COMPUTING KARMARKAR'S PROJECTIONS IN STOCHASTIC LINEAR PROGRAMMING

John R. Birge

Hengyong Tang

Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, MI 48109-2117

Technical Report 93-23

August 1993

Computing Karmarkar's Projections in Stochastic Linear Programming

John R. Birge
Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor MI 48109 U.S.A.

Hengyong Tang
Mathematics Department Liaoning University
Shenyang Liaoning 110036 P.R.China

August 18, 1993

Abstract

In this paper we compute Karmarkar's projections quickly using Moore-Penrose g-inverse and matrix factorization in stochastic linear programming. So computation work of $(A^T D^2 A)^{-1}$ is decreased.

Keywords: Stochastic Linear Programming, Two Stage Problem with Recourse, Karmarkar's Algorithm, Karmarkar's Projection, Moore-Penrose G-inverse, Matrix Factorization.

1 Introduction

Since Karmarkar proposed a new polynomial time algorithm for linear programming in 1984 [1], some research workers have started applying it to special large size problems of linear programming. One of them is block angular linear programming, especially, two stage problems of linear programming with recourse [2] [3] [4] [5] [6] [7].

In all variations of Karmarkar's algorithm, the major work is repeated computation of $(AD^2A^T)^{-1}$. All Karmarkar's algorithms applied to stochastic linear programming either reduce the number of dense columns that are in coefficient matrix of the linear programming or separate them explicitly from the other (nondense) columns to decrease computation work. [2] has discussed them in detail.

In this paper we compute $(AD^2A^T)^{-1}$ using Moore-Penrose g-inverse and QR factorization and taking advantage of the sparseness of A to decrease computation work.

In section [2], we review structure of stochastic linear programming to be discussed. In section 3 we review some characters of Moore-Prenrose g-inverse and QR factorization. Karmarkar's algorithm is formulated in section 4. In section 5, we give numerical method for computing $(AD)^+$ to compute $(AD^2A^T)^{-1}$. The complexity of the algorithm is analyzed in section 6.

2 Stochastic linear programming

We consider two stochastic linear programming with recourse that is defined over a discrete probability space [2]

$$\begin{array}{lrcl} \min & c_0^T x_0 & + & Q(x) \\ \text{s.t} & A_0 x_0 & = & b_0 \\ & x_0 & \geq & 0 \end{array} \tag{1}$$

where

$$Q(x) = \sum_{i=1}^{N} p_i Q(x_0, \xi(\omega_i))$$

,

$$\begin{array}{rcllcl} Q(x_0, \xi(\omega_i)) & = & \min & d_i^T x_i & & \\ & & \text{s.t} & W_i x_i & = & b_i - T_i x_0 \\ & & & x_i & \geq & 0 \end{array} \tag{2}$$

where $\xi(\omega)$ is random vector defined on the discrete probability space $(\Omega, \mathcal{F}, P)$, p_i is the probability of ω_i. x_0 is n_0−dimensional decision vector of the first stage. x_i is n_i−dimensional decision vector of the second stage. $A_0, b_0, c_0, W_i, T_i, b_i$ and b_i are matrices of size $m_0 \times n_0, m_0 \times 1, n_0 \times 1, m_i \times n_i, m_i \times n_0, m_i \times 1$ and $n_i \times 1$ respectively. $m_i \leq n_i, i = 0, 1, \ldots, N$. N is the number of the scenarios.

For purposes of discussion, we assume A_0 and W_i have full row rank.

Deterministic equivalent formulation of problems (1) (2) is

$$\begin{array}{llclcll} \min & x_0^T x_0 & + & \sum_{i=1}^N & c_i^T x_i & & \\ \text{s.t} & A_0 x_0 & & & = & b_0 & \\ & T_i x_0 & + & W_i x_i & = & b_i & i = 1, \ldots, N \\ & & & x_i & \geq & 0 & i = 0, \ldots, N \end{array} \tag{3}$$

where $c_i = p_i d_i$.

