Proc. Indian Acad. Sci. (Math. Sci.), Vol. 106, No. 1, February 1996, pp. 69–77.

Deriving Karmarkar's LP algorithm using angular projection matrix

V Ch VENKAIAH

R & D Group, Tata Elxsi (India) Ltd., Bangalore 560025, India
Present address: Motorola India Electronics Pvt. Ltd., Bangalore 560042
Permanent address: Department of Mathematics, Indian Institute of Technology, Delhi 110016

MS received 28 June 1993; revised 26 December 1994

Abstract. Understanding Karmarkar's algorithm is both desirable and necessary for its efficient implementation, for further improvement and for carrying out complexity analysis. In this report an algorithm based on the concept of angular projection matrix, to solve linear programming problems is derived. Surprisingly, this algorithm coincides with the affine version of Karmarkar's algorithm.

Keywords. Karmarkar's algorithm; angular projection matrix.

1. Introduction

The quest for solution of general optimization problems led to applications of the differential calculus and to the development of calculus of variations. Optimization problems are of much interest because of their occurrence in numerous situations that arise in government, military, industry and in economic theory.

Classical optimization techniques have been successfully employed in solving some of the optimization problems that are encountered in the physical sciences and engineering but are not found to be amenable to many new and important optimization problems that have emerged in the field of economics.

In 1947 Dantzig had developed an algorithm known as simplex algorithm that can be applied to a special but very important class of optimization problems known as linear programming problems. But the simplex algorithm has exponential time complexity in its worst case. In 1979 Khachiyan had given a polynomial time algorithm [4] known as ellipsoid algorithm, for linear programming problems which is theoretically interesting but not superior to simplex algorithm in practice. More recently Karmarkar has given a polynomial time algorithm [2] based on projective transformation technique for linear programming problems that has not only theoretical interest, but is claimed to be superior even in practice. This result has brought about a resurgence of interest in linear programming.

Several papers have appeared since the inception of Karmarkar's algorithm in 1984. All these articles dealt with either variation, extension or application of Karmarkar's algorithm. But none of them, at least to the best of our knowledge, have attempted to unearth the sequence of ideas that led to Karmarkar's algorithm. This is important because there is a consensus among the researchers that many crucial details of the implementation of Karmarkar's algorithm are not available. In this report, a sequence of algorithms representing the sequence of ideas that led to Karmarkar's algorithm, starting with an algorithm that solves the trivial linear programming problem wherein only the objective function and positivity constraints are present, have been derived. Each algorithm is a modified version of the previous one in the sequence. Each

modification is based on the observations that are listed. The final algorithm in the sequence can be seen to coincide with the affine version of Karmarkar's algorithm [1]. Therefore, these algorithms will aid in understanding the implementation details of the Karmarkar's algorithm.

We address the trivial linear programming problem in § 2 and propose an iterative algorithm to solve this problem. 'Angular' projection matrix for the general linear programming problem is discussed in § 3. Special linear programming problem, considered in § 4, can be seen to have commonalities with the problem considered by Karmarkar. Finally, the algorithm that we propose in § 5 can be identified with the affine version of Karmarkar's algorithm.

2. Trivial problem

Consider the following problem

$$\text{minimise} \quad c^t x$$

$$\text{subject to} \quad x \geqslant 0.$$

This problem is trivial because the solution can be easily obtained by setting those components of x to zero that correspond to positive components of c and other components of x to infinity. For example, if $c = [1 \ -1]^t$ then set $x_1 = 0$ and $x_2 = \infty$. We will then have either minimum or infimum depending on whether the solution is bounded or unbounded. We differ from the usual practice and propose an iterative method to solve the above problem. Given an initial feasible solution $x^0 > 0$ the iterative method converges to an optimal solution. At each iteration this method calculates a new direction vector, by 'angularly' projecting the cost vector. By 'angular' projection we shall mean projection that deviates from orthogonal projection by an amount which in our case is dependent on the distances of the currently computed point from the hyperplanes that make up the feasible region. Therefore angular projection matrix for the trivial case is computed using the distances from the axes to the currently computed point. The iterative method is given in the following algorithm. Graphical explanation of the algorithm is given in figure 1.

