SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK 14853

TECHNICAL REPORT NO. 777

February 1988

(Revised November 1988)

LINEAR PROGRAMMING*

by

Donald Goldfarb†
Dept. of IE&OR
Columbia University

Michael J. Todd††
School of OR&IE
Cornell University

*This manuscript will form a chapter of a volume on optimization in the series of Handbooks in Operations Research and Management Science, edited by G.L. Nemhauser and A.H.G. Rinnooy Kan.

†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.

TABLE OF CONTENTS

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 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 Nemirovoskii [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,...m, and b_j units are required at the j-th destination, j=1,2,...,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,...m, j=1,2,...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:

$$\text{minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{j=1}^{n} x_{ij} = a_i, \quad i=1,2,...m \tag{1.1}$$

$$\sum_{i=1}^{m} x_{ij} = b_j, \quad j=1,2,...n \tag{1.2}$$

$$x_{ij} \geq 0, \quad i=1,2,...m; \; j=1,2,...n.$$

The objective function that is being minimized is clearly equal to the total shipping cost. Each of the equality 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 that 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 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:

$$\text{minimize} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad i=1,2,\ldots,m$$

$$x_j \geq 0, \qquad j=1,2,\ldots,n$$

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:

$$\text{maximize} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij}x_j \leq b_i, \qquad i=1,2,\ldots,m$$

$$x_j \geq 0, \quad j=1,2,\ldots,n.$$

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

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

and the standard form

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

where A in both cases denotes an (m x n) matrix, $c \varepsilon |R^n$, $b \varepsilon |R^m$ and $x \varepsilon |R^n$ 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, $x_j' \geq 0$, $x_j'' \geq 0$ 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

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

can be 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

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

Finally, maximizing the linear function c^Tx is equivalent to minimizing $-c^Tx$.

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.

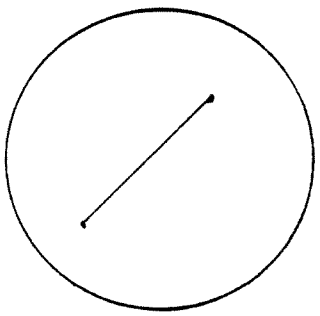
2. GEOMETRIC INTERPRETATION

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.

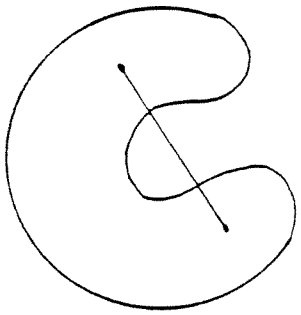
2.1 Definitions:

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 $0 \leq \lambda \leq 1$, are in C.

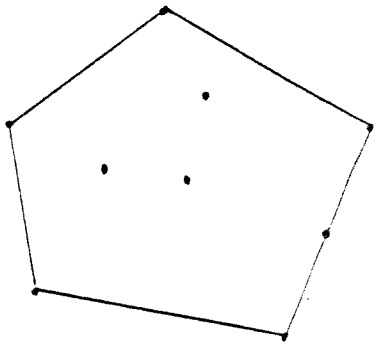
That is, for any two points in the set, the line segment between these points must lie in the set for it to be convex.



convex set



nonconvex set



set of all convex combinations

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

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

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 $\{x \in \mathbb{R}^n : x = A\alpha, \alpha \geq 0\}$ is a convex cone generated by the columns of $A \in \mathbb{R}^{n \times m}$. (Here, $\alpha \in \mathbb{R}^m$.)

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 $H = \{x \in |R^n : a^T x = \beta\}$ where $a \in |R^n$, $a \neq 0$, and $\beta \in |R$ is a hyperplane. The set $\overline{H} = \{x \in |R^n : a^T x \leq \beta\}$ is a closed half-space. The hyperplane H associated with the half-space $\overline{H}$ 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 $\overline{H}$. To see this let $y, z \in H$ and $w \in \overline{H}$. Then

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

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

$$a^T(w-z) = a^T w - a^T z \leq \beta - \beta = 0,$$

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

Definition 2.6: A set $S_a \subseteq |R^n$ 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 is convex; hence, a convex polyhedron as defined above is convex. Clearly, the set of feasible solutions of a linear programming problem, $P = \{x \in R^n : Ax \leq b, x \geq 0\}$ (or $\overline{P} = \{x \in R^n : Ax = b, x \geq 0\}$) is a convex polyhedron since it is the intersection of the half-spaces defined by the inequalities

$$a_1^T x \leq b_1, \ldots, a_m^T x \leq b_m,$$

and (2.1)

$$e_1^T x \geq 0, \ldots, e_n^T x \geq 0$$

where a_i^T is the i-th row of A and e_i^T is the i-th row of the (nxn) 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 $C \subseteq \overline{H}$, 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 $P = \{x \in |R^n \mid Ax \leq b, x \geq 0\}$ 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 optional 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 $\{x \in |R^n \mid Ax = b, x \geq 0\}$.

Theorem 2.1: A point $x \in P = \{x \in |R^n : Ax = b, x \geq 0\}$ 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 $x = \begin{pmatrix} \bar{x} \\ 0 \end{pmatrix}$, $\bar{x} > 0$, 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 , $(\bar{x} \pm \varepsilon\bar{w}) \geq 0$. Consequently the points $y' = \begin{pmatrix} \bar{x} + \varepsilon\bar{w} \\ 0 \end{pmatrix}$ and $y'' = \begin{pmatrix} \bar{x} - \varepsilon\bar{w} \\ 0 \end{pmatrix}$ are both in P. Further, since $x = (y' + y'')/2$, 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 x 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 an 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 this 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 2.2: 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 2.3: 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 2.6 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 $\{x \in \mathbb{R}^n \mid x = x_0 + \lambda d, \lambda \geq 0\}$ 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

$$Ad = 0 \text{ and } d \geq 0.$$

We can now state the following representation theorem.

Theorem 2.4 (Representation Theorem)

Any point $x \in P$ can be represented as

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

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

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

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

$$x = \mu x' + (1 - \mu)x''$$

$$= \mu \left(\sum_{i \in I} \lambda_i' v_i + d' \right) + (1 - \mu) \left(\sum_{i \in I} \lambda_i'' v_i + d'' \right)$$

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

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

Since $0 < \mu < 1$,

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

$$Ad' = Ad'' = 0, \; d' \geq 0 \text{ and } d'' \geq 0$$

it follows that

$$\lambda_i \equiv \mu\lambda_i' + (1 - \mu)\lambda_i'' \geq 0 \quad \text{for all } i \in I, \quad \sum_{i \in I} \lambda_i = 1$$

$$d \equiv \mu d' + (1 - \mu)d'' \geq 0, \quad \text{and} \quad Ad = 0,$$

and we have proved that x has the desired form.

Case b: $w \leq 0$

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 the desired form by induction, and (-w) is a direction of P, x clearly has the desired form.

Case c: $w \geq 0$

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 2.5: 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 2.6: If P is nonempty, then it has at least one vertex.

Proof: This follows immediately from Theorem 2.5 and its proof.

Theorem 2.7: If P is nonempty, then the minimum value of $z = c^T x$ 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 $c^T d < 0$. 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 $c^T d < 0$. 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

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

has an objective value that is bounded below by $c^T x$. But

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

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,

$$\begin{aligned} &\text{minimize} && z = c^T x \\ &\text{subject to} && Ax = b \\ & && x \geq 0, \end{aligned} \tag{3.1}$$

we need only consider the vertices of the polyhedron $P = \{x: Ax = b, x \geq 0\}$, 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 $\binom{n}{m} = \frac{n!}{m!(n-m)!}$ 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 $z = c^T x$, 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 be 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 $A = [B \mid N]$. We also partition c^T as $c^T = \begin{bmatrix} c_B^T & c_N^T \end{bmatrix}$ 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 nonsingular, the rows of M are linearly independent, and hence M is nonsingular. 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 $x_j \geq 0$ are satisfied as equalities and the point x satisfies more than n equations.

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

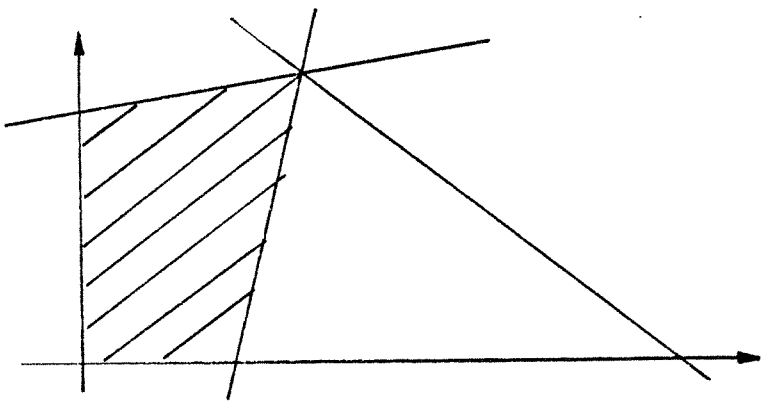


Figure 3.1. An Illustration of Degeneracy

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

$$P_k = \Big\{ x_{ij},\ 1 \le i, j \le k:\ \sum_{j=1}^{k} x_{ij} = 1,\ 1 \le i \le k;\ \sum_{i=1}^{k} x_{ij} = 1,\ 1 \le j \le k;$$
$$0 \le x_{ij},\ 1 \le i, j \le k \Big\},$$

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 this 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

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

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 nxn 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

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

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 <u>reduced</u> or <u>relative costs</u>

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

If $\bar{c}_j < 0$, then the gradient c of the objective function $z = c^Tx$ 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

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

Clearly the reduced cost $\bar{c}_j$ is the directional derivative of $z = c^T x$ 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

$$\frac{c^T \eta_q}{\|\eta_q\|} = \underset{j > m}{\text{Min}} \left\{ \frac{c^T \eta_j}{\|\eta_j\|} \right\}.$$

That is, the steepest edge is the one which makes the most obtuse angle ϕ_q with c, where $\phi_q = \cos^{-1} \left(\frac{c^T \eta_q}{\|c\| \cdot \|\eta_q\|} \right)$. This pivot rule can be implemented in a practicable manner for large problems if the quantities $\|\eta_j\|^2 = \eta_j{}^T \eta_j$ 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 <u>simplex multipliers</u> $\Pi^T = c_B^T B^{-1}$ followed by "pricing-out" all nonbasic columns, i.e.,

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

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 point $y \in P$ can be expressed as

$$y = x + \sum_{j=m+1}^{n} v_j \eta_j, \quad v_j \geq 0, \ j=m+1,\ldots,n,$$

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

Proof: Since $y \in P$, $Ay=b$ and $y = \begin{pmatrix} y_B \\ y_N \end{pmatrix} \geq 0$.

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 $y_N \geq 0$.

From this lemma we have that

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

for all $y \in P$. Since y is nonnegative, if the reduced costs $\bar{c}_j$ are nonnegative, it follows that $z(y) \geq z(x)$ 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 $x_i = 0$ 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_q 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 $x(\theta) \geq 0$ if and only if $x_B - \theta w \geq 0$ and $\theta \geq 0$. 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 $\theta \geq 0$ and $z(x(\theta)) \to -\infty$ as $\theta \to \infty$. In this case, $d = \eta_q$ is a direction with $c^T d = \bar{c}_q < 0$.

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

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

We have used 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 basis matrix B is concerned, one of its columns, a_p, is replaced by the column a_q, i.e.

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

From (3.5) - (3.10) it follows that

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

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 $c^T\bar{x} = c^Tx - \theta\bar{c}_q$ strictly less than c^Tx 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 $B = [a_{j_1},\ldots,a_{j_m}]$, be given. Let $\underset{\sim}{B} = \{j_1,\ldots,j_m\}$ denote the index set of basic variables; hence x_{j_i} denotes the i-th basic variable.

(1) Compute simplex multipliers by solving

$$B^T\Pi = c_B$$

for Π, and compute the reduced costs

$$\bar{c}_j = c_j - \Pi^Ta_j, \text{ for all } j \notin \underset{\sim}{B}.$$

(2) Check for optimality:

If $\bar{c}_j \geq 0$, for all $j \notin \underset{\sim}{B}$,

stop; the current solution is optimal.

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

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

(4) Check for an unbounded ray:

Compute w by solving

$$Bw = a_q .$$

If $w \leq 0$, 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

$$\frac{x_{j_p}}{w_p} = \min_{1 \leq i \leq m} \left\{ \frac{x_{j_i}}{w_i} \mid w_i > 0 \right\}$$

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

Set $x_q \leftarrow \theta = x_{j_p} / w_p$

$$x_{j_i} \leftarrow x_{j_i} - \theta w_i, \quad 1 \leq i \leq m$$

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

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

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 was 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 $x_{j_p} \geq 0$. 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 x m systems of equations $B^T\Pi = c_B$ 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	$\bar{c}^T_N$	$-z_0$
0	I	$B^{-1}N$	$B^{-1}b$

, (3.11)

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

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

(ii) for $0 \leq i \leq m$, $i \neq p$, 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

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

(column p)

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

This follows from the fact that

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

where

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

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 <u>product form</u> 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 schemas 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 9.6). 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-Jordon 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

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

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,...,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 $e_k^T B^{-1} a_j \neq 0$, y_i can be replaced by the nonbasic variable x_j. If $e_k^T B^{-1} a_j = 0$ for all $j \notin B$, 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 $\bar{c}_j \geq 0$), for x to be an optimal solution are both necessary and sufficient, as mentioned below the statement of that 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

$$\begin{aligned} &\text{minimize} && z = c^T x \\ &\text{subject to} && Ax = b, \quad x \geq 0 \end{aligned} \tag{4.2}$$

is optimal if, and only if,

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

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

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 $(y^T, \bar{w}^T)$ 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

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

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):

$$\begin{aligned} &\text{maximize} && v = b^T y \\ &\text{subject to} && A^T y \leq c. \end{aligned} \tag{4.4}$$

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 dual 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 $c^T = y^T A + w^T$, where $w^T \equiv (0, \bar{w}^T) \geq 0$. Relaxing the requirement that the first m components of w equal zero yields

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

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

Lemma 4.2. (Weak Duality) If x is primal feasible and y is dual feasible then $b^T y \leq c^T x$.

Proof: Since $Ax = b$, $v^T Ax = y^T b$ for any $y \in \mathbb{R}^m$, and since $A^T y \leq c$ and $x \geq 0$, $y^T Ax \leq c^T x$. 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 $c^T x = b^T y$, 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 $\min c^T x = \max b^T y$.

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 $c^T x = b^T y$. 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 $\Pi = B^{-T} c_B$. Now y is dual feasible, since

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

and

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

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 v. Indeed, at any

iteration of the simplex method, the simplex multipliers form a vector y with $c^Tx = b^Ty$, but unless all reduced costs are nonnegative, y is not dual feasible. So the algorithm maintains primal feasibility and $c^Tx = b^Ty$, and seeks 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

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

is unsolvable if and only if the system

$$\text{(II)} \quad y^TA \leq 0, \quad b^Ty > 0$$

is solvable.

Proof. Consider the primal-dual pair of linear programs

$$\text{(P)} \quad \text{minimize } 0^Tx \quad \text{subject to } Ax = b, \; x \geq 0$$

$$\text{(D)} \quad \text{maximize } b^Ty \quad \text{subject to } y^TA \leq 0$$

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

$$\begin{array}{lll} \text{(P)} & \text{minimize} \quad c^T x & \text{(D)} \quad \text{maximize} \quad b^T y \\ & \text{subject to} \quad Ax \geq b & \text{subject to} \quad A^T y \leq c \\ & \qquad\quad x \geq 0, & \qquad\quad y \geq 0, \end{array} \tag{4.6}$$

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 "$\geq$" in the minimization problem and "$\leq$" 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

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

and

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

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

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

and hence

$$c^T x \geq y^T Ax \geq y^T b. \tag{4.10}$$

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 $c^T x = y^T b$, 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

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

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) $Ax = b, \quad x \geq 0,$

(ii) $A^T y + w = c, \quad w \geq 0$

and

(iii) $w^T x = 0.$

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 in 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

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

where $\pi^* = B^{-T} c_B$ 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 is 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 interpreta-

tion 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:

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

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

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

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

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, $x_B = B^{-1}b$, $\pi^T = c_B^T B$, and $\bar{c}_N^T = c_N^T - \pi^T N \geq 0$. If $x_B \not\geq 0$, the point $x^T = (x_B^T, 0)$ 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 corresponding 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 (nxn) nonsingular matrix and u and v are any two vectors in $\mathbb{R}^n$ then $M + uv^T$ is nonsingular if and only $w \equiv 1 + v^T M^{-1} u \neq 0$.

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

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

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

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

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 $\bar{M} = M + e_q (e_p - e_q)^T$. Now using Lemma 4.3 and the fact that $e_q^T M^{-1} = e_q^T$ we have that

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

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

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

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

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

If $\alpha_j \geq 0$ for all nonbasic j, then u^TA is a nonnegative vector; hence $u^TAx = u^Tb$ cannot have a nonnegative solution since $u^Tb = x_p < 0$. 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 $\underset{\sim}{B} = \{j_1,\ldots,j_m\}$ denote the index set of basic variables. Compute an initial vector of feasible dual variables (simplex multipliers) by solving $B^T\Pi = c_B$, and compute $\bar{c}_j = c_j - \Pi^T a_j$, for all $j \notin \underset{\sim}{B}$.

(1) Check for primal feasibility:

If $x_B \geq 0$, stop; the current solution is feasible, and hence, optimal. Otherwise, continue.

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

Choose $j_p \in V \equiv \{j_i \in \underset{\sim}{B} \mid x_{j_i} < 0\}$.

(3) Check for infeasibility:

Compute u by solving $B^Tu = e_p$ for u and compute $\alpha_j = u^Ta_j$, for all $j \notin \underset{\sim}{B}$. If $\alpha_j \geq 0$ for all $j \notin \underset{\sim}{B}$, stop; the problem is infeasible.

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

Choose

$$-\bar{c}_q/\alpha_q = \min \{-\bar{c}_j/\alpha_j \mid \alpha_j < 0,\ j \notin \underset{\sim}{B}\} = -\gamma$$

(5) Update the reduced costs:

Set $\bar{c}_j \leftarrow \bar{c}_j - \gamma\alpha_j$, $j \notin \underset{\sim}{B}$, $j \neq q$

$\bar{c}_p \leftarrow -\gamma$.

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

Compute w by solving

$Bw = a_q$

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

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

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

$\underset{\sim}{B} \leftarrow \underset{\sim}{B} \cup \{q\} \setminus \{j_p\}$

$j_p \leftarrow q$

and go to step (1).

By updating the reduced costs at each iteration, the above version of 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^{-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 $j \notin \underset{\sim}{B}$ such that $\alpha_j < 0$. We can also compute Π directly at each iteration but this requires an extra B^{-T} operation. One practical disadvantage of the dual simplex method is that all of the n-m inner products $\alpha_j = u^T a_j$, $j \notin \underset{\sim}{B}$

must be performed, while in the primal method, one need only compute inner products $\pi^T a_j$ 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 nxn basis matrix $\hat{B}$, in contrast with the mxm 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 con-

straints, 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

$$\begin{aligned} &\text{minimize} && z(\theta) = (c + \theta d)^T x \\ &\text{subject to} && Ax = b,\ x \geq 0 \end{aligned} \tag{4.14}$$

Suppose that we have an optimal basic feasible solution for $\theta = \theta_0$, and we wish to determine the interval $\underline{\theta} \leq \theta \leq \overline{\theta}$ 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., $(c_N + \theta d_N)^T - (c_B + \theta d_B)^T B^{-1} N \geq 0$. Defining reduced costs $\bar{c}_N^T = c_N^T - c_B^T B^{-1} N$ and $\bar{d}_N^T = d_N^T - d_B^T B^{-1} N$, in terms of c and d alone, the above condition becomes $\bar{d}_N^T \geq -\bar{c}_N^T$. Consequently, the range is

$$\underline{\theta} = \max \{\max \{-\bar{c}_j/\bar{d}_j \mid \bar{d}_j > 0,\ j \notin B\} , -\infty\} \leq \theta \leq$$

$$\min \{\min \{-\bar{c}_j/\bar{d}_j \mid \bar{d}_j < 0,\ j \notin B\} , \infty\} = \overline{\theta}. \tag{4.15}$$

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

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

If $\theta_0 = 0$ and we choose $d = e_j$, then $[c_j + \underline{\theta}, c_j + \overline{\theta}]$ 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 $[\underline{\theta},\overline{\theta}]$ 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 the interval $[\underline{\underline{\theta}}, \underline{\theta}]$, assuming that $-\infty < \underline{\underline{\theta}}$, 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 $\underline{\theta} = -\overline{c}_j/\overline{d}_j$ 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 $[\overline{\theta},\overline{\overline{\theta}}]$ for $\theta \geq \overline{\theta}$.

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 number of iterations which are quadratic in the problem size (see section 7).

