On implementation of a Lagrangian dual method for solving multi-stage stochastic programming problems *

Xinwei Liu[†], Kim-Chuan Toh and Gongyun Zhao[‡]

Department of Mathematics, National University of Singapore, Singapore

Abstract

A Lagrangian dual method has been developed by [24] for solving multi-stage stochastic programs, which relaxes the nonanticipativity constraints by the Lagrangian dual approach and combines with the use of log-barrier method. This paper discusses in detail the implementation of this method. Moreover, we compare it with the well-known progressive hedging method developed by Rockafellar and Wets. Preliminary numerical results are presented.

keywords: Lagrangian dual, multi-stage stochastic programs, log-barrier method, progressive hedging method

1. Introduction

Stochastic programming deals with the optimization with data under uncertainty. Multi-stage stochastic programming problems arise in many practical situations, such as production and manpower planning, portfolio selections and so on. We consider the following stochastic program with recourse:

$$\begin{aligned} \min \quad & q_0(x) + E_{\xi_1} Q_1(x, \xi_1), && (1.1) \\ \text{s.t.} \quad & x \in X, && (1.2) \end{aligned}$$

*Research is partially supported by Grant R-146-000-006-112 of National University of Singapore.

†This author is on leave from Hebei University of Technology, Tianjin 300130, P.R. China

‡The paper is written during this author's visit to DAMTP of the University of Cambridge. The visit is granted by the Commonwealth Fellowship. This author would like to thank Professor Mike Powell for his support and hospitality

where for $t = 1, 2, \ldots, T$, the recourse functions are recursively defined by

$$Q_t(x, y_1, \cdots, y_{t-1}, \hat{\xi}_1, \cdots, \hat{\xi}_t) = \min_{y_t} q_t(x, y_1, \cdots, y_{t-1}, y_t, \hat{\xi}_1, \cdots, \hat{\xi}_t) \\ +E_{\xi_{t+1}} Q_{t+1}(x, y_1, \ldots, y_t, \hat{\xi}_1, \ldots, \hat{\xi}_y, \xi_{t+1}) \quad (1.3)$$

$$\text{s.t. } (x, y_1, \cdots, y_t) \in Y_{\hat{\xi}_1, \cdots, \hat{\xi}_t}. \quad (1.4)$$

Here $Q_{T+1} = 0$ and Q_1 has only deterministic variable $x \in \Re^{n_0}$. $\hat{\xi}_i$ is the realization of the random vector ξ_i. $y_i \in \Re^{n_i}$ is the decision vector in the i-th stage, which is generated recursively by (1.3) and (1.4), depending on $x, y_1, \cdots, y_{i-1}$ and $\hat{\xi}_1, \cdots, \hat{\xi}_i$, hence is random. q_t, $t = 0, 1, \ldots, T$, are real-valued functions on $\Re^{n_t}$. For $t \geq 1$, q_t is random since it is related to $\hat{\xi}_1, \cdots, \hat{\xi}_t$. The sets X and $Y_{\hat{\xi}_1, \cdots, \hat{\xi}_t}$ are convex. Again $Y_{\hat{\xi}_1, \cdots, \hat{\xi}_t}$ depends on random variables. The details on the formulation of multi-stage stochastic programs can be found, e.g. in Kall and Wallace [10].

There are many works devoted to stochastic linear programs, especially to two-stage stochastic linear programs, see [3], [5], [12], [23], and their references. Extending these methods to solving multi-stage stochastic nonlinear programming is not straightforward since they used the special structures and properties of stochastic linear programs.

Scenario analysis technique was introduced to deal with multi-stage stochastic programs by Rockafellar and Wets in 1987, which specifies the programs into a finite number of scenarios for the period considered. Each scenario occurs with a fixed and known probability. Scenarios are related if they are sharing a common history up to a moment of time. Figure 1 gives us a very direct and clear understanding on the technique.

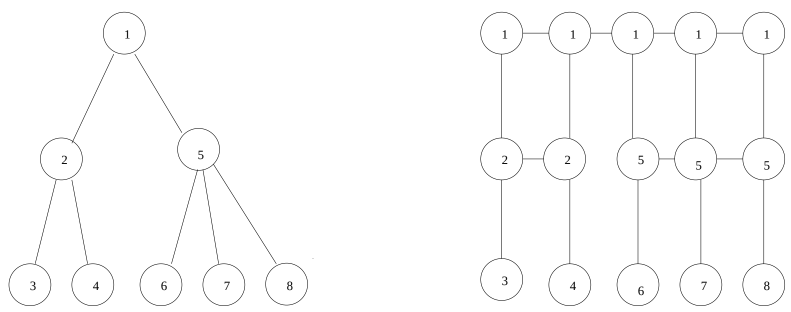

Figure 1: scenario analysis

Berland and Haugen [2] considered a different way of defining scenarios and hence altered the number of scenarios and the size of each scenario, which showed that the scenario analysis technique is flexible and can be taken for different problems.

Based on a scenario aggregation technique, Rockafellar and Wets (1991) proposed the progressive hedging method, which is an iterative algorithm for multi-stage stochastic programming. Suppose that there are finite instances for joint random vector $\xi = (\xi_1, \xi_2, \cdots, \xi_T)$ (Sampling can be introduced for the case with infinite instances, and an instance is also called a scenario), and S is the number of scenarios, $p_s(s = 1, \cdots, S)$ is the probability associated with the s-th scenario. The problem (1.1)-(1.4) can be reformulated as following:

$$\min\ F(z) = \sum_{s=1}^{S} p_s f_s(z_s) \tag{1.5}$$

$$s.t.\ z_s \in X_s,\ \ s = 1, 2, \cdots, S \tag{1.6}$$

$$z = (z_1; z_2; \cdots; z_S) \in Z \tag{1.7}$$

where $z_s \in \Re^n$, which consists of the decision vectors in all stages $i = 1, \cdots, T$, that is, $z_s = (x^s; y_1^s; \cdots; y_T^s)$, $X_s \subset \Re^n$, $f_s : \Re^n \to \Re$ and Z is a linear subspace of $\Re^{n \cdot S}$ defined by $z_{s_1 j} = z_{s_2 j} (j = 1, \cdots, i)$ for $\hat{\xi}_{s_1 j} = \hat{\xi}_{s_2 j} (j = 1, \cdots, i)$ ($z_{s_1 j}$ is the j-th subvector associated with stage j. Problem (1.5)-(1.7) may be very large and the storage and data-handling capability of a computer can be exceeded easily, thus can not be solved directly. The progressive hedging method developed by Rockafellar and Wets (1991) decomposed (1.5)-(1.7) into a set of scenario subproblems in each iteration and allowed them to be solved in parallel. We refer the readers to [17] for more details.

