Scenario formulation of stochastic linear programs and the homogeneous self-dual interior point method *

Jie Sun[†]and Xinwei Liu[‡]

Abstract. We consider a homogeneous self-dual interior point algorithm for solving multistage stochastic linear programs. The algorithm is particularly suitable for the so-called "scenario formulation" of the problem, whose constraint system consists of a large block-diagonal matrix together with a set of sparse nonanticipativity constraints. Due to this structure, the major computational work required by the homogeneous self-dual interior point method can be split into three steps, each of which is highly decomposable. Numerical results on some randomly generated problems and a multistage production planning problem are reported.

Key words: Multistage stochastic linear programs, interior point methods, decomposition

April 5, 2002; Revised May 12, 2004

*Research is partially supported by Grants RP314000-026/042/057-112 of National University of Singapore and the Singapore-MIT Alliance.

†Department of Decision Sciences and the Singapore-MIT Alliance, National University of Singapore Republic of Singapore. Fax: (65) 6779-2621. E-mail: jsun@nus.edu.sg

‡Department of Decision Sciences, National University of Singapore and Department of Applied Mathematics, Hebei University of Technology, Tianjin, China. E-mail: mathlxw@jsmail.hebut.edu.cn

1. Introduction

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

$$\begin{aligned}
\min\ \ & c_0^\top x + E_{\xi_1}(\min\ \ q_1(\xi_1)^\top y_1 + \cdots + E_{\xi_{T-1}}(\min\ \ q_{T-1}(\xi_{T-1})^\top y_{T-1})) && (1.1)\\
\text{s.t.}\ \ & B_0 x = b,\ \ x \geq 0, && (1.2)\\
& B_1(\xi_1)x + W_1(\xi_1)y_1 = h_1(\xi_1),\ \ y_1 \geq 0, && (1.3)\\
& B_t(\xi_t)y_{t-1} + W_t(\xi_t)y_t = h_t(\xi_t), \quad y_t \geq 0,\ \ t = 2, \cdots, T-1, && (1.4)
\end{aligned}$$

where $x \in \Re^{n_0}$ and $y_t \in \Re^{n_t}$, ξ_t is a random vector associated with stage $t+1$. The superscript "$\top$" represents the transpose and the letter "E" denotes the expected value. $B_t(\xi_t)$ and $W_t(\xi_t)$ are random matrices, $q_t(\xi_t)$ and $h_t(\xi_t)$ are random vectors, all of them are decided by the outcome of the random vector $\xi = (\xi_1, \cdots, \xi_{T-1})$. For convenience of computation, we assume that the support of ξ is finite. The sequence of all possible outcomes of ξ can then be depicted in terms of a scenario tree. Figure 1 shows an example of 3-stage scenario tree where we label the j-th event at the t-th stage as E_{tj}.

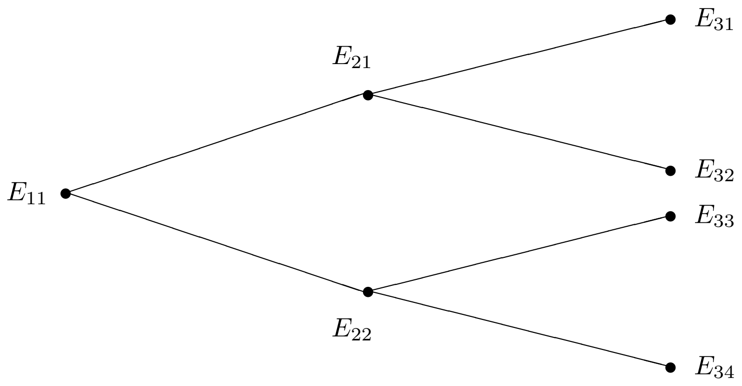

Figure 1. The structure of MSLP

Note that each outcome of the random vector ξ corresponds to a unique path from E_{11} to E_{Tj} on the scenario tree. Let F_t be the number of nodes in stage t. Note that the vector $(y_1, ..., y_{T-1})$ is itself random and let y_{tj} be the outcome of y_j at node (tj). Then a so-called deterministic equivalent for (1.1)-(1.4) can be written as follows:

$$\begin{aligned}
\min\ \ & c_0^\top x + \sum_{t=1}^{T-1}\sum_{j=1}^{F_t} p_{tj} y_{tj} && (1.5)\\
\text{s.t.}\ \ & B_0 x = b,\ \ x \geq 0, && (1.6)\\
& B_{1j}x + W_{1j}y_{1j} = h_{1j},\ \ y_{1j} \geq 0, j = 1, ..., F_1 && (1.7)\\
& B_{tj}y_{(t-1)k} + W_{tj}y_{tj} = h_{tj}, \quad y_{tj} \geq 0,\ \ j = 1, ..., F_t, t = 2, \cdots, T-1, && (1.8)
\end{aligned}$$

where p_{tj}, B_{tj}, W_{tj} and h_{tj} are computable from the given data and node $((t-1)k)$ is the parent of node (tj). Problem (1.5)-(1.8) are known as the *recourse formulation* of problem (1.1)-(1.4), which has been studied extensively in the literature, e.g., see the books of Kall and Wallace [9] and Birge and Louveoux [5] and related references therein. Specifically, the structure of its constraints allows various decomposition schemes to be used within the framework of the simplex method. However, those schemes may not be applicable within the framework of the interior point method because the latter requires different computation (e.g. finding the Newton direction) in each iteration. To fully exploit the power of interior point methods in the context of stochastic programming, some authors such as Berkelaar et al., Sun et al. and Zhao [3, 4, 19, 26] have investigated interior point decomposition methods for (1.5)-(1.8) and have provided computational evidence to show that those methods could be highly efficient.

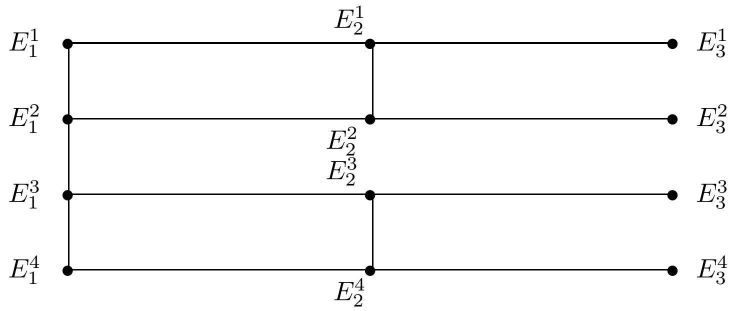

Figure 2. The structure of scenario formulation

In this paper we study an interior point decomposition method for a different formulation of multistage stochastic programs, which we call *the scenario formulation*. The method, known as the homogeneous self-dual interior point method (HSIPM for short), can take good advantage of the structure of the scenario formulation, so as to provide computational efficiency. There are two additional points motivating our choice of HSIPM for study. First, it avoids the process of finding an initial interior feasible solution, which can be quite a burden for stochastic programs. Second, the method can detect the infeasibility of the problem, which is of practical importance since the feedback from the solution stage are often helpful in developing a good model [4].

