A Primal-Dual Decomposition Algorithm for Multistage Stochastic Convex Programming

Arjan Berkelaar[*], Joaquim Gromicho[†], Roy Kouwenberg[‡], Shuzhong Zhang[§]

December 2000; revised July 2003

Abstract

This paper presents a new and high performance solution method for multistage stochastic convex programming. Stochastic programming is a quantitative tool developed in the field of optimization to cope with the problem of decision-making under uncertainty. Among others, stochastic programming has found many applications in finance, such as asset-liability and bond-portfolio management. However, many stochastic programming applications still remain computationally intractable because of their overwhelming dimensionality. In this paper we propose a new decomposition algorithm for multistage stochastic programming with a convex objective and stochastic recourse matrices, based on the path-following interior point method combined with the homogeneous self-dual embedding technique. Our preliminary numerical experiments show that this approach is very promising in many ways for solving generic multistage stochastic programming, including its superiority in terms of numerical efficiency, as well as the flexibility in testing and analyzing the model.

Keywords: multistage stochastic programming, convex objective, interior point method, homogeneous self-dual embedding.

[*]Econometric Institute, Erasmus University Rotterdam, The Netherlands.

[†]Vrij Universiteit, Amsterdam & ORTEC International, Gouda, The Netherlands.

[‡]Faculty of Commerce, University of British Columbia, Canada.

[§]Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong.

1 Introduction

Undoubtedly, stochastic programming is an extremely useful tool in decision making processes, as uncertainty is pervasive in all aspects of human activities. For instance, stochastic programming plays an increasingly important role in financial optimization models such as asset-liability and bond-portfolio management; in this particular application, one is referred to the recent book on asset liability management edited by Mulvey and Ziemba [26]. However, efficiently solving large-scale stochastic programming problems still remains a major challenge; see [11] and [21] for an introduction to stochastic programming. The main difficulty is caused by the nature of the stochastic programming model — its overwhelming dimensionality. This happens when one models the uncertainty at different decision stages with discrete-valued random variables (representing different future *scenarios*) and then takes into account all possible scenarios and their impacts, resulting in, typically, huge size deterministic-equivalent optimization problems. A successful solution method for stochastic programming should exploit the special structure of the problem in order to cut down computational times. For this purpose, most of the solution methods in the area are based on specialized decomposition; we refer to [11] and the references therein for a survey along this direction. A classical approach in that case is the so-called L-shaped method [29] and its multistage extension [8], which are variants of the Benders decomposition. Recently, multistage (linear) stochastic programs with millions of variables and constraints have been solved with parallel implementations of Benders decomposition [9, 16, 18]. Variants of Benders decomposition to solve problems with nonlinear structures can be found in Birge and Rosa [13]. There are relatively much fewer papers dealing with large scale stochastic programming problems with a general convex objective function. These include the augmented Lagrangian approach applied by Berger, Mulvey and Ruszczynski in [7] (see also [25]), a tree-dissection technique in the Cholesky decomposition proposed by Berger, Mulvey, Rothberg and Vanderbei in [6] (we shall discuss more on this method later), and a specialized interior-point decomposition method based on the barrier function proposed by Zhao [33]. In this paper we propose a new decomposition algorithm for multistage convex stochastic programming, based on the path-following interior point method.

The rapid developments in designing and implementing interior point methods (abbreviated as IPM hereafter) in recent years have led to a major breakthrough in the field of optimization (cf. [28] for various survey articles on interior point methods). A characteristic property of IPMs is that they typically require only a small number (say, less than fifty) of iterative steps to reach a high precision solution, almost regardless of the size of the problem. This property is certainly of critical importance in solving large scale stochastic programming problems. Moreover, the IPMs are suitable to handle *nonlinear* problems. In [12] Birge and Qi showed how decomposition can be achieved based on Karmarkar's original interior point method for two-stage stochastic linear programming. Within the interior point method realm, in fact, two types of decomposition methods have appeared. The first type, including [12], exploits the structure of two-stage stochastic linear programming which

is viewed as large size linear programming; see [23, 10, 14] for serial algorithms and [20, 31, 15] for parallel implementations. The second type of interior point decomposition methods typically specializes some of the interior point methodology, such as cutting planes or barrier methods, to solve stochastic programming problems; see e.g. [5, 3, 33, 34]. However, the key difficulty for the first type of IPMs in this context is that in each iteration it usually involves solving a very large, perhaps even ill-conditioned, direction-finding linear system.

In this paper we propose a new decomposition scheme for multistage stochastic convex programming. The new method is of the first type, i.e., it is based on the homogeneous self-dual interior point method, and exploits the problem structure. The crux of our approach is to break this direction-finding subproblem into a sequence of decomposed steps, based on a complete exploitation of the structure of the problem, thereby enabling efficient implementations including parallel computations. Another important feature of our approach is its generality: it allows for a nonlinear objective and a multistage model. Current applications of interior point decomposition are mostly limited to two-stage linear stochastic programs (see [12, 23, 10, 14, 20, 31, 15]). As far as we know there are not many efficient solution methods available for convex multistage stochastic programming problems, apart from the DQA-algorithm applied in [7] by Berger, Mulvey, and Ruszczynski, and the tree-dissection method proposed in [6] by Berger, Mulvey, Rothberg and Vanderbei. The method in [6] is also based on a general interior point approach and pays a great deal of attention to an efficient Cholesky decomposition for solving the search direction, taking advantage of the problem structure. In that respect our paper and [6] share a common feature. However, the underlying ideas are very different. The most distinctive difference is that our algorithm attempts to directly solve the search directions for all the variables, instead of factorizing the normal equations.

This paper is organized as follows. In Section 2 we shall introduce the *multistage stochastic convex programming* (MSSCP) problem, which is to be solved in this paper. Section 3 introduces a key technique, viz. homogeneous self-dual embedding, to be employed in our approach. In Section 4, we discuss how to solve the Newton equation resulting from the homogeneous self-dual embedded system for MSSCP. A scheme leading to a complete decomposition is presented in that section. In Section 5 we shall present some numerical results for the new decomposition method. For references, we compare our method with LOQO — a state-of-the-art commercial optimization solver, on some standard test problems as well as some randomly generated (but structured) instances. The results are indeed encouraging. In case the MSSCP problem does not have an optimal solution, we discuss in Section 6 how to use the additional Farkas type information, which is available in the self-dual embedding framework, to further diagnose the model. Finally, we conclude the paper in Section 7.

We use the following notations in this paper. The gradient of a function $f(x)$, $\nabla f(x)$, is a column vector; its Hessian matrix is denoted by $\nabla^2 f(x)$. The superscript ^{T} denotes the transpose of a matrix or a vector. We use e to denote the all-one vector with appropriate dimension depending on

the context. For a vector, say x, the corresponding capitalized letter X denotes a diagonal matrix with the components in x being the diagonal elements of X.

2 Multistage Stochastic Programming Formulation

We consider the situation where a decision-maker can take decisions (controls) at discrete dates $0, 1, \ldots, K$. The uncertainty in the model is represented by the set of possible states of the world, denoted by Ω, with generic elements (atoms) denoted by ω. The set Ω belongs to a probability space $(\Omega, \mathcal{F}, P)$, where $\mathcal{F}$ is the σ-field of subsets of Ω and P a probability measure. The set Ω with σ-field $\mathcal{F}$, $(\Omega, \mathcal{F})$ is called a measurable space. The probability measure P on $(\Omega, \mathcal{F})$ is a measure such that $P(\Omega) = 1$. The support of $(\Omega, \mathcal{F}, P)$ is the smallest closed subset of Ω with probability 1.

Furthermore, each decision at time t should be determined by information available up to time $t-1$ only. This enforces an informational constraint (so-called non-anticipativity constraint). To model this informational constraint we assume that the information structure consists of a given family of increasingly finer partitions of Ω, denoted by $\{\mathcal{F}_t,\ t = 0, 1, \ldots, K\}$. For every t, the partition $\mathcal{F}_t$ contains all available information to investors at time t. Non-anticipativity of the trading strategy means that $x_t(\varpi)$ should be $\mathcal{F}_{t-1}$ measurable. We assume that $\mathcal{F}_0 = \Omega$ and $\mathcal{F}_K = \{\{\omega\} : \omega \in \Omega\}$. These assumptions imply that:

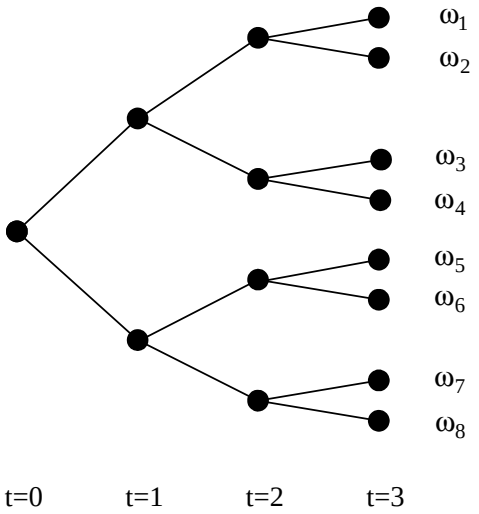
- no information has arrived at time 0;
- as time progresses, decision-makers gradually come to know about decreasing subsets of Ω which contain the true state of the world;
- the true state of the world is fully revealed at the horizon K.

