SOLVING STOCHASTIC LINEAR PROGRAMS VIA A VARIANT OF KARMARKAR'S ALGORITHM

John R. Birge
Department of Industrial & Operations Engineering
The University of Michigan

Liqun Qi
Department of Applied Mathematics
Tsinghua University

Technical Report 85-12

Solving Stochastic Linear Programs Via a Variant of Karmarkar's Algorithm

John R. Birge[1] and Liqun Qi[2]

Abstract: We present a variant of Karmarkar's algorithm for the special structure of stochastic linear programming. We give a worst case bound on the order of the running time that can be an order of magnitude better than that of Karmarkar's standard algorithm. A related variant is applied to the dual program, and its implications for very large-scale problems are given.

---

1. Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI 48109, USA. His work was supported by National Science Foundation Grant NO. ECS-8304065.

2. Department of Applied Mathematics, Tsinghua University, Beijing, China, and Department of Mathematics and Statistics, University of Pittsburgh, PA 15213, U.S.A. This author's work was supported by the 1984-1985 Andrew Mellon Postdoctoral Fellowship at University of Pittsburgh.

1. Introduction

The stochastic linear programming problem with fixed recourse (SLPF), whose random elements have discrete distributions can be formulated as:

$$\begin{aligned} &\min \quad c^T x + Q(x) \\ &\text{s.t.} \quad Ax \qquad\quad = b \\ &\qquad\quad x \geq 0 \end{aligned} \tag{1.1}$$

where

$$Q(x) = \sum_{k=1}^{N} p_k Q(x, \xi^k) \tag{1.2}$$

and for each $k = 1, \ldots, N$, the recourse cost $Q(x, \xi^k)$ is obtained by solving the recourse problem:

$$\begin{aligned} &Q(x, \xi^k) = \inf \{q^k y \mid Wy = h^k - T^k x,\ y \in R_+^{n_1}\} \\ &\xi^k = (q^k, h^k, T^k), \\ &p_k = \text{prob}\,[\xi(\omega) = \xi^k]. \end{aligned} \tag{1.3}$$

The sizes of matrices are consistent with $x \in R^{n_0}$, $y \in R^{n_1}$, $b \in R^{m_0}$, and $h^k \in R^{m_1}$, where $m_0 \leq n_0$, $m_1 \leq n_1$, and A and W have full row ranks. This formulation can also be regarded as part of an approximation scheme to SLPF, whose random elements have continuous distributions (see Birge and Wets [3]).

Substituting the expressions for Q in (1.1) , we obtain

$$\begin{aligned} &\min \quad c^T x + \sum_{k=1}^{n} p_k q^k y^k \\ &\text{s.t.} \quad Ax \qquad\qquad\quad = b \\ &\qquad T^k x + \qquad Wy^k \;\; = h^k, \quad k=1,\ldots,N. \\ &\qquad\quad x \geq 0, \quad y^k \geq 0, \quad k=1,\ldots,N. \end{aligned} \tag{1.4}$$

This is a highly structured large-scale linear programming problem. This structure is called dual block angular structure [15][16]. It has $n = n_0 + Nn_1$ variables and $m = m_0 + Nm_1$ constraints. The methods for solving it include:

the L-shaped method, proposed by Van Slyke and Wets [13]; the decomposition method, proposed by Dantzig and Madansky [7]; and the basis factorization method, proposed by Strazicky [14], and modified by Kall [10] and Wets [15]. The first method directly solves (1.4), while the other two solve the dual of (1.4). Birge [1][2] discussed the relationship between them. Also see [15][16] for other references.

If $T_1 = \cdots = T_N$, then (1.4) has a staircase structure. This type of problem is widely encountered in the context of dynamical systems, i.e., discrete versions of continuous linear programs or linear control problems [4][5][8][12][13][16].

In this paper, we study the approach of Karmarkar's new polynomial-time algorithm for linear programming [11] to solve (1.4). Karmarkar's algorithm is briefly described in Section 2. The order of his method is $O(n^{3.5} L^2 \ell n\, L\, \ell n \ell n\, L)$, where L is the length of the input. In Section 3, by means of the Sherman-Morrison-Woodbury formula and block matrix method, we explore the advantage of the special structure of (1.4) and obtain a variant of Karmarkar's algorithm for solving (1.4) with an order $O((n^{0.5}n_1^2 + \bar{n}n + n_0^3)nL^2 \ell n\, L\, \ell n \ell n\, L)$, where $\bar{n} = \max(n_0, n_1)$. Section 4 presents another variant of Karmarkar's algorithm to solve the dual of (1.4) and Section 5 describes its implementation in sequential approximation methods. Section 6 provides some special cases where additional computational savings are possible.

