AN EXPLICIT FACTORIZATION FOR SOLVING MULTISTAGE STOCHASTIC LINEAR PROGRAMS USING INTERIOR POINT METHODS

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

Technical Report 93-18

July 1993

An explicit factorization for solving Multistage Stochastic Linear Programs using Interior Point Methods.

D. Holmes
Department of Industrial and Operations Engineering
University of Michigan
Ann Arbor, MI 48109-2117

9 June 1993

Abstract: Multistage stochastic linear programs (MSLPs) have a deterministic representation which is a generalization of a dual block angular program. These programs have been shown in the literature to be difficult to solve using interior point methods, since they require solutions of very large and dense systems. Several methods for improving solution efficiencies for two-stage SLPs have been suggested in the literature, but not for multistage programs. Computational experience suggests that a direct factorization based on the Sherman-Morrison-Woodbury identity is superior in terms of numeric stability and speed. This paper derives a multistage generalization of the direct factorization technique and discusses its theoretical properties.

1 Introduction

Many practical problems that are influenced by uncertainty can be modeled as stochastic programs. Examples of problems that have been formulated include cash and portfolio management models (Mulvey [1]), electric power generation capacity planning (Louveaux, [2]), and forestry management (Gassman, [3]). Stochastic programs are multistage in nature, so that they sequentially make decisions before realization of stochastic parameters over time. Another common characteristic is that they relate stochasticity linearly to decision variables and costs. Such problems are called multistage stochastic linear programs with recourse (MSLPs).

One of the most undesirable characteristics of MSLPs is their excessive computational requirements. Since the size of these problems grows exponentially with the amount of stochastic information included in the formulation and the number of stages (discrete decision points), realistically sized problems can quickly become computationally intractable. The recent advent of interior point methods for the solution of large linear programs (Marsten, et.al. [4], Carolan, et. al. [5], and Monma and Marsten [6]) has held great promise for the efficient solution of these problems.

To be effective however, these algorithms require the efficient solution of a sequence of large, symmetric, positive definite systems of linear equations. Generally, the solutions to these systems are obtained by factoring the coefficient matrix into some equivalent triangular matrix and back solving with some right hand side. The ease with which the factorizations are obtained decreases significantly as the density of the coefficient matrix increases. Unfortunately, the structure of MSLPs can lead to quite dense systems, limiting the effectiveness of interior point methods for their solution.

The purpose of this paper is to propose an efficient factorization technique for solving these systems. The factorization is simply an extension of the two-stage Birge-Qi factorization (Birge and Qi [7]), which is based on a generalized version of the Sherman-Morrison-Woodbury formula. The two stage method has been shown to have a worst case computational complexity at least an order of the number of variables over that of the standard Karmarkar algorithm. We will show here that the multistage generalization has the same complexity in the number of variables, but grows exponentially in the number of stages. This is still troublesome, and methods for getting around this restriction will be the focus of future research.

2 Preliminaries

2.1 Multistage Stochastic Linear Programs

Here, we will assume that we are trying to solve multistage stochastic linear programs with fixed recourse. If the number of stages H is finite, general form of this problem is

$$\min_{y_0} \; c_0^T y_0 + E_{\xi_1}(\min_{y_1} c_1^T y_1 + \ldots E_{\xi_H|\xi_1 \ldots \xi_{H-1}}(\min_{y_H} c_H^T y_H)\ldots)$$

$$\begin{aligned} \text{s.t.} \quad A_0 y_0 &= b_0 \\ T_0 y_0 + W_1 y_1 &= \xi_1 \quad a.s. \\ &\vdots \\ T_{H-1} y_{H-1} + W_H y_H &= \xi_H \quad a.s. \\ u_0 \geq y_0 \geq l_0, u_t \geq y_t \geq l_y &\quad t = 1, \ldots, H \quad a.s. \end{aligned}$$

where bold face vectors are (possibly) stochastic. Were, the stochastic elements are defined over a discrete canonical probability space $(\Xi, \sigma(\Xi), P)$, where $\Xi = \Xi_1 \otimes \cdots \otimes \Xi_H$, and the S_t elements of Ξ_t are $\{(T_{ts}, W_{ts}, \xi_{ts}, c_{ts}), s = 1, \ldots, S_t\}$. The requirement that each stochastic constraint must hold almost surely may be written deterministically by defining a set of constraints for each realization. The data dependencies for a three stage problem with two realizations in each stage is shown graphically in Figure 1. The paths of the tree shown in Figure 1 correspond to scenarios that

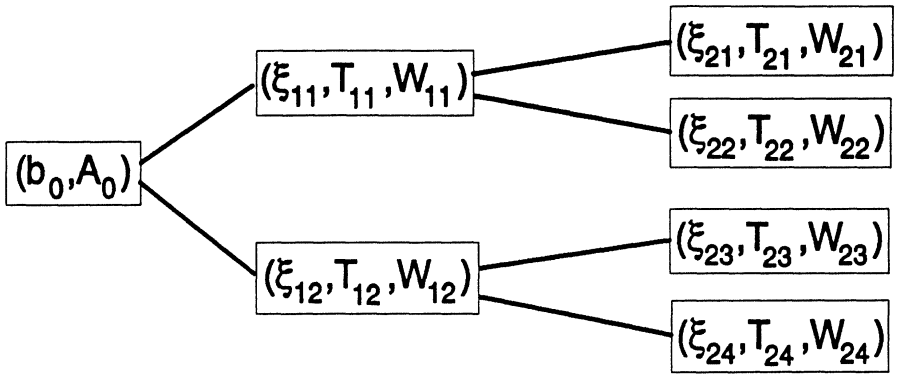

Figure 1: Multistage deterministic equivalents

describe complete events up to the last stage. Scenarios can also be defined for an intermediate period t which correspond to paths from the root node of the tree to nodes in the tth stage.

