The Evolution of Computationally Practical Linear Programming

Robert Fourer

President, AMPL Optimization Inc.
Professor Emeritus, Northwestern University

4er@ampl.com — +1 773-336-2675

AFOR 2017: International Conference on Advancing Frontiers in Operational Research: Towards a Sustainable World

Kolkata, India — 21-23 December 2017

The Evolution of Computationally Practical Linear Programming

Every student of Operations Research learns that linear programs can be solved efficiently. The fascinating story of *how* they came to be solved efficiently is little known, however.

We begin with an account of how a practicable primal simplex method for linear programming was first successfully implemented on primitive computers, by Dantzig, Orchard-Hays and others at the RAND Corporation in the early 1950s. Combining mathematics and engineering, success emerged from a series of innovative reorganizations of the computational steps. One of the key ideas arose unexpectedly as the by-product of an ill-advised plan for avoiding degenerate cycling.

At around the same time, early experts in linear programming developed a compact "tableau" scheme for carrying out the computations as a series of "pivots" on a matrix. We consider how the tableau form, impractical for computer implementations, was adopted by almost all textbooks, while the computationally practical form of the simplex method remained obscure.

The remainder of the presentation considers how computational practice in linear programming has continued to evolve to the present day. In fact the primal simplex method has become rarely used. We consider the new interior-point methods, which can outperform that simplex method particularly on very large problems; they also went through a period of innovations and false steps before being made practical. Also we consider how the dual simplex method, long considered only a curiosity, has become the preferred method of linear programming solvers today.

1948

Programming of Interdependent Activities II: Mathematical Model

- George B. Dantzig
- *Econometrica* **17** (1949)

"Linear Programming"

- Formulations & applications
- No algorithm

"It is proposed to solve linear programming problems . . . by means of large scale digital computers . . . . Several computational procedures have been evolved so far and research is continuing actively in this field."

PROGRAMMING OF INTERDEPENDENT ACTIVITIES
II MATHEMATICAL MODEL[1]

By George B. Dantzig

Activities (or production processes) are considered as building blocks out of which a technology is constructed. Postulates are developed by which activities may be combined. The main part of the paper is concerned with the discrete type model and the use of a linear maximization function for finding the "optimum" program. The mathematical problem associated with this approach is developed first in general notation and then in terms of a dynamic system of equations expressed in matrix notation. Typical problems from the fields of inter-industry relations, transportation, nutrition, warehouse storage, and air transport are given in the last section.

Introduction

The multitude of activities in which a large organization or a nation engages can be viewed not only as fixed objects but as representative building blocks of different kinds that might be recombined in varying amounts to form new blocks. If a structure can be reared of these blocks that is mutually self-supporting, the resulting edifice can be thought of as a technology. Usually the very elementary blocks have a wide variety of forms and quite irregular characteristics over time. Often they are combined with other blocks so that they will have "nicer" characteristics when used to build a complete system. Thus the science of programming, if it may be called a science, is concerned with the adjustment of the levels of a set of given activities (production processes) so that they remain mutually consistent and satisfy certain optimum properties.

It is highly desirable to have formal rules by which activities can be combined to form composite activities and an economy. These rules are set forth here as a set of postulates regarding reality. Naturally other postulates are possible; those selected have been chosen with a wide class of applications in mind and with regard to the limitations of present day computational techniques. The reader's attention is drawn to the last section of this report where a number of applications of the mathematical model are discussed. These are believed to be of sufficient interest in themselves, and may lend concreteness to the development which follows:

Postulates of a Linear Technology

Postulate I: *There exists a set* $\{A\}$ *of activities.*
Postulate II: *All activities take place within a time span 0 to* t_0.

[1] A revision of a paper presented before the Madison Meeting of the Econometric Society on September 9, 1948. This is the second of two papers on this subject, both appearing in this issue. The first paper, with sub-title "General Discussion," will be referred to by Roman numeral I.

200

1949

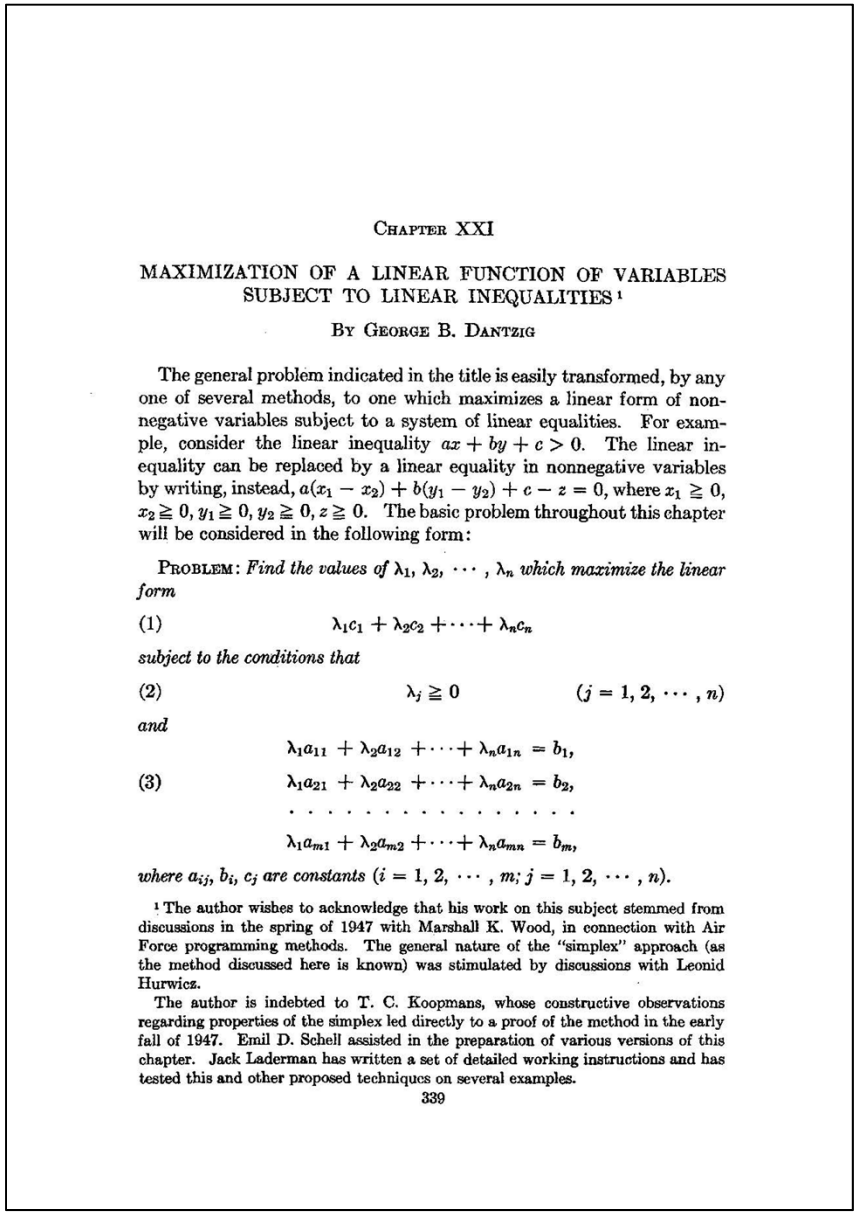
Maximization of a Linear Function of Variables Subject to Linear Inequalities

- George B. Dantzig
- *Activity Analysis of Production and Allocation* (1951)

"Simplex Method"

- Proof of convergence
- No computers

"As a practical computing matter the iterative procedure of shifting from one basis to the next is not as laborious as would first appear . . ."



CHAPTER XXI

MAXIMIZATION OF A LINEAR FUNCTION OF VARIABLES SUBJECT TO LINEAR INEQUALITIES[1]

BY GEORGE B. DANTZIG

The general problem indicated in the title is easily transformed, by any one of several methods, to one which maximizes a linear form of non-negative variables subject to a system of linear equalities. For example, consider the linear inequality $ax + by + c > 0$. The linear inequality can be replaced by a linear equality in nonnegative variables by writing, instead, $a(x_1 - x_2) + b(y_1 - y_2) + c - z = 0$, where $x_1 \geqq 0$, $x_2 \geqq 0$, $y_1 \geqq 0$, $y_2 \geqq 0$, $z \geqq 0$. The basic problem throughout this chapter will be considered in the following form:

PROBLEM: *Find the values of* $\lambda_1, \lambda_2, \cdots, \lambda_n$ *which maximize the linear form*

$$(1) \qquad \lambda_1 c_1 + \lambda_2 c_2 + \cdots + \lambda_n c_n$$

subject to the conditions that

$$(2) \qquad \lambda_j \geqq 0 \qquad (j = 1, 2, \cdots, n)$$

and

$$(3) \qquad \begin{aligned} \lambda_1 a_{11} + \lambda_2 a_{12} + \cdots + \lambda_n a_{1n} &= b_1, \\ \lambda_1 a_{21} + \lambda_2 a_{22} + \cdots + \lambda_n a_{2n} &= b_2, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots & \\ \lambda_1 a_{m1} + \lambda_2 a_{m2} + \cdots + \lambda_n a_{mn} &= b_m, \end{aligned}$$

where a_{ij}, b_i, c_j *are constants* $(i = 1, 2, \cdots, m; j = 1, 2, \cdots, n)$.

[1] The author wishes to acknowledge that his work on this subject stemmed from discussions in the spring of 1947 with Marshall K. Wood, in connection with Air Force programming methods. The general nature of the "simplex" approach (as the method discussed here is known) was stimulated by discussions with Leonid Hurwicz.

The author is indebted to T. C. Koopmans, whose constructive observations regarding properties of the simplex led directly to a proof of the method in the early fall of 1947. Emil D. Schell assisted in the preparation of various versions of this chapter. Jack Laderman has written a set of detailed working instructions and has tested this and other proposed techniques on several examples.

339

1953

An Introduction to Linear Programming

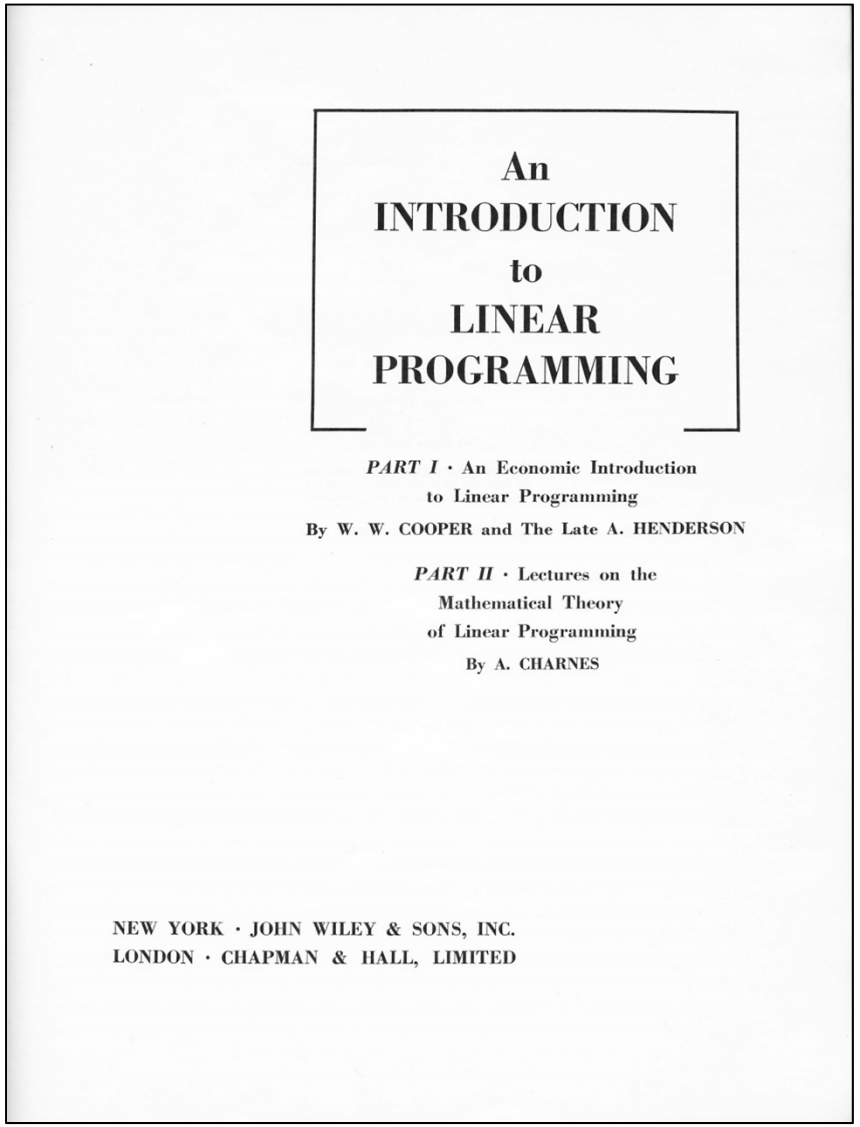
- W.W. Cooper, A. Henderson
- A. Charnes

"Simplex Tableau"

- Symbolic description
- Numerical example

"As far as computations are concerned it is most convenient to arrange the data at each stage in a 'simplex tableau' as shown in Table I.[12]"

"[12]A. Orden suggested this efficient arrangement developed by himself, Dantzig, and Hoffman."



