Origins of the Simplex Method

George B. Dantzig

1. Introduction

In the summer of 1947, when I began to work on the simplex method for solving linear programs, the first idea that occurred to me is one that would occur to any trained mathematician, namely the idea of step-by-step descent (with respect to the objective function) along edges of the convex polyhedral set from one vertex to an adjacent one. I rejected this algorithm outright on intuitive grounds—it had to be inefficient because it proposed to solve the problem by wandering along some path of outside edges until the optimal vertex was reached. I therefore began to look for other methods that gave more promise of being efficient, such as those that went directly through the interior [4].

Today we know that before 1947, four isolated papers had been published on special cases of the linear programming problem: papers by Fourier (1824) [10], de la Vallée Poussin (1911) [9], Kantorovich (1939) [13], and Hitchcock (1941) [11]. All except Kantorovich's paper proposed, as a solution method, descent along the outside edges of the polyhedral set, which is the way we describe the simplex method today. There is no evidence that these papers had any influence on each other. Evidently they sparked zero interest on the part of other mathematicians and were unknown to me when I first proposed the simplex method. As we shall see, the simplex algorithm evolved from a very different geometry, one in which it appeared to be very efficient.

The linear programming problem is to find

$$\min z, x \geq 0 \text{ such that } Ax = b,\ \ cx = z(\min)$$

where $x = (x_1, \ldots, x_n)$, A is an m by n matrix, and b and c are column and row vectors.

Curiously, up to 1947 when I first proposed that a model based on linear inequalities be used for planning activities of large-scale enterprises, linear inequality theory had produced only 40 or so papers, in contrast to linear equation theory and the related subjects of linear algebra and approximation, which had produced a vast literature [16]. Perhaps this disproportionate interest in linear equation theory was motivated more than mathematicians care to admit by its practical use as an important tool in engineering and physics, and by the belief that

linear inequality systems would not be practical to solve unless they had three or fewer variables [10].

My proposal served as a kind of trigger—ideas that had been brewing all through World War II but had never found expression burst forth like an explosion. Almost two years to the day after I first proposed that linear programming (LP) be used for planning, Koopmans organized the 1949 conference (now referred to as *The Zero-th Symposium on Mathematical Programming*) at the University of Chicago. There mathematicians, economists, and statisticians presented their research and produced a remarkable proceedings [15]. LP soon became part of the newly developing professional fields of Operations Research and Management Science. Today thousands of linear programs are solved daily throughout the world to schedule industry. These involve many hundreds, thousands, and sometimes tens of thousands of equations and variables. Some mathematicians rank LP as "the newest yet most potent of mathematical tools" [1].

John von Neumann, Tjalling Koopmans, Albert Tucker, and others well known today, some just starting their careers back in the late 1940s, played important roles in LP's early development. A group of young economists associated with Koopmans (R. Dorfman, K. Arrow, P. Samuelson, H. Simon, and others) became active contributors to the field. Their research on LP had a profound effect on economic theory and led to Nobel Prizes. Another group led by Tucker, notably including D. Gale and H. Kuhn, began the development of the mathematical theory.

This outpouring between the years of 1947 and 1950 coincided with the first building of digital computers. The computer became the tool that made the application of linear programming possible. Everywhere we looked, we found practical applications that no one earlier could have posed seriously as optimization problems because solving them by hand computation would have been out of the question. By good luck, clever algorithms in conjunction with computer development gave early promise that linear programming would become a practical science. The intense interest by the Defense Department in the linear programming application also had an important impact on the early construction of computers [5]. The U.S. National Bureau of Standards, with Pentagon funding, became a focal point for computer development under Sam Alexander; its Mathematics Group under John Curtis began the first experiments on techniques for solving linear programs, primarily by Alan Hoffman, Theodore Motzkin, and others [12].

Since we could see possible applications of linear programs everywhere we looked, it seemed only natural to suppose that there was an extensive literature on the subject. To my surprise, I found in my search of the contemporary literature of 1947 only a few references on

linear inequality systems and none on solving an optimization problem subject to linear inequality constraints.