The stochastic optimal control problem with linear constraints and a linear law of motion in this discrete-time setting is given by

$$
\begin{array}{lllll}
\min & f_0(x_0) & & +\sum_{t=1}^{K} E f_t(x_t(\varpi), \varpi) & \\
\text{s.t.} & W_0 x_0 & & & = h_0 \\
 & & B_t(\varpi) x_{t-1}(\varpi) & +W_t(\varpi) x_t(\varpi) & = h_t(\varpi),\ t = 1, ..., K,\ \varpi \in \mathcal{F}_t \\
 & x_0 \geq 0, & & x_t(\varpi) \geq 0,\ \varpi \in \mathcal{F}_t,\ t = 1, ..., K. &
\end{array}
$$

There are two methods for solving discrete-time stochastic optimal control problems: dynamic programming and stochastic programming. In this paper we are concerned with the second approach, namely stochastic programming, which is mainly a method for solving stochastic optimal control

Figure 1: Event Tree



Event tree with $\Omega = \{\omega_1, \ldots, \omega_8\}$ and $\mathcal{F}_0 = \Omega$, $\mathcal{F}_1 = \{\{\omega_1, \omega_2, \omega_3, \omega_4\}, \{\omega_5, \omega_6, \omega_7, \omega_8\}\}$, $\mathcal{F}_2 = \{\{\omega_1, \omega_2\}, \{\omega_3, \omega_4\}, \{\omega_5, \omega_6\}, \{\omega_7, \omega_8\}\}$, $\mathcal{F}_3 = \{\{\omega\} : \omega \in \Omega\}$.

problems by discretizing the random variables and representing these variables by scenarios. Assuming that Ω has a finite support, we can model the information structure outlined above with an *event tree* (or *scenario tree*). We index the nodes of this event tree (which correspond to ϖ) by $n \in \mathcal{F}_t$ for $t = 1, \ldots, K$. Figure 1 provides an example of an event tree. Variables defined on Ω are denoted by subscripts n and t, where $n \in \mathcal{F}_t$; by definition these variables are $\mathcal{F}_{t-1}$ measurable. In mathematical form the multistage (K-stage) stochastic programming model is:

$$
\begin{array}{lllll}
\min & f_0(x_0) & & +\sum_{t=1}^{K} \sum_{n \in \mathcal{F}_t} \pi_{nt} f_{nt}(x_{nt}) & \\
\text{s.t.} & W_0 x_0 & & & = h_0 \\
& & B_{nt} x_{a(n),t-1} & +W_{nt} x_{nt} & = h_{nt},\ t = 1, ..., K,\ n \in \mathcal{F}_t \\
& x_0 \geq 0, & & x_{nt} \geq 0,\ n \in \mathcal{F}_t,\ t = 1, ..., K. &
\end{array}
$$

where we assume that $f_0(x_0)$ and $f_{nt}(x_{nt})$, $t = 1, \ldots, K$, $n \in \mathcal{F}_t$ are twice differentiable and convex, $\{\mathcal{F}_t : t = 1, ..., K\}$ is a filtration with finite support, subscript (n, t) stands for scenario n at stage t, π_{nt} is its probability, $(a(n), t-1)$ is the ancestor of (n, t), and $(C(n), t+1)$ will be denoted the set of children of (n, t). As a convention we let $n = 0$ denote the current state. Therefore, the decision variables are x_0 (the immediate action) and x_{nt} (the recourse action to be taken if scenario n unfolds at stage t).

In our notation, each node on the scenario tree (at different stages) is associated with a unique n.

In this sense, the association t in the notation (n,t) is redundant. Nevertheless, we shall keep it in places where confusion is possible.

In general, the above formulation, which is also known as the *deterministic equivalent problem*, can be a large size convex program. The number of variables is of the order $O(C^K)$ where C represents the number of children following each scenario at an intermediate stage. For practical purposes we may assume that each of the matrices B_{nt}, and W_{nt} are reasonably sized. For technical reasons we also assume that all the matrices W_{nt} have full row ranks.

This problem has an associated dual problem, known as the Wolfe dual, which is the following

$$\begin{aligned}
\max \quad & f_0(x_0) - x_0^T \nabla f_0(x_0) + h_0^T y_0 + \sum_{t=1}^{K} \sum_{n \in \mathcal{F}_t} h_{nt}^T y_{nt} + \sum_{t=1}^{K} \sum_{n \in \mathcal{F}_t} \pi_{nt} \left(f_{nt}(x_{nt}) - x_{nt}^T \nabla f_{nt}(x_{nt}) \right) \\
\text{s.t.} \quad & W_0^T y_0 + \sum_{m \in C(0)} B_{m1}^T y_{m1} + s_0 = \nabla f_0(x_0) \\
& W_{nt}^T y_{nt} + \sum_{m \in C(n)} B_{m,t+1}^T y_{m,t+1} + s_{nt} = \pi_{nt} \nabla f_{nt}(x_{nt}),\ t = 1, ..., K-1,\ n \in \mathcal{F}_t \\
& W_{nK}^T y_{nK} + s_{nK} = \pi_{nK} \nabla f_{nK}(x_{nK}),\ n \in \mathcal{F}_K \\
& x_0 \geq 0,\ x_{nt} \geq 0,\ s_0 \geq 0,\ s_{nt} \geq 0,\ n \in \mathcal{F}_t,\ t = 1, ..., K.
\end{aligned}$$

The duality structure will be utilized in our approach. In fact, our computational approach relies on the homogeneous self-dual (HSD) method originally introduced by Xu, Hung and Ye [30] as a simplification of the self-dual embedding technique of Ye, Todd and Mizuno [32] for linear programming. This method was applied by Berkelaar, Dert, Oldenkamp, and Zhang [4] for solving two-stage stochastic linear programming problems. The crucial observation made by Berkelaar, Dert, Oldenkamp, and Zhang in [4] is that it is possible to completely decompose the direction-finding problem into subproblems, therefore enabling a decomposition-based implementation of the HSD technique. The aim here is to show that this result also holds for multistage problems with general convex objective functions as introduced above. Berkelaar, Dert, Oldenkamp, and Zhang [4] report numerical results for some stochastic linear programming problem that unambiguously show the speed-up attained when applying the decomposition algorithm compared to solving the deterministic equivalent directly by the HSD method. To put the picture in perspective we first introduce the idea of homogeneous self-dual embedding.

3 Homogeneous Self-Dual Embedding

The homogeneous self-dual method (HSD) for linear programming was proposed by Xu, Hung and Ye [30] as a simplification of the self-dual embedding technique of Ye, Todd and Mizuno [32]. This technique proves to be very efficient in solving linear programs (a refined version of the HSD method is actually implemented by Andersen and Andersen [1] in an optimization package called MOSEK). The

HSD method is also coded beyond linear programming. In fact, it is implemented for a much broader class of optimization problems known as *semidefinite programming*; see [22] and [27]. Furthermore, the HSD method is extended to a general convex programming framework by Andersen and Ye; see [2]. One of the advantages of the HSD method is that it requires no feasibility phase, allowing one to freely select any interior starting point (possibly infeasible). Moreover, the method is capable of detecting infeasibility which can be of great importance for stochastic programs. As a general merit of interior point methods, the number of iterations required to solve a linear program is typically low and insensitive to the dimension of the problem. This is an important property for solving large-scale stochastic programs. The main concern is how to implement each step of an interior point method efficiently. We observe that it is possible to completely decompose the direction-finding problem into subproblems, thereby enabling a decomposition-based implementation of the HSD technique. In a sense, our efforts can also be seen as trying to exploit the sparsity structure of the constrained matrix. However, unlike other general-purposed sparse matrix techniques in linear algebra, we make full use of the structure and the decomposition is tailor-made for this class of problems only. This issue will be addressed in detail in Section 4.

Consider the following convex programming problem with linear constraints:

$$\begin{array}{lll} (P) & \min & f(x) \\ & \text{s.t.} & Ax = b \\ & & x \geq 0. \end{array}$$

The Wolfe dual to the above problem is given by:

$$\begin{array}{lll} (D) & \max & f(x) - x^T \nabla f(x) + b^T y \\ & \text{s.t.} & A^T y + s = \nabla f(x) \\ & & x, s \geq 0. \end{array}$$

Any optimal solution to both (P) and (D) should solve the primal and dual feasibility conditions and attain a zero duality gap:

$$(O) \begin{cases} Ax = b,\ x \geq 0 \\ A^T y + s = \nabla f(x),\ s \geq 0 \\ b^T y - x^T \nabla f(x) = 0 \end{cases}$$

In recent years, interior point methods have received intensive research in the area of optimization. This research has demonstrated that an efficient implementation of interior point methods should properly combine the primal and the dual information. The crux behind primal-dual interior point methods is to linearize the system (O) and apply Newton's method.

The so-called homogeneous self-dual embedding technique was first proposed by Ye, Todd and Mizuno [32] for linear programming. A simplification was proposed by Xu, Hung and Ye [30]. Andersen and Ye [2] extended the method to convex programming. The main idea of this HSD technique

is to consider the following homogenized version of (O):

$$(H)\begin{cases} & Ax & & & -b\tau & = & 0 \\ -A^Ty & & -s & +\tau\nabla f(x/\tau) & & = & 0 \\ b^Ty & -x^T\nabla f(x/\tau) & & & -\kappa & = & 0 \\ & x\geq 0, & s\geq 0, & \tau\geq 0, & \kappa\geq 0. & & \end{cases}$$

If system (H) has a solution $(y^*, x^*, s^*, \tau^*, \kappa^*)$ such that $\tau^* > 0$ and $\kappa^* = 0$, then an optimal solution to (P) is simply x^*/τ^* and an optimal solution to (D) is $(y^*/\tau^*, s^*/\tau^*, x^*/\tau^*)$.

