
Computational Optimization and Applications 15, 215–234 (2000)

Solving Stochastic Linear Programs with Restricted Recourse Using Interior Point Methods*

PATRIZIA BERALDI beraldi@parcolab.unical.it
Dipartimento di Elettronica, Informatica e Sistemistica, Università degli Studi della Calabria, 87036 Rende (CS), Italy

ROBERTO MUSMANNO musmanno@ingle01.unile.it
Dipartimento di Ingegneria, dell'Innovazione, Università degli Studi di Lecce, 73100 Lecce, Italy

CHEFI TRIKI chefi@parcolab.unical.it
Dipartimento di Elettronica, Informatica e Sistemistica, Università degli Studi della Calabria, 87036 Rende (CS), Italy

Received July 8, 1997; Accepted February 12, 1999

Abstract. In this paper we present a specialized matrix factorization procedure for computing the dual step in a primal-dual path-following interior point algorithm for solving two-stage stochastic linear programs with restricted recourse. The algorithm, based on the Birge-Qi factorization technique, takes advantage of both the dual block-angular structure of the constraint matrix and of the special structure of the second-stage matrices involved in the model. Extensive computational experiments on a set of test problems have been conducted in order to evaluate the performance of the developed code. The results are very promising, showing that the code is competitive with state-of-the-art optimizers.

Keywords: stochastic linear programs, restricted recourse, interior point methods, structured matrix factorization

1. Introduction

Stochastic linear programming with restricted recourse is an approach proposed recently in [20] to control the robustness of recourse solutions in stochastic programming models. The need to have a solution that is not very sensitive to varying values of stochastic parameters, is particularly significant in real-world applications for which changes in decisions are often difficult to implement (problems of capacity planning, manufacturing, personnel management, economic dispatch of power systems, etc.).

The stochastic linear program with restricted recourse can be viewed as an extension of the classical two-stage stochastic program. In this latter model, the decision variables are partitioned into two distinct sets: $x \in R^{n_1}$, the set of first-stage variables represents decisions made before observing the values of uncertain parameters, and $y \in R^{n_2}$, the

*Research partially supported through contract "HPC-Finance" (no. 951139) of the INCO '95 project funded by the Directorate General III (industry) of the European Commission. Partial support also provided by the "HPC-Finance" partner institutions: Universities of Bergamo (IT), Calabria (IT), Cambridge (UK), Charles (CZ), Cyprus (CY), Erasmus (ND), Technion (IL) and the "Centro per il Calcolo Parallelo e i Supercalcolatori" (IT).

vector of second-stage, or *control*, decisions that can be viewed as *recourse* actions taken once a specific realization of the uncertain parameters has been observed. Uncertainty is incorporated in terms of random variables defined in some discrete probability space (Ω, F, P) and represented by a set of *scenarios* $\Omega = \{1, \ldots, N\}$ with associated probabilities $\{p_1, \ldots, p_N\}$.

The two-stage stochastic linear program can be posed in the following deterministic equivalent form:

$$
\begin{aligned}
\text{(SLP)} \quad \min \quad & c_0^T x + \sum_{l=1}^{N} p_l c_l^T y_l \\
\text{s.t.} \quad & A_0 x = b_0, \\
& T_l x + W_l y_l = h_l, \quad l = 1, \ldots, N, \\
& l_x \le x \le u_x, \\
& l_y \le y_l \le u_y, \quad l = 1, \ldots, N,
\end{aligned}
$$

where A_0 is a $m_1 \times n_1$ matrix, b_0 is a m_1 vector, c_0 is a n_1 decision vector, and $u_x, l_x \in R^{n_1}$ represent an upper and a lower bound on the first-stage vector, respectively. Furthermore, $c_l \in R^{n_2}$, $T_l \in R^{m_2 \times n_1}$, $W_l \in R^{m_2 \times n_2}$, and $h_l \in R^{m_2}$ represent a particular realization of the cost vector and the constraint coefficients corresponding to each scenario $l \in \Omega$. Finally, u_y and $l_y \in R^{n_2}$ represent upper and lower bound vectors on the second-stage variables, respectively.