2.1 *Algorithm A1*

step–1 Select an initial feasible solution $x^0 > 0$.
step–2 Set $y = c$ and $k = 0$.
step–3 Compute

$$\lambda = \min \left\{ \frac{x_i^k}{y_i} : y_i > 0 \right\}$$

step–4 Compute

$$x^{k+1} = x^k - \varepsilon \lambda y$$

where $0 < \varepsilon < 1$.
step–5 If the optimum is achieved then stop.
step–6 Set $k = k + 1$, $D_k = \text{diag}\,(x_1^k x_2^k x_3^k \ldots x_n^k)$ is a diagonal matrix with $x_i^k, i = 1, 2, \ldots, n$ as diagonal entities.
step–7 Compute $y = D_k c$ and go to step 3

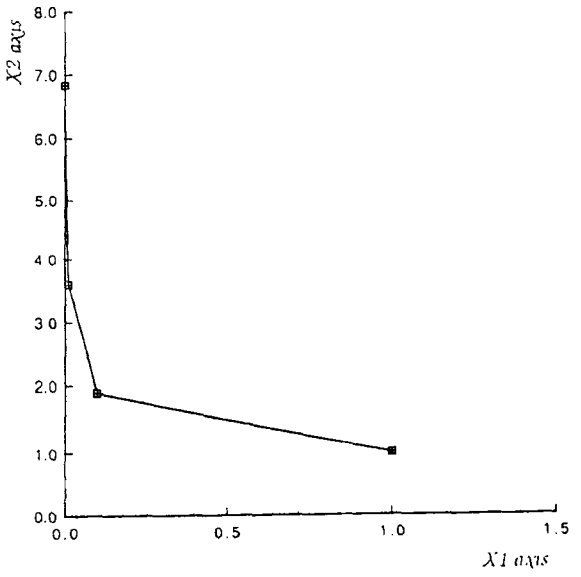

Figure 1. Algorithm A1 applied to example 1.

Initially the direction vector is set to the cost vector in step 2. Step 3 can be seen to compute the distance of the nearest axis from the point along the direction vector. Note that $-y$ is the actual direction vector because the cost function here is to be minimized. The new feasible solution is computed in step 4. If this feasible solution is an optimal solution then we stop. Otherwise in step 7, the direction vector is set to the angularly projected cost vector. Note that in the trivial case, D_k is an angular projection matrix. This fact is transparent from the following observations. The procedure is repeated by start computing λ. The algorithm is illustrated by the following numerical example.

$$\text{minimise} \quad x_1 - x_2$$

$$\text{subject to} \quad x_i \geqslant 0.$$

Note that $c^t = [1 \ -1]$. Let $x^0 = [1 \ 1]^t$ and $\varepsilon = 0{\cdot}9$. Then

$$y = [1 \ -1]^t, k = 0, \lambda = 1$$

$$x^1 = [1 \ 1]^t - 0{\cdot}9 * [1 \ -1]^t = [0{\cdot}1 \ 1{\cdot}9]^t$$

$$k = 1, D_1 = \text{diag}(0{\cdot}1 \ 1{\cdot}9)$$

$$y = [0{\cdot}1 \ -1{\cdot}9]^t, \lambda = 1$$

$$x^2 = [0{\cdot}1 \ 1{\cdot}9]^t - 0{\cdot}9 * [0{\cdot}1 \ -1{\cdot}9]^t = [0{\cdot}01 \ 3{\cdot}61]^t$$

$$k = 2, D_2 = \text{diag}(0{\cdot}01 \ 3{\cdot}61)$$

$$y = [0{\cdot}01 \ -3{\cdot}61]^t, \lambda = 1$$

$$x^3 = [0{\cdot}01 \ 3{\cdot}61]^t - 0{\cdot}9 * [0{\cdot}01 \ -3{\cdot}61]^t = [0{\cdot}001 \ 6{\cdot}859]^t$$

etc.

Observations

Let the general linear programming problem be in the following form

$$\begin{aligned} &\text{minimise} && c^t x \\ &\text{subject to} && Ax = b \\ & && x \geqslant 0. \end{aligned}$$

Also, let

$$Q_k = I - \frac{(1 - x_1^k)}{\|e_1\|^2} e_1 e_1^t - \cdots - \frac{(1 - x_n^k)}{\|e_n\|^2} e_n e_n^t,$$

where e_i is a column vector with n components one in the ith position and zeros elsewhere, I is the identity matrix and $x^k = (x_1^k, \ldots, x_n^k)$ is a feasible solution to the trivial linear programming problem. Then

