台大管理論叢
國立台灣大學管理學院
第一卷，第一期，中國民國七十九年五月

THE VARIANTS OF KARMARKAR'S AND SIMPLEX ALGORITHMS FOR LINEAR PROGRAMMING

Wen-hsien Chen

Department of Business Administration
National Taiwan University

ABSTRACT

We review and modify the Karmarkar's polynomial-time algorithm and its variants for linear programming. These variants are interior point algorithms, Newton barrier methods, and box method. Those algorithms still have polynomial-time computational complexity. For logarithm barrier function algorithm, each iteration updates a penalty parameter and finds an approximate Newton's direction associated with the Kuhn-Tucker system of equations. This paper briefly discusses those algorithms and some extensions of Karmarkar type algorithm to simplex method. We implemented those algorithms in Fortran programs and tested the computational results for iteration numbers and CPU times.

[Keywords] *– Linear programming, Simplex method, Karmarkar's algorithm, Interior point algorithm, Barrier function, Box method, Polynomial-time algorithm.*

I. INTRODUCTION

The birth of linear programming is usually identified with the development of the simplex method in 1947 by Dantzig [5]. Techniques for solving linear programming have been studied for four decades. The simplex method still remains the major algorithm used in linear programming, although recently interior point methods are serious competitors.

Two more recent approaches to solving linear programming are the ellipsoid method (Khachiyan [10]) and the projective algorithm (Karmarkar [9]). Todd [16] found that the ellipsoid method and projective algorithm are closely related. Both of the methods generate a shrinking dual ellipsoid containing the optimal dual solutions.

In particular, Karmarkar's projective method is a special case of a projected Newton method applied to the logarithmic barrier function. Many researchers are re-examining the effectiveness of methods that nonlinearize a linear program. The proof of polynomial complexity of Karmarkar's original algorithm suggests that a suitable nonlinear transformation may overcome the inherently combinatorial nature of the simplex method.

Lusting [12], Tomlin [17], and Adler et al. [1] have compared implementations of interior point algorithms with simplex method code. An implementation of Newton's barrier method reported by Gill et al. [7] presents the first extensive computational evidence indicating that an interior point algorithm can be comparable in speed with the simplex method.

Zikan and Cottle [21] presented the box method, this algorithm uses "boxes" or parallelepipeds. Each such box is associated with a basic solution of constrains of the problem. The computation of each search direction is done over the current box with subproblems of linear program. This interior point algorithm makes no use of nonlinear programming.

In the simplex method, the current solution is modified by introducing a non-zero coefficient for one of the columns in the constraint matrix.

Karmarkar's method allows the current solution to be modified by introducing several columns at once. The eliminating columns method (Ye [20]) is a modification of simplex algorithm.

This paper presents the modified variations of interior point algorithm by Karmarkar and an algorithm based on the logarithmic barrier function approach (Moniteiro and Adler [14]). The logarithmic barrier function method was first used for linear programming problems by Frisch [6]. The directions generated by this algorithm are essentially the same as the algorithm of Kojima et al. [11].

In section 2, we describe interior point algorithms with their initial interior point and stopping criterion. In section 3, we present the logarithmic barrier function algorithm. Section 4 and 5 are the box method algorithm and the eliminating columns in simplex algorithm, respectively. Section 6 is the computational results and section 7 is conclusion.

II. THE INTERIOR POINT ALGORITHMS

Consider the linear programming problem:

$$\begin{aligned} \text{LP1:} \quad & \max \quad c^T x \\ & \text{s.t.} \quad Ax \leqslant b \end{aligned}$$

where c and x are n-dimensional vectors, b is an m-dimensional vector and A is a full rank m by n matrix. We assume that LP1 has an interior point, x^0, and is bounded.

Starting at x^0, the algorithm presented generates a sequence of feasible interior for LP1, $\{x^1, x^2, \ldots, x^k, \ldots\}$ such that

$$c^T x^{k+1} > c^T x^k.$$

Suppose $x = (x_1, \ldots, x_n)^T$ is an n-vector, we denote $D = \mathrm{diag}(x_1, \ldots, x_n)$ to be the diagonal matrix, with the components of vector on the diagonal and $\| \ \|$ to be the Euclidean norm.

The stopping criterion is used to determine when to terminate the main loop of the algorithm. The algorithm is terminated when the relative improvement of the objective function is small, i.e., when

$$| c^T x^k - c^T x^{k-1} | / \max\{1, |c^T x^{k-1}|\} < \epsilon. \tag{2.1}$$

where ϵ is a given small positive tolerance.