Let

$$A = \begin{pmatrix} A_0 & & & \\ T_1 & W_1 & & \\ \vdots & & \ddots & \\ T_N & & & W_N \end{pmatrix},$$

$$b = (b_0, b_1, \dots, b_N)^T,$$

$$c = (b_0, c_1, \dots, c_N)^T,$$

$$x = (x_0, x_1, \dots, x_N)^T.$$

Problem (3) can be written as

$$\begin{array}{lrcl} \min & c^T x & & \\ \text{s.t} & Ax & = & b \\ & x & \geq & 0 \end{array} \tag{4}$$

where A, b and c are matrices of size $m \times n, m \times 1$ and $n \times 1$ respectively, where

$$m = \sum_{i=0}^{N} m_i, \qquad n = \sum_{i=0}^{N} n_i.$$

Its dual problem is

$$\begin{array}{ll} \max & b^T y \\ \text{s.t} & A^T y \leq c \end{array} \tag{5}$$

where y is $m-$dimensional dual variable vector.

3 Moore-Penrose g-inverse

In this section, we review some characters of the Moore-Penrose generalized inverse (g-inverse) [8] [9] which are useful in following sections.

We consider real $m \times n$ matrix A. Let A^+ be the Moore-Penrose g-inverse of A.

Character 1 *a)* $A^+AA^T = A^TAA^+ = A^T$.
b) $AA^T(A^T)^+ = (A^T)^+A^TA = A$.
c) $(A^TA)^+ = A^+(A^T)^+$.
d) $(AA^T)^+ = (A^T)^+A^+$.

Character 2 *a) If rank*$(A) = n$*, then*

$$A^+ = (A^TA)^{-1}A^T.$$

b) If rank$(A) = m$*, then*

$$A^+ = A^T(AA^T)^{-1}.$$

Character 3 *Let $A = BC$, where A, B and C be $m \times n$, $m \times r$ and $r \times n$ matrices respectively, and*

$rank(A) = rank(B) = rank(C) = r$, then

$$A^+ = C^+B^+.$$

Let A be $m \times n$ matrix and $rank(A) = r \le n \le m$, utilizing Householder transform A can be factorized into

$$A = Q\begin{pmatrix} U \\ 0 \end{pmatrix} = Q_rU, \tag{6}$$

where Q is $m \times m$ orthogonal matrix, U is an upper trapezoidal matrix and $rank(U) = r$, Q_r is a matrix composed of the first r columns of Q. If $r = n$, $U = R$ is an upper triangular matrix. Equality (6) is written as

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix} = Q_nR. \tag{7}$$

If $rank(A) = r \le m \le n$, equality (6) and equality (7) become

$$A = \begin{pmatrix} U & , & 0 \end{pmatrix}Q = UQ_r \tag{8}$$

and

$$A = \begin{pmatrix} R & , & 0 \end{pmatrix}Q = RQ_n, \tag{9}$$

where U and R are lower trapezoidal matrix and lower triangular matrix. Q_r is a matrix composed of the first r rows of Q.

4 Karmarkar's algorithm

We focus on a variation of Karmarkar's algorithm which is generally called the dual affine scaling method [11] [2]. This algorithm requires that dual problem (5) has an interior feasible solution y°.

Algorithm 1 *(A,b,c,y°,stopping criterion,$0 < \gamma < 1$)*

1. *k=0.*

2. *stop if optimality criterion is satisfied.*

3. $\nu^k = c - A^Ty^k$.

4. $D^k =$ *ding* $\{1/\nu_1^k, \ldots, 1/\nu_n^k\}$.

5. $h_y = (A(D^k)^2A^T)^{-1}b$.

6. $h_\nu = -A^T h_y$.

7. $\alpha = \gamma \times \min\{-\nu_i^k/(h_\nu)_i | (h_\nu)_i < 0, i = 1, \ldots, n\}$.

8. $y^{k+1} = y^k + \alpha h_y$.

9. $x^{k+1} = (D^k)^2 h_\nu$.