Consider now the right hand side parametric linear program

$$\text{minimize} \quad z(\theta) = c^T x$$

$$\text{subject to} \quad Ax = b + \theta d, \; x \geq 0.$$

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

$$\theta = \max\left\{ \max_{1 \leq i \leq m} \left\{ -\bar{b}_i/\bar{d}_i \mid \bar{d}_i > 0 \right\}, -\infty \right\} \leq \theta \leq$$

$$\min\left\{ \min_{1 \leq i \leq m} \left\{ -\bar{b}_i/\bar{d}_i \mid \bar{d}_i < 0 \right\}, \infty \right\} = \overline{\theta}.$$

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. Exploiting Structure.

The computation required by each iteration of the revised simplex method (section 3.2) is usually dominated by the solution of the linear systems $B^T\pi = c_B$ and $Bw = a_q$ and the update of the basis inverse representation (in product form or using an LU factorization - see section 3.3). Since B is an $m \times m$ nonsingular matrix, and large problems may have m of the order of 10^4, it is imperative to exploit any structure in the coefficient matrix A to decrease the computational work. In this section we discuss two kinds of special structure: upper bounds and network constraints. The next section considers other approaches to handling very large structured problems efficiently.

5.1 Upper Bounds and Compact (Partitioned) Inverse Methods.

Modifying the revised simplex method to handle upper-bounded variables efficiently can be motivated directly as in section 3. However, we will treat a more general case to give a flavor of the power of "compact inverse" methods and refer to the upper-bounded case as a simple illustration.

The methods we discuss here exploit the simple idea that if B has special structure, we may be able to solve systems with coefficient matrix B by solving systems of smaller dimension. More precisely, we will partition B of order m into smaller matrices of order $k \times k$, $k \times \ell$, $\ell \times k$ and $\ell \times \ell$ such that solving a system with coefficient matrix B is reduced to solving two (trivial) systems of order ℓ and one system of order k involving a smaller matrix called the working basis.

This idea has appeared before, in section 3: the key matrix there was M in (3.3), whose inverse was given in terms of B^{-1} in (3.4). In that context, M was the "basis", I the coefficient matrix in the "trivial" system, and B the "working basis". Thus the revised simplex method automatically exploits the special structure of the nonnegativities to deal with the $m \times m$ matrix B instead of the $n \times n$ matrix M. We can often decrease the dimension further by exploiting structure within B.

Consider the linear programming problem in standard form:

$$\begin{aligned} &\min\ c^T x \\ (P)\qquad &\begin{pmatrix} A \\ R \end{pmatrix} x \equiv \hat{A}x = \hat{b} \equiv \begin{pmatrix} b \\ r \end{pmatrix} \\ &x \geq 0 \end{aligned}$$

where we assume that $\hat{A}$ has full row rank m, and that the constraints have been partitioned into "general" constraints $Ax = b$ and "special" constraints $Rx = r$, where A and b have k rows, R and r have ℓ rows, and $k + \ell = m$.

Example 5.1. The upper bounded problem

$$\begin{aligned} \min\quad & \tilde{c}^T \tilde{x} \\ & \tilde{A}\,\tilde{x} = \tilde{b} \\ & 0 \leq \tilde{x} \leq \tilde{r} \end{aligned}$$

where $\tilde{A}$ has full row rank and $\tilde{r} > 0$, is an instance of (P) if we set

$$x^T = (\tilde{x}^T, \tilde{y}^T)$$

$$A = [\tilde{A}, 0], \quad b = \tilde{b}$$

$$R = [I, I], \quad r = \tilde{r}, \text{ and}$$

$$c^T = [\tilde{c}^T, 0]$$

so that $\tilde{y}$ is the vector of slack variables for the upper bounds.

Suppose we have a basic feasible solution of (P) whose first $m = k + \ell$ components are basic, and partition the cost vector and coefficient matrices as follows:

$$\begin{bmatrix} c^T \\ A \\ R \end{bmatrix} = \begin{bmatrix} c_B^T & c_C^T & c_D^T \\ B & C & D \\ S & T & H \end{bmatrix} \begin{matrix} 1 \\ \}k \\ \}\ell \end{matrix} \quad . \tag{5.1}$$

$$\qquad\qquad\quad k \qquad \ell \qquad n-m$$

The usual basis matrix is then

$$\hat{B} = \begin{bmatrix} B & C \\ S & T \end{bmatrix} \quad . \tag{5.2}$$

Since $[S,T]$ is of full rank ℓ it contains a nonsingular $\ell \times \ell$ submatrix. Assume the variables have been ordered so that T is nonsingular. We wish to express $\hat{B}^{-1}$ in terms of smaller inverses.

These involve the matrix T (here "T" stands for trivial: linear systems involving T are assumed very easy to solve) and the matrix

$$W = B - CT^{-1}S$$

(the working basis). Indeed, if we make row operations to eliminate block C, we find

$$\hat{B} = \begin{bmatrix} I & CT^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} W & 0 \\ S & T \end{bmatrix},$$

so that

$$\hat{B}^{-1} = \begin{bmatrix} W^{-1} & 0 \\ -T^{-1}SW^{-1} & T^{-1} \end{bmatrix} \begin{bmatrix} I & -CT^{-1} \\ 0 & I \end{bmatrix} \tag{5.3}$$

$$= \begin{bmatrix} W^{-1} & -W^{-1}CT^{-1} \\ -T^{-1}SW^{-1} & T^{-1}+T^{-1}SW^{-1}CT^{-1} \end{bmatrix}.$$

Henceforth we assume that we have W^{-1} and T^{-1} explicitly or in product form, or LU factorizations of W and T.

Example 5.1 (continued).

Since the (k+j)th row of $\hat{A}$ only has nonzeroes in the columns corresponding to $\tilde{x}_j$ and $\tilde{y}_j$, at least one of these variables must be basic at any basic solution, else $\hat{B}$ would have a zero row. If $\tilde{y}_j$ is basic, call it the jth key variable; otherwise, call $\tilde{x}_j$ the jth key variable. Now order the basic variables so that the key variables, in order, come last. Then T will be the identity matrix. For instance, if $k = 1$ and $\ell = 3$, so that $n = 2\ell = 6$, we have

$$\hat{A} = \begin{bmatrix} a_1 & a_2 & a_3 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

where a_j is the jth column of A (or of $\tilde{A}$). If $\tilde{x}_1$, $\tilde{x}_2$, $\tilde{y}_2$ and $\tilde{y}_3$ are basic, then $\tilde{x}_1, \tilde{y}_2$ and $\tilde{y}_3$ are the key variables, and

$$\hat{B} = \begin{bmatrix} a_2 & a_1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} B & C \\ S & T \end{bmatrix} \qquad (5.4)$$

Thus T is the 3×3 identity matrix, and $W = B - CT^{-1}S = B - CS = B$ is the 1×1 matrix a_2. Indeed we have

Lemma 5.1. In the upper-bound case, we always have $W = B$.

Proof. Our choice of key variables guarantees that $T = I$, so that $W = B - CT^{-1}S = B - CS$. Moreover, if the ith column of S is the jth unit vector, then the ith basic variable is $\tilde{x}_j$ which is not key, so the jth key variable is $\tilde{y}_j$. But then the jth column of C corresponds to a slack variable and is therefore 0. Thus each column of CS is the zero vector whence $W = B$.

Note that B consists of the columns of $\tilde{A}$ corresponding to basic variables $\tilde{x}_j$ whose slacks $\tilde{y}_j$ are also basic. In the nondegenerate case, therefore, the working basis consists of columns of $\tilde{A}$ corresponding to variables that are strictly between their lower and upper bounds.

We now return to the general case, and consider first the computation of the simplex multipliers $\hat{\pi}^T = (\pi^T, \rho^T)$, where π is a k-vector and ρ an ℓ-vector. Thus

$$(\pi^T, \rho^T) = (c_B^T, c_C^T)\hat{B}^{-1}$$

$$= (c_B^T, c_C^T) \begin{bmatrix} W^{-1} & 0 \\ -T^{-1}SW^{-1} & T^{-1} \end{bmatrix} \begin{bmatrix} I & -CT^{-1} \\ 0 & I \end{bmatrix}$$

from (5.3). Hence if

$$(\alpha^T,\beta^T) = (c_B^T, c_C^T) \begin{bmatrix} W^{-1} & 0 \\ -T^{-1}SW^{-1} & T^{-1} \end{bmatrix}$$

we find $\pi^T = \alpha^T$ and $\rho^T = -\alpha^T CT^{-1} + \beta^T$. Therefore we calculate

$$\beta^T = c_C^T T^{-1}$$

$$\pi^T = (c_B^T - \beta^T S)W^{-1}, \text{ and} \tag{5.5}$$

$$\rho^T = \beta^T - \pi^T CT^{-1}$$

to get $\hat{\pi}^T = (\pi^T,\rho^T)$. Note that we have organized the calculations so that T^{-1} appears twice (in the expressions for β^T and ρ^T) and W^{-1} once (in that for π^T).

We use $\hat{\pi}^T$ to price out nonbasic columns; thus if such a column is denoted $\hat{a}_j = \begin{pmatrix} d_j \\ h_j \end{pmatrix}$, with cost c_j, we find the reduced cost

$$\bar{c}_j = c_j - \hat{\pi}^T \hat{a}_j = c_j - \pi^T d_j - \rho^T h_j. \tag{5.6}$$

If all $\bar{c}_j$'s are nonnegative, the current solution is optimal; otherwise we choose q with $\bar{c}_q < 0$.

Example 5.1 (continued). In the case of upper bounds, T is the identity so that $\beta^T = c_C^T$. It is also easy to check as in the proof of Lemma 5.1 that $\beta^T S = 0$, so that

$$\pi^T = c_B^T W^{-1} = c_B^T B^{-1}. \tag{5.7}$$

Using the fact that T is the identity again and the form of β^T and C, we find that

$$\rho_j = \begin{cases} c_j - c_B^T B^{-1} a_j & \text{if the jth key variable is } \tilde{x}_j \\ \\ 0 & \text{if the jth key variable is } \tilde{y}_j. \end{cases} \tag{5.8}$$

Now consider pricing out the nonbasic columns. If $x_{\hat{j}} = \tilde{x}_j$ is nonbasic, then $\tilde{y}_j$ is the jth key variable and the reduced cost is

$$\bar{c}_{\hat{j}} = c_j - \pi^T a_j - \rho^T e_j = c_j - c_B^T B^{-1} a_j \tag{5.9}$$

using (5.6)-(5.8), since ρ_j is zero. On the other hand, if $x_{\hat{j}} = \tilde{y}_{\hat{j}}$ is nonbasic, then $\tilde{x}_j$ is the jth key variable and the reduced cost is

$$\bar{c}_{\hat{j}} = 0 - \pi^T 0 - \rho^T e_j = -\rho_j = -(c_j - c_B^T B^{-1} a_j). \tag{5.10}$$

Hence the rule for entering variables is simple and intuitive: if $\tilde{x}_j$ is nonbasic at its lower bound 0, then it is eligible to enter the

basis of its "usual" reduced cost as in (5.9) is negative. On the other hand, if $\tilde{x}_j$ is at its upper bound $\tilde{r}_j$ (so that the slack variable $\tilde{y}_j$ is nonbasic), then $\tilde{y}_j$ is eligible to enter the basis (and hence $\tilde{x}_j$ to decrease from its upper bound) if $\bar{c}_j$ in (5.10) is negative, so that the "usual" reduced cost $c_j - c_B^T B^{-1} a_j$ is positive.

Having determined an entering variable x_q, we need to calculate the updated column w, which we partition into $u \in R^k$ and $v \in R^\ell$:

$$w = \begin{pmatrix} u \\ v \end{pmatrix} = \hat{B}^{-1} \begin{pmatrix} d_q \\ h_q \end{pmatrix}$$

$$= \begin{bmatrix} W^{-1} & 0 \\ -T^{-1}SW^{-1} & T^{-1} \end{bmatrix} \begin{bmatrix} I & -CT^{-1} \\ 0 & I \end{bmatrix} \begin{pmatrix} d_q \\ h_q \end{pmatrix} \quad (5.11)$$

Thus we compute

$$t = T^{-1} h_q,$$

$$u = W^{-1}(d_q - Ct), \text{ and} \quad (5.12)$$

$$v = t - T^{-1} S u.$$

Again, T^{-1} occurs twice above and W^{-1} once.

If $w = \begin{pmatrix} u \\ v \end{pmatrix}$ is nonpositive, we conclude that (P) is unbounded; otherwise we use w in a minimum ratio test to determine the leaving basic variable and to update the values of the basic variables.

The final step of the iteration is to obtain (representations of) the inverses of the new matrices W and T. This can be the most complicated part of the iteration. We will now describe a special case which reduces to a standard basis inverse update, but the procedure to be followed in general depends very much on the particular structure in the matrix R.

The special case is when the leaving column is in the first part of the basis matrix $\hat{B}$. In that case, C and T remain unchanged while the new column, say $\begin{pmatrix} d_q \\ h_q \end{pmatrix}$, replaces the column $\begin{pmatrix} b_p \\ s_p \end{pmatrix}$ of $\hat{B}$. Notice that $W = B - CT^{-1}S$ has columns

$$We_j = b_j - CT^{-1}s_j \tag{5.13}$$

corresponding directly to the columns of $\begin{bmatrix} B \\ S \end{bmatrix}$. Thus W becomes $\overline{W}$ where $d_q - CT^{-1}h_q$ replaces the old column We_p; moreover, $d_q - CT^{-1}h_q = d_q - Ct$ has already been computed in (5.12). Hence in this case our representation of T^{-1} is unchanged, while that for W^{-1} is updated as in a standard column exchange, as discussed in section 3.4.

Example 5.1 (concluded) We assume that $\tilde{y}_j$ is entering the basis; the case for $\tilde{x}_j$ is similar. Then $d_q = 0$ and $h_q = e_j$.

Moreover, $\tilde{x}_j$ is then the jth key variable, and thus the jth column of C is a_j and $\tilde{x}_j$ is currently at its upper bound $\tilde{r}_j$. Using (5.12) we find

$$t = T^{-1}h_q = h_q = e_j;$$

$$u = W^{-1}(d_q - Ct) = W^{-1}(0 - Ce_j) \qquad (5.14)$$

$$= W^{-1}(-a_j) = -B^{-1}a_j; \text{ and}$$

$$v = t - T^{-1}Su = e_j + SB^{-1}a_j.$$

Note that the jth row of S is zero since $\tilde{x}_j$ is key and $\tilde{y}_j$ nonbasic; thus $v_j = 1 > 0$, so that we cannot get an indication of unboundedness (which would be most unexpected in a problem with all the variables bounded). Moreover, apart from $v_j = 1$, each component of $w = \binom{u}{v}$ is either zero, or a component of $-B^{-1}a_j$ or of $+B^{-1}a_j$, since each row of S is zero or a unit vector. The entry one corresponds to $\tilde{x}_j$, which decreases by one for each unit $\tilde{y}_j$ is increased, while the entries 0 correspond to variables $\tilde{x}_j$ or $\tilde{y}_j$ which are at their upper bounds and do not change as $\tilde{y}_j$ is increased. The other entries, components of $\pm B^{-1}a_j$, indicate the necessary changes in components $\tilde{x}_i$ and $\tilde{y}_i$ between their bounds as $\tilde{y}_j$ increases; the different signs occur since if $\tilde{x}_i$ increases, $\tilde{y}_i$ must decrease and vice-versa.

The different minimum ratios that might occur correspond to such a $\tilde{x}_i$ or $\tilde{y}_i$ becoming zero; if the minimum ratio occurs in row $k + j$, corresponding to $v_j = 1$, it indicates that $\tilde{x}_j$ has become nonbasic, i.e. it has travelled from its upper to its lower bound. In this last case, $\tilde{y}_j$ replaces $\tilde{x}_j$ as the jth key variable, T remains the identity, and $W = B$ remains unchanged! However, the values of the basic variables do change and the jth column of C changes from a_j to 0.

We have indicated above the easy case where $\tilde{y}_j$ replaces $\tilde{x}_j$; in addition, if $\tilde{y}_j$ replaces a variable that is not key, then T is unchanged and $W = B$ suffers a column exchange as described for the general case. Let us address the final case, where $\tilde{y}_j$ replaces a key variable other than $\tilde{x}_j$. Since each basis must contain either $\tilde{x}_i$ or $\tilde{y}_i$ for each i, the replaced variable must be a $\tilde{y}_i$ with $\tilde{x}_i$ also basic. Thus $\tilde{x}_i$ must become the new ith key variable, $\tilde{x}_j$ must fill its place in the non-key basic variables, and $\tilde{y}_j$ must replace $\tilde{x}_j$ as the jth key variable. (For example, consider the basis of (5.4), and suppose $\tilde{y}_1$ enters the basis and replaces $\tilde{y}_2$, so that $j = 1$ and $i = 2$. Then the new key variables will be $\tilde{y}_1$, $\tilde{x}_2$ and $\tilde{y}_3$, with $\tilde{x}_1$ the new non-key basic variable.) T remains the identity, and $W = B$, the matrix of columns associated with variables strictly between their bounds (assuming nondegeneracy) suffers a column exchange: a_j replaces a_i. Finally, a_i and 0 replace the ith and jth columns of C respectively, and e_j replaces the column of S that had been e_i. (In the example above, $\hat{B}$ becomes

$$\begin{bmatrix} a_1 & 0 & a_2 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ,$$

and it is easy to check in general that these are precisely the required changes.) Hence, after updating the representation of the inverse of W (which is either unchanged or undergoes a column exchange), we are ready for the next iteration.

The compact inverse idea has been applied to other forms of special constraints: generalized upper bounds [Dantzig and Van Slyke 1967], variable upper bounds [Schrage 1975], [Todd 1982], and network constraints, e.g. [Klingman and Russell 1975]. An elementary treatment of generalized upper bounds can be found for example in [Chvatal 1983] and a geometric view of compact inverse methods is presented by Todd [1983]. Our aim here has been to describe the basic ideas of the approach and illustrate them on a particularly simple problem, which is treated directly in most linear programming texts.

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 $(i_0,e_1,i_1,\ldots,e_\ell,i_\ell)$ 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 $1 \leq k \leq \ell$. It is from i_0 to i_ℓ and of length ℓ. A sequence satisfying all these requirements except that $i_0 = i_\ell$ 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 $\Sigma_{i \in V} b_i = 0$, and $c = (c_e)_{e \in E}$ is a vector of costs. Then the transshipment problem

$$\min \sum_{e \in E} c_e x_e$$

$$(P) \qquad \sum_{e:t(e)=i} x_e - \sum_{e:h(e)=i} x_e = b_i, \quad i \in V$$

$$x_e \geq 0, \quad e \in E$$

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

$$\begin{aligned} \min \quad & c^T x \\ & \hat{A} x = \hat{b} \\ & x \geq 0. \end{aligned}$$

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

$$\min c^T x$$

$$A x = b$$

$$x \geq 0.$$

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 are 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 $H' = (V\backslash\{i\}, E\backslash\{e\})$. By Lemma 5.2(ii) it is a spanning tree of $G' = (V\backslash\{i\}, E\backslash\{e\})$, 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

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

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

Lemma 5.3 show 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 $Ax = b$, $x \geq 0$ 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 Chvatal [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 $B^T\pi = c_B$ and $Bw = a_e$. Since B can be reordered to make it triangular, these systems can be solved very fast.

Let $e = (u,v) \in E\backslash F$. 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 depth $d(i)$ 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.

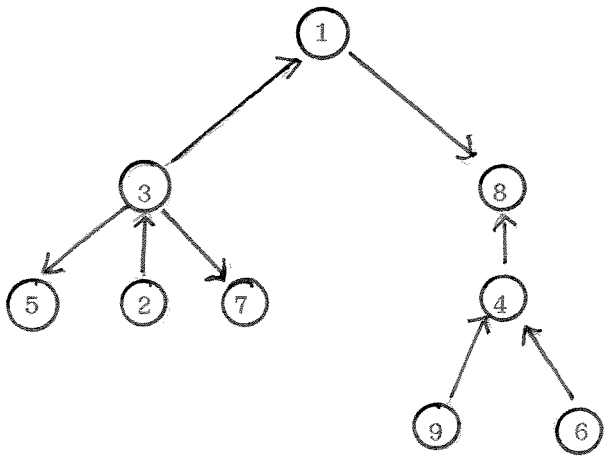
The system $B^T\pi = c_B$ can be written as

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

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:



and where the numbers adjacent to the arcs indicate their current flow x_f. The tree is represented by the three vectors p, d and s of length $|V|$ as follows:

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 $B^T\pi = c_B$, or $\pi_j - \pi_i = c_f$ for $f = (i,j) \in F$, where $\pi_r \equiv 0$. To do this, we calculate the π_i'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 $\pi_j - \pi_i \leq c_e$ 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 $e = (u,v) \in E\backslash F$ 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, $f = (1,8)$.

6) Update: $F \leftarrow (F \cup \{e\}) \backslash \{f\}$.

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 $d(i) > d(j)$.