The scenario formulation was used in the seminal work of Rockafellar and Wets on progressive hedging algorithm [16]. The related important references include [7, 8, 15, 17]. Imagine that the tree in Figure 1 is drawn in a different way like Figure 2. Rather than E_{31}, E_{32}, E_{33}, and E_{34}, the scenarios are simply numbered by superscripts $\{1, 2, 3, 4\}$ and we use subscript to represent the stage number, i.e. E_t^k indicates scenario k at stage t. Each scenario is represented by a horizontal line. Instead of $x, y_{11}, ...,$ the variables are defined along each scenario, say z_1^1, z_2^1, z_3^1, z_1^2, z_2^2, z_3^2, etc., where the superscript is the scenario number and the subscript is the stage number. Obviously, since $z_1^1, z_1^2, z_1^3, z_1^4$ represent the same decision vector at stage 1 (i.e. the vector x in the original formulation), we should have additional constraints

$$z_1^1 = z_1^2 = z_1^3 = z_1^4. \tag{1.9}$$

Similarly, since scenarios 1 and 2 share the same "history" up to stage 2 in Figure 1, an additional constraint

$$z_2^1 = z_2^2 \tag{1.10}$$

must be added, where both z_2^1 and z_2^2 represent the vector y_{11} in the recourse formulation. Each of such constraints is marked by a vertical line connecting the corresponding nodes. These new constraints are called *nonanticipativity* constraints. Note that all nonanticipativity constraints are homogeneous and therefore can be written in a matrix form as $Nz = 0$, where z is an appropriate vector representing the variables z_t^i. More accurately, let s be the total number of scenarios, c_i be the cost vector of scenario i, and $z_i \in \Re^n$ be the vector of variables $(z_1^i, z_2^i, ..., z_T^i)^\top$ in scenario i ($n = \sum_{i=0}^{T-1} n_i$), $i = 1, 2, ..., s$. Let vector z be the collection of all z_i, i.e. $z = (z_1^\top, \cdots, z_s^\top)^\top$. Finally, let $A_i \in \Re^{m \times n}$, $i = 1, 2, ..., s$. Then the scenario formulation of Problem (1.5)-(1.8) will have the following form.

$$\min \quad \sum_{i=1}^{s} c_i^\top z_i \tag{1.11}$$

$$\text{s.t.} \quad A_i z_i = b_i, \;\; z_i \geq 0, \;\; i = 1, \cdots, s \tag{1.12}$$

$$Nz = 0. \tag{1.13}$$

It is obvious that by our labeling the matrix N has the following structure.

$$N = \begin{bmatrix} N_1 & -N_1 & & & & \\ & N_2 & -N_2 & & & \\ & & & \cdots & & \\ & & & & N_{s-1} & -N_{s-1} \end{bmatrix}, \tag{1.14}$$

where all $N_i = [I_i \;\; 0]$ are matrices with n columns, and for any i, I_i is an identity matrix, whose size depends on the numbers of stages and decision variables sharing the same history between the i-th and $(i+1)$-th scenarios. Moreover, we set that

$$A = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \cdots & \\ & & & A_s \end{bmatrix}, \; b = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_s \end{bmatrix}, \; c = \begin{bmatrix} c_1 \\ c_2 \\ \cdots \\ c_s \end{bmatrix}. \tag{1.15}$$

Compared to the recourse formulation, the constraints in (1.12) are decoupled and the coupling constraint $Nz = 0$ is sparse and structured. Since the size of program (1.11)-(1.13) can be huge, it may not be possible to solve (1.11)-(1.13) by using general-purpose linear programming software. Instead, some sort of decomposition scheme must be developed to reduce the amount of computation to a reasonable level. Various decomposition methods can be found, e.g. in [3, 4, 5, 6, 7, 9, 15, 17, 18, 19, 22], some of them can accommodate parallel computation.

The decomposition method that we will discuss is particularly suitable for computing the Newton directions arising from the HSIPM applied to the scenario formulation (1.11)-(1.13). For simplicity of discussion, we may assume that, as explained in a previous paper of the authors [11], the constraint system (1.12)-(1.13) is of full row rank. We have shown [11] how this can be done in an efficient fashion through a pre-processing procedure. For general information on the HSIPM, we refer the reader to [20, 23, 24]. To be brief, we will begin our analysis directly from the special form of the HSIPM applied to problem (1.11)-(1.13).

The work is partially motivated by our former work [11] on an infeasible interior point method for stochastic programming. Compared to [11], the HSIPM has introduced some additional variables, which makes the decomposition of the direction-finding equations more complicated than that in [11]. We present a 3-step decomposition approach for the direction-finding equations, which appears to be new. All of these efforts seem to be paid off by the good performance of the HSIPM in our computational test and hence we believe that HSIPM is a good tool for solving multistage stochastic programs.

The organization of this paper is as follows. We discuss the 3-step HSIPM decomposition approach to problem (1.11)-(1.13) in Section 2. We provide an iterative method and a block-decomposition method for a core equation in HSIPM in Section 3. We summarize the proposed algorithm in Section 4 and report our preliminary numerical results in Section 5.

2. A 3-step decomposition approach

The HSIPM has polynomial complexity and is one of the most efficient interior point methods [1, 2]. It consists of a direction-finding step and a line search step where the direction-finding step consumes most of the computation time. We now discuss a 3-step decomposition approach which can reduce the computation in the direction-finding step.

At each iteration of the HSIPM we have $(z, u, w, v, \tau, \kappa)$ with $z > 0$, $v > 0$, $\tau > 0$ and $\kappa > 0$. The HSIPM, as described in [21], generates the new search direction $(d_z, d_u, d_w, d_v, d_\tau, d_\kappa)$ through solving the following system of equations

$$\begin{bmatrix} A & 0 & 0 & 0 & -b & 0 \\ N & 0 & 0 & 0 & 0 & 0 \\ 0 & A^\top & N^\top & I & -c & 0 \\ V & 0 & 0 & Z & 0 & 0 \\ -c^\top & b^\top & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & \kappa & \tau \end{bmatrix} \begin{bmatrix} d_z \\ d_u \\ d_w \\ d_v \\ d_\tau \\ d_\kappa \end{bmatrix} = - \begin{bmatrix} \eta r_{p1} \\ \eta r_{p2} \\ \eta r_d \\ Zv - \gamma\mu e \\ \eta r_g \\ \kappa\tau - \gamma\mu \end{bmatrix}, \tag{2.1}$$

where $u,\ w,\ v$ are the multipliers corresponding to the equality constraints (1.12) and (1.13) and the nonnegative constraints respectively, $V = \operatorname{diag}(v)$, $Z = \operatorname{diag}(z)$, $r_{p1} = Az - b\tau$, $r_{p2} = Nz$, $r_d = A^\top u + v - c\tau + N^\top w$, $r_g = -c^\top z + b^\top u - \kappa$.

We write (2.1) as two systems

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^\top & I \\ V & 0 & Z \end{bmatrix} \begin{bmatrix} d_z \\ d_u \\ d_v \end{bmatrix} + \begin{bmatrix} -b & 0 & 0 \\ -c & 0 & N^\top \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_\tau \\ d_\kappa \\ \tilde{w} \end{bmatrix} = - \begin{bmatrix} \eta r_{p1} \\ \eta r_{d0} \\ Zv - \gamma\mu e \end{bmatrix} \tag{2.2}$$

and

$$\begin{bmatrix} -c^\top & b^\top & 0 \\ 0 & 0 & 0 \\ N & 0 & 0 \end{bmatrix} \begin{bmatrix} d_z \\ d_u \\ d_v \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ \kappa & \tau & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_\tau \\ d_\kappa \\ \tilde{w} \end{bmatrix} = - \begin{bmatrix} \eta r_g \\ \kappa\tau - \gamma\mu \\ \eta r_{p2} \end{bmatrix}, \tag{2.3}$$

where $\tilde{w} = \eta w + d_w$ and $r_{d0} = A^\top u + v - c\tau$. Let $\tilde{d}_z, \tilde{d}_u, \tilde{d}_v$ be such that

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^\top & I \\ V & 0 & Z \end{bmatrix} \begin{bmatrix} \tilde{d}_z \\ \tilde{d}_u \\ \tilde{d}_v \end{bmatrix} = - \begin{bmatrix} \eta r_{p1} \\ \eta r_{d0} \\ Zv - \gamma\mu e \end{bmatrix}. \tag{2.4}$$

