COMPUTING KARMARKAR'S PROJECTIONS QUICKLY USING MATRIX FACTORIZATION

John R. Birge

Hengyong Tang

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

Technical Report 93-24

August 1993

Computing Karmarkar's Projections Quickly Using Matrix Factorization

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. So computation work of $(A^TD^2A)^{-1}$ is decreased.

Keywords: Linear Programming, Karmarkar's Algorithm, Karmarkar's Projection, Moore-Penrose G-inverse, Matrix Factorization.

1 Introduction

In 1984, Karmarkar [1] proposed a new polynomial time algorithm for linear programming based on repeated projective transformation. This algorithm creates a series of interior points converging to the optimal solution. His work has sparked tremendous interest and inspired other to modify his algorithm or to investigate similar methods for linear programming. A lot of variations of Karmarkar's algorithm have been made.

In all variations of Karmarkar's algorithm, the major work is repeated computation of $(A^TD^2A)^{-1}$ or solution of system of linear equations

$$A^TD^2Ay = b.$$

Since D changes at each iteration. To decrease computation work of Karmarkar's algorithm computing $(A^TD^2A)^{-1}$ quickly is needed. Many papers do their best to compute $(A^TD^2A)^{-1}$ quickly using rank-one update, approximate scaling matrix, updated Cholesky factorization and so on [1] [2] [3] [4]. [5] [6] have given efficient solution for block-angular linear programming, especially for two-stage stochastic linear programming.

In this paper we decrease the work of computing $(A^T D^2 A)^{-1}$ using Moore-Penrose g-inverse and matrix factorization.

In section 2, we review some characters of the Moore-Prenrose g-inverse. Karmarkar's algorithm is formulated in section 3. In section 4 , we give method for computation of $(A^T D^2 A)^{-1}$ using Moore-Penrose g-inverse and matrix factorization. The complexity of the algorithm is analysed in section 6.

2 Moore-Penrose g-inverse

In this section, we review some characters of the Moore-Penrose generalized inverse (g-inverse) [7] [8] 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*

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

3 Karmarkar's algorithm

For the purpose of discussion, we consider linear programming problem without generality

$$\begin{aligned} \max \quad & c^Tx \\ \text{s.t} \quad & Ax \leq b \end{aligned} \tag{1}$$

where c and x are n-vectors, b is an m-vector and A is a full rank $m \times n$ matrix, where $m \geq n$ and $c \neq 0$. Problem (1) has an interior feasible solution x°.

We focus on a variation of Karmarkar's algorithm which is generally called the dual affine scaling method [2].

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

1. *k=0.*
2. *stop if optimality criterion is satisfied.*
3. $\nu^k = b - Ax^k$.
4. $D^k = \text{ diag } \{1/\nu_1^k, \ldots, 1/\nu_m^k\}$.
5. $h_x = (A^T(D^k)^2A)^{-1}c$.
6. $h_\nu = -Ah_x$.
7. $\alpha = \gamma \times \min\{-\nu_i^k/(h_\nu)_i | (h_\nu)_i < 0, i = 1, \ldots, m\}$.
8. $x^{k+1} = x^k + \alpha h_x$.
9. $k = k + 1$, *goto 2.*

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

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

Thus step 5 of algorithm 1 can be written as

$$h_x = (D^kA)^+((D^kA)^+)^Tc.$$

In following section we will discuss computing $(D^kA)^+$ quickly. For simplicity, we write D^k as D. In general we let $rank(A) = r \leq n$.

4 Matrix factorization

Proposition 1 *Let D be a full rank diagonal matrix, A be a $m \times n$ matrix, and $A = BC$, where B is a full rank $m \times r$ matrix, C is a full rank $r \times n$ matrix, then*

$$\begin{aligned}
(DA)^+ &= C^+(DB)^+ \\
&= C^T(CC^T)^{-1}(B^TD^2B)^{-1}B^TD.
\end{aligned}$$

proof. Since D is a full rank square matrix, so $rank(DA) = r$ and $rank(B) = r$. According Character 3 and Character 2, We get needed result. This completes the proof of the proposition.

