Journal of Global Optimization **28:** 197–215, 2004.

A New Decomposition Technique in Solving Multistage Stochastic Linear Programs by Infeasible Interior Point Methods

XINWEI LIU[1] and JIE SUN[2]
[1]*Department of Decision Sciences, National University of Singapore, and Faculty of Science and Arts, Hebei University of Technology, Tianjin, China (e-mail: smaliuxw@nus.edu.sg);*
[2]*Department of Decision Sciences and Singapore-MIT Alliance, National University of Singapore (e-mail: jsun@nus.edu.sg).*

(Received and accepted in revised form 21 January 2003)

Abstract. Multistage stochastic linear programming (MSLP) is a powerful tool for making decisions under uncertainty. A deterministic equivalent problem of MSLP is a large-scale linear program with nonanticipativity constraints. Recently developed infeasible interior point methods are used to solve the resulting linear program. Technical problems arising from this approach include rank reduction and computation of search directions. The sparsity of the nonanticipativity constraints and the special structure of the problem are exploited by the interior point method. Preliminary numerical results are reported. The study shows that, by combining the infeasible interior point methods and specific decomposition techniques, it is possible to greatly improve the computability of multistage stochastic linear programs.

Key words: Stochastic linear programs, Infeasible primal-dual interior point method, Scenario analysis, Decomposition

1. Introduction

1.1. THE STOCHASTIC LINEAR PROGRAMMING MODEL

Multistage stochastic linear programming has extensive applications in production and manpower planning, portfolio selections, and many other management problems. A typical form of this model is as follows:

$$\min c_0^T x + E_{\xi_1}(\min q_1(\xi_1)^T y_1 + \cdots + E_{\xi_{T-1}}(\min q_{T-1}(\xi_{T-1})^T y_{T-1})) \tag{1.1}$$

$$\text{s.t. } Ax = b, \quad x \geqslant 0, \tag{1.2}$$

$$T_1(\xi_1)x + W_1(\xi_1)y_1 = h_1(\xi_1), \quad y_1 \geqslant 0, \tag{1.3}$$

$$T_k(\xi_k)y_{k-1} + W_k(\xi_k)y_k = h_k(\xi_k), \quad y_k \geqslant 0, \quad k = 2, \ldots, T-1 \tag{1.4}$$

where $x \in \Re^{n_0}$ and $y_k \in \Re^{n_k}$, $\xi_k$is a random vector associated with stage $k+1$. The superscript "T" represents the transpose and the letter "E" denotes the expected value. $T_i(\xi_k)$ and $W_i(\xi_k)$ are random matrices, $q_i(\xi_k)$ and $h_i(\xi_k)$ are random vectors, all of them are decided by the realization of the random vector $\xi = (\xi_1, \ldots, \xi_{T-1})$. For convenience of computation, it is often assumed that the size of the support of ξ is finite.

The problem described by (1.1)–(1.4) is a T-stage stochastic linear program with recourse. In the literature many algorithms have been designed for the special case of T = 2. We refer to [3,6–9,13] and the references therein for the related works. The recent work of Berkelaar et al. [3] is particularly interesting since it not only takes the problem structure into account, but also addresses the possible infeasibility of the problem.

Much of the research on multistage problems has been focused on decomposition techniques associated with different solution ideas. The L-shaped methods, for example, use a Benders decomposition of (1.1)–(1.4) to generate sets of feasibility cuts and optimality cuts alternatively until the optimal solution is obtained. Some other methods are based on Benders/Dantzig–Wolfe decomposition associated with cutting plane methods [1, 2] and log-barrier methods [30]. There also have been methods based on nonlinear programming approaches, e.g., [7,21,23]. A drawback of the nonlinear programming approach is that a phase-one algorithm is needed to generate a feasible point of the original problem, which is very expensive for multistage stochastic linear programs and often reduces the efficiency of the methods (e.g., see [7]). The seminal progressive hedging method (PHA for short) developed by Rockafellar and Wets [20] is generally shown to be effective [10–12,19]. It is however noted that the selection of a penalty parameter β in PHA is difficult and an unsuitable β may result in slow convergence.

With the rapid growth and development in interior point methods, applying interior point methods to solving large-scale stochastic programs has been a focal point of recent research, see [3–5,24]. For a nice expository article on this subject, see Zhang [27]. In this paper we present a new decomposition approach for solving multistage stochastic linear programs. Compared to the literature, the new features of this method include that

- the method is based on scenario decomposition (see below for details) rather than recourse decomposition (see, e.g., [4, 27, 29] for details);
- there is a preprocessing mode that can detect inconsistency of the constraints at an early stage of the algorithm;
- the sparsity of the nonanticipativity constraints and the special structure of the problem are exploited in the implementation; and
- the method is associated with the infeasible potential reduction algorithm rather than other type of interior point methods.

Similar to other decomposition methods, the search direction is generated by solving a set of primal-dual equations of size much less than the original problem and the solution process is parallelizable.

1.2. SCENARIO FORMULATION OF MSLP

A scenario is a realization of the joint random vector $\xi=(\xi_1,\ldots,\xi_{T-1})$. Assume that (Ω,Θ,P) is the associated probability space, where the support Ω is finite. Thus, there is a finite number of scenarios. Let the probability distribution of ξ be $\{(\xi^{(s)},p_s)|s=1,2,\ldots,S\}$ where $\xi^{(s)}=(\xi_1^{(s)},\xi_2^{(s)},\ldots,\xi_{T-1}^{(s)})$ and S is the number of scenarios.

Each scenario is associated with a sequence of decisions: $x_0^{(s)},y_1^{(s)},\ldots,y_{T-1}^{(s)}$. For simplicity of notation let $z_0^{(s)}=x_0^{(s)},z_1^{(s)}=y_1^{(s)},\ldots,z_{T-1}^{(s)}=y_{T-1}^{(s)}$ and let $z^{(s)}=(z_0^{(s)};\cdots;z_{T-1}^{(s)})\in\Re^n$ (where $n=\sum_{k=0}^{T-1}n_k$) be the decision vector associated with the s-th scenario. Let

$$B_s=\begin{bmatrix} A & & & & \\ T_1(\xi_1^{(s)}) & W_1(\xi_1^{(s)}) & & & \\ & T_2(\xi_2^{(s)}) & W_2(\xi_2^{(s)}) & & \\ & & \cdots & & \\ & & & T_{T-1}(\xi_{T-1}^{(s)}) & W_{T-1}(\xi_{T-1}^{(s)}) \end{bmatrix}\in\Re^{m\times n}, \tag{1.5}$$

$b_s=(b;h_1(\xi_1^{(s)});\ldots;h_{T-1}(\xi_{T-1}^{(s)}))$, and $c_s=p_s(c_0;q_1(\xi_1^{(s)});\ldots;q_{T-1}(\xi_{T-1}^{(s)}))$. Then a deterministic equivalent problem of (1.1)–(1.4) is the following linear programming problem.

$$\min\sum_{s=1}^{S}c_s^T z^{(s)} \tag{1.6}$$

$$\text{s.t.}\quad B_s z^{(s)}=b_s,\ z^{(s)}\geqslant 0,\ s=1,\ldots,S \tag{1.7}$$

$$Nz=0, \tag{1.8}$$

where $z=(z^{(1)};z^{(2)};\ldots;z^{(S)})\in\Re^{nS}$, N is selected such that constraints in (1.8) reflect the fact that scenarios sharing a common history up to any moment of time must also have a common decision up to that moment.