It is elementary to check that any solution (y, x, s, τ, κ) to (H) necessarily satisfies

$$x^Ts + \tau\kappa = 0,$$

or equivalently, $x_is_i = 0$ for all i, and $\tau\kappa = 0$.

Consider an arbitrary vector (y, x, s, τ, κ) with $x > 0$, $s > 0$, $\tau > 0$ and $\kappa > 0$. The homogeneous self-dual algorithm ([2]) applies a Newton step towards a solution of (H), i.e., we try to find $(d_y, d_x, d_s, d_\tau, d_\kappa)$ such that $(y + d_y, x + d_x, s + d_s, \tau + d_\tau, \kappa + d_\kappa)$ satisfies a perturbed and parameterized version of (H), known as the *analytic central path*. Finally we linearize the resulting nonlinear equation, yielding a linear system known as Newton's equation. In our particular case, the Newton system for the direction $(d_y, d_x, d_s, d_\tau, d_\kappa)$ is

$$(S)\begin{cases} & Ad_x & & -bd_\tau & & = & \eta r_P \\ -A^Td_y & +F_0d_x & -d_s & +F_1d_\tau & & = & \eta r_D \\ b^Td_y & -(\bar{F}_1)^Td_x & & +F_2d_\tau & -d_\kappa & = & \eta r_g \\ & Sd_x & +Xd_s & & & = & \gamma\mu e - Xs \\ & & & \kappa d_\tau & +\tau d_\kappa & = & \gamma\mu - \tau\kappa \end{cases}$$

where

$$r_P = \tau b - Ax,\ r_D = -\tau\nabla f(x/\tau) + A^Ty + s \text{ and } r_g = \kappa + x^T\nabla f(x/\tau) - b^Ty$$

are the feasibility residuals, and furthermore, we used the notation

$$F_0 = \nabla^2 f(x/\tau),\ F_1 = \nabla f(x/\tau) - \nabla^2 f(x/\tau)(x/\tau)$$

and

$$\bar{F}_1 = \nabla f(x/\tau) + \nabla^2 f(x/\tau)(x/\tau),\ F_2 = (x/\tau)^T\nabla^2 f(x/\tau)(x/\tau).$$

In the above equations, η and γ are two parameters, and $\mu = (x^Ts + \tau\kappa)/(n + 1)$.

The generic homogeneous self-dual algorithm works as follows. Suppose that we have an iterate $(y^k, x^k, s^k, \tau^k, \kappa^k)$ with $x^k > 0$, $s^k > 0$, $\tau^k > 0$ and $\kappa^k > 0$. Let

$$(y, x, s, \tau, \kappa) := (y^k, x^k, s^k, \tau^k, \kappa^k)$$

and let $\eta \in [0,1]$ and $\gamma \in [0,1]$. We first solve the system (S) to get search directions $(d_y, d_x, d_s, d_\tau, d_\kappa)$. Then, we choose a step-length $\alpha > 0$ such that

$$\begin{aligned} y' &= y + \alpha d_y \\ x' &= x + \alpha d_x > 0 \\ s' &= s + \alpha d_s > 0 \\ \tau' &= \tau + \alpha d_\tau > 0 \\ \kappa' &= \kappa + \alpha d_\kappa > 0. \end{aligned}$$

Let

$$(y^{k+1}, x^{k+1}, s^{k+1}, \tau^{k+1}, \kappa^{k+1}) := (y', x', s', \tau', \kappa')$$

and $k := k + 1$. Repeat the procedure until a given precision is reached.

The step-length $\alpha > 0$ should be chosen in a careful way. We first define a merit function as follows

$$\chi(y, x, s, \tau, \kappa) = \theta(x^T s + \tau\kappa) + \|r(y, x, s, \tau, \kappa)\|,$$

where θ is a parameter, and

$$r(y, x, s, \tau, \kappa) = \begin{pmatrix} r_P \\ r_D \\ r_g \end{pmatrix}$$

is the vector of residuals, and the norm can be either Euclidean or L_∞.

At each iteration the step-length is chosen such that the following conditions are satisfied:

$$\frac{(x^k)^T s^k + \tau^k \kappa^k}{(x^0)^T s^0 + \tau^0 \kappa^0} \geq \beta_1 \frac{\|r(y^k, x^k, s^k, \tau^k, \kappa^k)\|}{\|r(y^0, x^0, s^0, \tau^0, \kappa^0)\|}$$

where β_1 is some parameter and where $(y^0, x^0, s^0, \tau^0, \kappa^0)$ denotes the initial starting point. This condition prevents the iterates from converging towards a complementarity solution faster than attaining feasibility.

The second condition

$$\min\{X^k s^k, \tau^k \kappa^k\} \geq \beta_2 \mu$$

concerns the centrality; it prevents the iterates from converging towards the boundary of the positive orthant prematurely. The third condition:

$$\begin{aligned} &\chi(y^{k+1}, x^{k+1}, s^{k+1}, \tau^{k+1}, \kappa^{k+1}) \\ &\quad \leq \quad \chi(y^k, x^k, s^k, \tau^k, \kappa^k) + \alpha\beta_3(d_y^T,\ d_x^T,\ d_s^T,\ d_\tau,\ d_\kappa)\nabla\chi(y^k, x^k, s^k, \tau^k, \kappa^k) \end{aligned}$$

ensures that the merit function is reduced at each iteration.

4 Finding Search Directions

As we have mentioned at the end of Section 2, it is crucial to be able to efficiently compute the search directions, based on the linear system of equations (S), when the homogeneous self-dual embedding technique is applied to the deterministic-equivalent of multistage stochastic programming problem. To be precise, the linear system of concern is

$$
(\bar{S})\quad \left\{\begin{array}{lll}
W_0 d_{x_0} & -h_0 d_\tau & = \eta r_{P_0} \\
B_{nt} d_{x_{a(n),t-1}} + W_{nt} d_{x_{nt}} & -h_{nt} d_\tau & = \eta r_{P_{nt}}, \\
 & & t = 1, ..., K,\ n \in \mathcal{F}_t \\
-W_{nt}^T d_{y_{nt}} - \sum\limits_{m \in C(n)} B_{m,t+1}^T d_{y_{m,t+1}} + F_0^{nt} d_{x_{nt}} - d_{s_{nt}} & +F_1^{nt} d_\tau & = \eta r_{D_{nt}}, \\
 & & t = 0, ..., K-1,\ n \in \mathcal{F}_t \\
-W_{nK}^T d_{y_{nK}} + F_0^{nK} d_{x_{nK}} - d_{s_{nK}} & +F_1^{nK} d_\tau & = \eta r_{D_{nK}},\ n \in \mathcal{F}_K \\
S_0 d_{x_0} + X_0 d_{s_0} & & = \gamma\mu e - X_0 s_0 \\
S_{nt} d_{x_{nt}} + X_{nt} d_{s_{nt}} & & = \gamma\mu e - X_{nt} s_{nt}, \\
 & & t = 1, ..., K,\ n \in \mathcal{F}_t \\
 & \kappa d_\tau + \tau d_\kappa & = \gamma\mu - \tau\kappa \\
\sum\limits_{t=0}^{K} \sum\limits_{n \in \mathcal{F}_t} \left[h_{nt}^T d_{y_{nt}} - (\bar{F}_1^{nt})^T d_{x_{nt}} \right] & +F_2 d_\tau - d_\kappa & = \eta r_g
\end{array}\right.
$$

where

$$
\begin{array}{lll}
r_{P_0} & = & \tau h_0 - W_0 x_0 \\
r_{P_{nt}} & = & \tau h_{nt} - B_{nt} x_{a(n),t-1} - W_{nt} x_{nt},\ t = 1, ..., K,\ n \in \mathcal{F}_t \\
r_{D_{nt}} & = & -\tau \pi_{nt} \nabla f_{nt}(x_{nt}/\tau) + W_{nt}^T y_{nt} + \sum\limits_{m \in C(n)} B_{m,t+1}^T y_{m,t+1} + s_{nt},\ t = 0, ..., K-1,\ n \in \mathcal{F}_t \\
r_{D_{nK}} & = & -\tau \pi_{nK} \nabla f_{nK}(x_{nK}/\tau) + W_{nK}^T y_{nK} + s_{nK},\ n \in \mathcal{F}_K \\
r_g & = & \kappa + \sum\limits_{t=0}^{K} \sum\limits_{n \in \mathcal{F}_t} \left[\pi_{nt} x_{nt}^T \nabla f_{nt}(x_{nt}/\tau) - h_{nt}^T y_{nt} \right],
\end{array}
$$

with the convention that $\pi_{00} \equiv 1$ and $x_{00} \equiv x_0$. Moreover,

$$\begin{aligned}
F_0^{nt} &= \pi_{nt}\nabla^2 f_{nt}(x_{nt}/\tau),\ t = 0, ..., K,\ n \in \mathcal{F}_t,\\
F_1^{nt} &= \pi_{nt}\left[\nabla f_{nt}(x_{nt}/\tau) - \nabla^2 f_{nt}(x_{nt}/\tau)(x_{nt}/\tau)\right],\ t = 0, ..., K,\ n \in \mathcal{F}_t,\\
\bar{F}_1^{nt} &= \pi_{nt}\left[\nabla f_{nt}(x_{nt}/\tau) + \nabla^2 f_{nt}(x_{nt}/\tau)(x_{nt}/\tau)\right],\ t = 0, ..., K,\ n \in \mathcal{F}_t,\\
F_2 &= \sum_{t=0}^{K}\sum_{n\in\mathcal{F}_t} \pi_{nt}(x_{nt}/\tau)^T\nabla^2 f_{nt}(x_{nt}/\tau)(x_{nt}/\tau).
\end{aligned}$$