Defining a set of variables for each node in the decision tree, the deterministic equivalent program can be written as

$$\min \quad c_0 y_{1,0} + \sum_{k=1}^{\bar{N}_1} p_{k,1} c_{k,1} y_{k,1} + \cdots + \sum_{k=1}^{\bar{N}_H} p_{k,H} c_{k,H} y_{k,H}$$

$$\begin{aligned} \text{subject to} \qquad A_0 y_{1,0} &= b_0 \\ T_{j,t} y_{\alpha(j,t),t-1} + W_{j,t} y_{j,t} &= \xi_{j,t} \quad j = 1, \ldots, \bar{N}_t, \quad t = 1, \ldots, H \\ u_{j,t} \geq y_{j,t} &\geq l_{j,t} \qquad j = 1, \ldots, \bar{N}_t, \quad t = 1, \ldots, H \end{aligned} \tag{1}$$

where

$$\begin{aligned} N_t &= \text{Number of possible outcomes in stage } t \\ \bar{N}_t &= \text{Cumulative number of scenarios through stage } t, \bar{N}_t = N_1 \times \cdots \times N_t \\ p_{i,t} &= \text{Probability that scenario } i \text{ of stage } t \text{ occurs, } i = 1, \ldots, \bar{N}_t \\ c_{i,t} &= \text{Cost vector for scenario } i \text{ of stage } t \\ \xi_{i,t} &= \text{Right hand side vector for scenario } i \text{ of stage } t \\ T_{i,t} &= \text{Technology matrix for scenario } i \text{ of stage } t \\ W_{i,t} &= \text{Recourse matrix for scenario } i \text{ of stage } t, \; W_{i,t} \in \Re^{n_{i,t} \times m_{i,t}} \\ \alpha(j,t) &= \lfloor (j-1)/N_t \rfloor + 1 \end{aligned}$$

Here, $\alpha(j,t)$ finds the predecessor of node j in stage t.

The block structure for a two stage problem as well as the three stage problem depicted in Figure 1 is shown in Figure 2. The two stage deterministic equivalent is called a *dual block*

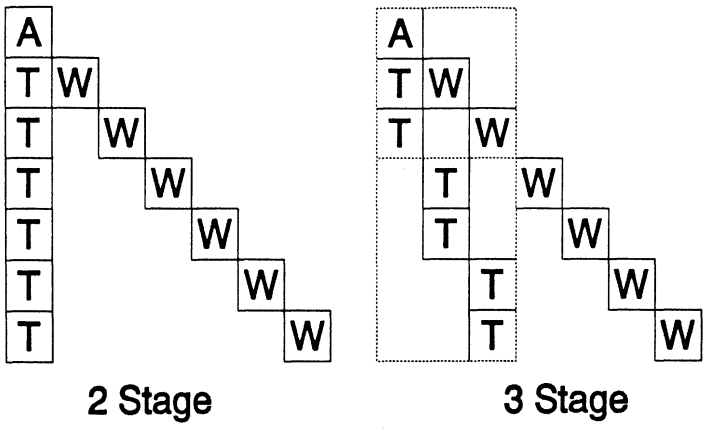

Figure 2: Block structure for deterministic equivalents

angular program. As can be seen from Figure 2, the multistage deterministic equivalent can be viewed as several dual block angular programs nested within each other. As will be seen in the next section, dual block angular programs are not well suited for interior point methods.

2.2 Interior Point Methods

In the last decade, several breakthroughs in general purpose linear programming algorithms have been made using path following methods (Gill, et. al. [8]). Karmarkar [9] pioneered these breakthroughs with the first algorithm that could be proven to converge to an optimal solution in polynomial, or $O(n^2m^2L)$ time, where n is the size of the problem and L is a measure of the problem's data. Other variants such as the dual affine scaling algorithm (Adler et.al. [10]) and the primal-dual method (Megiddo, [11]) have also been shown to converge in polynomial time. By contrast, the worst case complexity of the simplex method cannot be bounded by a polynomial.

For the purposes of discussion, we focus on the dual affine variant as applied to the dual-block angular program described above. Consider a linear program in the following standard equality form:

$$\begin{array}{lll} \textbf{(P)} & \text{minimize} & c^T x \\ & \text{subject to} & Ax = b \\ & & x \geq 0 \end{array}$$

where $A \in \Re^{m \times n}$ and has full row rank, $b \in \Re^m$ is a resource vector, and $c \in \Re^n$ is the objective function vector. The dual affine variant finds an optimal solution to the dual (D) to the problem (P):

$$\begin{array}{lll} \textbf{(D)} & \text{maximize} & b^T y \\ & \text{subject to} & A^T y \leq c \end{array}$$

where $y \in \Re^m$ is a dual vector to the equality constraints of (P). The algorithm requires an initial interior point y^0 that satisfies dual feasibility ($A^T y^0 < c$), and successively iterates on a current

point to find another interior point with a better objective function value. The algorithm proceeds as follows:

---

1. $k = 0$. Select a stopping criterion (e.g., Stop if $b^T y^{k+1} - b^T y^k < \epsilon$, where $\epsilon > 0$ is a small positive number.
2. Stop if optimality criterion is satisfied, otherwise calculate dual slack variables: $\nu^k = c - A^T y^k$.
3. Calculate the search direction, which is a function of a Newton step taken toward the "center" of the feasible region and a steepest descent step in some transformed space:
   Let $D^k = \text{diag}\{(1/\nu_1^k), \ldots, (1/\nu_m^k)\}$, $dy = (A(D^k)^2 A^T)^{-1} b$, and $dv = -A^T dy$
4. Calculate a step size:
   (a) Let $\alpha = \gamma \times \min\{\nu_i^k / -(dv)_i : (dv)_i < 0, i = 1, \ldots, m\}$
   Here, $0 < \gamma < 1$ is a step size parameter to insure that the next iterate will remain interior to the feasible solution set. For most practical purposes, $0.95 \leq \gamma < 1$ is sufficient.
5. Update dual variables, primal variables, and counters:
   (a) Let $y^{k+1} = y^k + \alpha dy$ and $x^{k+1} = (D^k)^2 dv$
   (b) Let $k = k + 1$
   (c) Goto 2

Dual Affine Algorithm

The vast majority of the computational effort required in the above procedure is to calculate the solution to the system $(AD^2A^T)dy = b$ (the iteration counter will be dropped whenever the context is clear), or to calculate some factorization of the matrix to enable quick solution of the system. These computations are common to every interior point algorithm developed thus far (Shanno and Bagchi, [12]). The matrix $M \equiv AD^2A^T$ is a large ($M \in \Re^{m \times m}$) symmetric, positive definite matrix for which several solution methods have been developed (see, e.g. Golub and Van Loan [13]). Generally speaking, a direct inversion of M is an inefficient solution method, and is seldom used in large scale implementations.