2. Karmarkar's algorithm

We briefly describe Karmarkar's algorithm in this section. This version of his algorithm was presented at the Symposium on Theory of Computing, Washington, D.C., April, 1984. It was later revised for publication in Combinatorica. Suppose that there is a linear programming problem:

$$\begin{aligned} &\min \quad c^T x \\ (2.1) \qquad &\text{s.t.} \quad Ax = b, \\ &\qquad\quad x \geq 0. \end{aligned}$$

where $x \in R^n$, $c \in Z^n$, $b \in Z^m$, $A \in Z^{mxn}$.

Suppose that we have a strictly interior feasible point $\bar{a}$ of (2.1), i.e.,

$$(2.2) \qquad A\bar{a} = b, \quad \bar{a} > 0.$$

Also suppose that we know a lower bound ℓ_0 of (2.1).

Karmarkar's algorithm creates a sequence of points $x^{(0)}, x^{(1)}, \ldots, x^{(k)}$ by these steps:

1. $x^{(0)} = \bar{a}$, k=0. An upper bound of (2.1): $u = c^T\bar{a}$. A lower bound of (2.1): $\ell = \ell_0$.
2. A tentative upper bound $u^1 = \ell + 2/3(u-\ell)$ and a tentative lower bound $\ell^1 = \ell + 1/3(u-\ell)$.
3. If $c^T x^{(k)} - \ell^1$ is small enough, i.e., less than a given positive number ε, then stop. Otherwise do $\{x^{k+1} = \phi(x^{(k)})\}$, where the function ϕ is defined later.
4. If $c^T x^{(k+1)} \leq u^1$, then do $\{u=u^1, k=k+1\}$ and go to step 2. If $f(x^{(k+1)}) \geq f(x^{(k)}) - \delta$, then do $\{\ell = \ell^1, k = k+1\}$ and go to Step 2. Otherwise do $\{k = k+1\}$ and go to Step 3.

f is the potential function: $f(x) = \sum_j \ell n \left(\frac{c^T x}{x_j}\right)$.

δ is a constant and depends upon the choice of the value of a parameter α in ϕ. A particular choice suggested in [11] is: $\alpha = 1/4$, $\delta = 1/8$.

$g = \phi(a)$ is defined by the following operations:

(i) Let $D = \text{diag}\{a_1, \ldots, a_n\}$, $e = (1, \ldots, 1)^T \in R^{n+1}$,

$$B = \left(\begin{array}{c|c} AD & -b \\ \hline e^T & \end{array}\right)$$

(ii) $c_p = (I_{n+1} - B^T(BB^T)^{-1} B) \begin{pmatrix} Dc \\ 0 \end{pmatrix}$, i.e.,

$$(2.2) \qquad c_p = \begin{pmatrix} Dc \\ 0 \end{pmatrix} - \begin{pmatrix} DA^T \\ -b^T \end{pmatrix} (AD^2A^T + bb^T)^{-1} AD^2c - \frac{c^Ta}{n+1} e.$$

$$(iii) \qquad g' = \frac{1}{n+1} e - \frac{\alpha}{\sqrt{n(n+1)}} \frac{c_p}{||c_p||_2}.$$

(iv) Suppose $g' = \begin{pmatrix} \bar{g} \\ g_{n+1} \end{pmatrix}$, where $\bar{g} \in R^n$, $g_{n+1} \in R$.

Then $g = \frac{1}{g_{n+1}} D\bar{g}$.

The main computational effort at each step k is to compute c_p by (2.2).

Karmarkar [11] also showed that to retain the linear decrease of the objective function value, it is only necessary to update those diagonal components D_{ii} of D from the corresponding diagonal components $\bar{D}_{ii}$ of old D if

$$(2.3) \qquad \left(\frac{D_{ii}}{\bar{D}_{ii}}\right)^2 \notin [1/2, 2],$$

after multiplying old D by a scaling factor . He showed that the average number of such ii's at each step is $O(n^{0.5})$. If D and D' differ only in the ith entry, the inverse of $M' = A(D'^2)A^T$ can be computed from the inverse of AD^2A^T in $O(n^2)$ arithmetic operations by means of rank-one updating formulas, such as the Sherman-Morrison formula [9] employed in [11].

Karmarkar showed that his algorithm converges on O(nL) steps. In the revised version of Karmarkar's algorithm [11], a combined form of the primal

and the dual problems was used and the lower bound ℓ_0 is unnecessary. Since a lower bound may be obtained in practice for stochastic linear programs and since the combined form of the primal and dual problem is complicated for practical use, we still use the original version of Karmarkar's algorithm. The primal-dual form may, however, be solved by combining our individual approaches to the primal and dual problems in Sections 3 and 4. Additional work requires a lower order number of operations. The result is a complexity bound that sums the primal and dual bounds.