With a large number of scenarios, program (1.6)–(1.8) may have very large size. Thus, decomposition techniques play an important role in the development of the algorithms, (see, e.g., [22]). Moreover, the parallel computers and Internet provide additional computing power if most of the decomposed computations can be split into jobs independent from each other.

1.3. NONANTICIPATIVITY CONSTRAINTS

Constraints in (1.8) are the so-called nonanticipativity constraints, which merely indicate the fact that if scenarios i and j have the same history up to stage k_{ij}, then the decision vectors $z^{(i)}$ and $z^{(j)}$ should satisfy

$$z_0^{(i)}=z_0^{(j)},z_1^{(i)}=z_1^{(j)},\ldots,z_{k_{ij}-1}^{(i)}=z_{k_{ij}-1}^{(j)}, \tag{1.9}$$

where $i,j\in\{1,2,\ldots,S\}$.

There are many equivalent forms of nonanticipativity constraints. Different algorithms may use different forms of them. Let i and $i+1$ be two consecutive indices in the scenario sequence $1,2,\ldots,S$. Suppose that scenarios i and $i+1$ share the same history up to stage k_i. We will use the form

$$z_t^{(i)}=z_t^{(i+1)},\ t=0,\ldots,k_i-1;\ i=1,\ldots,S-1, \tag{1.10}$$

where $z_t^{(i)}$ and $z_t^{(i+1)}$ are some of the subvectors of decision vectors $z^{(i)}$ and $z^{(i+1)}$ respectively. In this case N is a sparse and structured matrix (e.g., all entries are 0 or ±1). However, the combined system (1.7)–(1.8) may have some redundancy, which presents a difficulty in using the interior point methods under our consideration.

1.4. ORGANIZATION AND SOME NOTATIONS

This paper is organized as follows. In Section 2 we introduce the primal-dual infeasible-interior-point method for linear programming and apply it to problem (1.6)–(1.8). A procedure for deleting redundant nonanticipativity constraints is given. The overall decomposition scheme on the direction-finding subproblem is presented. In Section 3 we give a specific decomposition algorithm for a key equation in the direction-finding subproblem. Then the algorithm is proposed. Some preliminary numerical results are reported in Section 4. Our notations are consistent to most of the literatures in stochastic programming. The superscript (s) represents the s-th scenario, for example, $z^{(s)}$ is the decision vector associated with the s-th scenario. A subscript j usually designates the j-th subvector or the j-th component of a vector. Usually, capital letters are for matrices, lower case letters stand for vectors, and Greek letters denote scalars.

2. Applying infeasible interior point method to multistage stochastic linear programs

2.1. INFEASIBLE INTERIOR POINT METHOD FOR LINEAR PROGRAMMING

Consider the linear program in standard form

$$\min c^T x \tag{2.1}$$

$$\text{s.t.}\, Ax = b,\; x \geqslant 0 \tag{2.2}$$

and its dual problem

$$\max b^T y \tag{2.3}$$

$$\text{s.t.}\, A^T y + z = c,\; z \geqslant 0. \tag{2.4}$$

The so-called infeasible interior point methods start from some $x_0 > 0$ and $z_0 > 0$ and y_0, but $Ax_0 - b$ may not be zero and z_0 may not equal to $c - A^T y_0$, that is, (x_0, y_0, z_0) may not be feasible for the primal and dual programs and it is an interior but possibly infeasible starting point. At each iteration the infeasible interior point methods generate the search direction (d_x, d_y, d_z) by solving the system of linear equations

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} = - \begin{bmatrix} Ax - b \\ A^T y + z - c \\ Xz - \mu e \end{bmatrix} \tag{2.5}$$

for a scalar $\mu > 0$, where $Z = \text{diag}(z)$ $X = \text{diag}(x)$, and $e = (1, \ldots, 1)^T \in \Re^n$. Then a line search is performed based on different criteria and μ is updated such that $\mu \downarrow 0$. At termination the algorithms can find primal and dual optimal solutions if they exist or detect that such optimal solutions do not exist. The differences of various methods reside in the way to reduce μ, the way to do line searches, or even the form of right hand side of (2.5). See for examples [14,15,17,28,25] for details. These methods are generally regarded as among the most efficient interior point methods for large-scale linear programming. In theory, most of them have polynomial worst-case complexity.

Two features of the infeasible interior point methods are important for multistage stochastic linear programs. First, they start from an infeasible point. Second, the methods can detect infeasibility. Due to the complexity of multistage problems, a multistage stochastic linear program is often intrinsically ill-modeled. Thus, a method that detects infeasibility can provide important information for the user.

A popular version of the infeasible interior point methods is the Mizuno–Kojima–Todd (MKT) primal-dual potential reduction algorithms [18]. In addition to solving system (2.5), in their algorithms the function

$$\phi(x,z) = (n+\nu)\ln(x^T z) - e^T \ln(Xz) - n \ln n \tag{2.6}$$

and its variant

$$\begin{aligned} \psi(x,y,z) = {} & (n+\nu+1)\ln(x^T z) - e^T \ln(Xz) - n\ln n - \\ & \ln(x^T z - \sigma \|(Ax-b, A^T y + z - c)\|) \end{aligned} \tag{2.7}$$

are taken as the potential functions for their Algorithm I and Algorithm II, respectively, where $\nu > 0$ and $\sigma > 0$ are constants. The line search is done by selecting the stepsize $\alpha \in (0, 1]$ such that

$$\phi(x+\alpha d_x, z+\alpha d_z) - \phi(x,z) \leqslant -\delta \tag{2.8}$$

$$(x+\alpha d_x)^T(z+\alpha d_z) - (1-\alpha)x^T z \leqslant 0 \tag{2.9}$$

for Algorithm I and

$$\psi(x+\alpha d_x, y+\alpha d_y, z+\alpha d_z) - \psi(x,y,z) \leqslant -\delta \tag{2.10}$$

for Algorithm II for a given constant $\delta > 0$. Since the implementation of Algorithm I is straightforward, we will use this algorithm in our computational test although other infeasible interior point methods can be applied without essential difference.