10. $k = k+1$, *goto 2.*

The major computation work of the algorithm is computing $(A(D^k)^2A^T)^{-1}$. Since D^k is $m \times m$ full rank diagonal matrix, then $rank(AD^k) = m$, and

$$\begin{aligned}(AD^k)^+ &= (AD^k)((AD^k)(AD^k)^T)^{-1}, \\ ((AD^k)^+)^T(AD^k)^+ &= (A(D^k)^2A^T)^{-1}(AD^k)(AD^k)((AD^k)(AD^k)^T)^{-1} \\ &= (A(D^k)^2A^T)^{-1}.\end{aligned}$$

Thus step 5 of algorithm 1 can be written as

$$h_y = ((AD^k)^+)^T(AD^k)^+ b.$$

In following section we will discuss computing $(AD^k)^+$ quickly. For simplicity, we write D^k as D.

5 Computing $(AD)^+$

$$A = \begin{pmatrix} A_0 & & & \\ T_1 & W_1 & & \\ \vdots & & \ddots & \\ T_N & & & W_N \end{pmatrix},$$

Correspondingly we partition D into

$$D = \begin{pmatrix} D_0 & & & \\ & D_1 & & \\ & & \ddots & \\ & & & D_N \end{pmatrix}.$$

Thus

$$AD = \begin{pmatrix} A_0D_0 & & & \\ T_1D_0 & W_1D_1 & & \\ \vdots & & \ddots & \\ T_ND_0 & & & W_ND_N \end{pmatrix},$$

where A_0D_0 and W_iD_i are full row rank matrices.

Algorithm 2 *(QR factorization)*

1. *Do QR factorization for* A_0D_0

$$A_0D_0 = (R_0, 0)Q_0.$$

Thus

$$AD = \begin{pmatrix} R_0 & 0 & & & \\ T_1^{0,1} & T_1^{0,2} & W_1D_1 & & \\ \vdots & \vdots & & \ddots & \\ T_N^{0,1} & T_N^{0,2} & & & W_ND_N \end{pmatrix} Q^0,$$

where

$$Q^0 = \begin{pmatrix} Q_0 & & & \\ & I & & \\ & & \ddots & \\ & & & I \end{pmatrix}.$$

set $k = 1$.

2. *Do QR factorization for* $(T_k^{k-1,2}, W_kD_k)$

$$(T_k^{k-1,2}, W_kD_k) = (R_k, 0)Q_k.$$

Thus

$$AD = \begin{pmatrix} R_0 & & & & & & & \\ T_1^{0,1} & R_1 & & & & & & \\ & & \ddots & & & & & \\ T_k^{0,1} & T_k^{1,1} & \cdots & R_k & 0 & & & \\ T_{k+1}^{0,1} & T_{k+1}^{1,1} & \cdots & T_{k+1}^{k,1} & T_{k+1}^{k,2} & W_{k+1}D_{k+1} & & \\ \vdots & \vdots & & \vdots & \vdots & & \ddots & \\ T_N^{0,1} & T_N^{1,1} & \cdots & T_N^{k,1} & T_N^{k,2} & & & W_ND_N \end{pmatrix} Q^0Q^1\cdots Q^k,$$

where

$$Q^k = \begin{pmatrix} I & & & & & & \\ & \ddots & & & & & \\ & & I & & & & \\ & & & Q_k & & & \\ & & & & I & & \\ & & & & & \ddots & \\ & & & & & & I \end{pmatrix}.$$

3. If $k = N$, terminate.

otherwise $k = k + 1$ go to 2.