3. Computation Reduction

To unify the notation, we rewrite (1.4) as

$$
\begin{aligned}
\min \quad & \sum_{k=0}^{N} c_k^T x_k \\
\text{s.t.} \quad & A_0 x_0 = b_0, \\
& A_k x_0 + W x_k = b_k, \quad k = 1,\ldots,N, \\
& x_k \geq 0, \quad k = 0,\ldots,N
\end{aligned}
\tag{3.1}
$$

where the sizes of matrices are consistent with x_k, $c_k \in R^{n_k}$, $b_k \in R^{m_k}$, $m_k \leq n_k$, $n_1 = n_2 = \ldots = n_N$, $m_1 = \ldots = m_N$, A_0 and W have full row ranks. If an element of a vector or a matrix needs to be specified, we use parentheses, e.g., $(A_k)ij$.

Let $n = n_0 + Nn_1$, $m = m_0 + Nm_1$,

$$
x = \begin{bmatrix} x_0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix}, \quad c = \begin{bmatrix} c_0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ c_N \end{bmatrix}, \quad b = \begin{bmatrix} b_0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ b_N \end{bmatrix}, \quad A = \begin{bmatrix} A_0 & & & \\ A_1 & W & & 0 \\ \cdot & & \cdot & \\ \cdot & & & \cdot \\ \cdot & 0 & & \cdot \\ A_N & & & W \end{bmatrix}.
$$

Then (3.1) can be expressed exactly the same as (2.1). It is now ready for application of Karmarkar's algorithm as described in Section 2. Suppose

$a=(a_0^T,\ldots,a_N^T)^T$ is the current iteration point of Karmarkar's algorithm. Let D_k = diag $\{(a_k)_1, (a_k)_2\ldots,(a_k)n_k\}$, D = diag $\{D_0,\ldots,D_N\}$. As discussed in Section 2, the main computational work at each step of Karmarkar's algorithm is to calculate (2.2), which is dominated by calculating M^{-1}, where

$$M = AD^2A^T + bb^T. \tag{3.2}$$

<u>Theorem 3.1</u> Let $S_0 = -I_2 \in R^{m_0 x m_0}$, $S_k = WD_k^2W^T$, k = 1,...,N, S = diag $\{S_0,\ldots,S_N\}$. Then S^{-1} = diag $\{S_0, S_1^{-1}, \ldots, S_N^{-1}\}$. Let $B_k = (A_k\ b_k)$ for k = 0,...,N, $\bar{D}_0$ = diag$\{D_0, 1\}$, I_1 and I_2 be unit matrices in R^{n_0+1} and R^{m_0} respectively. Let

$$G_1 = \bar{D}_0^{-2} + \sum_{k=0}^{N} B_k^T S_k^{-1} B_k, \quad G_2 = -B_0 G_1^{-1} B_0^T, \tag{3.3}$$

$$U = \begin{pmatrix} B_0 & I_2 \\ B_1 & 0 \\ \cdot & \cdot \\ \cdot & \cdot \\ B_N & 0 \end{pmatrix}.$$

If G_1 is invertible, then G_2 and M are invertible and

$$M^{-1} = S^{-1} - S^{-1} U \begin{pmatrix} I_1 & G_1^{-1}B_0^T \\ 0 & I_2 \end{pmatrix} \begin{pmatrix} I_1 & 0 \\ 0 & G_2^{-1} \end{pmatrix} \begin{pmatrix} I_1 & 0 \\ B_0 & I_2 \end{pmatrix} \begin{pmatrix} G_1^{-1} & 0 \\ 0 & I_2 \end{pmatrix} U^T S^{-1}. \tag{3.4}$$

<u>Proof</u> Since W has full row rank, S_k is invertible for k = 1,...,N. Therefore, S is invertible. In Karmarkar's algorithm, a is positive, i.e., D is always invertible. Let $\bar{D}$ = diag $\{\bar{D}_0, I_2\}$. Let

$$G = \begin{pmatrix} G_1 & -B_0^T \\ -B_0 & 0 \end{pmatrix}, \qquad \tilde{U} = \begin{pmatrix} B_0\bar{D}_0 & I_2 \\ \cdot & 0 \\ \cdot & \cdot \\ \cdot & \cdot \\ B_N\bar{D}_0 & 0 \end{pmatrix} = U\bar{D}.$$

It is seen that $M = S + \tilde{U}\tilde{U}^T$. According to the Sherman-Morrison-Woodbury formula [9], M is invertible and

$$M^{-1} = S^{-1} - S^{-1}\tilde{U}(I + \tilde{U}^T S^{-1}\tilde{U})^{-1}\tilde{U}^T S^{-1}$$

$$= S^{-1} - S^{-1}U(\bar{D}^{-2} + U^T S^{-1}U)^{-1}U^T S^{-1} \tag{3.5}$$

$$= S^{-1} - S^{-1}UG^{-1}U^T S^{-1}$$

if and only if $(I + \tilde{U}^T S^{-1}\tilde{U})$ is invertible, i.e., G is invertible.