There are two main strategies for solving the system $(AD^2A^T)dy = b$. They are *iterative methods* and *direct methods.*

1. *Iterative methods* generate a sequence of approximations to dy. Since these methods only involve vector-matrix multiplication, they are computationally more attractive and require less storage than alternative solution procedures. Convergence to solutions of these linear systems can be unacceptably slow, unless special tricks (e.g., matrix preconditioners) are employed. Examples of iterative methods include the Jacobi, Gauss-Seidel, Chebychev, Lanczos, and Conjugate Gradient methods. Meijerink and van der Vorst [14] discuss the use of these methods in interior point algorithms.
2. *Direct methods* calculate the exact solution to the set of equations $(AD^2A^T)dy = b$ by factoring the matrix (AD^2A^T), and using backwards/forwards substitution to find dy. The most

common schemes in use are (LU) factorization and Cholesky (LL^T) factorization. The effectiveness of these methods are dependent on the use of special data structures and pivoting rules, and on the characteristics of the coefficient matrix itself. Examples of software implementations include YSMP (Eisenstadt [15]) and SPARSPAK (Chu, et. al. [16]). Direct and iterative methods can also be combined by using ideas from the direct solution procedures to generate an effective preconditioner that improves the convergence of iterative methods.

The ease with which direct methods may be used depends heavily on the amount of *fill-in*, or density of the factorized matrix. Matrices which can be rearranged to minimize fill-in can be stored using less memory, and hence require fewer operations to update the factorization or obtain a solution. However, matrices that are ill-structured will generate an extremely dense matrix M, and hence can be quite inefficient to solve.

The density of the matrix (AD^2A^T) largely depends on the number of dense columns that are contained in the original matrix A. Unfortunately, the dual block angular program that is the deterministic equivalent of a two stage stochastic program contains many dense columns. To see this, let $D_0^2 \in \Re^{m_0 \times m_0}$ and $D_l^2 \in \Re^{m_l \times m_l}$ be defined by $D_l^2 = \text{diag}\,\{(\nu_1^l)^{-2}, \ldots, (\nu_{m_l}^l)^{-2}\}, l = 0, \ldots, N$. Suppose further that $T^l = T, W^l = W, l = 1, \ldots, N$. Then the required system to solve can be calculated symbolically as

$$AD^2A^T = \begin{bmatrix} AD_0^2A^T & AD_0^2T^T & AD_0^2T^T & \cdots & AD_0^2T^T \\ TD_0^2A^T & TD_0^2T^T + WD_1^2W^T & TD_0^2T^T & \cdots & TD_0T^T \\ TD_0^2A^T & TD_0^2T^T & TD_0^2T^T + WD_2^2W^T & \cdots & TD_0T^T \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ TD_0^2A^T & TD_0^2T^T & TD_0^2T^T & \cdots & TD_0T^T + WD_N^2W^T \end{bmatrix}$$

Clearly, the presence of the columns associated with the T matrices creates an extremely dense M matrix to factorize. For this reason, Arantes and Birge [17] have shown that dual block angular programs in the primal form are expensive (whenever possible) to solve, even with basic preprocessing or row reordering to reduce fill-in.

Since multistage stochastic programs are also dual block angular problems, they suffer from the same fill-in problem that plagues two-stage stochastic programs. Figure 3 shows the block structure for the multistage program shown in figures 1 and 2. Although the matrix is not completely dense like its two stage counterpart, it still has a substantial number of nonzeros off the diagonal. Unfortunately, the AD^2A^T matrices include columns which have nonzeros near the top and bottom of the matrix. As a result, we would expect the matrices to be resistant to sparsity reducing techniques such as row and column reorderings.

Practically, dense projection matrices occur even in problems with relatively sparse T matrices. For example, a three stage (two scenario) version of the problem SCAGR7 (see, e.g. Gassman [18] or Birge [19]) is shown in Figure 4. The left hand figure is the nonzero structure of the deterministic equivalent's AA^T matrix, and the right hand figure is the nonzero structure of AA^T's Cholesky factorization (with minimum degree heuristic applied.) As can be seen, the Cholesky factorizations can be dense and hence computationally difficult to solve.

2.3 Explicit Factorizations of Stochastic Linear Programs

The solution to the set of equations that determine search directions in affine scaling algorithms may also be accomplished by decompositions specific to the dual block angular structure. Birge and Qi [7] propose using a generalized version of the Sherman-Morrison-Woodbury formula (reviewed below) for the inverse of a matrix to efficiently obtain the search direction dy. While the full

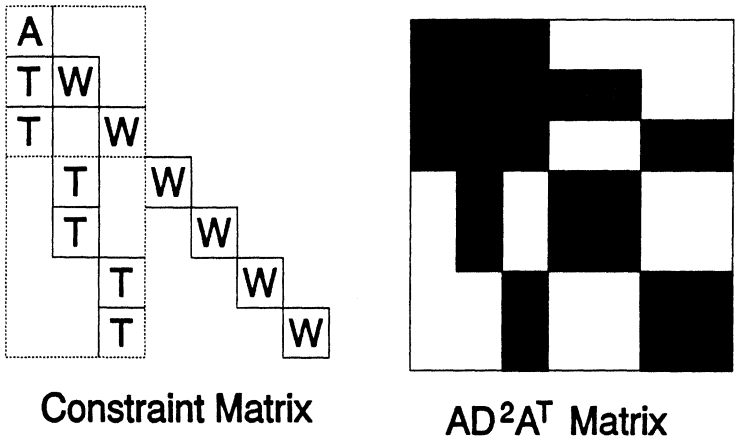

Figure 3: AD^2A^T block structure for multistage problems.

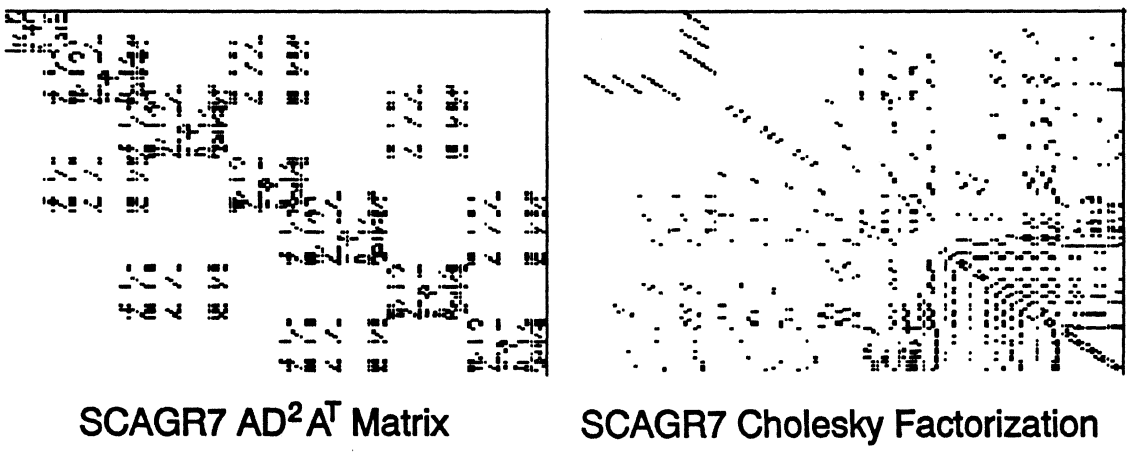

Figure 4: SCAGR7, 3 stages, 2 realizations per stage.