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

$$\text{While } d(k) > d(j)$$

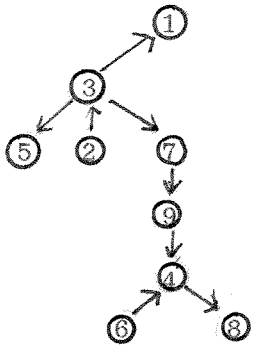
$$\pi_k \leftarrow \pi_k + \bar{c}_e$$

$$k \leftarrow s(k).$$

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., Chvatal [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



with representation

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

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 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

$$\begin{aligned} \min\ z &= c^T x \\ Ax &= b \\ x &\geq 0 \end{aligned} \tag{6.1}$$

where A is $m \times n$ and the columns of A are only known implicitly (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

$$\begin{aligned} \min\ z &= c^T x \\ A_0 x &= b_0 \\ A_1 x &= b_1 \\ x &\geq 0, \end{aligned} \tag{6.2}$$

where the constraints have been partitioned into "general" constraints $A_0 x = b_0$ and "special" constraints $A_1 x = b_1$, $x \geq 0$. 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,...,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 jth pattern, i = 1,2,...,m, j = 1,2,...,n. For example, the pattern described above yields a column $(3,1,0,\ldots,0)^T$ 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 ($z = \Sigma_j x_j$) 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 $B^T\pi = e$ 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

$$\bar{c}_j = 1 - \pi^T a_j \qquad (6.3)$$

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

$$\begin{aligned} \max_a \quad & \pi^Ta \\ & w^Ta \leq W \\ & a \in Z_+^m. \end{aligned} \tag{6.4}$$

This is an instance of the so-called knapsack problem. (Think of π_i as the value and w_i the weight of the ith 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 $0 \leq v < w_{min}$, where $w_{min} = \min w_i$. We obtain $f(W)$ by using the recursion

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

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 rewite as

$$\begin{aligned} \min\ z &= c^T x \\ A_0 x &= b_0 \\ x &\in X \end{aligned} \qquad (6.5)$$

where

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

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

$$\begin{aligned} \min\ & c_0^T x_0 + c_1^T x_1 + \ldots + c_k^T x_k \\ & A_{00} x_0 + A_{01} x_1 + \ldots + A_{0k} x_k = b_0 \\ & A_{11} x_1 \qquad\qquad\quad = b_1 \\ & \qquad\ddots \qquad\qquad\quad \vdots \\ & \qquad\qquad\quad A_{kk} x_k = b_k \\ & x_0, x_1, \ldots, x_k \geq 0, \end{aligned} \tag{6.7}$$

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 $c^T = (c_0^T, c_1^T, \ldots, c_k^T)$ and $A_0 = [A_{00}, A_{01}, \ldots, A_{0k}]$, and $X = \{x = (x_0^T, x_1^T, \ldots, x_k^T)^T : x_0 \geq 0,\ A_{jj}x_j = b_j,\ x_j \geq 0,\ j = 1, \ldots, k\}$. 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

$$\begin{aligned} \min\ & c_1^T x_1 + \ldots + c_k^T x_k \\ & A_{01}x_1 + \ldots + A_{0k}x_k = b_0 \\ & A_{11}x_1 \qquad\qquad\quad = b_1 \\ & \qquad\qquad\quad \ddots \qquad\quad \vdots \\ & \qquad\qquad\quad A_{kk}x_k = b_k \\ & x_1, \ldots, x_k \geq 0. \end{aligned} \tag{6.8}$$

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

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

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

as

$$\begin{aligned} \min\ & c_1^T x_1 + \ldots + c_k^T x_k \\ & A_{01}x_1 + \ldots + A_{0k}^T x_k = b_0 \\ & x_1 \in X_1, \ldots, \quad x_k \in X_k, \end{aligned} \tag{6.9}$$

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

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

$$SP_j(\pi_0) \qquad \begin{aligned} \min\ & (c_j^T - \pi_0^T A_{0j})x_j \\ & x_j \in X_j. \end{aligned} \tag{6.10}$$

Of course, the corporation would like to set the prices $-\pi_0$ so that, when each division solves its corresponding subproblem $SP_j(\pi_0)$ to get an optimal solution $\overline{x}_j$, $\sum_j A_{0j}\overline{x}_j = b_0$. 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 $(\overline{\pi}_0, \overline{x}_1, \ldots, \overline{x}_k)$ an <u>equilibrium</u> if $\overline{x}_j$ solves $SP_j(\overline{\pi}_0)$ for each j and $\sum_j A_{0j}\overline{x}_j = b_0$.

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

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

$$A_{0j}^T\overline{\pi}_0 + A_{jj}^T\overline{\pi}_j \leq c_j,\ (A_{0j}^T\overline{\pi}_0 + A_{jj}^T\overline{\pi}_j - c_j)^T\overline{x}_j = 0 \qquad (6.11a)$$

But then

$$A_{jj}^T\overline{\pi}_j \leq (c_j - A_{0j}^T\overline{\pi}_0),\ (A_{jj}^T\overline{\pi}_j - (c_j - A_{0j}^T\overline{\pi}_0))^T\overline{x}_j = 0, \qquad (6.11b)$$

which shows again by duality that $\overline{x}_j$ is primal optimal (and $\overline{\pi}_j$ dual optimal) in $SP_j(\overline{\pi}_0)$. For the converse, let $\overline{\pi}_j$ be an optimal dual solution to $SP_j(\overline{\pi}_0)$. By dualilty, we have (6.11b), and hence (6.11a), which implies that $(\overline{x}_1,\ldots,\overline{x}_k)$ and $(\overline{\pi}_0,\overline{\pi}_1,\ldots,\overline{\pi}_k)$ 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 $\overline{\pi}_0$, then it appears that we could solve (6.8) by solving k small linear programming problems (the $SP_j(\overline{\pi}_0)$) instead of one large one.

Unfortunately, two difficulties now arise. First, it is far from clear how a suitable vector $\overline{\pi}_0$ can be found. Second, even if an appropriate $\overline{\pi}_0$ were known, obtaining $\overline{x}_1,\ldots,\overline{x}_k$ would not be easy. Indeed, given nondegeneracy, an optimal solution to (6.8) will have $m_0 + \Sigma_{j\geq 1} m_j$ positive variables, where b_j is an m_j-vector for each j. On the other hand, a basic optimal solution to $SP_j(\overline{\pi}_0)$ will have only m_j positive variables, so that putting these together will yield only $\Sigma_{j\geq 1} m_j$ positive variables. The conclusion is that any equilibrium $\overline{\pi}_0$ 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 represent the polyhedron X in (6.6) in terms of its vertices and (certain of its) directions, as in Theorem 2.4. We need a slight extension:

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

$$d = \mu_1 d_1 + \mu_2 d_2, \ \mu_1 \geq 0, \ \mu_2 \geq 0,$$

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

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 $\sum_{i \in I} \lambda_i = 1$, $\lambda_i \geq 0$ for all $i \in I$, $\mu_j \geq 0$ 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 2.4. An important consequence is

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

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

(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 <u>master problem</u>

$$\begin{aligned} \min\ & (c^TV)\lambda + (c^TD)\mu \\ & (A_0V)\lambda + (A_0D)\mu = b_0 \\ & e^T\lambda = 1 \\ & \lambda \geq 0,\ \mu \geq 0. \end{aligned} \tag{6.12}$$

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 $m_0 + 1$ 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), $(\overline{\lambda},\overline{\mu})$, with associated simplex multipliers $\overline{\pi}_0$ (for the first m_0 constraints) and $\overline{\sigma}$. Here $\overline{\lambda}_i > 0$ implies that we know the corresponding vertex v_i of X, and similarly for $\overline{\mu}_j$ 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

$$c^T v_i - \bar{\pi}_0^T A_0 v_i - \bar{\sigma} < 0 \qquad (6.13)$$

or an extreme direction d_j with reduced cost

$$c^T d_j - \bar{\pi}_0^T A_0 d_j < 0; \qquad (6.14)$$

if 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 $(c^T - \bar{\pi}_0^T A_0)v_i$ 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

$$SP(\bar{\pi}_0) \quad \min\{(c^T - \bar{\pi}_0^T A_0)x: \ x \in X\} \qquad (6.15)$$

(cf. (6.10)).

Let us discuss each possible outcome of solving $SP(\bar{\pi}_0)$. 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, $SP(\bar{\pi}_0)$ 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 $\theta \geq 0$ and $(c^T - \bar{\pi}_0^T A_0)\eta_q < 0$. In fact η_q is of the form

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

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 $A_0x = b_0$.

Finally, $SP(\overline{\pi}_0)$ may have a finite optimal solution $\overline{x}$. Then, by the fundamental theorems of linear programming (see in particular Theorem 2.7 and its proof), $(c^T - \overline{\pi}_0^T A_0)d \geq 0$ for all directions d and $\overline{x}$ can be taken to be a vertex of X (and the revised simplex method finds such a vertex). Hence (6.14) fails for all j; no column arising from an extreme direction is a candidate for entering the basis. If $(c^T - \overline{\pi}_0^T A_0)\overline{x} \geq \sigma$ 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 $(\overline{\lambda}, \overline{\mu})$ is optimal in (6.12) and $x^* = V\overline{\lambda} + D\overline{\mu}$ optimal in (6.5). On the other hand, if $(c^T - \overline{\pi}_0^T A_0)\overline{x} < \sigma$, then setting $v_i = \overline{x}$ 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 $SP(\overline{\pi}_0)$ is unbounded, the revised simplex method applied to it yields an extreme direction d_j satisfying (6.14), so that the column

$$\begin{bmatrix} A_0 & d_j \\ 0 & \end{bmatrix} \quad \text{with cost} \quad c^T d_j$$

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

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

$$\begin{bmatrix} A_0 & v_j \\ 1 & \end{bmatrix} \quad \text{with cost} \quad c^T v_i$$

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

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

Proof. We only need to show the last part. Clearly, since (6.13) and (6.14) fail for all $i \in I$, $j \in J$, $(\overline{\pi}_0,\overline{\sigma})$ is feasible in the dual of (6.12), and hence $(\overline{\lambda},\overline{\mu})$ and $(\overline{\pi}_0,\overline{\sigma})$ are respectively primal and

dual optimal so that $c^T V\bar{\lambda} + c^T D\bar{\mu} = \bar{\pi}_0^T b_0 + \bar{\sigma}$. 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 $c^T x^* = c^T V\bar{\lambda} + c^T D\bar{\mu}$, the optimal value of (6.12). Now since $\bar{\pi}_1$ is dual optimal in $SP(\bar{\pi}_0)$, it is dual feasible:

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

and has value

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

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 $SP(\bar{\pi}_0)$ is precisely $\bar{\sigma}$, and, by complementary slackness, all v_i with $\bar{\lambda}_i$ positive will be alternate optimal solutions to $SP(\bar{\pi}_0)$. 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 $SP(\pi_0^*)$ 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 $SP(\bar{\pi}_0)$ 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

$$\min\ c_0^T x_0 + (c_1^T V_1)\lambda_1 + (c_1^T D_1)\mu_1 + \ldots + (c_k^T V_k)\lambda_k + (c_k^T D_k)\mu_k$$

$$A_{00}x_0 + (A_{01}V_1)\lambda_1 + (A_{01}D_1)\mu_1 + \ldots + (A_{0k}V_k)\lambda_k + (A_{0k}D_k)\mu_k = b \qquad (6.18)$$

$$\begin{aligned} e^T\lambda_1 &= 1 \\ &\vdots \\ e^T\lambda_k &= 1 \end{aligned}$$

$$x_0,\ \lambda_1,\ \mu_1, \ldots, \lambda_k, \mu_k \geq 0.$$

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 $\overline{\pi}_0, \overline{\sigma}_1, \ldots, \overline{\sigma}_k$. If any x_0-variable is eligible to enter the basis, we make the appropriate pivot. Otherwise, we solve $SP_j(\overline{\pi}_0)$ (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 $SP_j(\overline{\pi}_0)$ is less than $\overline{\sigma}_j$, then the optimal vertex v_{ji} again yields an eligible column. Finally, if all optimal values are equal to the corresponding $\overline{\sigma}_j$, then we have the optimal solution to (6.18), and $x_0^* = \overline{x}_0$, $x_j^* = V_j\overline{\lambda}_j + D_j\overline{\mu}_j$, $j = 1,2,\ldots,k$, 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 $SP(\overline{\pi}_0)$ (or all $SP_j(\overline{\pi}_0)$'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 $\overline{\sigma}$ and the optimal value of $SP(\overline{\pi}_0)$. 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.

7. The Complexity of Linear Programming.

The previous sections have described in some detail the solution of linear programming problems by the simplex method. The next sections will discuss two new methods, which enjoy the desirable property of polynomial-time boundedness. Here we wish to motivate briefly these new developments by outlining what is known about the complexity of the simplex method. Quite complete surveys are provided by Shamir [1987] and Megiddo [1987].

A vast amount of practical experience in solving real-world linear programming problems has confirmed that the number of iterations required grows roughly linearly with m, the number of rows of the constraint matrix, and sublinearly with n, the number of columns. Since, in most problems, n is a small multiple of m, usually at most 10m, the folklore claims that the typical number of iterations for both Phase I and Phase II is between m and 4m. Qualitatively similar results have been obtained by several authors using Monte Carlo experimentation. For details, see [Shamir 1987].

In contrast to its excellent empirical performance, the simplex method can be exceedingly slow on specially-constructed examples. This was first demonstrated by Klee and Minty [1972], who gave a class of d $\times$ 2d examples for which the simplex variant employing the most negative reduced cost rule required 2^d-1 iterations, visiting every vertex of the feasible region. Similar behavior was shown by others for several different simplex variants and for network problems.

Thus there is a large gap between practical experience and theoretical (worst-case) bounds. The main distinction here is between a polynomially growing number of iterations and an exponentially growing number. Actually, real problems seem to require only $O(m)$ iterations, but each might require up to $O(mn)$ arithmetic operations. (We write $O(f(n))$ for a function that is bounded by some constant times $f(n)$ as n tends to infinity.) A major open question is whether there is a simplex variant requiring only a polynomial (in m and n) number of iterations for any $m \times n$ linear programming problem.

The next sections describe two radically new methods for solving linear programming problems, the ellipsoid method and Karmarkar's projective method, which require only a polynomial amount of time (in terms of m, n and the number of bits L necessary to define the problem). Of course, it is natural that L should occur in the number of bit operations required, since it determines the precision necessary to state the optimal solutions exactly. However, the algorithms to be described require, in theory, a multiple of L iterations, even if exact arithmetic is allowed. The question thus arises as to whether a <u>strongly polynomial</u> algorithm exists for linear programming, one which requires a number of elementary arithmetic operations that is polynomial in m and n only and which, when applied to problems with rational data of length L, generates numbers whose sizes (numbers of bits) are bounded by polynomials in m, n and L. While this question

remains open, Tardos [1986] has shown how to use polynomial linear programming algorithms to generate strongly polynomial algorithms for those classes of problems where the input length of the constraint matrix A is bounded by a polynomial in m and n. (Note that no restriction is placed on the input length of b and c.) Roughly speaking, to solve

$$(P) \quad \min \{c^T x: Ax = b, x \geq 0\}$$

Tardos uses a subroutine to solve a related problem

$$(\hat{P}): \quad \min \{\hat{c}^T x: Ax = b, x \geq 0\},$$

where each component of $\hat{c}$ is a rational whose size is bounded by a polynomial in m and n. If $\hat{c}$ is appropriately defined, then any index j for which $\hat{c}_j - a_j^T \hat{y}$ is sufficiently large has the property that x_j must be zero in any optimal solution to (P) (here $\hat{y}$ is an optimal dual solution to $(\hat{P})$). In this way, variables are eliminated recursively. When no further eliminations are possible all feasible solutions to the current $(\hat{P})$ are optimal, and a feasible solution is found using similar ideas. If the components of b are "large", then the subroutine works by dualizing $(\hat{P})$ and proceeding as above, first approximating b by some $\hat{b}$, etc. Details may be found in [Tardos 1986].

Tardos' algorithm clearly applies to network flow problems, and in fact was originally developed in that setting. Many strongly

polynomial algorithms have been derived for special network flow problems such as the assignment, shortest path and maximum flow problem. See Chapter 4.

Megiddo [1984] has given an algorithm for linear programming problems whose time-complexity is linear in the number of constraints when the dimension d is fixed; however the complexity grows faster than exponentially with d. See [Megiddo 1987] for a description of this method and later developments.

To conclude this section we describe work done to explain the large gap between practice and theory for the simplex method using probabilistic analysis. A deterministic simplex variant is chosen, along with a probability distribution (or class of distributions satisfying certain properties) on problem instances. Then it is shown that the average number of iterations required by the chosen variant on problems generated randomly from the chosen distribution grows slowly with m or n or both. Currently the best bound of this nature, $O(\min\{m^2,n^2\})$, has been obtained independently by Adler, Karp and Shamir [1983], Adler and Megiddo [1985] and Todd [1986a], for a lexicographic variant of Dantzig's self-dual parametric algorithm [Dantzig 1963]. The probabilistic model assumes that the problem instances $\max\{c^Tx: Ax \leq b, x \geq 0\}$ are generated according to some distribution that satisfies certain nondegeneracy conditions as well as the sign-invariance property that if S and T are diagonal matrices with diagonal entries ± 1, then (SAT, Sb, Tc) has the same distribution as (A,b,c). Notice that setting the ith diagonal entry of S (T) to -1 is equivalent to reversing the sense of the ith inequality of $Ax \leq b$ $(x \geq 0)$.

Significant work on the problem has also been done by Borgwardt, Smale and Haimovich. Borgwardt was the first to obtain a polynomial expected number of steps - see [Borgwardt 1987]. The method that he analyzes is a special variable-dimension phase I method together with a parametric objective algorithm for phase II. The problem instances are of the form $\max\{c^T x: Ax \leq e\}$ where e is a vector of ones and the vector c and the rows of A are generated independently from a rotationally-symmetric distribution. If A is $m \times n$, the expected number of iterations is $O(m^4 n^{1/(m-1)})$.

Both probability distributions have some rather unsatisfactory features, not typical of real-world instances. The sign-invariant model generates problem instances which are almost always infeasible or unbounded, while Borgwardt's model generates problem instances which all have the origin as an obvious feasible solution. However, the results obtained provide a significant theoretical justification to the good behavior of the simplex method observed in practice. For further details on this work and earlier developments in the subject we refer to [Shamir 1987] [Megiddo 1987] and [Borgwardt 1987].

8. The Ellipsoid Method

There have been many alternatives to the simplex algorithm proposed for solving linear programming problems. In this section and the next, we discuss two such methods that are distinguished from previous proposals by their property of polynomial-time boundedness. Under the assumption that all the data of a linear programming problem are integers, these methods require a number of elementary operations that is bounded by a polynomial function of the length of an encoding of the input. For a linear programming problem in standard form, this length can be estimated by

$$L = \sum_{i=0}^{m} \sum_{j=0}^{n} \lceil \log(|a_{ij}| + 1) + 1 \rceil \qquad (8.1)$$

where $a_{i0} \equiv b_i$ and $a_{0j} \equiv c_j$. An elementary operation is the addition, subtraction, multiplication or comparison of two bits; for the purpose of polynomial algorthims, we can equivalently allow elementary arithmetic operations on real numbers where the number of digits before and after the decimal point is bounded by a polynomial in the length of the input.

The concept of polynomial-time algorithms and the related notion of computational complexity have proved extraordinarily fruitful in the study of combinatorial optimization problems--we refer to Chapters 5 and 6 for more details. In the context of linear programming the significance of polynomial-time boundedness (especially in view of the

notoriously bad practical performance of the ellipsoid method) seems less compelling; we will discuss this further below.

This section is divided into four parts. In 8.1 we introduce the basic idea of the algorithm. The precise statement of the algorithm and its volume-reduction properties are given in 8.2, along with some extensions to the basic method. In 8.3 we give a sketch of the proof that this algorithm requires only polynomial time to solve linear programming problems, and we conclude in 8.4 by discussing its theoretical and computational significance.

8.1. The Idea of the Ellipsoid Algorithm

Let us note first that the ellipsoid method was not devised by Khachian in 1979; it is an outgrowth of the method of central sections of Levin [1965] and Newman [1965] and the method of generalized gradient descent with space dilation of Shor [1970] and was developed by Yudin and Nemirovskii [1976] for convex, not necessarily differentiable, programming. Further remarks on its antecedents can be found in Bland, Goldfarb and Todd [1981]. The contribution of Khachian [1979 and 1980] was to demonstrate that this method could be used to provide a polynomial algorithm for linear programming.

The ellipsoid method can be applied most directly to problems in inequality form. Thus we will assume that we wish to solve

$$\min a_0^T y, \quad \text{subject to} \quad y \in Y, \tag{8.2}$$

where

$$Y = \{y \in \mathbb{R}^m: a_j^T y \leq c_j \quad \text{for} \quad j = 1,2,\ldots,n\}. \tag{8.3}$$

Note that the dual of a problem in standard form is of this type. For the moment, we assume that the components of the vectors $a_j \in R^m$ and the scalars c_j are arbitrary real numbers. We also suppose for simplicity that we only require a <u>feasible</u> point $\bar{y} \in Y$. For the optimization problem, we can proceed in either of two ways. We can consider the feasibility problem for a sequence of polyhedra

$$Y(c_0) = \{y \in \mathbb{R}^m: a_j^T y \leq c_j \quad \text{for} \quad j = 0,1,\ldots,n\}, \tag{8.4}$$

depending on a decreasing sequence $\{c_0\}$ of upper bounds on the optimal value of (8.2). An alternative approach is to combine primal and dual feasibility and equality of objective function values into a large system of inequalities--see section 8.3.