The result of the algorithm is

$$AD = (R, 0)Q = R(\overline{Q}_m, 0) \tag{10}$$

where

$$R = \begin{pmatrix} R_0 & & & \\ T_1^{0,1} & R_1 & & \\ \vdots & \vdots & \ddots & \\ T_N^{0,1} & T_N^{1,1} & & R_N \end{pmatrix}$$

is a full rank lower triangular square matrix,

$$Q = Q^0 Q^1 \cdots Q^N = \begin{pmatrix} Q_0 & & & \\ & Q_1 & & \\ & & \ddots & \\ & & & Q_N \end{pmatrix}$$

is an orthogonal matrix, and

$$\overline{Q}_m = \begin{pmatrix} Q_0 & & & & \\ & Q_1 & & & \\ & & \ddots & & \\ & & & Q_s & \\ & & & & \overline{Q}_{s+1} \end{pmatrix}$$

is a matrix composed of the first m rows of Q. $\overline{Q}_{s+1}$ is a matrix composed of first some rows of Q_{s+1}.

¿From (10) we can compute $(AD)^+$.

Theorem 1

$$(AD)^+ = (\overline{Q}_m, 0)^T \begin{pmatrix} I & & & \\ & \ddots & & \\ & & I & \\ & & & (\overline{Q}_{s+1}\overline{Q}_{s+1}^T)^{-1} \end{pmatrix} R^{-1}.$$

proof. By character 3 we have

$$\begin{aligned} (AD)^+ &= (R(\overline{Q}_m, 0))^+ = (\overline{Q}_m, 0)^+ R^+ \\ &= (\overline{Q}_m, 0)^T (\overline{Q}_m \overline{Q}_m^T)^{-1} R^T (RR^T)^{-1} \\ &= (\overline{Q}_m, 0)^T (\overline{Q}_m \overline{Q}_m^T)^{-1} R^{-1}. \end{aligned}$$

Since Q_i $(i = 0, 1, \dots, s)$ are orthogonal matrices we have

$$
\begin{aligned}
(\overline{Q}_m \overline{Q}_m^T)^{-1} &= \left(\begin{pmatrix} Q_0 & & & \\ & \ddots & & \\ & & Q_s & \\ & & & \overline{Q}_{s+1} \end{pmatrix} \begin{pmatrix} Q_0^T & & & \\ & \ddots & & \\ & & Q_s^T & \\ & & & \overline{Q}_{s+1}^T \end{pmatrix} \right)^{-1} \\
&= \begin{pmatrix} Q_0 Q_0^T & & & \\ & \ddots & & \\ & & Q_s Q_s^T & \\ & & & \overline{Q}_{s+1}\overline{Q}_{s+1}^T \end{pmatrix}^{-1} \\
&= \begin{pmatrix} I & & & \\ & \ddots & & \\ & & I & \\ & & & (\overline{Q}_{s+1}\overline{Q}_{s+1}^T)^{-1} \end{pmatrix}.
\end{aligned}
$$

This completes the proof of the theorem.

Theorem 1 shows that to compute $(AD)^+$, we just have to do QR factorization of AD and compute $(\overline{Q}_{s+1}\overline{Q}_{s+1}^T)^{-1}$ and R^{-1}. $\overline{Q}_{s+1}\overline{Q}_{s+1}^T$ is a very small matrix, R is full rank lower triangular square matrix. So the major work of computing $(AD)^+$ is QR factorization of AD. Since this factorization is implemented by block, computation work is greatly decreased.

6 Arithmetic complexity

For the purpose of the discussion we suppose that

$$m_0 = m_1 = \cdots = m_N, \qquad n_0 = n_1 = \cdots = n_N.$$

Theorem 2 *If computation of $(AD^k)^+$ is implemented using algorithm 2, then the arithmetic complexity of computing Karmarkar's projections is*

$$O(Nm_0^3 + N^2(n_0 - m_0) + N^2 m_0^2).$$

proof. The major operations of computing $(A(D^k)^2A^T)^{-1}$ are:

The QR factorization of $(T_i^{i-1,2}, W_iD_i)$ needs

$$
\begin{aligned}
& 2(m_i^2(n_i + \sum_{k=0}^{i-1}(n_i - m_i)) - \frac{1}{2}m_i^2(m_i + n_i + \sum_{k=0}^{i-1}(n_i - m_i)) + \frac{1}{3}m_i^3) \\
= \; & 2(m_0^2(n_0 + i(n_0 - m_0)) - \frac{1}{2}m_0^2(m_0 + n_0 + i(n_0 - m_0)) + \frac{1}{3}m_0^3)
\end{aligned}
$$

arithmetic operations [10].