Mulvey and Vladimirou[13] applied the progressive hedging method (PHA in short) to stochastic generalized networks and achieved satisfactory results. Based on their numerical experiences for the solution of stochastic fisheries management models by PHA, Helgason and Wallace[9] suggested to solve the scenario subproblems in each iteration approximately. It has been noted by Mulvey and Vladimirou, Helgason and Wallace, and Chun and Robinson that the numerical behaviour of PHA can be affected largely by the selection of the penalty parameter associated with the proximal term. The convergence of PHA can be very slow if this parameter is selected unsuitably.

The numerical experiments of Chun and Robinson[7] showed that PHA is not the best alternative for the loosely-coupled scenario analysis problems, and the bundle-based decomposition is more competitive than PHA.

Based on scenario analysis, Ruszczyński [19] suggested an augmented Lagrangian method with the use of separable approximations. The convergence theory is also given for the method. Moreover, Ruszczyński(1993) developed a new parallel decomposition method, which admits the subproblems in an iteration to be solved asynchronously.

A new iterative method based on scenario analysis is proposed recently by Zhao (1998), which relaxes the nonanticipativity constraints by the Lagrangian dual approach and combines with the use of Log-Barrier method. We refer to this method as barrier Lagrangian dual method. The paper [24] is focused on fundamental theoretical aspects of the method, such as the global convergence and polynomial-time complexity. In this paper, we will describe the implementation on this method in detail. An algorithm based on this method is applied to solve some multi-stage stochastic programming problems. The results are compared with those obtained by PHA.

This paper is organized as follows. In section 2, the progressive hedging algorithm is described. We state the Lagrangian dual method and some topics on implementation in section 3. We can refer to Zhao (1998) for further theoritical results. Some multi-stage stochastic programming problems and the corresponding numerical results are presented in section 4. Some conclusions are given in section 5.

2. The progressive hedging method

Problem (1.5)-(1.7) is called the deterministic equivalent of (**??**)-(**??**), where constraints (1.7) are the so-called nonanticipativity constraints, which mean that the decision should be the same if they have the same history. In general, (1.7) can be expressed by a system of linear equations, and there are several formulations for the coefficient matrix. For example, Ruszczyński[19] uses a full rank matrix A with two nonzero elements 1 and -1 in each row of the matrix. Suppose that $\mathcal{K}$ is a set of scenarios at stage t with the same history, p_i is the associated probability with the i-th scenario, Rockaffellar and Wets write the nonanticipativity constraint as

$$z_{it} = \sum_{j \in \mathcal{K}} \frac{p_j}{\sum_{k \in \mathcal{K}} p_k} z_{jt}, \ \forall i \in \mathcal{K}, \tag{2.1}$$

where z_{it} is the subvector associated with stage t. In this case, the constraint (1.7) can be written as

$$Kz = 0$$

where K is a square matrix and not of full rank. If all $p_j (j = 1, \cdots, S)$ are the same, then (2.1) reduces to the simplest situation:

$$z_{it} = \frac{1}{|\mathcal{K}|} \sum_{j \in \mathcal{K}} z_{jt}, \ i \in \mathcal{K}, \tag{2.2}$$

where $|\mathcal{K}|$ is the cardinality of $\mathcal{K}$. For K defined by (2.1), $J = I - K$ is the orthogonal projection on the subspace Z.

Instead of (1.5)-(1.7), Rockafellar and Wets(1991) considered the augmented Lagrangian

$$L_\beta(z, u) = F(z) - u^T z + \frac{1}{2}\beta\|Kz\|^2 \tag{2.3}$$

for $u = Ky$ and $z_s \in X_s$ for $s = 1, 2, \cdots, S$, where y is the approximate multiplier related to the nonanticipativity constraints, $\beta > 0$ is a penalty parameter. Since the augmented term $\|Kz\|^2$ is not decomposable into seperate terms for each scenario, the progressive hedging algorithm generates the new iteration point z_{k+1} by solving the following subproblem

$$\min_z \ F(z) - u_k^T z + \frac{1}{2}\beta\|z - \tilde{z}_k\|^2 \tag{2.4}$$

$$\text{s.t. } z_s \in X_s, \quad s = 1, 2, \cdots, S \tag{2.5}$$

where $\tilde{z}_k = Jz_k$ and z_k is the current iteration point, which decomposes into solving for each scenario s the subproblem

$$\min_{z_s} \ f_s(z_s) - u_{ks}^T z_s + \frac{1}{2}\beta\|z_s - \tilde{z}_{ks}\|^2 \tag{2.6}$$

$$\text{s.t. } z_s \in X_s. \tag{2.7}$$

The difference between (2.3) and (2.4) is that the augmented term in (2.3) is replaced by the decomposable proximal term in (2.4). u_k is updated by $u_{k+1} = u_k - \beta K z_{k+1}$.

The progressive hedging algorithm can be stated as follows.

Algorithm 2.1 *(The progressive hedging algorithm)*

Step 0 *Given the tolerance $\epsilon > 0$, penalty parameter $\beta > 0$, set initial price $u_0 = 0$. Given the initial approximation of solution z_0, compute its projection $\tilde{z}_0 = Jz_0$. Let $k = 0$;*

Step 1 *Solve all the scenario subproblems (2.6)-(2.7) to generate z_{k+1};*

Step 2 *If $\{\|J(z_{k+1} - z_k)\|_p^2 + \|Kz_{k+1}\|_p^2\}^{\frac{1}{2}} < \epsilon$, stop;*

Step 3 *Compute $\tilde{z}_{k+1} = Jz_{k+1}$ and $u_{k+1} = u_k - \beta K z_{k+1}$. Let $k = k + 1$ and go to step 1.*

In the algorithm $\|z\|_p = \{E(\|z_s\|^2)\}^{\frac{1}{2}}$ is a norm defined by expectation. If ξ has a finite distribution with same probability for each instance, then $\|z\|_p^2 = \frac{1}{S}\|z\|^2$, where $\|\cdot\|$ is the standard Euclidean norm. Instead of $\|Kz_{k+1}\| < \epsilon$, the reason for using $\{\|J(z_{k+1} - z_k)\|_p^2 + \|Kz_{k+1}\|_p^2\}^{\frac{1}{2}} < \epsilon$ as the stopping condition in step 3 of Algorithm 2.1 is that $\|Kz_{k+1}\|_p$ can be very small even if z_{k+1} is far from the optimal solution of the original problem. On the other hand, it has been proved by Rockafellar and Wets that for

$$\gamma_k = \{\|J(z_{k+1} - z_k)\|_p^2 + \|Kz_{k+1}\|_p^2\}^{\frac{1}{2}}, \tag{2.8}$$

$\{\gamma_k\}$ is a nonincreasing monotonic sequence.