An
INTRODUCTION
to
LINEAR
PROGRAMMING

PART I · An Economic Introduction to Linear Programming
By W. W. COOPER and The Late A. HENDERSON

PART II · Lectures on the Mathematical Theory of Linear Programming
By A. CHARNES

NEW YORK · JOHN WILEY & SONS, INC.
LONDON · CHAPMAN & HALL, LIMITED

Terminology

Linear program

Minimize $\boldsymbol{c} \cdot \boldsymbol{x}$

Subject to $A\,\boldsymbol{x} = \boldsymbol{b}$

$\boldsymbol{x} \geq \boldsymbol{0}$

m constraints on n variables: $m < n$

Data

$\boldsymbol{b} = (b_1, \ldots, b_m)$

$\boldsymbol{c} = (c_1, \ldots, c_n)$

$A = [a_{ij}]$, with m rows $\boldsymbol{a^i}$ and n columns $\boldsymbol{a_j}$

Variables

$\boldsymbol{x} = (x_1, \ldots, x_n)$

Terminology *(cont'd)*

Basis

- $\mathcal{B}$, $\mathcal{N}$, sets of basic and nonbasic column indices
  - $|\mathcal{B}| = m$, $|\mathcal{N}| = n - m$
- $\boldsymbol{c}_{\mathcal{B}}$, $\boldsymbol{x}_{\mathcal{B}}$, corresponding subvectors of $\boldsymbol{c}$, $\boldsymbol{x}$

Basis matrix

- B, nonsingular $|\mathcal{B}| \times |\mathcal{B}|$ submatrix of A
- $B^{-1} = [z_{ij}]$, with $|\mathcal{B}|$ rows $\boldsymbol{z}^i$ and $|\mathcal{B}|$ columns $\boldsymbol{z}_j$

Tableau Simplex Method

Given a tableau of values

$y_{ij},\ i \in \mathcal{B},\ j \in \mathcal{N}$: the transformed columns $\boldsymbol{y}_j = \boldsymbol{B}^{-1}\boldsymbol{a}_j$

$y_{i0} \equiv x_i,\ i \in \mathcal{B}$ (the basic solution)

$y_{0j} \equiv d_j,\ j \in \mathcal{N}$ (the reduced costs)

Choose an entering variable

$p \in \mathcal{N}$: $d_p < 0$

Choose a leaving variable

$q \in \mathcal{B}$: $x_q/y_{qp} = \min_{y_{ip}>0} x_i/y_{ip}$

Update

- One "pivot" step on the tableau

Disadvantages

Computational inefficiency

- $|\mathcal{B}| \times |\mathcal{N}| = m(n-m)$ additions & multiplications
- $|\mathcal{B}| \times |\mathcal{N}|$ numbers to write and store

Numerical instability

- Rigid computational rules

1953

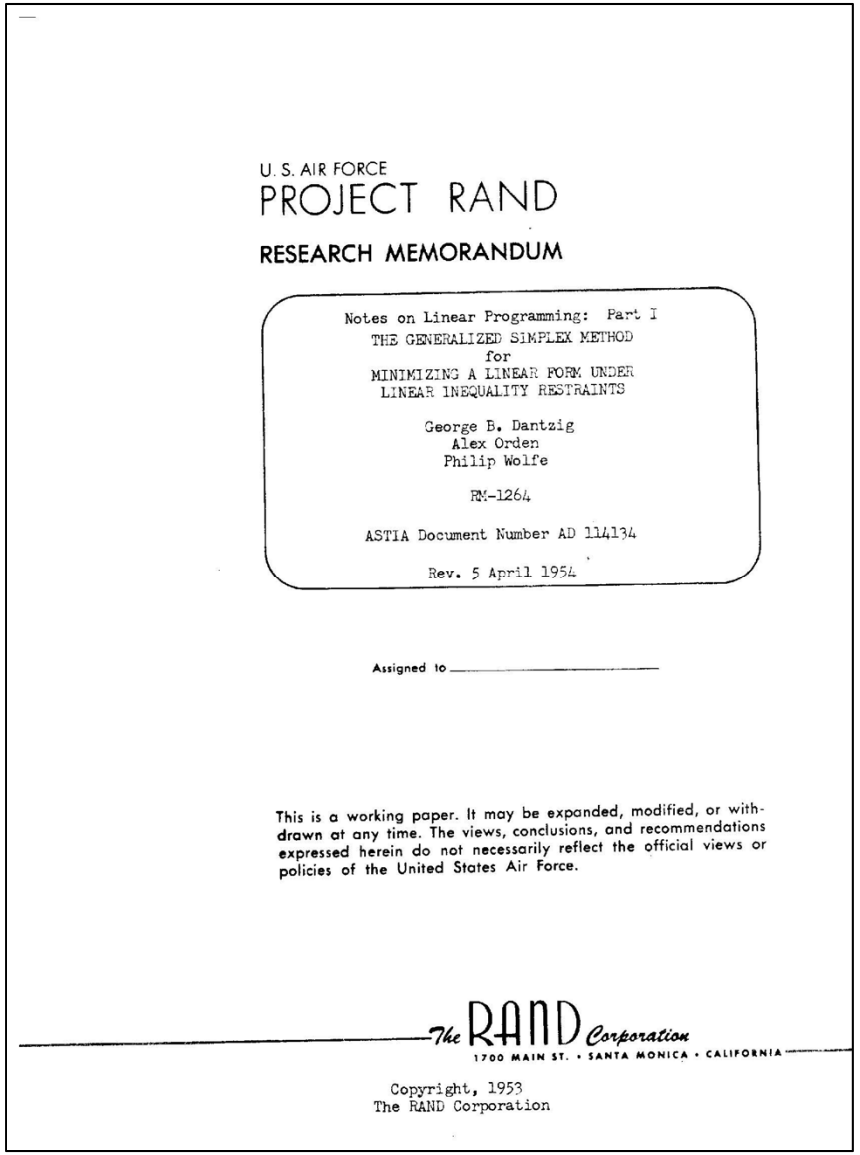
The Generalized Simplex Method for Minimizing a Linear Form under Linear Inequality Restraints

- George B. Dantzig, Alex Orden, Philip Wolfe
- *Project RAND Research Memorandum RM-1264*

"Lexicographic Simplex Method"

- Prevent degenerate cycling
- Reorganize computations

"The k+1st iterate is closely related to the kth by simple transformations that constitute the computational algorithm [6], . . ."



U. S. AIR FORCE
PROJECT RAND

RESEARCH MEMORANDUM

Notes on Linear Programming: Part I
THE GENERALIZED SIMPLEX METHOD
for
MINIMIZING A LINEAR FORM UNDER
LINEAR INEQUALITY RESTRAINTS

George B. Dantzig
Alex Orden
Philip Wolfe

RM-1264

ASTIA Document Number AD 114134

Rev. 5 April 1954

Assigned to _____

This is a working paper. It may be expanded, modified, or withdrawn at any time. The views, conclusions, and recommendations expressed herein do not necessarily reflect the official views or policies of the United States Air Force.

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA

1953

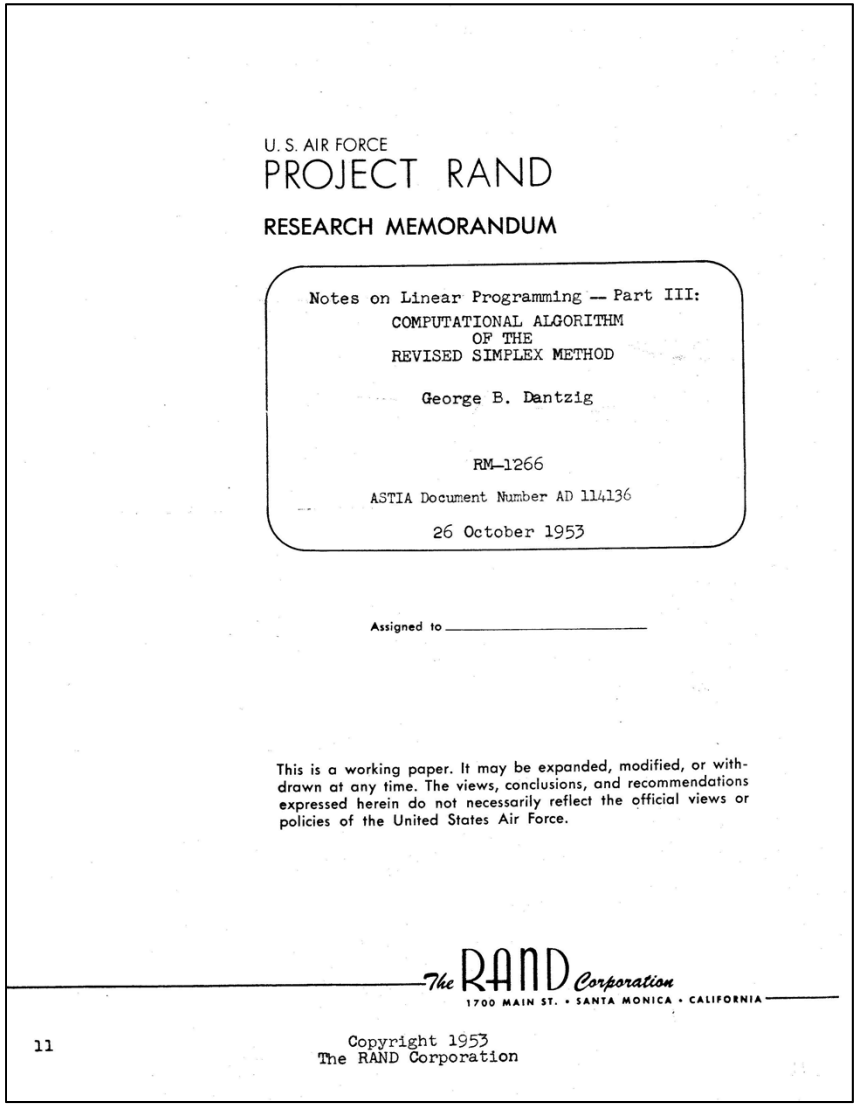
Computational Algorithm of the Revised Simplex Method

- George B. Dantzig
- *Project RAND Research Memorandum RM-1266*

"Revised Simplex Method"

- Break ties for leaving variable
- Update basis inverse

"The transformation of just the inverse (rather than the entire matrix of coefficients with each cycle) has been developed because it has several important advantages over the old method: . . ."