The algorithm to find a point in Y (if one exists) operates by generating a sequence $\{E_k\}$ of ellipsoids with centers $\{y_k\}$. At each step Y is contained in E_k, and either $y_k \in Y$ (in which case the algorithm terminates) or a new ellipsoid E_{k+1} is constructed with $Y \subseteq E_{k+1}$ and the volume of E_{k+1} smaller than that of E_k by some fixed factor Φ depending only on m. See Figure 8.1.

In order to start the method and provide a guarantee of its terminating, we assume for now:

(A1) For some known $y_0 \in R^m$ and $R > 0$, the polyhedron Y is contained in the ball $S(y_0,R) = \{y \in \mathbb{R}^m: \|y-y_0\| \leq R\}$; and

(A2) There is a known $\rho > 0$ such that, if Y is nonempty, it contains a ball $S(y^*,\rho)$ of radius ρ.

Assumption (A1) allows us to choose $E_0 = S(y_0,R)$. If $\text{vol}(E_{k+1}) \leq \phi\ \text{vol}(E_k)$ for all k, then for $K > \lceil m \log(\rho/R)/\log \phi \rceil$, the volume of E_K is smaller than that of a sphere of radius ρ, so that if no feasible y_k has been generated in K iterations, (A2) allows us to conclude that Y is empty.

8.2. Statement of the Algorithm

We will represent each ellipsoid E_k by means of its center y_k and a symmetric positive definite $m \times m$ matrix B_k:

$$E_k = \{y \in \mathbb{R}^m: (y-y_k)^T B_k^{-1} (y-y_k) \leq 1\}. \tag{8.5}$$

Note that, if $B_k = J_k J_k^T$, then $E_k = \{y_k + J_k z: \|z\| \leq 1\}$, the image of the unit ball under the affine transformation $z \to y_k + J_k z$. It follows that the volume of E_k is

$$\text{vol } E_k = (\det B_k)^{1/2}\ \text{vol}(S(0,1)). \tag{8.6}$$

From assumption (A1), we can choose $B_0 = R^2 I$ to ensure that $Y \subseteq E_0$.

At each iteration k, we assume we have E_k as in (8.5) with $Y \subseteq E_k$, and we check whether $y_k \in Y$. If so, we stop; otherwise, we choose j with $a_j^T y_k > c_j$. In this case, we know that Y is contained in

$$\frac{1}{2} E_k \equiv \{y \in E_k : a_j^T y \leq a_j^T y_k\}, \tag{8.7}$$

which is half the ellipsoid E_k cut off by a hyperplane through its center. We now construct the ellipsoid E_{k+1} of minimum volume containing $\frac{1}{2} E_k$. See Figure 8.1(a).

Let $\tau = 1/(m+1)$, $\delta = m^2/(m^2-1)$ and $\sigma = 2/(m+1)$, and set

$$y_{k+1} = y_k - \tau B_k a_j / (a_j^T B_k a_j)^{1/2}, \tag{8.8}$$

$$B_{k+1} = \delta \left[B_k - \sigma \frac{B_k a_j a_j^T B_k}{a_j^T B_k a_j} \right]. \tag{8.9}$$

Theorem 8.1 (see, e.g., Bland, Goldfarb, and Todd [1981]). The ellipsoid E_{k+1}, given as in (8.5) from y_{k+1} and B_{k+1} in (8.8-8.9), is the minimum volume ellipsoid containing $\frac{1}{2} E_k$. We have

$$\frac{\text{vol } E_{k+1}}{\text{vol } E_k} = \left[\frac{m^2}{m^2-1} \right]^{\frac{m-1}{2}} \frac{m}{m+1} < \exp\left[- \frac{1}{2(m+1)} \right]. \tag{8.10}$$

Let us now state the algorithm concisely:

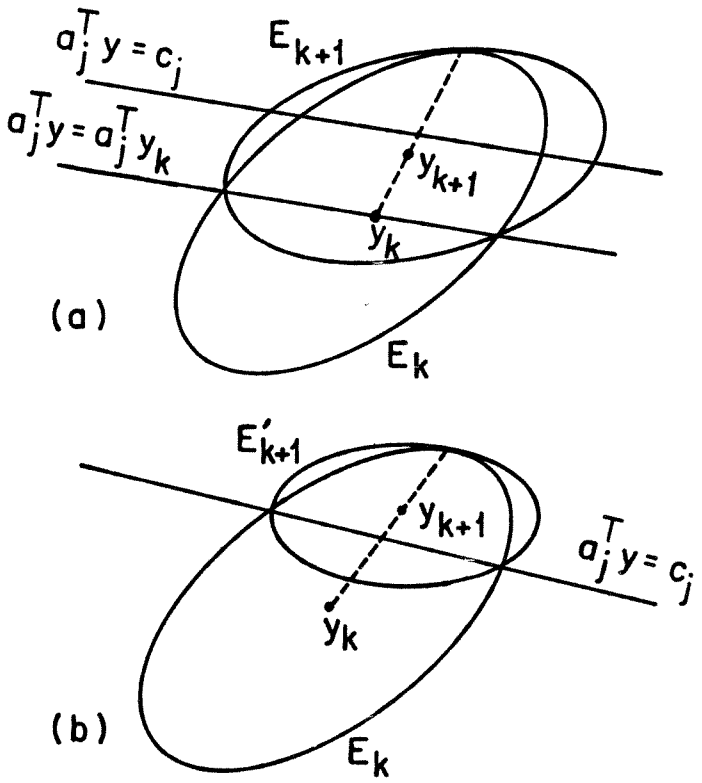


Figure 8.1

Algorithm 8.1.

Input: $m \times n$ matrix $A = [a_1,\ldots,a_n]$, n-vector $c = [c_1,\ldots,c_m]^T$ such that Y defined by (8.3) satisfies (A1)-(A2).

Output: m-vector $y \in Y$ or determination that $Y = \phi$.

Initialization: Set $y_0 = 0$, $B_0 = R^2 I$, $K = \lceil 2m(m+1)\ell n(R/\rho)\rceil + 1$.

For $k = 0,1,\ldots,K-1$ do

Iteration k: If $y_k \in Y$, STOP: result is $y = y_k$.
Otherwise, choose j with $a_j^T y_k > c_j$.
Update y_k and B_k using (8.8)-(8.9).

STOP: result is "$Y = \phi$."

From Theorem 8.1 and the discussion at the end of the last section we can conclude that this algorithm is correct.

In section 8.3 we will sketch how a modification of algorithm 8.1 can be used to provide a polynomial time algorithm for linear programming. To conclude this section we discuss some of the extensions and improvements that have been proposed for this basic method.

The most obvious change, suggested by a number of authors (see [Bland, Goldfarb and Todd 1981]), is to replace the semi-ellipsoid $\frac{1}{2}E_k$ in (8.7) with the smaller set

$$\{y \in E_k: a_j^T y \le c_j\}. \tag{8.11}$$

Thus we use so-called "deep" cuts. See Figure 8.1(b). If we define

$$\alpha = (a_j^T y_k - c_j)/(a_j^T B_k a_j)^{1/2}, \tag{8.12}$$

then $\alpha \ge 0$ by our choice of j, and the set (8.11) is nonempty iff $\alpha \le 1$. For $\alpha \in [0,1]$, the minimum volume ellipsoid E_{k+1} containing (8.11) is given as in (8.5), with y_{k+1} and B_{k+1} defined by (8.8)-(8.9) with revised parameters

$$\tau = \frac{1+m\alpha}{m+1}, \quad \delta = (1-\alpha^2)\frac{m^2}{m^2-1}, \quad \sigma = \frac{2(1+m\alpha)}{(1+\alpha)(m+1)}. \tag{8.13}$$

The corresponding ratio of the volumes of successive ellipsoids is

$$\frac{\text{vol } E_{k+1}}{\text{vol } E_k} = \left[\frac{m^2(1-\alpha^2)}{m^2-1}\right]^{\frac{m-1}{2}} \frac{m(1-\alpha)}{m+1}. \tag{8.14}$$

Further reductions in the volume ratio can be achieved using surrogate cuts and parallel cuts (see Bland, Goldfarb, and Todd [1981]).

Finally, we note that a completely different way to represent the ellipsoid E_k has been proposed by Burrell and Todd [1985]. Since (A1) provides a bound on each component of y in Y, it is easy to find $\ell^0 \in \mathbb{R}^n$ such that

$$Y = \{y \in \mathbb{R}^m: \ell^0 \leq A^T y \leq c\}.$$

At iteration k, suppose we have a vector $\ell^k = (\ell^k_j)$ of lower bounds on $A^T y$ for $y \in Y$, so that

$$Y = \{y \in \mathbb{R}^m: \ell^k \leq A^T y \leq c\}. \tag{8.15}$$

Suppose we also have a diagonal $n \times n$ matrix D_k with nonnegative diagonal entries, such that AD_kA^T is positive definite. It then follows immediately from (8.15) that Y is contained in the ellipsoid E_k given by

$$E_k = \{y \in \mathbb{R}^m: (A^T y - \ell^k)^T D_k (A^T y - c) \leq 0\}. \tag{8.16}$$

This can also be written in the form (8.5), with B_k a scalar multiple of $(AD_kA^T)^{-1}$, and center y_k the solution of the linear equations

$$(AD_kA^T)y = AD_k\, r^k \tag{8.17}$$

with $r^k = (\ell^k + c)/2$.

If y_k is in Y, the algorithm terminates. Otherwise, we choose an index j with $a_j^T y_k > c_j$. If ℓ^k_j is greater than $\lambda_j = \min\{a_j^T y: y \in E_k\}$ then $\ell^{k+1} = \ell^k$; otherwise we can update just

this component of ℓ^k so that ℓ_j^{k+1} is at least λ_j. Indeed, Burrell and Todd show how to obtain the new jth component of ℓ^{k+1} as the value of a dual feasible solution of $\min\{a_j^T y: A^T y \leq c\}$, and how this dual solution can be obtained using the representation (8.16). Having updated ℓ^k to ℓ^{k+1}, we increase the jth diagonal entry of D_k suitably to get D_{k+1}, and then the resulting ellipsoid E_{k+1} is exactly the minimum volume ellipsoid containing

$$\{y \in E_k: \ell_j^{k+1} \leq a_j^T y \leq c_j\}.$$

(In fact, this is a slight simplification of the true situation; if E_k already depends on ℓ_j^k - the jth diagonal entry of D_k is nonzero - and ℓ_j^k is increased to ℓ_j^{k+1}, then we must compensate for the effect of the changed lower bound on E_k before updating the lower bounds and the ellipsoid.)

There are many advantages to the representation (8.16). First, Y is contained in E_k regardless of round-off errors as long as $D_k \geq 0$ and ℓ^k is a vector of valid lower bounds. Second, these lower bounds are certified by maintaining dual solutions for the linear programming problems $\min\{a_j^T y: A^T y \leq c\}$. Third, if in fact Y is empty, and α in (8.12) exceeds one at some iteration k, then the representation can be used to produce a vector x with $Ax = 0$ and $c^T x < 0$, demonstrating $Y = \emptyset$ by Farkas' lemma. Details may be found in [Burrell and Todd 1985]. Here we merely wish to point out that the equations (8.17) for determining the center y_k of the ellipsoid E_k

are the normal equations for the (weighted) least-squares problem

$$\min_{y \in \mathbb{R}^m} \| D_k^{1/2} (A^T y - r^k) \|, \tag{8.18}$$

where the weighting matrix $D_k^{1/2}$ is a diagonal matrix with diagonal entries equal to the square roots of those of D_k. Moreover, as we proceed from one iteration to the next, the linear system (8.17) changes in just two respects: the jth component of ℓ^k (and hence of r^k) may increase, and the jth diagonal of D_k increases, where j indexes a violated constraint at y_k.

8.3. Polynomial-time Solvability

First we suppose we wish to find a feasible point in

$$Y = \{y \in \mathbb{R}^m : A^T y \leq c\} \tag{8.19}$$

in time polynomial in the length of input of the system,

$$L = \sum_{i=0}^{m} \sum_{j=1}^{n} (\log(|a_{ij}| + 1) + 1) + 1 \qquad (8.20)$$

Here A is an $m \times n$ integer matrix, c is an integer n-vector, and $a_{0j} \equiv c_j$.

We wish to use the argument in section 7.1, which relied on assumptions (A1) and (A2). Unfortunately, these assumptions may fail to hold, but the argument can be salvaged using perturbations. Let $c' = c + 2^{-L}e$ where e is an n-vector of ones, and

$$Y' = \{y \in \mathbb{R}^m : A^T y \leq c'\}. \qquad (8.21)$$

Then one can prove (see, e.g., Khachian [1980] or Gacs and Lovasz [1981]):

Lemma 8.1. If Y is nonempty, then so is $Y \cap S(0,2^L)$; moreover, there is some y_* with $S(y_*,2^{-2L}) \subseteq Y' \cap S(0,2^L)$. If Y is empty, so is Y'.

Thus the argument of section 8.1 can be applied to $Y' \cap S(0,2^L)$ with $R = 2^L$, $\rho = 2^{-2L}$. Moreover, the algorithm is precisely algorithm 7.1 with the stopping rule $y_k \in Y'$ replacing $y_k \in Y$. Lemma 8.1 shows that Y' being nonempty implies the same for Y. Thus we can determine feasibility of the system $A^T y \leq c$ in $O(m^2 \ell n(R/\rho)) = O(m^2 L)$ steps, assuming exact arithmetic. Finally, it is not hard to use an algorithm for determining feasibility recursively to actually obtain a

solution for a feasible system of inequalities. Alternatively, one can use a rounding method (see [Grötschel, Lovasz, and Schrijver 1981]) to obtain a feasible point in Y from one in Y'.

The simplest way to solve the optimization problem

$$\min[a_0^T y: y \in Y\} \tag{8.22}$$

with Y given as in (8.19) is to recast it as the feasibility problem:

$$\begin{aligned} A^T y &\leq c \\ Ax &\leq -a_0 \\ -Ax &\leq a_0 \\ -x &\leq 0 \\ c^T x + a_0^T y &\leq 0. \end{aligned} \tag{8.23}$$

Then the algorithm as described can be applied; the length of input of (8.23) is polynomial in the length of input

$$L = \sum_{i=0}^{m} \sum_{j=0}^{n} (\log(|a_{ij}| + 1) + 1) \tag{8.24}$$

of (8.22), where $a_{0j} \equiv c_j$ and $a_0 = (a_{i0})$.

Alternatively, one can use a sliding-objective method (see [Bland, Goldfarb, and Todd 1981] and [Grötschel, Lovasz and Schrijver 1981]), as follows. First, a feasible point $\bar{y}$ to Y (or Y') is found;

then the constraint $a_0^T y \leq c_0 = a_0^T \bar{y}$ is added and the iterations continued, starting by choosing the 0th constraint as "violated". Thus at the kth iteration we work with the polyhedron

$$\{y \in \mathbb{R}^m : a_i^T y \leq c_i \quad \text{for} \quad i = 0,1,\ldots,n\}$$

where $c_0 = \min\{a_0^T y_\ell : A^T y_\ell \leq c,\ 0 \leq \ell \leq k\}$ is revised and a_j chosen as a_0 whenever a feasible iterate is generated. The polynomiality of this version can be established by considering the volumes of sections of the cone with vertex an optimal solution of (8.22) and base a small ball of radius 2^{-2L} in Y or Y'.

To conclude the section we note that the analysis above assumes that the iterates y_k and B_k are computed exactly. For a polynomial algorithm, however, we can only use finite precision. Because the resulting rounding may cause the fundamental inclusion $Y \subseteq E_k$ to fail, we must slightly enlarge the ellipsoid E_k by increasing δ in (8.9) slightly. See, e.g., [Khachian 1980] or [Grotschel, Lovasz and Schrijver 1981]. Then y_k and B_k can be updated and rounded to $O(L)$ bits before and after the decimal point. In so doing, the number of iterations is increased only by a constant factor. Hence we finally have:

Theorem 8.2. The ellipsoid method can be adapted to find a feasible solution to (8.19) or an optimal solution to (8.22), if one exists, in time bounded by a polynomial in the length of its input, given by (8.20) or (8.24) respectively.

The number of bit operations to determine feasibility of (8.19) is bounded (see [Khachian 1980]) by

$$O(n^4L^2). \tag{8.25}$$

Note that this contrasts with the earlier estimate $O(n^6L^2)$ given in Khachian [1979] and quoted by Karmarkar [1984a,b].

8.4. Theoretical and Computational Significance

The discovery of a polynomial-time algorithm for linear programming resolved an important question concerning the computational complexity of this problem. For many years, linear programming had been thought to be much easier than the so-called NP-complete problems (see Chapter 5), which include for example the notorious traveling salesman and graph coloring problems; yet no polynomial algorithm was known. Khachian's result thus settled a significant theoretical question. As we shall see, its theoretical implications are much broader than this result indicates.

For practical computation, however, the ellipsoid method seems at present to be of little use. Even with considerable enhancements, as described in [Bland, Goldfarb and Todd 1981] and outlined above, convergence appears to be very slow on problems of even moderate dimension; the volume reduction achieved at each iteration is not much better than that given by (8.10). While the worst-case complexity of

several variants of the simplex method is exponential, its practical performance is far better than that of the ellipsoid method. We must point out, however, that good results have been reported by Ecker and Kupferschmidt [1985] for certain nonlinear programming problems of small to medium dimension. In addition, Shor [1985] quotes successful experience on nonsmooth optimization problems using various space dilation methods, which differ from the ellipsoid method in using different constants τ, δ, σ than those in (8.13)--see Chapter 7.

Finally we return to the broader theoretical significance of the ellipsoid method. This stems from the way in which the algorithm uses the constraints of a linear programming problem. Indeed, the discussion above shows that it is enough to recognize whether a given $\bar{y}$ is feasible, and, if not, to generate some constraint $a_j^T y \leq c_j$ that is violated by $\bar{y}$. Hence no a priori listing of the constraints is necessary as long as they can be obtained in this way as needed. This permits the application of the ellipsoid method to a host of combinatorial optimization problems where the constraints are only known implicitly and may be exponential (in the dimension of the problem) in number. This potentiality was noted by Grötschel, Lovasz and Schrijver [1981], Karp and Papadimitriou [1980] and Padberg and Rao [1980], and is developed extensively in Grötschel, Lovasz and Schrijver [1986]. An elementary discussion is contained in Chapter 5.

In summary, the ellipsoid method is of little significance for practical computation in linear programming, but its theoretical significance in linear, nonlinear and combinatorial optimization is profound.

9. Karmarkar's Projective Scaling Algorithm

In 1984, a new polynomial-time algorithm for linear programming was developed by N.K. Karmarkar of AT&T Bell Laboratories [Karmarkar 1984a,b]. In contrast to the ellipsoid method, this new algorithm seems to have the potential to rival the simplex method for practical computation. It is still too early to make a definitive comparison, and claims and counter-claims on the efficiency of the new method have been put forward at a sequence of conferences. At the moment, it can certainly be asserted that Karmarkar's algorithm has considerable promise, and some outstanding results for very large sparse problems have been obtained (e.g. [Karmarkar and Sinha 1985] [Karmarkar and Ramakrishnan 1988]); on the other hand, codes developed outside Bell Laboratories ([Adler, Karmarkar, Resende and Veiga 1986], [Gill, Murray, Saunders, Tomlin and Wright 1986], [McShane, Monma and Shanno 1988] and [Monma and Morton 1987]) report times from five or more times faster to five times slower than the widely-used MINOS code [Murtagh and Saunders 1977] which implements the simplex method when the objective function is linear.

In this section we will describe the new algorithm and the ideas (coming from nonlinear programming and projective geometry) on which it is based. Our discussion is divided into seven parts: we describe the idea of the algorithm in 9.1, state it precisely in 9.2, and sketch the proof of polynomiality in 9.3. Section 9.4 reviews extensions and variants, while section 9.5 describes related approaches, including the so-called "affine scaling" method, which turns out to have been proposed by the Soviet mathematician I.I. Dikin as long ago as 1967,

and the recent path-following methods. Section 9.6 discusses implementation of the algorithms, and we conclude in 9.7 by commenting on their theoretical and computational significance. We remark here that these comments stand in sharp contrast to our conclusions for the ellipsoid method; while for computation Karmarkar's algorithm has potentially very high significance, its theoretical consequences appear far more limited than those of the earlier method.

9.1. The Idea of the Projective Scaling Algorithm

As with the ellipsoid method, Karmarkar's algorithm has a preferred form for linear programming problems; in this case the canonical form is

$$\begin{aligned} \min\ & c^T x \\ & A x = 0 \\ & x \in \Delta \end{aligned} \tag{9.1}$$

where Δ is the standard simplex in $\mathbb{R}^n$,

$$\Delta = \{x \in \mathbb{R}^n : e^T x = 1, \quad x \geq 0\} \tag{9.2}$$

and e is the n-vector of ones. We also assume that

(Ai) A is an $m \times n$ matrix of rank m;

(Aii) $Ae = 0$, so that $x^0 = e/n$ is feasible; and

(Aiii) the optimal value v of (9.1) is zero.