Such $\tilde{d}_z, \tilde{d}_u, \tilde{d}_v$ exist since the coefficient matrix of (2.4) is nonsingular. Then (2.2) can be written as

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^\top & I \\ V & 0 & Z \end{bmatrix} \begin{bmatrix} d_z \\ d_u \\ d_v \end{bmatrix} + \begin{bmatrix} -b & 0 & 0 \\ -c & 0 & N^\top \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_\tau \\ d_\kappa \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} A & 0 & 0 \\ 0 & A^\top & I \\ V & 0 & Z \end{bmatrix} \begin{bmatrix} \tilde{d}_z \\ \tilde{d}_u \\ \tilde{d}_v \end{bmatrix}. \tag{2.5}$$

Eliminating d_z, d_u, d_v in (2.5) and (2.3), we have

$$G \begin{bmatrix} d_\tau \\ d_\kappa \\ \tilde{w} \end{bmatrix} = - \begin{bmatrix} \tilde{r}_g(\tilde{d}_z, \tilde{d}_u) \\ \kappa\tau - \gamma\mu \\ \tilde{r}_{p2}(\tilde{d}_z) \end{bmatrix}, \tag{2.6}$$

and the solution of (2.6) is substituted into the following equation to obtain d_z, d_u, d_v

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^\top & I \\ V & 0 & Z \end{bmatrix} \begin{bmatrix} d_z \\ d_u \\ d_v \end{bmatrix} = - \begin{bmatrix} \tilde{r}_{p1}(d_\tau) \\ \tilde{r}_d(d_\tau, \tilde{w}) \\ Zv - \gamma\mu e \end{bmatrix}, \tag{2.7}$$

where

$$\begin{aligned} \tilde{r}_g(\tilde{d}_z, \tilde{d}_u) &= \quad -c^\top(\eta z + \tilde{d}_z) + b^\top(\eta u + \tilde{d}_u) - \eta\kappa, \\ \tilde{r}_{p2}(\tilde{d}_z) &= \quad N(\eta z + \tilde{d}_z), \\ \tilde{r}_{p1}(d_\tau) &= \quad \eta Az - b(\eta\tau + d_\tau), \\ \tilde{r}_d(d_\tau, \tilde{w}) &= \quad \eta(A^\top u + v) - c(\eta\tau + d_\tau) + N^\top \tilde{w} \end{aligned}$$

and

$$G = \begin{bmatrix} \theta & -1 & -b'^\top - c'^\top \\ \kappa & \tau & 0 \\ b' - c' & 0 & NMN^\top \end{bmatrix}, \tag{2.8}$$

with

$$\begin{aligned}
M &= V^{-1}Z - V^{-1}ZA^\top(AV^{-1}ZA^\top)^{-1}AV^{-1}Z, \\
\theta &= b^\top(AV^{-1}ZA^\top)^{-1}b + c^\top Mc, \\
b' &= NV^{-1}ZA^\top(AV^{-1}ZA^\top)^{-1}b, \\
c' &= NMc.
\end{aligned}$$

The key idea in the 3-step decomposition is to note that solving (2.1) is equivalent to solving consecutively the three systems of equations (2.4), (2.6) and (2.7), each of which is in turn highly decomposable. We elaborate this point in more detail.

First, note that (2.4) can be separated into s independent systems of equations:

$$\begin{bmatrix} A_i & 0 & 0 \\ 0 & A_i^\top & I \\ V_i & 0 & Z_i \end{bmatrix} \begin{bmatrix} \tilde{d}_{z_i} \\ \tilde{d}_{u_i} \\ \tilde{d}_{v_i} \end{bmatrix} = - \begin{bmatrix} \eta r_{p1_i} \\ \eta r_{d0_i} \\ Z_i v_i - \gamma\mu e \end{bmatrix}, \tag{2.9}$$

where $i = 1, \cdots, s$ and $r_{p1_i} = A_i z_i - b_i\tau$, $r_{d0_i} = A_i^\top u_i + v_i - c_i\tau$.

Second and similarly, if we partitioned N into $(\tilde{N}_1, \cdots, \tilde{N}_s)$ with $\tilde{N}_i$ having n columns, and let $\tilde{r}_{p1_i}(d_\tau) = \eta A_i z_i - b_i(\eta\tau + d_\tau)$, $\tilde{r}_{d_i}(d_\tau, \tilde{w}) = \eta(A_i^\top u_i + v_i) - c_i(\eta\tau + d_\tau) + \tilde{N}_i^\top \tilde{w}$, then (2.7) can be decomposed into the following s systems of equations:

$$\begin{bmatrix} A_i & 0 & 0 \\ 0 & A_i^\top & I \\ V_i & 0 & Z_i \end{bmatrix} \begin{bmatrix} d_{z_i} \\ d_{u_i} \\ d_{v_i} \end{bmatrix} = - \begin{bmatrix} \tilde{r}_{p1_i}(d_\tau) \\ \tilde{r}_{d_i}(d_\tau, \tilde{w}) \\ Z_i v_i - \gamma\mu e \end{bmatrix}. \tag{2.10}$$

The coefficient matrices in (2.9) and (2.10) are of the same structure as that in the primal-dual methods for standard linear programming (see [20, 23]). Thus, they can be solved by utilizing the fully developed techniques for standard linear programming. Another obvious advantage for (2.9) and (2.10) is the suitability for parallel solution.

Finally, consider the solution of (2.6). Since (2.6) can be solved by

$$\begin{bmatrix} \kappa/\tau + \theta & -b'^\top - c'^\top \\ b' - c' & NMN^\top \end{bmatrix} \begin{bmatrix} d_\tau \\ \tilde{w} \end{bmatrix} = - \begin{bmatrix} \tilde{r}_g(\tilde{d}_z, \tilde{d}_u) + (\kappa\tau - \gamma\mu)/\tau \\ \tilde{r}_{p2}(\tilde{d}_z) \end{bmatrix}, \tag{2.11}$$

and

$$d_\kappa = (\gamma\mu - \kappa\tau)/\tau - (\kappa/\tau)d_\tau, \tag{2.12}$$

it is left to solve (2.11) by decomposition. For any point $(z, u, v, w, \tau, \kappa)$ with $z > 0$, $v > 0$, $\tau > 0$ and $\kappa > 0$, define $\theta' = \kappa/\tau + \theta$. Then $\theta' > 0$ and we have

$$\begin{aligned}
&\begin{bmatrix} \theta' & -b'^\top - c'^\top \\ b' - c' & NMN^\top \end{bmatrix} = \\
&\begin{bmatrix} 1 & \\ \frac{1}{\theta'}(b' - c') & I \end{bmatrix} \begin{bmatrix} \theta' & \\ & NMN^\top + \frac{1}{\theta'}(b' - c')(b' + c')^\top \end{bmatrix} \begin{bmatrix} 1 & -\frac{1}{\theta'}(b' + c')^\top \\ & I \end{bmatrix}.
\end{aligned} \tag{2.13}$$