Consider a given $m \in \mathcal{F}_K$. From the sixth equation in $(\bar{S})$ we obtain

$$d_{s_{mK}} = -X_{mK}^{-1}S_{mK}d_{x_{mK}} + X_{mK}^{-1}(\gamma\mu e - X_{mK}^{-1}s_{mK}). \tag{1}$$

Substituting (1) into the fourth equation in $(\bar{S})$ yields

$$-W_{mK}^T d_{y_{mK}} + (F_0^{mK} + X_{mK}^{-1}S_{mK})d_{x_{mK}} - X_{mK}^{-1}(\gamma\mu e - X_{mK}^{-1}s_{mK}) + F_1^{mK}d_\tau = \eta r_{D_{mK}},$$

and therefore

$$\begin{aligned}
d_{x_{mK}} &= (F_0^{mK} + X_{mK}^{-1}S_{mK})^{-1}W_{mK}^T d_{y_{mK}} - (F_0^{mK} + X_{mK}^{-1}S_{mK})^{-1}F_1^{mK}d_\tau\\
&\quad +(F_0^{mK} + X_{mK}^{-1}S_{mK})^{-1}\left[\eta r_{D_{mK}} + X_{mK}^{-1}(\gamma\mu e - X_{mK}^{-1}s_{mK})\right].
\end{aligned}$$

Let

$$M_{mK} := F_0^{mK} + X_{mK}^{-1}S_{mK}. \tag{2}$$

We have

$$d_{x_{mK}} = M_{mK}^{-1}W_{mK}^T d_{y_{mK}} - M_{mK}^{-1}F_1^{mK}d_\tau + M_{mK}^{-1}\left[\eta r_{D_{mK}} + X_{mK}^{-1}(\gamma\mu e - X_{mK}^{-1}s_{mK})\right]. \tag{3}$$

Now consider an $n \in \mathcal{F}_{K-1}$ such that $m \in C(n)$, i.e., $n = a(m)$. Using (3) and the second equation in $(\bar{S})$ we get

$$B_{mK}d_{x_{n,K-1}} + W_{mK}M_{mK}^{-1}W_{mK}^T d_{y_{mK}} - W_{mK}M_{mK}^{-1}W_{mK}^T q_{mK}d_\tau = W_{mK}M_{mK}^{-1}W_{mK}^T v_{mK}$$

for all $m \in C(n)$, and consequently

$$d_{y_{mK}} = -(W_{mK}M_{mK}^{-1}W_{mK}^T)^{-1}B_{mK}d_{x_{n,K-1}} + q_{mK}d_\tau + v_{mK}, \tag{4}$$

where

$$q_{mK} := (W_{mK}M_{mK}^{-1}W_{mK}^T)^{-1}\left[h_{mK} + W_{mK}M_{mK}^{-1}F_1^{mK}\right] \tag{5}$$

and

$$v_{mK} := (W_{mK}M_{mK}^{-1}W_{mK}^T)^{-1}\left\{\eta r_{P_{mK}} - W_{mK}M_{mK}^{-1}\left[\eta r_{d_{mK}} + X_{mK}^{-1}(\gamma\mu e - X_{mK}s_{mK})\right]\right\}. \tag{6}$$

Now we substitute (4) back into the third equation in $(\bar{S})$. Further using the sixth equation in $(\bar{S})$, this yields

$$-W_{n,K-1}^T d_{y_{n,K-1}} + \left[F_0^{n,K-1} + X_{n,K-1}^{-1} S_{n,K-1} + \sum_{m \in C(n)} B_{mK}^T (W_{mK} M_{mK}^{-1} W_{mK}^T)^{-1} B_{mK} \right] d_{x_{n,K-1}}$$
$$+ \left[F_1^{n,K-1} - \sum_{m \in C(n)} B_{mK}^T q_{mK} \right] d_\tau$$
$$= \eta r_{D_{n,K-1}} + \sum_{m \in C(n)} B_{mK}^T v_{mK} + X_{n,K-1}^{-1} (\gamma \mu e - X_{n,K-1} s_{n,K-1}).$$

Letting

$$M_{n,K-1} := F_0^{n,K-1} + X_{n,K-1}^{-1} S_{n,K-1} + \sum_{m \in C(n)} B_{mK}^T (W_{mK} M_{mK}^{-1} W_{mK}^T)^{-1} B_{mK} \tag{7}$$

we may rewrite the above expression as

$$\begin{aligned} d_{x_{n,K-1}} &= M_{n,K-1}^{-1} W_{n,K-1}^T d_{y_{n,K-1}} - M_{n,K-1}^{-1} \left[F_1^{n,K-1} - \sum_{m \in C(n)} B_{mK}^T q_{mK} \right] d_\tau + \\ & \quad M_{n,K-1}^{-1} \left[\eta r_{D_{n,K-1}} + \sum_{m \in C(n)} B_{mK}^T v_{mK} + X_{n,K-1}^{-1} (\gamma \mu e - X_{n,K-1} s_{n,K-1}) \right]. \end{aligned} \tag{8}$$

We trace one more stage back, and let k be at stage $K-2$ such that $n \in C(k)$, i.e., $k = a(n)$. Similar as before, we substitute (8) into the second equation in $(\bar{S})$, which is

$$B_{n,K-1} d_{x_{k,K-2}} + W_{n,K-1} d_{x_{n,K-1}} - h_{n,K-1} d_\tau = \eta r_{P_{n,K-1}},$$

obtaining

$$d_{y_{n,K-1}} = -(W_{n,K-1} M_{n,K-1}^{-1} W_{n,K-1}^T)^{-1} B_{n,K-1} d_{x_{k,K-2}} + q_{n,K-1} d_\tau + v_{n,K-1}, \tag{9}$$

where

$$q_{n,K-1} := (W_{n,K-1} M_{n,K-1}^{-1} W_{n,K-1}^T)^{-1} \left\{ h_{n,K-1} + W_{n,K-1} M_{n,K-1}^{-1} \left[F_1^{n,K-1} - \sum_{m \in C(n)} B_{mK}^T q_{mK} \right] \right\} \tag{10}$$

and

$$v_{n,K-1} := (W_{n,K-1} M_{n,K-1}^{-1} W_{n,K-1}^T)^{-1} \cdot$$
$$\left\{ \eta r_{P_{n,K-1}} - W_{n,K-1} M_{n,K-1}^{-1} \left[\eta r_{D_{n,K-1}} + X_{n,K-1}^{-1} (\gamma \mu e - X_{n,K-1} s_{n,K-1}) + \sum_{m \in C(n)} B_{mK}^T v_{mK} \right] \right\}. \tag{11}$$

Once again, we substitute (9) back into the third equation in $(\bar{S})$ and use the sixth equation at the same time. This yields an equation from which we can solve for $d_{x_{k,K-2}}$ as follows

$$\begin{aligned} d_{x_{k,K-2}} &= M_{k,K-2}^{-1} W_{k,K-2}^T d_{y_{k,K-2}} - M_{k,K-2}^{-1} \left[F_1^{k,K-2} - \sum_{n \in C(k)} B_{k,K-1}^T q_{k,K-1} \right] d_\tau + \\ & \quad M_{k,K-2}^{-1} \left[\eta r_{D_{k,K-2}} + \sum_{n \in C(k)} B_{n,K-1}^T v_{n,K-1} + X_{k,K-2}^{-1} (\gamma \mu e - X_{k,K-2} s_{k,K-2}) \right]. \quad (12) \end{aligned}$$