We will see later how to convert a general linear programming problem into one in this form, satisfying the assumptions above.

Note that the dual of problem (9.1) is to find $y \in \mathbb{R}^m$ and $z \in \mathbb{R}$ to

$$\begin{aligned} \max \quad & z \\ & A^T y + ze \leq c. \end{aligned} \tag{9.3}$$

There are two interesting properties of this dual problem. First, it is clearly feasible; indeed, for any given $\bar{y} \in \mathbb{R}^m$ it is easy to choose z to make $(\bar{y},z)$ feasible--the best such z is $\min_j (c-A^T\bar{y})_j$. Second, our assumption that the optimal value of (9.1) and hence of (9.3) is zero means that we are really seeking $\bar{y} \in \mathbb{R}^m$ with $A^T\bar{y} \leq c$, and this feasibility problem is exactly what the ellipsoid method of section 7 is designed for.

It will be convenient to call a feasible solution x to (9.1) strictly feasible if each component of x is positive. Karmarkar's algorithm generates a sequence $\{x^k\}$ of strictly feasible solutions, starting with $x^0 = e/n$, that satisfy

$$c^T x^k \leq \exp(-k/5n) c^T x^0 \tag{9.4}$$

for all k. This performance guarantee suffices to provide a polynomial algorithm as follows. If the length of input of the integer data A, c is

$$L = \sum_{i=0}^{m} \sum_{j=1}^{n} \lceil \log(|a_{ij}| + 1) \rceil + 1, \qquad (9.5)$$

where $a_{0j} \equiv c_j$, then from a strictly feasible solution x^k with $c^T x^k \leq 2^{-2L} c^T x^0$ an exact solution can be found by moving to a vertex with objective function value no greater. Hence $O(nL)$ iterations are enough to obtain an optimal solution. (In practice, several variants of Karmarkar's algorithm appear to require only 20-50 iterations even for very large problems.)

The basic idea of an iteration is as follows. Suppose we are given a strictly feasible solution $x^k \in \Delta$. We then make a <u>projective transformation</u> (or projective scaling) of the simplex Δ that carries Δ into itself and x^k into the center of the simplex e/n. Indeed, the vertices of Δ are fixed under this transformation, and this would force a linear or affine transformation to be the identity; the extra freedom of a projective transformation allows us to deform the interior of Δ to map x^k into e/n. The advantage of such a transformation is that the current iterate in the transformed space is far from all the inequality constraints $x \geq 0$. While forcing a certain subset of these to be equalities forms the foundation of the simplex method, the projective scaling method seeks to avoid the combinatorial complexities of switching among an exponential number of such active sets.

Following the transformation, we ignore the inequality constraints and make a move in the direction of the negative of the gradient of the

transformed objective function, projected onto the transformed equality constraints. A suitable step size in this direction leads to a new strictly feasible point in the transformed space. An application of the inverse projective transformation then yields the next iterate x^{k+1}.

Obtaining the crucial inequality (9.4) requires a more complicated analysis, since the linear objective function of (9.1) becomes nonlinear under the projective transformation. Karmarkar introduces the potential function

$$f(x) = n \ell n\, c^T x - \sum_j \ell n\, x_j = \sum_j \ell n(c^T x / x_j) \qquad (9.6)$$

to measure progress towards the solution. This important function is invariant under the projective transformations used, and can be reduced by a constant at each iteration. This reduction then implies (9.4)--details are given later.

9.2 Statement of the Algorithm

First we describe the projective transformation. We suppose given a strictly feasible solution $x^k \in \Delta$. To simplify the notation, we suppress the index k and write $\overline{x}$ for x^k.

Define the diagonal matrix $\overline{X}$ to have as its diagonal entries the components of $\overline{x}$, so that $\overline{X}e = \overline{x}$. Then the projective transformation is defined by

$$x \to \hat{x} = \frac{\overline{X}^{-1}x}{e^T\overline{X}^{-1}x} \tag{9.7a}$$

with inverse

$$\hat{x} \to x = \frac{\overline{X}\hat{x}}{e^T\overline{X}\hat{x}} . \tag{9.7b}$$

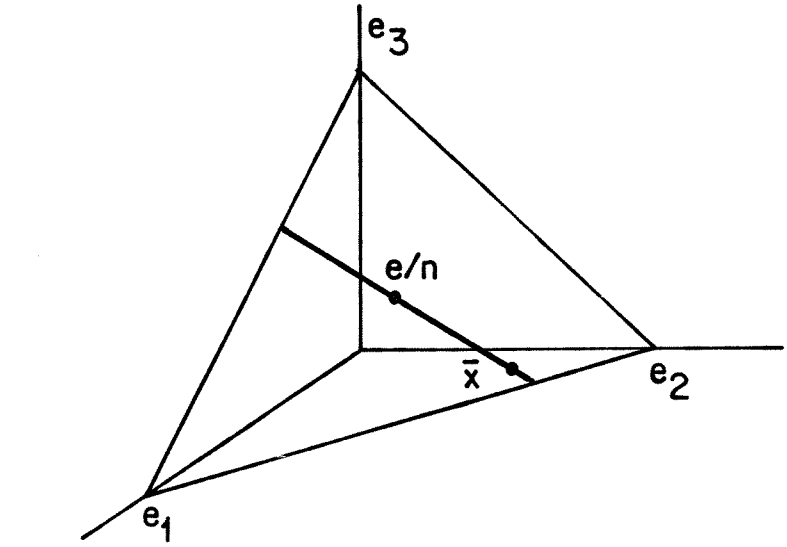
Note that $\overline{x}$ is indeed transformed into $\hat{x} = e/n$.

We illustrate this transformation in the case $n = 3$, $m = 1$, with $A = [-7,2,5]$. Then the feasible region is the line segment joining $(2/9,7/9,0)^T$ and $(5/12,0,7/12)^T$, and $e/n = (1/3,1/3,1/3)^T$ is indeed feasible. We let $\overline{x} = (1/4,2/3,1/12)^T$. See Figure 9.1(a).

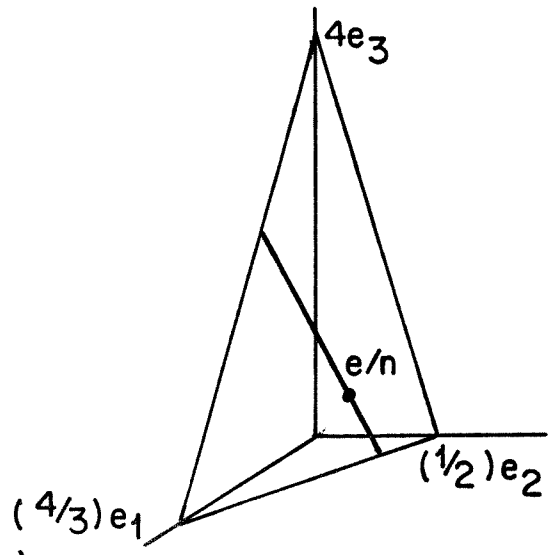
To aid in understanding the projective transformation, we carry it out in two steps. The first is just a diagonal scaling taking x to $\tilde{x} = n^{-1}\overline{X}^{-1}x$. Thus $\overline{x}$ is taken into e/n, but the simplex Δ is distorted. In our example, $\overline{x}$ is transformed to $(1/3,1/3,1/3)^T$, and Δ is taken into the triangle with vertices $(4/3)e_1$, $(1/2)e_2$, and $4e_3$, where e_1, e_2 and e_3 are the unit vectors in $\mathbb{R}^3$. The feasible region is transformed into the intersection of this distorted triangle with the null-space of $\hat{A} = A\overline{X} = [-7/4,4/3,5/12]$, which is the line segment joining $(8/27,7/18,0)^T$ and $(5/9,0,7/3)^T$. See Figure 9.1(b).

Finally we transform the distorted simplex into Δ by performing a radial projection taking $\tilde{x}$ to $\hat{x} = \tilde{x}/e^T\tilde{x}$. Both e/n and the subspace $\{\tilde{x}: \hat{A}\tilde{x} = 0\}$ are invariant under this transformation, but e/n becomes the center of the new undistorted simplex. In our example, the feasible region becomes the line segment joining $(16/37,21/37,0)^T$ and $(5/26,0,21/26)^T$. See Figure 9.1(c).

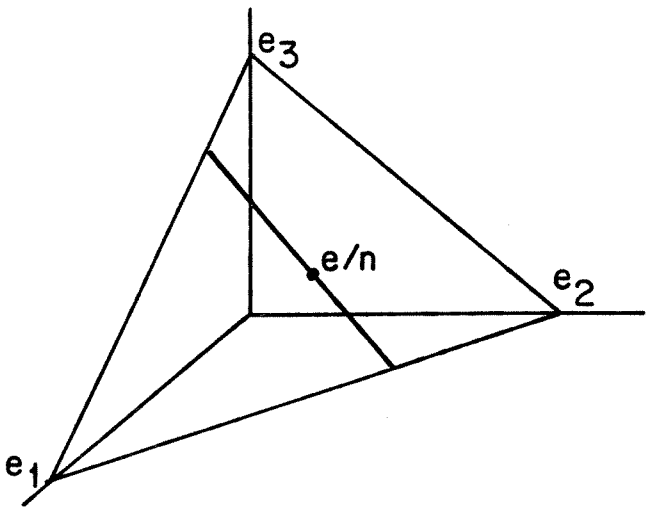
Now we return to the general case. Using (9.7b) to substitute for



(a)



(b)



(c)

Figure 9.1

x in (9.1), we obtain the equivalent problem:

$$\min \frac{c^T \overline{X} \hat{x}}{e^T \overline{X} \hat{x}}$$

$$A\overline{X}\hat{x} = 0 \qquad (9.8)$$

$$\hat{x} \in \Delta$$

with a linear fractional objective function. However, assumption (Aiii) implies that it is sufficient to obtain a solution with objective value zero, and for this it is enough if the numerator is zero. Hence, defining

$$\hat{A} = A\overline{X}, \quad \hat{c} = \overline{X}c, \qquad (9.9)$$

we are led to

$$\min \hat{c}^T \hat{x}$$

$$\hat{A}\hat{x} = 0 \qquad (9.10)$$

$$\hat{x} \in \Delta.$$

This has the identical form to our original problem (9.1). Moreover, in this transformed space, our current solution is in the center of Δ, $\hat{x} = e/n$.

In order to improve this solution, we replace the constraint $\hat{x} \in \Delta$ with its n inequalities by a stronger but smoother constraint. Let

$$S'(e/n, \rho) = \{x \in \mathbb{R}^n : e^T x = 1, \quad \|x - e/n\| \leq \rho\} \qquad (9.11)$$

denote the ball of radius ρ around e/n in the affine space

$\{x \in \mathbb{R}^n: e^T x = 1\}$. It is easy to check that

$$S'(e/n, r) \subseteq \Delta \subseteq S'(e/n, R) \tag{9.12}$$

where

$$r = (n(n-1))^{-1/2}, \quad R = ((n-1)/n)^{1/2}. \tag{9.13}$$

Then, for any $\rho > 0$, the solution to

$$\begin{aligned} \min\ & \hat{c}^T \hat{x} \\ & \hat{A}\,\hat{x} = 0 \\ & \hat{x} \in S'(e/n, \rho) \end{aligned} \tag{9.14}$$

is given by

$$\hat{x}(\rho) = e/n + \rho \hat{d}/\|\hat{d}\|, \tag{9.15}$$

where $\hat{d}$ is the negative of the orthogonal projection of $\hat{c}$ onto the null space of

$$B = \begin{bmatrix} \hat{A} \\ e^T \end{bmatrix}. \tag{9.16}$$

Thus

$$\hat{d} = -(I - B^T(BB^T)^{-1}B)\hat{c}. \tag{9.17}$$

The algorithm chooses $\rho = \alpha/n < \alpha r$, where $0 < \alpha < 1$ is chosen so

that $\hat{x}(\rho)$ is feasible and so that the potential function defined below is sufficiently reduced. Setting $\alpha = 1/3$ at each iteration suffices, although other choices are preferable in practice. We thus have:

Algorithm 9.1. (projective scaling).

Input: m×n matrix A, n vector c such that (Ai)-(Aiii) are satisfied.

Output: n-vector x feasible in (9.1) with $c^T x \leq 2^{-q} c^T e/n$.

Initialization: Set $x^0 = e/n$.

For $k = 0,1,\ldots,5nq$ do

Iteration k: If $c^T x^k \leq 2^{-q} c^T e/n$, STOP: result is $x = x^k$.

Otherwise, let $X_k = \text{diag}(x^k)$.

Define $\hat{c} = X_k c$, $\hat{A} = AX_k$.

Compute $\hat{d}$ from (9.16) and (9.17), and hence $\|\hat{d}\|$

and $\hat{x} = e/n + \alpha_k \hat{d}/n\|\hat{d}\|$, where $\alpha_k = 1/3$.

Set $x^{k+1} = X_k \hat{x}/e^T X_k \hat{x}$.

In the next subsection we show why this algorithm will indeed stop in the required 5nq iterations.

9.3 Polynomial-time Solvability

In this section we briefly describe how algorithm 9.1 can be used to solve arbitrary linear programming problems with integer data in

polynomial time. We first show that the algorithm does indeed provide the desired accuracy.

Let us recall the potential function

$$f(x) \equiv f(x;c) \equiv \sum_j \ell n(c^T x/x_j) \qquad (9.18)$$

where we have explicitly shown the dependence on the cost vector c. We write

$$\hat{f}(\hat{x}) \equiv f(\hat{x};\hat{c}).$$

The first result demonstrates the value of this function in dealing with the projective transformation used in the algorithm:

Lemma 9.1. If x and $\hat{x}$ are related as in (9.7), then

$$\hat{f}(\hat{x}) = f(x) + \ell n \det \overline{X}. \qquad (9.19)$$

The proof is straightforward from the definitions. Thus if we can reduce $\hat{f}$ by a constant from its value at e/n we can reduce f by the same amount using the corresponding step in the untransformed space.

Next we show

Lemma 9.2. Let $\hat{x} = e/n + \alpha\hat{d}/n\|\hat{d}\|$, where $\hat{d} = -(I-B^T(BB^T)^{-1}B)\hat{c}$. Then

$$n \ell n\, \hat{c}^T\hat{x} \leq n \ell n(\hat{c}^T e/n) - \alpha. \qquad (9.20)$$

Proof. We note that $\hat{c}^T\hat{x}(\rho) = \hat{c}^T e/n - \rho\|\hat{d}\|$, since $\hat{c}^T\hat{d} = -\hat{d}^T\hat{d} = -\|\hat{d}\|^2$. Also, with $\rho = R$, problem (9.14) is a relaxation of (9.10), so that its optimal value is nonpositive. Hence $\hat{c}^T\hat{x}(R) \leq 0$ by (Aiii), implying $-\|\hat{d}\| \leq -(\frac{1}{R})\hat{c}^T e/n$. Therefore

$$\hat{c}^T\hat{x} = \hat{c}^T e/n - (\alpha/n)\|\hat{d}\| \leq (1 - \alpha/nR)\hat{c}^T e/n < (1-\alpha/n)\hat{c}^T e/n$$

since $R < 1$. Taking logarithms and using $\ell n(1 - \alpha/n) \leq -\alpha/n$ gives the result.

The following lemma bounds the barrier function part $-\Sigma_j \ell n\, \hat{x}_j$, in the potention function. A slightly sharper version is quoted in [Karmarkar 1984b], which can be proved using power series.

Lemma 9.3. If $\hat{x} \in S'(e/n, \alpha/n)$, then

$$-\Sigma_j \ell n\, \hat{x}_j \leq -\Sigma_j \ell n(1/n) + \alpha^2/2(1-\alpha)^2. \qquad (9.21)$$

Proof. For each j, using the Taylor series expansion of $\ell n(1+(n\hat{x}_j-1))$ there is a ξ_j between 1 and $n\hat{x}_j$ such that

$$\ell n(n\hat{x}_j) = \ell n\, 1 + (n\hat{x}_j-1) - \frac{1}{2\xi_j^2}(n\hat{x}_j-1)^2.$$

Now $\hat{x} \in S'(e/n,\alpha/n)$ implies that $n\hat{x}_j \geq 1 - \alpha$, so $\xi_j \geq 1 - \alpha$ and hence

$$\ell n(n\hat{x}_j) \geq n\hat{x}_j - 1 - \frac{1}{2(1-\alpha)^2}(n\hat{x}_j - 1)^2.$$

Thus, since $\Sigma_j\, n\hat{x}_j = n$ and $\Sigma_j(n\hat{x}_j - 1)^2 = \|n\hat{x} - e\|^2 = n^2\|\hat{x} - e/n\|^2 \leq \alpha^2$, we obtain

$$\Sigma_j\, \ell n\,(n\hat{x}_j) \geq -\alpha^2/2(1-\alpha)^2,$$

from which (9.21) is immediate.

From Lemmas 9.1 to 9.3 we conclude

Theorem 9.1. If the sequence $\{x^k\}$ is defined by algorithm 9.1 then

$$f(x^k) \leq f(x^0) - k/5. \qquad (9.22)$$

Proof. By lemmas 9.2 and 9.3, at each iteration we have

$$\hat{f}(\hat{x}) \leq \hat{f}(e/n) - \alpha_k + \alpha_k^2/2(1-\alpha_k)^2;$$

with $\alpha_k = 1/3$ we find

$$\hat{f}(\hat{x}) \leq \hat{f}(e/n) - 1/5.$$

Now using lemma 9.1 gives

$$f(x^{k+1}) \leq f(x^k) - 1/5$$

whence (9.22) follows.

Corollary 9.1. With $\{x^k\}$ as above, (9.4) holds:

$$c^T x^k \leq \exp(-k/5n) c^T x^0.$$

Proof. From (9.22),

$$n \ell n\, c^T x^k - \sum_j \ell n\, x_j^k \leq n \ell n\, c^T x^0 - \sum_j \ell n\, x_j^0 - k/5.$$

But $\Sigma_j \ell n\, x_j$, for $x \in \Delta$, is maximized by $x = x^0$; hence

$$n \ell n\, c^T x^k \leq n \ell n\, c^T x^0 - k/5,$$

and (9.4) follows by dividing by n and exponentiating.

By the remarks earlier in the section we now have a polynomial-time algorithm for problems of the form (9.1).

To handle a general linear programming problem, we first convert it to a feasibility system

$$Mz \leq u \qquad (9.23)$$

as in section 8.3, taking care to bound the primal and dual variables so that there is no non-trivial solution to $Mz \leq 0$. Then consider the

linear programming problem

$$\begin{aligned} \min \quad & \lambda \\ & Mz + Is - ut + (u-Me-e)\lambda = 0 \\ & e^T z + e^T s + t + \lambda = 1 \\ & z,s,t,\lambda \geq 0 \end{aligned} \tag{9.24}$$

(where e represents a vector of ones of appropriate dimension throughout). If $Mz \leq u$, then a scaling of $(z, u-Mz, 1, 0)$ is an optimal solution to this problem with value 0. Conversely, given such an optimal solution, the precaution taken above ensures that $t > 0$, and hence by scaling we find a feasible solution to (9.23). If the algorithm fails to achieve the promised reduction in the potential function at any iteration, then (9.23) is infeasible, and we can proceed in a similar way to determine the feasibility of the primal and dual constraints.

The complexity of each iteration is dominated by the work necessary to compute $\hat{d}$. For this, we must solve $BB^T w = B\hat{c}$. Using the fact that $A\bar{X}e = A\bar{x} = 0$, and writing $w^T = (\bar{y}^T, \tilde{z})$, we find that $\bar{y}$ and $\tilde{z}$ satisfy

$$(A\bar{X}^2A^T)\bar{y} = A\bar{X}^2c, \tag{9.25}$$

$$(e^Te)\tilde{z} = c^T\bar{x}. \tag{9.26}$$

Hence $\tilde{z} = c^T\bar{x}/n$. Note that (9.25) are the normal equations for the

weighted least-squares problem

$$\min_{\bar{y}\in\mathbb{R}^m} \|\bar{X}(A^T\bar{y}-c)\|; \tag{9.27}$$

compare with (8.17)-(8.18). Having obtained $\bar{y}$, we calculate $\hat{d}$ from

$$\hat{d} = -\bar{X}(c - A^T\bar{y}) + (c^T\bar{x}/n)e. \tag{9.28}$$

Finally, all computations can be carried out with $O(L)$ precision, so that each iteration requires about $O(n^3L)$ bit operations. Since at most $O(nL)$ iterations are required to obtain the necessary accuracy in the objective function, we have

Theorem 9.2. Karmarkar's algorithm can be adapted to solve a general linear programming problem with m constraints and p variables in about $O(n^4L^2)$ bit operations, where $n = m+p$.

We note that the complexity is of the same order as that for the ellipsoid method, given in (8.25). Karmarkar described a modification of his algorithm that removed a factor $n^{1/2}$; we will comment on this in the next subsection, which describes extensions and further research related to the projective method.