For simplicity of statement, we define $\zeta = \tilde{r}_g(\tilde{d}_z, \tilde{d}_u) + (\kappa\tau - \gamma\mu)/\tau$ and $\tilde{r}_p = \tilde{r}_{p2}(\tilde{d}_z)$. Thus, (2.11) can be solved by three steps:

$$\begin{bmatrix} 1 & \\ \frac{1}{\theta'}(b' - c') & I \end{bmatrix} \begin{bmatrix} \nu_1 \\ q \end{bmatrix} = - \begin{bmatrix} \zeta \\ \tilde{r}_p \end{bmatrix}, \tag{2.14}$$

$$\begin{bmatrix} \theta' & \\ & NMN^\top + \frac{1}{\theta'}(b' - c')(b' + c')^\top \end{bmatrix} \begin{bmatrix} \nu_2 \\ p \end{bmatrix} = \begin{bmatrix} \nu_1 \\ q \end{bmatrix}, \quad \text{and} \tag{2.15}$$

$$\begin{bmatrix} 1 & -\frac{1}{\theta'}(b' + c')^\top \\ & I \end{bmatrix} \begin{bmatrix} d_\tau \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} \nu_2 \\ p \end{bmatrix}. \tag{2.16}$$

Obviously, equations (2.14) and (2.16) can be solved very easily. In order to solve the equation (2.15), we need only to consider the solution of the equation

$$(NMN^\top + b''c''^\top)p = q, \tag{2.17}$$

where $b'' = \frac{1}{\sqrt{\theta'}}(b' - c')$, $c'' = \frac{1}{\sqrt{\theta'}}(b' + c')$, p is the unknown vector and $q = -\tilde{r}_p + \frac{\zeta}{\sqrt{\theta'}}b''$ by (2.14).

Direct use of the inverse of matrix $NMN^\top + b''c''^\top$ may destroy the sparsity and special structure of $NMN^\top$. We need the following result to avoid this.

Proposition 2.1

$$1 + c''^\top (NMN^\top)^{-1} b'' \neq 0. \tag{2.18}$$

Proof. By expressions of b'' and c'', we have

$$1 + c''^\top (NMN^\top)^{-1} b'' = 1 + \frac{1}{\theta'}\left(b'^\top (NMN^\top)^{-1} b' - c'^\top (NMN^\top)^{-1} c'\right). \tag{2.19}$$

Since $NMN^\top$ is positive definite by Proposition 3.1 in the next section, and $\theta' > 0$, we need only to prove

$$1 - \frac{1}{\theta'} c'^\top (NMN^\top)^{-1} c' \geq 0. \tag{2.20}$$

Inequality (2.20) follows from the fact that

$$\begin{aligned} c^\top Mc - c'^\top (NMN^\top)^{-1} c' &= c^\top Mc - c^\top MN^\top (NMN^\top)^{-1} NMc \\ &= c^\top M^{\frac{1}{2}}(I - M^{\frac{1}{2}} N^\top (NMN^\top)^{-1} NM^{\frac{1}{2}})M^{\frac{1}{2}} c \\ &\geq 0. \end{aligned} \tag{2.21}$$

■

It follows from Proposition 2.1 and the Sherman-Morrison formula that we have

$$(NMN^\top + b''c''^\top)^{-1} = (NMN^\top)^{-1} - \frac{(NMN^\top)^{-1} b''c''^\top (NMN^\top)^{-1}}{1 + c''^\top (NMN^\top)^{-1} b''}. \tag{2.22}$$

Thus, we can solve (2.17) by firstly solving

$$(NMN^\top)p = b'', \tag{2.23}$$

and

$$(NMN^\top)p = q \tag{2.24}$$

to generate p_1 and p_2 respectively, and then calculating

$$\tilde{p} = p_2 - \frac{c''^\top p_2}{1 + c''^\top p_1} p_1, \tag{2.25}$$

which is the solution of (2.17). Then, by (2.14)-(2.16), we have

$$\tilde{w} = \tilde{p}, \tag{2.26}$$

$$d_\tau = -\frac{\zeta}{\theta'} + \frac{1}{\sqrt{\theta'}} c''^\top \tilde{p}. \tag{2.27}$$

We call equations (2.23) and (2.24) the "core" equations and their solution is also decomposable. We leave the detailed discussion to the next section.

In summary, we have a 3-step decomposition method for solving (2.1) as follows. We solve (2.4),(2.6), and (2.7), consecutively. Within each step the computation is decomposable. The first and the third steps reduce to solving s small systems (2.9) and (2.10), respectively, while the computation of the second step is the topic of the next section.

3. Block decomposition of the core equation

The core step in the HSIPM is to solve an equation system of the form

$$(NMN^\top)p = q, \tag{3.1}$$

where

$$M = D - DA^\top(ADA^\top)^{-1}AD \tag{3.2}$$

and D is a positive definite diagonal matrix, A and N are defined in (1.15) and (1.14), respectively. Thus, M is an s-block diagonal matrix. We first show that (3.1) is solvable.

Proposition 3.1 *If $(A^\top, N^\top)$ is of full column rank, then $NMN^\top$ is positive definite.*

Proof. Let $ND^{1/2} = U$ and $AD^{1/2} = V$. Then $NMN^\top = U(I - V^\top(VV^\top)^{-1}V)U^\top$, which is positive semidefinite since $I - V^\top(VV^\top)^{-1}V$ is a projection matrix. If $(A^\top, N^\top)$ is of full column rank, then $(V^\top, U^\top)$ is of full column rank. By QR decomposition we have

$$(V^\top, U^\top) = [Q_1, Q_2, Q_3]\begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \\ 0 & 0 \end{bmatrix}, \tag{3.3}$$

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

$$\begin{aligned} NMN^\top &= U(I - V^\top(VV^\top)^{-1}V)U^\top \\ &= UU^\top - UV^\top(VV^\top)^{-1}VU^\top \\ &= R_{12}^\top R_{12} + R_{22}^\top R_{22} - R_{12}^\top R_{11}(R_{11}^\top R_{11})^{-1}R_{11}^\top R_{12} \\ &= R_{22}^\top R_{22}. \end{aligned} \tag{3.4}$$

The positive definiteness of $NMN^\top$ follows from the nonsingularity of R_{22}. ■

Corollary 3.2 *Under the assumption of Proposition 3.1 $\hat{N}M\hat{N}^\top$ is also positive definite, where*

$$\hat{N} = \begin{bmatrix} N_1 & N_1 & & & \\ & N_2 & N_2 & & \\ & & \cdots & & \\ & & & N_{s-1} & N_{s-1} \end{bmatrix}. \tag{3.5}$$

Proof. It is obvious that $(A^\top, N^\top)$ is of full column rank if and only if $(A^\top, \hat{N}^\top)$ is of full column rank. The rest of the proof is same as Proposition 3.1. ■

We now take a closer look at the structure of $NMN^\top$. If we define the i-th block of M to be M_i $(i = 1, \cdots, s)$, then

$$NMN^\top = \begin{bmatrix} N_1(M_1 + M_2)N_1^\top & -N_1M_2N_2^\top & & \\ -N_2M_2N_1^\top & N_2(M_2 + M_3)N_2^\top & & \\ & -N_3M_3N_2^\top & & \\ & & \cdots & \\ & & & N_{s-1}(M_{s-1} + M_s)N_{s-1}^\top \end{bmatrix} \tag{3.6}$$

which is a block tri-diagonal matrix. This structure makes it simple to solve (3.1). We discuss two possible methods. The first is an iterative method that appears to be more suitable for parallel machines. The second is a block-$LDL^\top$ decomposition method that will be used in our computational experiment.