From the above formulation, it is evident that the overall constraint matrix denoted by $A \in R^{m \times n}$ ($m = m_1 + Nm_2$, and $n = n_1 + Nn_2$) has a dual block-angular structure, that can be exploited in the development of efficient solution algorithms:

$$
A = \begin{pmatrix}
A_0 & & & & \\
T_1 & W_1 & & & \\
T_2 & & W_2 & & \\
\vdots & & & \ddots & \\
T_N & & & & W_N
\end{pmatrix}.
$$

In the classical (SLP) model, the first-stage variables x are scenario-invariant, but different recourse decisions y_l are allowed corresponding to each scenario $l \in \Omega$. Since no restriction is imposed on the variability of the recourse decisions across scenarios, large dispersion of the recourse vectors y_l, $l \in \Omega$, may be observed. The restricted recourse approach [20] limits the variablity of the recourse actions by directly adding a "dispersion" constraint to the (SLP) model. The approach uses similar ideas to those of the robust optmization [16], which represents the first attempt to force the stochastic solution to obey to some desidered characteristics. However, robust optimization is concerned with solution robustness (i.e. optimality) and model robustness (i.e. feasibility), and it does not examine the recourse robustness (in effect, the two approaches cannot be considered equivalent or alternative).

In this paper, we develop a specialized interior point method for the solution of the restricted recourse model. The method specializes the linear algebra involved in the computation of the dual step, which is by far far the most time consuming part in the whole procedure. Our approach is based on the exploitation of both the dual block-angular structure of the two-stage stochastic program and the special structure of the second-stage matrices involved in the model.

The paper is organized as follows. In Section 2, we introduce the restricted recourse stochastic program and we describe the specialized interior point procedure developed for its solution. In Section 3, we investigate the effect of the restricted recourse on the two-stage stochastic programs. Section 4 is devoted to the discussion of the experimental results. Finally, the last section presents concluding remarks.

2. The restricted recourse stochastic program

The general two-stage stochastic model with restricted recourse (SLPRR for short) is obtained from the (SLP) model by adding a constraint that limits the variability of the recourse decisions across the scenarios. Such a constraint can be formulated, for example, by using the mean values computed on the basis of the recourse variables.

More specifically, given a set y_l, $l \in \Omega$, of second-stage variables with a corresponding probability p_l, the *mean recourse vector* is defined as:

$$\bar{y} = \sum_{l=1}^{N} p_l y_l. \tag{1}$$

The mean dispersion of the recourse variables y_l, $l \in \Omega$, from the mean recourse vector $\bar{y}$ can be expressed, can be expressed by using a suitable norm. In the sequel, the Euclidean norm is considered:

$$\rho = \sum_{l=1}^{N} p_l \, \|y_l - \bar{y}\|. \tag{2}$$

Thus, the (SLPRR) model can be formulated as follows:

$$\begin{aligned}
\text{(SLPRR)} \quad \min \quad & c_0^T x + \sum_{l=1}^{N} p_l c_l^T y_l \\
\text{s.t.} \quad & A_0 x = b_0, \\
& T_l x + W_l y_l = h_l, \quad l = 1, \ldots, N, \\
& \sum_{l=1}^{N} p_l \, \|y_l - \bar{y}\| \leq \epsilon, \\
& l_x \leq x \leq u_x, \\
& l_y \leq y_l \leq u_y, \quad l = 1, \ldots, N.
\end{aligned} \tag{3}$$

By reducing parametrically the tolerance $\epsilon > 0$, a sequence of recourse solutions with a controlled degree of variability is obtained. Naturally, more stringent robustness conditions cause an increase in the objective function value. Therefore, the goal is to identify a solution strategy with a relatively low dispersion that, on the other hand, does not imply an excessive increase in the expected cost.

The introduction of the "dispersion constraint" (3) in the model makes the problem very difficult to solve because the constraint is nonlinear and nonseparable. In order to overcome this difficulty, three different approximation procedures to the general model have been proposed and analyzed in [20]. They are based on a linearization of the dispersion constraint, but differ in the choice of the point around which the search of a solution with an acceptable degree of dispersion is confined. In this paper, on the basis of the computational experiments carried out in [20], we consider the most appropriate scheme in terms of cost-robustness tradeoff. It consists of restricting the variablity among all recourse vectors from a common, but arbitrary point $z \in R^{n_2}$.