9.4 Extensions and Variants

We first point out that the analysis of the decrease of the potential function given in lemmas 9.2 and 9.3 can be improved, as has

been shown by Blair [1986], Padberg [1986] and others. This leads to a best value for α_k of $n(n-1)r/(2n-3)$ in algorithm 9.1 and a resulting decrease in f of $(1-\ell n\ 2) \approx \cdot 3$ at each iteration. While this is of interest in the theoretical analysis, in practice it is advisable to perform a line search in the direction $\hat{d}$; typically far larger reductions in the potential function can be achieved.

One of the severest limitations of the canonical form (9.1) is the requirement (Aiii) that its optimal value be zero. Of course, if the optimal value is known to be z^*, then replacing c by $c - z^*e$ gives a problem with the same optimal solution but value 0. In Karmarkar [1984a] this idea was developed with z^* an estimate of the optimal value, bounds on which were updated as the algorithm progressed. More satisfactory solutions were proposed in Todd and Burrell [1986] and Anstreicher [1986] where lower bounds $\underline{z}$ on z^* based on duality and geometry, respectively, were used. We now show how the duality bounds, which are the stronger of the two, are derived. Note that equations (9.25)-(9.26) with c replaced by $c-ze$ can be written

$$(A\overline{X}^2A^T)\overline{y} = A\overline{X}^2(c-ze) \qquad (9.29)$$

$$(e^Te)\tilde{z} = (c-ze)^T\overline{x} \qquad (9.30)$$

Let us write $\overline{y} = \overline{y}(z)$ to denote the dependence on z. Corresponding to $\overline{y}$ is the value

$$\bar{z} = \bar{z}(z) = \min_j (c - A^T \bar{y}(z))_j \qquad (9.31)$$

so that $(\bar{y},\bar{z})$ is feasible in (9.3) and $\bar{z}$ is a lower bound on v, the value of (9.1).

Todd and Burrell show that if $\bar{z}(\underline{z}) \leq \underline{z}$ then, even though $\underline{z}$ may be less than v, the potential function

$$f(\cdot;\ c-\underline{z}e) \qquad (9.32)$$

is decreased by a constant amount. If $\bar{z}(\underline{z}) > \underline{z}$, so that a better lower bound is found, a search is made for an improved bound $\underline{z}'$ with $\bar{z}(\underline{z}') = \underline{z}'$; such a value can be found by a minimum ratio test. Then $\underline{z}'$ replaces $\underline{z}$, and again the potential function (9.32) can be reduced by a constant amount. In this way an algorithm is developed for problems in the form (9.1) satisfying (Ai) and (Aii) but not necessarily (Aiii); this algorithm generates strictly feasible primal solutions x^k and feasible dual solutions (y^k,z^k) which satisfy

$$(c^T x^k - z^k) \leq \exp(-k/5n)(c^T x^0 - z^0), \qquad (9.33)$$

so that the duality gap converges linearly to zero.

Several authors (Anstreicher [1986], Gay [1987], Gonzaga [1985], Jensen and Steger, see Steger [1985], and Ye and Kojima [1987]) have developed an extension that permits a simple treatment of standard form

problems. Suppose we have a strictly feasible solution $\bar{x}$ to

$$\begin{aligned} \min\ & c^T x \\ & A x = 0 \\ & g^T x = 1 \\ & x \geq 0 \end{aligned} \tag{9.34}$$

where $g \geq 0$. Then a scalar multiple of $\bar{x}$ is feasible in

$$\begin{aligned} \min\ & (c-zg)^T x \\ & A x = 0 \\ & e^T x = 1 \\ & x \geq 0; \end{aligned} \tag{9.35}$$

moreover, if z is the optimal value of (9.34), then a scalar multiple of its optimal solution is optimal in (9.35). Thus we can apply the ideas above to (9.35) and hence reduce $f(\cdot; c-zg)$ for a suitable z. At each iteration, z is a lower bound on the optimal value of (9.34) corresponding to a dual feasible solution, and these bounds are updated when possible. In this way, a polynomial algorithm can be derived for (9.34) assuming its feasible region is bounded. Details may be found in the cited papers. Note that the standard form problem

$$\begin{aligned} \min\ & c^T x \\ & A x = b \\ & x \geq 0 \end{aligned}$$

can be rewritten as

$$\begin{aligned} \min\ & (c^T, 0)\begin{pmatrix} x \\ \xi \end{pmatrix} \\ & (A, -b)\begin{pmatrix} x \\ \xi \end{pmatrix} = 0 \\ & (0,1)\begin{pmatrix} x \\ \xi \end{pmatrix} = 1 \\ & x \geq 0,\ \xi \geq 0, \end{aligned}$$

which is of the required form with $g^T = (0,1)$. In addition, Anstreicher shows that fractional linear programming problems can also be handled in this way. A related method was proposed by de Ghellinck and Vial [1986].

The bulk of the work at each iteration consists of obtaining the projection of the gradient in the transformed space, or equivalently solving the linear system (9.25) or (9.29). Thus there is considerable interest in using an approximate projection; such a projection however is unlikely to preserve feasibility if computed in the way described above. An alternative method uses a representation of the null space of $\hat{A}$. Suppose $\hat{Z}$ satisfies $\hat{A}\hat{Z} = 0$ and $[\hat{A}^T, \hat{Z}]$ is square and nonsingular. Then

$$-(I - \hat{A}^T(\hat{A}\hat{A}^T)^{-1}\hat{A}) = -\hat{Z}(\hat{Z}^T\hat{Z})^{-1}\hat{Z}^T$$

and this permits the determination of the direction $\hat{d}$ by solving

$$(\hat{Z}^T\hat{Z})w = \hat{Z}^T\hat{c} \qquad (9.36)$$

and setting $\hat{d} = -\hat{Z}w$. The important point is that $\hat{d}$ is in the null space of $\hat{A}$ even if w is an inexact solution of (9.36). Moreover, if Z is an appropriate null-space matrix for A, then $\hat{Z} = \overline{X}^{-1}Z$ is suitable for $\hat{A} = A\overline{X}$; thus, except to take care of numerical problems, the null-space matrix need not be recomputed at every iteration. Computing the search direction thus was proposed by Shanno and Marsten [1985], who also consider a reduced gradient version, and by Goldfarb and Mehrotra [1988a], who show how accurate $\hat{d}$ needs to be to preserve polynomiality. Goldfarb and Mehrotra [1988b] also show how to combine their inexact projection approach with the ideas in [Todd and Burrell 1986] for improving a lower bound $\underline{z}$ for problems with unknown optimal value. These authors all use the technology of the simplex method and basis matrices to derive a null-space matrix for A or $\hat{A}$.

In [Goldfarb and Mehrotra 1988c] an algorithm is proposed that does not even preserve feasibility; this method must operate on a primal-dual system similar to (9.24). However, when an iterative method is used, the dimensionality of the corresponding system is less important than its structure, and Goldfarb and Mehrotra have shown how to exploit the primal-dual structure efficiently. Another algorithm that does not require feasibility is that of de Ghellinck and Vial [1986]. While this paper requires exact projection, Vial [1986] shows how to preserve polynomiality with inexact projections. Finally, applying Karmarkar's projective method to the dual problem, as proposed by Gay [1987], allows projections to be done inexactly because of the flexibility afforded by the inequality constraints of the dual.

In fact, the first papers to suggest inexact projections were [Karmarkar 1984a,b]. The idea was to approximate the diagonal matrix $\overline{X}$ so that, apart from a multiplicative factor, it differed from the diagonal matrix used in the previous iteration in only a few entries. Thus $A\overline{X}^2A^T$ and its inverse differ from (a multiple of) the corresponding matrices at the previous iteration by a matrix of low rank. Karmarkar showed that each entry of the diagonal matrix could be kept within a constant factor of the appropriate component of x while performing only an average of $O(\sqrt{n})$ rank-one updates per iteration; this led to a reduction of $O(\sqrt{n})$ in the worst-case bound. Recently, Shanno [1988] has investigated this idea further and given encouraging computational results; Shanno notes that approximating $\overline{X}$ can avoid some numerical difficulties when on degenerate problems $A\overline{X}^2A^T$ approaches singularity. An alternative algorithm which updates an approximate $\overline{X}$ using a quasi-Newton method, has been proposed by Dennis, Morshedi and Turner [1987].

Relationships between Karmarkar's algorithm and the ellipsoid method have been investigated by Todd [1986b] and Ye [1986], who show that Karmarkar's algorithm generates a sequence of shrinking ellipsoids containing all dual optimal solutions. In fact, Ye shows that, when viewed in the space of dual slack vectors, the logarithm of the volume of the containing ellipsoid is exactly Karmarkar's potential function.

Finally, Kapoor and Vaidya [1986] have shown how to extend Karmarkar's method to convex quadratic programming. This extension is complicated by the fact that a projective transformation maps the convex

quadratic objective into a nonconvex conic function. The next section discusses approaches related to Karmarkar's method which are more easily extended to this case.

9.5 Related methods

Among the drawbacks of Karmarkar's algorithm are the complications that result from its use of projective transformations. One way to avoid these complications is to use affine rather than projective transformations at each iteration. This gives rise to the so-called "affine scaling" algorithm which was proposed by several researchers, including Barnes [1986] and Vanderbei, Meketon and Freedman [1986], who give convergence analyses. Indeed, this method was developed before Karmarkar's method by Dikin [1967, 1974], who also provided convergence results. In contrast to Karmarkar's, this method operates directly on problems given in standard form and does not require knowledge of the optimal value. Unfortunately, however, no polynomial bound is known for this algorithm.

Let $\overline{x}$ be a strictly feasible solution (i.e., $\overline{x} > 0$) of the standard form problem

$$\begin{aligned} \min\ & c^T x \\ & A x = b \\ & x \geq 0 \end{aligned} \tag{9.37}$$

and consider the affine transformation defined by

$$x \to \hat{x} = \overline{X}^{-1} x, \quad \hat{x} \to x = \overline{X}\hat{x},$$

which takes $\overline{x}$ into the interior point e, which is equidistant from all inequality constraints. The linear programming problem in the transformed space is

$$\begin{aligned} \min\ & \hat{c}^T\hat{x} \\ & \hat{A}\,\hat{x} = b \\ & \hat{x} \geq 0, \end{aligned}$$

with $\hat{A} = A\overline{X}$, $\hat{c} = \overline{X}c$ as before. We then take a step in the negative projected gradient direction

$$\hat{d} = -(I - \hat{A}^T(\hat{A}\hat{A}^T)^{-1}\hat{A})\hat{c},$$

cf. (9.17). In the original space, this direction is the negative of the (scaled) reduced cost vector:

$$\overline{d} = -\overline{X}^2(c - A^T\overline{y}) \quad \text{where} \tag{9.38}$$

$$(A\overline{X}^2A^T)\overline{y} = A\overline{X}^2c; \tag{9.39}$$

note the similarity to (9.25) and (9.28). Then $\overline{x}$ is replaced by $\overline{x} + \overline{\alpha d}$ for a suitable $\overline{\alpha} > 0$. Here $\overline{\alpha}$ is typically chosen to be a large fraction (say .95) of α_{max}, where $\alpha_{max} = \max\{\alpha: \overline{x} + \alpha\overline{d} \geq 0\}$.

More formally, we have

Algorithm 9.2. (primal affine scaling).

Given a strictly feasible solution x^0 to (9.37), choose $0 < \gamma < 1$.

For $k = 0,1,\ldots,$ do

Iteration k: Let $X_k = \mathrm{diag}(x^k)$.

Compute the solution $\bar{y}^k$ to

$$(AX_k^2A^T)y = AX_k^2c, \tag{9.40}$$

and let $\hat{d} = -X_k(c-A^T\bar{y}^k)$, $d^k = X_k\hat{d}$.

If $d^k \geq 0$, STOP: problem is unbounded.

Otherwise, let $\alpha = -\gamma/\min_j \hat{d}_j$ and set $x^{k+1} = x^k + \alpha d^k$.

Test for convergence.

We have not been specific about the convergence test, since without lower bounds it is not clear when x^k is sufficiently accurate. Note that, if $\hat{d} \leq 0$ at any iteration, then y^k is dual feasible so that b^Ty^k yields a valid lower bound. Usually, the algorithm is terminated when the improvement obtained is small, i.e.

$$\frac{c^Tx^k - c^Tx^{k+1}}{\max\{1, |c^Tx^k|\}} \leq \epsilon. \tag{9.41}$$

Observe that $c^Tx^k - c^Tx^{k+1} = \alpha c^TX_k(I - X_kA^T(AX_k^2A^T)^{-1}AX_k)X_kc = \alpha\|\hat{d}\|^2 > 0$, so that the algorithm is a descent method, in contrast to Karmarkar's.

The choice of α above corresponds to moving a proportion γ of the way to the boundary. Alternatively, one could set $\alpha = \gamma/\|\hat{d}\|$, so that the step moves a proportion γ of the way to the boundary of the

inscribed ball to the nonnegative orthant centered at $\hat{x} = e$ in the transformed space. This version is analyzed by Barnes [1986] and Dikin [1974].

A popular way of using the affine scaling method is to apply it to the dual of problem (9.37).

$$\begin{aligned} \max \quad & b^T y \\ & A^T y \leq c. \end{aligned} \tag{9.42}$$

The advantage, as with the projective scaling method, is that the direction can be computed inexactly without losing feasibility. By introducing slack variables $s = c - A^T y$, eliminating the unrestricted variables y, and applying the affine scaling method to the resulting problem in standard form, one is led to the dual affine scaling method of Adler, Karmarkar, Resende and Veiga [1986]. We state this as

Algorithm 9.3 (dual affine scaling).

Given an interior feasible solution y^0 to (9.42), choose $0 < \gamma < 1$. For $k = 0,1,\ldots,d$.

Iteration k. Let $s^k = c - A^T y^k > 0$ and $S_k = \mathrm{diag}(s^k)$.

Compute the solution d^k to

$$(AS_k^{-2} A^T)d = b \tag{9.43}$$

and let $u^k = -A^T d^k$, $\hat{u} = S_k^{-1} u^k$,

and $x^k = -S_k^{-2} u^k$. If $u^k \geq 0$, STOP:

problem (9.42) is unbounded and hence (9.37) is infeasible. Otherwise, let $\alpha = -\gamma / \min_j \hat{u}_j$ and set $y^{k+1} = y^k + \alpha d^k$ (so that $s^{k+1} = s^k + \alpha u^k$).

Test for convergence.

Note that $Ax^k = b$, and if $u^k \leq 0$, then x^k is a feasible solution to (9.37), yielding an upper bound $c^T x^k$ to the optimal value of (9.42). Usually, a termination criterion analogous to (9.41) is used; now it is easy to see that $b^T y^k$ is increasing.

There appears to be no suitable potential function for these affine scaling algorithms, and a polynomial bound is thought to be unlikely. Nevertheless, they perform well in practice--see Vanderbei, Meketon and Freedman [1986] and Adler, Karmarkar, Resende and Veiga [1986]. Convergence has been established under various nondegeneracy conditions--see, e.g., Dikin [1974], Barnes [1986], Vanderbei, Meketon and Freedman [1986] and Vanderbei and Lagarias [1988]. Assuming suitable nondegeneracy conditions one can also show that the vectors y^k defined in Algorithm 9.2 converge to the optimal dual solution and the vectors x^k defined in Algorithm 9.3 converge to the optimal primal solution.

Recently Gonzaga [1988] and Ye [1988] have described methods using the same affine transformation as in the primal affine scaling method, but where the direction in the transformed space is the negative projected gradient of a potential function instead of the cost vector. The potential function has the form

$$f(x,\underline{z}) = q \ \ell n(c^T x - \underline{z}) - \Sigma_j \ \ell n \ x_j$$

where $q = n+\sqrt{n}$ and $\underline{z}$ is a lower bound. These methods are polynomial, and Ye's algorithm in fact only requires $O(\sqrt{n}\ L)$

iterations, as in the path-following algorithms to be discussed below. Ye's method generates sequences of both primal and dual feasible solutions, and whenever the dual solution is changed, he shows that his primal and dual solutions lie close to the central paths to be defined below. Hence his algorithm combines ideas from the projective scaling (potential functions), affine scaling (affine transformations) and path-following (central paths) methods, and achieves what is currently the best complexity bound on the number of iterations.

Potential functions can be thought of as barrier functions. If the optimal value z^* is known, there is no parameter required, so that they can be considered "exact" barrier functions, similar to the exact penalty functions used in nonlinear programming. On the other hand, the relationship of Karmarkar's algorithm to classical logarithmic barrier methods is explored by Gill, Murray, Saunders, Tomlin and Wright [1986]. In particular, it is shown that Karmarkar's algorithm takes a step in the same direction as a projected Newton barrier method with a certain value of the penalty parameter. Gill et al. also describe an implementation of the barrier method and provide extensive computational results.

The idea of approximately following the path of minimizers of a parametrized family of logarithmic barrier problems gives rise to a class of methods which are referred to as "path-following". The logarithmic barrier function method was first proposed by Frisch [1955] for solving convex programming problems and its properties in the

context of nonlinear programming have been carefully studied by Fiacco and McCormick [1968]. For linear programming problems this path of minimizers is called the "central path". It has been thoroughly studied by Megiddo [1986] and by Bayer and Lagarias [1987a] who also show that it is the locus of "analytic centers" of a family of polyhedra determined by the constraints of the linear program together with a parametrized objective value constraint $c^T x \leq \mu$ (or $c^T x = \mu$).

If the polyhedron $P \equiv \{x \in \mathbb{R}^n \mid a_i^T x \leq \beta_i,\ i = 1,\ldots,m\}$ has a nonempty interior $\mathrm{Int}(P)$, and we let

$$f(x) = \ell n \prod_{i=1}^{m} (\beta_i - a_i^T x),\ x \in \mathrm{Int}(P),$$

then ξ is said to be an analytic center of (the linear system of inequalities defining) P if $f(\xi) \geq f(x)$ for all $x \in \mathrm{Int}(P)$. Huard's [1967] "method of centers" for solving (nonlinear) convex programming problems was the first method to use the approach of approximately following a path of centers. In the linear programming case the notion of "analytic center" is explored in Sonnevend [1985,1986] and Bayer and Lagarias [1987a] and is the basis for Renegar's [1988] algorithm. That algorithm was the first path-following algorithm proposed for linear programming and, like the algorithms described below, it approximately follows the central path by using Newton's method and terminates in at most $O(\sqrt{n}\ L)$ iterations. A modified version requiring at most $O(n^3 L)$ arithmetic operations has been developed by Vaidya [1987].

To solve the standard form linear program (9.37), the logarithmic barrier function method considers the family of problems

$$P(\mu): \quad \text{minimize } f(x;\mu) \equiv c^T x - \mu \sum_{j=1}^{n} \ell n\, x_j$$
$$\text{subject to } Ax = b,\ x > 0,$$

where the barrier parameter μ is positive. Assume that (9.37) has a strictly positive feasible solution and that the set of its optimal solutions is nonempty and bounded -- or equivalently, that the dual problem (9.42) has a solution with strictly positive slacks $s = c - A^T y$. Then $P(\mu)$ has a unique global minimizer $x(\mu)$ for all $\mu > 0$ and $x(\mu)$ converges to an optimal solution of (9.37) as $\mu \to 0$. For each μ, the minimizer $x(\mu)$ satisfies the Karush-Kuhn-Tucker necessary conditions

$$SXe - \mu e = 0 \tag{9.44a}$$
$$Ax - b = 0,\ x > 0 \tag{9.44b}$$
$$A^T y + s - c = 0. \tag{9.44c}$$

Since $\mu > 0$ and $x > 0$, (9.44a) implies that the vector of dual slacks s is strictly positive. Hence y is feasible in the dual problem (9.42) and in fact solves the corresponding dual barrier problem

$$D(\mu): \qquad \text{maximize} \quad b^T y + \mu \sum_{j=1}^{n} \ell n\, s_j$$
$$A^T y + s = c$$
$$s > 0.$$

Kojima et al. [1987a] were the first to propose an algorithm for following the path of solutions $\Gamma \equiv \{x(\mu), y(\mu), s(\mu) : \mu > 0\}$ to (9.44) as $\mu \to 0$. Their algorithm, which converges in at most $O(nL)$ steps, and a similar but faster $O(\sqrt{n}\, L)$ algorithm developed independently by Kojima et al. [1987b] and Monteiro and Adler [1987a], both apply a single step of Newton's method to (9.44) to obtain an approximation $(\bar{x}, \bar{y}, \bar{s})$ to $(x(\bar{\mu}), y(\bar{\mu}), s(\bar{\mu}))$, given an approximation (x,y,s) to $(x(\mu), y(\mu), s(\mu))$ where $\bar{\mu}$ is slightly smaller than μ.