Problem (2.6)-(2.7) can be solved by standard nonlinear programming methods, such as the well-known successive quadratic programming (SQP in short) methods. For stochastic linear programming, (2.6)-(2.7) is a convex quadratic programming. Since the special structure of the decomposition, the optimal point at the k-th iteration can be taken as the initial approximation of the solution of the approximate problem at $(k+1)$-th iteration. This is similar to the "warm start" technique used in SQP methods for nonlinear programs. In our implementation, (2.6)-(2.7) is solved approximately. The tolerance for termination is taken as $\gamma_k \cdot 10^{-3}$.

3. The Lagrangian dual method

3.1. Description of the algorithm. The Lagrangian dual method is presented for solving convex stochastic programs, and deals with the nonanticipativity constraints in the way as Ruszczyński[19] suggests. A is the coefficient matrix, we have $Az = \sum_{s=1}^{S} A_s z_s$, where A_s is the s-th block of A with n columns. Suppose that $X_s = \{z_s : c_s(z_s) \le 0\}$, where $c_s : \Re^n \to \Re^{m_s}$. By using the Lagrangian dual approach and Log-Barrier method, problem (1.5)-(1.7) is converted to the following problem:

$$\begin{aligned} \max_u \min_z \ & \sum_{s=1}^{S} p_s f_s(z_s) - r \sum_{s=1}^{S} \sum_{i=1}^{m_s} \ln(-c_{si}(z_s)) \\ & -u^T Az + \frac{1}{2}\beta\|Az\|^2, \end{aligned} \tag{3.1}$$

where $z = (z_1; \cdots; z_S)$ is assumed to be an interior point of (1.5)-(1.6), $r > 0$ is the barrier parameter and tends to zero gradually.

In order to decompose (3.1), a decomposition technique should be introduced to deal with the augmented term. For any fixed r, we introduce an approximate problem to (3.1):

$$\begin{aligned} \max_u \min_z \ & \sum_{s=1}^{S} \{p_s f_s(z_s) - r \sum_{i=1}^{m_s} \ln(-c_{si}(z_s)) \\ & -u^T A_s z_s + \frac{1}{2}\beta\|A_s(z_s - \tilde{z}_{-s})\|^2\}, \end{aligned} \tag{3.2}$$

where $(\tilde{z}_{-1}; \tilde{z}_{-2}; \cdots; \tilde{z}_{-S}) = (I - A^T(AA^T)^{-1}A)z_-$ is the projection of z_- on Z and z_- is the current iteration point.

Define

$$G_s(z_s, u, r; \tilde{z}_{-s}) = p_s f_s(z_s) - r \sum_{i=1}^{m_s} \ln(-c_{si}(z_s))$$
$$-u^T A_s z_s + \frac{1}{2}\beta \|A_s(z_s - \tilde{z}_{-s})\|^2. \tag{3.3}$$

Then (3.2) can be written as

$$\max_u \min_z \sum_{s=1}^{S} G_s(z_s, u, r; \tilde{z}_{-s}). \tag{3.4}$$

If the initial approximation of the solution of (3.4) is the interior point of (1.5)-(1.6), then for any fixed u and the barrier parameter r, problem (3.4) can be decomposed into S unconstrained subproblems, which is

$$\min_{z_s} G_s(z_s, u, r; \tilde{z}_{-s}) \text{ for } s = 1, \cdots, S \tag{3.5}$$

and z_s is the interior point of (1.5)-(1.6).

For twice continuously differentiable convex functions f and $c_s(s = 1, \cdots, S)$, if an initial interior point of (1.5)-(1.6) is given, (3.5) is a set of smooth strictly convex barrier subproblems and can be solved by many efficient methods, such as Newton method, for unconstrained optimization. Problem (3.4) is called the main problem. Suppose that z_+ is the solution of (3.5), a Newton step on u can be generated by applying Newton method to (3.4):

$$d = -\{\sum_{s=1}^{S} A_s(\nabla^2_{z_s} G_s(z_{+s}, u, r; \tilde{z}_{-s}))^{-1} A_s^T\}^{-1} A z_+, \tag{3.6}$$

where $\nabla^2_{z_s} G_s(z_{+s}, u, r; \tilde{z}_{-s})$ is the Hessian of the function in (3.5) at the point z_+, by (3.3), which is not affected by $\tilde{z}_-$. Thus, u can be updated by

$$u_+ = u + \alpha d, \tag{3.7}$$

where α is a stepsize decided by some line search procedure. Then a new approximation of the solution will be derived by solving (3.5).

Suppose that $(u^*(r), z^*(r))$ is the solution of (3.1) associated with r, it has been proved by Zhao (1998) that if $r \to 0$, $(u^*(r), z^*(r))$ will converge to the solution of (1.5)-(1.7).

One of the difficulties for the above procedure is the computation of (3.6). Some methods have been proposed in [24] to compute the Newton direction d defined in (3.6). By exploiting special structure of A, the Newton direction d can be computed in $O(n^3 S)$ arithmetic operations, where n is the number of variables in the subproblem (1.5)-(1.7).

Now we described the algorithm in detail.