2.2. PREPROCESSING: FULL RANK REDUCTION

Any efficient implementation of interior point methods requires that the matrix A in (2.2) has full row rank. This is not an easy task when it comes to problem (1.6)–(1.8). Let

$$B = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \cdots & \\ & & & B_S \end{bmatrix} \text{ and } b = \begin{bmatrix} b_1 \\ \vdots \\ b_S \end{bmatrix}. \tag{2.11}$$

Let us call problem (1.6)–(1.8) *scenario-consistent* if the set $\{z | Bz = b\}$ is not empty. This consistency can be decided by checking the consistency of the systems $B_s z^{(s)} = b_s$ for $s = 1, \ldots, S$. This checking process will enable us to terminate the algorithm early in case the problem is inconsistent. There are some computer packages which can be utilized for this purpose. For example, the "*rank*" command in MATLAB can give the rank of B_s and $[B_s, b_s]$. Thus we find inconsistency whenever $\text{rank}[B_s] < \text{rank}[B_s, b_s]$ for some s. By checking consistency and deleting redundant constraints (if any) in $[B_s, b_s], s = 1, \ldots, S$, we may assume that B is of full row rank. Suppose that the nonanticipativity constraints in (1.8) take the form of (1.10). Then N is also of full row rank.

Even if both B and N are of full row rank, the matrix $(B^T, N^T)^T$ may not be of full row rank. See the following example.

EXAMPLE 2.1. Consider a simple three-stage stochastic linear program with constraints (1.2)–(l.4) being

$$x_1 - x_2 = 1, \quad x_1 \geqslant 0, x_2 \geqslant 0, \tag{2.12}$$

$$2x_1 + x_2 - y_1 = \xi_1, \quad y_1 \geqslant 0, \tag{2.13}$$

$$2y_1 - y_2 = \xi_2, \quad y_2 \geqslant 0. \tag{2.14}$$

Suppose that ξ_1 and ξ_2 have two realizations respectively, thus $S=4$. So N defined by (1.10) is

$$N=\begin{bmatrix} I\,0\,0 & -I & 0\,0 & & & & \\ 0\,1\,0 & 0 & -1\,0 & & & & \\ & I & 0\,0 & -I\,0\,0 & & & \\ & & & I\,0\,0 & -I & 0\,0 \\ & & & 0\,1\,0 & 0 & -1\,0 \end{bmatrix}, \tag{2.15}$$

where I is a 2×2 unit matrix. Hence N is a 8×16 matrix. It is easy to note that B defiined by (2.11) is a 12×16 matrix. Thus, $(B^T,N^T)^T$ is not of full row rank. The example is not specially constructed, so it reflects a common fact for stochastic linear programs. For instance, the test problems randomly generated in Section 4, about 20–30% of the rows in $(B^T,N^T)^T$ are redundant.

We consider how to eliminate constraints in (1.10) such that $(B^T,N^T)^T$ is of full row rank. Since constraints in (1.7) have full row rank, we only have to eliminate some of the nonanticipativity constraints. Notice that the structure of the left hand side of equality constraints in (1.7)–(1.8) is

$$A=\begin{bmatrix} B \\ N \end{bmatrix}=\begin{bmatrix} B_1 & & & & & \\ & B_2 & & & & \\ & & B_3 & & & \\ & & & \cdots & & \\ & & & & B_{S-1} & \\ & & & & & B_S \\ [I_1,0] & [-I_1,0] & & & & \\ & [I_2,0] & [-I_2,0] & & & \\ & & & \cdots & & \\ & & & & [I_{S-1},0] & [-I_S,0] \end{bmatrix}, \tag{2.16}$$

where $I_i, i=1,\cdots,S$ are unit matrices of suitable sizes, depending on the number of decision variables in common of scenarios i and $i+1$. By re-arranging the rows, we have

$$\operatorname{rank}(A)=\operatorname{rank}\begin{bmatrix} B_1 & & & & & \\ [I_1,0] & [-I_1,0] & & & & \\ & B_2 & & & & \\ & [I_2,0] & [-I_2,0] & & & \\ & & B_3 & & & \\ & & & \cdots & & \\ & & & & [I_{S-1},0] & [-I_{S-1},0] \\ & & & & & B_S \end{bmatrix}, \tag{2.17}$$

The special block diagonal structure of A suggests the following approach. Firstly, after suitable column re-ordering, the resulting system from A (denoted by $\bar{A}$) has the form

$$\bar{A}z := \begin{bmatrix} [M_1, J_1] & & & & & & \\ K_1 & L_1 & & & & & \\ & [M_2, J_2] & & & & & \\ & K_2 & L_2 & & & & \\ & & [M_3, J_3] & & & & \\ & & & \cdots & & & \\ & & & & K_{S-1} & L_{S-1} & \\ & & & & & [M_S, J_S] & \end{bmatrix} z = \begin{bmatrix} b_1 \\ c_1 \\ b_2 \\ c_2 \\ \vdots \\ b_S \end{bmatrix} =: \bar{b}, \tag{2.18}$$

where $M_s, s = 1, \ldots, S$ are square nonsingular matrices. That is, we normalize each B_s such that the leftmost square block in B_s is nonsingular for all s. In doing so, each block of the form $[I, 0][-I, 0]$ becomes the form $[KL]$.

Next, we do block Gaussian elimination as follows. We first eliminate part of K_1 using M_1 resulting in $[0, K_1']$. Then using M_2 to eliminate both part of L_1 and part of K_2, resulting $[0, L_1']$ and $[0, K_2']$, respectively. We proceed this using $M_3, \ldots,$ until M_S. We obtain an equivalent system

$$\tilde{A}z := \begin{bmatrix} [M_1, J_1] & & & & & & \\ [0, K_1'] & [0, L_1'] & & & & & \\ & [M_2, J_2] & & & & & \\ & [0, K_2'] & [0, L_2'] & & & & \\ & & [M_3, J_3] & & & & \\ & & & \cdots & & & \\ & & & & [0, K_{s_1}'] & [0, L_{S-1}'] & \\ & & & & & [M_S, J_S] & \end{bmatrix} z = \begin{bmatrix} \bar{b}_1 \\ \bar{c}_1 \\ \bar{b}_2 \\ \bar{c}_2 \\ \vdots \\ \bar{b}_S \end{bmatrix} =: \tilde{b}, \tag{2.19}$$

We discuss three cases.

