Proc. 45th International Congress of the Indian Society of Theoretical and Applied Mechanics (2000) 88-96

16

Karmarkar Form of Linear Program and Algorithm: Precise Presentation

S.K. Sen[1] and S. Sen[2]

Abstract A linear program (LP), can be defined as *Minimize (Min)* $z = \mathbf{c}^t\mathbf{x}$ *subject to* $A\mathbf{x} \leq \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$, where A is an $m \times n$ matrix and $\mathbf{0}$ is the n-dimensional column vector (n-vector) of 0s. A form of LP equivalent to the foregoing LP and an algorithm (for this form), both due to N. Karmarkar, are presented here precisely and concisely. This Karmarkar form of LP (KLP) is *Min* $z = \mathbf{c}^t\mathbf{x}$ *subject to* $A\mathbf{x} = \mathbf{0}$, $\mathbf{e}^t\mathbf{x} = 1$, $\mathbf{x} \geq \mathbf{0}$, $\mathbf{x} = e/n$ *is feasible, minimal z-value* = 0, where e is the n-vector of 1s. Both the form and the operational aspects of the algorithm presented here are more easily followed. The algorithm is readily implementable/programmable on a computer. The Karmarkar algorithm (KA) uses a transformation from projective geometry to create a set of transformed variables $\mathbf{y}$. This transformation f always transforms the current point into the centre of the feasible region in the space defined by the transformed variables. If f takes the point $\mathbf{x}$ into the point $\mathbf{y}$ then we write $f(\mathbf{x}) = \mathbf{y}$. The KA begins in the transformed space in a direction that tends to improve z without violating feasibility. This yields a point $\mathbf{y}^1$, close to the boundary of the feasible region, in the transformed space. The new point is $\mathbf{x}^1$ that satisfies $f(\mathbf{x}^1) = \mathbf{y}^1$. The procedure is iterated replacing $\mathbf{x}^0$ by $\mathbf{x}^1$ until the z-value for $\mathbf{x}^k$ is sufficiently close to 0. An intelligent implementation of KA, however, does need a deeper insight (into the algorithm) that avoids redundant/partial duplication of computation/codes and that possibly reduces the number of iterations. This projective transformation based polynomial-time interior-point iterative algorithm is claimed to be more efficient than the widely used exponential-time exterior-point iterative method called the simplex algorithm for large LPs. The simplex algorithm and its variations have been the most widely used methods in linear optimization for over three decades (sixties—eighties) and is still being extensively used certainly for small and medium LPs. The KA is increasingly finding its place in literature/textbooks on linear programming/operations research. It is also stimulating in terms of visualizing every derived mathematical step geometrically (maximum three dimensions can be visualized, higher dimensions are just straight-forward mathematical extensions and cannot be visualized) or achieving the desired geometrical path/destination using the appropriate mathematics. Thus, we believe that there is a scope for such a presentation for the readers who desire to get a quick feel about this landmark algorithm. A MATLAB program for the KA is appended for ready check and for a quick feel about its convergence.

[1] Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore 560 012, India; e-mail: sksen @serc.iisc.ernet.in; Fax: 091-80-3602648
[2] Department of Computer Science and Engineering, Dr. Ambedkar Institute of Technology, Bangalore 560 056, India; e-mail: sagarsen@usa.net

1. Notations

We use the following convention and notations. A bold lower case letter (such as **c**, **b**, **x**) always indicates a column vector. A bold zero, viz., **0**, denotes a null column vector (i.e., a column vector of 0s) of appropriate order (including the order 1). An upper case letter (such as A, P) denotes a matrix and t, when used as a superscript, indicates the transpose. The specific symbols used here have the following meaning.