Algorithm 3.1 *(The Lagrangian dual algorithm)*

Step 0 *Given $\nu > 0$, $r_0 > 0$, $\beta \geq 0$, and tolerances on stopping criterions. An initial interior point of (1.5)-(1.6) z_0 is given, compute $\tilde{z}_0 = (I - A^T(AA^T)^{-1}A)z_0$. Set $u_0 = 0$, $j = 0$;*

Step 1 *Start the outer iteration. Let $z_0 = z_j$ and $u_0 = u_j$, $\tilde{z}_0 = \tilde{z}_j$. By using z_0 as the initial point, we solve the unconstrained optimizations*

$$\min_{z_s} \ G_s(z_s, u_0, r_j; \tilde{z}_{0s}) \tag{3.8}$$

iteratively. Let $\bar{z}_{0s}$ be the solution for $s = 1, 2, \cdots, S$, compute the Newton direction

$$d_0 = -\{\sum_{s=1}^{S} A_s(\nabla^2_{z_s} G_s(\bar{z}_{0s}, u_0, r_j; \tilde{z}_{0s}))^{-1} A_s^T\}^{-1} A\bar{z}_0. \tag{3.9}$$

Let $k = 0$;

Step 2 *Start the inner iterations. Update u_k by $u_{k+1} = u_k + \alpha_k d_k$, where α_k is the step-size along d_k such that*

$$\sum_{s=1}^{S} G_s(z_{(k+1)s}, u_{k+1}, r_j; \tilde{z}_{ks}) = \max_{0<\alpha\leq 1} \min_{z} \sum_{s=1}^{S} G_s(z_s, u_k + \alpha d_k, r_j; \tilde{z}_{ks}), \tag{3.10}$$

where $z_{(k+1)s}$ minimizes $G_s(z_s, u_{k+1}, r_j; \tilde{z}_{ks})$;

Step 3 *Calculate $\tilde{z}_{k+1} = (I - A^T(AA^T)^{-1}A)z_{k+1}$ and*

$$d_{k+1} = -\{\sum_{s=1}^{S} A_s(\nabla^2_{z_s} G_s(z_{(k+1)s}, u_{k+1}, r_j; \tilde{z}_{(k+1)s}))^{-1} A_s^T\}^{-1} A z_{(k+1)}; \tag{3.11}$$

Step 4 *Check the stopping criterion on u (call it "criterion 1"). If it is satisfied, goto step 5; Otherwise, let $k = k + 1$ and goto step 2.*

Step 5 *Check the stopping criterion on r (call it "criterion 2"). If it holds, stop; Otherwise, let $r_{j+1} = \nu r_j$, $z_{j+1} = z_{k+1}$, $u_{j+1} = u_{k+1}$, $\tilde{z}_{j+1} = \tilde{z}_{k+1}$, $j = j + 1$ and goto step 1.*

Let $\delta(r_j, u_{k+1}) = \sqrt{\frac{1}{r_j} d_k^T(-Az_{k+1})}$. Zhao [24] suggests to take $\delta(r_j, u_{k+1}) \leq \epsilon_0$ and $r_j \leq \epsilon$ as the stopping criterion 1 and 2 respectively, where $\epsilon_0 > 0$ and $\epsilon > 0$ are tolerance scalars. Since A has special structure, computing $\tilde{z}_k (k = 0, 1, \cdots)$ is not very difficult although S may be very large. In our implementation of Algorithm 3.1, the exact line search in step 2 is not neccessary, so the cost of the algorithm can be reduced largely.

3.2. The solution of the unconstrained Log-barrier subproblem. Since f_s and c_{si} are convex, functions defined by (3.3) are strictly convex. Thus, (3.5) are a set of strictly convex unconstrained optimization problems. We do not solve these problems exactly. In our implementation, we use the stopping condition

$$\|d_z\| \leq 10^{-3} r, \tag{3.12}$$

where d_z is the Newton direction, $\|\cdot\|$ is the Euclidean norm, r is the logarithmic barrier. Thus, the exactness of the solution to (3.5) is changed with r. Moreover, by selecting the initial trial step such that the new iteration point is feasible to the original problem, we have taken two strategies for doing line search. At first, we do an inexact line search, the condition

$$G_s(z_s + \alpha d_{zs}, u, r; \tilde{z}_s) \leq G_s(z_s, u, r; \tilde{z}_s) + 0.05\alpha \nabla_s G_s(z_s, u, r; \tilde{z}_s)^T d_{zs} \tag{3.13}$$

is used to decide if we will accept the stepsize α. If the answer is negative, then we proceed to do the exact line search procedure. In our implementation, we use the golden section search method, which is successful and effective for the unconstrained optimization. It is noted when the iteration is far from the solution to the original problem, the step-sizes for most cases are decided by (3.13). But when the iteration is near the solution, the exact line search may be needed for the process of the algorithm.

3.3. The line search on u. Theoretically, the convergence can be guaranteed by using the step-length $\sigma = 1/(1+\delta)$. In practice, however, this step-length is usually too small. Hence, line search is needed. It is noticed that the evaluation of the dual objective function is almost the same as the evaluation of its gradient, both need to solve an entire set of subproblems. Therefore, the line search should use both the function and its gradient and check as few points as possible.

Let $f(u)$ be the dual objective function and $g(\alpha) = f(u_0 + \alpha \Delta u)$. Then $g'(\alpha) = \nabla f(u_0 + \alpha \Delta u)^T \Delta u$. Given $\alpha_1 > 0$, we will use the four values $g(0)$, $g'(0)$, $g(\alpha_1)$ and $g'(\alpha_1)$ to construct a curve and use its maximum to approximate the maximum of $g(\alpha)$. Normally, an interpolation polynomial is used as an approximate curve. But in our implementation this does not work well. As we observed, $g(\alpha)$ has a very special form (see Fig. 2), its upward side and downward side are both very stiff, and almost linear on a large partion of the curve. Hence we assume the approximate curve to be linear in the intervals $[0, t_0]$ and $[t_1, \alpha_1)$, and quadratic in $[t_0, t_1]$, i.e.

$$h(\alpha) = \begin{cases} g(0) + g'(0)\alpha, & \alpha \in [0, t_0) \\ b_0 + b_1\alpha + b_2\alpha^2, & \alpha \in [t_0, t_1) \\ g(\alpha_1) + g'(\alpha_1)(\alpha - \alpha_1), & \alpha \in [t_1, \alpha_1] \end{cases} \tag{3.14}$$