T. S. Motzkin, in his definitive 1936 Ph.D. thesis on linear inequalities [16], makes no mention of optimizing a function subject to a system of linear inequalities. However, 15 years later at the First Symposium on Linear Programming (June 1951), Motzkin declared: "There have been numerous rediscoveries [of LP] partly because of the confusingly many different geometric interpretations which these problems admit." He went on to say that different geometric interpretations allow one "to better understand and sometimes to better solve cases of these problems as they appeared and developed from a first occurrence in Newton's *Methodus Fluxionim* to right now."

The "numerous rediscoveries" that Motzkin referred to were probably the two or three papers we have already cited concerned with finding the least sum of absolute deviations, minimizing the maximum deviation of linear systems, or determining whether a solution to a system of linear inequalities exists. Fourier pointed out as early as 1824 that these were all equivalent problems [10]. Linear programs, however, had also appeared in other guises. In 1928, von Neumann [20] formulated the zero-sum matrix game and proved the mini-max theorem, a forerunner of the duality theorem of linear programming (for which he is also due credit) [3]. In 1936, Neyman and Pearson considered the problem of finding an optimal critical region for testing a statistical hypothesis. Their Neyman-Pearson Lemma is a statement about the Lagrange Multipliers associated with an optimal solution to a linear program [18].

After I had searched the contemporary literature of 1947 and found nothing, I made a special trip to Chicago in June 1947 to visit T. J. Koopmans to see what economists knew about the problem. As a result of that meeting, Leonid Hurwicz, a young colleague of Koopmans, visited me in the Pentagon in the summer and collaborated with me on my early work on the simplex algorithm, a method which we described at the time as "climbing up the bean pole": We were maximizing the objective.

Later I made another special trip, this one to Princeton in the fall of 1947, to visit the great mathematician Johnny von Neumann to learn what mathematicians knew about the subject. This was after I had proposed the simplex method, but before I realized how efficient it was going to be [4].

The origins of the simplex method go back to one of two famous unsolved problems in mathematical statistics proposed by Jerzy Neyman, which I mistakenly solved as a homework problem; it later became part of my Ph.D. thesis at Berkeley [8]. Today we would describe this problem as proving the existence of optimal Lagrange multipliers for a semi-infinite linear program with bounded variables.

Given a sample space Ω whose sample points u have a known probability distribution $dP(u)$ in Ω, the problem I considered was to prove the existence of a critical region ω in Ω that satisfied the conditions of the Neyman-Pearson Lemma. More precisely, the problem concerned finding a region ω in Ω that minimized the Lebesgue-Stieltjes integral defined by (3) below, subject to (1) and (2):

$$\int_{\omega} dP(u) = \alpha \tag{1}$$

$$\alpha^{-1} \int_{\omega} f(u) dP(u) = b \tag{2}$$

$$\alpha^{-1} \int_{\omega} g(u) dP(u) = z(\min) \tag{3}$$

where $0 < \alpha < 1$ is the specified "size" of the region; $f(u)$ is a given vector function of u with $m - 1$ components whose expected value z over ω is specified by the vector b; and $g(u)$ is a given scalar function of u whose unknown expected value z over ω is to be minimized.

Instead of finding a critical region, we can try to find the characteristic function $\phi(u)$ with the property that $\phi(u) = 1$ if $u \in \omega$ and $\phi(u) = 0$ if $u \notin \omega$. The original problem can then be restated as:

Find $\min z$ and a function $\phi(u)$ for $u \in \Omega$ such that:

$$\int_{u \in \Omega} \phi(u) dP(u) = \alpha \qquad 0 \leq \phi(u) \leq 1$$

$$\alpha^{-1} \int_{u \in \Omega} f(u) \phi(u) dP(u) = b$$

$$\alpha^{-1} \int_{u \in \Omega} g(u) \phi(u) dP(u) = z(\min)$$

A discrete analog of this semi-infinite linear program can be obtained by selecting n representative sample points $u^1, \ldots, u^j, \ldots, u^n$ in Ω and replacing $dP(u^j)$ by discrete point probabilities $\Delta_j > 0$, where n may be finite or infinite. Setting

$$x_j = (\Delta_j / \alpha) \cdot \phi(u^j) \qquad 0 \leq x_j \leq \Delta_j / \alpha$$

the approximation problem becomes the bounded variable LP:

$$\text{Find } \min z,\ 0 \leq x_j \leq \Delta_j / \alpha_j: \qquad \sum_{1}^{n} x_j = 1 \qquad \sum_{1}^{n} A_{.j} x_j = b \tag{4}$$

$$\sum_{1}^{n} c_j x_j = z(\min)$$

where $f(u^j) = A_{\cdot j}$ are $m-1$ component column vectors, and $g(u^j) = c_j$.