calculation of the step size in affine scaling may be performed generally in $O(m^2n)$ operations, the decomposition they propose reduces the computational complexity of the two stage program to $O(n^2)$ operations (assuming $n_l \sim n$ for all $l = 0,\ldots,N$). After reviewing the two stage factorization, we will posit its theoretical extension to multiple stages and discuss its implementation.

2.3.1 Two stage Birge and Qi factorization

The main result obtained by Birge and Qi notes that the matrix AD^2A^T may be written as the sum of a block diagonal matrix D and the product of two similar matrices U and V (defined below). Given this representation, the Sherman-Morrison-Woodbury formula, which is reproduced here for convenience, may be used to find the inverse of the matrix. The notation used in the following follows that of Section 2.2.

Lemma 1 : *For any matrices A, U, V such that A and $(I + V^TA^{-1}U)$ are invertible,*

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^TA^{-1}U)^{-1}V^TA^{-1} \tag{2}$$

Proof: See, e.g. Golub and Van Loan [13].

Theorem 1 (Birge and Qi [7], Birge and Holmes [20]) *Consider the feasible region of the dual block angular program*

$$\begin{array}{lccll} \textit{minimize} & c_0^Tx_0 & + & \sum_{l=1}^N c_l^Ty_l & \\ \textit{subject to} & A_0x_0 & & = b^0 & \\ & T_lx_0 & + & W_ly_l & = b_l \quad l = 1,\ldots N \\ & x_0, & & y_l & \geq 0 \end{array} \tag{3}$$

and its dual solution $(y^0,\ldots,y^N)$. Let $M = AD^2A^T$, $S = diag\{S_0, S_1,\ldots,S_N\}$, where $S_l = W_lD_l^2W_l^T, l = 1,\ldots,N$, $S_0 = I_2 \in \Re^{m_0 \times m_0}$, and $D_l^2 = diag\{(\nu_1^l)^{-2},\ldots,(\nu_{m_l}^l)^{-2}\}$. Furthermore, let I_1 and I_2 be identity matrices of dimension n_0 and m_0, respectively. Also, let

$$G_1 = (D_0)^2 + \sum_{l=0}^{N} T_l^TS_l^{-1}T_l, \qquad G_2 = -A_0G_1^{-1}A_0^T$$

$$U = \begin{pmatrix} A_0 & I_2 \\ T_1 & 0 \\ \vdots & \vdots \\ T_N & 0 \end{pmatrix}, \qquad V = \begin{pmatrix} A_0 & -I_2 \\ T_1 & 0 \\ \vdots & \vdots \\ T_N & 0 \end{pmatrix}$$

If A_0 and W_l has full row rank, for $l = 1,\ldots,N$, G_2 and M are invertible and

$$M^{-1} = S^{-1} - S^{-1}U \begin{bmatrix} I_1 & G_1^{-1}A_0^T \\ 0 & -I_2 \end{bmatrix} \begin{bmatrix} I_1 & 0 \\ 0 & G_2^{-1} \end{bmatrix} \begin{bmatrix} I_1 & 0 \\ A_0 & I_2 \end{bmatrix} \begin{bmatrix} G_1^{-1} & 0 \\ 0 & I_2 \end{bmatrix} \tag{4}$$

Proof: In the affine scaling algorithm, ν_k is positive, so D is invertible. By assumption, W_l has full row rank, so S_l is invertible for $l = 1,\ldots,N$ and S is invertible. Let $\bar{D} =$ diag $\{D_0, I_2\}$ and

$$\tilde{U} = \begin{pmatrix} A_0D_0 & I_2 \\ T_1D_0 & 0 \\ \vdots & \vdots \\ T_ND_0 & 0 \end{pmatrix}, \qquad \tilde{V} = \begin{pmatrix} A_0D_0 & -I_2 \\ T_1D_0 & 0 \\ \vdots & \vdots \\ T_ND_0 & 0 \end{pmatrix}$$

Note that $\tilde{U} = U\bar{D}$ and $\tilde{V} = V\bar{D}$. By construction, $M = S + \tilde{U}\tilde{V}$. Applying Lemma 1 (the Sherman-Morrison-Woodbury formula), M is invertible and

$$\begin{aligned} M^{-1} &= (S + \tilde{U}\tilde{V})^{-1} \\ &= S^{-1} - S^{-1}U\bar{D}(I + \bar{D}^T V^T S^{-1} U\bar{D})^{-1}\bar{D}V^T S^{-1} \\ &= S^{-1} - S^{-1}U\bar{D}\bar{D}^{-1}(\bar{D}^{-2} + V^T S^{-1}U)^{-1}\bar{D}^{-1}\bar{D}V^T S^{-1} \\ &= S^{-1} - S^{-1}U(\bar{D}^{-2} + V^T S^{-1}U)^{-1}V^T S^{-1} \\ &= S^{-1} - S^{-1}UG^{-1}V^T S^{-1} \end{aligned}$$

where

$$G = \begin{bmatrix} G_1 & A_0^T \\ -A_0 & 0 \end{bmatrix}$$

Then

$$V^T S^{-1} U = \begin{bmatrix} A_0 A_0^T + \sum_{l=1}^{N} T_l S_l^{-1} T_l & A_0^T \\ -A_0 & -I_2 \end{bmatrix}$$