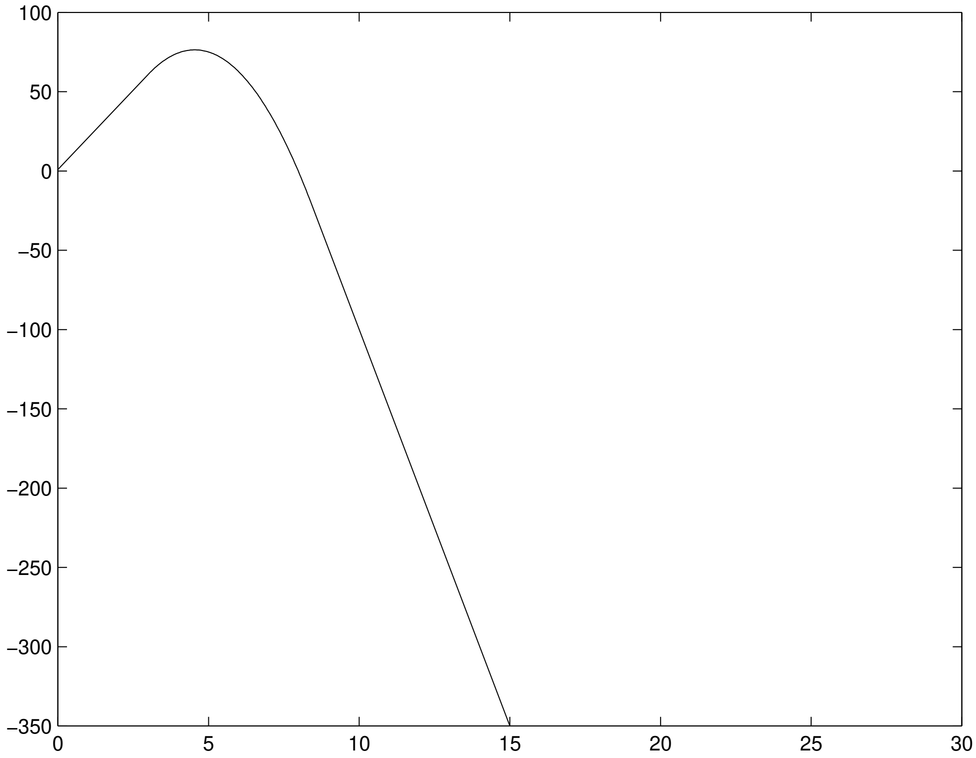

Figure 2: Line search on μ

Here, we assume α_1 is on the downward side, i.e. $\alpha_1 > t_1$. We will consider the other cases later on.

Let ζ be the intersection of the two lines, we observed that $t_0 = \zeta/3$ ia an appropriate choice. So we are left to determine b_0, b_1, b_2 and t_1, and it can be done by using $h(t_0)$, $h'(t_0)$, $h(t_1)$ and $h'(t_1)$, i.e. by solving the following equations

$$g(0) + g'(0)t_0 = b_0 + b_1 t_0 + b_2 t_0^2, \tag{3.15}$$

$$g'(0) = b_1 + 2b_2 t_0, \tag{3.16}$$

$$g(\alpha_1) + g'(\alpha_1)(t_1 - \alpha_1) = b_0 + b_1 t_1 + b_2 t_1^2, \tag{3.17}$$

$$g'(\alpha_1) = b_1 + 2b_2 t_1. \tag{3.18}$$

Then the maximum of $h(\alpha)$ is obtained at $\bar{\alpha} = -\frac{b_1}{2b_2}$. The system (3.15)-(3.18) has a closed form solution which can be derived by a few steps of algebraic manupulations.

In the following two cases, we do not perform the above process, but simply choose α_1 as the step-length:

$$(i)\ g(\alpha_1) \geq g(0) + 0.5 g'(0)\alpha_1; \tag{3.19}$$

$$(ii)\ g'(\alpha_1) \geq -0.1 g'(0). \tag{3.20}$$

In case (i), $g(\alpha_1)$ is large enough. In case (ii), α_1 is either on the upward side or is not too far down from the maximum. Our line search subroutine goes as follows:

Algorithm 3.2

Step 0. *Set* $\alpha_1 = ustfac * \sigma$ *(*$ustfac$ *depends on the size of the problem, e.g.* $ustfac = \sqrt{ns}$*). Compute* $g(0)$, $g'(0)$, $g(\alpha_1)$ *and* $g'(\alpha_1)$*. Set* $count = 1$.

Step 1. *If at least one of the conditions (i) and (ii) is satisfied, or* $count \geq 3$*, stop; Otherwise, goto step 2.*

Step 2. *Solve (3.15)-(3.18) and reset* $\alpha_1 = -\frac{b_1}{2b_2}$*, and* $count = count + 1$*. goto step 1.*

In our experiment, "count" has seldom reached 3. This shows the line search is very efficient.

4. Multi-stage stochastic linear test problems

Algorithm 2.1 and Algorithm 3.1 are programmed in MATLAB and run under version 5.3 on a microcomputer with the Linux operating system. In this section, we report some preliminary numerical results on the algorithms applied to the multi-stage stochastic linear programming test problems. Since it is a standard quadratic programming in this case, the subproblem (2.6)-(2.7) in Algorithm 2.1 is solved by the M-file *quadprog.m* in MATLAB library. For all test problems, we use 10^{-3} as the tolerance number for Algorithm 2.1, and 10^{-6} for the barrier parameter in Algorithm 3.1.

Since the two algorithms use different stopping criteria and the exact optimal solutions are not known, comparisom of objective values and the violation of constraints are made.

In the following tables, NIT and NSS stand for the number of iterations and the number of solved scenario subproblems respectively. RNC and VIC represent the l_2 norms of the residual of the nonanticipativity constraints and the violation of the inequality constraints respectively. CPU is the CPU time of the computer. OBF is the optimal value of the original objective function. For Algorithm 2.1, NIT is always the same as NSS. Since Algorithm 3.1 is an interior point method, VIC is always zero for Algorithm 3.1. Thus, NSS and VIC are not listed specially Algorithm 2.1 and Algorithm 3.1 respectively.