Symbol	*Meaning*
A	an $m \times n$ matrix $[a_{ij}]$
c	an n-dimensional vector or simply n-vector $[c_i] = [c_1\ c_2 \,..\, c_n]^t$
b	an m-vector $[b_j] = [b_1\ b_2 \,..\, b_m]^t$
e	a vector $[1\ \ 1\ \,..\ 1]^t$ of appropriate order
s	an m-vector $[s_j] = [s_1\ \ s_2 \,..\, s_m]^t$ of slack variables
v	an n-vector $[v_i] = [v_1\ \ v_2 \,..\, v_n]^t$ of surplus variables
x	an n-vector $[x_i] = [x_1\ \ x_2\ \,..\, x_n]^t$
$\mathbf{x}^k$ or $\mathbf{y}^k$	k-th iterate of the vector **x** or **y**
x_u^k or y_u^k	k-th iterate of the u-th element of **x** or **y**
diag($\mathbf{x}^k$)	$n \times n$ diagonal matrix whose (i,i)-th element is x_i^k
$j = 1(1)n$	$j = 1, 2, .., n$
$\|\ \|$	Euclidean norm
α	a real positive number < 1
x, y, s $\geq$ **0**	$\mathbf{x} \geq \mathbf{0}, \mathbf{y} \geq \mathbf{0}, \mathbf{s} \geq \mathbf{0}$
Min (Max)	Minimize (Maximize)
X^+	minimum-norm least-squares inverse (p-inverse) of the matrix X

2. Introduction

A linear programming problem or, equivalently, a linear program (LP) is defined, in the standard form, as

$$Min\ \mathbf{c}^t\mathbf{x}\ \ subject\ to\ \ A\mathbf{x} = \mathbf{b},\ \ \mathbf{x} \geq \mathbf{0}. \quad (1)$$

The LP (1) is solved by the simplex method/revised simplex method/a variation of the simplex method (exterior-point method) designed and developed by G. Dantzig during early 1950s [Dantzig 1963, Beale 1955, 1968, Luenberger 1973, Gass 1975, Vajda 1974, Murty 1976, Krishnamurthy and Sen 2000]. This method dominated the linear programming scene solving millions of optimization problems in almost all scientific and engineering areas. However, considerable amount of research went into this area and many special-purpose algorithms were designed and used with a significant success. All these algorithms are exponential-time (nonpolynomial-time). The simplex method is *exponential-time* in the worst case. This implies that if an LP of size[1] n is solved by the simplex method, there exists a positive number *p* such that for any n, an LP of size n

[1] The size of an LP could be defined as the number of symbols needed to represent the LP in binary notation.

can be sölved in at most $p2^n$ operations. The simplex mehtod may even enter into a cycling (infinite loop) though very rarely [Beale 1955]. Efforts to develop a polynomial-time algorithm for LPs did not meet with any success till almost the end of 1970s. In 1979, L.G. Khachiyan reported the first known interior-point iterative algorithm called the Ellipsoid method [Khachiyan 1979] to solve LPs. Then, in 1984, N. Karmarkar proposed the second polynomial-time $O(n^{3.5})$ interior-point iterative method [Karmarkar 1984, Hooker 1986, Murty 1989, Winston 1994] based on a projective transformation.

We provide here, without proof, illustration, and explanation, the conversion of any LP to the Karmarkar form of LP (KLP) in Sec. 3. For the proof and illustration, one could refer the foregoing concerned references. In Sec. 4, we present Karmarkar Algorithm (KA) precisely and concisely so that one could simply solve an LP just by mechanically following the steps. Here also we omit the proof as well as much explanation which are available in Karmarkar's paper [Karmarkar 1984]. Conclusions appear in Sec.5. A MATLAB Version 5.1 program for the KA is appended for ready verification and feel about the algorithm.

3. Conversion of an LP to KLP

A standard LP (constraints in an equality form) or any LP whose constraints are in an inequality form can be converted to a KLP as follows.

Consider the LP

$$Max\ z = \mathbf{c}^t\mathbf{x} \ \ subject\ to\ \ A\mathbf{x} \le \mathbf{b},\ \mathbf{x} \ge \mathbf{0}. \quad (2)$$

The dual of LP (2) is

$$Min\ \ w = \mathbf{b}^t\mathbf{y} \ \ subject\ to\ A^t\mathbf{y} \ge \mathbf{c},\ \mathbf{y} \ge \mathbf{0}. \quad (3)$$

From the duality theorem, we know that if the n-vector $\mathbf{x}$ is feasible in (2), the m-vector y is feasible in (3), and the z-value in (2) equals the w-value in (3) then $\mathbf{x}$ is maximal for (2). This implies that any feasible solution of the following set of constraints will produce the maximal solution of (2).

