1)\mu_1 + \ldots + (c_k^{\mathrm{T}}V_k)\lambda_k + (c_k^{\mathrm{T}}D_k)\mu_k & \\
\text{subject to} & A_{00}x_0 + (A_{01}V_1)\lambda_1 + (A_{01}D_1)\mu_1 + \ldots + (A_{0k}V_k)\lambda_k + (A_{0k}D_k)\mu_k = b\,, \\
& e^{\mathrm{T}}\lambda_1 = 1\,, \\
& \qquad\ddots \qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
& \qquad\qquad\qquad\qquad\qquad\qquad e^{\mathrm{T}}\lambda_k = 1\,, \\
& x_0, \lambda_1, \mu_1, \ldots, \lambda_k, \mu_k \geqslant 0\,, & (6.18)
\end{array}$$

where the columns of V_j and D_j are the vertices and extreme directions, respectively, of X_j, and the components of the vectors λ_j and μ_j give the corresponding weights. Note that (6.18) has $m_0 + k$ rows, rather than $m_0 + 1$; but this is still a great reduction from the number in (6.7). At any iteration, we will have simplex multipliers $\bar{\pi}_0, \bar{\sigma}_1, \ldots, \bar{\sigma}_k$. If any x_0-variable is eligible to enter the basis, we make the appropriate pivot. Otherwise, we solve $\mathrm{SP}_j(\bar{\pi}_0)$ (see (6.10)) for each j. If any such problem is unbounded, the extreme direction generated yields an eligible column for (6.18). If not, and if the optimal value of some $\mathrm{SP}_j(\bar{\pi}_0)$ is less than $\bar{\sigma}_j$, then the optimal vertex v_{ji} again yields an eligible column. Finally, if all optimal values are equal to the corresponding $\bar{\sigma}_j$, then we have the optimal solution to (6.18), and $x_0^* = \bar{x}_0$, $x_j^* = V_j\bar{\lambda}_j + D_j\bar{\mu}_j$, $j = 1, 2, \ldots, k$, is an optimal solution to (6.7).

This idea of having several subproblems (rather than just one) is also appropriate in multicommodity flow problems; for a discussion of these and various solution strategies, including primal and dual decomposition methods, see Kennington (1978). Another important special case is where A has a staircase structure, which arises in multiperiod models. In this case, nested decomposition approaches may be attractive (Ho and Manne 1974). More recent developments are discussed in Ho (1987), who also includes a summary of computational experience.

Papers and textbooks of the 60's and 70's generally cite poor computational results for the decomposition approach compared to applying a sophisticated revised simplex code directly to (6.2). The folklore suggests that slow final convergence is typical. However, there is some indication (see (Ho 1987) and the papers cited therein) that 'long tails' are the fault more of numerical accuracy difficulties than of the algorithmic approach. Here it is worth pointing out that, at any iteration where $\mathrm{SP}_j(\bar{\pi}_0)$ (or all $\mathrm{SP}_j(\bar{\pi}_0)$'s) has an optimal solution, one can obtain a feasible dual solution to the master problem, with a duality gap equal to the difference between $\bar{\sigma}$ and the optimal value of $\mathrm{SP}_j(\bar{\pi}_0)$. Hence one can terminate early with a feasible solution which is guaranteed to be close to optimal. With such early termination, large-scale dynamic problems have been solved faster by sophisticated implementations of decomposition approaches than using IBM's MPSX simplex code—see again (Ho 1987). For problems of more moderate size, however, say up to 5000 rows, it seems generally preferable to use a good revised simplex code on the original problem, ignoring its structure.