3.1. A Jacobi or Gauss-Seidel iteration method. We split $NMN^\top$ as

$$NMN^\top = P + K, \tag{3.7}$$

where

$$P = \text{diag}\left[N_1(M_1 + M_2)N_1^\top, N_2(M_2 + M_3)N_2^\top, \cdots, N_{s-1}(M_{s-1} + M_s)N_{s-1}^\top\right]$$

and

$$K = \begin{bmatrix} 0 & -N_1M_2N_2^\top & & & \\ -N_2M_2N_1^\top & 0 & & & \\ & -N_3M_3N_2^\top & & & \\ & & \cdots & & \\ & & & 0 & -N_{s-2}M_{s-1}N_{s-1}^\top \\ & & & -N_{s-1}M_{s-1}N_{s-2}^\top & 0 \end{bmatrix},$$

and solve equation $(NMN^\top)p = q$ iteratively by

$$Pp_+ = q - Kp, \tag{3.8}$$

where p_+ is the new approximation to the solution and p is the current iterate. Since both $P + K = NMN^\top$ and $P - K = \hat{N}M\hat{N}^\top$ are positive definite (Corollary 3.2), it is easy to see that iteration (3.8) will be convergent, as proved by the following proposition.

Proposition 3.3 *Let P and K be symmetric matrices. Suppose that both $P - K$ and $P + K$ are positive definite. Then $\rho(P^{-1}K) < 1$, where $\rho(\cdot)$ stands for the spectral radius. Thus, the process of iteration (3.8) is convergent.*

Proof. Since $2P = (P - K) + (P + K)$, P is positive definite. For any $x \in \Re^n$, let $\tilde{x} = P^{-1}Kx$, $u = x - \tilde{x}$. Let $A = P - K$. Then $\tilde{x} = P^{-1}Kx$, which implies that

$$x^\top Ax - \tilde{x}^\top A\tilde{x} = x^\top Pu - \tilde{x}^\top Ku = u^\top(Px - K\tilde{x}) = u^\top(P + K)u. \tag{3.9}$$

Let λ be any eigenvalue of $P^{-1}K$. Then there is $x \neq 0 \in \Re^n$ such that

$$\tilde{x} = P^{-1}Kx = \lambda x.$$

Since $x^\top Ax - \tilde{x}^\top A\tilde{x} = (1 - \lambda^2)x^\top Ax$, and $u^\top(P + K)u = (1 - \lambda)^2x^\top(P + K)x$, we have from (3.9) that

$$(1 - \lambda^2)x^\top Ax = (1 - \lambda)^2x^\top(P + K)x,$$

which implies that $|\lambda| \leq 1$ since $x^\top Ax > 0$ and $x^\top(P + K)x > 0$.

If $\lambda = 1$, then $Kx = Px$, that is $Ax = 0$, which contradicts the positive definiteness of A. Thus, we have $|\rho(P^{-1}K)| < 1$, which implies that $(P^{-1}K)^k \to 0$ as positive integer $k \to \infty$. It follows from (3.8) that, for any positive integer k,

$$p_k - p^* = -(P^{-1}K)(p_{k-1} - p^*) = \cdots = [-(P^{-1}K)]^k(p_0 - p^*), \tag{3.10}$$

where p^* is the solution such that $(P + K)p^* = q$, p_0 is the initial point. Thus, we have $p_k \to p^*$ as $k \to \infty$, which is the desired result. ∎

Since P is a block diagonal matrix, (3.8) can be implemented in parallel.

3.2. A direct block $LDL^\top$-factorization method.

Proposition 3.4 *Consider the tri-block-diagonal matrix*

$$NMN^\top = \begin{bmatrix} H_1 & U_1 & & & \\ U_1^\top & H_2 & U_2 & & \\ & U_2^\top & H_3 & & \\ & & \cdots & \cdots & \cdots \\ & & & U_{J-1}^\top & H_J \end{bmatrix}, \tag{3.11}$$

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

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

If $NMN^\top$ *is positive definite, then all* Q_i *are nonsingular, and* $NMN^\top$ *can be decomposed into*

$$\begin{bmatrix} I_1 & & & \\ U_1^\top Q_1^{-1} & I_2 & & \\ & \cdots & \cdots & \\ & & U_{J-1}^\top Q_{J-1}^{-1} & I_J \end{bmatrix} \begin{bmatrix} Q_1 & & & \\ & Q_2 & & \\ & & \cdots & \\ & & & Q_J \end{bmatrix} \begin{bmatrix} I_1 & Q_1^{-1}U_1 & & \\ & I_2 & Q_2^{-1}U_2 & \\ & & \cdots & \cdots \\ & & & I_J \end{bmatrix}, \tag{3.12}$$

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

Proof. $NMN^\top$ is positive definite if and only if all of its principal submatrices are positive definite. Thus,

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

are positive definite. Hence, Q_1 is nonsingular. Since

$$\begin{bmatrix} I_1 & 0 \\ -U_1^\top Q_1^{-1} & I_2 \end{bmatrix} \begin{bmatrix} H_1 & U_1 \\ U_1^\top & H_2 \end{bmatrix} \begin{bmatrix} I_1 & -Q_1^{-1}U_1 \\ 0 & I_2 \end{bmatrix} = \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix}, \tag{3.13}$$

we have

$$\det \begin{bmatrix} H_1 & U_1 \\ U_1^\top & H_2 \end{bmatrix} = \det \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix}, \tag{3.14}$$

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

(3.12) can be derived directly by matrix multiplication. ∎

Note that, in solving system $NMN^\top p = q$, we need compute the inverse of matrix Q_i whose size is related to the number of stages of the program but not related to the number of scenarios, thus is not very large.

4. A decomposition-based homogeneous and self-dual algorithm

The predictor-corrector technique is proved to be very efficient for primal-dual interior point methods in solving linear programming (see [12, 13, 21, 23]). Based on the predictor-corrector version of the HSIPM and the decomposition techniques discussed in sections 2 and 3, we present a decomposition-based homogeneous self-dual interior point algorithm for problem (1.11)-(1.13).

Algorithm 4.1 *(The algorithm for MSLP)*

Step 1. Generate N *by (1.13) such that* $(A^\top, N^\top)$ *has full column rank;*

Step 2. Given the initial point $(z_0, u_0, v_0, w_0, \tau_0, \kappa_0)$*, and the tolerance scalar* ϵ*. Let* $k := 0$*;*

Step 3. While the stopping criterion is not satisfied, do

Step 3.1 Predictor step: Compute $\mu_k = (z_k^\top v_k + \tau_k \kappa_k)/(ns+1)$*, set* $(z,u,v,w,\tau,\kappa) = (z_k,u_k,v_k,w_k,\tau_k,\kappa_k)$*,* $\eta = 1$*,* $\gamma = 0$*.*
For $i = 1, \cdots, s$*, calculate* r_{p1_i} *and* r_{d0_i}*, solve (2.9) to generate auxiliary direction* $(\tilde{d}_{z_i}, \tilde{d}_{u_i}, \tilde{d}_{v_i})$*;*
Calculate ζ *and* $\tilde{r}_p$*. Derive* $(\hat{d}_\tau, \tilde{w})$ *by (2.27) and (2.26), and then calculate* $\hat{d}_\kappa$ *by (2.12),* $\hat{d}_w = \tilde{w} - w$*;*
Calculate $\tilde{r}_{p1_i}$ *and* $\tilde{r}_{d_i}$*, and then solve (2.10) to generate the search direction* $(\hat{d}_{z_i}$*,* $\hat{d}_{u_i}$*,* $\hat{d}_{v_i})$ *for* $i = 1, \cdots, s$*;*
Compute

$$\hat{\alpha} = -\frac{0.99995}{\min(Z^{-1}\hat{d}_z, V^{-1}\hat{d}_v, \hat{d}_\tau/\tau, \hat{d}_\kappa/\kappa, -0.99995)}. \tag{4.1}$$