Using Gauss elimination method, we factorize A into $A = LU$, where L is a $m \times r$ unit lower trapezoidal matrix, U is a $r \times n$ full rank upper trapezoidal matrix.Thus

$$(DA)^+ = U^T(UU^T)^{-1}(L^TD^2L)^{-1}L^TD.$$

$(UU^T)^{-1}$ does not change at each iteration. If $r = n$, U is full rank upper triangular matrix, we have

$$(DA)^+ = V^{-1}(L^TD^2L)^{-1}L^TD.$$

So major work of computing $(DA)^+$ is the computation of $(L^TD^2L)^{-1}$.

We partition L into

$$L = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix}$$

where L_1 is a $r \times r$ unit lower triangular matrix, L_2 is a $(m-r) \times r$ matrix. Correspondingly we partition D into

$$D = \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix}$$

where D_1 is $r \times r$ full rank diagonal matrix, D_2 is $(m-r) \times (m-r)$ diagonal matrix. Thus

$$\begin{aligned} L^TD^2L &= \begin{pmatrix} L_1^TD_1, L_2^TD_2 \end{pmatrix} \begin{pmatrix} D_1L_1 \\ D_2L_2 \end{pmatrix} \\ &= L_1^TD_1^2L_1 + L_2^TD_2^2L_2 \\ &= L_1^TD_1^2L_1 + \sum_{i=1}^{m-r} d_{r+i}^2 l_i l_i^T, \end{aligned} \tag{2}$$

where

$$D_2 = \text{diag } \{d_{r+1}, \ldots, d_m\},$$

$$L_2^T = (l_1, \ldots, l_{m-n}).$$

$L_1^TD_1^2L_1$ is already Cholesky factorization form. A Cholesky factorization of L^TDL can be computed using a series of rank-one updates by (2).

To compute a Cholesky factorization of L^TDL, $m-r$ rank-one updates are needed. But m rank-one updates have to be done to compute a Cholesky factorization of A^TD^2A in general Karmarkar's algorithm [3]. So computation work is decreased. As in [3], rank-one update can be done by Fletcher and Powell [9].

5 Complexity of algorithm

Proposition 2 *If computation of* $(A^T(D^k)^2A)^{-1}$ *is done by LU factorization, then Algorithm 1 solves Problem* (1) *in no more than*

$$mn^2 + O(m^{2.5}(m-n)R)$$

arithmetic operations, where R *is a measure of the problem's size.*

proof. The major operation work of the algorithm is:

LU factorization needs

$$mn^2 - (m+n)n^2/2 + n^3/3 = O(mn^2)$$

arithmetic operations [11].

Algorithm 1 finds an optimal solution to problem (1) in $O(\sqrt{n}R)$ iterations [12] [13].

$(m-n)$ rank-one updates need $O(m^2(m-n)R)$ arithmetic operations.

So algorithm 1 solves problem (1) at most $mn^2 + O(n^{2.5}(m-n)R)$ arithmetic operations. This completes the proof of the proposition.

In general, R is very large. So

$$O(m^3R) >> m^2n + O(m^2(m-n)R),$$

especially, when $m-n$ is small.

Using approximation x^k, ν^k, the complexity of algorithm 1 can be reduced [12] [13].

References

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

[2] 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.

[3] David F. Shanno, "computing Karmarkar projections quickly," Mathematical Programming, 41, 61-71, 1988.

[4] I. Adler, N. Karmarkar, M. G. C. Resende and G. Veiga, " Date structures and programming techniques for the implementation of Karmarkar's algorithm," ORSA Journal on Computing 1(2), 1989.

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

[6] 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.

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

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

[9] R. Fletcher and M. J. D. Powell, "On the modification of LDL^T factorizations," Mathematics of Computation, 28, 1067-1087, 1974.

[10] Yinyu Ye and Masakazu Kojima, "Recovering optimal dual solution in Karmarkar's polynomial algorithm for linear programming," Mathematical Programming, 39, 305-317, 1987.

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

[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.