Case 1. K_1' has full row rank. By changing column order in K_1' we may write K_1' as $[M_1', J_1']$. Thus the first three rows of $\bar{A}$ can be written as

$$\begin{bmatrix} M_1 \cdots & \cdots & \cdots \cdots \\ & M_1' \cdots & \cdots \cdots \\ & & M_2 \cdots \cdots \end{bmatrix} \tag{2.20}$$

without affecting other part of $\widetilde{A}$.

Case 2. Not K_1' but $[K_1', L_1']$ has full row rank. Let this rank be m. Then we interchange independent columns in L_1' with nonindependent columns of K_1' so that in the resulting second row in (2.19) the leftmost m columns of K_1' are independent.

The resulting $\widetilde{A}$ will have the following form

$$\widetilde{A} = \begin{bmatrix} [M_1, J_1'] & & & & & \\ [0, K_1''] & [0, L_1''] & & & & \\ [0, J_2'] & [M_2, J_2''] & & & & \\ [0, K_2'''] & [0, K_2''] & [0, L_2'] & & & \\ & & [M_3, J_3] & & & \\ & & & \cdots & & \\ & & & & [0, K_{S-1}'] & [0, L_{S-1}'] \\ & & & & & [M_S, J_S] \end{bmatrix}, \tag{2.21}$$

where the first m columns of K_l'' are independent and only the first m columns of J_2' and K_2''' can be nonzero. Then use the first m columns of K_1'' to eliminate the first m columns of J_2' and K_2'''. This may result in that the $m+1st$, $m+2nd, \ldots$ columns of J_2' and K_2''' become nonzero. If this happens, we move these columns of $\tilde{A}$ to the right of J_2''. The resulting first three columns of (2.19) will take the form of (2.20). Thus, in this case we obtain the same row-echelon form as in Case 1.

Case 3. $[K_1', L_1']$ is not of full row rank. Then either $[K_1', L_1', c_1]$ contains a redundant row or $\text{rank}[K_1', L_1'] < \text{rank}[K_1', L_1', c_1]$ that shows inconsistency of the system. We delete a redundant row or terminate the algorithm, respectively. If the system is consistent and after all redundant rows are deleted, the case reduces to Case 1 or Case 2 above.

So far we have shown that, if the system is consistent, then the first three rows can be made to have the so-called row-echelon form (2.20), in which M_1, M_1', M_2 are nonsingular square matrices. Proceed from this form in the same way for the third, fourth and fifth rows and so on, we will either stop at a certain step and claim that the system is inconsistent or we will eventually reduce the system to an equivalent block row-echelon form with all first square sub-blocks of the row-blocks being nonsingular. This shows that the matrix $\bar{A}$ has full row rank.

We record all deleted rows in this process and delete the corresponding rows of N, resulting in $\bar{N}$. In the end we obtain a modified system $(B^T, (\bar{N})^T)^T$ and start the interior point method with it. In other words, the above process is just used to eliminate redundancy from the original system. The structure of the original system is preserved for the decomposition method in the next section.

We remark that while at the first glance the above preprocessing looks very tedious, it is important to notice that the process is fully parallelizable in the sense that the processing of blocks $1-3, 3-5, 5-7, \ldots$ are independent and hence can be done in parallel and in arbitrary order.

Back to the example, (2.15) in Example 2.1 may be reduced to

$$\begin{bmatrix} 1 & 0_{1\times 3} & -1 & 0_{1\times 3} & & & & \\ & & 1 & 0_{1\times 3} & -1 & 0_{1\times 3} & & \\ & & & & 1 & 0_{1\times 3} & -1 & 0_{1\times 3} \end{bmatrix} \tag{2.22}$$

In summary, we have shown that we can detect the inconsistency of system

$$\begin{bmatrix} B \\ N \end{bmatrix} z = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

and in case that the system is consistent we can remove all redundant equations so that the resulting system

$$\begin{bmatrix} B \\ \bar{N} \end{bmatrix} z = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

has full row rank. In view of this, from now on we assume that the matrix (B^T, N^T) in (1.7)–(1.8) is of full column rank. We will equivalently speak of that $(B^T, N^T)^T$ has full row rank and that (B^T, N^T) has full column rank.

2.3. DECOMPOSING THE DIRECTION-FINDING PROBLEM

In this subsection we consider the application of the Mizuno-Kojima-Todd method to the problem (1.6)–(1.8). A minor trick is used to develop our decomposition method. Rather than taking primal constraints (1.7) and (1.8) as an integration as done by direct extension of the interior point method, we separate (1.7) and (1.8) into different parts, and do the same for the corresponding dual variables u and w.

Let

$$F(z,u,v,w) = \begin{bmatrix} Bz - b \\ B^T u + v + N^T w - c \\ ZVe \\ Nz \end{bmatrix} \tag{2.23}$$

where $Z = \operatorname{diag}(z_j^{(s)})$, $V = \operatorname{diag}(v_j^{(s)})$. Then the system for finding the search direction (2.5) can be written as the following system of linear equations.

$$J(z,u,v,w) \begin{bmatrix} d_z \\ d_u \\ d_v \\ d_w \end{bmatrix} = -F(z,u,v,w) + \begin{bmatrix} 0 \\ 0 \\ \mu e \\ 0 \end{bmatrix}, \tag{2.24}$$

where J is the Jacobi of $F, \mu = z^T v/(nS)$.

It is easy to compute that

$$J(z,u,v,w) = \begin{bmatrix} B & 0 & 0 & 0 \\ 0 & B^T & I & N^T \\ V & 0 & Z & 0 \\ N & 0 & 0 & 0 \end{bmatrix}. \tag{2.25}$$

Moreover, it can be seen that if $z > 0$ and $v > 0$, then (2.24) is equivalent to solving the following three systems of linear equations:

$$\begin{bmatrix} B & 0 & 0 \\ 0 & B^T & I \\ V & 0 & Z \end{bmatrix} \begin{bmatrix} \tilde{d}_z \\ \tilde{d}_u \\ \tilde{d}_v \end{bmatrix} = - \begin{bmatrix} Bz - b \\ B^T u + v - c \\ ZVe - \mu e \end{bmatrix} \tag{2.26}$$

$$N(V^{-1}Z - V^{-1}ZB^T(BV^{-1}ZB^T)^{-1}BV^{-1}Z)N^T(w + d_w) = -N(z + \tilde{d}_z). \tag{2.27}$$

and

$$\begin{bmatrix} B & 0 & 0 \\ 0 & B^T & I \\ V & 0 & Z \end{bmatrix} \begin{bmatrix} d_z \\ d_u \\ d_v \end{bmatrix} = - \begin{bmatrix} Bz - b \\ B^T u + v + N^T(w + d_w) - c \\ ZVe - \mu e \end{bmatrix} \tag{2.28}$$

The following result shows that the coefficient matrix in (2.27) is positive definite under the full rank assumption.