$$\mathbf{c}^t\mathbf{x} - \mathbf{b}^t\mathbf{y} = 0,\ A\mathbf{x} \le \mathbf{b},\ A^t\mathbf{y} \ge \mathbf{c},\ \ \mathbf{x}, \mathbf{y} \ge \mathbf{0} \quad (4)$$

Inserting slack and surplus variables into (4) we get

$$\mathbf{c}^t\mathbf{x} - \mathbf{b}^t\mathbf{y} = 0,\ A\mathbf{x} + I_m\mathbf{s} = \mathbf{b},\ A^t\mathbf{y} - I_n\mathbf{v} = \mathbf{c},\ \ \mathbf{x}, \mathbf{y}, \mathbf{s}, \mathbf{v} \ge \mathbf{0}, \quad (5)$$

where $\mathbf{s} = [\ s_1\ \ s_2\ \ ..\ s_m]^t$ is the m-vector of slack variables, $\mathbf{v} = [\ v_1\ \ v_2\ \ ..\ v_n]^t$ is the n-vector of surplus variables, I_m is the unit matrix of order m, and I_n is the unit matrix of order n. We now append to (5) yet another constraint such that the feasible solution of (5) satisfies the equation

$$e^t x + e^t y + e^t s + e^t v + d_1 = k, \qquad (6)$$

where k is to be found/supplied such that the sum of the values of all the variables $\leq$ k. The variable $d_1 \geq 0$ is a dummy (slack) variable. This yields

$$\mathbf{c}^t\mathbf{x}-\mathbf{b}^t\mathbf{y}=0,\ \ Ax+I_m s=b,\ \ A^t y-I_n v=c,\ \ e^t x+e^t y+e^t s+e^t v+d_1=k,\ \ \mathbf{x},\ \mathbf{y},\ \mathbf{s},\ \mathbf{v},\ d_1 \geq \mathbf{0},\ \ (7)$$

To make nonzero right-hand sides of (7) zero, we introduce yet another dummy variable d_2, where $d_2 = 1$. Thus, we obtain

$$\begin{gathered}\mathbf{c}^t\mathbf{x}-\mathbf{b}^t\mathbf{y}=0,\ \ A\mathbf{x}+I_m\mathbf{s}-I_m\mathbf{b}d_2=\mathbf{0},\ \ A^t\mathbf{y}-I_n\mathbf{v}-I_n\mathbf{c}d_2=\mathbf{0},\ \ \mathbf{e}^t\mathbf{x}+\mathbf{e}^t\mathbf{y}+\mathbf{e}^t\mathbf{s}+\mathbf{e}^t\mathbf{v}+d_1-kd_2=0,\\ \mathbf{e}^t\mathbf{x}+\mathbf{e}^t\mathbf{y}+\mathbf{e}^t\mathbf{s}+\mathbf{e}^t\mathbf{v}+d_1+d_2 = k + 1,,\ \ \mathbf{x},\ \mathbf{y},\ \mathbf{s},\ \mathbf{v},\ d_1,\ d_2 \geq \mathbf{0} \qquad ..\ \ (8)\end{gathered}$$

Allowing the following change of variables

$$\mathbf{x}=(k+1)\mathbf{x}',\ \ \mathbf{y}=(k+1)\mathbf{y}',\ \ \mathbf{s}=(k+1)\mathbf{s}',\ \ \mathbf{v}=(k+1)\mathbf{v}',\ \ d_1=(k+1)d_1',\ \ d_2=(k+1)d_2'$$

we obtain

$$\begin{gathered}\mathbf{c}^t\mathbf{x}'-\mathbf{b}^t\mathbf{y}'=0,\ \ A\mathbf{x}'+I_m\mathbf{s}'-I_m\mathbf{b}d_2'=\mathbf{0},\ \ A^t\mathbf{y}'-I_n\mathbf{v}'-I_n\mathbf{c}d_2'=\mathbf{0},\ \ \mathbf{e}^t\mathbf{x}'+\mathbf{e}^t\mathbf{y}'+\mathbf{e}^t\mathbf{s}'+\mathbf{e}^t\mathbf{v}'+d_1'-kd_2'=0,\\ \mathbf{e}^t\mathbf{x}'+\mathbf{e}^t\mathbf{y}'+\mathbf{e}^t\mathbf{s}'+\mathbf{e}^t\mathbf{v}'+d_1'+d_2' = 1,\ \ \mathbf{x}',\ \mathbf{y}',\ \mathbf{s}',\ \mathbf{v}',\ d_1',\ d_2' \geq \mathbf{0} \qquad ..\ \ (9)\end{gathered}$$