Step 3.2 Corrector step: Set

$$(z', u', v', w', \tau', \kappa') = (z,u,v,w,\tau,\kappa) + \hat{\alpha}(\hat{d}_z, \hat{d}_u, \hat{d}_v, \hat{d}_w, \hat{d}_\tau, \hat{d}_\kappa), \tag{4.2}$$

let $\mu' = (z'^\top v' + \tau'\kappa')/(ns+1)$ *and compute*

$$\gamma = \begin{cases} (\mu'/\mu_k)^2, & \text{if } \mu'/\mu_k \le 0.01, \\ \min\{0.1, \max[(\mu'/\mu_k)^3, 0.0001]\}, & \text{otherwise}, \end{cases} \tag{4.3}$$

and $\eta = 1 - \gamma$*.*
Calculate r_{p1_i} *and* r_{d0_i}*, and then solve (2.9) to generate auxiliary direction* $(\tilde{d}_{z_i}$*,* $\tilde{d}_{u_i}$*,* $\tilde{d}_{v_i})$ *for* $i = 1, \cdots, s$*;*
Calculate ζ *and* $\tilde{r}_p$*. Derive* $(d_\tau, \tilde{w})$ *by (2.27) and (2.26), and then calculate* d_κ *by changing (2.12) as*

$$d_\kappa = (\gamma\mu - \kappa'\tau' - \hat{d}_\kappa \hat{d}_\tau)/\tau' - (\kappa'/\tau')d_\tau, \tag{4.4}$$

$d_w = \tilde{w} - \eta w'$*;*
For $i = 1, \cdots, s$*, calculate* $\tilde{r}_{p1}$ *and* $\tilde{r}_d$*, and then solve (2.10) by replacing the right-hand-side term* $(Zv - \gamma\mu e)$ *with* $(Z'v' + diag(\hat{d}_z)\hat{d}_v - \gamma\mu e)$ *to generate the search direction* $(d_{z_i}, d_{u_i}, d_{v_i})$*;*
Compute the step-size α *by (4.1) with* Z*,* V*,* τ*,* κ *and* $\hat{d}_z$*,* $\hat{d}_v$*,* $\hat{d}_\tau$*,* $\hat{d}_\kappa$ *replaced by* Z'*,* V'*,* τ'*,* κ' *and* d_z*,* d_v*,* d_τ*,* d_κ*. Set*

$$\begin{aligned} &(z_{k+1}, u_{k+1}, v_{k+1}, w_{k+1}, \tau_{k+1}, \kappa_{k+1}) \\ &= (z', u', v', w', \tau', \kappa') + \alpha(d_z, d_u, d_v, d_w, d_\tau, d_\kappa). \end{aligned} \tag{4.5}$$

Step 3.3 Let $k := k+1$ *and return to Step 3.*

Step 4. If the stopping criterion on optimality is satisfied, then we have the optimal solution

$$(z^*, u^*, v^*, w^*) = (z_k/\tau, u_k/\tau, v_k/\tau, w_k/\tau); \tag{4.6}$$

Else, MSLP is infeasible or unbounded.

How to select a suitable centering parameter γ is an important issue in the predictor-corrector algorithm. In Algorithm 4.1, we use the formulae in [21].

5. Numerical results

We report our preliminary numerical results in this section. The goal of the computational experiment is twofold. First, we test the efficiency of the proposed algorithm by comparing it with an HSIPM that

Probs	m	n	T	s	NC
rand1	5(3,2)	15(10,5)	2	16(16)	105
rand2	5(4,1)	15(10,5)	2	64(64)	378
rand3	5(2,3)	15(5,10)	2	121(121)	360
rand4	6(4,1,1)	15(9,3,3)	3	16(4,4)	99
rand5	8(2,3,3)	15(3,6,6)	3	64(8,8)	231
rand6	9(2,4,3)	14(4,5,5)	3	121(11,11)	350
rand7	7(4,1,1,1)	15(9,2,2,2)	4	64(4,4,4)	423
rand8	10(1,4,3,2)	18(3,5,5,5)	4	216(6,6,6)	1000
rand9	10(4,2,2,1,1)	20(8,3,3,3,3)	5	256(4,4,4,4)	1896

Table 1. The size of test problems

directly solves equation (2.1) in finding the search direction. Randomly generated problems in [11] are used for this purpose. Second, rather than just use randomly generated problems, we test the algorithm on a real production planning problem to convince ourself that the algorithm can be similarly efficient in practice.

5.1. Implementation and test on randomly generated problems. The sizes of the test problems are listed in Table 1, where m and n are the numbers of rows and columns of A_i , respectively, which are also the total numbers of constraints and variables in each scenario over all stages, exclusive of the nonanticipativity constraints. T is the number of stages and s is the number of scenarios. NC represents the number of nonanticipativity constraints. The data in the parentheses in the m-column show the numbers of constraints in each stage. For example, for problem rand5, we see from the table that this problem is a 3-stage problem, for which there are 2 constraints in the 1st stage, 3 constraints in the 2nd stage, and 3 constraints in the 3rd stage, respectively. Similarly, the data in the parentheses in the n-column show the numbers of variables in each stage, while the data in the parentheses in the s-column show the numbers of scenarios in each stage, exclusive of the 1st stage.

For each test problem, all entries of data are located in $(-0.5, 0.5)$. The right-hand-side term is generated such that e is a primal feasible solution, where e is the all-one vector, whose size is compatible with the problem. For $i = 1, \cdots, s$, $c_i = \frac{1}{ns}e$, the structure of A_i is as follows:

$$A_i = \begin{bmatrix} \times & & & & & \\ \times & \times & & & & \\ & \times & \times & & & \\ & & & \cdots & & \\ & & & & \times & \times \end{bmatrix}, \tag{5.1}$$

where each $\times$ represents a block with suitable sizes.

Algorithm 4.1 is programmed in MATLAB code and is run under MATLAB version 6.0 on a Hewlett Packard C3600 workstation with UNIX system. Equations (2.23) and (2.24) are solved by the direct decomposition technique in section 3.2. The algorithm terminates whenever the conditions:

$$\frac{|c^\top z - b^\top u|}{\tau + |b^\top u|} < \epsilon, \tag{5.2}$$

$$\frac{\|(Az - b\tau, Nz)\|}{\tau + \|z\|} < \epsilon, \tag{5.3}$$

$$\frac{\|A^\top u + v + N^\top w - c\tau\|}{\tau + \|v\|} < \epsilon, \tag{5.4}$$

or

$$\mu/\mu_0 < \epsilon', \quad \tau/\kappa < \sigma\epsilon \tag{5.5}$$

Prob	r_b0	obj0	r_b	r_c	r_g	obj
rand1	3.97e+00	7.2751	5.68e-08	1.49e-07	8.23e-07	7.1016
rand2	8.48e+00	10.9550	8.57e-08	6.42e-08	7.43e-08	6.5249
rand3	1.33e+01	11.5485	7.45e-07	4.48e-08	4.26e-07	7.4671
rand4	2.22e+00	6.9106	2.79e-07	1.67e-06	8.61e-06	3.9761
rand5	8.63e+00	12.1235	2.81e-07	8.59e-07	3.34e-05	9.4387
rand6	1.37e+01	12.5335	3.22e-06	6.71e-09	8.15e-08	11.6186
rand7	4.99e+00	10.0806	8.44e-07	9.38e-07	2.65e-05	6.5624
rand8	2.22e+01	15.9753	9.85e-06	2.18e-06	4.36e-05	13.4244
rand9	2.43e+01	17.8885	9.61e-06	4.22e-06	5.32e-06	14.3460

Table 2. Numerical results for random problems

are satisfied, where $\|\cdot\|$ is the ℓ_2 norm, $\epsilon = 10^{-6}$ and $\epsilon' = 10^{-10}$. Some heuristic and sophisticated technique such as the minimum local fill-in ordering (see [12, 21, 25]) have been introduced to improve the accuracy of the algorithm and the stability of the decomposition.