Since n, the number of discrete j, could be infinite, I found it more convenient to analyze the LP problem in the geometry of the finite $(m + 1)$ dimensional space associated with the coefficients in a column. I did so initially with the convexity constraint (4) but with no explicit upper bound on the non-negative variables x_j [2,3,15]. The first coefficient in a column (the one corresponding to (4)) is always 1, so my analysis omitted the initial 1 coordinate. Each column $(A_{\cdot j}, c_j)$ becomes a point (y, z) in R^m where $y = (y_1, \ldots, y_{m-1})$ has $m - 1$ coordinates.

The problem can now be interpreted geometrically as one of assigning weights $x_j \geq 0$ to the n points $(y^j, z^j) = (A_{\cdot j}, c_j)$ in R^m so that the "center of gravity" of these points (see Figure 1) lies on the

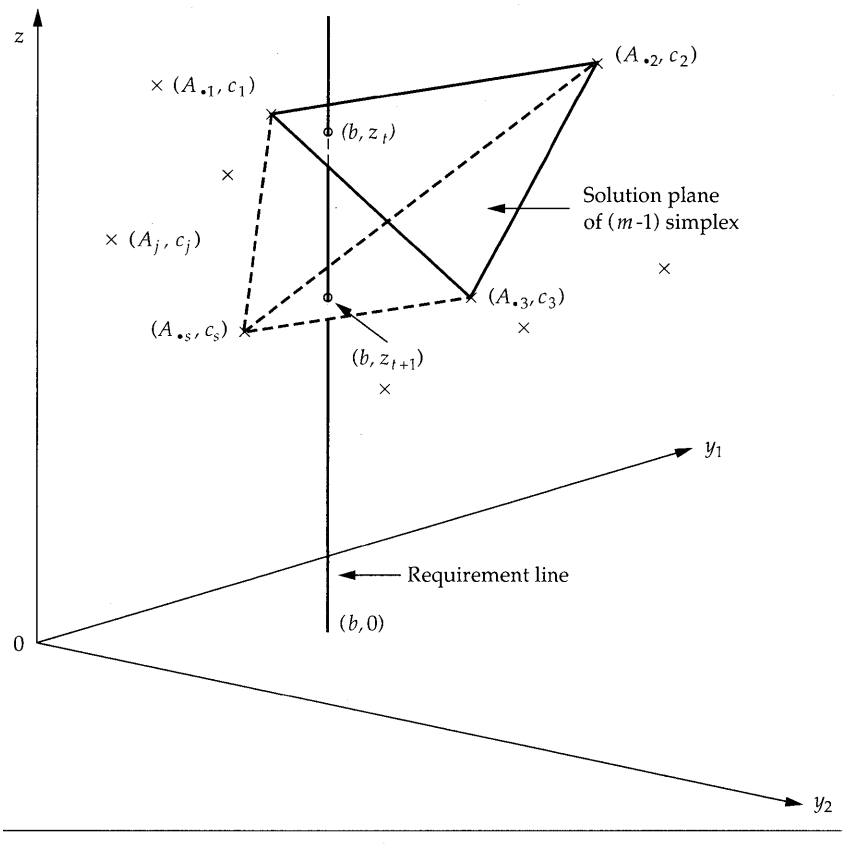

FIGURE 1.
The m-dimensional simplex.

vertical "requirement" line (b, z) and such that its z coordinate is as small as possible.

2. Simplex Algorithm

Step t of the algorithm begins with an $(m-1)$-simplex, as shown in Figure 1, defined by some m points $(A._{j_i}, c_{j_i})$ for $i = (1, \ldots, m)$ and m weights $x^0_{j_i} > 0$ (in the nondegenerate case), such that $\sum A._{j_i} x_{j_i} = b$. In the figure, the vertices of the $m - 1 = 2$ dimensional simplex correspond to $j_1 = 1$, $j_2 = 2$, $j_3 = 3$. The line (b, z) intersects the plane of the simplex (the triangle in the figure) in an interior point (b, z_t). A point $(A._s, c_s)$ is then determined whose vertical distance below this "solution" plane of the simplex is maximal.