PROPOSITION 2.1. *Suppose that* $z > 0$ *and* $v > 0$, $Z = \mathrm{diag}(z)$ *and* $V = \mathrm{diag}(v)$. *Let* $M = N(V^{-1}Z - V^{-1}ZB^T(BV^{-1}ZB^T)^{-1}BV^{-1}Z)N^T$. *Then* M *is positive semi-definite. Furthermore, if* (B^T, N^T) *is of full column rank, then* M *is positive definite.*

Proof. Matrix $U(I - V^T(VV^T)^{-1}V)U^T$ is positive semi-definite since the $I - V^T(VV^T)^{-1}V$ is a projection matrix. If (V^T, U^T) is of full column rank, by QR decomposition, we have

$$(V^T, U^T) = [Q_1, Q_2, Q_3] \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \\ 0 & 0 \end{bmatrix} \tag{2.29}$$

where $Q = [Q_1, Q_2, Q_3]$ is a unitary orthogonal matrix, R_{11} and R_{22} are upper triangle matrices with all diagonal entries being nonzero. Thus we have $V = R_{11}^T Q_{11}^T$ and $U = R_{12}^T Q_1 + R_{22}^T Q_2^T$. Hence,

$$\begin{aligned} U(I - V^T(VV^T)^{-1}V)U^T &= UU^T - UV^T(VV^T)^{-1}VU^T \\ &= R_{12}^T R_{12} + R_{22}^T R_{22} - R_{12}^T R_{11}(R_{11}^T R_{11})^{-1} R_{11}^T R_{12} \\ &= R_{22}^T R_{22} \end{aligned} \tag{2.30}$$

The positive definiteness of the left matrix in (2.30) follows from the nonsingularity of R_{22}.

Since $z > 0$ and $v > 0$, let $\bar{N} = NV^{-1/2}Z^{-1/2}$ and $\bar{B} = BV^{-1/2}Z^{-1/2}$, we have

$$M = \bar{N}(I - \bar{B}^T(\bar{B}\bar{B}^T)^{-1}\bar{B})\bar{N}^T. \tag{2.31}$$

Moreover, $(\bar{B}^T, \bar{N}^T)$ is of full column rank if and only if (B^T, N^T) is of full column rank. The desired result follows. □

3. The algorithm

We present our algorithm in this section. The solutions of problems (2.26), (2.27) and (2.28) resulted from decomposition will be specialized in the subsections.

3.1. SOLVE EQUATIONS (2.26) AND (2.28) IN PARALLEL.

We first notice that equation (2.26) can be split into the following systems of equations

$$\begin{bmatrix} B_s & 0 & 0 \\ 0 & B_s^T & I \\ V^{(s)} & 0 & Z^{(s)} \end{bmatrix} \begin{bmatrix} \tilde{d}_z^{(s)} \\ \tilde{d}_u^{(s)} \\ \tilde{d}_v^{(s)} \end{bmatrix} = - \begin{bmatrix} B_s z^{(s)} - b_s \\ B_s^T u^{(s)} + v^{(s)} - c_s \\ Z^{(s)} v^{(s)} - \mu e \end{bmatrix} \tag{3.1}$$

where $s = 1, 2, \cdots, S$. Correspondingly, if we partition N into S blocks, each with n columns as $(N_1, N_2, \cdots, N_S)$, then (2.28) is equivalent to S systems of equations

$$\begin{bmatrix} B_s & 0 & 0 \\ 0 & B_s^T & I \\ V^{(s)} & 0 & Z^{(s)} \end{bmatrix} \begin{bmatrix} d_z^{(s)} \\ d_u^{(s)} \\ d_v^{(s)} \end{bmatrix} = - \begin{bmatrix} B_s z^{(s)} - b_s \\ B_s^T u^{(s)} + v^{(s)} + N_s^T \hat{w} - c_s \\ Z^{(s)} v^{(s)} - \mu e \end{bmatrix} \tag{3.2}$$

where $\hat{w} = \lambda + d_w$. It is easy to note that (3.1) and (3.2) can be solved in parallel for $s = 1, 2, \ldots, S$. In particular, note that the coefficient matrices in (3.1) and (3.2) have the same structure as that in primal-dual methods for standard linear programming, thus all existing theoretical and practical techniques on decomposition for standard linear programming can be included to deal with (3.1) and (3.2) in our method.

3.2. DECOMPOSITION OF (2.27)

How to solve the linear equation (2.27) efficiently is one of the key problems for our algorithm. Since the size of N is comparable to the number of scenarios S, (2.27) can be large when S is very large. Thus, to solve it directly may be very expensive even if the coefficient matrix of (2.27) is symmetric positive definite.

The idea is to try to exploit the sparsity of the coefficient matrix and its special structure. We have the following results, which are similar to that in Section 5 of Zhao [29].

PROPOSITION 3.1. *Consider the tri-block-diagonal matrix*

$$G=\begin{bmatrix} H_1 & U_1 & & & \\ U_1^T & H_2 & U_2 & & \\ & U_2^T & H_3 & & \\ & & \cdots & \cdots & \cdots \\ & & & U_{J-1}^T & H_J \end{bmatrix}, \tag{3.3}$$

where $H_i \in \Re^{l_i \times l_i}(i=1,2,\ldots,J)$*. let*

$$Q_1 = H_1, Q_i = H_i - U_{i-1}^T Q_{i-1}^{-1} U_{i-1} (i=2,\ldots,J).$$

If G *is positive definite, then all* Q_i *are nonsingular, and* G *can be decomposed as*

$$\begin{aligned} G=&\begin{bmatrix} I_1 & & & \\ U_1^T Q_1^{-1} & I_2 & & \\ & \cdots & \cdots & \\ & & U_{J-1}^T Q_{J-1}^{-1} & I_J \end{bmatrix} \begin{bmatrix} Q_1 & & & \\ & Q_2 & & \\ & & \cdots & \\ & & & Q_J \end{bmatrix} \\ &\begin{bmatrix} I_1 & Q_1^{-1}U_1 & & \\ & I_2 & Q_2^{-1}U_2 & \\ & & \cdots & \cdots \\ & & & I_J \end{bmatrix}, \end{aligned} \tag{3.4}$$