Suppose G_1 is invertible. Since G_1 is a symmetric matrix, so is G_1^{-1}, and we can write $G_1^{-1} = G_1^{-1/2} G_1^{-1/2}$, where $G_1^{-1/2}$ is also a symmetric matrix (although possibly complex). Since A_0 has full row rank, $B_0 G_1^{-1/2}$ also has full row rank. Since $n_0 + 1 \geq m_0$, $G_2 = -(B_0 G_1^{-1/2})(B_0 G_1^{-1/2})^T$ is invertible. It is easy to see that

$$G\begin{pmatrix} I_1 & G_1^{-1}B_0^T \\ 0 & I_2 \end{pmatrix}\begin{pmatrix} I_1 & 0 \\ 0 & G_2^{-1} \end{pmatrix}\begin{pmatrix} I_1 & 0 \\ B_0 & I_2 \end{pmatrix}\begin{pmatrix} G_1^{-1} & 0 \\ 0 & I_2 \end{pmatrix} = \begin{pmatrix} I_1 & 0 \\ 0 & I_2 \end{pmatrix} = I_{m_0+n_0+1}.$$

Hence, G is invertible. According to (3.5), M is invertible and (3.4) holds. ■

A variant of Karmarkar's algorithm for solving linear programming problem (3.1) is as follows:

Algorithm 3.1

(1) Apply Karmarkar's algorithm described in Section 2 to (3.1) with A,b,D,C specified in this section.

(2) Replace the expression of $M^{-1} = (AD^2A^T + bb^T)^{-1}$ in (2.2) by expression (3.4).

(3) Update old D by multiplying by the scaling factor and when

$$\left(\frac{(D_k)_{ii}}{\text{old } (D_k)_{ii}}\right) \in [1/2, 2], \tag{3.6}$$

use old $(D_k)_{ii}$ instead of $(D_k)_{ii}$ in (2.2) and (3.4).

(4) When (3.6) does not hold, use the repeated rank-one updating technique to calculate S_k^{-1}, and G_1, i.e.,

$$S_k' = S_k + \alpha W_i W_i^T,$$

(3.7)
$$(S_k')^{-1} = S_k^{-1} - \beta(S_k^{-1}W_i)(S_k^{-1}W_i)^T,$$
$$G_1' = G_1 - \beta(B_k^T S_k^{-1} W_i)(B_k^T S_k^{-1} W_i)^T,$$

where W_i is the ith column vector of W,

$$\alpha = (D_k)_{ii}^2 - \text{old } (D_k)_{ii}^2,$$
$$\beta = \alpha/(1 + \alpha W_i^T S_k^{-1} W_i).$$

Then $S_k, S_k^{-1}, G_1 \longleftarrow S_k', (S_k')^{-1}, G_1'$.

We see that mathematically, Algorithm 3.1 is equivalent to Karmarkar's algorithm, when G_1 is invertible at each step. The difference is in the numerical calculation orders.

Theorem 3.2 (Complexity). Suppose G_1 is invertible at each step. Let $\bar{n} = \max(n_0, n_1)$. Then the overall running time of Algorithm 3.1 is $O((n^{0.5} n_1^2 + n\bar{n} + n_0^3)nL^2 \ell n L \ell n \ell n L)$.

Proof At each step, we have:

work	number of arithmetic operations (in average)
calculate S^{-1} by (3.7)	$O(n^{0.5} n_1^2)$
calculate G_1 by (3.7)	$O(n^{0.5} n_1^2 + n_0^3)$
calculate G_2 directly	$O(n_0^3)$
calculate G_1^{-1} and G_2^{-1}	$O(n_0^3)$
multiply A with a vector	$O(m\bar{n})$
multiply S^{-1} with a vector	$O(mm_1)$
multiply U with a vector	$O(mn_0)$

Other efforts are small compared with the operations above. Sum up the above numbers to yield the number of arithmetic operations at each step in

average, $O(n^{0.5}\, n_1^2 + n\bar{n} + n_0^3)$. Since there are $O(nL)$ steps and since each arithmetic operation needs a precision of $O(L)$ we obtain our conclusion. ■

We can roughly compare this order with $O(n^{3.5}\, L^2\, \ell n\, L\, \ell n \ell n\, L)$, the order of Karmarkar's algorithm in general cases. Suppose $N \sim n_0 \sim n_1$, then $O((n^{0.5}\, n_1^2 + n\bar{n} + n_0^3)nL^2\, \ell n\, L\, \ell n \ell n\, L) = O(n^{2.5}L^2\, \ell n\, L\, \ell n \ell n\, L)$, as we mentioned in Section 1.