Repeating this procedure, we get the following recursive formula

$$d_{y_{mt}} = -(W_{mt} M_{mt}^{-1} W_{mt}^T)^{-1} B_{mt} d_{x_{n,t-1}} + q_{mt} d_\tau + v_{mt}, \quad (13)$$

where $n = a(m)$, and

$$\begin{aligned} d_{x_{mt}} &= M_{mt}^{-1} W_{mt}^T d_{y_{mt}} - M_{mt}^{-1} \left[F_1^{mt} - \sum_{k \in C(m)} B_{k,t+1}^T q_{k,t+1} \right] d_\tau + \\ & \quad M_{mt}^{-1} \left[\eta r_{d_{mt}} + \sum_{k \in C(m)} B_{k,t+1}^T v_{k,t+1} + X_{mt}^{-1} (\gamma \mu e - X_{mt} s_{mt}) \right], \quad (14) \end{aligned}$$

for $t = K, K-1, ..., 1$ and $m \in \mathcal{F}_t$, where the convention is that $C(m) = \emptyset$ whenever $t = K$ and $m \in \mathcal{F}_t$. The matrices M_{mt} and vectors q_{mt} and v_{mt} are generated according to the following recursive formula

$$(R_1) \begin{cases} M_{n,t-1} &= F_0^{n,t-1} + X_{n,t-1}^{-1} S_{n,t-1} + \sum_{m \in C(n)} B_{mt}^T (W_{mt} M_{mt}^{-1} W_{mt}^T)^{-1} B_{mt} \\ q_{n,t-1} &= (W_{n,t-1} M_{n,t-1}^{-1} W_{n,t-1}^T)^{-1} \left\{ h_{n,t-1} + W_{n,t-1} M_{n,t-1}^{-1} \left[F_1^{n,t-1} - \sum_{m \in C(n)} B_{mt}^T q_{mt} \right] \right\} \\ v_{n,t-1} &= (W_{n,t-1} M_{n,t-1}^{-1} W_{n,t-1}^T)^{-1} \cdot \\ & \quad \left\{ \eta r_{P_{n,t-1}} - W_{n,t-1} M_{n,t-1}^{-1} \left[\eta r_{D_{n,t-1}} + X_{n,t-1}^{-1} (\gamma \mu e - X_{n,t-1} s_{n,t-1}) + \sum_{m \in C(n)} B_{mt}^T v_{mt} \right] \right\} \end{cases}$$

with $m \in C(n)$ and $m \in \mathcal{F}_t$; see (7), (10) and (11). The terminal values are given by the following formula

$$(T) \begin{cases} M_{mK} &= F_0^{mK} + X_{mK}^{-1} S_{mK} \\ q_{mK} &= (W_{mK} M_{mK}^{-1} W_{mK}^T)^{-1} \left[h_{mK} + W_{mK} M_{mK}^{-1} F_1^{mK} \right] \\ v_{mK} &= (W_{mK} M_{mK}^{-1} W_{mK}^T)^{-1} \left\{ \eta r_{P_{mK}} - W_{mK} M_{mK}^{-1} \left[\eta r_{D_{mK}} + X_{mK}^{-1} (\gamma \mu e - X_{mK} s_{mK}) \right] \right\} \end{cases}$$

where $m \in \mathcal{F}_K$; see (2), (5) and (6).

To further simplify the notation let us rewrite (14) as

$$d_{x_{mt}} = M_{mt}^{-1} W_{mt}^T d_{y_{mt}} - p_{mt} d_\tau + u_{mt} \tag{15}$$

with

$$p_{mt} = M_{mt}^{-1} \left[F_1^{mt} - \sum_{k \in C(m)} B_{k,t+1}^T q_{k,t+1} \right] \tag{16}$$

$$u_{mt} = M_{mt}^{-1} \left[\eta r_{D_{mt}} + \sum_{k \in C(m)} B_{k,t+1}^T v_{k,t+1} + X_{mt}^{-1} (\gamma \mu e - X_{mt} s_{mt}) \right]. \tag{17}$$

In particular we have

$$d_{x_0} = M_0^{-1} W_0^T d_{y_0} - p_0 d_\tau + u_0. \tag{18}$$

Substituting this into the first equation in $(\bar{S})$ we have

$$W_0 (M_0^{-1} W_0^T d_{y_{mt}} - p_0 d_\tau + u_0) - h_0 d_\tau = \eta r_{P_0}.$$

Equivalently,

$$d_{y_0} = (W_0 M_0^{-1} W_0^T)^{-1} (W_0 p_0 + h_0) d_\tau + (W_0 M_0^{-1} W_0^T)^{-1} (\eta r_{P_0} - W_0 u_0).$$

We denote

$$\alpha_0 = (W_0 M_0^{-1} W_0^T)^{-1} (W_0 p_0 + h_0) \tag{19}$$

$$\beta_0 = (W_0 M_0^{-1} W_0^T)^{-1} (\eta r_{P_0} - W_0 u_0) \tag{20}$$

and so

$$d_{y_0} = \alpha_0 d_\tau + \beta_0. \tag{21}$$

Substituting (21) into (18) we get

$$\begin{aligned} d_{x_0} &= M_0^{-1} W_0^T d_{y_0} - p_0 d_\tau + u_0 \\ &= M_0^{-1} W_0^T (\alpha_0 d_\tau + \beta_0) - p_0 d_\tau + u_0 \\ &=: \psi_0 d_\tau + \phi_0 \end{aligned} \tag{22}$$

with

$$\psi_0 = M_0^{-1} W_0^T \alpha_0 - p_0 \tag{23}$$

and

$$\phi_0 = M_0^{-1} W_0^T \beta_0 + u_0. \tag{24}$$

Now we wish to find a general expression for $d_{x_{mt}}$ and $d_{y_{mt}}$ in terms of d_τ. Let us write them as

$$d_{y_{mt}} = \alpha_{mt} d_\tau + \beta_{mt} \tag{25}$$

$$d_{x_{mt}} = \psi_{mt} d_\tau + \phi_{mt} \tag{26}$$

where $m \in \mathcal{F}_t$ and $t = 0, 1, ..., K$.

A forward recursive formula for α_{mt}, β_{mt}, ψ_{mt} and ϕ_{mt} can be found, based on (15) and (13), as follows

$$(R_2) \begin{cases} \alpha_{mt} = -(W_{mt} M_{mt}^{-1} W_{mt}^T)^{-1} B_{mt} \psi_{n,t-1} + q_{mt} \\ \beta_{mt} = -(W_{mt} M_{mt}^{-1} W_{mt}^T)^{-1} B_{mt} \phi_{n,t-1} + v_{mt} \\ \psi_{mt} = M_{mt}^{-1} W_{mt}^T \alpha_{mt} - p_{mt} \\ \phi_{mt} = M_{mt}^{-1} W_{mt}^T \beta_{mt} + u_{mt} \end{cases}$$

where $m \in \mathcal{F}_t$, $n \in \mathcal{F}_{t-1}$ and $n = a(m)$. Putting together (19), (20), (23) and (24), we have the following initial values for the recursion

$$(I) \begin{cases} \alpha_0 = (W_0 M_0^{-1} W_0^T)^{-1} (W_0 p_0 + h_0) \\ \beta_0 = (W_0 M_0^{-1} W_0^T)^{-1} (\eta r_{P_0} - W_0 u_0) \\ \psi_0 = M_0^{-1} W_0^T \alpha_0 - p_0 \\ \phi_0 = M_0^{-1} W_0^T \beta_0 + u_0. \end{cases}$$

Now, we eliminate d_κ using the seventh and the eighth equations in $(\bar{S})$. Then we substitute $d_{x_{nt}}$ and $d_{y_{nt}}$, according to (25) and (26), into the resulting equation. This yields

$$\sum_{t=0}^{K} \sum_{n \in \mathcal{F}_t} \left[h_{nt}^T (\alpha_{nt} d_\tau + \beta_{nt}) - (\bar{F}_1^{nt})^T (\psi_{nt} d_\tau + \phi_{nt}) \right] + F_2 d_\tau - \left(-\tau^{-1} \kappa d_\tau + \tau^{-1} (\gamma \mu - \tau \kappa) \right) = \eta r_g$$

and so

$$d_\tau = \frac{\eta r_g + \tau^{-1}(\gamma \mu - \tau \kappa) - \sum_{t=0}^{K} \sum_{n \in \mathcal{F}_t} \left[h_{nt}^T \beta_{nt} - (\bar{F}_1^{nt})^T \phi_{nt} \right]}{\sum_{t=0}^{K} \sum_{n \in \mathcal{F}_t} \left[h_{nt}^T \alpha_{nt} - (\bar{F}_1^{nt})^T \psi_{nt} \right] + F_2 + \tau^{-1} \kappa}. \tag{27}$$

All other search directions can be solved using (27). In particular, from the seventh equation in $(\bar{S})$ we have

$$d_\kappa = -\tau^{-1} \kappa d_\tau + \tau^{-1} (\gamma \mu - \tau \kappa) \tag{28}$$

and from (25) and (26) we get the value for $d_{y_{mt}}$ and $d_{x_{mt}}$. Furthermore, the value for $d_{s_{mt}}$ can be obtained from the sixth equation in $(\bar{S})$, i.e.

$$d_{s_{mt}} = -X_{mt}^{-1} S_{mt} d_{x_{mt}} + X_{mt}^{-1} \left(\gamma \mu e - X_{mt} s_{mt}\right). \tag{29}$$

To summarize, the search directions implied by the Newton equation $(\bar{S})$ can be solved in the following way. First, we generate the working matrices and vectors M_{mt}, q_{mt}, v_{mt}, p_{mt} and u_{mt} according to the recursive scheme (R_1) with the terminal values determined by (T). This works in a backward manner. Then, we generate another set of working vectors α_{mt}, β_{mt}, ψ_{mt} and ϕ_{mt} using the recursive scheme (R_2) with the initial values given by (I). Finally, the search directions are computed by the formulae (27), (28), (25), (26) and (29). Clearly, the time and space complexity of such a decomposition scheme is linear with respect to the total number of scenarios.

The crux of this computational scheme is to generate M_{mt}, q_{mt}, v_{mt}, p_{mt} and u_{mt}. This, however, can be accelerated if we have a parallel computational environment, as computing $(M_{n,t-1}, q_{n,t-1}, v_{n,t-1}, p_{n,t-1}, u_{n,t-1})$ from $(M_{mt}, q_{mt}, v_{mt}, p_{mt}, u_{mt})$ can be done independent of each other. The same can be said about the computation of α_{mt}, β_{mt}, ψ_{mt} and ϕ_{mt}. In the computation, one actually only needs to store M_{mt}, q_{mt}, v_{mt}. All the other quantities can easily be derived from this information.