4.1. The stochastic linear programs with coefficients generated by random number generators. We formulate the programs similar to [4], which have the following form:

$$\min\ c_1 z_1 + c_2 z_2 + \cdots + c_T z_T \tag{4.1}$$

$$\text{s.t. } z_t \in \Re^{n_t},\ \ t = 1, \cdots, T \tag{4.2}$$

$$l_t \leq z_t \leq u_t, \ \ t = 1, \cdots, T \tag{4.3}$$

$$A_1 z_1 \leq b_1 \in \Re^{m_1} \tag{4.4}$$

$$A_{21} z_1 + A_{22} z_2 \leq b_2 \in \Re^{m_2} \tag{4.5}$$

$$\cdots \tag{4.6}$$

$$A_{T1} z_1 + A_{T2} z_2 + \cdots + A_{TT} z_T \leq b_T \in \Re^{m_T}. \tag{4.7}$$

For simplicity of statement, we suppose that all scenarios have same probabilities. The coefficients in (4.1) is

$$c = (c_1, c_2, \cdots, c_T)^T = (-1, -1, \cdots, -1)^T. \tag{4.8}$$

For $t = 1, \cdots, T$, we assume that all components of l_t and u_t are 0 and 100 respectively. All coefficient matrices in (4.4)-(4.7) are generated by random number generators. Concretely speaking, suppose that there are S_i scenarios for the random vector in stage i (that is, $S = S_1 S_2 \cdots S_T$ and $S_1 = 1$), firstly we generate S $m \times n$ matrices with all entries chosen from a uniform distribution on the interval $(-0.5, 0.5)$, then let A_1 be the first $m_1 \times n_1$ block of the first $m \times n$ matrix for all scenarios S/S_1, A_{21}, A_{22} be the $m_2 \times n_1$ and $m_2 \times n_2$ blocks consisting of entries from row $(m_1 + 1)$ to $(m_1 + m_2)$ and column 1 to n_1, $n_1 + 1$ to $n_1 + n_2$ respectively of the first $m \times n$ matrix for first $S/(S_1 S_2)$ scenarios and be the $m_2 \times n_1$ and $m_2 \times n_2$ blocks consisting of entries from row $(m_1 + 1)$ to $(m_1 + m_2)$ and column 1 to n_1, $n_1 + 1$ to $n_1 + n_2$ respectively of the $(S/(S_1 S_2) + 1) - th$ $m \times n$ matrix for scenarios $S/(S_1 S_2) + 1$ to $2S/(S_1 S_2)$. We proceed the similar process till all $A_{ij} (i, j = 1, \cdots, T)$ are generated. In order to keep (4.4)-(4.7) consistent, we select the right-hand-sides $b_i = A_{i1} e_1 + \cdots + A_{ii} e_i + \hat{b}_i (i = 1, \cdots, T)$, where $e_j (j = 1, \cdots, i)$ is a vector with dimension n_j and all entries being one, $\hat{b}_i \in \Re^{m_i}$ is a vector with all entries chosen from a uniform distribution on the interval $(0, 1)$, which changes with the stage and scenario similar to A_{ij}. Thus, $z = (1, 1, \cdots, 1)$ is feasible to the program.

Let $m = \sum_{i=1}^{T} m_i$. By changing the numbers n, m, T and S of variables, constraints, stages and scenarios respectively, we generate a set of multi-stage stochastic linear test problems, which are summerized in Table 1. Our numerical results for Algorithm 2.1 and Algorithm 3.1 are listed in Table 2 and Table3 respectively.

In order to observe how the penalty parameter β affects the algorithms, we select β three different values respectively to test Algorithm 2.1 and Algorithm 3.1. The numerical results are listed in Table 4.

4.2. A multi-stage production planning problem. We consider the concrete application of stochastic programming to a multi-stage production planning problem. In order to meet a random demand for its products over several periods, a factory must decide on a production schedual and plan increases/decreases in production activity with different

Table 1. Test problems

Problem name	Number of stage	Number of scenario	nonanticipativity constraints	Size of the problem row× column
LP1	2	6	20	146×48
LP2	3	6	24	168×54
LP3	3	16	105	681×208
LP4	3	16	192	1248×320
LP5	5	32	483	2595×672
LP6	4	54	808	4102×1080
LP7	5	256	5124	24836×7168

Table 2. Numerical results for Algorithm 2.1

Problem	β	NIT(NSS)	RNC	VIC	CPU	OBF
LP1	0.1	364	3.1029e-03	1.0897e-29	148.39	-175.8469
LP2	0.1	278	3.2195e-03	2.5367e-29	119.92	-178.8426
LP3	1	4176	8.2998e-04	3.2203e-28	6140.11	-198.7202
LP4	0.1	1027	2.4948e-03	1.6911e-26	2599.65	-209.0337
LP5	0.1	5189	2.3438e-03	2.0888e-26	29820.43	-408.3469
LP6	0.1	5611	4.0961e-03	6.8383e-28	37204.17	-126.3110
LP7						

Table 3. Numerical results for Algorithm 3.1

Problem	β	NIT	NSS	RNC	CPU	OBF
LP1	5×10^{-3}	17	37	1.5637e-09	5.92	-175.8471
LP2	5×10^{-3}	35	58	1.1266e-10	8.80	-178.8348
LP3	5×10^{-3}	49	70	1.5498e-09	35.17	-198.7540
LP4	10^{-5}	57	88	9.8654e-08	74.61	-209.0342
LP5	10^{-5}	79	112	2.3945e-05	186.68	-408.3594
LP6	5×10^{-3}	135	182	1.5633e-09	454.43	-126.3128
LP7	10^{-5}	293	560	2.1963e-08	11616.03	-619.6291

Table 4. The CPU time for different β

	Algorithm 2.1			Algorithm 3.1		
Problem	$\beta = 0.01$	$\beta = 0.1$	$\beta = 1$	$\beta = 10^{-3}$	$\beta = 10^{-4}$	$\beta = 10^{-5}$
LP1	180.59	148.39	357.92	6.31	7.31	8.61
LP2	122.23	119.92	681.92	8.75	9.99	11.31
LP3	8200.42	11530.47	6140.11	35.31	36.19	41.82
LP4	8212.74	2599.65	18613.95	76.63	79.16	74.61
LP5	32746.13	29820.43	58825.20	262.21	209.55	186.68
LP6	38699.68	37204.17		480.99	497.25	526.64
LP7					12565.36	11616.03

periods. The details on modeling this problem as the optimization program can be seen in [14]. As a test problem for multi-stage stochastic program with general recourse, it is described with complete data in [11]. At each stage, the decision is selected as a function of the decisions and realizations of the random variables of the previous stages, and the expected future outcomes of the random variables. It can be thought as a dynamic programming type of problem. Since the high dimension of the state space of the problem, however, it seems computationally difficult to solve it by some dynamic programming based scheme.