Algebraically, the equation $z = \pi y + \pi_0$ of the plane associated with the simplex is found by solving the system of m equations $\pi A._{j_i} + \pi_0 = c_{j_i}$, $j_i = (j_1, \ldots, j_m)$. Next, let $j = s$ be the index of $(A._s, c_s)$, the point most below this plane, namely

$$s = \arg \min_j [c_j - (\pi A._j + \pi_0)]$$

If $[c_s - (\pi A._s + \pi_0)]$ turns out to be non-negative, the iterative process stops. Otherwise, the m-simplex, the tetrahedron in Figure 1, is formed as the convex combination of the point $(A._s, c_s)$ and points lying in the $(m-1)$-simplex. The requirement line (b, z) intersects this m-simplex in a segment $(b, z_{t+1}), (b, z_t)$ where $z_{t+1} < z_t$. The face containing (b, z_{t+1}) is then selected as the new $(m-1)$-simplex. Operationally the point $(A._s, c_s)$ replaces $(A._{j_r}, c_{j_r})$ for some r. The index r is not difficult to determine algebraically.

Geometrical insight as to why the simplex method is efficient can be gained by viewing the algorithm in two dimensions (see Figure 2). Suppose a piecewise linear function $z = f(y)$ is defined as the underbelly of the convex hull of the points $(y^j, z^j) = (A._j, c_j)$. We wish to determine $z = f(b)$ and to find two points $(y^j, z^j), (y^k, z^k)$ and weights $(\lambda, \mu) \geq 0$ on these two points such that $\lambda y^j + \mu y^k = b$, $\lambda + \mu = 1$, $\lambda z^j + \mu z^k = f(b)$. In this two-dimensional case, the simplex method resembles a kind of secant method in which, given any slope σ, it is cheap to find a point (y^s, z^s) of the underbelly such that the slope (actually the slope of a support) at y^s is σ, but in which it is not possible, given b, to find easily the two points (y^t, z^t) and (y^k, z^k) and corresponding weights (λ, μ) for determining $z = f(b)$.

In Figure 2, the algorithm is initiated (in phase II of the simplex method) by two points, say (y^1, z^1) and (y^6, z^6), on opposite sides of the requirement line. The slope of the "solution" line joining them is σ_1. Next, one determines that the point (y^5, z^5) is the one most below the line joining (y^1, z^1) to (y^6, z^6) with slope σ_1. This is done

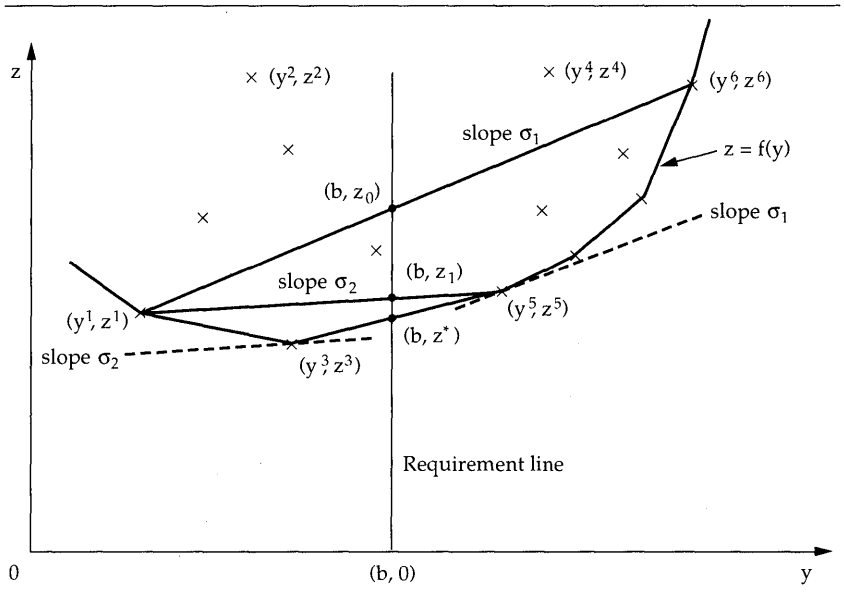

FIGURE 2.
The underbelly of the convex hull, $z = f(y)$.

algebraically by simply substituting the coordinates (y^j, z^j) into the equation of the solution line $z - z^6 = \sigma_1(y - y^6)$ and finding the point $j = s$ such that $\sigma_1(y^j - y^6) - (z^j - z^6)$ is maximal. For the preceding example, $s = 5$ and thus (y^5, z^5) replaces (y^6, z^6). The steps are then repeated with (y^1, z^1) and (y^5, z^5). The algorithm finds the optimum point (b, z^*) in two iterations with the pair (y^3, z^3), (y^5, z^5).