5 Numerical Results

In order to evaluate the performance of our proposed method, we made two implementations of it. Neither of them uses parallel computation at this stage.

The first implementation is a straightforward transcription of the procedure stipulated in Section 4 to MatLab 6.

The second implementation is written in C++ using Microsoft Visual Studio 6. This uses no commercially available libraries.

Both implementations were created and executed on a PC with 512 megabytes RAM and an AMD Athlon MP 1533 MHz processor running Windows XP Professional.

The decomposed homogeneous self-dual embedding algorithm is applied on the following sets of problems:

- 2 stage benchmark problems from the SLPT [36] and POSTS [35] test sets.
- Randomly generated multistage linear problems.
- Two sets of randomly generated multistage nonlinear problems.

name	core	time	stoch	first	second	n	size
phone	.cor	.tim	.sto.1	1×9	23×93	2	24×102
chem	.cor	.tim	.sto	38×57	46×71	3	130×199
airl	.cor	.tim	.sto.first	2×6	6×12	26	152×306
airl	.cor	.tim	.sto.second	2×6	6×12	26	152×306
assets	.cor	.tim	.sto.small	5×13	5×13	15	75×195
assets	.cor	.tim	.sto.large	5×13	5×13	111	555×1443
pltexpA2	.cor	.tim	-6.sto	62×188	104×272	7	686×1820
pltexpA2	.cor	.tim	-16.sto	62×188	104×272	17	1726×4540

Table 1: Our selection of benchmark test cases

All the problems considered were feasible and finite.

For the first two classes of problems we generated the explicit deterministic equivalent and solved using a trial version of LOQO [37].

All problems where defined in MatLab 6. For the test problems, available in SMPS [38], we wrote a SMPS parser in MatLab 6. This parser, in its present version, can only handle 2 stage problems whose STOCH files use BLOCKS or INDEP sections; see [38] for details. Moreover, we cannot handle RANGES or BOUNDED variables yet. This constrained our choice of test problems to those described in Table 1.

The first group of four columns of Table 1 identify the problem by its name and the extension of the corresponding CORE, TIME and STOCH files.

The second group of columns describes the problem size and structure by giving the dimensions of the node problem in the first and second stages and the number of nodes. Finally, it also lists the size of the corresponding deterministic equivalent.

Each of these was solved by both implementations and the corresponding deterministic equivalent was solved by LOQO. The results obtained can be found in Table 2.

This table is divided into three sets of columns, listing respectively the solution found and the time taken by each implementation. The time is formatted as MM:SS.mmm where MM is minutes, SS seconds and mmm milliseconds. The time was measured with the cputime function of MatLab.

LOQO shows its supremacy on these problems. This supremacy is due to two facts:

- The algorithm used by LOQO.
- The quality of the implementation.

MatLab		C++		LOQO	
solution	time	solution	time	solution	time
36.9000	00:00.109	36.9000	00:00.015	36.9000	00:00.000
-13008.8063	00:02.984	-13009.1023	00:00.093	-13009.1608	00:00.015
256525.6986	00:00.406	256524.9316	00:00.047	256526.8405	00:00.030
272317.5082	00:00.390	272318.3429	00:00.047	272318.7406	00:00.030
-691.4757	00:00.234	-691.4624	00:00.031	-691.4833	00:00.015
-717.3030	00:01.375	-717.3037	00:00.234	-717.3254	00:00.109
-9.4704	00:14.375	-9.4793	00:08.047	-9.4793	00:00.234
-9.6454	00:48.859	-9.6622	00:21.468	-9.6623	00:00.734

Table 2: Results on the benchmark test cases

The algorithm used by LOQO is an interior point method quite similar to ours, therefore we should at least match. The quality of the implementation is the biggest challenge.

LOQO is written in highly optimized C code and includes superior routines for sparse matrix manipulations.

Neither of our implementations can match with the many years and repeated optimizations of LOQO. However, as we shall soon see, we are indeed able to state the superiority of our decomposition scheme.

The MatLab implementation is no match to the C code, therefore we proceed by comparing only the C++ implementation with LOQO.

Our implementation makes no use of the sparsity inside the matrices defining the different nodes. This is something we still need to do. However, by generating problems with increasing density we can show that our decomposition algorithm excels a general purpose code by taking better advantage of the structure of the problem.

In order to generate sparse instances we use a similar technique to that described by Gonzaga in [39].

The objective functions are composed of vectors with 20% components equal to 0 and 80% equal to 1. The probability of each node was taken equal.

The problems where generated by adjusting the right hand side in such a way that a given nonzero vector is feasible.

By adding random sparse matrices we obtain less sparse matrices. We generated denser cases by adding 64 matrices.

Figure 2 shows the constraint matrix of the deterministic equivalent of a problem with 2 children

Figure 2: An instance with very sparse nodes

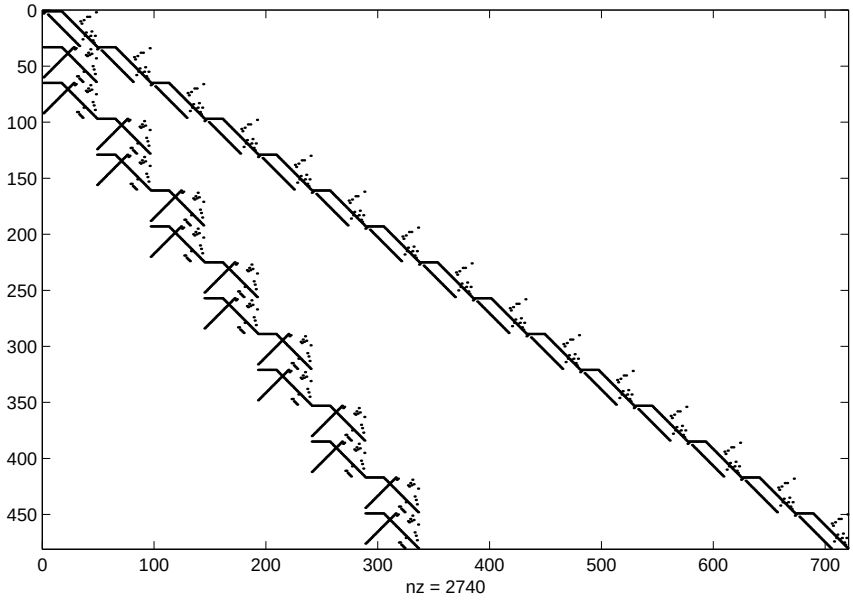
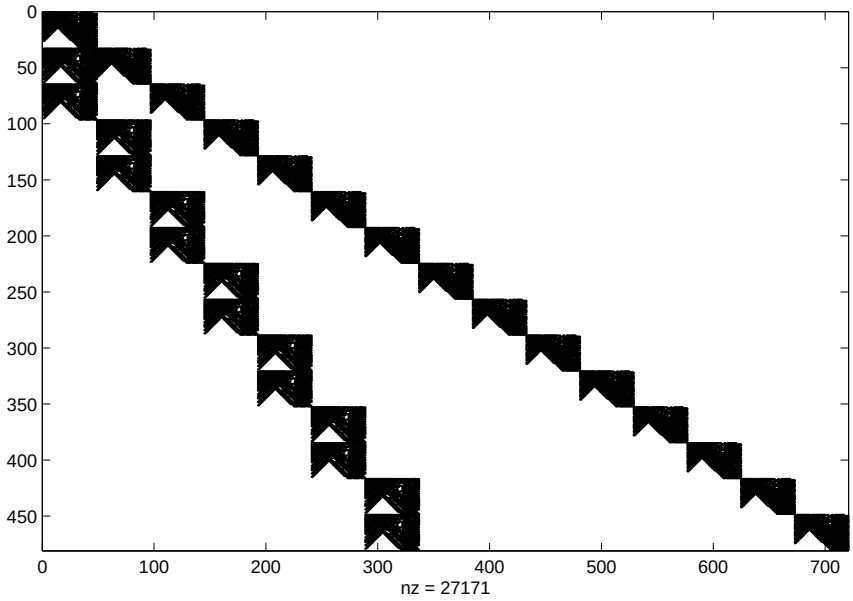


Figure 3: An instance with dense nodes



dimension					C++		LOQO	
node	c	s	n	size	solution	time	solution	time
64 x 72	8	2	9	576 x 648	908.3474	00:00.593	908.3478	00:00.125
48 x 64	16	2	17	816 x 1088	676.5941	00:00.906	676.5930	00:00.202
32 x 48	32	2	33	1056 x 1584	411.4839	00:01.234	411.4801	00:00.484
24 x 32	8	3	73	1752 x 2336	1280.7962	00:00.827	1280.7874	00:00.453

Table 3: Sparse nodes

dimension					C++		LOQO	
node	c	s	n	size	solution	time	solution	time
64 x 72	8	2	9	576 x 648	927.0170	00:01.000	927.0173	00:01.233
48 x 64	16	2	17	816 x 1088	649.5509	00:01.656	649.5497	00:02.452
32 x 48	32	2	33	1056 x 1584	464.1122	00:01.327	464.1072	00:02.875
24 x 32	8	3	73	1752 x 2336	1254.2051	00:00.906	1254.2015	00:02.077

Table 4: Dense nodes

per node and 4 stages. The instance depicted in this figure has rather sparse nodes, whose matrices where generated in the same fashion as described in [39].