1. Q_k is an angular projection matrix that projects a given vector angularly onto the coordinate axes.
2. The coefficient matrix A is identically zero in the trivial case and hence $P = I_n = [e_1 e_2 \ldots e_n] = I$ can be interpreted as the null space of A.
3. $Q_k P = P D_k = D_k$ in the trivial case.
4. D_k is a positive definite matrix.
5. Conventionally, if the currently computed solution x^k is on a set of bounding hyperplanes, which we call the set of touching hyperplanes of the feasible region, we project the direction vector (the orthogonal projection of c onto the null space of A, denoted by c_p) orthogonally onto the intersection of the set of touching hyperplanes and proceed to compute the next solution [5]. Note that this conventional procedure takes into account only those hyperplanes at zero distance from the point but not the remaining hyperplanes and their distances. Whereas the proposed algorithm computes $Q_k P c = P D_k c = D_k c$, angular projection of the direction vector; thereby giving weightage to all distances of the point from the hyperplanes. Therefore, if these observations are correctly translated to the general linear programming problem, it is reasonable to believe that the resulting algorithm will be computationally more efficient than the conventional algorithms. In our earlier paper [6] some of these observations were not correctly translated to the general case. Therefore the resulting algorithm inherited convergence problems, see [3].

3. Angular projection matrix for the general case

It appears that designing an algorithm for general linear programming problems parallel to algorithm A1 is fairly simple if we have a method to compute an angular projection matrix in the null space of A (the coefficient matrix of the constraint set $Ax = b$) which projects angularly a given vector and possesses the above observed properties. But computing an angular projection matrix Q that has these properties is equivalent to computing w_i's in the expression

$$Q = I - \frac{w_1}{P_{11}} P_1 P_1^t - \cdots - \frac{w_n}{P_{nn}} P_n P_n^t,$$

such that for some positive definite matrix W, $QP = PW$, where P is the orthogonal projection matrix that spans the null space of A. Note that $P_i^t P_i = P_{ii}$. P_{ii} is zero if the ith column of P, i.e. P_i, is zero. This happens if the ith coordinate axis is normal to the null space of A which is a special case and can be easily handled at the implementation level. Since P is symmetric and

$$QP = \left[P_1 - \sum_{i=1}^{n} \left(w_i \frac{P_{1i}}{P_{ii}} \right) P_i P_2 - \sum_{i=1}^{n} \left(w_i \frac{P_{2i}}{P_{ii}} \right) P_i \ldots P_n - \sum_{i=1}^{n} \left(w_i \frac{P_{ni}}{P_{ii}} \right) P_i \right]$$

$$= PW,$$

it follows that

$$W = \begin{pmatrix} 1 - w_1 & -w_1 \dfrac{P_{12}}{P_{11}} & \cdots & -w_1 \dfrac{P_{1n}}{P_{11}} \\ -w_2 \dfrac{P_{12}}{P_{22}} & 1 - w_2 & \cdots & -w_2 \dfrac{P_{2n}}{P_{22}} \\ \vdots & \vdots & \vdots & \vdots \\ -w_n \dfrac{P_{1n}}{P_{nn}} & -w_n \dfrac{P_{2n}}{P_{nn}} & \cdots & 1 - w_n \end{pmatrix}.$$

We require W to be positive definite. So, first it should be symmetric. Therefore, w_i's must satisfy the following relations:

$$\frac{w_1}{P_{11}} = \frac{w_2}{P_{22}} = \cdots = \frac{w_n}{P_{nn}}. \tag{1}$$

Since diagonal dominance together with symmetry is a sufficient condition for positive definiteness, choose w_i so that they satisfy (1) and so that the choice results in a diagonal dominance of W. Assuming $|P_{12}| < P_{22}$, $|P_{13}| < P_{33}, \ldots$ (which may not be the case), the choice of w_i such that $\sum_{i=1}^{n} w_i = 1$, results W to be diagonally dominant and hence in positive definite matrix. **But**

$$\sum_{i=1}^{n} w_i = 1$$

and the relations given in (1) imply

$$w_1 = \frac{P_{11}}{\sum P_{ii}}, \ldots, w_n = \frac{P_{nn}}{\sum P_{ii}}.$$

Observations

1. These latter expressions resemble the projective transformation of Karmarkar.
2. Unlike D in the trivial case, W is not a function of the distances of the hyperplanes from the currently computed solution. Hence if the current solution is closer or farther from a hyperplane, this is not reflected in the W matrix as it does in the corresponding D matrix of the trivial case.

Since there is a breakdown in our line of thinking, let us take a fresh look at the angular projection matrix expression and write down the requirements. Writing the

angular projection matrix

$$Q = I - \frac{1-w_1}{P_{11}} P_1 P_1^t - \cdots - \frac{1-w_n}{P_{nn}} P_n P_n^t,$$

we have the following.