The QR factorization of AD needs

$$\begin{aligned}&\sum_{i=1}^{N} 2(m_0^2(n_0+i(n_0-m_0)) - \frac{1}{2}m_0^2(m_0+n_0+i(n_0-m_0)) + \frac{1}{3}m_0^3) \\ = \;& 2N(m_0^2(n_0+\frac{N-1}{2}(n_0-m_0)) - \frac{1}{2}m_0^2(m_0+n_0+\frac{N-1}{2}(n_0-m_0)) + \frac{1}{3}m_0^3)\end{aligned}$$

arithmetic operations.

Computing R^{-1} needs

$$\frac{1}{2}m^2 = \frac{1}{2}(N+1)^2 m_0^2$$

arithmetic operations.

Thus the arithmetic complexity of computing $(A(D^k)^2A^T)^{-1}$ is

$$O(Nm_0^3 + N^2(n_0-m_0) + N^2m_0^2).$$

This completes the proof of the theorem.

Algorithm 1 finds an optimal solution to problem (4) and problem (5) in $O(\sqrt{m}L)$ iterations [12] [13], where L is a measure of problem's size. So the arithmetic complexity of algorithm in which algorithm 2 is used to compute $(A(D^k)^2A^T)^{-1}$ is

$$O(\sqrt{m}(Nm_0^3 + N^2(n_0-m_0) + N^2m_0^2)L).$$

But the arithmetic complexity of the general Karmarkar's algorithm is

$$O(m^{3.5}L) = O(\sqrt{m}N^3m_0^3L).$$

References

[1] N. Karmarkar, "A new polynomial time algorithm for linear programming," Combination, Vol. 4, 373-395, 1984.

[2] J. R. Birge and D. F. Holmes, "Efficient solution of two-stage stochastic linear programming using interior point methods," Computational Optimization and Application, 1, 245-276, 1992.

[3] J. R. Birge and L. Qi, "Computing block-angular Karmarkar projection with application to stochastic programming," Mgt. Sci., 34:12, 1472-1479, 1988.

[4] J. Arantes and J. Birge, "Matrix structure and interior point methods in stochastic programming," Presentation at Fifth Int. Stochastic Programming Conf., Ann Arbor, MI, 1989.

[5] J. R. Birge, R. M. Freund and R. J. Vanderbei, "Prior reduced fill-in in the solution of equations in interior point algorithms," Tech. Report, Sloan school of Management, M.I.T. Cambridge, MA, 1990.

[6] I. Lustig, R. Marsten and D. Shanno, "Computational experience with a primal-dual interior point method for linear programming," Linear Algebra and its Application, vol. 152, 191-222, 1991.

[7] I. Lustig, R. Mulvey and T. Carpenter, "Formulating stochastic programs for interior point methods," Oper. Res. vol. 39:5, 757-770, 1991.

[8] Xuchu He, Basic Theory and Algorithms of Generalized Inverse (in Chinese), Shanghei Science and Technology Press, Shanghei, 1985.

[9] R. Pao, Matra S. K., Generalized Inverse of Matrices and Its Application, Wily, New York, 1971.

[10] Gene H. Golub and Charles F. Van Loan, Matrix Computations, The Johns Hopkins University Press, Baltimore, Maryland, 1983.

[11] I. Adler, N. Karmarkar, M. G. C. Resende, and G. Veiga, "An implementation of Karmarkar's algorithm for linear programming," Mathematical Programming, 44, 297-355, 1989.

[12] Renato D. C. Monteiro and Ilan Adler, "Interior path following primal-dual algorithms Part I: linear programming," Mathematical Programming, 44, 27-41, 1989.

[13] Renato D. C. Monteiro and Ilan Adler, "Interior path following primal-dual algorithms Part II: Convex quadratic programming, Mathematical Programming, 44, 43-66, 1989.