In practical applications, one would expect that most of the points $(A_{\cdot j}, c_j)$ would lie above the underbelly of their convex hull. We would therefore expect that very few j would be extreme points of the underbelly. Because the algorithm only chooses $(A_{\cdot s}, c_s)$ from among the latter, and because these typically would be rare, I conjectured that the algorithm would have very few choices and would take about m steps in practice.

It is not difficult, of course, to construct cases that take more than m iterations, so let me make some remarks about the rate of convergence of z_t to z^*, the minimum value of z, in the event that the method takes more than m iterations.

3. Convergence Rate of the Simplex Method

Assume there exists a constant $1 \geq \overline{\theta} > 0$ such that for every iteration τ, the values of all basic variables $x^\tau_{j_i}$ satisfy

$$1 \geq x^\tau_{j_i} \geq \overline{\theta} > 0 \quad \text{for all } j_i$$

At the start of iteration t, by eliminating the basic variables from the objective equation, we obtain

$$z_{t-1} - z = \sum (-\bar{c}_j^t)x_j$$

where $\bar{c}_{j_i}^t = 0$ for all basic $j = j_i$. If $(-\bar{c}_s^t) = \max(-\bar{c}_j^t) \leq 0$, the iterative process stops with the current basic feasible solution optimal. Otherwise, we increase nonbasic x_s to $x_s = \theta_t \geq \bar{\theta}$ and adjust basic variables to obtain the basic feasible solution to start iteration $t + 1$.

Let $z^* = \min z$ and $x_j = x_j^* \geq 0$ be the corresponding optimal x_j. We define Δ_t as $z_t - z^*$.

Theorem 1. *Independent of the number of variables n,*

$$(\Delta_t/\Delta_0) \leq (1 - \theta_1)(1 - \theta_2) \cdots (1 - \theta_t) \leq e^{-\Sigma\theta_\tau} \leq e^{-\bar{\theta}\cdot t}$$

where $\theta_t \geq \bar{\theta} > 0$ is the value of the incoming basic variable x_s on iteration t.

Proof.

$$\Delta_{t-1} = z_{t-1} - z^* = \sum (-\bar{c}_j^t)x_j^* \leq (-\bar{c}_s^t) \sum x_j^* = (-\bar{c}_s^t)$$

$$\Delta_{t-1} - \Delta_t = z_{t-1} - z_t = (-\bar{c}_s^t)x_s = (-\bar{c}_s^t)\theta_t \geq \Delta_{t-1} \cdot \theta_t$$

where the inequality between the last two terms is obtained by applying the preceding inequality. Rearranging terms and applying t iteratively for $t = 2, 3, \ldots$

$$\Delta_t \leq (1 - \theta_t)\Delta_{t-1} < e^{-\theta_t}\Delta_{t-1} \leq e^{-\bar{\theta}}\Delta_{t-1}$$
$$\leq e^{-\bar{\theta}}(e^{-\bar{\theta}}\Delta_{t-2}) \cdots \leq e^{-\bar{\theta}\cdot t}\Delta_1$$

Corollary 2. *Assuming θ_τ has "on the average" the same average value as any other $x_{j_i}^\tau$, namely $(1/m)$, then the expected number of iterations t required to affect an e^{-k}-fold decrease in Δ_0 will be less than km iterations, that is,*

$$(\Delta_t/\Delta_0) < e^{-\Sigma\theta_\tau} \doteq e^{-t/m}$$

Thus, under the assumption that the value of the incoming variable is $1/m$ on the average, a thousand-fold decrease in $\Delta_t = z_t - z^*$ could be expected to be obtained in less than $7m$ iterations because $e^{-7} < .001$.

It was considerations such as these back in 1947 that led me to believe that the simplex method would be very efficient.