A solution for this problem has been presented in [14]. In this paper, by using scenario analysis technique, we reformulate it as a 5-stage stochastic program with recourse. $x_{j,t}$, $u_{i,t}(j = 1, 2; t = 1, \cdots, 4)$, w_t^+ and $w_t^-(t = 1, 2, 3)$ are taken as variables in the first stage. $y_{j,t}^+$ and $y_{j,t}^-(j = 1, 2; t = 1, \cdots, 4)$ are variables in the $(t + 1)$-th stage (see [11]). We divide the range of each random variable ξ into several intervals (say, $I_i(i = 1, \cdots, s)$) with the same probabilities, and calculate $\xi_i = E\{\xi|I_i\}$. Then the random variable ξ is approximated by the discrete random variable with realizations $\xi_i(i = 1, \cdots, s)$ and its corresponding probability $\frac{1}{s}$. By eliminating the equality constraints, we derive a determinictic equavalent of the 5-stage stochastic program with recourse, which has only inequality constraints:

$$\begin{aligned} \min \sum_{t=1}^{4}\sum_{j=1}^{2}(c_j + q_{j,t}^0 + (-1)^t a_{3j} e^+)x_{j,t} + \sum_{t=1}^{4}\sum_{i=1}^{3} d_i u_{i,t} + \sum_{t=1}^{3}(e^+ + e^-)w_t^- \\ + \sum_{t=1}^{4}\sum_{j=1}^{2}(q_{j,t}^+ + q_{j,t}^0)y_{j,t}^+ - \sum_{t=1}^{4}\sum_{j=1}^{2} q_{j,t}^0 \xi_{j,t} \end{aligned} \tag{4.9}$$

$$\text{s.t. } x_{j,t} \geq 0, \quad j = 1, 2; t = 1, \cdots, 4 \tag{4.10}$$

$$u_{i,t} \geq 0, \quad i = 1, \cdots, 3; t = 1, \cdots, 4 \tag{4.11}$$

Table 5. The solution for the multi-stage production planning

		$t=1$	2	3	4
$x:$	$j=1$	335.90	284.10	444.34	403.85
	$j=2$	563.18	585.39	398.85	376.92
$u:$	$i=1$	159.54	63.32	0.00	0.00
	$i=2$	0.00	0.00	0.00	0.00
	$i=3$	450.00	450.00	375.00	350.00
$w^-:$		0.00	825.00	275.00	

$$w_t^- \geq 0, \quad t = 1, \cdots, 3 \tag{4.12}$$

$$\sum_{j=1}^{2} a_{3j}(x_{j,t} - x_{j,t+1}) - w_t^- \leq 0, \quad t = 1, \cdots, 3 \tag{4.13}$$

$$\sum_{j=1}^{2} a_{ij} x_{j,t} - u_{i,t} \leq b_{i,t}, \quad i = 1, \cdots, 3; t = 1, \cdots, 4 \tag{4.14}$$

$$u_{i,t} \leq f_{i,t}, \quad i = 1, \cdots, 3; t = 1, \cdots, 4 \tag{4.15}$$

$$y_{j,t}^+ \geq 0, \quad j = 1, 2; t = 1, \cdots, 4 \tag{4.16}$$

$$x_{j,1} + y_{j,1}^+ \geq \xi_{j,1}, \quad j = 1, 2 \tag{4.17}$$

$$x_{j,t-1} + x_{j,t} + y_{j,t-1}^+ + y_{j,t}^+ \geq \xi_{j,t-1} + \xi_{j,t}, \quad j = 1, 2; t = 2, 3, 4 \tag{4.18}$$

where $q_{j,t}^0 = \sum_{i=t}^4 q_{j,t}^-$, and the coefficients are given in [11]. It is easily noted that (4.9)-(4.18) is a 5-stage problem with 31 variables (23 variables for the first stage and 2 for each other stage) and 66 constraints (50 constraints for the first stage and 4 for each other stage). By using the discrete random variable with two realizations to approximate each random variable, the problem is a stochastic program with 256 scenarios. The solution given by Algorithm 2.1 and Algorithm 3.1 is listed in Table 5. Some small-size reformulations are also used to test Algorithm 2.1 and Algorithm 3.1. The sizes of problems are listed in Table 6. The numerical results are listed in Table 7 and Table 8.

4.3. The manpower planning problem. An employer must decide upon the level of regular staff at various skill levels. In order to meet the random demand for services at minimum cost, the employer should make a schedual for temporary outside manpower or regular staff overtime. This problem can be modeled as a multi-stage stochastic program with general recourse, which is decribed in [11]. We have tested four problems with different sizes (see Table 6), the numerical results are listed in Table 7 and Table 8.

Table 6. Production and manpower planning problems

Problem name	Number of stage	Number of scenario	nonanticipativity constraints	Size of the problem row× column
PP1	3	16	189	701×240
PP2	3	256	3285	11477×3840
PP3	5	256	7233	24129×7936
MP1	3	16	117	549×240
MP2	3	64	525	2253×960
MP3	5	16	249	1065×432
MP4	5	256	4869	17925×6912

Table 7. Numerical results for Algorithm 2.1

Problem	β	NIT(NSS)	RNC	VIC	CPU	OBF
PP1	0.5	131	4.3648e-03	1.0663e-25	194.50	243958.6404
PP2	0.1	142	1.7598e-03	2.7965e-23	3542.23	247373.7902
PP3	5×10^{-3}	197	1.6161e-02	2.9970e-20	15622.59	594574.8905
MP1	10^{-3}	330	1.6033e-03	2.6230e-23	503.26	80628.8417
MP2	10^{-4}	798	4.7010e-03	2.4118e-22	5348.49	80664.8142
MP3	10^{-4}	83	4.4519e-04	7.6776e-23	311.39	85830.4461
MP4	10^{-4}	406	1.9306e-02	1.3776e-21	19940.98	86149.2761

Table 8. Numerical results for Algorithm 3.1

Problem	β	NIT	NSS	RNC	CPU	OBF
PP1	10^{-4}	26	42	2.6847e-07	22.03	243958.6110
PP2	10^{-4}	20	41	4.3284e-08	540.49	247373.7894
PP3	10^{-4}	25	42	1.6554e-07	1656.56	594575.0862
MP1	5×10^{-5}	34	58	2.9510e-08	24.72	80628.8422
MP2	10^{-6}	39	64	4.2034e-04	123.11	80664.8127
MP3	10^{-5}	17	29	2.3034e-09	23.80	85830.4465
MP4	10^{-6}	52	95	2.0830e-07	2043.56	86149.2823