where sizes of identity matrices $I_i(i=1,\ldots,J)$ *are the same as* Q_i*.*

Proof. G is positive definite if and only if all of its principal submatrices are positive definite. Thus,

$$H_1 \text{ and } \begin{bmatrix} H_1 & U_1 \\ U_1^T & H_2 \end{bmatrix}$$

are positive definite. Hence, Q_1 is nonsingular. since

$$\begin{bmatrix} I_1 & 0 \\ -U_1^T Q_1^{-1} & I_2 \end{bmatrix} \begin{bmatrix} H_1 & U_1 \\ U_1^T & H_2 \end{bmatrix} \begin{bmatrix} I_1 & -Q_1^{-1}U_1 \\ 0 & I_2 \end{bmatrix} = \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix}, \tag{3.5}$$

we have

$$\det \begin{bmatrix} H_1 & U_1 \\ U_1^T & H_2 \end{bmatrix} = \det \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix}, \tag{3.6}$$

so $\det Q_2 \neq 0$, that is, Q_2 is nonsingular. Similarly, we can prove for all $i = 2,\ldots,J, Q_i = H_i - U_{i-1}^T Q_{i-1}^{-1} U_{i-1}$ is nonsingular.

(3.4) can be derived directly by matrix multiplication. □

PROPOSITION 3.2. *Suppose that* N *is selected such that* (B^T, N^T) *has full column rank. Then the coefficient matrix of the linear equation* (2.27) *is a positive deftnite tri-block-diagonal matrix.*

Proof. We consider two cases.

(1) N has the form of

$$\begin{bmatrix} N_1 & -N_1 & & & & \\ & N_2 & -N_2 & & & \\ & & \cdots & \cdots & & \\ & & & & N_{S-1} & -N_{S-1} \end{bmatrix} \tag{3.7}$$

with all $N_i (i=1,\ldots,S-1)$ having n columns. Let

$$P=V^{-1}Z-V^{-1}ZB^T(BV^{-1}ZB^T)^{-1}BV^{-1}Z$$

Correspondingly, by (2.11),

$$P=\mathrm{diag}(P_1,\ldots,P_S), \tag{3.8}$$

where $P_s=V^{(s)^{-1}}\ Z^{(s)}-V^{(s)^{-1}}\ Z^{(s)}\ B_s^T(B_sV^{(s)^{-1}}\ Z^{(s)}B_s^T)^{-1}B_sV^{(s)^{-1}}Z^{(s)}\in\Re^{n\times n}$ for $s=1,\ldots,S$ are symmetric matrices. By doing matrix multiplications, we have

$$NPN^T=$$
$$\begin{bmatrix} N_1(P_1+P_2)N_1^T & -N_1P_2N_2^T & & \\ -N_2P_2N_1^T & N_2(P_2+P_3)N_2^T & -N_2P_3N_3^T & \\ & \cdots & \cdots & \cdots \\ & & -N_{S-1}P_{S-1}N_{S-2}^T & N_{S-1}(P_{S-1}+P_S)N_{S-1}^T \end{bmatrix} \tag{3.9}$$

which is a tri-block-diagonal matrix of the form G in Proposition 3.1. The positive definiteness is guaranteed by Proposition 2.2.

(2) There are some rows missing in (3.7). In this case, the original problem is decoupled into several smaller problems, which results in that NPN^T consists of several independent blocks, with each block being block tri-diagonal. For instance, if only the i-th row is missing, then N is of the form

$$\begin{bmatrix} \begin{pmatrix} N_1 & -N_1 & & \\ & \cdots & \cdots & \\ & & N_{i-1} & -N_{i-1} \end{pmatrix} & \\ & \begin{pmatrix} N_{i+1} & -N_{i+1} & & \\ & \cdots & \cdots & \\ & & N_{S-1} & -N_{S-1} \end{pmatrix} \end{bmatrix}. \tag{3.10}$$

In this case, NPN^T will consists of two big diagonal blocks, say $\mathrm{diag}(N_1\bar{P}_1N_1^T, N_2\bar{P}_2N_2^T)$ with $\bar{P}_1=\mathrm{diag}(P_1,\ldots,P_i)$ and $\bar{P}_2=\mathrm{diag}(P_{i+1},\ldots,P_S)$ correspondingly. Then each big block can be computed by (1), respectively. The problem with multi-missing rows is similar. □

Applying this process to Example 2.1, since N has three rows with $N_i=[1,0,0,0], i=1,2,3$, each block in matrix (3.9) is a single number. Thus NPN^T is a 3×3 tri-diagonal symmetric positive semidefinite matrix, and (3.4) is the so-called LDL-decomposition of $G=NPN^T$.

3.3. OUR ALGORITHM

For problem (1.6)–(1.8), the potential function (2.6) can be written as

$$\phi(z,v)=(nS+\nu)\ln(z^T v)-e^T\ln(Zv), \tag{3.11}$$

where $e\in\Re^{nS}$ is a vector with all entries being one.

Now we can state our algorithm for the deterministic equivalent (1.6)–(1.8) of multistage stochastic linear programs.

ALGORITHM 3.1. (The decomposition algorithm for multistage stochastic linear program)

Step 1. Choose initial constants $\delta_0\in(0,1], \delta_1\in(0,1), \epsilon>0$ and $\sigma>0$ (which may depend on nS), the stopping tolerance ϵ. Set the initial infeasible point $(z_0,u_0,v_0,w_0)=\delta_0\epsilon_0(e,0,e,0)$. Let $k=0$;

Step 2. Check the stopping criteria (3.14)–(3.16). If they hold, stop;

Step 3. Let $\mu_k=1/(nS+\nu)\, z_k^T v_k$. Solve (3.1) to generate auxiliary directions $(\tilde d_{zk}, \tilde d_{uk}, \tilde d_{vk})$ in parallel;

Step 4. Solve the linear equation (2.27) to derive $\hat w_k$, let $d_{w_k}=\hat w_k-w_k$;

Step 5. Solve the system of linear equations (3.2) to generate the search direction (d_{zk},d_{uk},d_{vk});

Step 6. Select the least positive integer γ such that

$$\phi(z_k+\delta_1^\gamma d_{zk}, v_k+\delta_1^\gamma d_{vk})-\phi(z_k,v_k)\leqslant-\sigma, \tag{3.12}$$

$$(z_k+\delta_1^\gamma d_{zk})^T(v_k+\delta_1^\gamma d_{vk})-(1-\delta_1^\gamma)z_k^T v_k\geqslant 0. \tag{3.13}$$

If we can not find such a γ, then stop;

Step 7. Let $\alpha_k=\delta_1^\gamma$ and $(z_{k+1},u_{k+1},v_{k+1})=(z_k,u_k,v_k)+\alpha_k(d_{zk},d_{uk},d_{vk})$. Update w_k by $w_{k+1}=(1-\alpha_k)w_k+\alpha_k\hat w_k$. Let $k=k+1$ and go to Step 2.