It is fortunate that when the simplex algorithm for solving linear programs was first being developed back in 1947, the column geometry, and not the row geometry, was used. As we have seen, the column geometry suggested a different algorithm, one that promised to be very efficient. Accordingly, I developed a variant of the algorithm without the convexity constraint (4) and arranged in the fall of 1947

to have the Bureau of Standards test it on George Stiegler's nutrition problem [19]. Of course, I soon observed that what appeared in the column geometry to be a new algorithm was, in the row geometry, the vertex descending algorithm that I had rejected earlier.

It is my opinion that any well-trained mathematician viewing the linear programming problem in the row geometry of the variables would have immediately come up with the idea of solving it by a vertex descending algorithm, as did Fourier, de la Vallée Poussin, and Hitchcock before me—each of us proposing it independently of the other. I believe, however, that if anyone had to consider it as a practical method, as I had to, he or she would have quickly rejected it on intuitive grounds as a stupid idea without merit. My own contributions towards the discovery of the simplex method were (1) independently proposing the algorithm, (2) initiating the development of the software necessary for its practical use, and (3) observing (by viewing the problem in the geometry of the columns rather than the rows) that contrary to geometric intuition, following a path on the outside of the convex polyhedron might be an efficient procedure.

4. The Role of Sparsity in the Simplex Method

To determine $s = \arg\min_j [c_j - (\pi A_{\cdot j} + \pi_0)]$ requires forming the scalar product of two vectors π and $A_{\cdot j}$ for each j. This "pricing out" operation, as it is called, is usually cheap because the vectors $A_{\cdot j}$ are sparse; that is, they typically have few nonzero coefficients (perhaps, on the average, four or five nonzeros). Nevertheless, if the number of columns n is large, say several thousand, pricing can use up a lot of time. (Parallel processors could be used very effectively for pricing by assigning subsets of the columns to different processors [6].)

In single processors, various *partial pricing* schemes are used. One scheme, used in the MINOS software system, is to partition the columns into subsets of some k columns each [17]. The choice of s is restricted to columns that price out negative among the first k until there are none, and then moving on to the next k, and so forth. Another scheme used is to price out all the columns and rank them according to how negative they price out. A subset of j, say the 50 most negative in rank, are then used to iteratively select s until this subset no longer has a column that prices out negative. A new subset is then generated for selecting s, and the process is repeated. Partial pricing schemes are very effective when n is large, especially for matrix structures that contain so-called "GUB" (Generalized Upper Bound) rows [7].

Besides the pricing-out of the columns, the simplex method requires that the current basis B, that is, the columns $(j_1, \ldots, j_m)$ used to form the simplex in Figure 1, be maintained from iteration t to $t + 1$ in a form that makes it easy to compute two vectors v and π, where

$Bv = A_{\cdot s}$ and $\pi B = (c_{j_1}, \ldots, c_{j_m})$. The matrix B is typically sparse. In problems where the number of rows m is greater than 1000, the percent of nonzeros may be less than 1/2 of 1 percent. Even for such B, it is not practical to maintain B^{-1} explicitly, because it could turn out to be 100 percent dense. Instead, B is often represented as the product of a lower and upper triangular matrix in which each is maintained as a product of elementary matrices, with every effort being made to keep the nonunit column of each elementary matrix as sparse as possible. Maintaining this sparsity is important, because otherwise, for the case of $m = 1000$ the algorithm would have to manipulate data sets with millions of nonzero numbers. Solving systems $Bv = A_{\cdot s}$ in order to determine which variable leaves the basis would become too costly.

5. The Role of Near-Triangularity of the Basis

The success of the simplex method in solving very large problems encountered in practice depends on two properties found in almost every practical problem. First, the basis is usually sparse. Second, one can usually rearrange the rows and columns of the various bases encountered in the course of solution so that they are nearly triangular. Near-triangularity makes it relatively inexpensive to represent it as a product of a lower and upper triangular matrix and to preserve much of the original sparsity.

Even if the bases were sparse but not nearly triangular, solving systems $Bv = A_{\cdot s}$ could be too costly to perform.

The success of solving linear programming therefore depends on a number of factors: (1) the power of computers, (2) extremely clever algorithms, and most of all (3) a lot of good luck that the matrices of practical problems will be sparse and that their bases, after rearrangement, will be nearly triangular.