One alternate way to follow the central path $\{x(\mu): \mu > 0\}$, or equivalently Γ, was proposed by Gonzaga [1987]. Given x, Gonzaga's method computes $\bar{x}$ by taking a single projected Newton step h for problem $P(\bar{\mu})$. That is, it computes h by solving the equality constrained quadratic program

$$QP(\bar{\mu}) \qquad \text{minimize} \quad \tfrac{1}{2}\, \bar{\mu}\, h^T X^{-2} h + (c - \bar{\mu} X^{-1} e)^T h$$
$$\text{subject to} \quad Ah = 0,$$

obtained by approximating $f(x;\bar{\mu})$ in $P(\bar{\mu})$ by its second order Taylor series expansion about the feasible point x. Notice that if h solves $QP(\bar{\mu})$ then there exists a vector $\bar{y} \in \mathbb{R}^m$ such that

$$\bar{\mu} X^{-2} h + c - \bar{\mu} X^{-1} e - A^T \bar{y} = 0. \qquad (9.45)$$

Since $Ah = 0$ we obtain from (9.45) that

$$h = \frac{1}{\bar{\mu}} X^2 (c - \bar{\mu} X^{-1} e - A^T \bar{y})$$

where

$$(AX^2A^T)\bar{y} = AX^2(c - \bar{\mu} X^{-1} e). \qquad (9.46)$$

We can now formally state

Algorithm 9.4. (primal path following)

Choose a strictly feasible solution $x^0 > 0$ to (9.37) and constants $\mu_1 > 0$, $\tau > 0$, and $\sigma > 0$. For $k = 0,1,\ldots,$ do

Iteration k: Let $X_k = \mathrm{diag}(x^k)$.

Compute the solution y^{k+1} to

$(AX_k^2A^T)y^{k+1} = AX_k^2(c-\mu_{k+1}X_k^{-1}e)$

and let $s^{k+1} = c - A^T y^{k+1}$,

$h^k = -(1/\mu_{k+1})X_k^2(s^{k+1}-\mu_{k+1}X_k^{-1}e)$

and $x^{k+1} = x^k + h^k$.

If $x^{k+1^T} s^{k+1} < \tau$, STOP: x^{k+1} is a near-optimal solution.

Otherwise, set $\mu_{k+2} = (1-\sigma/\sqrt{n})\mu_{k+1}$.

We shall now prove that by properly choosing the initial point x^0, the barrier parameter's initial value μ_1 and its reduction factor σ, and the termination criterion τ, Algorithm 9.4 will obtain a solution x^k such that $c^T x^k - c^T x^* \leq 2^{-L}$ in $O(\sqrt{n}\, L)$ iterations. Our proof depends upon three lemmas which show, respectively, that

(i) if the norm of the scaled step $X_k^{-1} h^k$ is small then (y^{k+1}, s^{k+1}) is dual feasible and the duality gap $s^{k+1^T} x^{k+1} = (c - A^T y^{k+1})^T x^{k+1} = c^T x^{k+1} - b^T y^{k+1}$ is a small multiple of μ_{k+1};

(ii) if $\|X_k^{-1} h^k\|$ is small, then $\|X_{k+1}^{-1} h^{k+1}\|$ is also small; and

(iii) if x^0 is chosen to be the analytic center of the feasible region of (9.37) and μ_1 is chosen suitably large then $\|X_0^{-1} h^0\|$ is small.

Note that, if $\|X_k^{-1} h^k\|$ is small, then the norm of the left hand side of (9.44a) is a small multiple of μ, where $s = s^{k+1}$, $x = x^k$, and $\mu = \mu_{k+1}$. Hence x^k is close to the central path.

Lemma 9.4. If $\|X_k^{-1} h^k\| \leq \delta \leq 1$, then (y^{k+1}, s^{k+1}) is dual feasible and

$$0 \leq s^{k+1^T} x^{k+1} \leq \mu_{k+1} (\delta + \sqrt{n})^2$$

Proof: Observe that on iteration k (9.45) is equivalent to

$$s^{k+1} - \mu_{k+1} X_k^{-1} w^k = 0, \tag{9.47a}$$

where

$$s^{k+1} = c - A^T y^{k+1} \tag{9.47b}$$

and

$$w^k = e - X_k^{-1} h^k. \tag{9.47c}$$

Since $\|X_k^{-1}h^k\| \le \delta \le 1$, $w^k \ge 0$ from (9.47c) and $s^{k+1} \ge 0$ from (9.47a). Consequently (y^{k+1}, s^{k+1}) is dual feasible and the duality gap is

$$x^{k+1^T} s^{k+1} = \mu_{k+1} x^{k+1^T} X_k^{-1} w^k \le \mu_{k+1} \|X_k^{-1} x^{k+1}\| \cdot \|w^k\|$$
$$\le \mu_{k+1} (\delta + \sqrt{n})^2$$

since $\|X_k^{-1}x^{k+1}\| \le \|X_k^{-1}\ x^{k+1} - e\| + \|e\| = \|X_k^{-1}h^k\| + \|e\|$

and $\|w^k\| \le \|e\| + \|X_k^{-1}h^k\|$.

Lemma 9.5. Let $0 < \delta < 1$ and $0 < \sigma \le \bar{\sigma} \equiv \dfrac{\delta - \delta^2}{1 + \delta/\sqrt{n}}$.

If $\|X_k^{-1}h^k\| \le \delta$ then $\|X_{k+1}^{-1}\ h^{k+1}\| \le \delta$

Proof. Since $Ah^{k+1} = 0$, it follows from (9.47b) for iterations k and $k+1$ that $h^{k+1^T} s^{k+1} = h^{k+1^T} c = h^{k+1^T} s^{k+2}$. Hence from (9.47a) we have that $\mu_{k+1} h^{k+1^T} X_k^{-1} w^k = \mu_{k+2}\ h^{k+1^T} X_{k+1}^{-1} w^{k+1}$. Substituting for w^k and w^{k+1} using (9.47c) and rearranging terms yields

$$\mu_{k+2} h^{k+1^T} X_{k+1}^{-2} h^{k+1} = \mu_{k+1} h^{k+1^T} X_k^{-2} h^k + \mu_{k+2} h^{k+1^T} X_{k+1}^{-1}\ e - \mu_{k+1} h^{k+1^T} X_k^{-1} e.$$

Now using $\mu_{k+2} = (1 - \sigma/\sqrt{n})\mu_{k+1}$ and the identity

$$X_{k+1}^{-1}e - X_k^{-1}e = -X_{k+1}^{-1}\, X_k^{-1}\, h^k$$

yields

$$\|X_{k+1}^{-1}h^{k+1}\|^2 = h^{k+1^T}X_{k+1}^{-2}h^{k+1} = \frac{1}{1-\sigma/\sqrt{n}}\, h^{k+1^T}\, X_{k+1}^{-1}[(X_{k+1}X_k^{-1}-I)X_k^{-1}h^k - \frac{\sigma}{\sqrt{n}}\, e].$$

It then immediately follows that

$$\|X_{k+1}^{-1}h^{k+1}\| \leq \frac{\delta^2+\sigma}{1-\sigma/\sqrt{n}} \equiv \theta$$

since $\|X_{k+1}\, X_k^{-1} - I\| = \max_j \{\frac{x_j^{k+1}}{x_j^k} - 1\} = \max_j \{h_j^k/x_j^k\} \leq \delta$. Finally, if $\sigma \leq \bar{\sigma}$ then $\theta \leq \delta$. Moreover $\delta = \frac{1}{2}$ and $\sigma = 1/6$ satisfy the conditions imposed on δ and σ.

Lemma 9.6. If $x^0 > 0$ is feasible to (9.37) and

$$X_0^{-1}e = A^T v, \text{ for some } v \in \mathbb{R}^m, \tag{9.48a}$$

and

$$\|X_0 c\| \leq \mu_1 \delta, \tag{9.48b}$$

then

$$\|X_0^{-1}h^0\| \leq \delta. \tag{9.49}$$

Proof: On the initial iteration, we have from (9.45) and (9.48a) that

$$\mu_1 X_0^{-2} h^0 + c - \mu_1 A^T v - A^T y^1 = 0 \quad \text{for some} \quad v \text{ and } y_1 \in \mathbb{R}^m.$$

Since $Ah^0 = 0$, premultiplying by h^{0^T} yields

$$\|X_0^{-1} h^0\|^2 = -\frac{1}{\mu_1} h_0^T c = -\frac{1}{\mu_1} h^{0^T} X_0^{-1} X_0 c \leq \frac{1}{\mu_1} \|X_0^{-1} h^0\| \, \|X_0 c\|.$$

whence (9.49) follows from (9.48b).

The condition (9.48a) requires that x^0 be an analytic center for (the linear system defining) the feasible region of (9.37), since it and the requirements that x^0 be strictly positive and feasible are the Karush-Kuhn-Tucker conditions for the point x^0 to maximize $\sum_{j=1}^{n} \ell n \, x_j$ subject to $Ax = b$, $x > 0$.

One way to satisfy (9.48a) is to scale the variables in (9.37) so that each is bounded by one and add an artificial variable and column to the problem (i.e., a "big-M" approach) so that the point e is feasible. This results in the linear program

$$\begin{aligned} \text{minimize} \quad & c^T \tilde{x} + M \tilde{x}_{k-1} \\ \text{subject to} \quad & A\tilde{x} + (b/\rho - Ae)\tilde{x}_{k+1} = b/\rho \\ & \tilde{x} + \tilde{x}_{k-1} + \tilde{x}_k = k \\ & \tilde{x} \geq 0, \; \tilde{x}_{k-1} \geq 0, \; \tilde{x}_{k-1} \geq 0, \end{aligned}$$

where $k = n + 2$ and $\tilde{x} = x/\rho$. Moreover, it suffices to take $\ell n \, M$

and $\ell n\, \rho$ to be $O(L)$ (e.g., see [Kojima et al. 1987a] and [Monteiro and Adler 1987a,b]). Hence we can take $\mu_1 = 2^{O(L)}$ in Algorithm 9.4.

From Lemmas 9.4-9.6 and the above remarks we immediately obtain

Theorem 9.3. Let $\ell n\, \mu_1 = O(L)$ and $\ell n\, \tau = -O(L)$ and let the initial feasible point x^0 and constants δ and σ satisfy the conditions of Lemmas 9.4 and 9.5. Then for all $k \geq 1$, x^k is feasible to (9.37), (y^k, s^k) is dual feasible,

$$\|X_k^{-1} h^k\| \leq \delta,$$

and

$$s^{k^T} x^k \leq \mu_k (\delta + \sqrt{n})^2,$$

and Algorithm 9.4 terminates in at most $O(\sqrt{n}\, L)$ iterations.

This theorem shows that the path following algorithms have a worst case complexity bound on the number of iterations that is better than the bound for Karmarkar's algorithm and its variants by a factor of $n^{1/2}$. In [Gonzaga 1987] it is shown how to apply to the path following algorithms the modification introduced by Karmarkar, which allows each iteration to be performed in $O(n^{2.5})$ arithmetic operations amortized over all iterations. Consequently these algorithms have a complexity bound of $O(n^3 L)$ arithmetic operations--the best bound yet achieved.

One of the nice things about path following algorithms is that they can be extended in a straightforward way to solve convex

quadratic programs [Monteiro and Adler 1987b] [Goldfarb and Liu 1988] [Mehrotra and Sun 1987], linear complementarity problems [Kojima et al. 1987b], quadratically constrained convex programs [Jarre 1988], [Mehrotra and Sun 1988] and general convex programs [Monteiro and Adler 1987c], [Mehrotra and Sun 1988b]. On the negative side, the nice theoretical complexity bounds for these algorithms depend crucially on closely following the central path and hence require that very small steps be taken. In contrast the projective methods and the recently introduced potential function methods of Gonzaga [1988] and Ye [1988] allow large steps to be taken without destroying their theoretical properties.

9.6 Implementation

All variants of Karmarkar's algorithm and related methods described in the previous section require the solution of a sequence of symmetric positive definite systems of linear equations of the form

$$B_k y = r \tag{9.50}$$

where $B_k = AD_k^2 A^T$

and D_k is a positive diagonal matrix; e.g., see (9.25), (9.39) and (9.46). There are two basic approaches for solving such systems--direct methods and iterative methods --although as we shall see, sophisticated methods often combine both approaches.

Direct methods usually involve computing either the LU factorization

$$B_k = LU \tag{9.51}$$

using Gaussian elimination or the Cholesky factorization

$$B_k = \tilde{L}\tilde{L}^T, \tag{9.52}$$

where L, U^T and $\tilde{L}$ are all m x m lower triangular matrices. All have positive diagonal elements, with those of L identically equal to one. Because of symmetry

$$L = D_u^{-1} U^T = D_u^{-1/2} \tilde{L},$$

where D_u is a diagonal matrix formed by setting all off-diagonal elements of U to zero. After B_k has been factorized, say into LU, the solution of (9.50) is easily obtained by successively solving the triangular systems

$$Lw = r \text{ and } Uy = w$$

for w and y, respectively.

Because B_k is positive definite the factorizations (9.51) and (9.52) are numerically stable without any need for pivoting. If B_k is sparse, this means that we can try to find a symmetric permutation, PB_kP^T, of the rows and columns of B_K--here P is a permutation matrix--to give as sparse a factorization, or equivalently, to produce as little "fill-in," as possible. Although finding such a permutation is

NP-hard [Yannakakis 1981], there are several efficient ordering heuristics which usually yield good results. These include the "minimum degree" ordering heuristic [Rose 1970], [Tinney and Walker 1967] and the "minimum local fill-in" ordering heuristic [Markowitz, 1957], [Tinney and Walker 1967]. The former is a symmetrized version of the so-called Markowitz criterion [Markowitz 1957] which was proposed for unsymmetric matrices. That criterion selects a pivot so as to minimize the product of the number of nonzeros in the pivotal row in columns that have not been pivotal and the number of nonzeros in the pivotal column in rows that have not been pivotal, excluding the pivot element itself in each case. Clearly, this product is an upper bound on the actual amount of fill-in produced by the prospective pivot. The "minimum local fill-in" heuristic minimizes the number of non-zeros actually created on a single pivot and is far more expensive to execute than the "minimum degree ordering" heuristic.

Since the matrix B_k has the same sparsity structure on every iteration k, "symbolic factorization of B_k--i.e., determination of the pivot sequence (symmetric permutation) and fill-in pattern from the non-zero pattern of AA^T and formation of the data structure for U, including fill-in entries--needs to be done only once at the start. Consequently, the extra effort required to execute the "minimum local fill-in" heuristic does not necessarily rule out its use in favor of the "minimum degree" heuristic.

When the matrix A contains several dense columns, B_k will be dense even if the overall density of A is very low. One way to deal with this situation, so that advantage can be taken of the sparsity in the non-dense columns of A, is to use the Sherman-Morrison-Woodbury (SMW) modification formula to effectively compute projections. Specifically, if A is partitioned into sparse columns $\bar{A}$ and dense

columns $\bar{\bar{A}}$ and the rows and columns of D_k are partitioned conformally, then

$$B_k = \bar{A}\,\bar{D}_k^2\bar{A}^T + \bar{\bar{A}}\,\bar{\bar{D}}_k^2\bar{\bar{A}}^T$$

and the SMW formula yields

$$B_k^{-1} = \bar{B}_k^{-1} - \bar{B}_k^{-1}\bar{\bar{A}}\,\bar{\bar{D}}_k^2\,(I - \bar{\bar{A}}^T\bar{B}_k^{-1}\,\bar{\bar{A}}\,\bar{\bar{D}}_k^2)\bar{\bar{A}}^T\,\bar{B}_k^{-1},$$

where $\bar{B}_k = \bar{A}\,\bar{D}_k\bar{A}^T$ (see Lemma 4.3 for the rank-one case).

An alternate approach described in [Adler et al., 1986] and [Adler et al., 1987] is to compute the Cholesky factorization of $\bar{B}_k$, i.e.,

$$\bar{B}_k = \bar{L}_k\bar{L}_k^T$$

and solve (9.50) by the "preconditioned conjugate gradient" method using $\bar{L}_k$ as a preconditioner (e.g., see [Golub and Van Loan, 1983]. This is equivalent to solving

$$\bar{L}_k^{-1}\,B_k\,\bar{L}_k^{-T}\,u = \bar{L}_k^{-1}r$$

by the conjugate gradient method and then setting $y = \bar{L}_k^T u$. The conjugate gradient method has been used with other or no preconditioners in several variants of Karmarkar's method to compute approximate projections. For example, see [Gill et al. 1986], [Goldfarb and Mehrotra 1988a], [Karmarkar and Ramakrishnan 1988], and [Shanno and Marsten 1985].

During symbolic factorization all memory addresses needed to perform all subsequent numerical factorizations can be computed. This makes it possible for the symbolic factorization step to automatically generate loop-free code--i.e., a loop-free sequence of machine or FORTRAN instructions (the latter uses subscripts in place of memory addresses) for performing the numerical factorization--specific to the matrix at hand. See [Duff, Erisman and Reid 1986] and [Gay 1988] for details. However, the storage requirements of such an approach grow very rapidly with problem size. An alternate approach is to generate an "operations list"--i.e., an array of pointers to the locations of the elements that are arithmetically combined in each

line of standard loop-free FORTRAN code in the numerical factorization step. This approach since it requires less memory than automatic generation of loop-free code can be used for moderately sized problems (e.g., see [Adler et al. 1987]) but its enormous storage demands make it prohibitive for large problems.

Although the factorization of B_k is the most time consuming task in Karmarkar's algorithm, the time required just to form B_k is also significant. (A similar statement applies more generally to the normal equations for solving a linear least squares problem.) Since B_k can be expressed as a sum of symmetric rank-one outer products

$$B_k = \sum_{j=1}^{m} D_k^2(j,j)\ (Ae_j)(Ae_j)^T,$$

and only D_k changes from iteration to iteration, computational effort can be saved by computing all outer-products $(Ae_j)(Ae_j)^T$, $j=1,\ldots,m$, at the start of the algorithm, storing their upper-diagonal elements for later use. Additional savings are effected if inexact projections, as first suggested by Karmarkar [1984a, b], are used. In this scheme an approximate scaling matrix $\tilde{D}_k$ is used and $\tilde{D}_k(j,j)$ is kept the same as $\tilde{D}_{k-1}(j,j)$ unless the relative difference $|D_k(j,j)-\tilde{D}_{k-1}(j,j)|/|\tilde{D}_{k-1}(j,j)|$ is not small, in which case $\tilde{D}_k(j,j)$ is set to the exact value $D_k(j,j)$, and B_k is computed as

$$B_k = B_{k-1} + \sum_{j=1}^{m} (\tilde{D}_k^2(j,j) - \tilde{D}_{k-1}^2(j,j))(Ae_j)(Ae_j)^T.$$

9.7 Theoretical and Computational Significance

It is far too early to make a definitive statement on the comparative computational performance of Karmarkar's algorithm and the simplex method. There are many variants of both algorithms, and more importantly, differences in implementation lead to vastly different behaviors on large sparse problems. Both methods are very sensitive to the structure of the problem; simplex methods perform very poorly on time-staged formulations, whereas the projective method prefers such "low-bandwidth" problems; on the other hand, the reverse situation is likely to hold if each basis matrix B is relatively sparse with BB^T relatively dense. However, a number of computational investigations with great variability in variant used, sophistication of implementation, and classes of problems attacked, have all confirmed a highly significant finding: the new methods typically require only 20-50 iterations to provide highly accurate solutions even to very large problems, and the number of iterations grows very slowly if at all with the problem dimension. This result contrasts strongly with the worst-case bound, which indicates at least linear growth with n.

On the other hand, the time taken per iteration with any straightforward implementation renders the complete algorithm uncompetitive with modern simplex codes, and it is necessary to take great care in handling the large sparse least-squares subproblems that arise. When this is done, the result can be a computer code that

appears at least comparable to the simplex implementations, and at least on some large-scale problems superior--see [Adler, Karmarkar, Resende and Veiga 1986], [Gill, Murray, Saunders, Tomlin and Wright 1986] and [Karmarkar and Ramakrishnan 1988].

In contrast to our cautiously optimistic view of the new algorithm's computational significance, it appears that its theoretical implications are far more limited than those of the ellipsoid method. Indeed, Karmarkar's algorithm requires the linear programming problem to be given explicitly with all its constraints and variables listed, and does not appear directly susceptible to column or constraint generation. Thus it cannot be used to provide polynomial algorithms for several combinatorial optimization problems that have been successfully analyzed by the ellipsoid method. Similarly, it seems hard to develop in this framework the many useful post-optimality techniques for handling the addition of new variables or constraints to a given linear programming problem that has already been solved. Doubtless much research will be devoted to providing such flexibility for Karmarkar's algorithm and methods related to it in the coming years.

This striking contrast between the ellipsoid method and the projective algorithm with respect to their computational and theoretical significance prompts us to conclude by remarking on the close similarity between the fundamental subproblems to be solved at each iteration of the two methods. In the ellipsoid method, the center $\bar{y}$ of the current ellipsoid is the solution to the weighted least-squares problem

$$\min_{y \in \mathbb{R}^m} \|D^{1/2}(A^T\bar{y} - r)\|$$

as in (8.18), while in Karmarkar's algorithm the dual solution $\bar{y}$ which leads to the search direction $\hat{d}$ used in the primal solves the weighted least-squares problem

$$\min_{y \in \mathbb{R}^m} \|\bar{X}(A^T\bar{y} - c)\|$$

as in (9.27). The difference lies in the way the weighting matrix $D^{1/2}$ or $\bar{X}$ changes at each iteration. In the ellipsoid method, as we have indicated, only one diagonal entry of D changes; this leads to a simple update of $\bar{y}$, but a very large number of iterations. In projective algorithms all diagonal entries of $\bar{X}$ change, in a sophisticated way; this means a fresh least-squares subproblem must be solved at each iteration, but the number of iterations required appears very low.