4. Dual Formulation

The dual problem to (3.1) can also be solved by Karmarkar's algorithm. The program has block-angular structure which leads to computational reductions similar to those of Section 3. An advantage is also gained for the use of Karmarkar's algorithm in approximation schemes.

The dual of (3.1) is

$$\text{(4.1)} \qquad \max \quad b_0^T \pi_0 + \sum_{k=1}^{N} b_k^T \pi_k$$

$$\text{s.t.} \quad A_0^T \pi_0 + \sum_{k=1}^{N} A_k \pi_k \leq c_0$$

$$W^T \pi_k \leq c_k, \quad k = 1,\ldots,N.$$

An alternative with constraints of the form in (2.1) can be easily obtained. Rewrite (4.1) as

$$\text{(4.2)} \qquad \max \quad \sum_{k=0}^{N} \bar{b}_k^T y_k$$

$$\text{s.t.} \quad \sum_{k=0}^{N} \bar{A}_k^T y_k = c_0,$$

$$\bar{W}^T y_k = c_k, \; k = 1,\ldots,N;$$

$$y_k \geq 0, \; k=0,\ldots,N;$$

where $\bar{A}_0^T = (A_0^T, -A_0^T, I)$, $\bar{A}_k^T = (A_k^T, -A_k^T, 0)$, $k=1,\ldots,N$, $\bar{b}_k^T = (b_k^T, -b_k^T, 0)$, $k=1,\ldots,N$; $\bar{W}^T = (W^T, -W^T, I)$ and the y_k variables are defined accordingly.

The only necessary modification of Karmarkar's algorithm is to change the sign on the second term in Step iii to reflect maximization. The computational effort is again dominated by calculating M^{-1} where $M = AD^2A^T + cc^T$.

Note that

$$AD^2A^T = \begin{bmatrix} \sum_{k=0}^{N} \bar{A}_k^T D_k^2 \bar{A}_k & \bar{A}_1^T D_1^2 \bar{W} & \cdots & \bar{A}_N^T D_N^2 \bar{W} \\ \bar{W}^T D_1^2 \bar{A}_1 & & & 0 \\ \vdots & & & \\ \bar{W}^T D_N^2 \bar{A}_N & 0 & & \bar{W}^T D_N^2 \bar{W} \end{bmatrix} .$$

Now, observe that $M = P + U\,V^T$ where

$$P = \begin{bmatrix} \sum_{k=0}^{N} \bar{A}_K^T D_K \bar{A}_K & & & \\ & \bar{W}^T D_1^2 \bar{A}_1 & \bar{W}^T D_1^2 \bar{W} & 0 \\ & \vdots & & \\ & \bar{W}^T D_N^2 \bar{A}_N & 0 & \bar{W}^T D_N^2 \bar{W} \end{bmatrix} ,$$

$$U = \begin{bmatrix} I & c_0 \\ & c_1 \\ 0 & \vdots \\ & c_N \end{bmatrix} ,$$

and

$$V = \begin{bmatrix} 0 & c_0 \\ \overline{W}^T D_1^2 \overline{A}_1 & c_1 \\ \vdots & \vdots \\ \overline{W}^T D_N^2 \overline{A}_N & c_N \end{bmatrix}.$$

The Sherman-Morrison-Woodbury formula yields

$$M^{-1} = P^{-1} - P^{-1} U(I + V^T P^{-1} U)^{-1} V^T P^{-1}. \tag{4.3}$$

Let

$$H = \sum_{k=0}^{N} \overline{A}_k^T D_k^2 \overline{A}_k,$$

$$H_k = \overline{W}^T D_k^2 \overline{A}_k, \quad k = 1,\ldots,N,$$

and $$J_k = \overline{W}^T D_k^2 \overline{W}, \quad k = 1,\ldots,N,$$

then

$$P^{-1} = \begin{bmatrix} H^{-1} & & & \\ -H_1 H^{-1} & J_1^{-1} & & 0 \\ \vdots & & \ddots & \\ & & & \\ -H_N H^{-1} & 0 & & J_N^{-1} \end{bmatrix} \tag{4.4}$$

and

$$(I \pm V^T P^{-1} U) = \left[\begin{array}{c|c} I - \sum_{k=1}^{N} H_k^T H_k H^{-1} & -\sum_{k=1}^{N} (H_k^T H_k H^{-1} c_0 + H_k^T J_k^{-1} c_0) \\ \hline c_0^T H^{-1} - \sum_{k=1}^{N} c_k^T H_k H^{-1} & I + c_0^T H^{-1} c_0 + \sum_{k=1}^{N} (c_k^T H_k H^{-1} c_0 + c_k^T J_k^{-1} c_k) \end{array}\right] \tag{4.5}$$

Equations (4.3), (4.4), and (4.5) can then be used to develop another variant of Karmarkar's algorithm for (4.2).