U. S. AIR FORCE
PROJECT RAND
RESEARCH MEMORANDUM

Notes on Linear Programming — Part III:
COMPUTATIONAL ALGORITHM
OF THE
REVISED SIMPLEX METHOD

George B. Dantzig

RM-1266

ASTIA Document Number AD 114136

26 October 1953

Assigned to _____

This is a working paper. It may be expanded, modified, or withdrawn at any time. The views, conclusions, and recommendations expressed herein do not necessarily reflect the official views or policies of the United States Air Force.

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA

11

Revised Simplex Method

Given a matrix of inverse values

$z_{ij},\ i \in \mathcal{B},\ j \in \mathcal{B}$: the basis inverse B^{-1}

$z_{i0} \equiv x_i,\ i \in \mathcal{B}$ (the basic solution)

$z_{0i} \equiv \pi_i,\ i \in \mathcal{B}$ (the dual prices)

Choose an entering variable

$p \in \mathcal{N}$: $d_p = c_p - \boldsymbol{\pi} \cdot \boldsymbol{a_p} < 0$

Choose a leaving variable

$y_{ip} = \boldsymbol{z^i} \cdot \boldsymbol{a_p}$

$q \in B$: $x_q / y_{qp} = \min_{y_{ip}>0} x_i / y_{ip}$

Update

- One "pivot" step on the inverse

Advantages

Smaller tableau update

". . . In the original method (roughly) $m \times n$ new elements have to be recorded each time. In contrast, the revised method (by making extensive use of cumulative sums of products) requires the recording of about m^2 elements . . . ."

Sparse operations

"In most practical problems the original matrix of coefficients is largely composed of zero elements. . . . The revised method works with the matrix in its original form and takes direct advantage of these zeros."

$$d_p = c_p - \boldsymbol{\pi} \cdot \boldsymbol{a_p}$$
$$y_{ip} = \boldsymbol{z^i} \cdot \boldsymbol{a_p}$$

Disadvantages

Inefficiency

- $|\mathcal{B}| \times |\mathcal{B}| = m^2$ additions & multiplications
- $|\mathcal{B}| \times |\mathcal{B}|$ numbers to write and store

Instability

- Rigid computational rules

However . . .

". . . the revised method (by making extensive use of cumulative sums of products) requires the recording of about m^2 elements (and an alternative method [5] can reduce this to m . . .)."

1953

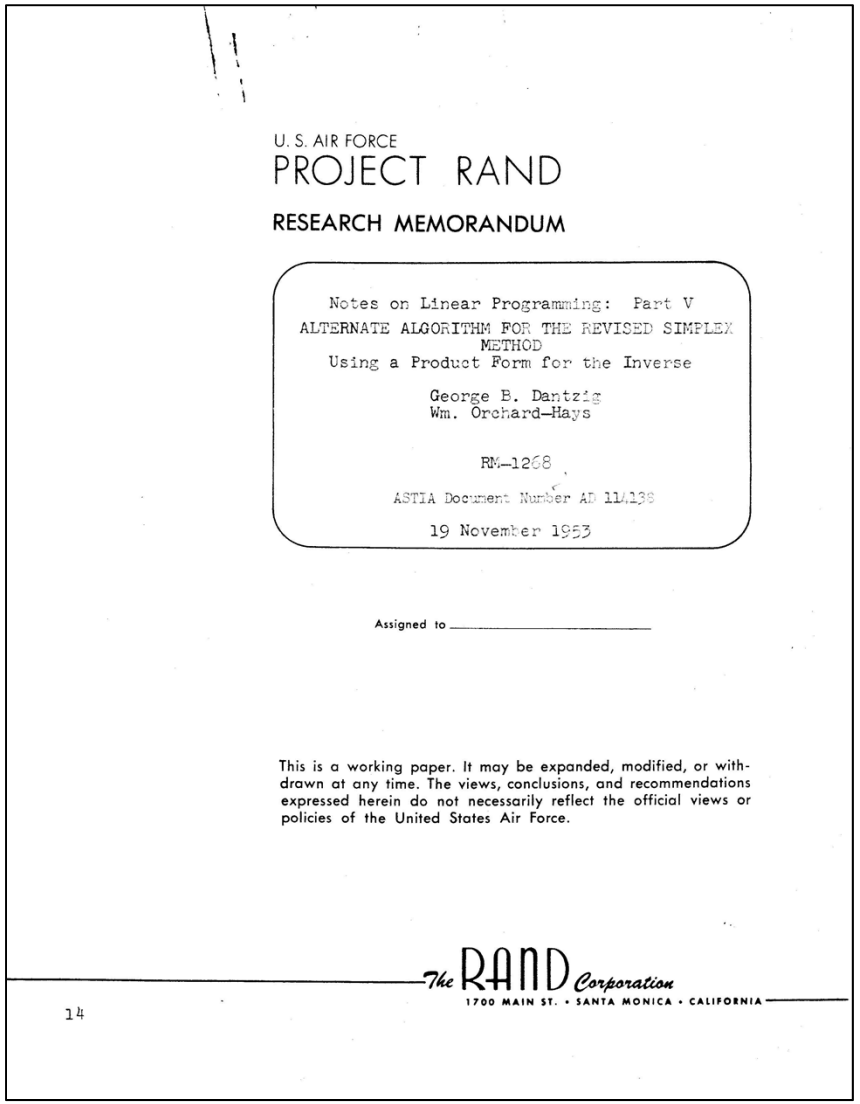
Alternate Algorithm for the Revised Simplex Method

- George B. Dantzig, Wm. Orchard-Hays
- *Project RAND Research Memorandum RM-1268*

"Product Form for the Inverse"

- Fully sparse representation
- Practical computation

"Using the I.B.M. Card Programmed Calculator, . . . where the inverse matrix is needed at one stage and its transpose at another, this is achieved simply by turning over the deck of cards representing the inverse."



U. S. AIR FORCE
PROJECT RAND
RESEARCH MEMORANDUM

Notes on Linear Programming: Part V
ALTERNATE ALGORITHM FOR THE REVISED SIMPLEX METHOD
Using a Product Form for the Inverse

George B. Dantzig
Wm. Orchard-Hays

RM-1268

ASTIA Document Number AD 114238

19 November 1953

Assigned to _____

This is a working paper. It may be expanded, modified, or withdrawn at any time. The views, conclusions, and recommendations expressed herein do not necessarily reflect the official views or policies of the United States Air Force.

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA

14

Product-Form Simplex Method

Given

$\boldsymbol{x}_{\mathcal{B}}$ (the basic solution)

$B^{-1} = E_k^{-1} E_{k-1}^{-1} \cdots E_2^{-1} E_1^{-1}$ (factorization of the basis inverse)

Choose an entering variable

$\boldsymbol{\pi} = \boldsymbol{c}_{\mathcal{B}} E_k^{-1} E_{k-1}^{-1} \cdots E_2^{-1} E_1^{-1}$

$p \in \mathcal{N}$: $c_p - \boldsymbol{\pi} \cdot \boldsymbol{a}_p < 0$

Choose a leaving variable

$\boldsymbol{y}_p = E_k^{-1} E_{k-1}^{-1} \cdots E_2^{-1} E_1^{-1} \boldsymbol{a}_p$

$q \in \mathcal{B}$: $x_q / y_{qp} = \min_{y_{ip} > 0} x_i / y_{ip}$

Update

- add a factor E_{k+1}^{-1} derived from $\boldsymbol{y}_p$
- update basic solution to $\boldsymbol{x}_{\mathcal{B}} - (x_q / y_{qp})\, \boldsymbol{y}_p$

Factorization of the Inverse

Form of the factors

- E_i is an identity matrix except for one column
- . . . and so is E_i^{-1}

1	0	0	1.7	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	0.5	0	0
0	0	0	3.4	1	0
0	0	0	0	0	1

Storage of the factors

- nonzeros only of the one column, in (row,value) pairs
- diagonal element first

Update of the factors

- E_{k+1} is an identity matrix except for $\boldsymbol{y}_p$ in column q

Practical Simplex Method

Given

$\boldsymbol{x}_{\mathcal{B}}$ (the basic solution)

a factorization of B suitable for computation

Choose an entering variable

solve $B^{\boldsymbol{T}}\boldsymbol{\pi} = \boldsymbol{c}_{\mathcal{B}}$

$p \in \mathcal{N}$: $c_p - \boldsymbol{\pi} \cdot \boldsymbol{a_p} < 0$

Choose a leaving variable

solve $B\boldsymbol{y}_p = \boldsymbol{a}_p$

$q \in \mathcal{B}$: $x_q / y_{qp} = \min_{y_{ip}>0} x_i / y_{ip}$

Update

- update factorization to reflect change of basis
- update basic solution to $\boldsymbol{x}_{\mathcal{B}} - (x_q / y_{qp})\, \boldsymbol{y_p}$

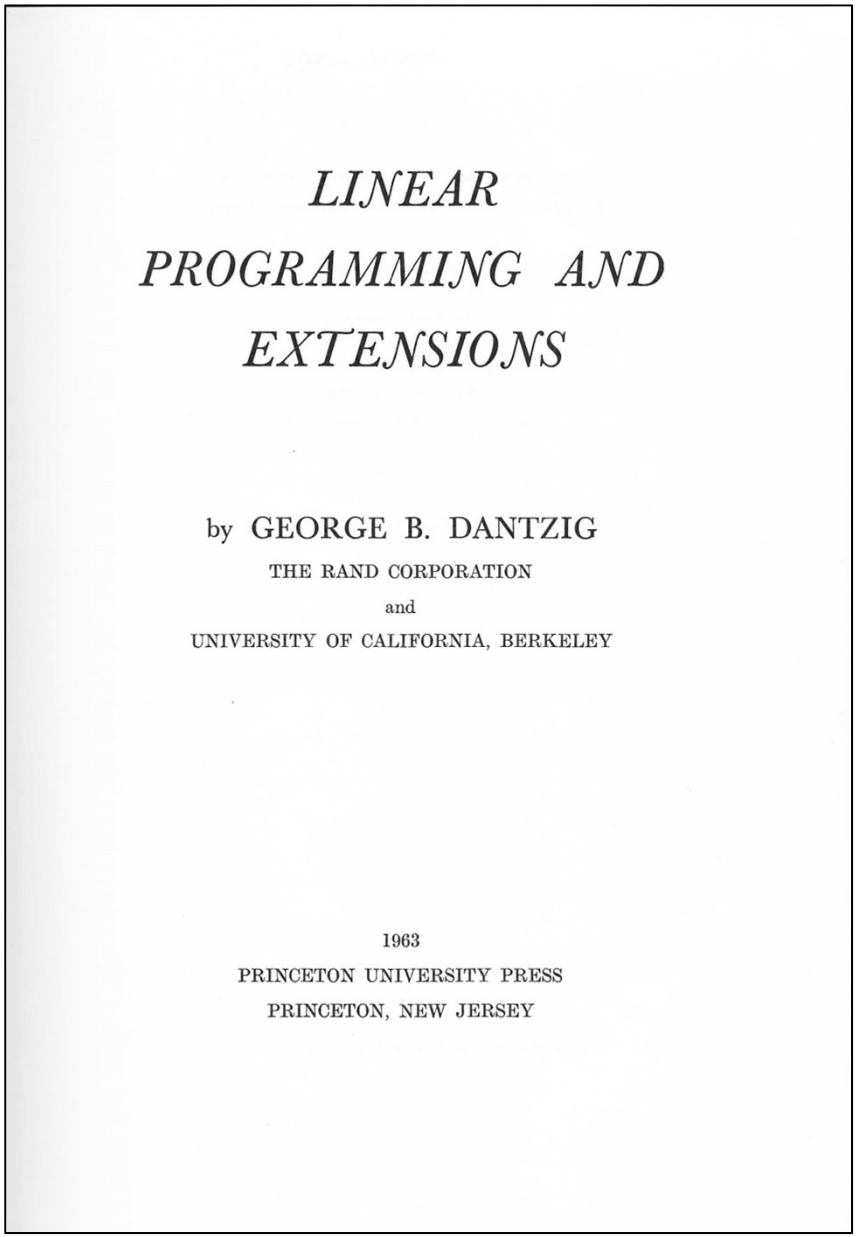
1963

Linear Programming and Extensions

- George B. Dantzig

"... the *simplex algorithm* ... starts with a canonical form, consists of a sequence of pivot operations, and forms the main *subroutine* of the simplex method."