and $\bar{D}^2 + V^T S^{-1} U = G$. So,

$$M^{-1} = S^{-1} - S^{-1}UG^{-1}V^T S^{-1} \tag{5}$$

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

G_1 can be expanded as

$$G_1 = (D_0)^2 + A_0 A_0^T + \sum_{l=1}^{N} T_l S_l^{-1} T_l^T$$

By construction, $(D_0)^2$ and $A_0 A_0^T$ are positive definite and symmetric. Since S_l is positive definite for all $l = 1, \ldots, N$, $T_l S_l^T T_l^T$ is positive definite and symmetric. The sum of positive definite matrices is again positive definite, so G_1 is positive definite and symmetric. So, G_1^{-1} exists, is symmetric, and can be written as $G_1 = G_1^{1/2} G_1^{1/2}$, where $G_1^{1/2}$ is also symmetric. By assumption, A_0 has full row rank, so $A_0 G_1^{1/2}$ has full row rank. Consequently, $G_2 = -A_0 G_1^{-1} A_0^T$ is invertible, and

$$G \begin{bmatrix} I_1 & G_1^{-1}A_0^T \\ 0 & -I_2 \end{bmatrix} \begin{bmatrix} I_1 & 0 \\ 0 & G_2^{-1} \end{bmatrix} \begin{bmatrix} I_1 & 0 \\ A_0 & I_2 \end{bmatrix} \begin{bmatrix} G_1^{-1} & 0 \\ 0 & I_2 \end{bmatrix}$$

$$= \begin{bmatrix} G_1 & A_0^T \\ -A_0 & 0 \end{bmatrix} \begin{bmatrix} G_1^{-1} + G_1^{-1}A_0^T G_2^{-1} A_0 G_1^{-1} & G_1^{-1}A_0^T G_2^{-1} \\ -G_2^{-1}A_0 G_1^{-1} & -G_2^{-1} \end{bmatrix} = \begin{bmatrix} I_1 & 0 \\ 0 & I_2 \end{bmatrix}.$$

Since G is invertible, (5) holds. Consequently, M is invertible and (4) holds. □

Using Theorem 1 to explicitly compute the inverse of the matrix M is not the most efficient way to determine the search direction dy. However, the system of equations $Mdy = b$ may be efficiently solved using Theorem 1 by solving (in order)

$$Sp = b \qquad Gq = V^T p \qquad Sr = Uq \tag{6}$$

and setting $dy = p - r$. To verify that $dy = M^{-1}b$, note that

$$\begin{aligned} dy &= p - r = S^{-1}b - S^{-1}Uq \\ &= [S^{-1} - S^{-1}U(G^{-1}V^T S^{-1})]b \\ &= M^{-1}b \end{aligned}$$

Further simplification of the second equation of (6) may be made by symbolically expanding G into its components G_1 and A_0. Let $q^T = (q_1^T, q_2^T)$, where $q_1 \in \Re^{n_0}$ and $q_2 \in \Re^{m_0}$. Then solving

$$\begin{bmatrix} G_1 & A_0^T \\ -A_0 & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} \hat{p}_1 \\ \hat{p}_2 \end{bmatrix} = \begin{bmatrix} A_0^T & T_1^T & \cdots & T_N^T \\ -I_2 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} p_0 \\ \vdots \\ p_N \end{bmatrix} \tag{7}$$

implies that

$$\begin{aligned} q_2 &= (A_0 G_1^{-1} A_0^T)^{-1}(\hat{p}_2 - A_0 G_1^{-1} \hat{p}_1) \\ &= -G_2^{-1}(\hat{p}_2 - A_0 G_1^{-1} \hat{p}_1) \\ q_1 &= (G_1)^{-1}(\hat{p}_1 - A_0^T q_2). \end{aligned}$$

Solving (6) requires Cholesky factorizations of S_l, G_1, and G_2. At each iteration of the affine scaling algorithm, an update of S_l to reflect the current dual solution must precede the solution of $Mdy = b$. Birge and Holmes [20] use these facts to describe a pseudo-code implementation for solving $AD^2A^T = b$ for two-stage stochastic programs. They also discuss the decomposition's theoretical and empirical properties.

2.3.2 Multistage Birge and Qi Factorization

The Birge and Qi factorization can be extended to discrete support multistage stochastic program. Multistage stochastic programs generalize two stage stochastic programs by allowing more than one recourse decision to be made over the problem horizon. As was mentioned in Section 2, these programs also have deterministic equivalents which are "nested" dual block angular programs. This fact can be used to apply a nested version of the Birge-Qi factorization to find iterates of interior point algorithms.

Consider the deterministic equivalent linear program formulated in Section 2, as well as the following the definitions.

$$\begin{aligned} D_{i,t} &= \text{Diagonal matrix for scenario } i \text{ of stage } t \text{ required by the interior point algorithm} \\ \tilde{A}_t &= \text{The constraint matrix of the original problem truncated to} \\ &\quad t \text{ stages, where } 2 \le t \le H, \text{ and } \tilde{A}_t \in \Re^{\tilde{n}_t \times \tilde{m}_t}, t = 1, \ldots, H \\ \tilde{D}_t &= \text{The diagonal matrix associated with } \tilde{A}_t, \quad t = 1, \ldots, H \\ \tilde{M}_t &= \tilde{A}_t(\tilde{D}_t)^2(\tilde{A}_t)^T \quad t = 1, \ldots, H \end{aligned}$$

These definitions will be used to decompose the projection matrix AD^2A^T for the nested dual block angular program into submatrices that can be fit into the generalized version of the Sherman-Morrison-Woodbury formula.

To see this decomposition, let

$$\tilde{T}_{k,t} = \begin{bmatrix} & T_{\beta(k,1),t} & \\ 0_A & \vdots & 0_B \\ & T_{\beta(k,N_t),t} & \end{bmatrix}, k = 1, \cdots, \bar{N}_{t-1} \quad \beta(k,k') = \bar{N}_t(k-1) + k'$$

where 0_A is a zero matrix with $n_A = \tilde{n}_{t-2} + \sum_{k'=1}^{k-1} n_{k',t-1}$ columns and 0_B is a zero matrix with $\sum_{k'=k+1}^{\bar{N}_{t-1}} n_{k',t-1}$ columns. Here, $\beta(k,k')$ is the index of descendant k' of the kth problem in stage t. $\beta(k,k')$ is also a function of t, but will be left implicit to simplify the presentation. Let $\tilde{W}_{k,T} =$

diag $\{W_{\beta(k,1),H}, \ldots, W_{\beta(k,N_H),H}\}$ for $k = 1, \ldots, \bar{N}_{H-1}$. For any $2 \leq t \leq H$, the constraint matrix of the original linear program (1) may be rewritten as

$$\tilde{A}_H = \begin{bmatrix} \tilde{A}_{H-1} & & & \\ \tilde{T}_{1,H} & \tilde{W}_{1,H} & & \\ \vdots & & \ddots & \\ \tilde{T}_{\bar{N}_{H-1},H} & & & \tilde{W}_{\bar{N}_{H-1},H} \end{bmatrix}$$