Requirements

1. w_i should be function of x_i.
2. If all P_i are parallel (that is, if $P_1 = k_j P_j$ for all j and for some constant k_j) then we have

$$Q = I - [n - (w_1 + w_2 + \cdots + w_n)] \frac{P_1 P_1^t}{P_{11}}.$$

So, $w_1 + w_2 + \cdots + w_n = n$. Otherwise, the algorithm that we design based on this angular projection matrix may trace the same path again and again without actually improving the solution.

4. Algorithm for the special linear programming problem

If we choose $w_i = x_i$ then according to our discussion, the following linear programming problem

$$\begin{aligned} &\text{minimise} && c^t x \\ &\text{subject to} && Ax = b \\ & && \sum_{i=1}^{n} x_i = n \\ & && x \geqslant 0 \end{aligned}$$

that has from the second requirement an extra constraint $\sum x_i = n$, can be solved by the following procedure, in which x^0 is an initial feasible solution. Note that the equation $\sum x_i = n$ and the inequalities $x_i \geqslant 0$ specify a simplex. So, the set of constraints $Ax = b$, $\sum x_i = n$, $x_i \geqslant 0$ specify a feasible region which is the intersection of the simplex and the space specified by $Ax = b$. Observe the commonality between this problem and the problem considered by Karmarkar.

4.1 *Procedure*

step–1 Set $A = \begin{bmatrix} A \\ e \end{bmatrix}, b = \begin{bmatrix} b \\ n \end{bmatrix}$, and $k = 0$, where $e = [11 \ldots 1]$.

step–2 Compute $P = I - A^+ A$ and $c_p = Pc$, where A^+ is the Moore–Penrose inverse of A.

step–3 Compute

$$\lambda = \min \left\{ \frac{x_i^k}{c_{p_i}} : c_{p_i} > 0 \right\}$$

and set $k = k + 1$.

step–4 Compute

$$x^k = x^{k-1} - \varepsilon \lambda c_p$$

where $0 < \varepsilon < 1$.

step–5 Compute

$$Q = I - \frac{(1 - x_1^k)}{P_{11}} P_1 P_1^t - \cdots - \frac{(1 - x_n^k)}{P_{nn}} P_n P_n^t.$$

step–6 Compute $c_p = QPc$ and go to step 3.

Projection matrix P whose columns span the null space of A and the initial direction vector c_p, which is the orthogonal projection of the cost vector c onto the null space of A, is computed in step 2. Step 3 computes the distance of the nearest bounding hyperplane of the convex polytope that specify the feasible region from the point along the direction vector. The new feasible solution is computed in step 4. Step 5 computes the angular projection matrix and in the last step new direction vector for the next iteration is computed by angularly projecting Pc in the null space of A.

But computational experience shows that this algorithm does not converge to an optimal solution. For example the algorithm when applied to the problem

$$\begin{aligned} \text{minimise} \quad & x_1 + 2x_2 + 3x_3 + 4x_4 \\ \text{subject to} \quad & x_1 + x_2 + x_3 + x_4 = 4 \\ & x_i \geqslant 0 \end{aligned}$$

converges to [2·2322 1·2540 0·5138 0·0000] which is not an optimal solution. Optimal solution is [4 0 0 0].

However, the algorithm designed in the next section, based on this procedure, can easily be identified with the affine version of Karmarkar's algorithm and hence can solve the general linear programming problem.

5. Proposed algorithm

Assume that the procedure given in §4.1 converges to an optimal solution and hence can solve the previously mentioned special class of linear programming problems. Then the natural question that arises is the following. How can we extend this procedure to solve the general linear programming problems? A way around this problem can be found if we can find a transformation that converts a general linear programming problem to a special linear programming problem and vice-versa. A method that incorporates this idea is the following.

Let x^k denote the solution of the given linear programming problem, obtained at the kth iteration. Also, let $D_k = \text{diag}(x_1^k x_2^k \ldots x_n^k)$ be a diagonal matrix. If $\|x^k - x^{k-1}\| < 2^{-L}$, where L is the word length, then x^k is an optimal solution. Otherwise, use the transformation $x = D_k y$ and append a new constraint $\sum_{i=1}^{n} y_i = n$ to obtain the special linear programming problem

$$\begin{aligned} \text{minimise} \quad & (D_k c)^t y \\ \text{subject to} \quad & AD_k y = b \\ & \sum_{i=1}^{n} y_i = n \\ & y \geqslant 0 \end{aligned}$$

for which $y^0 = (1 \cdots 1)$ is an initial feasible solution. Use the previous procedure to

obtain a solution y^1, which may not be optimal, of the special linear programming problem. Then compute $x^{k+1} = D_k y_1$. Increment k by one and repeat the process. Stepwise, the method is given below.