It is easy to note that our stopping criterion is identical to that used in [13, 21]. If conditions (5.2)-(5.4) are satisfied, then we have the primal-dual optimal solution (z^*, u^*, v^*, w^*). Otherwise, (5.5) is satisfied, problem (1.11)-(1.13) or its dual problem is infeasible or nearly infeasible.

We select the initial point $(z_0, u_0, v_0, w_0) = \sigma(e, 0, e, 0)$, $\tau_0 = 2\sigma$ with $\sigma = 5$, $\kappa_0 = 1$. The numerical results are presented in Tables 2 and 3, where

$$\begin{aligned} r_b &= \|(Az/\tau - b,\ Nz/\tau)\|, \\ r_c &= \|A^\top u/\tau + v/\tau - c + N^\top w/\tau\|, \\ r_g &= z^\top v/\tau^2, \end{aligned}$$

and obj is the optimal value of the objective function. We use r_b0 and obj0 to stand for the values of r_b and the objective function at the initial point, respectively. It can be seen that z_0/τ_0 is infeasible for all test problems. That is, our test always starts from an infeasible solution.

The results in Table 2 show that all of the random test problems in Table 1 have been solved by Algorithm 4.1. It was noticed in [3, 11] that the number of iterations is typically very low and insensitive to the number of scenarios in applying interior point methods to multistage stochastic programs. The results here confirms that this is also true for Algorithm 4.1.

We also implemented the algorithm without decomposition. That is, we solve the direction-finding equation (2.1) directly in the predictor and corrector steps by the technique suggested in [24] (see (16) and (17) in [24]). The number of iterations and the speed-ups of the decomposition approach (approximately 20% in average) are also shown in the Table 3, which shows the savings in time may be increased as the sizes of problems increase.

To have a better understanding on the speed-ups, we have solved a set of three-stage problems originated from problem rand6 in Table 1. We keep numbers T, m and n of stages, constraints and variables for each scenario unchanged, but change the numbers of scenarios for the last two stages. We observed that the computational times for the decomposition approach increases only linearly with the number of scenarios, whereas the computational times for the direct solver increase quadratically with the number of scenarios, showing a potential of bigger savings as the number of scenarios increases, see Figure 1. The similar phenomenon was observed by [3] in its decomposition method for two-stage stochastic linear programs.

In order to further observe the behavior of the algorithm, we also test a set of randomly generated problems, in which the numbers n, m and s are the same, but T (the number of stages) is changed from 2 to 4. The data of the problems and the results are presented in Table 4, where iter represents the number of iterations and the computational time is recorded in the last column. From the table we can see the

Prob	problem size	direct solution	decomposition (approximate speed-up)
rand1	185×240	9	8 (16%)
rand2	698×960	9	9 (16%)
rand3	965×1815	11	10 (23%)
rand4	195×240	8	7 (14%)
rand5	743×960	11	11 (16%)
rand6	1439×1694	8	7 (22%)
rand7	871×960	12	11 (15%)
rand8	3160×3888	26	23 (27%)
rand9	4456×5120	24	21 (25%)

Table 3. Comparison with direct approach

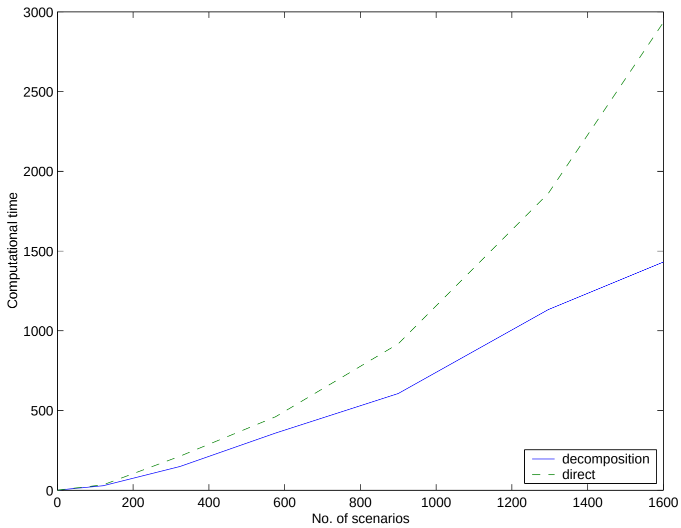

Figure 1: A comparison as number of scenarios increases

m	n	s	T	NC	problem size	iter	time(seconds)
29(20,9)	45(30,15)	64(64)	2	630	[2486 2880]	11	26.37
29(20,6,3)	45(30,10,5)	64(16,4)	3	822	[2678 2880]	12	29.13
29(20,6,3)	45(30,10,5)	64(4,16)	3	870	[2726 2880]	14	35.26
29(20,6,3)	45(30,10,5)	64(8,8)	3	854	[2710 2880]	11	26.47
29(20,3,6)	45(30,5,10)	64(16,4)	3	726	[2582 2880]	15	32.18
29(20,3,6)	45(30,5,10)	64(4,16)	3	750	[2606 2880]	15	32.59
29(20,3,6)	45(30,5,10)	64(8,8)	3	742	[2598 2880]	14	29.34
29(20,3,3,3)	45(30,5,5,5)	64(4,4,4)	4	846	[2702 2880]	19	39.94

Table 4. Results on problems with fixed m, n, s and changed T

number of iterations and the computational time tend to increase as the number of stages increases. The reason might be that the structure of nonanticipativity constraints becomes more complicated when T increases, hence the computations involving $NMN^\top$ are more difficult and less accurate. As a result, both number of iterations and computational time increase.

As a referee of this paper pointed out, the number of nonanticipativity constraints may increase with the size of problem (1.1)-(1.4), which would make the core step more involved and make it difficult to find a solution that satisfies the nonanticipativity constraints. Note that the number of nonanticipativity constraints will not change if only the number of variables in the last stage is increased. The size of $(NMN^\top)$ will keep unchanged in this case. Therefore, this problem may not be serious for two-stage problems.

5.2. A multistage production planning problem. We consider the solution of a multistage stochastic programming problem in the literature [10]. In order to satisfy random demands for its products over several stages, a factory must decide its production scheme, including the increments and/or decrements on its products in production activity with different stages.

We consider a five-stage case of the problem. In the first stage (the dummy stage) of production, the manager should make a production schedule for the late four stages on the amounts of its two products and extra capacity of three production activities because of the change of demands. Thus, we have decision variables x_{jt} and u_{it}, and

$$x_{jt} \geq 0,\ u_{it} \geq 0,\ j = 1, 2;\ i = 1, 2, 3;\ t = 1, \cdots, 4. \tag{5.6}$$

They satisfy the following constraints

$$\sum_{j=1}^{2} a_{ij} x_{jt} - u_{it} \leq b_{it},\ i = 1, 2, 3;\ t = 1, \cdots, 4 \tag{5.7}$$

$$u_{it} \leq f_{it},\ i = 1, 2, 3;\ t = 1, \cdots, 4 \tag{5.8}$$

where b_{it} and f_{it} are the normal capacity and maximum expansion of capacity for production activity i in stage $t+1$, respectively. Let w_t^+ and w_t^- be the changes in utilization of production activity 3 from stage t to stage $t+1$. The constraints on w_t^+ and w_t^- are as follows:

$$w_t^+ \geq 0,\ w_t^- \geq 0,\ t = 1, 2, 3 \tag{5.9}$$

$$\sum_{j=1}^{2} a_{3j} (x_{j(t+1)} - x_{jt}) = w_t^+ - w_t^-,\ t = 1, 2, 3. \tag{5.10}$$

Since both w_t^+ and w_t^- depend only on x_{jt}, which are not random, we view them as the variables in the first stage.