Having defined the new dual block angular version of the constraint matrix of (1), we can now apply Theorem 1 of the previous section. Let

$$S_{k,H} = (W_{k,H})(D_{k,H})^2(W_{k,H})^T \text{ for } k = 1, \ldots, \bar{N}_H$$

and $S_H = \text{diag}\{I_{H-1}, S_{1,H}, \ldots, S_{\bar{N}_H,H}\}$, where $\dim I_{H-1} = \dim \tilde{A}_{H-1}$. Let $\bar{D} = \text{diag}\{\tilde{D}_{H-1}, I_{H-1}\}$ and

$$U_H = \begin{bmatrix} \tilde{A}_{H-1} & I_{H-1} \\ \tilde{T}_{1,H} & 0 \\ \vdots & \vdots \\ \tilde{T}_{\bar{N}_{H-1},H} & 0 \end{bmatrix}, \qquad V_H = \begin{bmatrix} \tilde{A}_{H-1} & -I_{H-1} \\ \tilde{T}_{1,H} & 0 \\ \vdots & \vdots \\ \tilde{T}_{\bar{N}_{H-1},H} & 0 \end{bmatrix}$$

If $\tilde{U}_H = U_H\bar{D}$ and $\tilde{V}_H = V_H\bar{D}$, we can decompose $\tilde{M}_H$ into

$$\tilde{M}_H = AD^2A^T = S_H + (\tilde{U}_H)(\tilde{V}_H)^T.$$

Developing a procedure for finding the interior point search direction for the multistage problem is now a matter of redefining the appropriate matrices that were used in the proof of Theorem 1. Key to this development is the definition of the G and G_1 matrices. To avoid yet another subscript, the G_1 and G_2 matrices used in the two-stage presentation will be rewritten as $G1$ and $G2$. Let

$$G1_{k,H} = \begin{cases} \tilde{D}^2_{H-2} & k = 0 \\ D^2_{k,H-1} + \sum_{k'=1}^{N_t} (T_{\beta(k',k),H})^T (S_{\beta(k',k),H})^{-1} T_{\beta(k',k),H} & k = 1, \ldots, \bar{N}_{H-1} \end{cases}$$

and $G1_H = \text{diag}\{G_{0,H}, \ldots, G_{\bar{N}_{H-1},H}\}$. Unlike the $G1$ matrix previously defined for the two-stage stochastic linear program, the multistage $G1$ does not include $(\tilde{A}_{H-1})(\tilde{A}_{H-1})^T$. Instead, we will include $(\tilde{A}_{H-1})(\tilde{A}_{H-1})^T$ explicitly in the matrix G. Let G be

$$G = \begin{bmatrix} G1_H + (\tilde{A}_{H-1})(\tilde{A}_{H-1})^T & \tilde{A}^T_{H-1} \\ -\tilde{A}_{H-1} & 0 \end{bmatrix}.$$

Following the discussion presented after Theorem 1, the equations $Mdy = b$ may be solved using:

$$S_H p_H = b \qquad G_H q_H = V_H^T p_H \qquad S_H r_H = U_H q_H \tag{8}$$

and setting $dy_H = p_H - r_H$. The most complex set of equations to solve is the second, $G_H q_H = V_H^T p_H$. Let $\hat{p}_H = (\hat{p}^A_H, \hat{p}^B_H) = V_H^T p_H$. Solving

$$\begin{bmatrix} G1_H + (\tilde{A}_{H-1})(\tilde{A}_{H-1})^T & \tilde{A}^T_{H-1} \\ -\tilde{A}_{H-1} & 0 \end{bmatrix} \begin{bmatrix} \hat{q}^A_H \\ \hat{q}^B_H \end{bmatrix} = \begin{bmatrix} \hat{p}^A_H \\ \hat{p}^B_H \end{bmatrix}$$

for $q_H = (\hat{q}_H^A, \hat{q}_H^B)$ gives

$$\begin{aligned} q_H^B &= [\tilde{A}_{H-1}[G1_H + \tilde{A}_{H-1}(\tilde{A}_{H-1})^T]^{-1}\tilde{A}_{H-1}^T]^{-1}(\hat{p}_H^B - \tilde{A}_{H-1}[G1_H + (\tilde{A}_{H-1})(\tilde{A}_{H-1})^T]^{-1}\hat{p}_H^A) \\ q_H^A &= [G1_H + \tilde{A}_{H-1}(\tilde{A}_{H-1})^T]^{-1}(\hat{p}_H^A - \tilde{A}_{H-1}^T q_H^B) \end{aligned}$$

Let $\overline{G1}_H = G1_H + (\tilde{A}_{H-1})(\tilde{A}_{H-1})^T$. Then finding q_H requires finding the solutions of the form $\overline{G1}_H\, x = y$. Since $G1_H$ is a block diagonal matrix, and $\tilde{A}_{H-1}$ is the constraint matrix corresponding to another (smaller) multistage problem, $\overline{G1}_H$ will have the same block structure as $\tilde{M}_H$, the projection matrix of the original problem. $\overline{G1}_H$ may then be decomposed into a sum of block diagonal matrices and the product of two similar matrices. Let

$$S_{k,H-1} = \begin{cases} I_{H-2} + \tilde{D}_{H-2} & k = 0 \\ (W_{k,H-1})(W_{k,H-1})^T + D_{k,H-1}^2 + & \\ \quad \sum_{k'=1}^{N_t}(T_{\beta(k',k),H})^T(S_{\beta(k',k),H})^{-1}T_{\beta(k',k),H} & k = 1,\ldots,\bar{N}_{H-1} \end{cases} \tag{9}$$

and $S_{H-1} = \mathrm{diag}\{S_{0,H-1},\ldots,S_{\bar{N}_{H-1},H-1}\}$. Also, if U_{H-1} and V_{H-1} are defined similarly as above (using $\tilde{A}_{H-2}$ instead of $\tilde{A}_{H-1}$), then $\overline{G1}_{H-1} = S_{H-1} + (U_{H-1})(V_{H-1})^T$, and Theorem 1 may again be invoked. Since the proof of the basic step in this decomposition was proved in Section 2.3.1, we will skip a formal proof here. Instead, we will proceed to define a pseudo-code algorithm to find interior point iterates. The algorithm is

---

Procedure **finddy** (S, $\tilde{A}$, U, V, t, b, dy) **begin**

1. (Solve $Sp = b$). Solve $S_{k,t}p_{k,t} = b_{k,t}$ for $p_{k,t}, k = 0,\ldots,\bar{N}_t$

2. (Solve $G_t q_t = V_t^T p_t$).