For over 40 years the simplex method has reigned supreme as the preferred method for solving linear programs. Its efficiency is the historical reason for the practical success of the field. As of this writing, however, the algorithm is being challenged by new interior methods proposed by N. Karmarkar [14] and others and by methods that exploit special structure. If these new methods turn out to be more successful than the simplex method for solving practical linear programs, it will be not because of any theoretical reasons having to do with polynomial time for solving worst-case general linear programs, but because they can more effectively exploit the sparsity and near triangularity of practical problems than the simplex method is able to do.

References

[1] R. Coughlin and D. E. Zitarelli, *The Ascent of Mathematics* (New York: McGraw-Hill, 1984).

[2] G. B. Dantzig, "Linear Programming," in *Problems for the Numerical Analysis of the Future*, Proceedings of the Symposium on Modern Calculating Machinery and Numerical Methods, UCLA (July 29–31, 1948); *Appl. Math.* 15, National Bureau of Standards (1951): 18–21.

[3] G. B. Dantzig, *Linear Programming and Extensions* (Princeton University Press, 1963).

[4] G. B. Dantzig, "Reminiscences about the Origins of Linear Programming," in *Math. Programming*, R. W. Cottle, M.L. Kelmanson, and B. Korte (eds.), Proceedings of the International Congress on Mathematical Programming, Rio de Janeiro (1984): 105–12.

[5] G. B. Dantzig, "Impact of Linear Programming on Computer Development," *OR/MS Today* 14 (1988): 12–17.

[6] G. B. Dantzig, "Planning Under Uncertainty Using Parallel Computing," *Ann. Oper. Res.* 14 (1988): 1–16.

[7] G. B. Dantzig and R. M. Van Slyke, "Generalized Upper Bounding Techniques," *J. Comp. System Sci.* 1 (1967): 213–26.

[8] G. B. Dantzig and A. Wald, "On the Fundamental Lemma of Neyman and Pearson," *Ann. Math. Stat.* 22 (1951): 87–93.

[9] Ch. de la Vallée Poussin, "Sur la Methode de l'Approximation Minimum," *Ann Soc. Sci. Bruxelles Ser. I* 35 (1910–1911): 1–16.

[10] J. B. J. Fourier, "Solution d'une question particuliere du calcul des inegalities," original 1826 paper with an abstract of an 1824 paper reprinted in *Oeuvres de Fourier*, Tome II (Olms: Hildesheim, 1970), 317–19.

[11] F. L. Hitchcock, "The Distribution of a Product from Several Sources to Numerous Localities," *J. Math. Phys.* 20 (1941): 224–30.

[12] A. J. Hoffman, M. Mannos, D. Sokolousky, and N. Wiegmann, "Computational Experience in Solving Linear Programs," *SIAM J.* 1 (1953): 17–33.

[13] L. V. Kantorovich, *Mathematical Methods in the Organization and Planning of Production* (Publication House of the Leningrad State University, 1939). Translated in *Management Sci.* 6 (1960): 366–422.

[14] N. Karmarkar, "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica* 4 (1984): 373–95.

[15] T. C. Koopmans (ed.), *Activity Analysis of Production and Allocation* (New York: John Wiley & Sons, 1951).

[16] T. S. Motzkin, "Beitrage zur Theorie der Linearen Ungleichungen" (Jerusalem: Doctoral Thesis, University of Zurich, 1936).

[17] B. A. Murtagh and M. A. Saunders, "MINOS 5.0 User's Guide," Technical Report SOL 83–20, Systems Optimization Laboratory, Department of Operations Research, Stanford University (1983).

[18] J. Neyman and F. S. Pearson, "Contributions to the Theory of Testing Statistical Hypotheses," *Statist. Res. Mem.*, I and II (1936, 1938).

[19] G. J. Stigler, "The Cost of Subsistence," *J. of Farm Econ.* 27 (1945): 303–14.

[20] J. von Neumann, "Zur Theorie de Gesellschaftsspiele," *Math. Ann.* 100 (1928): 295–320. Translated by Sonya Bargmann in *Contributions to the Theory of Games*, Vol IV, A.W. Tucker and R.D. Luce (eds.), *Ann. of Math. Stud.* 40 (Princeton University Press, (1959): 13–42.