Now, we present an interior point algorithm for LP1.

```
begin  Algorithm A
    Let x^0 and γ be given such that Ax^0 < b and 0 < γ < 1.
    Set k: = 0
  while condition (2.1) is not satisfied do
  begin
    v: = b − Ax^k
    D: = diag (1/v_1, . . . , 1/v_m)
    d_x: = (A^T D^2 A)^-1 c
    d_v: = −Ad_x
    α: = γ × min{v_i/(d_v)_i | (d_v)_i < 0, i = 1, . . . , m}
    x^(k+1): = x^k + αd_x
    k: = k + 1
  end
end.
```

If $v = b - Ax$ is the vector of slack variables, we apply the affine transformation D to y:

$$w = Dy.$$

Then rewriting LP1 in terms of w

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax + D^{-1} w = b \\ & w \geqslant 0 \end{aligned}$$

The algorithm selects a search direction

$$d = [d_x \;\; d_w] .$$

This direction is a solution to the homogeneous system

$$DAd_x + d_w = 0 \tag{2.2}$$

$$A^T D(DAd_x + d_w) = 0 \tag{2.3}$$

so that

$$d_x = -(A^T D^2 A)^{-1} A^T D d_w . \tag{2.4}$$

We select d_w as the steepest ascent direction

$$d_w = -DA(A^T D^2 A)^{-1} c. \tag{2.5}$$

From (2.5) we extract a solution to (2.2)

$$d_x = (A^T D^2 A)^{-1} c \tag{2.6}$$

Also from (2.5), we can apply the inverse affine transformation to d_w resulting in

$$d_v = -A(A^T D^2 A)^{-1} c \tag{2.7}$$

The above formulation allows for inexact projections without loss of feasibility. In practice, we compute d_x according to (2.6) and adjust

$$d_v = -Ad_x \tag{2.8}$$

to maintain feasibility.

The algorithm requires that an initial interior point x^0 be provided. Numerically, it is desirable that this initial point be far from the facets of the polyhedral set defining the solution space. Such an initial interior point can be obtained by the following procedure:

Set $x^0 := (\|b\|/\|Ac\|)c$.
If $v = b - Ax^0 > 0$, then
  x^0 is the initial interior point.
else
  $y^0 := -2 \times \min\{v_i \mid i = 1, \ldots, m\}$
  μ: = a large constant.
  $M := \mu \times |c^T x^0|/y^0$
  $e^T := (1,1,\ldots,1)^T$
Applied (x^0, y^0) and algorithm A to the following LP problem:
  LP_a: max $c^T x - My$
    s.t. $Ax - e^T y \leqslant b$
Suppose it is terminated at iteration k with (x^k, y^k)
  if $y^k < 0$ then x^k is a interior point of LP1
  if $y^k > 0$ then LP1 is infeasible

Now we consider the following standard form of linear programming

$$\text{LP2:} \quad \min \quad c^T x$$
$$\text{s.t.} \quad Ax = b$$
$$x \geqslant 0$$

where A is an m × n matrix with rank m, $m < n$, $b \in R^m$, $c \in R^n$.

We describe another interior point algorithm for LP2.

begin Algorithm B

Let x^0 be an initial starting point.

Set k: = 0

Choose ϵ and α are small positive scalars.

while stopping criterion not satisfied do

begin

$\overline{x}$: = x^k

D: = diag($\overline{x}_1, \ldots, \overline{x}_n$)

H: = AD

u: = $[I - H^T(HH^T)^{-1}H]Dc$

$\| u \|$: = $\sqrt{u^T u}$

if $\| u \| \leqslant \epsilon$ then

stop, x^k is an approximately optimal solution to LP2.

else

p: = $u/\| u \|$

if $p_i \leqslant \epsilon$, for i = 1, . . . , n then

the LP2 problem is unbounded.

else

endif

q: = Max{$p_i | i = 1, \ldots, n$}

λ: = 1/q

x^{k+1}: = $x^k - \alpha\lambda D_p$

Set k: = k + 1

endif

end

end.

Suppose $\bar{x}$ is a interior feasible point for LP2.

$$D = \text{diag}\,(\bar{x}_1, \ldots, \bar{x}_m)$$
$$H = AD$$
$$y = (HH^T)^{-1} HDc$$
$$u = [I - H^T(HH^T)^{-1} H]\, Dc$$

Karmarkar [9] proved that if LP2 is feasible and nondegenerate with optimal solution x^*, and $\bar{x}$ is a interior point, such that $\bar{x} \in N(x^*, \delta)$, then $\bar{y}$ is a dual solution of LP2, such $\bar{y} \in N(x^*, \xi)$, where δ and ξ are sufficiently small positive scalars. Then

$$u = Dc - DA^T\bar{y}$$
$$= D(c - A^T\bar{y})$$

If $u/\|u\| < \epsilon$, since ϵ is for arbitrary small, so the dual of LP2 is infeasible, that implies the LP2 is unbounded. If $\|u\| < \epsilon$, then

$$D(c - A^T\bar{y}) = 0$$

$$\bar{x}_i = c_i - \sum_{j=1}^{m} A_{ji}\bar{y}_i, \qquad \text{for } i = 1, \ldots, n$$

By the complementary slackness theorem, $\bar{x}$ and $\bar{y}$ are the optimal solutions of LP2 and dual of LP2, respectively. This proves that Algorithm B can converges to the optimal solution of LP2.