We now enforce that a solution (geometrically, a point in [2n + 2m + 2] dimensional polytope [Lord et al 1996] defined by (9)) that sets all variables equal is feasible in (9). This is achieved by adding the third dummy variable d_3' to the last but one constraint in (9) and then adding a multiple of d_3' to each of its preceding constraints. This multiple is chosen so that the sum of the coefficients of all variables in each constraint (except the last two) equals zero. This yields KLP (10).

Min d_3' *subject to*

$$\begin{gathered}\mathbf{c}^t\mathbf{x}'-\mathbf{b}^t\mathbf{y}' - (\mathbf{e}^t\mathbf{c}-\mathbf{e}^t\mathbf{b})d_3'=0,\ \ A\mathbf{x}'+I_m\mathbf{s}'-I_m\mathbf{b}d_2' - [A\mathbf{e} + I_m(1-d_2')\mathbf{e}]d_3'=\mathbf{0},\\ A^t\mathbf{y}'-I_n\mathbf{v}'-I_n\mathbf{c}d_2'-[A^t\mathbf{e}-I_n(1-d_2')\mathbf{e}]d_3'=\mathbf{0},\ \ \mathbf{e}^t\mathbf{x}'+\mathbf{e}^t\mathbf{y}'+\mathbf{e}^t\mathbf{s}'+\mathbf{e}^t\mathbf{v}'+d_1'-kd_2'-(2n+2m+1-k)d_3'=0,\\ \mathbf{e}^t\mathbf{x}'+\mathbf{e}^t\mathbf{y}'+\mathbf{e}^t\mathbf{s}'+\mathbf{e}^t\mathbf{v}'+d_1'+d_2' + d_3'= 1,\ \ \mathbf{x}',\ \mathbf{y}',\ \mathbf{s}',\ \mathbf{v}',\ d_1',\ d_2',\ d_3' \geq \mathbf{0} \qquad ..\ \ (10)\end{gathered}$$

Observe that we cannot write the expression $\mathbf{e}^t\mathbf{x}'+\mathbf{e}^t\mathbf{y}'+\mathbf{e}^t\mathbf{s}'+\mathbf{e}^t\mathbf{v}'$ as $\mathbf{e}^t(\mathbf{x}'+\mathbf{y}'+\mathbf{s}'+\mathbf{v}')$ since the order of $\mathbf{e}^t$ differs from $\mathbf{x}'$ to $\mathbf{y}'$, in general. In the KLP (10) the solution (point) $[x_1'\ x_2' \ldots x_n'\ \ y_1'\ y_2' \ldots y_m'\ \ s_1'\ s_2' \ldots s_n'\ \ v_1'\ v_2' \ldots v_m'\ \ d_1'\ \ d_2'\ \ d_3']^t = (1/(2n+2m+3))\mathbf{e}^t$ is feasible. Since d_3' should be zero in a feasible solution of (9), we need to minimize d_3' in (10). If (9) is feasible then the minimum value of d_3' in KLP (10) will be zero and the remaining 2n+2m+2 variables in a minimal solution of (10) will give a feasible solution to (9). The values of $x_1, x_2, \ldots, x_n$ in the minimal solution of (10) will produce an optimal solution of the original LP (2). The KLP (10) is now ready for solution by the KA.

Example Consider the LP *Max* $c^t x$ *subject to* $Ax \le b$, $x \ge 0$, where

$$A = \begin{bmatrix} 1 & 2 & 1 \\ -4 & -2 & 3 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad c = \begin{bmatrix} -2 \\ -7 \\ 2 \end{bmatrix}, \quad x = [x_1\ x_2\ x_3]^t,\ m = 2,\ \ n = 3.$$

From KLP (10), we have, choosing k=20 (conservatively) and setting $x_j = 21x_j'$ $j = 1(1)n$, $y_i = 21y_i'$ $i = 1(1)m$, $s_i = 21s_i'$ $i = 1(1)m$, $v_j = 21v_j'$ $j = 1(1)n$, $d_1 = 21d_1'$, $d_2 = 21d_2'$,