step–1 Compute an initial feasible solution x^0 of the general linear programming problem and set $k = 0$.

step–2 Set $D_k = \text{diag}(x_1^k x_2^k \ldots x_n^k)$.

step–3 Set $A_d = \begin{bmatrix} AD_k \\ e \end{bmatrix}$, $j = 0$, and $y^j = [1 \cdots 1]$.

step–4 Compute $P = I - A_d^+ A_d$ and $c_p = PD_k c$.

step–5 Compute

$$\lambda = \min \left\{ \frac{y_i^j}{c_{p_i}} : c_{p_i} > 0 \right\}.$$

step–6 Compute

$$y^{j+1} = y^j - \varepsilon \lambda c_p$$

where $0 < \varepsilon < 1$ and set $j = j + 1$.

step–7 Compute

$$Q = I - \frac{(1 - y_1^j)}{P_{11}} P_1 P_1^t - \cdots - \frac{(1 - y_n^j)}{P_{nn}} P_n P_n^t.$$

step–8 Compute $c_p = QPD_k c$.

step–9 If $\| y^j - y^{j-1} \| \geqslant 2^{-L}$ go to step 5.

step–10 Compute $x^{k+1} = D_k y^j$ and set $k = k + 1$.

step–11 If $\| x^k - x^{k-1} \| \geqslant 2^{-L}$ go to step 2.

Steps 2, 3 and 4 transform the general linear programming problem to a special linear programming problem for which $y^0 = [1 \cdots 1]$ is an initial feasible solution, compute orthogonal projection matrix P that span the null space of A_d and compute initial direction vector c_p of the special linear programming problem. Steps 5 to 9 constitute the procedure described in §4.1 to solve the special linear programming problem. Step 10 computes an improved solution of the general linear programming problem from that of the special linear programming problem.

Observations

1. Numerical calculations reveal that the algorithm converges to an optimal solution.
2. Repetition of steps 5 to 9 is not contributing to significant improvement of the solution.
3. Computational requirement of the steps 5 to 9 is of the order of n^3, n is the number of columns of A.

These observations motivate us to design the following algorithm. This new algorithm does away with angular projection matrix and related computation. It can be readily-identified with the affine version of Karmarkar's algorithm [see § 1 of 1].

5.1 *The algorithm*

step–1 Compute an initial feasible solution x^0 of the given linear programming problem and set $k = 0$, $y_0 = e = [1 \ldots 1]$.

step–2 Set $D_k = \text{diag}(x_1^k\, x_2^k \ldots x_n^k)$.

step–3 Set $A_d = \begin{bmatrix} AD_k \\ e \end{bmatrix}$.

step–4 Compute $P = I - A_d^{+} A_d$ and $c_p = PD_k c$.

step–5 Compute

$$\lambda = \min\left\{\frac{1}{c_{p_i}} : c_{p_i} > 0\right\}.$$

step–6 Compute

$$y^1 = y^0 - \varepsilon\lambda c_p$$

where $0 < \varepsilon < 1$.

step–7 Compute $x^{k+1} = D_k y^1$ and set $k = k + 1$.

step–8 If $\| x^k - x^{k-1} \| \geqslant 2^{-L}$ go to step 2.

Acknowledgement

Most of the work was carried out when the author was with the Central Research Laboratory, Bharat Electronics Limited, Bangalore.

References

[1] Dennis Jr J E, Morshedi A M and Kathryn Turner, A variable metric variant of the Karmarkar algorithm for linear programming, *Math. Program.* **39** (1987) 1–20

[2] Karmarkar N, A new polynomial-time algorithm for linear programming, *Tech. Rep.* (1984) (New Jersey: AT and T Bell Labs)

[3] Joachim Kaschel, A note to the paper "An efficient algorithm for linear programming" of V Ch Venkaiah, *Proc. Indian Acad. Sci.* **102** (1992) 155–158

[4] Khachiyan L G, A polynomial algorithm in linear programming, *Doklady Akademiia Nauk SSSR* **244**:S (1979), 1093–1096, translated in *Sov. Mathematics Doklady* **20**:1 (1979), 191–194

[5] Rosen J B, The gradient projection method for nonlinear programming. Part I: Linear constraints, *SIAM J. Appl. Math.* **8** (1960) 181–217

[6] Venkaiah V Ch, An efficient algorithm for linear programming, *Proc. Indian Acad. of Sci.* **100** (1990) 295–301