III. THE LOGARITHMIC BARRIER FUNCTION ALGORITHM

We consider the pair of the standard form linear program and its dual:

$$\begin{array}{lll} \text{LP3:} & \min & c^T x \\ & \text{s.t.} & Ax = b \\ & & x \geqslant 0 \end{array} \qquad \begin{array}{lll} \text{D3:} & \max & b^T y \\ & \text{s.t.} & A^T y + z = c \\ & & z \geqslant 0 \end{array}$$

where A is an m $\times$ n matrix with rank m, b and y are m-vectors, and c, x, z are n-vectors. We assume both LP3 and D3 are feasible.

We denote a point $w \equiv (x,y,z) \in R^n \times R^m \times R^n$. The logarithmic barrier function technique is applied to the problem LP3.

$$\begin{array}{lll} P_\mu: & \min & c^T x - \mu \sum_{j=1}^{n} \ln x_j \\ & \text{s.t.} & Ax = b \\ & & x > 0 \end{array}$$

where $\mu > 0$ is the penalty parameter. As μ converge to 0, we would expected the optimal solutions of P_μ to converge to an optimal solution of LP3.

We denote $X = \text{diag}\,(x_1, \ldots, x_n)$ and $Z = \text{diag}\,(z_1, \ldots, z_n)$. The objective function of the problem P_μ is a strictly convex function. This implies that the problem P_μ has at most one global minimum, and that this global minimum, if it exists, satisfies the Kuhn-Tucker stationary condition:

$$\begin{array}{ll} (1) & ZXe - \mu e = 0 \\ (2) & Ax - b = 0,\ x > 0 \\ (3) & A^T y + z - c = 0 \end{array} \qquad (3.1)$$

Let $w \in S \times T$. We denote $f(w) = (f_1(w), \ldots, f_n(w)))^T$ by

$$f_i = x_i z_i, \quad i = 1, \ldots, n$$

With this notation, the first equation of (3.1) becomes:

$$f_i(w(\mu)) \equiv x_i(\mu)z_i(\mu), \quad i = 1, \ldots, n. \tag{3.2}$$

We denote by Γ the path of solutions $w(\mu)$, $\mu > 0$, i.e.,

$$\Gamma = \{w(\mu) \equiv (x(\mu), y(\mu), z(\mu)) \mid \mu > 0\}$$

The algorithm C which will be presented in following, will "follow" this path Γ with the objective of approaching the desired solutions of the original problems (P) and (D).

We are now ready to describe the algorithm. At the beginning of the algorithm, we assume that an initial point $w^0 \equiv (x^0, y^0, z^0) \in S \times T$ is available such that the following criterion of closeness with respect to the path Γ is satisfied:

$$\| f(w^0) - \mu^0 e \| \leqslant \theta \mu^0 \tag{3.3}$$

where μ^0 is a positive constant, $\theta = 0.1$, and $e = (1, \ldots, 1)^T$.

We now state the algorithm.

begin Algorithm C

Let w^0 satisfy (3.1) and $\mu^0 > 0$ satisfy (3.3).

Let ϵ be a tolerance for the duality gap.

Let $\theta := 0.1$, $\delta := 0.1$, $\gamma := 0.1$

Set $k := 0$.

while $c^T x^k - b^T y^k < \epsilon$ do

begin

$f := x^k$, $g := z^k$, $X := \mathrm{diag}(x_1, \ldots, x_n)$, $Z := \mathrm{diag}(z_1, \ldots, z_n)$

Choose $s = (u, v) \in R^n_+ \times R^n_+$, satisfying:

$$\left(\frac{|f_i - u_i|}{|u_i|}\right) \leqslant \gamma, \quad i = 1, \ldots, n$$

$$\left(\frac{|g_i - v_i|}{|v_i|}\right) \leqslant \gamma, \quad i = 1, \ldots, n$$

$\mu^{k+1} := \mu^k(1 - \delta/\sqrt{n})$

$U := \mathrm{diag}(u_1, \ldots, u_n)$ and $V := \mathrm{diag}(v_1, \ldots, v_n)$

$\Delta x^k := [V^{-1} - V^{-1}UA^T(AV^{-1}UA^T)^{-1}AV^{-1}](XZe - \mu^{k+1}e)$

$\Delta y^k := -[(AV^{-1}UA^T)^{-1}AV^{-1}](XZe - \mu^{k+1}e)$

$\Delta z^k := [A^T(AV^{-1}UA^T)^{-1}AV^{-1}](XZe - \mu^{k+1}e)$

$\Delta w^k := (\Delta x^k, \Delta y^k, \Delta z^k)$

$w^{k+1} := w^k - \Delta w^k$.