Min d_3' *subject to*

$$\begin{bmatrix} -2 & -7 & 2 & -1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \\ 1 & 2 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & -4 \\ -4 & -2 & 3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -2 & 4 \\ 0 & 0 & 0 & 1 & -4 & 0 & 0 & -1 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 2 & -2 & 0 & 0 & 0 & -1 & 0 & 0 & 7 & -6 \\ 0 & 0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & -1 & 0 & -2 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -20 & 9 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1' \\ x_2' \\ x_3' \\ y_1' \\ y_2' \\ s_1' \\ s_2' \\ v_1' \\ v_2' \\ v_3' \\ d_1' \\ d_2' \\ d_3' \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

All variables ≥ 0.

The foregoing LP is the required KLP for the KA. Thus, without any confusion or loss of generality, the general form of KLP can be written as

Min $z = \mathbf{c}^t\mathbf{x}$ *subject to* $A\mathbf{x} = \mathbf{0}$, $\mathbf{e}^t\mathbf{x} = 1$, $\mathbf{x} \ge \mathbf{0}$, $\mathbf{x} = e/n$ *is feasible*, minimal z-value = 0, (11)

where the matrix A is $m \times n$. We will be using this general form for the KA.

4. The Karmarkar Algorithm (KA)

Consider the KLP (11). Assume that a feasible solution having a minimal z-value < ε (ε is a small positive value compared to the average element of A, **b**, **c**) is acceptable. The KA is then as follows.

Step 1 Input A, **b**, **c**, m, n. Set n-vector $\mathbf{e} = [1\ \ 1 \ldots 1]^t$.

Step 2 Set the feasible point (solution) $\mathbf{x}^0 = e/n$, the iterate k = 0.

Step 3 If $\mathbf{c}^t\mathbf{x}^k < \varepsilon$ then stop else go to Step 4.

Step 4 Compute the new point (an n-vector) y^{k+1} in the transformed n-dimensional unit simplex S (S is the set of points y satisfying $e^t y = 1$, $x \geq 0$) given by

$$y^{k+1} = x^0 - \alpha c_p / [\sqrt{(n(n-1))} \|c_p\|],$$

where

$$c_p = (I_n - P^t(PP^t)^+ P)[\mathrm{diag}(x^k)]c, \quad P = \begin{bmatrix} A[\mathrm{diag}(x^k)] \\ e^t \end{bmatrix}, \quad 0<\alpha<1.$$

$\alpha = 0.25$ is known to ensure convergence. P is the $(m+1) \times n$ matrix whose last row e^t is a vector of 1s. $(PP^t)^+$ is the p-inverse (Lakshmikantham et al 1996) of the matrix PP^t.

Step 5 Compute now a new point x^{k+1} in the original space using the Karmarkar Centring transformation to determine the point corresponding to the point y^{k+1}:

$$x^{k+1} = q/(e^t q),$$

where

$$q = [\mathrm{diag}(x^k)]y^{k+1}.$$

Increase k by 1 and return to Step 3.

Remark The computation of x^{k+1} in Step 5 may equivalently be written as $x_j^{k+1} = x_j^k y_j^{k+1} / \Sigma(x_t^k y_t^{k+1})$ $j = 1(1)n$, where the summation runs from $t = 1$ to n.

Example Consider the example of Sec. 3. If we now call the 8×13 matrix A, the left-hand side 13- vector **x**, and the right-hand side 8-vector **b** then the KA gives us, in the first iteration,

$$y^1 = [.0672\ .0683\ .0701\ .0753\ .0709\ .0706\ .0770\ .0692\ .0824\ .0733\ .0769\ .0750\ .0781]^t,$$

$$x^1 = [.0068\ 0\ .0408\ .0136\ .0272\ 0\ 0\ 0\ .3061\ 0\ .5578\ .0476\ 0]^t.$$

To obtain 4 decimal places accuracy in the elements of **x**, we need to go up to 1247 iterations. Thus, retaining the elements of **x** correct up 4 places, we have

$$x^{1247} = [.0068\ 0\ .0407\ .0139\ .0272\ .0003\ .0001\ .0006\ .3066\ .0001\ .5560\ .0476\ 0]^t.$$