Algorithm 4.1

1) Apply Karmarkar's algorithm from Section 2 to the data in (4.2).

2) Replace the M^{-1} expression by (4.3).

3) Update old D by the scaling factor δ and if (3.6) holds, use old $(D_k)_{ii}$ instead of $(D_k)_{ii}$ in (2.2) and (4.3).

4) If (3.6) does not hold, use a repeated rank-one updating scheme to find $(H')^{-1}$, $H_k'p$ and $(J_k')^{-1}$, where

(4.6)
$$H' = H + \alpha(\bar{A}_k^T)_i \, (\bar{A}_k)_{i.},$$
$$H_k' = H_k + \alpha(\bar{W}^T)_i \, (\bar{A}_k)_{i.},$$
and
$$J_k' = J_k + \alpha(\bar{W}^T)_i \, \bar{W}_{i.},$$

where $T_{i.}$ denotes the ith row of matrix T, and α is as defined above. Equations (4.6) are then used repeatedly (with the Sherman-Morrison formulas for the inverses of H^{-1} and J_k^{-1}) to update P^{-1} and $(I + V^T P^{-1} U)$. $(I + V^T P^{-1} U)^{-1}$ can then be calculated directly.

Algorithm 4.1 is equivalent to Karmarkar's algorithm but the structure again allows for a reduction in computational effort.

Theorem 4.2. The overall running time of Algorithm 4.1 is $O((n^{0.5} \bar{n}^2 + n\bar{n} + n_0^3)nL^2 \ln L \ln\ln L)$.

Proof: The computational effort in each step is:

work	average operations
Calculate $(H')^{-1}$	$O(n^{0.5} n_0^2)$
Calculate $(J_k')^{-1}$	$O(n^{0.5} n_1^2)$
Calculate $H_k'(H')^{-1}$	$O(n^{0.5} n_0\bar{n})$
Calculate $(H_k')^T(J_k')^{-1}$	$O(n^{0.5} n_1^2)$

Calculate $(I + V^T P^{-1} U)^{-1}$	$O(n_0^3)$
Multiply A with a vector	$O(m\, n_0 + Nn_1(m_1 + n_1))$
Multiply P^{-1} with a vector	$O(n\, \bar{n})$
Multiply V^T with a vector	$O(n_0\, n)$

All other effort is dominated by these operations. Note that $H_k'(H')^{-1}$ and $(H_k')^T(J_k')^{-1}$ are found sequentially by updating $(H')^{-1}$ and $(J_k')^{-1}$ first and then using the formula in (4.6) or H_k'. The number of operations in each step is $O(n^{0.5}\bar{n}^2 + n_0^3 + n\bar{n})$, and the result is obtained as in Theorem 3.2.

A rough comparison for $N \sim n_0 \sim n$ yields $O((n^{0.5}\bar{n}^2 + n\bar{n} + n_0^3)nL^2\ \ell n\ L\ \ell n \ell n\ L)$ $= O(n^{2.5}L^2\ \ell n\ L\ \ell n \ell n\ L)$. This bound is the same as that provided in Theorem 3.2.

5. Dual Use in Approximations

The dual provides an additional benefit. In using (3.1) to approximate a stochastic linear program with more than N realizations of the random vector b_k, additional realizations $b_{N+1},\ldots,b_{N+\ell}$ are added to (3.1) and the probabilities p_k, $k=1,\ldots,N+\ell$ are updated. (See Birge and Wets [3] for procedures to do this.) A feasible solution to the new problem (3.1) with $N'=N+\ell$ is not readily found but given a feasible solution (4.2) another can be found easily if $T_1=T_2=\ldots=T_N=T$ and $q_1=q_2=\ldots=q_N=q$. Under these assumptions let the old solution be $(\pi_0^{old}, \pi_1^{old},\ldots, \pi_N^{old})$ where

$$W^T \pi_k^{old} = p_k\, q,$$

and

$$A_0^T \pi_0^{old} + \sum_{k=1}^{N} T^T \pi_k^{old} = c_0.$$

Define a new solution by

(5.1)
$$\pi_0^{New} = \pi_0^{Old},$$

$$\pi_k^{New} = \left(\sum_{i=1}^{N} \pi_i^{Old}\right) p_k^{New}, k=1,\ldots,N+\ell.$$