Set $k := k + 1$

end

end.

If we denote the left hand side of equations (3.1) by $H(w) \equiv H(x,y,z)$, the Newton's direction Δw at $w \in S \times T$ is defined by

$$D_w H(w)\Delta w = H(w)$$

where $\Delta w = (\Delta x, \Delta y, \Delta z) \in R^n \times R^m \times R^n$ and $D_w H(w)$ denotes the Jacobian of H at $w \equiv (x,y,z)$. We observe that $D_w H(x,y,z)$ is given by

$$J(x,z) \equiv D_w H(w) = \begin{pmatrix} Z & 0 & X \\ A & 0 & 0 \\ 0 & A^T & I \end{pmatrix}$$

The direction Δw is defined by the following system of linear equations

$$J(u,v)\Delta w = H(x,y,z)$$

where the points $u \in R^n$ and $v \in R^n$ will be chosen to approximate $x \in R^n$ and $z \in R^n$ respectively. More specifically, $\Delta w = (\Delta x, \Delta y, \Delta z)$ is defined by the following equations

$$V\Delta x + U\Delta z = XZe - \mu e \tag{3.4}$$

$$A\Delta x = 0 \tag{3.5}$$

$$A^T\Delta y + \Delta z = 0 \tag{3.6}$$

where $\mu > 0$ is a prespecified penalty parameter, $U = \mathrm{diag}(u_1, \ldots, u_n)$, $V = \mathrm{diag}(v_1, \ldots, v_n)$.

Note that the solution $\Delta w = (\Delta x, \Delta y, \Delta z)$ of the system of equations (3.4)–(3.6) depends on the current inerate $w = (x,y,z)$, on the Jacobian of H at the "approximation" $s = (u,v)$ of (x,z), and on the penalty parameter $\mu > 0$. In order to indicate this dependence, we denote the solution $(\Delta x, \Delta y, \Delta z)$ by $\Delta w(w,s,\mu)$.

By simple calculation, we obtain the following expressions for $\Delta x, \Delta y, \Delta z$.

$$\Delta x = [V^{-1} - V^{-1}UA^T(AV^{-1}UA^T)^{-1}AV^{-1}](XZe - \mu^{k+1}e)$$
$$\Delta y = -[(AV^{-1}UA^T)^{-1}AV^{-1}](XZe - \mu^{k+1}e)$$
$$\Delta z = [A^T(AV^{-1}UA^T)^{-1}AV^{-1}](XZe - \mu^{k+1}e)$$

Therefore, to calculate the direction $\Delta w \equiv (\Delta x, \Delta y, \Delta z)$, the inverse of the matrix $(AV^{-1}UA^T)$ needs to be calculated. If the current diagonal matrix $V^{-1}U$ differs from the previous one by exactly k diagonal elements then, by performing k rank-one updates, we are able to compute the inverse of the matrix $(AV^{-1}UA^T)$ in $O(n^2k)$ arithmetic operations. Observe that all the other operations involved in the computation of $\Delta w \equiv \Delta w(w,s,\mu)$ is of the order of $O(n^2)$ arithmetic operations.

IV. BOX METHOD ALGORITHM

We consider the pair of the standard form linear program and its dual:

$$\begin{array}{llll} \text{LP4:} & \max & c^T x & \\ & \text{s.t.} & Ax \leqslant b & \end{array} \qquad \begin{array}{llll} \text{D4:} & \min & b^T y & \\ & \text{s.t.} & A^T y = c & \\ & & y \geqslant 0 & \end{array}$$

where A is an $m \times n$ matrix with rank n, $m \geqslant n$, b is m-vector, and c, x are n-vectors.

Let $X = \{x \mid Ax \leqslant b\}$. We make the following assumptions:

(1) $\| A_i. \| = 1$, for $i = 1, \ldots, m$

(2) There exists $x^0 \in R^n$ such that $Ax^0 < b$

(3) $\{x \mid c^T x \geqslant c^T x^0, x \in X\}$ is bounded.

We now state the box method as algorithm D.

begin Algorithm D

Let $x^0 \in$ int X and choose $\theta \in (0,1)$.

Set k: = 0

while stopping criterion not satisfied do

begin

w^k: = $b - Ax^k$ and y: = w^k

find an index set $E \subseteq \{1, \ldots, m\}$ such that

$\{A_i. \mid i \in E\}$ are linearly independent and

$\sum_{i \in E} y_i$ is minimum.