Observe that here $x = [x_1'\ x_2'\ x_3'\ y_1'\ y_2'\ s_1'\ s_2'\ v_1'\ v_2'\ v_3'\ d_1'\ d_2'\ d_3']^t$. Hence $x_1' = .0068$, $x_2' = 0, \ldots, d_3' = 0$. Thus the required (true) solution correct up to 3 places is, noting that $k = 20$, $x_1 = 21x_1'$, $x_2 = 21x_2', \ldots, d_2 = 21d_2'$,

$$[x_1\ x_2\ x_3\ y_1\ y_2\ s_1\ s_2\ v_1\ v_2\ v_3\ d_1\ d_2\ d_3']^t$$
$$= [.143\ 0\ .855\ .292\ .571\ .001\ .003\ .012\ 6.439\ .002\ 11.675\ 1\ .001]^t.$$

Sum(sum(abs(a))) = sum of all elements of the matrix a..

Sum ([illegible]

5. Conclusions

Need for d_3'. It is not readily seen *a priori* that the original LP is feasible. If it is known that the LP is feasible then we need not bring d_3' in the KA at all. If the LP is not feasible due to inconsistency in the constraints and we do not use d_3' then we will end up getting incorrect solution. While the simplex algorithm needs artificial variables to tackle/detect inconsistency in the constraints, the KA needs d_3'.

Enhanced dimension of KLP If the original LP is in an inequality form ($A\mathbf{x} \leq \mathbf{b}$) then the corresponding KLP will have $2(n+m)+3$ variables where A is $m \times n$. Clearly there has been an increase of $n+2m+3$ variables (and hence the increase in the dimension of the polytope defined by $Ax \leq \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$) over the original LP. If, on the other hand, the original LP is in an equality form ($Ax = \mathbf{b}$) then the corresponding KLP will have relatively small dimension.

Non-feasibility of error-free computation The KA needs the computation of $\sqrt{(n(n-1))}$ which cannot be computed exactly, in general. Hence, unlike simplex and other methods [Lakshmikantham et al 1993, 1997, 2000], the KA is not amenable to error-free computation.

Polynomial-time noniterative algorithm – an open problem The KA is polynomial-time iterative needing clearly too many iterations compared to the simplex algorithm. A mathematically noniterative (direct) polynomial algorithm for an LP is still an open problem. However, a heuristic direct polynomial algorithm which is significantly useful in solving many real world LPs does exist [Sen and Ramful 2000]. It may be seen that the nonnegativity condition ($\mathbf{x} \geq \mathbf{0}$) is the real difficulty in the way of developing direct algorithm.

Parallel implementation The KA is relatively easy to be implemented/programmed on a parallel machine unlike the simplex method.

General Observe that *max* $\mathbf{c}^t\mathbf{x}$ is the same as *min* $-\mathbf{c}^t\mathbf{x}$. There has been a surge of interest among scientists/operations researchers to relook into the LP after the publication of the KA in 1984 [Renegar 1988, Lord et al 1996, Hooker 1986, Barnes 1986, Sen et al 1995, Murty 1989]. Consequently, there have been several inerior-point polynomial-time iterative algorithms (which are indeed excellent) reported in the literature. We do feel that a through conceptual knowledge of the KA, specifically from the geometrical point of view, is not only refreshing and enjoyable but also an important basis for further research in linear optimization.

Appendix
MATLAB Program for Karmarkar Algorithm (KA)

A MATLAB 5.1 version program for the KA is presented below for the reader to readily check the algorithm for different kind of LPs including extreme ones (not large) and get a feel of it. No effort has been made to make the program more efficient so as to differ from the KA presented here. Observe that a MATLAB program is not meant to solve really large LPs. The inputs to this program are A, b, c, k (a parameter that differs from problem to problem), m, n.

```
function[ ] = karmarkar(A,b,c,k,m,n);
%A is an  mxn  matrix; e=n-vector of 1s needed later.
%This is KA for the LP  min  z= c^tx s.t. Ax=0, x>=0, e^tx=1, x=e/n is feasible,
%minimal z-value=0.   k = 20  here. k differs from problem to problem.
e=ones(n,1); x0=e/n; x=x0; alp=0.25; I=eye(n); eps=0.00005; n2=sqrt(n(n-1));  <- Replace by *
%eps=0.00005 should be replaced by eps=0.00005*(average of the elements of A,b, & c)
%for 4 significant digit accuracy in the solution (not the true solution) by KA.

for j=1:3000
    if (k+1)* c'*x<eps
        string ' eps, iteration no., x '
        eps, j, x'
    break
    end
P=[A*diag(x);e'];
cp=(I-P'*pinv(P*P')*P)*diag(x)*c;
y=x0-alp*cp/(n2*norm(cp));
q=diag(x)*y;
x=q/(e'*q);
string 'The iteration no. and solution are'
j, x'
end;
xt=(k+1)*x;
string 'The iteration no. and true solution are'
j, xt'
string 'The epsilon producing 4 significant digit accuracy is'
eps
```