   (a) Solve

   $$S_{\beta(k',k),t}(u_{\beta(k',k),t})_i = (T_{\beta(k',k),t})_{\cdot i}$$

   for $u_{\beta(k',k),t})_i, i = 1,\ldots,n_{\beta(k',k),t}, k' = 1,\ldots,N_t, k = 1,\ldots,\bar{N}_{t-1}$.

   (b) Set $(G1_{0,t})_{ii} = (\tilde{D}_{t-2}^2)_{ii}$ for $i = 1,\ldots,\tilde{n}_{t-2}$.
   Set $(G1_{k,t})_{\cdot i} = (D_{k,t-1}^2)_{ii} + \sum_{k'=1}^{N_t}(T_{\beta(k',k),t})^T(u_{\beta(k',k),t})_i$, $i = 1,\ldots,n_{\beta(k',k),t}, k = 1,\bar{N}_{t-1}$.
   Let $G1_t = \mathrm{diag}\{G_{0,t},\ldots,G_{\bar{N}_{t-1},t}\}$.

   (c) Form $\hat{p}^A$ and $\hat{p}^B$. Let

   $$\hat{p}_{k,t} = \begin{cases} \tilde{A}_{t-1}^T p_{k,t} + \sum_{k'=1}^{N_t}(T_{\beta(k',k),t})^T(p_{\beta(k',k),t}) & k = 1,\ldots,\bar{N}_{t-1} \\ -p_{k',t} & k = \bar{N}_{t-1}+1 \ldots 2\bar{N}_{t-1} \\ & \quad k' = k - \bar{N}_{t-1} + 1 \end{cases}$$

   Let $\hat{p}^A = (\hat{p}_{1,t},\ldots,\hat{p}_{\bar{N}_{t-1}})$ and $\hat{p}^B = (\hat{p}_{\bar{N}_{t-1}+1,t},\ldots,\hat{p}_{2\bar{N}_{t-1}})$.

   (d) Form $\overline{G1}_t = G1_t + (\tilde{A}_{t-1})(\tilde{A}_{t-1})^T$.

   (e) Let S' be defined as in (9).

   (f) Solve $\overline{G1}_t u = \hat{p}_t^A$ for $t > 1$ by calling **finddy** ($S', \tilde{A}_{t-1}, U_{t-1}, V_{t-1}, t-1, \hat{p}_t^A, u$) .
   If $t = 1$, solve $\overline{G1}_t u = \hat{p}_t^A$ directly. Set $v = \hat{p}^B - \tilde{A}_{t-1}u$.

   (g) Solve $\overline{G1}_t w_i = (\tilde{A}_t)_{\cdot i}^T$ for $t > 1$ by calling **finddy** ($S', \tilde{A}_{t-1}, U_{t-1}, V_{t-1}, t-1, (\tilde{A}_t)_{\cdot i}^T, w_i$) .
   For $t = 1$, solve $\overline{G1}_t w_i = (\tilde{A}_t)_{\cdot i}^T$ directly. Set $G2 = \tilde{A}_t[w_1 \cdots w_{\tilde{m}_t}]$.

(h) Solve $G2q^B = v$ for q^B and solve $\overline{G1}_t q^B = \hat{p}^A - (\tilde{A}_{t-1})^T q^B$ for q^B by calling
finddy ($S', \tilde{A}_{t-1}, U_{t-1}, V_{t-1}, t-1, \hat{p}^A - (\tilde{A}_{t-1})^T q^B, q^B$) for $t > 1$.
Solve $G2q^B = v$ for q^B directly if $t = 1$.

3. (Solve $Sr = Uq$). Let $\hat{q}_{0,t} = \tilde{A}_{t-1} q^A + q^B$. Let $\hat{q}_{k,t} = \tilde{T}_{k,t} q^A, k = 1, \ldots, \bar{N}_{t-1}$. Let $\hat{q}_{k',t} = (\hat{q}_{\beta(k,1),t}, \ldots, \hat{q}_{\beta(k,N_t),t})$ Solve $S_{k',t} r_{k',t} = \hat{q}_{k',t}$ for $r_{k',t}$ for $k' = 1, \ldots, \bar{N}_t$.

4. Set $dy_{k,t} = p_{k,t} - r_{k,t}$ for $k = 1, \ldots, \bar{N}_t$.

end.

Forming the Multistage dy using Birge and Qi's Decomposition

Each iterate of the interior point algorithm is started by calling **finddy** $(S_H, \tilde{A}_H, U_H, V_H, H, b, dy)$. Implementing the above algorithm requires setting up a "line" of Cholesky factorizations, with those involving $S_{k,H}$ on one end, and those involving $S_{k,1}$ on the other. Finding the search direction then involves passing data from the $S_{k,H}$ end to the $S_{k,1}$ end, and collecting results in the reverse order. Parallel processes may be exploited at each stage by assigning sets of $S_{k,t}$ factorizations to concurrent processing units.

Theorem 2 (Time Complexity of Multistage Birge and Qi decomposition) *Let N_t be the number of distinct possibilities in stage t, $\bar{N}_t = N_1 \times \cdots \times N_t$ be the number of cumulative scenarios through stage t ($N_1 = 1$), and $\tilde{N}_t = \sum_{t'=1}^{t} \bar{N}_{t'}$ be the cumulative number of nonzero blocks in a multistage SLP with t stages. Let $\bar{N}_A = \sum_{t=1}^{H} \bar{N}_t$, $\bar{N}_B = \sum_{t=1}^{H-1} \bar{N}_t$, $\tilde{N}_A = \sum_{t=1}^{H-1} \tilde{N}_t^2$, and $\tilde{N}_B = \sum_{t=1}^{H-1} \tilde{N}_t^3$. Assuming $T_{k,t}$ and $W_{k,t}$ have dimension $m \times n$ for all k, t, the overall time complexity of the Birge and Qi decomposition is*

$$O(\bar{N}_A(n^3 + mn + mn^2) + \bar{N}_B(n^3 + n^2) + \tilde{N}_A mn + \tilde{N}_B(m + 2mn^2))$$