We use the following stopping criteria:

$$\frac{|c^T z-b^T u|}{1+|b^T u|}<\epsilon_1, \tag{3.14}$$

$$\frac{\|(Bz-b,Nz)\|}{1+\|z\|}<\epsilon_2, \tag{3.15}$$

$$\frac{\|B^T u+v+N^T w-c\|}{1+\|v\|}<\epsilon_3 \tag{3.16}$$

where $\|\cdot\|$ is the ℓ_2 norm, scalars $\epsilon_1,\epsilon_2,\epsilon_3$ are prescribed tolerances. It is easy to note that our stopping criteria are identical to that used in [16].

Table 1. Three-stage random test problems

Probs	S	NC	Size(row × column)
r3stage1	4	28 (12)	68×80
r3stage2	9	80 (34)	170×180
r3stage3	16	156 (66)	316×320
r3stage4	25	256 (108)	506×500
r3stage5	36	380 (160)	740×720
r3stage6	49	528 (222)	1018×980
r3stage7	64	700 (294)	1340×1280

Table 2. Numerical results by Algorithm 3.3

Probs	μ	Iter	RPC	RDC	RNC	CPU
r3stage1	9.0409e-06	17	2.6664e-10	3.0089e-15	6.2035e-08	1.420
r3stage2	2.5396e-06	20	1.3819e-09	7.2871e-15	2.5585e-07	4.010
r3stage3	3.0877e-06	24	1.4812e-09	1.8986e-14	4.5757e-07	8.840
r3stage4	9.4097e-07	25	4.1383e-09	1.0429e-14	1.8498e-06	15.000
r3stage5	3.0015e-07	37	9.5042e-09	1.0137e-14	4.9727e-06	33.170
r3stage6	8.6278e-07	29	4.2417e-09	1.8045e-14	6.1104e-06	36.750
r3stage7	4.9829e-07	33	5.3222e-09	1.2502e-14	3.4834e-06	57.620

4. Numerical results

Algorithm 3.3 is applied to solving a set of randomly generated feasible test problems in this section. The matrices B_s are generated randomly and have the same structure as (1.5), all entries of B_s are located in $(-0.5, 0.5)$. Correspondingly b_s are selected randomly such that the vector with all entries being one is feasible to the problem. Simply, we let $c_s = 1/(nS)(1, \dots, 1)^T$ for all s, thus the problem is bounded, i.e., there is no sequence $\{z_k\}$ such that z_k is feasible for all k but $c^T z_k \to -\infty$.

We programmed Algorithm 3.3 in MATLAB code and run under version 5.3. The initial parameters are selected as $\delta_0 = 1, \delta_1 = 0.8$ and $\epsilon_0 = 5$, which imply that the starting points for all test problems are infeasible. We choose $\sigma = 10^{-3}/(nS)$. It is noted that if the prescribed maximal iteration number is not surpassed, the selection of σ will not change the behavior of the algorithm. Furthermore, although theoretical results show that convergence of the algorithm in $O(\sqrt{n}L)$ iterations can be guaranteed for (2.1)–(2.2) if ν is chosen around $O(\sqrt{n})$, practical experiences indicate that much faster convergences are observed when ν are set around

Table 3. Random test problems with different stages and scenarios

Probs	m	n	T	S	NC	Size(row×column)
rand1	5	15	2	16	150 (105)	230×240
rand2	5	15	2	64	630 (378)	950×960
rand3	5	15	2	121	600 (360)	1205×1815
rand4	6	15	3	16	171 (99)	267×240
rand5	8	15	3	64	525 (231)	1037×960
rand6	9	14	3	121	1030 (350)	2119×1694
rand7	7	15	4	64	783 (423)	1231×960
rand8	10	18	4	216	2595 (1000)	4755×3888
rand9	10	20	5	256	4092 (1896)	6652×5120

$O(n^{1.5})$ and $O(n^2)$, see [26]. In our implementation, we let $\nu = 1/4\ nS\sqrt{nS}$. We select the tolerances $\epsilon_1 = \epsilon_2 = \epsilon_3 = 10^{-4}$.

Firstly we solve a set of three-stage stochastic linear programs by our code. There are same numbers of variables and constraints respectively in each stage for these problems, i.e., $n_0 = 4$, $n_1 = n_2 = 8$, $m_1 = 2$, $m_2 = 5$, $m_3 = 3$. The details on these test problems are listed in Table 1, where NC is the number of nonanticipativity constraints before and after preprocessing, the last column is the size of the reformulation problem (1.6)–(1.8) before preprocessing.

The numerical results are presented in Table 2, where μ is the average error of the complement constraints, Iter represents the number of iterations, RPC and RDC represent the ℓ_2 norms of residues of primal constraints (the first row of (2.23)) and its dual constraints (the second row of (2.23)) respectively. For convenience of observing the preprocessing, we calculate RNC as the ℓ_2 norm of constraints (1.10). CPU represents the Central Processing Unit time (in seconds) for running our MATLAB code in solving the problem. Since we solve (2.26) and (2.28) based on decomposition and in series by using a *for* cycle in our code, which results in a large fraction of CPU time, we believe that CPU time listed in the table can be decreased greatly by a parallel implementation.

The results in Table 2 show that all of the random test problems in Table 1 have been solved by Algorithm 3.3, the approximate primal-dual solutions are derived. One of very promising properties in applying interior point methods to multistage stochastic programs is that the number of iterations is typically very low and insensitive to the number of scenarios.

We also solve a set of random test problems with different stages and scenarios, the details on these test problems are listed in Table 3. The numerical results derived by Algorithm 3.3 are presented in Table 4.