"Because some readers might find that the matrix notation of §8.5 [The Simplex Algorithm in Matrix Form] obscures the computational aspects, we have tended to avoid its use here."



LINEAR PROGRAMMING AND EXTENSIONS

by GEORGE B. DANTZIG

THE RAND CORPORATION

and

UNIVERSITY OF CALIFORNIA, BERKELEY

1963

PRINCETON UNIVERSITY PRESS

PRINCETON, NEW JERSEY

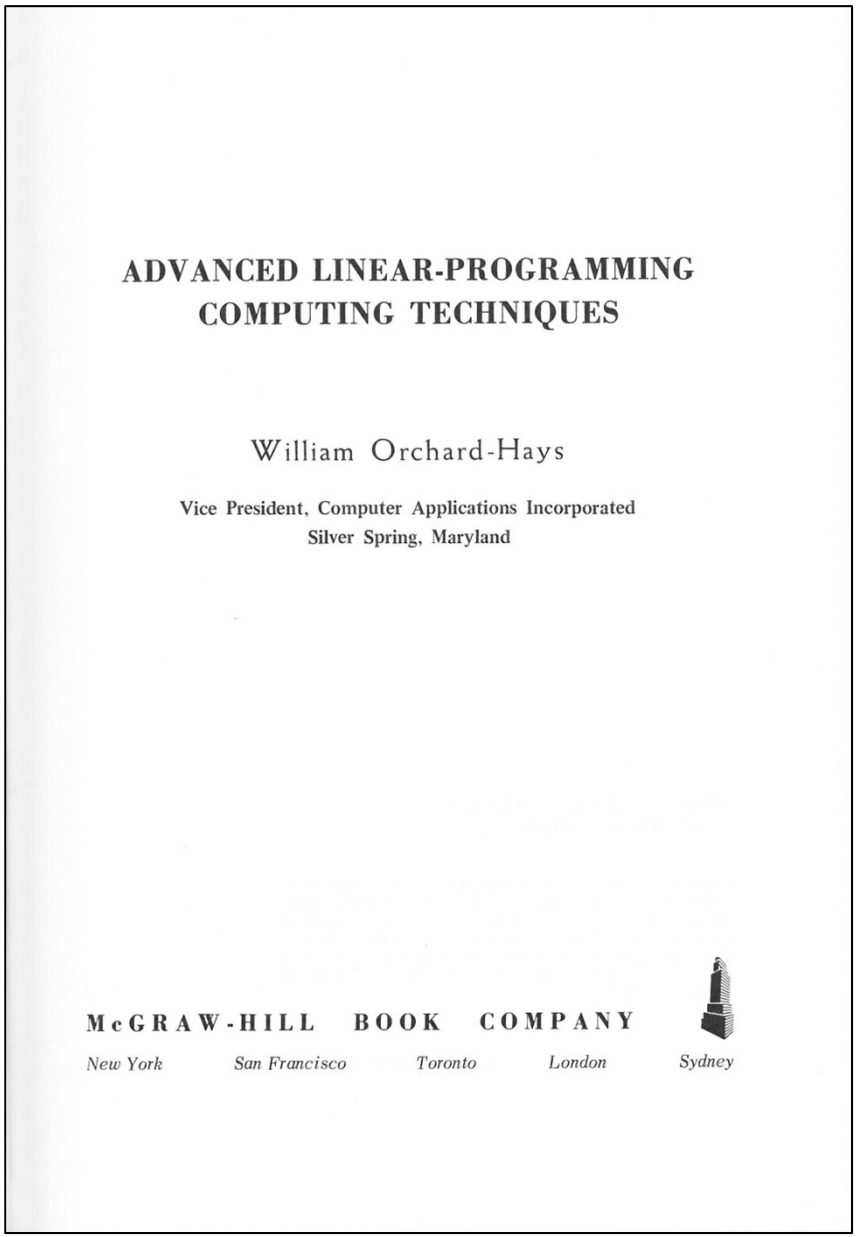
1968

Advanced Linear-Programming Computing Techniques

- William Orchard-Hays

"Except for [a few sections], the contents of the book reflect actual and extensive experience."

"I hope that the many users of mathematical programming systems implemented on today's large computers find the book valuable as background for the largely undocumented algorithms embedded in these systems. If it should also be found useful as a course text, all objectives will have been achieved."



ADVANCED LINEAR-PROGRAMMING
COMPUTING TECHNIQUES

William Orchard-Hays

Vice President, Computer Applications Incorporated
Silver Spring, Maryland

McGRAW-HILL BOOK COMPANY

New York San Francisco Toronto London Sydney

Tableau Simplex *Revisited*

Simple

- No linear algebra
- No matrices & inverses
- All computations in one "pivot" step
- Easy to set up for hand calculation

Familiar

- Professors learned it
- Textbooks use it
- Proofs use it

But not inevitable . . .

Essential Simplex Method

Given

$\boldsymbol{x}_{\mathcal{B}}$ (the basic solution)

B (the basis)

Choose an entering variable

solve $B^{\boldsymbol{T}}\boldsymbol{\pi} = \boldsymbol{c}_{\mathcal{B}}$

$p \in \mathcal{N}$: $c_p - \boldsymbol{\pi} \cdot \boldsymbol{a_p} < 0$

Choose a leaving variable

solve $B\boldsymbol{y}_p = \boldsymbol{a}_p$

$q \in \mathcal{B}$: $x_q / y_{qp} = \min_{y_{ip}>0} x_i / y_{ip}$

Update

- update basic solution to $\boldsymbol{x}_{\mathcal{B}} - (x_q / y_{qp})\, \boldsymbol{y_p}$

Essential Simplex Method

Course notes for use in teaching

- Optimization Methods I:
  Solving Linear Programs by the Simplex Method
- http://www.4er.org/CourseNotes

1978

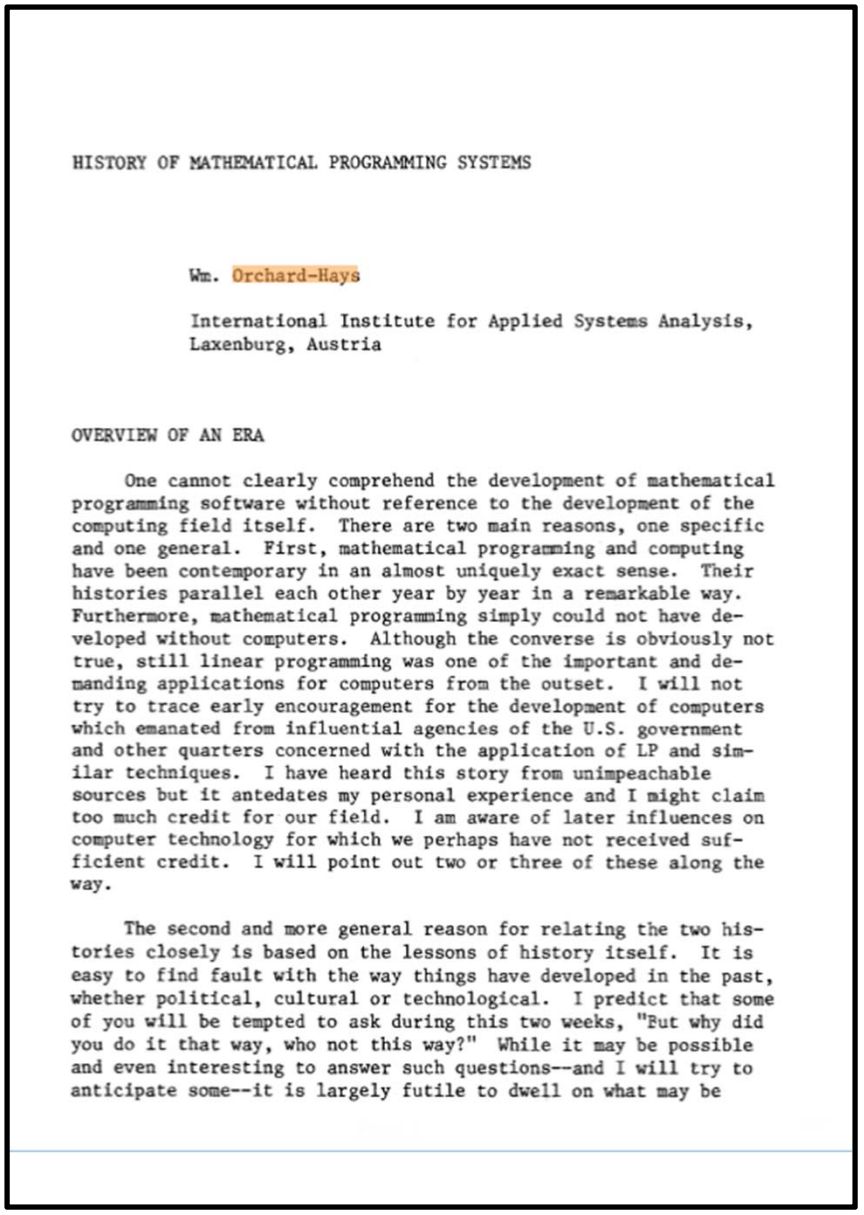
History of Mathematical Programming Systems

- William Orchard-Hays
- *Design and Implementation of Optimization Software*, H.J. Greenberg, ed.

"Overview of an Era"

- Better implementations
- More powerful computers

"One cannot clearly comprehend the development of mathematical programming software without reference to the development of the computing field itself."



HISTORY OF MATHEMATICAL PROGRAMMING SYSTEMS

Wm. Orchard-Hays

International Institute for Applied Systems Analysis,
Laxenburg, Austria

OVERVIEW OF AN ERA

One cannot clearly comprehend the development of mathematical programming software without reference to the development of the computing field itself. There are two main reasons, one specific and one general. First, mathematical programming and computing have been contemporary in an almost uniquely exact sense. Their histories parallel each other year by year in a remarkable way. Furthermore, mathematical programming simply could not have developed without computers. Although the converse is obviously not true, still linear programming was one of the important and demanding applications for computers from the outset. I will not try to trace early encouragement for the development of computers which emanated from influential agencies of the U.S. government and other quarters concerned with the application of LP and similar techniques. I have heard this story from unimpeachable sources but it antedates my personal experience and I might claim too much credit for our field. I am aware of later influences on computer technology for which we perhaps have not received sufficient credit. I will point out two or three of these along the way.

The second and more general reason for relating the two histories closely is based on the lessons of history itself. It is easy to find fault with the way things have developed in the past, whether political, cultural or technological. I predict that some of you will be tempted to ask during this two weeks, "But why did you do it that way, who not this way?" While it may be possible and even interesting to answer such questions--and I will try to anticipate some--it is largely futile to dwell on what may be

1978

History of Mathematical Programming Systems

- William Orchard-Hays
- *Design and Implementation of Optimization Software*

First, mathematical programming and computing have been contemporary in an almost uniquely exact sense. Their histories parallel each other year by year in a remarkable way.

Furthermore, mathematical programming simply could not have developed without computers. Although the converse is obviously not true, still linear programming was one of the important and demanding applications for computers from the outset.