Proof: Let $T(t)$ be the time complexity of the Birge and Qi decomposition for a problem with t stages. At each step in the algorithmic statement given above, we have the following time complexities:

Step	Work	Complexity
1	Solve $Sp = b$	$O(\bar{N}_t(n^3 + mn)$
2(a)	Solve $S^{-1}T$	$O(\bar{N}_{t-1}(n^3 + n^2))$
2(b)	Find $G1$	$O(\bar{N}_t mn^2)$
2(c)	Form $\hat{p}$	$O(\bar{N}_t mn^2 + \tilde{N}_t^2(mn))$
2(d)	Form $\overline{G1}_t$	$O(\tilde{N}_t^3(mn))$
2(e)	Form S'	Obtainable from 2(b)
2(f), 2(g)	Find $G2$	$O(\tilde{N}_t^3(mn^2) + \tilde{N}_t^2(mn) + 2T(t-1))$
2(h)	Solve for q	$O(\tilde{N}_t^3(m^3) + \tilde{N}_t^2(mn) + T(t-1))$
3	Solve $Sr = Uq$	$O(\bar{N}_t(mn))$

Other computational efforts are small compared to the above operations. Adding the time complexities for a generic stage t gives

$$O(\bar{N}_t(n^3 + 2mn + 2mn^2) + \bar{N}_{t-1}(n^3 + n^2) + 3(\tilde{N}_t^2 mn + \tilde{N}_t^3(m + 2mn^2) + 3T(t-1)))$$

Since the procedure will be called $H-1$ times, the time complexity of the algorithm can simplify to

$$O\left(\sum_{t=1}^{H}\bar{N}_t(n^3+2mn+2mn^2)+\sum_{t=1}^{H-1}(n^3+n^2)+\sum_{t=1}^{T}(\tilde{N}_t^2(mn)+\tilde{N}_t(m+2mn^2))\right)$$

Redefining the sums as above simplifies the time complexity to

$$O(\bar{N}_A(n^3+mn+mn^2)+\bar{N}_B(n^3+n^2)+\tilde{N}_Amn+\tilde{N}_B(m+2mn^2)) \quad \square$$

3 Conclusion

This technical report has presented a multistage generalization of the Birge-Qi factorization for solving two stage stochastic linear programs using interior point methods. The Birge-Qi factorization has been shown in (Birge and Holmes [20]) to be computationally superior and numerically more stable than the most common solution tricks available. The same benefits can be extended to multistage stochastic linear programs by noting that multistage problems can be written as nested two-stage problems, and applying the same idea.

References

[1] J.M. Mulvey, 1984. "A Network Portfolio Approach for Cash Management," *Journal of Cash Management 4*, 46-48.

[2] F.V. Louveaux, 1980. " A solution method for multistage stochastic programs with recourse with application to an energy investment problem,", *Operations Research 28*, 889-902.

[3] H.I. Gassman, to appear. "Optimal harvest of a forest in the presence of uncertainty," *Canadian Journal of Forest Research.*

[4] R. Marsten, R. Subramanian, M. Saltzman, I. Lustig, and D. Shanno, 1990. "Interior Point Methods for Linear Programming: Just call Newton, Lagrange, and Fiacco and McCormick!," *Interfaces 20:4*, 105-116.

[5] W.J. Carolan, J.E. Hill, J.L. Kennington, S. Niemi, and S.J. Wichmann, 1990. "An empirical evaluation of the KORBX algorithms for military airlift applications," *Operations Research 9:2*, 169-184.

[6] C.L. Monma and A.J. Morton, 1987. "Computational Experience with a Dual Affine Variant of Karmarkar's Method for linear Programming," Manuscript, Bell Communications Research.

[7] J. R. Birge and L. Qi, 1988. "Computing Block-angular Karmarkar Projections with Applications to Stochastic Programming," *Management Science 34:12*, 1472-1479.

[8] P.E. Gill, W. Murray, M.A. Saunders, J.A. Tomlin, and M.H. Wright, 1986. "On Projected Newton Methods for Linear Programming and Equivalence to Karmarkar's Projection Method," *Mathematical Programming 36*, 183-201.

[9] N. Karmarkar, 1984. "A New Polynomial Time Algorithm for Linear Programming," *Combinatorica 4*, 373-395.

[10] I. Adler, N. Karmarkar, M.G.C. Resende, and G. Veiga, 1986. "An Implementation of Karmarkar's Algortihm for Linear Programming," Report No. ORC 86-8 (Revised May, 1987), Operations Research Center, University of California, Berkely, California.

[11] N. Megiddo, 1986. "Pathways to the Optimal Set in Linear Programming," Technical Report RJ 5295, IBM Almaden Research Center, San Jose, California.

[12] D.F. Shanno, and A. Bagchi, 1990. "A Unified View of Interior Point Methods for Linear Programming," *Annals of Operations Research 22*, 55-70.

[13] G.H. Golub and C.F. van Loan, 1983. **Matrix Computations**, The Johns Hopkins University Press, Baltimore, Maryland.

[14] J.A. Meijerink and H.A. van der Vorst, 1977. "An Iterative Solution Method for Linear Equation Systems of which the Coefficient matrix is a Symmetric M-matrix," *Mathematical Computations, 31* 148-162.

[15] S.C. Eisenstadt, M.C. Gurshy, M.H. Shultz, and A.H. Sherman, 1981. "The Yale Sparse Matrix Package, I. The Symmetric Codes," *ACM Transactions on Mathematical Software.*

[16] E. Chu, A. George, J. Liu, and E. Ng, 1984. "SPARSPAK: Waterloo Sparse Matrix Package User's Guide for SPARSPAK-A," Research Report CS-84-36, Department of Computer Science, University of Waterloo, Waterloo, Ontario.

[17] J. Arantes and J. Birge, 1989. "Matrix Structure and Interior Point Methods in Stochastic Programming," Presentation, *Fifth International Stochastic Programming Conference*, Ann Arbor, Michigan.

[18] H. I. Gassman, 1990. "MSLiP: A computer code for the multistage stochastic linear programming problem." *Mathematical Programming*, 47, pp. 407-423.

[19] J. R. Birge, 1985. "Decomposition and partitioning methods for multistage stochastic linear programs," *Operations Research*, 33, pp. 989-1007.

[20] J. R. Birge, and D. Holmes, 1992. "Efficient solution of two-stage stochastic linear programs using interior point methods," *Computational Optimization and Applications*, 1, pp. 245-276.