The solution in (5.1) is then feasible in (4.2) with objective value $b_0^T \pi_0^{old} + \sum_{k=1}^{N+\ell} b_k^T \left(\sum_{i=1}^{N} \pi_i^{Old}\right) p_k^{New}$. The total effort for solving the new (4.2) should be less than resolving (3.1) because no procedures to find a feasible interior point are necessary (assuming, of course, that π^{Old} is interior). The objective is generally close to the old optimal value of (4.2) and should be close to the new optimal value. The effort factor for the number of steps to solve the new (4.2) should therefore be much smaller than $O(nL)$.

A similar procedure can be used in the primal problem (3.1) if the randomness is limited to the objective function, i.e., $T^1=T^2=\ldots=T^N=T$ and $h^1=h^2=\ldots=h^N=h$ but $q^i \neq q^j$ for all $i \neq j$. Again, a starting feasible solution can be easily found by letting

(5.2)
$$x_k^{new} = x_k^{old}, \; k=0,\ldots,N,$$

and
$$x_j^{New} = \underset{x_i^{old}}{\operatorname{argmin}} \{q^j x_i^{old}, \; i=1,\ldots,N\}, \; j=N+1,\ldots,N+\ell \; .$$

Now, let $\bar{p} = \sum_{j=N+1}^{N+\ell} p_j^{new}$, z^*_{old} be the objective value of the old (3.1) with x_k^{Old} and let

$$z^0_{New} = \bar{p}\, c^T x_0^{new} + \sum_{j=N+1}^{N+\ell} p_j^{new}\, q^j\, x_j^{new}.$$

The new initial objective value in (3.1) is then

$$z_{New} = (1-\bar{p})\, c^T x_0^{old} + \sum_{j=1}^{N} p_j^{New}\, q^j\, x_j^{old} + z^0_{New},$$

and we wish to find

$$(5.3) \qquad z^*_{New} = \min\{(1-\bar{p})c^T x_0 + \sum_{j=1}^{N} p_j^{New} q^j x_j + z'_N\},$$

where $z'_N = \bar{p}\, c^T x_0 + \sum_{j=N+1}^{N+\ell} p_j^{New} q^j x_j$. Now, assume that $p_j^{New}/(1-\bar{p}) = p_j^{old}$, i.e., that the old probabilities have the same relative values in the new problem, so that x^{old} minimizes $(1-\bar{p})\, c^T x_0 + \sum_{j=1}^{N} p_j^{new} q^j x_j$. We then have

$$(5.4) \qquad z^*_{New} \leq (1-\bar{p}) z^*_{old} + \max z'_N$$

$$z^*_{New} \geq (1-\bar{p}) z^*_{old} + \min z'_N.$$

Let L' be the size of the data for the objective and constraints relating only to z'_N. This yields the following result.

Theorem 5.1 The overall running time of Algorithm 3.1 starting from a solution defined by (5.2) for $N+\ell$ realizations of the objective q^k, assuming $T^1=T^2=\ldots=T^{N+\ell}=T$, $h^1=h^2=\ldots=h^{N+\ell} = h$, and $p_j^{New}/(1-\bar{p}) = p_j^{old}$, is $O((n^{0.5} n_1^2 + n\bar{n} + n_0^3)nL\ L'\ \ell n\ L\ \ell n \ell n\ L)$.

Proof: The number of operations per iteration follows from Theorem 3.2. The number of iterations $O(n\ L')$ follows from (5.4) for $2^{O(L')} \geq z'_N \geq -2^{O(L')}$. ■

Theorem 5.1 specifies the comments above about the reduction in effort in solving the enlarged problem. The structure of the primal under the assumptions of Theorem 5.1 allows for a specific bound to be obtained. The entire approximating process can then be seen as a variable dimension variant of Karmarkar's algorithm in which the number of iterations is $O(\bar{\ell}\ n\ L')$ where $\bar{\ell}$ additional problems of size L' are added after the initial solution of (3.1).

6. Special Cases

The work of the variants of Karmarkar's algorithm is reduced for the following special cases.