The quarter century from the late 1940s to the early 1970s constituted an era, one of the most dynamic in the history of mankind. Among the many technological developments of that period — and indeed of any period — the computing field has been the most virulent and astounding.

. . . the nature of the computing industry, profession, and technology has by now been determined — all their essential features have existed for perhaps five years. One hopes that some of the more recent developments will be applied more widely and effectively but the technology that now exists is pretty much what will exist, leaving aside a few finishing touches to areas already well developed, such as minicomputers and networks.

Further Evolution of Computationally Practical Linear Programming

Robert Fourer

President, AMPL Optimization Inc.
Professor Emeritus, Northwestern University

4er@ampl.com — +1 773-336-2675

AFOR 2017: International Conference on Advancing Frontiers in Operational Research: Towards a Sustainable World

Kolkata, India — 21-23 December 2017

1984

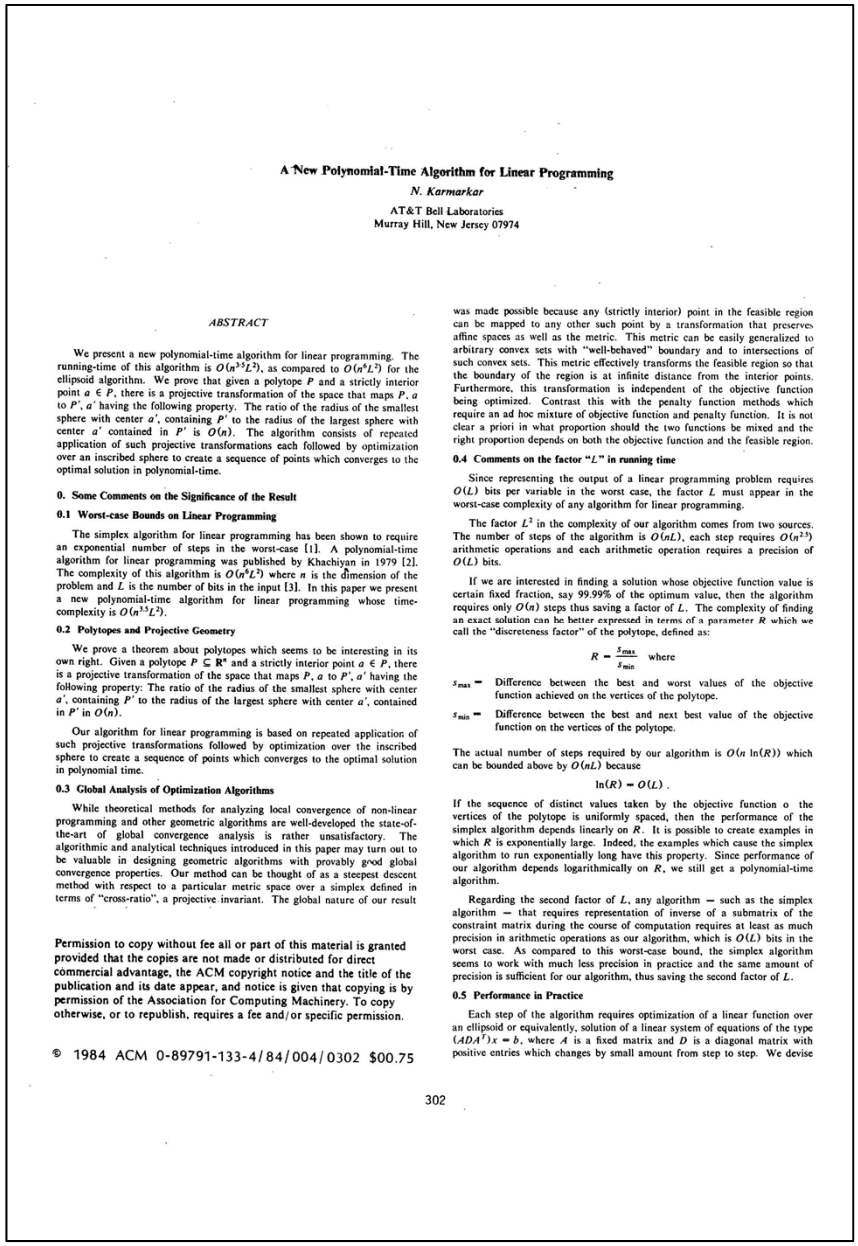
A New Polynomial-Time Algorithm for Linear Programming

- N. Karmarkar
- *Proceedings 16th Annual ACM Symposium on the Theory of Computing (1984) 302-311.*

"Projective Transformations"

- Emphasis on theory

"The running-time of this algorithm is $O(n^{3.5}L^2)$, as compared to $O(n^6L^2)$ for the ellipsoid algorithm."



A New Polynomial-Time Algorithm for Linear Programming

N. Karmarkar
AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

We present a new polynomial-time algorithm for linear programming. The running-time of this algorithm is $O(n^{3.5}L^2)$, as compared to $O(n^6L^2)$ for the ellipsoid algorithm. We prove that given a polytope P and a strictly interior point $a \in P$, there is a projective transformation of the space that maps P, a to P', a' having the following property. The ratio of the radius of the smallest sphere with center a', containing P' to the radius of the largest sphere with center a' contained in P' is $O(n)$. The algorithm consists of repeated application of such projective transformations each followed by optimization over an inscribed sphere to create a sequence of points which converges to the optimal solution in polynomial-time.

0. Some Comments on the Significance of the Result

0.1 Worst-case Bounds on Linear Programming

The simplex algorithm for linear programming has been shown to require an exponential number of steps in the worst-case [1]. A polynomial-time algorithm for linear programming was published by Khachiyan in 1979 [2]. The complexity of this algorithm is $O(n^6L^2)$ where n is the dimension of the problem and L is the number of bits in the input [3]. In this paper we present a new polynomial-time algorithm for linear programming whose time-complexity is $O(n^{3.5}L^2)$.

0.2 Polytopes and Projective Geometry

We prove a theorem about polytopes which seems to be interesting in its own right. Given a polytope $P \subseteq \mathbf{R}^n$ and a strictly interior point $a \in P$, there is a projective transformation of the space that maps P, a to P', a' having the following property: The ratio of the radius of the smallest sphere with center a', containing P' to the radius of the largest sphere with center a', contained in P' in $O(n)$.

Our algorithm for linear programming is based on repeated application of such projective transformations followed by optimization over the inscribed sphere to create a sequence of points which converges to the optimal solution in polynomial time.

0.3 Global Analysis of Optimization Algorithms

While theoretical methods for analyzing local convergence of non-linear programming and other geometric algorithms are well-developed the state-of-the-art of global convergence analysis is rather unsatisfactory. The algorithmic and analytical techniques introduced in this paper may turn out to be valuable in designing geometric algorithms with provably good global convergence properties. Our method can be thought of as a steepest descent method with respect to a particular metric space over a simplex defined in terms of "cross-ratio", a projective invariant. The global nature of our result was made possible because any (strictly interior) point in the feasible region can be mapped to any other such point by a transformation that preserves affine spaces as well as the metric. This metric can be easily generalized to arbitrary convex sets with "well-behaved" boundary and to intersections of such convex sets. This metric effectively transforms the feasible region so that the boundary of the region is at infinite distance from the interior points. Furthermore, this transformation is independent of the objective function being optimized. Contrast this with the penalty function methods which require an ad hoc mixture of objective function and penalty function. It is not clear a priori in what proportion should the two functions be mixed and the right proportion depends on both the objective function and the feasible region.

0.4 Comments on the factor "L" in running time

Since representing the output of a linear programming problem requires $O(L)$ bits per variable in the worst case, the factor L must appear in the worst-case complexity of any algorithm for linear programming.

The factor L^2 in the complexity of our algorithm comes from two sources. The number of steps of the algorithm is $O(nL)$, each step requires $O(n^{2.5})$ arithmetic operations and each arithmetic operation requires a precision of $O(L)$ bits.

If we are interested in finding a solution whose objective function value is certain fixed fraction, say 99.99% of the optimum value, then the algorithm requires only $O(n)$ steps thus saving a factor of L. The complexity of finding an exact solution can be better expressed in terms of a parameter R which we call the "discreteness factor" of the polytope, defined as:

$$R = \frac{s_{max}}{s_{min}} \quad \text{where}$$

s_{max} = Difference between the best and worst values of the objective function achieved on the vertices of the polytope.

s_{min} = Difference between the best and next best value of the objective function on the vertices of the polytope.

The actual number of steps required by our algorithm is $O(n \ln(R))$ which can be bounded above by $O(nL)$ because

$$\ln(R) = O(L) .$$

If the sequence of distinct values taken by the objective function o the vertices of the polytope is uniformly spaced, then the performance of the simplex algorithm depends linearly on R. It is possible to create examples in which R is exponentially large. Indeed, the examples which cause the simplex algorithm to run exponentially long have this property. Since performance of our algorithm depends logarithmically on R, we still get a polynomial-time algorithm.

Regarding the second factor of L, any algorithm — such as the simplex algorithm — that requires representation of inverse of a submatrix of the constraint matrix during the course of computation requires at least as much precision in arithmetic operations as our algorithm, which is $O(L)$ bits in the worst case. As compared to this worst-case bound, the simplex algorithm seems to work with much less precision in practice and the same amount of precision is sufficient for our algorithm, thus saving the second factor of L.

0.5 Performance in Practice

Each step of the algorithm requires optimization of a linear function over an ellipsoid or equivalently, solution of a linear system of equations of the type $(ADA^T)x = b$, where A is a fixed matrix and D is a diagonal matrix with positive entries which changes by small amount from step to step. We devise

1990

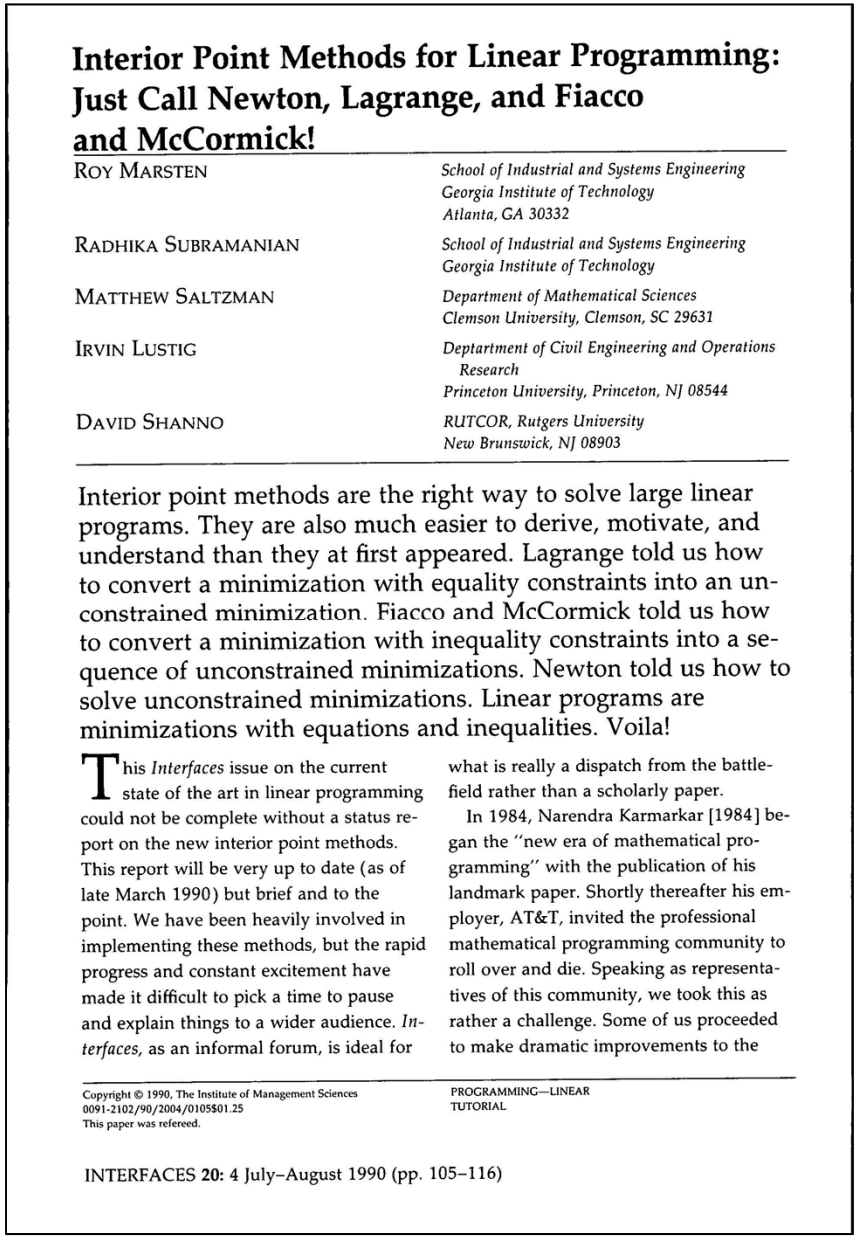
Interior Point Methods for Linear Programming: Just Call Newton, Lagrange, and Fiacco and McCormick!