B: = A_E. which is an $n \times n$ nonsingular matrix

z^k: = $B^{-1} y_E$

q: = Az^k

p: = $\min\{y_i/q_i \mid q_i > 0, i = 1, \ldots, m\}$

s: = θp

if p = 1 and $c^T B^{-1} \geqslant 0$ then

the vector $x^{k+1} = x^k + z^k$ is optimal.

else

x^{k+1}: = $x^k + sz^k$

endif

set k: = k + 1

end

end.

The proof of the convergence to optimal solution of Algorithm D is given in [21].

V. ELIMINATING COLUMNS IN SIMPLEX ALGORITHM

We consider the canonical form of linear program

$$\begin{aligned} \text{LP5:} \quad & \min \quad c^T x \\ & \text{s.t.} \quad Ax = b \\ & \qquad x \geqslant 0 \end{aligned}$$

where A is an m × n matrix, b is m-vector, and c, x, are n-vectors.

The simplex algorithm is revised to the following algorithm:

begin Algorithm E

Let x^0 be a feasible basic solution with canonical form LP5

Set k: = 0

while stopping criterion not satisfied do

begin

if $c \geqslant 0$ then

stop, x^k is optimal solution

else

endif

if $S = \{\delta \mid c - \delta e^T A \geqslant 0\}$ is nonempty then

$\Delta := -(e^T b) \times \max S$

else $\Delta := \infty$

endif

$p := \min\{a_{ij}/b_i \mid b_i > 0,\ 1 \leqslant i \leqslant m\}$

if $c_j + \Delta \times \min\{0,p\} > 0$ then

eliminate column j

else

endif

select a entering variable x_s

check the boundedness condition

select a leaving variable x_r

suppose the a_{rs} is the pivot element

if $a_{is} \leqslant 0,\ i \neq r$ or

$b_i/a_{is} > \max\{\Delta/|c_s|, b_r/a_{rs}\}$ for all $a_{is} > 0,\ i \neq r$

then pivot on a_{rs} and eliminate column for x_r

else pivot on a_{rs}

endif

replace A, c, and b with the new values

set k: = k + 1

end

end.

The proofs of the eliminate column theorems are given in [20].

VI. IMPLEMENTATION AND COMPUTATIONAL RESULTS

Those five algorithms described above were implemented using FORTRAN 77 and the subroutines library on IBM 3039 and IBM 4341. Some linear programming test problems were collected by Systems Optimization Laboratory in the Department of Operations Research at Stanford University, the other test problems are real-life problems. The statistics for the nine test problems are shown in Table 1.

Table 1. Test problem statistics

Problems	Rows	Columns	Nonzeros	optimal objective value
EX1	10	18	32	+3.1569741e+01
AFIRO	28	32	88	−4.6475315e+02
ADLITTLE	56	97	465	+2.2549496e+05
SHARE2B	99	79	802	−4.1573224e+02
ISRAEL	175	142	2358	−8.9664483e+05
E226	226	282	3038	−1.8751929e+01
BANDM	306	472	2659	−1.5862802e+02
SHIP08	778	2467	7194	+1.9200982e+06
SCSD8	397	2750	8584	+9.0500000e+02

Algorithms A, and B were run on the IBM 3039 at U.C.-Berkeley, U.S.A. Execution times are compared to those of the simplex code MINOS 4.0 using IBM 3039 [1]. Algorithms C, D, and E were conducted on the IBM 4341 at National Defense Management College, Taiwan. All CPU times represent the time in seconds to read in the data from MPS format, solve the linear program, and write out a solution. Algorithms A, B, and C terminate when

the relative improvement in the objective function falls below $\epsilon = 10^{-8}$. The computational results are presented in Table 2, Table 3, Figure 1 and Figure 2.

Table 2. Computational Results for Algorithms A, B, and MINOS

Problems	Algorithm A		Algorithm B		Simplex MINOS	
	Itrs.	CPU	Itrs.	CPU	Itrs.	CPU
EX1	23	0.08	22	0.07	48	0.03
AFIRO	20	0.05	18	0.04	6	0.01
ADLITTLE	24	0.13	22	0.14	119	0.23
SHARE2B	29	0.32	28	0.33	119	0.26
ISRAEL	37	4.01	42	4.31	281	1.54
E226	34	1.58	42	1.62	568	3.68
BANDM	39	1.87	45	1.83	356	2.87
SHIP08	32	1.35	30	1.32	657	9.16
SCSD8	23	1.82	22	1.78	1590	14.53

The number of iterations (itrs. in Table 2 and Table 3) required by these interior point algorithms (Algorithms A, B, C) is small when compared to the simplex method and its variants (Algorithm E). The work per iteration of these interior point algorithms is more than the work required by the simplex method. Algorithm D has more number of iterations and CPU times than other algorithms.