Figure 2 the corresponding matrix when the nodes are generated by adding 64 sparse matrices.

We generated problems with levels of sparsity similar to Figure 2 and the results obtained can be found in Table 3. In all cases LOQO takes advantage of the sparsity of the deterministic equivalent and excels our implementation.

Things change dramatically, however, when the density of the nodes increases. Table 4 shows that our implementation beats LOQO in such cases.

Finally we report results concerning nonlinear convex problems. We generated the constraints as described before and took the following as objective functions:

$$f(x) = -\sum \log(x_i)$$

and

$$f(x) = \sum x_i^2.$$

The results are listed in Tables 5 and **??** respectively.

dimension					C++		
node	c	s	n	size	primal	dual	time
64 x 72	8	2	9	576 x 648	-763.4518	-763.4169	00:01:046
48 x 64	16	2	17	816 x 1088	-3868.7153	-3868.6309	00:01:156
32 x 48	32	2	33	1056 x 1584	-7946.0333	-7945.9155	00:01:078
24 x 32	8	3	73	1752 x 2336	-8306.4226	-8306.3381	00:01:359
24 x 32	8	4	585	14040 x 18720	-66565.5549	-66565.4703	00:10:453
24 x 32	8	5	4681	112344 x 149792	-532638.6140	-532638.5294	01:26:531
12 x 16	8	6	37449	449388 x 599184	-1582354.8202	-1582354.7351	06:04:203
6 x 8	7	7	137257	823542 x 1098056	-1514452.8967	-1514452.8104	09:10:734

Table 5: Nonlinear problems logarithmic

dimension					C++		
node	c	s	n	size	primal	dual	time
64 x 72	8	2	9	576 x 648	144.0000	144.0000	00:00:609
48 x 64	16	2	17	816 x 1088	204.0000	204.0000	00:00:671
32 x 48	32	2	33	1056 x 1584	264.0000	264.0000	00:00:640
24 x 32	8	3	73	1752 x 2336	438.0000	438.0000	00:00:734
24 x 32	8	4	585	14040 x 18720	3510.0000	3510.0000	00:05:890
24 x 32	8	5	4681	112344 x 149792	28086.0000	28085.9999	00:47:953
12 x 16	8	6	37449	449388 x 599184	112347.0001	112346.9997	02:32:328
6 x 8	7	7	137257	823542 x 1098056	205885.5003	205885.4995	05:48:328

Table 6: Nonlinear problems square root

Our results show that this algorithm stands well against a state of the art commercial package when solving linear multistage stochastic programs. Moreover, it enables solving nonlinear multistage stochastic problems of one million variables in less than 10 minutes using a simple personal computer.

Since personal computers with dual processor are becoming common place, a parallel implementation of our algorithm, which is still to come, will make it even more efficient as a practical tool to solve multistage stochastic problems.

6 Analyzing Infeasibility

Any optimization model is only an approximation of reality. This means in particular that if a model returns with an undesirable solution, or declares that no solution exists at all, then it might be: 1) the model is not appropriate; or 2) the model is good, but some 'soft constraints' must be relaxed in order to produce a sensible solution. In either case, it is of crucial importance to understand why the model does not have an optimal solution. For stochastic programming, testing the model is especially relevant, because very often the model is only a rough approximation of the real situation in the presence of uncertainty.

A major advantage of using a self-dual embedded model is that if the primal or the dual problem is infeasible, then, instead of getting insignificant output, such as "Infeasible Model", we always get a Farkas type certificate. As we will see in this section, this information is useful in order to analyze the cause of the infeasibility. To see this, consider a linear programming problem given as

$$\begin{array}{ll} \max & b^T y \\ \text{s.t.} & A^T y \leq c. \end{array}$$

Suppose that the problem is infeasible. Then by the Farkas lemma and a well known result of Goldman and Tucker [17], there exists $x^* \geq 0$ with maximal cardinality of positive components, such that $Ax^* = 0$ and $c^T x^* < 0$. Let the positive support of x^* be I, i.e., $I = \{i \mid x_i^* > 0\}$. Now we claim that if we remove the set of constraints indexed by I, then the remaining problem must be feasible. The proof is as follows. Let the complement set of I be J. Further let us assume, for the sake of obtaining a contradiction, that the system $A_J^T y \leq c_J$ is still infeasible. Then, by applying the Farkas lemma once more, we conclude that there is $u \geq 0$ such that $A_J u = 0$ and $c_J^T u < 0$. As a by-product, we also know $u \neq 0$. Let

$$\bar{x}_i = \begin{cases} u_i, & \text{if } i \in J \\ 0, & \text{if } i \notin J \end{cases}$$

This implies that $x^* + \bar{x} \geq 0$, $A(x^* + \bar{x}) = 0$, and $c^T(x^* + \bar{x}) < 0$. Moreover, $x^* + \bar{x}$ has a positive support set which is larger in cardinality than that of x^*. This yields a contradiction. As a result, the claimed fact is proven.

One implication of the above result is the following. Suppose that we have a Farkas type infeasibility certificate with maximal support. Then, its positive support part corresponds to the set of constraints that are causing the infeasibility. In other words, this is the set of scenarios that requires scrutiny. Numerically, if all the data are properly scaled, then we may interpret a higher-valued dual multiplier (Farkas type certificate) to correspond to the scenario that is likely to have more responsibilities for causing the infeasibility. Certainly, due to the primal-dual symmetricity, exactly the same thing can be said about a linear program in the primal form

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0. \end{array}$$

In this case, if the problem is infeasible then the corresponding Farkas type certificate is $-A^T y^* \geq 0$ such that $b^T y^* > 0$.

Let us consider one example to illustrate this point. Suppose that there is a risky asset, say a common stock, whose value may go up, or down, or remain unchanged in the next period. To be specific, assume that with 40% chance it will go up 10%, with 30% chance it will go down 4%, and with 30% chance it will be unchanged. Let us concentrate on a two-period situation. A riskless investment will always yield a 2% return rate in the same period, no matter what happens. Suppose we have a notional 1$ to invest. Now we wish to find an investment plan, allocating our wealth and possibly updating our portfolio at the end of period 1, in such a way that our wealth will never go below value g, and at the same time, the expected return rate is maximized at the investment horizon, i.e., at the end of period 2.

Clearly, in this case the scenario tree involves 13 nodes (see Figure 4). We denote the current stage to be (0,0), and denote the set of scenarios at the first stage (up till down) be (1,1), (2,1) and (3,1), each followed by three children. In particular, (1,1) is followed by (4,2), (5,2) and (6,2), (2,1) is followed by (7,2), (8,2) and (9,2), and (3,1) is followed by (10,2), (11,2) and (12,2); see Figure 4.

Let the uncertain return rate of the stock (over one period) be ξ. In our model, $\xi_u = 0.1$ (up), $\xi_e = 0$ (equal), and $\xi_d = -0.04$ (down). The riskless rate over one period is $r = 0.02$. Let x^s_{nt} be the amount invested in the stock, and x^b_{nt} be the amount invested in the riskless asset, at stage t if scenario n unfolds. Then, the constraints in the model are:

$$\begin{array}{l} x^s_0 + x^b_0 = 1, \\ (1+\xi)x^s_{a(n),0} + (1+r)x^b_{a(n),0} = x^s_{n1} + x^b_{n1}, \text{ for } n = 1,2,3 \\ x^s_{n2} + x^b_{n2} \geq g, \text{ for } n = 4,5,...,12. \end{array}$$

We further assume that no short selling is allowed. In our notation, this is equivalent to the following

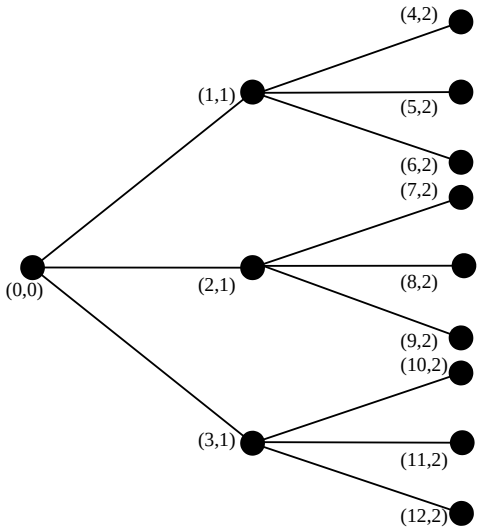


Figure 4: Simple Event Tree: Example

constraint matrices

$$
\begin{aligned}
&W_0 = [1, 1],\ h_0 = 1,\\
&B_{n1} = -[1+\xi, 1+r],\ W_{n1} = [1, 1],\ h_{n1} = 0,\\
&B_{n2} = \begin{bmatrix} -(1+\xi), & -(1+r) \\ 0, & 0 \end{bmatrix},\ W_{n2} = \begin{bmatrix} 1, & 1, & 0 \\ 1, & 1, & -1 \end{bmatrix},\ h_{n2} = \begin{bmatrix} 0 \\ g \end{bmatrix}.
\end{aligned}
$$

Note here that we have introduced 9 slack variables for the guaranteed return constraints: $x^s_{n2}+x^b_{n2} \geq g,\ n = 4, 5, ..., 12$.