In summary, our study shows that, by combining the idea of scenario analysis and the infeasible interior points and by using decomposition techniques in solv-

Table 4. Numerical results by Algorithm 3.3

Probs	μ	Iter	RPC	RDC	RNC	CPU
rand1	1.6567e-06	30	1.4052e-10	3.8721e-16	4.5849e-10	8.022
rand2	1.0640e-07	37	8.8344e-10	2.9383e-16	6.6548e-09	49.040
rand3	3.4071e-07	33	1.0692e-10	7.9457e-17	3.4897e-10	90.651
rand4	4.3286e-07	24	5.5526e-10	2.7498e-15	4.7873e-08	6.249
rand5	2.4215e-07	39	5.7452e-10	4.5740e-16	9.0860e-09	50.773
rand6	2.2568e-07	24	9.7530e-10	7.1767e-16	7.5748e-08	66.776
rand7	5.3785e-07	41	1.4356e-09	1.8850e-14	8.1710e-08	56.391
rand8	3.0685e-07	43	9.7072e-09	1.5395e-14	2.7300e-05	440.163
rand9	2.4809e-07	47	3.9130e-08	3.7443e-14	1.9365e-04	850.543

ing the Newton equations, it is possible to greatly improve the computability of multistage stochastic linear programs.

Acknowledgements

Research is partially supported by Singapore-MIT Alliance and Grant R-314-000-026/028/042-112 of National University of Singapore.

References

Ariyawansa, K.A. and Hudson, D.D. (1991), Performance of a benchmark parallel implementation of the Van Slyke and Wets algorithm for two-stage stochastic programs on the Sequent/Balance, *Concurrency Practice and Experience*, 3, 109–128.

Bahn, O., du Merle, O., Goffin, J.-L. and Vial, J.P., (1995), A cutting plane method from analytic centers for stochastic programming, *Math. Program.*, 69, 45–73.

Berkelaar, A., Dert, C., Oldenkamp, B. and Zhang, S. (2002), A primal-dual decomposition-based interior point approach to two-stage stochastic linear programming, *Oper. Res.,* 50, 904–915.

Berkelaar, A., Kouwenberg, R. and Zhang, S. (2000), A primal-dual decomposition algorithm for multistage stochastic convex programming, Tech. Report SEEM2000-07, The Chinese University of Hong Kong.

Birge, J.R. (1995), Current trends in stochastic programming computation and applications, Tech. Report 95–15, Department of Industrial and Operations Engineering, University of Michigan.

Birge, J.R. and Holmes, D.F. (1992), Efficient solutions of two-stage stochastic linear programs using interior point methods, *Computational Optim. Appl.*, 1, 245–276.

Birge, J.R. and Louveaux, F. (1997), Introduction to Stochastic Programming, Springer, Berlin.

Birge, J.R. and Qi, L. (1988), Computing block-angular Karmarkar projections with applications to stochastic programming, *Management Science*, 34, 1472–1479.

Carpenter, T., Lustig, I. and Mulvey, J. (1991), Formolating stochastic programs for interior point methods, *Oper. Res.*, 39, 757–770.

Chun, B.J. (1992), Scenario analysis modeling and decomposition methods, Tech. Report 92-8, Department of Industrial Engineering, University of Wisconsin-Madison.

Chun, B.J. and Robinson, S.M. (1995), Scenario analysis via bandle decomposition, *Ann. Oper. Res.*, 56, 39–63.

Helgason, T. and Wallace, S.W. (1991), Approximate scenario solations in the progressive hedging algorithm, *Ann. Oper. Res.*, 31, 425–444.

Higle, J.L. and Sen, S. (1996), Stochastic Decomposition: a statistical method for large-scale stochastic linear programming, Kluwer Academic Publishers, Dordrecht, The Netherlands.

Kojima, M., Megiddo, N. and Mizuno, S. (1993), A primal-dual infeasible-interior-point algorithrn for linear programming, *Math. Program.*, 61, 263–280.

Lustig, L.J. (1990), Feasibility issues in a primal-dual interior point method for linear programming, *Math. Program.*, 49, 145–162.

Mehrotra, S. (1992), On the implementation of a primal-dual interior point method, *SIAM J. Optim.*, 2, 575–601.

Mizuno, S. (1994), Polynomiality of infeasible interior point algorithms for linear programming, *Math. Program.*, 67 109–119.

Mizuno, S., Kojima, M. and M.J. Todd, (1995), Infeasible-interior-point primal-dual potential reduction algorithms for linear programming, *SIAM J. Optim.*, 5, 52–67.

Mulvey, J.M. and Vladimirous, H. (1991), Applying the progressive hedging algorithm to stochastic generalized networks, *Ann. Oper. Res.*, 31, 399–424.

Rockafellar, R.T. and Wets, R.J-B. (1991), Scenarios and policy aggregation in optimization under uncertainty, *Math. Oper. Res.*, 16, 119–147.

Ruszczyński, A. (1993), Parallel decomposition of multistage stochastic programming problems, *Math. Program.* 58 201-228.

Ruszczyński, A. (1997), Decomposition methods in stochastic programming, Math. Program., 79, 333–353.

Ruszczyński, A. (1999), Some advances in decomposition methods for stochastic linear programming, Stochastic programming, State of the art, 1998 (Vancouver, BC). *Ann. Oper. Res.*, 85, 153–172.

Sun, J., Wee, K.E. and Zhu, J.S. (1995), An interior point method for solving a class of linearquadratic stochastic programming problems, in: Qi, L. and Womersley, R. (eds.), Recent Advances in Nonsmooth Optimization, World Sci. Publishing, River Edge, NJ, 392–404.

Ye, Y., Todd, M.J. and Mizuno, S. (1994), An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm, *Math. Oper. Res.*, 19, 53–67.

Ye, Y., Kortanek, K.O., Kaliski, J. and Huang, S. (1993), Near boundary behavior of primal-dual potential reduction algorithms for linear programming, *Math. Program.*, 58, 243–255.

Zhang, S. (in press), A primal-dual interior point and decomposition approach to multi-stage stochastic programming, in: Yao, D.D., Zhang, H. and Zhou, H.Y. (eds.), Stochastic Models and Optimization, Springer, Berlin.

Zhang, Y. (1994), On the convergence of a class of infeasible interior-point methods for the horizontal linear complementarity problem, *SIAM J. Optim.*, 4, 208–227.

Zhao, G.Y. A Lagrangian dual method with self-concordant barrier for multi-stage stochastic convex programming, Working Paper, Dept of Mathematics, National University of Singapore, 1998, revised 2000.

Zhao, G.Y. (2001), A log-barrier method with Benders decomposition for solving two-stage stochastic linear programs, *Math. Program.*, 90, 507–536.