These interior point algorithms attained 8 digit accuracy in the objective function on all test problems with finite number of iterations. The selection of an initial interior solution plays a significant role in the fast convergence. For small problems, the simplex method and its variant have less execution

times than these interior point algorithms. However, as problem size increase, the interior point algorithms seem to have less iteration numbers and execution times.

Since we used two different size of computers to test the computational results, but we do compared the variants of Karmarkar's algorithm with the variants of simplex method. For example, compared the Algorithms A and B with the simplex MINOS program in one computer, and compared Algorithm C with D and E in another computer.

Table 3. Computational Results for Algorithms C, D, and E

Problems	Algorithm C		Algorithm D		Algorithm E	
	Itrs.	CPU	Itrs.	CPU	Itrs.	CPU
EX1	29	0.20	35	0.18	40	0.07
AFIRO	22	0.12	15	0.15	6	0.05
ADLITTLE	30	0.38	54	0.49	106	0.41
SHARE2B	35	0.67	66	0.81	97	0.55
ISRAEL	45	7.52	137	6.25	225	3.63
E226	45	5.70	236	8.25	535	6.52
BANDM	52	4.31	187	7.85	312	4.75
SHIP08	39	3.38	241	16.94	572	18.74
SCSD8	26	5.05	302	25.37	1459	23.54

VII. CONCLUSION

Karmarkar's algorithm has variants which are interior point algorithms and allow one to run with low iteration counts and solve linear programs with upper bounds. Some interior point algorithms are nonlinear programming

approaches such as barrier methods. But box method algorithm makes no use of nonlinear programming approach.

Qualitative aspects (e.g., choice of α, θ, or ϵ) can be found in the interior point algorithms. The promise provided by these low iteration numbers and CPU times for larger size problems is encouraging to warrant further research into Karmarkar's algorithm and its variants.

VIII. ACKNOWLEDGEMENT

I would like to thank Ilan Adler, R. C. Monterio, and Richard Wright for their helpful insights, suggestions and programming supports during the course of this study. I would also like to thank the University of California at Berkeley, National Defense Management College and National Taiwan University for allowing me to use their computers.

IX. REFERENCES

[1] I. Adler, N. Karmarkar, M. G. C. Resende and G. Veiga, "Data Structures and Programming Techniques for the Implementation of Karmarkar's Algorithm," Working Paper, Operations Research Center, University of California, Berkeley, CA, 1987.

[2] E. R. Barnes, "A Variation on Karmarkar's Algorithm for Solving Linear Programming Problems," *Mathematical Programming* 36, 174-182, 1986.

[3] V. Chandru and B. S. Kochar, "A Class of Algorithms for Linear Programming," Research Memorandum 85-14, School of Industrial Engineering, Purdue University, West Lafayette, IN, 1986.

[4] W. H. Chen, "The New Algorithm for Linear Programming," *Mathmedia* 4, 2, 30-36, 1980.

[5] G. B. Dantzig, *"Linear Programming and Extensions,"* Princeton University Press, Princeton, NJ, 1963.

[6] K. R. Frisch, "The Logarithmic Potential Method of Convex Programming," unpublished manuscript, University Institute of Economics, Oslo, Norway, 1955.

[7] P. E. Gill, W. Murray, M. A. Saunders, J. A. Pomlin and M. H. Wright, "On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method," *Mathematical Programming* 36, 183-209, 1986.

[8] J. H. Hooker, "Karmarkar's Linear Programming Algorithm," *Interfaces* 16, 75-90, 1986.

[9] N. Karmarkar, "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica* 4, 373-395, 1984.

[10] L. G. Khachiyan, "A Polynomial Algorithm for Linear Programming," *Doklady Akad. Nauk USSR* 244, 1093-1096, 1979, Translated in *Soviet Math. Doklady* 20, 191-194, 1979.

[11] M. Kojima, S. Mizuno and A. Yoshise, "A Primal-Dual Interior Point Algorithm for Linear Programming," Report No. B-188, Department of Information Sciences, Yokyo Institute of Technology, Tokyo, Japan, 1987.

[12] I. J. Lustig, "A Practical Approach to Karmarkar's Algorithm," Technical Report SOL 85-5, Depattment of Operations Research, Stanford University, 1985.

[13] N. Meggido, "Pathways to the Optimal Set in Linear Programming," Research Reports IBM Almaden Research Center, San Jose, CA, 1986.

[14] R. C. Monteiro and I. Adler, "An $O(n^3 L)$ Primal-Dual Interior Point Algorithm for Linear Programming," Working Paper, Operations Research Center, University of California, Berkeley, 1987.

[15] J. Renegar, "A Polynomial-Time Algorithm Based on Newton's Method for Linear Programming," *Mathematical Programming*, 40, 59-93, 1988.