References

Adler, I., Karmarkar, N., Resende, M.G.C., and Veiga, G. (1986), "An implementation of Karmarkar's algorithm for linear programming," Working Paper, Operations Research Center, University of California, Berkeley, California.

Adler, I., Karmarkar, N., Resende, M.G.C., and Veiga, G. (1987), "Data structures and programming techniques for the implementation of Karmarkar's algorithm," Working Paper, Operations Research Center, University of California, Berkeley, California.

Adler, I., Karp, R.M., and Shamir, R. (1983b), "A simplex variant solving an m x d linear program in $O(\min(m^2, d^2))$ expected number of pivot steps," Report UCB/CSD 83/158, Computer Science Division, University of California, Berkeley, California.

Adler, I. and Megiddo, N. (1984), "A simplex algorithm whose average number of steps is bounded between two quadratic functions of the smaller dimension," Journal of the Association for Computing Machinery 32, 871–895.

Anstreicher, K.M. (1986), "A monotonic projective algorithm for fractional linear programming," Algorithmica 1, 483–498.

Balinski, M.L. and Russakoff, A. (1972), "On the assignment polytope," Mathematical Programming 3, 257–258.

Barnes, E.R. (1986), "A variation on Karmarkar's algorithm for solving linear programming problems," Mathematical Programming 36, 174–182.

Bartels, R.H. (1971), "A stabilization of the simplex method," Numerische Mathematik 16, 414–434.

Bartels, R.H. and Golub, G.H. (1969), "The simplex method for linear programming using LU decomposition," Communications of the ACM 12, 266–268.

Bayer, D. and Lagarias, J.C. (1987a), "The nonlinear geometry of linear programming, I. Affine and projective scaling trajectories," to appear in Transactions of the American Mathematical Society.

Bayer, D. and Lagarias, J.C. (1987b), "The nonlinear geometry of linear programming, II. Legendre transform coordinates and central trajectories," to appear in Transactions of the American Mathematical Society.

Beale, E.M.L. (1954), "An alternative method for linear programming," Proceedings of the Cambridge Philosophical Society 50, 513–523.

Blair, C.E. (1986), "The iterative step in the linear programming algorithm of N. Karmarkar," Algorithmica 1, 537–539.

Bland, R.G., Goldfarb, D., and Todd, M.J. (1981), "The ellipsoid method: a survey," Operations Research 29, 1039–1091.

Bolza, O. (1914), "Über variations probleme mit Ungleichungen als Nebenbedingungen," Mathematische Abhandlungen (H.A. Schwartz, ed.), 1–18.

Borgwardt, K.H. (1987), The Simplex Method: A Probabilistic Analysis, Springer–Verlag, Berlin.

Burrell, B.P. and Todd, M.J. (1985), "The ellipsoid method generates dual variables," Mathematics of Operations Research 10, 688–700.

Chvátal, V. (1983), Linear Programming, Freeman, New York.

Dantzig, G.B. (1951), "Maximization of a linear function of variables subject to linear inequalities," Activity Analysis of Production and Allocation (Tj. C. Koopmans, ed.), Wiley, New York, 339–347.

Dantzig, G.B. (1963), Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey.

Dantzig, G.B. and Van Slyke, R.M. (1967), "Generalized upper bounding techniques," Journal of Computer System Sciences 1, 213–226.

Dantzig, G.B. and Wolfe, Ph. (1960), "Decomposition principle for linear programs," Operations Research 8, 101–111.

de Ghellinck, G. and Vial, J.–Ph. (1986), "A polynomial Newton method for linear programming," Algorithmica 1, 425–453.

Dennis, J.E. Jr., Morshedi, A.M., and Turner, K. (1987), "A variable–metric variant of the Karmarkar algorithm for linear programming," Mathematical Programming 39, 1–20.

Dikin, I.I. (1967), "Iterative solution of problems of linear and quadratic programming," Doklady Akademiia Nauk SSSR 174, 747–748 [English translation: Soviet Mathematics Doklady 8, 674–675].

Dikin, I.I. (1974), "On the convergence of an iterative process," Upravlyaemye Sistemi 12, 54–60 (in Russian).

Dorfman, R., Samuelson, P.A., and Solow, R.M. (1958), Linear Programming and Economic Analysis, McGraw–Hill, New York.

Duff, I.S., Erisman, A.M., and Reid, J.K. (1986), Direct Methods for Sparse Matrices, Clarendon Press, Oxford.

Ecker, J.G. and Kupferschmidt, M. (1985), "A computational comparison of the ellipsoid algorithm with several nonlinear programming algorithms," SIAM Journal on Control and Optimization 23, 657–674.

Farkas, J. (1902), "Theorie der einfachen Ungleichungen," Journal für die reine und angewandte Mathematik 124, 1–27.

Fiacco, A.V. and McCormick, G.P. (1968), Nonlinear Programming: Sequencial Unconstrained Minimization Techniques, Wiley, New York.

Forrest, J.J.H. and Tomlin, J.A. (1972), "Updating triangular factors of the basis to maintain sparsity in the product form of the simplex method," Mathematical Programming 2, 263–278.

Fourier, J.B.J. (1826), "Solution d'une question particuliere du calcul des inégalités," Nouveau Bulletin des Sciences par la Société Philomathique de Paris, 99–100.

Frisch, K.R. (1955), "The logarithmic potential method of convex programming," Memorandum, University Institute of Economics, Oslo, Norway.

Gács, P. and Lovász, L. (1981), "Khachiyan's algorithm for linear programming," Mathematical Programming Study 14, 61–68.

Gale, D. (1960), The Theory of Linear Economic Models, McGraw–Hill, New York.

Gallo, G. and Sandi, C. (1986), "Netflow at Pisa," Mathematical Programming Study 26.

Gay, D. (1987), "A variant of Karmarkar's linear programming algorithm for problems in standard form," Mathematical Programming 37, 81–90.

Gay, D.M. (1988), "Massive memory buys little speed for complete, in–core sparse Cholesky factorizations," manuscript, AT&T Bell Laboratories, Murray Hill, New Jersey.'

Gill, P.E., Murray, W., Saunders, M.A., Tomlin, J.A., and Wright, M.H. (1986), "On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method," Mathematical Programming 36, 183–209.

Gilmore, P.C. and Gomory, R.E. (1961), "A linear programming approach to the cutting–stock problem," Operations Research 9, 849–859.

Gilmore, P.C. and Gomory, R.E. (1963), "A linear programming approach to the cutting–stock problem – Part II," Operations Research 11, 863–888.

Goldfarb, D. and Liu, S. (1988), "An $0(n^3L)$ primal interior point algorithm for convex quadratic programming," manuscript, Department of Industrial Engineering and Operations Research, Columbia University, New York.

Goldfarb, D. and Mehrotra, S. (1988a), "A relaxed version of Karmarkar's method," Mathematical Programming 40, 289–315.

Goldfarb, D. and Mehrotra, S. (1988b), "Relaxed variants of Karmarkar's algorithm for linear programs with unknown optimal objective value," Mathematical Programming 40, 183–195.

Goldfarb, D. and Mehrotra, S. (1988c), "A self–correcting version of Karmarkar's algorithm," manuscript, Department of Industrial Engineering and Operations Research, Columbia University, New York, to appear in SIAM Journal on Numerical Analysis.

Goldfarb, D. and Reid, J.K. (1977), "A practicable steepest–edge simplex algorithm," Mathematical Programming 12, 361–371.

Golub, G.H. and Van Loan, C.F. (1983), Matrix Computations, Johns Hopkins University Press, Baltimore, Maryland.

Gonzaga, C. (1985), "A conical projection algorithm for linear programing," manuscript, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, to appear in Mathematical Programming.

Gonzaga, C. (1987), "An algorithm for solving linear programming in $O(n^3L)$ operations," Memorandum NO UCB/ERL M87/10, Electronics Laboratory, College of Engineering, University of California, Berkeley, California.

Gonzaga, C. (1988), "Polynomial affine algorithms for linear programming," Report ES–139/88, Universidade Federal do Rio de Janeiro, Brazil.

Grötschel, M., Lovász, L., and Schrijver, A. (1981), "The ellipsoid method and its consequences in combinatorial optimization," Combinatorica 1, 169–197.

Grötschel, M., Lovász, L., and Schrijver, A. (1988), The Ellipsoid Method and Combinatorial Optimization, Springer Verlag, Heidelberg.

Hellerman, E. and Rarick, D. (1971), "Reinversion with the preassigned pivot procedure, " Mathematical Programming 1, 195–216.

Hellerman, E. and Rarick, D. (1972), "The partitioned preassigned pivot procedure (P^4)," in: Sparse Matrices and their Applications (D.J. Rose and R.A. Willoughby, eds.), Plenum Press, New York, 67–76.

Ho, J.K. (1987), "Recent advances in decomposition," Mathematical Programming Study 31, 119–128.

Huard. P. (1967), "Resolution of mathematical programming with nonlinear constraints by the method of centers," in: Nonlinear Programming (J. Abadie, ed.), North–Holland, Amsterdam.

Jarre, F. (1987), "On the convergence of the method of analytic centers when applied to convex quadratic programs," manuscript, Institut fuer Angewandte Mathematik und Statistik, Universitaet Wurzburg, Wurzburg.

John, F. (1948), "Extremum problems with inequalities as subsidiary conditions," Studies and Essays, Interscience, New York, 187–204.

Kantorovich, L.V. (1939), Mathematical Methods of Organizing and Planning Production (in Russian), Publication House of the Leningrad State University, Leningrad [English translation: Management Science 6, (1959–60), 366–422].

Kapoor, S. and Vaidya, P.M. (1986), "Fast algorithms for convex quadratic programming and multicommodity flows," Proceedings of the 18th ACM Symposium on Theory of Computing, 147–159.

Karmarkar, N. (1984a), "A new polynomial time algorithm for linear programming," Proceedings of the 16th Annual ACM Symposium on the Theory of Computing, 302–311.

Karmarkar, N. (1984b), "A new polynomial time algorithm for linear programming," Combinatorica 4, 373–395.

Karmarkar, N. and Ramakrishnan, K.G. (1988), "Implementation and computational results of the Karmarkar algorithm for linear programming, using an iterative method for computing projections," extended abstract. AT&T Bell Laboratoriess, Murray Hill, NJ, presented at the XIII International Symposium on Mathematical Programming, Tokyo.

Karmarkar, N. and Sinha, L.P. (1985), "Application of Karmarkar's algorithm to overseas telecommunications facilities planning," paper presented at XII International Symposium on Mathematical Programming, Boston.

Karp, R.M. and Papadimitriou, C.H. (1982), "On linear characterizations of combinatorial optimization problems," SIAM Journal on Computing 11, 620–632.

Karush, W. (1939), Minima of Functions of Several Variables with Inequalities as Side Constraints, M.Sc. Dissertation, Department of Mathematics, University of Chicago, Chicago, Illinois.

Kennington, J.L. (1978), "A survey of linear cost multicommodity network flows," Operations Research 26, 209–236.

Khachian, L.G. (1979), "A polynomial algorithm in linear programming," (in Russian), Doklady Akademiia Nauk SSSR 244, 1093–1096 [English translation: Soviet Mathematics Doklady 20, 191–194].

Khachian, L.G. (1980), "Polynomial algorithms in linear programming," (in Russian), Zhurnal Vychisditel'noi Matematiki i Matematicheskoi Fiziki 20, 51–68 [English translation: USSR Computational Mathematics and Mathematical Physics 20, 53–72].

Klee, V. and Minty, G.J. (1972), "How good is the simplex algorithm?, in: Inequalities III (O. Shisha, ed.), Academic Press, New York, 159–175.

Klingman, D. and Russell, R. (1975), "Solving constrained transportation problems," Operations Research 23, 91–106.

Kojima, M., Mizuno, S., and Yoshise, A. (1987a), "A primal–dual interior point method for linear programming," Research Report No. B–188, Department of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan.

Kojima, M., Mizuno, S. and Yoshise, A. (1987b), "A polynomial–time algorithm for a class of linear complementarity problems," Research Report No. B–193, Department of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan, to appear in Mathematical Programming.

Kuhn, H.W. and Tucker, A.W. (1951), "Nonlinear programming," in: Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley, California, 481–492.

Lemke, C.E. (1954), "The dual method of solving the linear programming problem," Naval Research Logistics Quarterly 1, 36–47.

Levin, A. Yu. (1965), "On an algorithm for the minimization of convex functions," (in Russian), Doklady Akademii Nauk SSSR 160, 1244–1247 [English translation: Soviet Mathematics Doklady 6, 286–290].

Markowitz, H.M. (1957), "The elimination form of the inverse and its application to linear programming," Management Science 3, 255–269.

McShane, K.A., Monma, C.L., and Shanno, D. (1988), "An implementation of a primal–dual interior point method for linear programming," Rutcor Research Report, Rutgers University, New Brunswick, New Jersey.

Megiddo, N. (1984), "Linear programming in linear time when the dimension is fixed," Journal of the Association for Computing Machinery 31, 114–127.

Megiddo, N. (1986), "Pathways to the optimal set in linear programming," Research Report RJ 5295, IBM Almaden Research Center, San Jose, California.

Megiddo, N. (1987), "On the complexity of linear programming," Advances in Economic Theory (T. Bewley, ed.), Cambridge University Press, Cambridge, 225–268.

Megiddo, N. and Shub, M. (1986), "Boundary behavior of interior point algorithms for linear programming," IBM Research Report RJ 5319, T.J. Watson Research Center, Yorktown Heights, New York 10598.

Mehrotra, S. and Sun, J. (1987), "An algorithm for convex quadratic programming that requires $0(n^{3.5}L)$ arithmetic operations," manuscript, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

Mehrotra, S. and Sun, J. (1988a), "A method of analytic centers for quadratically constrained convex quadratic programs," manuscript, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

Mehrotra, S. and Sun, J. (1988b), "An interior point algorithm for solving smooth convex programs based on Newton's method," manuscript, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

Monma, C.L. and Morton, A.J. (1987), "Computational experience with a dual affine variant of Karmarkar's method for linear programming," Operations Research Letters 6, 261–267.

Monteiro, R.C. and Adler, I. (1987a), "An $O(n^3L)$ primal–dual interior point algorithm for linear programming," manuscript, Department of Industrial Engineering and Operations Research, University of California, Berkeley, California, to appear in Mathematical Programming.

Monteiro, R.C. and Adler, I. (1987b), "An $O(n^3L)$ interior point algorithm for convex quadratic programming," manuscript, Department of Industrial Engineering and Operations Research, University of California, Berkeley, California, to appear in Mathematical Progrmming.

Monteiro, R.C. and Adler, I. (1987c), "An extension of Karmarkar type algorithm to a class of convex separable programming problems with global linear rate of convergence", manuscript, Department of Industrial Engineering and Operations Research, University of California, Berkeley, California.

Murtagh, B.A. and Saunders, M.A. (1978), "Large–scale linearly constrained optimization," Mathematical Programming 14, 41–72.

Murty, K.G. (1980), "Computational complexity of parametric linear programming," Mathematical Programming 19, 213–219.

Murty, K.G. (1983), Linear Programming, John Wiley, New York.

Newman, D.J. (1965), "Location of the maximum on unimodal surfaces," Journal of the Association for Computing Machinery 12, 395–398.

Padberg, M.W. (1985), "A different convergence proof of the projective method for linear programming," Report, New York University, New York, New York.

Padberg, M.W. and Rao, M.R. (1980), "The Russian method and integer programming," GBA Working Paper, New York University, New York, New York (to appear in Annals of Operations Research).

Reid, J.K. (1982), "A sparsity–exploiting variant of the Bartels–Golub decomposition for linear programming bases," Mathematical Programming 24, 55–69.

Renegar, J. (1988), "A polynomial–time algorithm based on Newton's method for linear programming," Mathematical Programming 40, 59–93.

Rose, D.J. (1970), "Symmetric elimination on sparse positive definite systems and potential flow network problem," Ph.D. Thesis, Harvard University, Cambridge, Massachusetts.

Saunders, M.A. (1976), "A fast stable implementation of the simplex method using Bartels–Golub updating," in: Sparse Matrix Computations (J.R. Bunch and D.J. Rose, eds.), Academic Press, New York, 213–226.

Schrage, L. (1975), "Implicit representation of variable upper bounds in linear programming," Mathematical Programming Study 4, 118–132.

Schrijver, A. (1986), Theory of Linear and Integer Programming, John Wiley, Chichester – New York – Brisbane – Toronto – Singapore.

Shamir, R. (1987), "The efficiency of the simplex method: a survey," Management Science 33, 301–334.

Shanno, D.F. (1988), "Computing Karmarkar projections quickly," Mathematical Programming 41, 61–71.

Shanno, D.F. and Marsten, R.E. (1985), "On implementing Karmarkar's algorithm," Working Paper, Graduate School of Administration, University of California, Davis, California.

Shor, N.Z. (1970), "Utilization of the operation of space dilatation in the minimization of convex functions," (in Russian), Kibernetika 1, 6–12 [English translation: Cybernetics 6, 7–15].

Shor, N.Z. (1985), Minimization Methods for Non–Differentiable Functions, Springer–Verlag, Berlin.

Sonnevend, G. (1985), "An analytical centre for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming," Proceedings of the 12th IFIP Conference on System Modeling and Optimization, Budapest, to appear in Lecture Notes in Control Information Sciences, (Springer–Verlag).

Sonnevend, G. (1986), "A new method for solving a set of linear (convex) inequalities and its application for identification and optimization," Proceedings of the Symposium on Dynamic Modelling, IFAC–IFORS. Budapest.

Steger, A. (1985), "An extension of Karmarkar's algorithm for bounded linear programming problems," M.S. Thesis, SUNY at Stonybrook, New York.

Tardos, E. (1986), "A strongly polynomial algorithm to solve combinatorial linear programs," Operations Research 34, 250–256.

Tinney, W.F. and Walker, J.W. (1967), "Direct solutions of sparse network equations by optimally ordered triangular factorization," Proceedings of the IEEE 55, 1801–1809.

Todd, M.J. (1982), "An implementation of the simplex method for linear programming problems with variable upper bounds," Mathematical Programming 23, 34–49.

Todd, M.J. (1983), "Large–scale linear programming: geometry, working bases and factorization," Mathematical Programming 26, 1–20.

Todd, M.J. (1986a), "Polynomial expected behavior of a pivoting algorithm for linear complementarity and linear programming problems," Mathematical Programming 35, 173–192.

Todd, M.J. (1986b), "Improved bounds and containing ellipsoids in Karmarkar's linear programming algorithm," to appear in Mathematics of Operations Research.

Todd, M.J. (1988), "Polynomial algorithms for linear programming," in: Advances in Optimization and Control (H.A. Eiselt and G. Pederzoli, eds.), Springer–Verlag Berlin, 49–66.

Todd, M.J. and Burrell, B.P. (1986), "An extension of Karmarkar's algorithm for linear programming using dual variables," Algorithmica 1, 409–424.

Vaidya, P.M. (1987), "An algorithm for linear programming which requires $O(((m + n)n^2 + (m + n)^{1.5}n)L)$ arithmetic operations," preprint, AT&T Bell Laboratories, Murray Hill, New Jersey.

Valentine, F.A. (1937), "The problem of Lagrange with differential inequalities as added side conditions," Contributions to the Calculus of Variations 1933–37, University of Chicago Press.

Vanderbei, R.J. and Lagarias, J.C. (1988), "I.I. Dikin's convergence result for the affine–scaling algorithm," manuscript, AT&T Bell Laboratories, Murray Hill, NJ.

Vanderbei, R.J., Meketon, M.S., and Freedman, B.A. (1986), "A modification of Karmarkar's linear programming algorithm," Algorithmica 1, 395–407.

Vial, J.–Ph. (1986), "Approximate projections in a projective method for the linear feasibility problem," Département d'Economie Commerciale et Industrielle, Université de Genéve, Genéve, Switzerland and CORE Discussion Paper 8713, Louvain, Belgium (1987).

Yannakakis, M. (1981), "Computing the minimum fill–in is NP–Complete," SIAM Journal on Algebraic and Discrete Methods 2, 77–79.

Ye, Y. (1987), "Karmarkar's algorithm and the ellipsoid method," Operations Research Letters 4, 177–182.

Ye, Y. (1988), "A class of potential functions for linear programming," manuscript, Department of Management Sciences, The University of Iowa, Iowa City, Iowa.

Ye, Y. and Kojima, M. (1987), "Recovering optimal dual solutions in Karmarkar's polynomial algorithm for linear programming," Mathematical Programming 39, 305–317.

Yudin, D.B. and Nemirovskii, A.S. (1976), "Informational complexity and efficient methods for the solution of convex extremal problems," (in Russian), Ékonomika i Matematicheskie Metody 12, 357–369 [English translation: Matekon 13 (2), 3–25].

Zadeh, N. (1973), "A bad network problem for the simplex method and other minimum cost flow algorithms," Mathematical Programming 5, 255–266.