- Roy Marsten *et al.*
- *Interfaces* **20** (July-August 1990) 105-116.

"Interior Point Methods"

- Elementary description
- Practical success

"Interior point methods are the right way to solve large linear programs. They are also much easier to derive, motivate, and understand than they at first appeared."



Interior Point Methods for Linear Programming: Just Call Newton, Lagrange, and Fiacco and McCormick!

ROY MARSTEN — *School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332*

RADHIKA SUBRAMANIAN — *School of Industrial and Systems Engineering, Georgia Institute of Technology*

MATTHEW SALTZMAN — *Department of Mathematical Sciences, Clemson University, Clemson, SC 29631*

IRVIN LUSTIG — *Deptartment of Civil Engineering and Operations Research, Princeton University, Princeton, NJ 08544*

DAVID SHANNO — *RUTCOR, Rutgers University, New Brunswick, NJ 08903*

Interior point methods are the right way to solve large linear programs. They are also much easier to derive, motivate, and understand than they at first appeared. Lagrange told us how to convert a minimization with equality constraints into an unconstrained minimization. Fiacco and McCormick told us how to convert a minimization with inequality constraints into a sequence of unconstrained minimizations. Newton told us how to solve unconstrained minimizations. Linear programs are minimizations with equations and inequalities. Voila!

This *Interfaces* issue on the current state of the art in linear programming could not be complete without a status report on the new interior point methods. This report will be very up to date (as of late March 1990) but brief and to the point. We have been heavily involved in implementing these methods, but the rapid progress and constant excitement have made it difficult to pick a time to pause and explain things to a wider audience. *Interfaces*, as an informal forum, is ideal for what is really a dispatch from the battlefield rather than a scholarly paper.

In 1984, Narendra Karmarkar [1984] began the "new era of mathematical programming" with the publication of his landmark paper. Shortly thereafter his employer, AT&T, invited the professional mathematical programming community to roll over and die. Speaking as representatives of this community, we took this as rather a challenge. Some of us proceeded to make dramatic improvements to the

0091-2102/90/2004/0105$01.25
This paper was refereed.

PROGRAMMING—LINEAR
TUTORIAL

INTERFACES **20:** 4 July–August 1990 (pp. 105–116)

Essential Interior-Point Method

Primal linear program

- Minimize $c^T x$
- Subject to $Ax = b$
  $x \geq 0$

Dual linear program

- Maximize $b^T \pi$
- Subject to $A^T \pi \leq c$

Optimality conditions

- Primal feasibility: $Ax^* = b,\ x \geq 0$
- Dual feasibility: $A^T \pi^* + \sigma^* = c,\ \sigma^* \geq 0$
- Complementarity: $x_j^* \sigma_j^* = 0$ for every $j = 1, \ldots, n$

Write $X = \text{diag}(x),\ \Sigma = \text{diag}(\sigma),\ e = (1, 1, \ldots, 1)$

Then . . .

Essential Interior-Point Method

Equations to be solved

- $Ax = b$
  $A^T\pi + \sigma = c$
  $X\Sigma e = 0$
- $x \geq 0$
  $\sigma \geq 0$

Suppose we have some $\bar{x} > 0,\ \bar{\sigma} > 0$

Consider a step to $\bar{x} + \Delta x,\ \bar{\pi} + \Delta\pi,\ \bar{\sigma} + \Delta\sigma$

- $A(\bar{x} + \Delta x) = b$
- $A^T(\bar{\pi} + \Delta\pi) + (\bar{\sigma} + \Delta\sigma) = c$
- $(\bar{X} + \Delta X)(\bar{\Sigma} + \Delta\Sigma)e = 0$

Then . . .

Essential Interior-Point Method

Multiply out, simplify, approximate

- $A\Delta x = 0$
- $A^T\Delta\pi + \Delta\sigma = 0$
- $\bar{X}\Delta\sigma + \bar{\Sigma}\Delta x = -\bar{X}\bar{\Sigma}e - \Delta X\Delta\Sigma e$

Solve linear equations for Δx, $\Delta\pi$, $\Delta\sigma$

- $A(\bar{X}\bar{\Sigma}^{-1})A^T\Delta\pi = b$,
  a symmetric, positive semi-definite linear system
- $\Delta x = -\bar{x} + (\bar{X}\bar{\Sigma}^{-1})A^T\Delta\pi$
- $\Delta\sigma = -\bar{\sigma} - \bar{X}^{-1}\,\bar{\Sigma}\,\Delta x$

Then . . .

Essential Interior-Point Method

Step to a new interior point

- $\bar{x} + \theta\, \Delta x,\ \bar{\pi} + \theta\, \Delta \pi,\ \bar{\sigma} + \theta\, \Delta \sigma$
- where $\theta \leq 1$ is chosen small enough that $\bar{x} + \theta\, \Delta x > 0,\ \bar{\sigma} + \theta\, \Delta \sigma > 0$

Repeat

- until the points converge

Course notes for use in teaching

- Optimization Methods III:
  Solving Linear Programs by Interior-Point Methods
- http://www.4er.org/CourseNotes

Practical Interior-Point Method

Refinements and improvements

- Long steps
- Fast, parallel solution of the equations
  - Exploit sparsity of $A(\bar{X}\bar{\Sigma}^{-1})A^T$
  - Handle dense columns specially
- Infeasible starting points
- Barrier term
  - Minimize $c^T x - \mu \sum_{j=1}^{n} \log x_j$
  - Replace $x_j^* \sigma_j^* = 0$ by $x_j^* \sigma_j^* = \mu$
  - Let $\mu \to 0$ as iterations proceed
- Predictor-corrector
  - Predictor step with $\mu = 0$
  - Corrector step with $\mu > 0$, $\Delta X \Delta \Sigma$ added to equation

1990

Interior Point Methods for Linear Programming: Just Call Newton, Lagrange, and Fiacco and McCormick!

- Roy Marsten *et al.*
- *Interfaces* **20** (July-August 1990) 105-116.

"Barrier Methods"

We now have a robust, reliable, and efficient implementation of the primal-dual interior point method for linear programs. The immediate future holds the challenge of carrying this new methodology into the areas of nonlinear and integer programming, . . .

Interior Point Methods for Linear Programming: Just Call Newton, Lagrange, and Fiacco and McCormick!

ROY MARSTEN — *School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332*

RADHIKA SUBRAMANIAN — *School of Industrial and Systems Engineering, Georgia Institute of Technology*

MATTHEW SALTZMAN — *Department of Mathematical Sciences, Clemson University, Clemson, SC 29631*

IRVIN LUSTIG — *Deptartment of Civil Engineering and Operations Research, Princeton University, Princeton, NJ 08544*

DAVID SHANNO — *RUTCOR, Rutgers University, New Brunswick, NJ 08903*

Interior point methods are the right way to solve large linear programs. They are also much easier to derive, motivate, and understand than they at first appeared. Lagrange told us how to convert a minimization with equality constraints into an unconstrained minimization. Fiacco and McCormick told us how to convert a minimization with inequality constraints into a sequence of unconstrained minimizations. Newton told us how to solve unconstrained minimizations. Linear programs are minimizations with equations and inequalities. Voila!

This *Interfaces* issue on the current state of the art in linear programming could not be complete without a status report on the new interior point methods. This report will be very up to date (as of late March 1990) but brief and to the point. We have been heavily involved in implementing these methods, but the rapid progress and constant excitement have made it difficult to pick a time to pause and explain things to a wider audience. *Interfaces*, as an informal forum, is ideal for what is really a dispatch from the battlefield rather than a scholarly paper.

In 1984, Narendra Karmarkar [1984] began the "new era of mathematical programming" with the publication of his landmark paper. Shortly thereafter his employer, AT&T, invited the professional mathematical programming community to roll over and die. Speaking as representatives of this community, we took this as rather a challenge. Some of us proceeded to make dramatic improvements to the

0091-2102/90/2004/0105$01.25
This paper was refereed.

PROGRAMMING—LINEAR
TUTORIAL

INTERFACES **20:** 4 July–August 1990 (pp. 105–116)

Mixed-Integer Programming

More general model class

- Linear with some (or all) integer-valued variables
- Some convex quadratic extensions

More powerful modeling paradigm

- Model indivisible quantities with integer variables
- Model *logic* with binary (zero-one) variables

Until 1990, mostly too hard to solve

- Computers too limited in
  speed, number of processors, memory, storage
- Implementations too simple

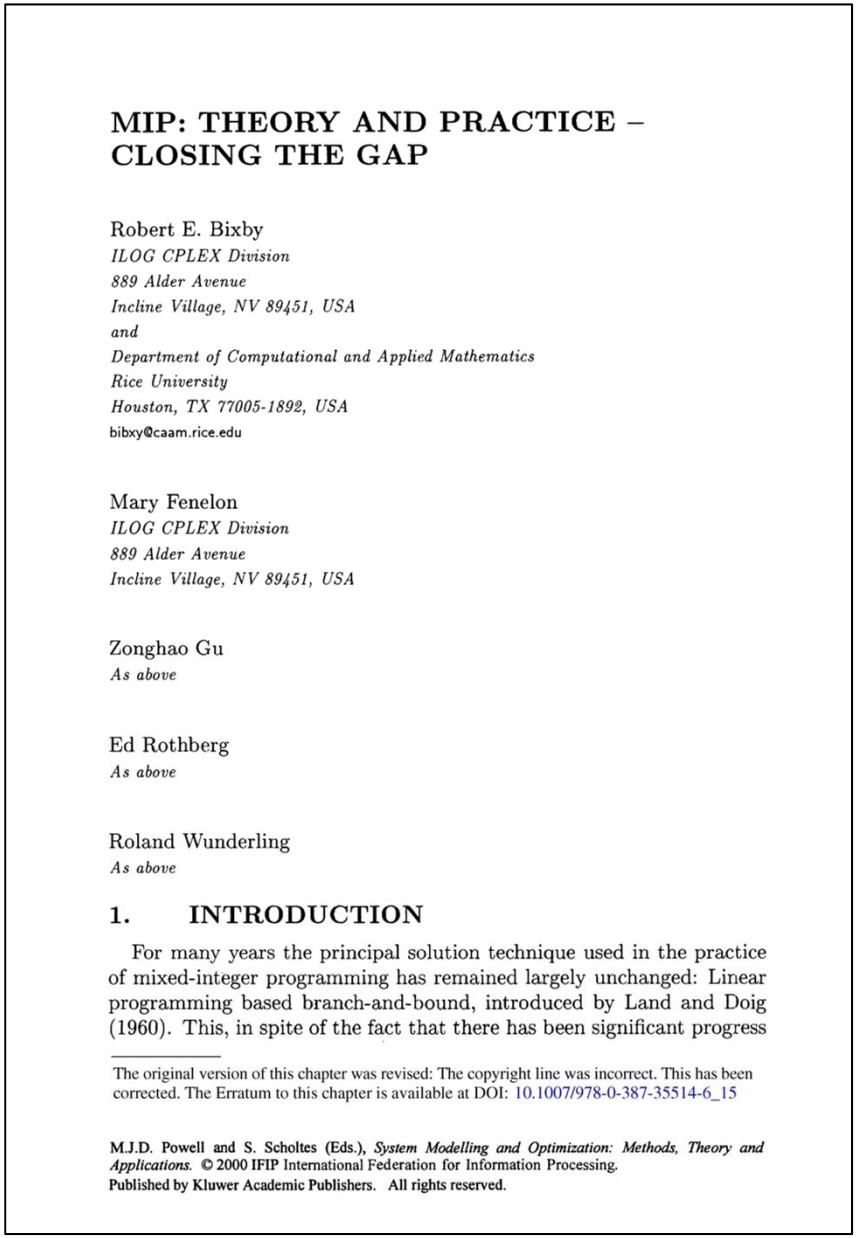
But then it became practical . . .