References

1. Dantzig, G., Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey, 1963.
2. Beale, E.M.L., Cycling in the dual simplex algorithm, *Naval Research Logistics Quartly*, **2**, 1955, 269-275.
3. Beale, E.M.L., Mathematical Programming in Practice, Pitman, London, 1968.
4. Luenberger, D.G., Introduction to Linear and Nonlinear Programming, Addison-Wesley, Reading, Massachusetts, 1973.
5. Gass, S.I., Linear Programming: Methods and Applications, McGraw-Hill, New York.
6. Vajda, S., Theory of Linear and Nonlinear Programming, Longman, London, 1974.
7. Murty, K.G., Linear and Combinatorial Programming, Wiley, New York, 1976.
8. Murty, K.G., Linear Complementarity, Linear and Nonlinear Programming, Verlag, Berlin, 1989.
9. Krishnamurthy, E.V.; Sen, S.K., Numerical Algorithms: Computations in Science and Engineering, Affiliated East West Press, New Delhi, 2000.

```
* eps=0.00005*sum(sum(abs(A)))*sum(abs(b))
        *sum(abs(c))/((m*n
```

10. Khachiyan, L.G., A polynomial algorithm in linear programming, *Doklady Akademia Nauk SSSR*, 244, 1979, 1093-1096.
11. Karmarkar, N., A new polynomial-time algorithm in linear programming, Technical Report, AT & T Bell Labs., New Jersey, 1984, also *Combinatorica*, **4**, 1984, 373-395.
12. Hooker, J.N., Karmarkar's linear programming algorithm, *Interfaces*, **16**(4), 1986, 75-90.
13. Winston, W.L., Operations Research: Applications and Algorithms, Duxbury Press, California, 1994.
14. Sen, S.K.; Ramful, A., A direct heuristic algorithm for linear programming, *Proc. Indian Acad. Sci. (Math. Sci.)* **110** (1), 2000, 79-101.
15. Lakshmikantham, V.; Sen, S.K.; Jain, M.K.; Ramful, A., O(n^3) noniterative heuristic algorithm for linear programs with error-free implementation, *Applied Maths. Comput.*, **110**, 2000, 53-81.
16. Lakshmikantham, V.; Sen, S.K.; Sivasundaram, S., An inequality sorting algorithm for a class of linear programming problems, *J. Math. Anal. Appl.*, **174**(2), 1993, 450-460.
17. Lakshmikantham, V.; Sen, S.K.; Howell, G., Vectors versus matrices: p-inversion, cryptographic application, and vector implementation, *Neural, Parallel, and Scientific Computations*, **4**, 1996, 129-140.
18. Lord, E.A.; Venkaiah, V. Ch.; Sen, S.K., A shrinking polytope method for linear programming, *Neural, Parallel and Scientific Computations*, **4**, 1996, 325-340.
19. Lakshmikantham, V.; Maullo, A.K.; Sen, S.K.; Sivasundaram, S., Solving linear programming problems exactly, *Appl. Maths. Comput.*, **81**, 1997, 69-80.
20. Renegar, J., A polynomial-time algorithm, based on Newton's method for linear programming, *Math. Prog.*, **40**, 1988, 59-93.
21. Barnes, E.R., A variation of Karmarkar's algorithm for solving linear programming problems, *Math. Program.*, **36**, 1986, 174-182.
22. Sen, S.K.; Sivasundaram, S.; Venkaiah, V.Ch., Barnes' algorithm for linear programming: on detection of basic variables, *Nonlinear World*, **2**, 1995.