Table 9. Numerical results for stochastic nonlinear program

Algorithm	NIT	NSS	RNC	VIC	CPU	OBF
Algorithm 2.1	330	330	2.0637e-03	2.4814e-29	280.83	-149.2547
Algorithm 3.1	11	24	7.5654e-10	0	7.33	-149.2470

5. Multi-stage stochastic nonlinear test problems

Algorithm 2.1 and Algorithm 3.1 are also used to solve multi-stage stochastic nonlinear test problems. The first test problem is derived by adding a nonlinear part $\frac{1}{2}z_1^2 + e^{z_2}$ to the objective function and a nonlinear inequality constraint

$$\sin(z_1) + z_2^2 + e^{1+z_3} \leq 25 \tag{5.1}$$

in LP2. Since LP2 is a 3-stage 6-scenario stochastic linear program, and there are 3-variable and 2-constraint for each stage, 3-scenario for stage 2 and 2-scenrio for stage 3, the generated problem is a 3-stage 6-scenario stochastic nonlinear program, and the nonlinear constraint is in the first stage. The numerical results for this problem are listed in Table 9.

The other test problems are derived by adding 4 nonlinear constraints to LP1, LP2, LP3 and LP4, where the first two constraints are

$$z_1^2 + z_2^2 - 2z_3 \leq 15, \tag{5.2}$$

$$3z_1 + z_2^3 + \frac{1}{2}z_3^2 \leq 160, \tag{5.3}$$

which are related to the first stage of problems, the rest two constraints are corresponding to the second and third stage, i.e., for two-stage problem LP1, they are constraints to the second stage, one is to the second stage and another is to the third stage for LP2, Lp3 and LP4. The type of these constraints are

$$\sum_{i=1}^{\bar{n}_2} d_{1i} z_i^2 \leq 300, \tag{5.4}$$

$$\sum_{i=1}^{\bar{n}_3} d_{2i} z_i^3 \leq 750, \tag{5.5}$$

where $\bar{n}_j (j = 2, 3)$ is the number of variables up to the j-th stage, $d_{ji} (j = 1, 2)$ are related to scenario, thus are random. We generate d_{ji} by random number generator such that $0 < d_{ji} < 1$ for all j and i, β is selected as 0.1 and 0.005 in Algorithm 2.1 and Algorithm 3.1 respectively. The numerical results are presented in Table 10.

Table 10. Numerical results for stochastic nonlinear programs

Alg	Prob	NIT	NSS	RNC	VIC	CPU	OBF
Alg 2.1	NLP1	249	249	3.8277e-03	1.1331e-12	126.39	-33.7382
	NLP2	116	116	2.8263e-03	8.3683e-13	79.36	-40.7556
	NLP3	267	267	4.5741e-03	6.0279e-12	502.46	-41.4489
	NLP4	340	340	3.6461e-03	3.0051e-12	1.1867e+03	-52.4869
Alg 3.1	NLP1	24	60	1.6279e-09	0	36.86	-33.7373
	NLP2	19	36	1.6193e-09	0	18.67	-40.7544
	NLP3	45	75	4.3503e-11	0	107.27	-41.4499
	NLP4	70	116	2.2919e-11	0	235.74	-52.4852

6. Observations and Remarks

From their experiment for Algorithm 2.1, Helgason and Wallace[9] observed that the penalty parameter β should be as small as possible, provided it is large enough to guarantee convergence. We observed from our implementation for Algorithm 2.1, however, small penalty may need less number of iterations, a suitable large penalty parameter can result in less CPU time, the reason is that the scenario subproblems may be solved more rapidly. We noted that implementation on Algorithm 3.1 has the same case. An interesting observation on Algorithm 2.1 is that the l_2 norms of residuals of the nonanticipativity constraints are not very small at the solution.

References

[1] A. Berkelaar, C. Dert, B. Oldenkamp and S. Zhang, *A primal-dual decomposition-based interior point approach to two-stage stochastic linear programming,* EI Report 9918/A, EUR.

[2] N.J. Berland and K.K. Haugen, *Mixing stochastic dynamic programming and scenario aggregation,* Stochastic programming, algorithms and models (Lillehammer, 1994). Ann. Oper. Res. 64 (1996), 1-19.

[3] J.R. Birge, *Current trends in stochastic programming computation and applications,* Technique Report 95-15, Department of Industrial and Operations Engineering, University of Michigan, 1995.

[4] J.R. Birge, M.A.H. Dempster, H.I. Gassmann, E.A. Gunn, A.J. King and S.W. Wallace, *A standard input format for multistage stochastic linear programs,* COAL Newsletter 17(1987), 1–19.

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

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

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

[8] H.I. Gassmann, *MSLiP: A computer code for the multistage stochastic linear programming problem,* Math. Prog., 47(1990) 407–423.

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

[10] P. Kall and S.W. Wallace, *Stochastic Programming,* John Wiley & Sons, NY, 1994.

[11] 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.

[12] J. Mayer, *Stochastic Linear Programming Algorithms: A Comparison based on a Model Management System,* Gordon and Breach Science Publishers, 1998.

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

[14] R.J. Peters, K. Boskma and H.A.E. Kuper, *Stochastic programming in production planning: a case with non-simple recourse,* Statistica Neerlandica 31(1977), 113-126.

[15] A. Prékopa, *Stochastic Programming,* Kluwer Academic Publishers, 1995.

[16] S.M. Robinson, *Extended scenario analysis,* Ann. Oper. Res. 31(1991), 385-398.

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

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

[19] A. Ruszczyński, *On convergence of an augmented Lagrangian decomposition method for sparse convex optimization,* Math. Oper. Res. 20(1995), 634-656.

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

[21] K.C. Toh, *Primal-dual path-following algorithms for determinant maximization problems with linear matrix inequalities*, Computational Optim. Appl., 14(1999), 309-330.

[22] G. Zhao, *Interior-point methods with decomposition for solving large-scale linear programs,* JOTA, 102(1999), 169-192.

[23] G. Zhao, *A log-barrier method with Benders decomposition for solving two-stage stochastic programs.* Research Report No. 711, Dept of Mathematics, National University of Singapore, Singapore, 1997, revised 1999. To appear in *Mathematical Programming.*

[24] G. Zhao, *Barrier functions in the Lagrangian dual method for solving multistage stochastic nonlinear programs,* 1998.