1999

MIP: Theory and Practice — Closing the Gap

- Robert E. Bixby *et al.*
- *System Modelling and Optimization: Methods and Applications, 19-49*

One important consequence of this work is that for large models barrier algorithms are no longer dominant; each of primal and dual simplex, and barrier is now the winning choice in a significant number of cases.



MIP: THEORY AND PRACTICE – CLOSING THE GAP

Robert E. Bixby
ILOG CPLEX Division
889 Alder Avenue
Incline Village, NV 89451, USA
and
Department of Computational and Applied Mathematics
Rice University
Houston, TX 77005-1892, USA
bibxy@caam.rice.edu

Mary Fenelon
ILOG CPLEX Division
889 Alder Avenue
Incline Village, NV 89451, USA

Zonghao Gu
As above

Ed Rothberg
As above

Roland Wunderling
As above

1. INTRODUCTION

For many years the principal solution technique used in the practice of mixed-integer programming has remained largely unchanged: Linear programming based branch-and-bound, introduced by Land and Doig (1960). This, in spite of the fact that there has been significant progress

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35514-6_15

M.J.D. Powell and S. Scholtes (Eds.), *System Modelling and Optimization: Methods, Theory and Applications.*

Mixed-Integer Programming

The most successful solver category

- Multi-strategy approach
  - Presolve routines to reduce size, improve formulation
  - Feasibility heuristics for better upper bounds
  - Constraint ("cut") generators for better lower bounds
  - Multi-processor branching search
- Solve times reduced by many orders of magnitude
  - Better algorithmic ideas and implementations
  - Faster computers with more processors, memory, storage
- Continuing improvements for 25 years!

Dominated by commercial solvers

- CPLEX
- Gurobi
- Xpress

Mixed-Integer Programming

The main source of linear programs

- Solve continuous relaxation
- Re-solve with added constraints
  - Variables fixed or bounded
  - Cuts added
- Still have an optimal but not feasible basis

Need an algorithm for re-solving quickly

- Simplex must regain feasibility, then optimize
- Barrier needs "well centered" start — cannot make good use of a previous solution
- ***Dual simplex*** is most effective

Dual Terminology

Primal linear program

Minimize $\boldsymbol{c} \cdot \boldsymbol{x}$

Subject to $A\,\boldsymbol{x} = \boldsymbol{b}$

$\boldsymbol{x} \geq \boldsymbol{0}$

Dual linear program

Maximize $\boldsymbol{\pi} \cdot \boldsymbol{b}$

Subject to $\boldsymbol{\pi}\,A \leq \boldsymbol{c}$

Simplex methods

- Primal simplex method works on the primal LP
- Dual simplex method works on the dual LP
- Different forms of LP imply different computations

Dual Geometry

Basic solution

- $\boldsymbol{\pi}\, A \leq \boldsymbol{c}$ is n inequalities on $m < n$ variables
- Pick any m inequalities corresponding to linearly independent rows of A, and make them tight
- Solve the resulting equations $\bar{\boldsymbol{\pi}}\, B = \boldsymbol{c}_B$

Vertex (basic feasible) solution

- Also $\bar{\boldsymbol{\pi}} \cdot \boldsymbol{a}_j \leq c_j$ for all of the other $n - m$ inequalities

Edge

- Relax one basic constraint q to become $<$ instead of $=$
- Then $\boldsymbol{\pi}\, B = \boldsymbol{c}_B - \theta \boldsymbol{e}^q$, $\theta \geq 0$
- Or, $\boldsymbol{\pi} = \bar{\boldsymbol{\pi}} - \theta \boldsymbol{\rho}^q$ where $\boldsymbol{\rho}^q B = \boldsymbol{e}^q$

Essential Dual Simplex Method

Given

$\boldsymbol{\pi}$ (a dual basic solution)

B (the basis)

Choose a tight equation to relax

solve $B\boldsymbol{x}_{\mathcal{B}} = \boldsymbol{b}$

$q \in \mathcal{B}$: $\boldsymbol{x}_q < 0$

Choose a new equation to become tight

solve $\boldsymbol{\rho}^q B = \boldsymbol{e}^q$

$p \in \mathcal{N}$: $\delta_p / \boldsymbol{\rho}^q \boldsymbol{a}_p = \min_{j \in \mathcal{N}: \boldsymbol{\rho}^q \boldsymbol{a}_j \geq 0} \delta_j / \boldsymbol{\rho}^q \boldsymbol{a}_j$ where $\delta_j = c_j - \boldsymbol{\pi} \boldsymbol{a}_j$

Update

- update dual slacks by

  $\delta_q \leftarrow \delta_p / \boldsymbol{\rho}^q \boldsymbol{a}_p$ and $\delta_j \leftarrow \delta_j - \left(\delta_p / \boldsymbol{\rho}^q \boldsymbol{a}_p\right) \cdot \left(\boldsymbol{\rho}^q \boldsymbol{a}_j\right)$ for all $j \in \mathcal{N}$

Practical Dual Simplex Method

Refinements and improvements

- Store A also by row to speed computation of $\boldsymbol{\rho}^q \boldsymbol{a}_j$ for typically sparse $\boldsymbol{\rho}^q$ and $\boldsymbol{a}_j$
- Step along "steepest edge" to reduce number of iterations
  - Only one extra linear system solve per iteration to update edge steepness values
  - Less than primal simplex which also requires one extra set of $\boldsymbol{a}_j$ inner products per iteration
- Get further benefits when variables are bounded
  - All basic solutions are dual-feasible
  - Simplex method can take longer steps

What's Most Widely Used?

Dual simplex

- Especially when there are integer variables
- Excellent results for continuous LPs as well

Concurrent dual simplex, primal simplex, barrier

- Run all 3 on different cores
- Stop when the first one finishes

1980

The APEX Systems: Past and Future

- C.B. Krabek, R.J. Sjoquist and D.C. Sommer
- *SIGMAP Bulletin* **29** (April 1980) 3–23.

Over the past seven years we have perceived that the size distribution of general structure LP problems being run on commercial LP codes has remained about stable. . . . That this distribution has not noticeably changed despite a massive change in solution economics is unexpected.

We do not feel that the linear programming user's most pressing need over the next few years is for a new optimizer that runs twice as fast on a machine that costs half as much (although this will probably happen). Cost of optimization is just not the dominant barrier to LP model implementation.

Why aren't more larger models being run? It is not because they could not be useful; it is because we are not successful in using them. . . . They become unmanageable. LP technology has reached the point where anything that can be formulated and understood can be optimized at a relatively modest cost.

1982

On the Development of a General Algebraic Modeling System in a Strategic Planning Environment

- J. Bisschop and A. Meeraus
- *Mathematical Programming Study* **20** (1982) 1-29.

The heart of it all is the fact that solution algorithms need a data structure which . . . is impossible to comprehend by humans, while, at the same time, meaningful problem representations for humans are not acceptable to machines. We feel that the two translation processes required (to and from the machine) can be identified as the main source of difficulties and errors.

Mathematical Programming Study 20 (1982) 1–29
North-Holland Publishing Company

ON THE DEVELOPMENT OF A GENERAL ALGEBRAIC MODELING SYSTEM IN A STRATEGIC PLANNING ENVIRONMENT*

Johannes BISSCHOP** and Alexander MEERAUS
Development Research Center, The World Bank, Washington, DC 20433, U.S.A.

Received 18 March 1980
Revised manuscript received 8 May 1981

Modeling activities at the World Bank are highlighted and typified. Requirements for successful modeling applications in such a strategic planning environment are examined. The resulting development of a General Algebraic Modeling System (GAMS) is described. The data structure of this system is analyzed in some detail, and comparisons to other modeling systems are made. Selected aspects of the language are presented. The paper concludes with a case study of the Egyptian Fertilizer Sector in which GAMS has been used as a modeling tool.

Key words: Algebraic Modeling System, Modeling Language, Strategic Planning, Applications.

1. Introduction

The first portion of this paper focuses on the dynamics of modeling activities in a strategic planning environment such as the World Bank. This environment is broadly characterized by long-term, often ill-defined and poorly understood issues, which require near immediate decision making. It is the long-term impact of the decisions that make them important. Government planning agencies and corporate planning offices are other examples of a strategic planning environment. Mathematical models are a potentially powerful tool during the process of making good plans and decisions in such an environment, but their effective use has often been limited. This is not only due to the extensive resource requirements in terms of technical skills, money and time, but also because of such intangible issues as the low reliability of model generators, and the extensive communication problems that occur during the dissemination of models and their results.

The second portion of the paper focuses on our efforts to eliminate some of the current barriers to successful modeling applications, namely the development of a General Algebraic Modeling System (GAMS). The aim of this system

* The views and interpretations in this document are those of the authors and should not be attributed to the World Bank, to its affiliated organizations or to any individual acting in their behalf.
** Presently at Shell Research, Amsterdam, The Netherlands.