[16] M. J. Todd, "Improved Bounds and Containing Ellipsoids in Karmarkar's Linear Programming Algorithm," Manuscript, School of Operations

Research and Industrial Engineering, Cornell University, 1987.

[17] J. A. Tomlin, "An Experimental Approach to Karmarkar's Projective Method for Linear Programming," Manuscript, Ketron Inc., Mountain View, California, 1985.

[18] P. M. Vaidya, "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, N. J., 1987.

[19] Z. L. Wei, "An Exact Solution to Linear Programming Using an Interior Point Method," Working Paper, Operations Research Center, University of California, Berkeley, 1986.

[20] Y. Ye, "Eliminating Columns in the Simplex Method for Linear Programming," Technical Report SOL 87-14, Department of Operations Research, Stanford University, 1987.

[21] K. Zikan and R. W. Cottle, "The Box Method for Linear Programming," Technical Report SOL 87-6, Department of Operations Research, Stanford University, 1987.

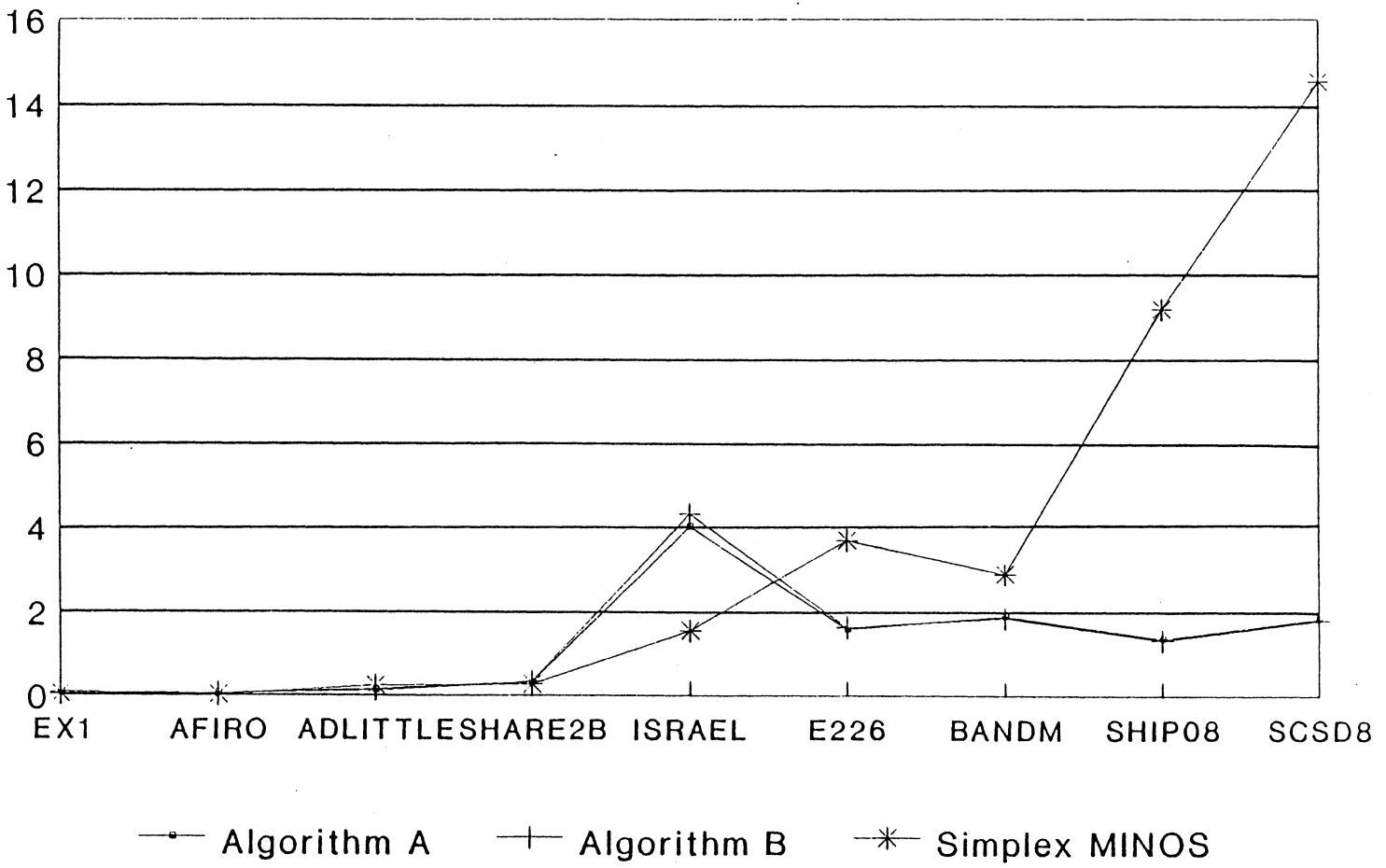

Figure 1. Computational Results for Algorithms A, B, and MINOS

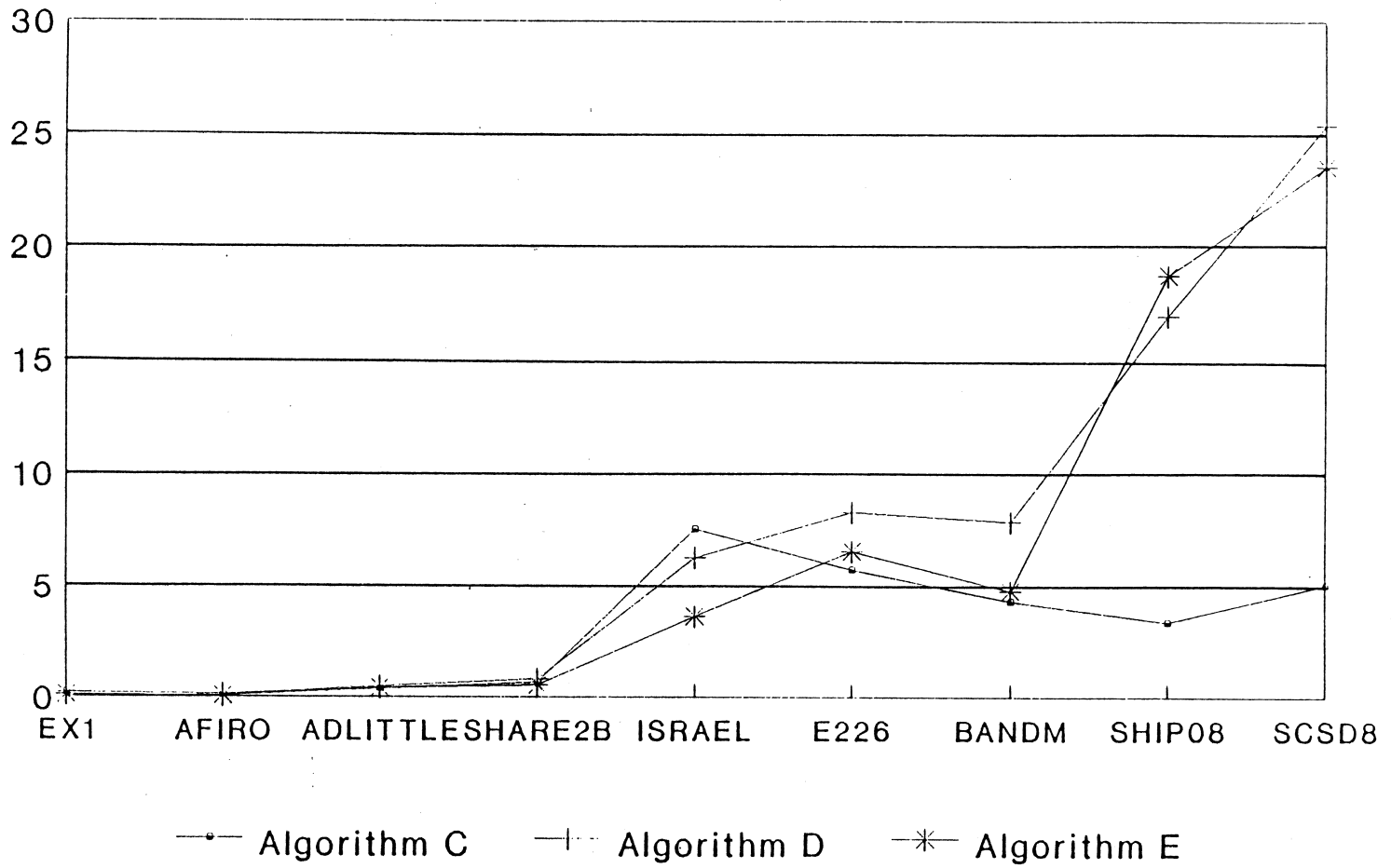

Figure 2. Computational Results for Algorithms C, D, and E

台大管理論叢
國立台灣大學管理學院
第一卷，第一期，中國民國七十九年五月

線性規劃 Karmarmar 相關演算之探討

陳 文 賢

摘 要

本論文主要是檢討並修改線性規劃 Karmarmar 相關演算法。這些演算法包括：內部點演算法，牛頓數值法，以及方塊法。這些方法對解線性規劃均有多項式求解時間。我們討論並改進這些方法，使求解速度更快。一共提出六套演算法，並寫成電腦程式。利用一些現有的線性規劃問題資料，在電腦上比較其求解速度。