Now we run our algorithm on this model. When the investor invests his or her wealth in the riskfree asset over the two-periods in our model, his final wealth will be 1.0404 ($1 * 1.02 * 1.02$). A higher guaranteed return is not possible due to the uncertainty of the risky asset. Hence, for $g \leq 1.0404$, the model is solvable. For instance, if $g = 1$, then $x^s_0 = 0.6601$ and $x^b_0 = 0.3399$, with an expected return rate at the horizon being 5.03%. In comparison, the riskless return rate at the horizon is 4.04%. As soon as $g > 1.0404$, the model becomes demanding a guaranteed return rate higher than the riskless rate. This is certainly impossible in the world where no arbitrage opportunity exists, and the model should return as being infeasible. Indeed, this turns out to be the case. Set for instance $g = 1.05$. The model is infeasible. The point here is, our algorithm also provides a dual certificate

$$
\begin{aligned}
-A^T y^* \quad = \quad & [0.3548, 0.0068, 0.1983, 0.0138, 0.2513, 0.0120, 0.2972, 0.0077,\\
& 0.0248, 0.0248, 1.7230, 0.0243, 0.0243, 2.8122, 0.0242, 0.0242, 4.4356,\\
& 0.0240, 0.0240, 2.1821, 0.0238, 0.0238, 3.7830, 0.0239, 0.0239, 5.6368,\\
& 0.0239, 0.0239, 2.5476, 0.0237, 0.0237, 4.5298, 0.0237, 0.0237, 6.7116].
\end{aligned}
$$

If we examine the values in the certificate carefully, then we will find that there are 9 numbers that are significantly higher than 1, and they correspond to the constraints:

$$x^s_{n2} + x^b_{n2} \geq g, \text{ for } n = 4, 5, ..., 12.$$

As we know, these constraints are indeed the ones that are causing the infeasibility.

Similar analysis applies to the situation when the problem has an unbounded optimum value. In portfolio applications, this phenomenon is known as the existence of an arbitrage opportunity. In particular, the primal Farkas type certificate plays the role of displaying one such arbitrage opportunity. In this case, the meaning of the solution produced by our algorithm is quite obvious.

7 Concluding Remarks

In this paper we study a new computational methodology for solving the multistage stochastic programming model with a convex objective. A disadvantage of stochastic programming is the so-called 'curse of dimensionality'. As scenario trees grow exponentially with the number of states of the world, the problem quickly grows out of hand. An efficient method should cope with this problem. Our approach is as follows. First, we rely on the use of primal-dual interior point method. As a general merit of interior point methods, the number of iterations required to solve an optimization problem is typically low and insensitive to the dimension of the problem. This is obviously an important property for solving large-scale stochastic programs. Second, we decompose the problem completely, alleviating the 'curse of dimensionality'. Rather than solving a large-scale model at once, we only need to solve much smaller submodels defined by the subtrees of the scenario tree. Finally, we rely on the homogeneous self-dual (HSD) method, which was introduced for linear programming by Ye, Todd and Mizuno [32], simplified by Xu, Hung and Ye [30], and extended to convex programming by Andersen and Ye [2]. One of the advantages of the HSD method is that it requires no feasibility phase, allowing us to select any interior starting point (possibly infeasible). Moreover, the method is capable of detecting infeasibility and linking this infeasibility directly to a certain (set of) scenario(s). Finally, since the algorithm is based on a primal-dual interior point method, we obtain solutions to both the primal and the dual problem, yielding additional information.

References

[1] E.D. Andersen, and K.D. Andersen, *The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm*, High Performance Optimization Tech-

niques, pp. 197-232, eds. J.B.G. Frenk, K. Roos, T. Terlaky and S. Zhang, Kluwer Academic Publishers, 1999.

[2] E.D. Andersen, and Y. Ye, *A computational study of the homogeneous algorithm for large-scale convex optimization*, Computational Optimization and Applications 10, 243-269, 1998.

[3] K.A. Ariyawansa, and P.L. Jiang, Polynomial cutting plane algorithms for two-stage stochastic linear programs based on ellipsoids, volumetric centers and analytic centers, *Working Paper*, Department of Pure and Applied Mathematics, Washington State University, Pullman, USA, 1996.

[4] A. Berkelaar, C. Dert, B. Oldenkamp, and S. Zhang, *A Primal-Dual Decomposition-Based Interior Point Approach to Two-Stage Stochastic Linear Programming*, Operations Research 50, 904-915, 2002.

[5] O. Bahn, O. du Merle, J.-L. Goffin, and J.P. Vial, *A cutting plane method from analytic centers for stochastic programming*, Mathematical Programming 69, 45-73, 1995.

[6] A.J. Berger, J.M. Mulvey, E. Rothberg, and R.J. Vanderbei, Solving multistage stochastic programs using tree dissection, *Technical Report SOR-95-07*, Department of Civil Engineering and Operations Research, Princeton University, USA, 1995.

[7] A.J. Berger, J.M. Mulvey, and A. Ruszczynski, *An extension of the DQA algorithm to convex stochastic programs*, SIAM Journal on Optimization 4, 735-753, 1994.

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

[9] J.R. Birge, C.J. Donohue, D.F. Holmes, and O.G. Svintsitski, *A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs*, Mathematical Programming 75, 327-352, 1996.

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

[11] J.R. Birge, and F. Louveaux, *Introduction to Stochastic Programming*, Springer, New York, 1997.

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

[13] J.R. Birge, and C.H. Rosa, *Parallel decomposition of large-scale stochastic nonlinear programs*, Annals of Operations Research 64, 39-65, 1996.

[14] I.C. Choi, and D. Goldfarb, *Exploiting special structure in a primal-dual path-following algorithm*, Mathematical Programming 58, 33-52, 1993.

[15] J. Czyzyk, R. Fourer and S. Mehrotra, *Using a massively parallel processor to solve large sparse linear programs by an interior-point method*, SIAM Journal on Scientific Computing 19, 553-565, 1998.

[16] E. Fragnière, J. Gondzio, and J.-P. Vial, *Building and solving large-scale stochastic programs on an affordable distributed computing system*, Annals of Operations Research 99, 167-187, 2000.

[17] A.J. Goldman, and A.W. Tucker, Polyhedral convex cones, in H.W. Kuhn and A.W. Tucker eds., *Linear Inequalities and Related Systems*, Princeton University Press, New Jersey, pp. 19-40, 1956.

[18] J. Gondzio, and R. Kouwenberg, *High performance computing for asset liability management*, Operations Research 49, 879-891, 2001.

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

[20] E.R. Jessup, D. Yang, and S.A. Zenios, *Parallel factorization of structured matrices arising in stochastic programming*, SIAM Journal on Optimization 4, 833-846, 1994.

[21] P. Kall and S.W. Wallace, Stochastic Programming, John Wiley and Sons, Chichester, 1994.

[22] Z.Q. Luo, J.F. Sturm, and S. Zhang, *Conic convex programming and self-dual embedding*, Optimization Methods and Software 14, 169-218, 2000.

[23] I.J. Lustig, J.M. Mulvey and T.J. Carpenter, *The formulation of stochastic programs for interior point methods*, Operations Research 39, 757-770, 1991.

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

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

[26] J.M. Mulvey, and W.T. Ziemba eds., *Asset and Liability Management from a Global Perspective*, Cambridge University Press, 1998.

[27] J.F. Sturm, *Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones*, Optimization Methods and Software 11-12, 625–653, 1999. Special issue on Interior Point Methods (CD supplement with software).

[28] T. Terlaky ed., Interior Point Methods of Mathematical Programming, Kluwer Academic Publishers, Dordrecht, 1996.

[29] R. Van Slyke, and R.J.-B. Wets, *L-shaped linear programs with application to optimal control and stochastic programming*, SIAM Journal on Applied Mathematics 17, 638-663, 1969.

[30] X. Xu, P.F. Hung, and Y. Ye, *A simplified homogeneous self-dual linear programming algorithm and its implementation*, Annals of Operations Research 62, 151-171, 1996.

[31] D. Yang, and S.A. Zenios, *A Scalable parallel interior point algorithm for stochastic linear programming and robust optimization*, Computational Optimization and Applications 7, 143-158, 1997.

[32] Y. Ye, M.J. Todd, and S. Mizuno, *An $\mathcal{O}(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm*, Mathematics of Operations Research 19, 53-67, 1994.

[33] G.Y. Zhao, A Lagrangian dual method with self-concordant barrier for multi-stage stochastic convex nonlinear programming, *Working paper*, Department of Mathematics, National University of Singapore, 1998.

[34] G.Y. Zhao, *A log-barrier method with Benders decomposition for solving two-stage stochastic linear programs*, Mathematical Programming 90, 507-536, 2001.

[35] J.R. Birge and D.F. Holmes, *A (PO)rtable (S)tochastic programming (T)est (S)et (POSTS)*, Northwestern University, http://users.iems.nwu.edu/ jrbirge//html/dholmes/POSTresults.html

[36] A. Felt, *Test-Problem Collection for Stochastic Linear Programming*, University of Wisconsin-Stevens Point, http://www.uwsp.edu/math/afelt/slptestset.html

[37] R.J. Vanderbei, *LOQO*, Princeton University, http://www.orfe.princeton.edu/ loqo/

[38] H.I. Gassmann, *The SMPS format for stochastic linear programs*, Dalhousie University, http://www.mgmt.dal.ca/sba/profs/hgassmann/SMPS2.htm

[39] C.G. Gonzaga, *Generation of Degenerate Linear Programming Problems*, Federal University of Santa Catarina, http://jurere.mtm.ufsc.br/ clovis/files/degenerate.ps