(1) Simple recourse [16]

In this case, $W = (I, -I)$, where I is the unit matrix in $R^{m_1 \times m_1}$, and $n_1 = 2m_1$. Write $x_k = (x_k^+, x_k^-)$, $x_k^+, x_k^- \in R^{m_1}$, or $k=1,\ldots,N$ in (3.1). Let $D_k = \text{diag}\{D_k^+, D_k^-\}$ in (3.2), where $D_k^+, D_k^- \in R^{m_1 \times m_1}$, or $k=1,\ldots,N$. Then in Theorem 3.1, we have

$$S_k = WD_k^2W^T = (D_k^+)^2 + (D_k^-)^2,$$

$$S_k^{-1} = \text{diag}\{((a_k^+)_1^2+(a_k^-)_1^2)^{-1},\cdots,((a_k^+)_{m_1}^2+(a_k^-)_{m_1}^2)^{-1}\}, \text{ where} \tag{4.1}$$

$D_k^+ = \text{diag}\{(a_k^+)_1,\cdots,(a_k^+)_{m_1}\}$ and $D_k^- = \text{diag}\{(a_k^-)_1,\cdots,(a_k^-)_{m_1}\}$, for $k=1,\ldots,N$. Therefore, S^{-1} can be easily calculated in (3.4). Substitute D_k in (3.6) by D_k^+, D_k^-. Then according to (3.3), (3.7) can be replaced by

$$G_1^{\dagger} = G_1 + \beta'(B_k)_i^T(B_k)_i, \tag{4.2}$$

$$\beta' = ((a_k^+)_i^2 + (a_k^-)_i^2)^{-1}_{\text{new}} - ((a_k^+)_i^2 + (a_k^-)_i^2)^{-1}_{\text{old}},$$

where $(B_k)_i$ is the ith row vector of B_k.

The work is reduced but the order of the algorithm is the same. Similar modifications can be done for the dual approach.

(2) Network recourse [17]

In this case, $W = \begin{pmatrix} E & 0 \\ I & I \end{pmatrix}$ where E is the network node-arc incidence matrix. The calculation work of S_k and S_k^{-1} can still be reduced but again the order is the same.

Reference

[1] J. Birge, 1984. A Dantzig-Wolfe decomposition variant equivalent to basis factorization, Mathematical Programming Study, to appear.

[2] J. Birge, 1985. The relationship between the L-shaped method and dual basis factorization for stochastic linear programming, in Y. Ermoliev and R. Wets, eds., Numerical Methods in Stochastic Programming, to appear.

[3] J. Birge and R. Wets, 1985. Designing approximation schemes for stochastic optimization problems, in particular, for stochastic programs with recourse, Mathematical Programming Study, to appear.

[4] J. Bisschop and A. Meeraus, 1977. Matrix augmentation and partitioning in the updating of the basis inverse, Mathematical Programming, 13, 241-254.

[5] J. Bisschop and A. Meeraus, 1980. Matrix augmentation and structure preservation in linearly constrained control problems, Mathematical Programming, 18, 7-15.

[6] A. Charnes, T. Song, and M. Wolfe, 1984. An explicit solution sequence and convergence of Karmarkar's algorithm, Research Report CCS501, Center for Cybernetic Studies, University of Texas at Austin, 1984.

[7] G. Dantzig and A. Mandansky, 1961. On the solution of two-stage linear programs under uncertainty, Proc. Fourth Berkeley Symposium on Mathematical and Probability, Vol. 1, University of California Press, Berkeley, 1961. 165-176.

[8] R. Fourer, 1984. Staircase matrices and systems, SIAM Review, 26, 1-70.

[9] G. H. Golub and C. F. Van Loan, 1983. Matrix Computations, John Hopkins University Press.

[10] P. Kall, 1979. Computational methods for solving two-stage stochastic linear programming problems, Z. Angew. Math. Phys., 30, 261-271.

[11] N. Karmarkar, 1984. A new polynomial-time algorithm for linear programming, Combinatorica, 4, 373-395.

[12] A. Perold and G. Dantzig, 1979. A basis factorization method for block triangular linear programs in: Sparse Matrix Proceedings, 1978, I. Duff and G. Stewart, eds., SIAM Publications, Philadelphia, 283-312.

[13] R. Van Slyke and R. Wets, 1969. L-shaped linear programs with applications to optimal control and stochastic programming, SIAM J. Appl. Math. 17, 638-663.

[14] B. Strazicky, 1980. Some results concerning an algorithm for the discrete recourse problem, in: Stochastic Programming, M. Dempster, ed., Academic Press, London, 263-274.

[15] R. Wets, 1983. Stochastic programming: Solution techniques and approximation schemes, in: Mathematical Programming: The State of the Art, 1982, A. Bachem, M. Groetschel and B. Korte, eds., Springer-Verlag, 566-603.

[16] R. Wets, 1985. Large scale linear programming techniques in stochastic programming, in: Y. Ermoliev and R. Wets, eds., Numerical Methods in Stochastic Programming, to appear.

[17] S. Wallace, 1985. On network structured stochastic optimization problem, report no. 842555-8, Chr. Michelsen Institute, Bergen, Norway.