In stage $t+1$, since the demand for products ξ_{jt} is random, which results in that the amounts of purchased deficit products y_{jt}^+ and stored surplus products y_{jt}^- are random, the constraints are

$$y_{jt}^+ \geq 0,\ y_{jt}^- \geq 0,\ j = 1, 2;\ t = 1, \cdots, 4 \tag{5.11}$$

x_{jt}	t=1	t=2	t=3	t=4
j=1	335.9037	284.0982	444.3442	403.8439
j=2	563.1821	585.3843	398.8522	376.9239
u_{jt}				
j=1	159.5272	63.3164	0.0008	0.0011
j=2	0.0006	0.0006	0.0005	0.0006
j=3	449.9874	449.9864	374.9994	349.9996
w^+	0.0004	0.0003	0.0003	
w^-	0.0012	824.9871	274.9997	

Table 5. The 1st-stage solution to the production problem with 256 scenarios

$$x_{j1} + y_{j1}^+ - y_{j1}^- = \xi_{j1},\ j = 1, 2 \tag{5.12}$$

$$x_{jt} + y_{j(t-1)}^- + y_{jt}^+ - y_{jt}^- = \xi_{jt},\ j = 1, 2;\ t = 2, 3, 4. \tag{5.13}$$

Our objective is to minimize the cost in finishing the production schedule, which is

$$\begin{aligned}\sum_{t=1}^{4}\{\sum_{j=1}^{2} c_j x_{jt} + \sum_{i=1}^{3} d_i u_{it}\} + \sum_{t=1}^{3}\{e^+ w_t^+ + e^- w_t^-\}\\ +E\{\min \sum_{t=1}^{4}\sum_{j=1}^{2}[q_{jt}^+ y_{jt}^+ + q_{jt}^- y_{jt}^-]\},\end{aligned} \tag{5.14}$$

where c_j and d_i are the cost of producing product j and the cost of extra capacity of production activity i, e^+, e^- are costs of change of production activity 3, q_{jt}^+ and q_{jt}^- are costs of deficit and surplus product j in stage t, respectively.

All data in (5.6)-(5.14) are given in [10]. The demands $\xi_{jt}(j = 1, 2;\ t = 1, \cdots, 4)$ are independent and normal. In order to solve this problem by the algorithm in this paper, we approximate each random variable by the corresponding discrete random variable. As a result, if each ξ_{jt} has k outcomes, then the problem is a 5-stage problem with k^8 scenarios. We consider $k = 2$. Thus, the problem has 256 scenarios.

Since our algorithm only deal with the problem with linear equality and nonnegative constraints, we introduce nonnegative slack variables z_{it} and v_{it} to (5.7) and (5.8) such that

$$\sum_{j=1}^{2} a_{ij} x_{jt} - u_{it} + z_{it} = b_{it},\ i = 1, 2, 3;\ t = 1, \cdots, 4 \tag{5.15}$$

$$u_{it} + v_{it} = f_{it},\ i = 1, 2, 3;\ t = 1, \cdots, 4. \tag{5.16}$$

Then, for each scenario, the problem has 27 constraints and 50 variables in the first stage, 2 constraints and 4 variables for each other stage (not including all nonnegative constraints).

Our algorithm solves this problem in 34 iterations with computational time 587.37 (seconds). The solution for the variables in the first stage are listed in Table 5.

We also report the solution to this problem in Table 6 with 6561 scenarios, a much larger number of scenarios (in which case $k = 3$). The optimal solution is obtained in 44 iterations with computational time 10658.66 (seconds).

Acknowledgment. We would like to thank Professor Toh of Department of Mathematics of National University of Singapore for providing us with a code of sparse Cholesky factorization. We are also grateful to the area editor and the anonymous referees for their valuable suggestions.

x_{jt}	t=1	t=2	t=3	t=4
j=1	348.9676	271.0601	467.1376	403.8355
j=2	557.5716	590.9584	389.0827	376.9268
u_{jt}				
j=1	183.7440	39.0509	0.0001	0.0001
j=2	0.0001	0.0001	0.0001	0.0006
j=3	449.9221	449.9096	374.9962	349.9970
w^+	0.0002	0.0001	0.0001	
w^-	0.0014	824.9746	274.9979	

Table 6. The 1st-stage solution to the production problem with 6561 scenarios

References

[1] E.D. Anderson and K.D. Anderson, *The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm,* High Performance Optimization Techniques, pp. 197-232, J.B.G. Frenk, K. Roos, T. Terlaky, and S. Zhang eds. Kluwer Academic Publishers, 1999.

[2] E.D. Anderson and Y. Ye, *On a homogeneous algorithm for the monotone complementarity problem,* Math. Program. 84 (1999) 375-400.

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

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

[5] J.R. Birge and F. Louveaux, *Introduction to Stochastic programming,* Springer-Verlag, 1997.

[6] J. Blomvall and P. O. Lindberg, *A Riccati-based primal interior point solver for multistage stochastic programming,* European J. Operational Res., 143 (2002) 452-461.

[7] B.J. Chun and S.M. Robinson, *Scenario analysis via bundle decomposition,* Ann. Oper. Res., 56(1995), 39-63.

[8] T. Helgason and S.W. Wallace, *Approximate scenario solutions in the progressive hedging algorithm,* Ann. Oper. Res., 31(1991), 425-444.

[9] P. Kall and S.W. Wallace, *Stochastic Programming,* Wiley, West Sussex, England, 1994.

[10] A.J. King, *Stochastic programming problems: examples from the literature,* Numerical Techniques for Stochastic Optimization, Y. Ermoliev and R. J-B Wets, eds., Springer-Verlag, 1988, 543-567.

[11] X.W. Liu and J. Sun, *A new decomposition technique in solving multistage stochastic linear programs by infeasible interior point methods,* J. Global Optim. 28 (2004) 197-215.

[12] I.J. Lustig, R.E. Marsten and D.F. Shanno, *On implementing Mehrotra's predictor-corrector interior-point method for linear programming,* SIAM J. Opt., 2 (1992) 435-449.

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

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

[15] J.M. Mulvey and A. Ruszczyński, *A new scenario decomposition method for large-scale stochastic optimization,* Operations Research, 43 (1995) 477-490.

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

[17] A. Ruszczyński, *Decomposition methods in stochastic programming,* Math. Prog., 79 (1997) 333-353.

[18] M. C. Steinbach, *Hierarchical sparsity in multistage convex stochastic programs,* Stochastic Optimization: Algorithms and Applications, S. P. Uryasev and P. M. Pardalos, eds., Kluwer Academic Publishers, 2001, 385-410.

[19] J. Sun, K.E. Wee and J.S. Zhu, *An interior point method for solving a class of linear-quadratic stochastic programming problems,* in: Recent advances in nonsmooth optimization, L. Qi and R. Womersley eds. World Sci. Publishing, River Edge, NJ, 1995, 392-404.

[20] S.J. Wright, *Primal-Dual Interior-Point Methods,* SIAM, Philadelphia, 1997.

[21] X. Xu, P. Hung and Y. Ye, *A simplified homogeneous and self-dual linear programming algorithm and its implementation,* Ann. Oper. Res., 62 (1996) 151-171.

[22] D. Yang and S.A. Zenios, *A scalable parallel interior point algorithm for stochastic linear programming and robust optimization,* Computational Optim. Appl., 7 (1997) 143-158.

[23] Y. Ye, *Interior Point Algorithms, Theory and Analysis,* John Wiley & Sons, Inc, 1997.

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

[25] Y. Zhang, *Solving large-scale linear programs by interior-point methods under the MATLAB environment,* Optimization Methods and Software, 10 (1998) 1-31.

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