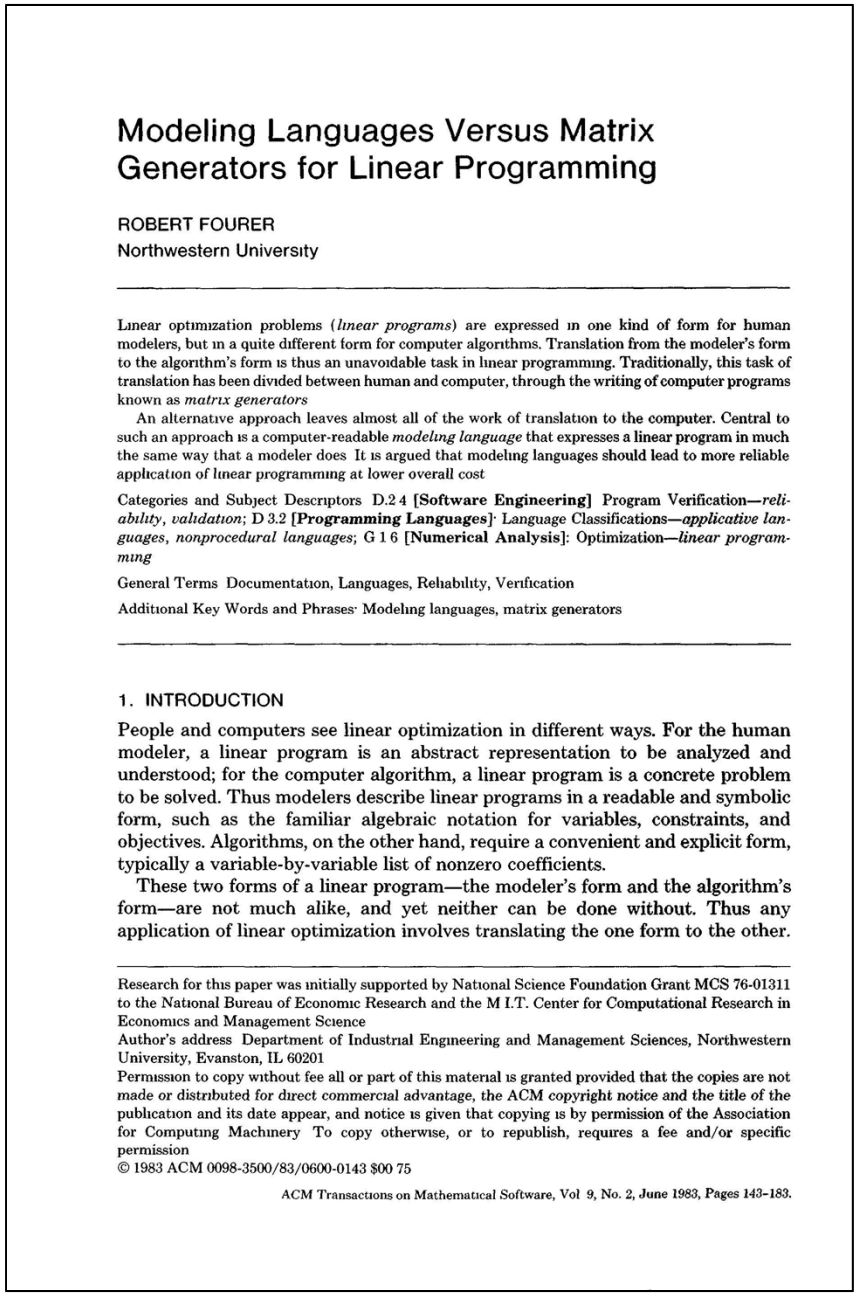
1

1983

Modeling Languages versus Matrix Generators for Linear Programming

- Robert Fourer
- *ACM Transactions on Mathematical Software* **9** (1983) 143-183.

People and computers see linear optimization in different ways. . . . These two forms of a linear program — the modeler's form and the algorithm's form — are not much alike, and yet neither can be done without. Thus any application of linear optimization involves translating the one form to the other.



Modeling Languages Versus Matrix Generators for Linear Programming

ROBERT FOURER
Northwestern University

Linear optimization problems (*linear programs*) are expressed in one kind of form for human modelers, but in a quite different form for computer algorithms. Translation from the modeler's form to the algorithm's form is thus an unavoidable task in linear programming. Traditionally, this task of translation has been divided between human and computer, through the writing of computer programs known as *matrix generators*

An alternative approach leaves almost all of the work of translation to the computer. Central to such an approach is a computer-readable *modeling language* that expresses a linear program in much the same way that a modeler does It is argued that modeling languages should lead to more reliable application of linear programming at lower overall cost

Categories and Subject Descriptors D.2.4 **[Software Engineering]** Program Verification—*reliability, validation*; D.3.2 **[Programming Languages]**: Language Classifications—*applicative languages, nonprocedural languages*; G.1.6 **[Numerical Analysis]**: Optimization—*linear programming*

General Terms Documentation, Languages, Reliability, Verification

Additional Key Words and Phrases: Modeling languages, matrix generators

1. INTRODUCTION

People and computers see linear optimization in different ways. For the human modeler, a linear program is an abstract representation to be analyzed and understood; for the computer algorithm, a linear program is a concrete problem to be solved. Thus modelers describe linear programs in a readable and symbolic form, such as the familiar algebraic notation for variables, constraints, and objectives. Algorithms, on the other hand, require a convenient and explicit form, typically a variable-by-variable list of nonzero coefficients.

These two forms of a linear program—the modeler's form and the algorithm's form—are not much alike, and yet neither can be done without. Thus any application of linear optimization involves translating the one form to the other.

Research for this paper was initially supported by National Science Foundation Grant MCS 76-01311 to the National Bureau of Economic Research and the M.I.T. Center for Computational Research in Economics and Management Science
Author's address Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60201

Modeler's Form: Multicommodity

Given

O Set of origins (factories)

D Set of destinations (stores)

P Set of products

and

a_{ip} Amount available, for each $i \in O$ and $p \in P$

b_{jp} Amount required, for each $j \in D$ and $p \in P$

l_{ij} Limit on total shipments, for each $i \in O$ and $j \in D$

c_{ijp} Shipping cost per unit, for each $i \in O$, $j \in D$, $p \in P$

d_{ij} Fixed cost for shipping any amount from $i \in O$ to $j \in D$

s Minimum total size of any shipment

n Maximum number of destinations served by any origin

Multicommodity Transportation

Modeler's Form

Determine

X_{ijp} Amount of each $p \in P$ to be shipped from $i \in O$ to $j \in D$

Y_{ij} 1 if any product is shipped from $i \in O$ to $j \in D$
0 otherwise

to minimize

$$\sum_{i \in O} \sum_{j \in D} \sum_{p \in P} c_{ijp} X_{ijp} + \sum_{i \in O} \sum_{j \in D} d_{ij} Y_{ij}$$

Total variable cost plus total fixed cost

Multicommodity Transportation

Modeler's Form

Subject to

$\sum_{j \in D} X_{ijp} \leq a_{ip}$ for all $i \in O, p \in P$

Total shipments of product p out of origin i must not exceed availability

$\sum_{i \in O} X_{ijp} = b_{jp}$ for all $j \in D, p \in P$

Total shipments of product p into destination j must satisfy requirements

$\sum_{p \in P} X_{ijp} \leq l_{ij} Y_{ij}$ for all $i \in O, j \in D$

When there are shipments from origin i to destination j, the total may not exceed the limit, and Y_{ij} must be 1

Multicommodity Transportation

Modeler's Form

Subject to

$\sum_{p \in P} X_{ijp} \geq sY_{ij}$ for all $i \in O, j \in D$

When there are shipments from origin i to destination j, the total amount of shipments must be at least s

$\sum_{j \in D} Y_{ij} \leq n$ for all $i \in O$

Number of destinations served by origin i must be as most n

Special Features of Modeler's Form

Symbolic

General

- Independent of data for particular cases

Concise

- Length of description depends on complexity of the model, not size of the linear program

Understandable

Same in a Modeling Language (AMPL)

Symbolic data

```
set ORIG;   # origins
set DEST;   # destinations
set PROD;   # products

param supply {ORIG,PROD} >= 0;  # availabilities at origins
param demand {DEST,PROD} >= 0;  # requirements at destinations
param limit {ORIG,DEST} >= 0;   # capacities of links

param vcost {ORIG,DEST,PROD} >= 0;  # variable shipment cost
param fcost {ORIG,DEST} > 0;        # fixed usage cost

param minload >= 0;                 # minimum shipment size
param maxserve integer > 0;         # maximum destinations served
```

Multicommodity Transportation

AMPL Formulation

Symbolic model: variables and objective

```
var Trans {ORIG,DEST,PROD} >= 0;    # actual units to be shipped
var Use {ORIG, DEST} binary;        # 1 if link used, 0 otherwise

minimize Total_Cost:
   sum {i in ORIG, j in DEST, p in PROD} vcost[i,j,p] * Trans[i,j,p]
 + sum {i in ORIG, j in DEST} fcost[i,j] * Use[i,j];
```

$$\sum_{i \in O} \sum_{j \in D} \sum_{p \in P} c_{ijp} X_{ijp} + \sum_{i \in O} \sum_{j \in D} d_{ij} Y_{ij}$$

Multicommodity Transportation

AMPL Formulation

Symbolic model: constraints

```
subject to Supply {i in ORIG, p in PROD}:
   sum {j in DEST} Trans[i,j,p] <= supply[i,p];

subject to Demand {j in DEST, p in PROD}:
   sum {i in ORIG} Trans[i,j,p] = demand[j,p];

subject to Multi {i in ORIG, j in DEST}:
   sum {p in PROD} Trans[i,j,p] <= limit[i,j] * Use[i,j];

subject to Min_Ship {i in ORIG, j in DEST}:
   sum {p in PROD} Trans[i,j,p] >= minload * Use[i,j];

subject to Max_Serve {i in ORIG}:
   sum {j in DEST} Use[i,j] <= maxserve;
```

Multicommodity Transportation

AMPL Formulation

Explicit data independent of symbolic model

```
set ORIG := GARY CLEV PITT ;
set DEST := FRA DET LAN WIN STL FRE LAF ;
set PROD := bands coils plate ;

param supply (tr):  GARY   CLEV   PITT :=
            bands    400    700    800
            coils    800   1600   1800
            plate    200    300    300 ;

param demand (tr):
            FRA   DET   LAN   WIN   STL   FRE   LAF :=
   bands    300   300   100    75   650   225   250
   coils    500   750   400   250   950   850   500
   plate    100   100     0    50   200   100   250 ;

param limit default 625 ;

param minload := 375 ;
param maxserve := 5 ;
```

AMPL Formulation

Explicit data (continued)

```
param vcost :=

 [*,*,bands]:  FRA  DET  LAN  WIN  STL  FRE  LAF :=
        GARY    30   10    8   10   11   71    6
        CLEV    22    7   10    7   21   82   13
        PITT    19   11   12   10   25   83   15

 [*,*,coils]:  FRA  DET  LAN  WIN  STL  FRE  LAF :=
        GARY    39   14   11   14   16   82    8
        CLEV    27    9   12    9   26   95   17
        PITT    24   14   17   13   28   99   20

 [*,*,plate]:  FRA  DET  LAN  WIN  STL  FRE  LAF :=
        GARY    41   15   12   16   17   86    8
        CLEV    29    9   13    9   28   99   18
        PITT    26   14   17   13   31  104   20 ;

param fcost:   FRA  DET  LAN  WIN  STL  FRE  LAF :=
        GARY  3000 1200 1200 1200 2500 3500 2500
        CLEV  2000 1000 1500 1200 2500 3000 2200
        PITT  2000 1200 1500 1500 2500 3500 2200 ;
```

AMPL Solution

Model + data = problem instance to be solved

```
ampl: model multmip3.mod;
ampl: data multmip3.dat;

ampl: option solver gurobi;

ampl: solve;

Gurobi 7.0.0: optimal solution; objective 235625
332 simplex iterations
23 branch-and-cut nodes

ampl: display Use;

Use [*,*]

:    DET FRA FRE LAF LAN STL WIN  :=
CLEV   1   1   1   0   1   1   0
GARY   0   0   0   1   0   1   1
PITT   1   1   1   1   0   1   0
;
```

AMPL Solution

Solver choice independent of model and data

```
ampl: model multmip3.mod;
ampl: data multmip3.dat;

ampl: option solver cplex;

ampl: solve;

CPLEX 12.7.0.0: optimal integer solution; objective 235625
135 MIP simplex iterations
0 branch-and-bound nodes

ampl: display Use;

Use [*,*]

:    DET FRA FRE LAF LAN STL WIN  :=
CLEV   1   1   1   0   1   1   0
GARY   0   0   0   1   0   1   1
PITT   1   1   1   1   0   1   0
;
```

Multicommodity Transportation

AMPL Solution

Solver choice independent of model and data

```
ampl: model multmip3.mod;
ampl: data multmip3.dat;

ampl: option solver xpress;

ampl: solve;

XPRESS 29.01: Global search complete
Best integer solution found 235625
4 integer solutions have been found, 7 branch and bound nodes

ampl: display Use;

Use [*,*]

:    DET FRA FRE LAF LAN STL WIN  :=
CLEV   1   1   1   0   1   1   0
GARY   0   0   0   1   0   1   1
PITT   1   1   1   1   0   1   0
;
```