Before formalizing the method for solving the (SLPRR) model, we recall from [20] the following concepts and notation.

Given an optimal solution $\{x^*, y_1^*, y_2^*, \ldots, y_N^*\}$ of the (SLP) problem, the maximum and the minimum values for each recourse component i, over all recourse vectors y_l^*, $l \in \Omega$, are defined as:

$$\bar{u}_i = \max_{l \in \Omega} \left(y_l^*\right)_i, \quad i = 1, \ldots, n_2, \tag{4}$$

$$\bar{l}_i = \min_{l \in \Omega} \left(y_l^*\right)_i, \quad i = 1, \ldots, n_2, \tag{5}$$

and $\bar{u}_y = (\bar{u}_1, \ldots, \bar{u}_{n_2})$, and $\bar{l}_y = (\bar{l}_1, \ldots, \bar{l}_{n_2})$ represent the *virtual upper bound* and *virtual lower bound* vectors, respectively.

Thus, the *virtual range* vector ω is defined as:

$$\omega = \bar{u}_y - \bar{l}_y. \tag{6}$$

The components of the virtual range represent the coordinate dimensions of the smallest n_2-dimensional hyper-rectangle that contains all the current recourse solutions $y_l, l \in \Omega$. The dimension δ of the smallest n_2-dimensional hypercube containing all current recourse solutions is defined as:

$$\delta = \max_{i=1,\ldots,n_2} \{\omega_i\}, \tag{7}$$

whereas the hypercube $d \in R^{n_2}$ is:

$$d = \delta \mathbf{1}, \tag{8}$$

where $\mathbf{1}$ is the n_2-dimensional vector of ones.

At this point, we are able to present a formal description of the procedure for enforcing restricted recourse on (SLP).

Restricted recourse procedure

1. Set $k = 0$. Select a tolerance $\epsilon > 0$, and the restriction factor $\lambda \in (0, 1)$. Find an optimal solution $\{x^k, y_1^k, \ldots y_N^k\}$ of problem (SLP).
2. Compute the mean recourse $\bar{y}^k$ (Eq. (1)), and the recouse dispersion ρ^k (Eq. (2)). If $\rho^k < \epsilon$, the solution is robust within the prescribed tolerance, and the procedure terminates. Otherwise, the virtual bounds $\bar{u}_y^k, \bar{l}_y^k$ (by Eqs. (4) and (5)), the virtual range ω^k (by Eq. (6)) and the range d^k (by Eqs. (7) and (8)) are computed.
3. Solve the following linearized restricted recourse stochastic program:

$$
\begin{aligned}
\text{(RRSP)} \quad \min \quad & c_0^T x + \sum_{l=1}^{N} p_l c_l^T y_l \\
\text{s.t.} \quad & A_0 x = b_0, \\
& T_l x + W_l y_l = h_l, \quad l = 1, \ldots, N, \\
& -\frac{\lambda}{2} d^k \le y_l - z \le \frac{\lambda}{2} d^k, \quad l = 1, \ldots, N, \\
& l_x \le x \le u_x, \\
& l_y \le y_l \le u_y, \quad l = 1, \ldots, N, \\
& l_y \le z \le u_y,
\end{aligned}
$$

to obtain a new solution $\{x^{k+1}, z^{k+1}, y_1^{k+1}, \ldots, y_N^{k+1}\}$. If the problem is infeasible for the current settings of the robustness bound no further robustness can be achieved and the procedure terminates. Otherwise, set $k = k + 1$ and return to Step 2.

The vector $z \in R^{n_2}$ represents the cluster point around which all recourse decisions are confined. The determination of the optimal coordinates of this point is left to the model. This freedom in the choice of the position of z allows a high flexibilty of the method, but, on the other hand, leads to an increase of the size of the corresponding problem (n_2 additional variables).

By setting slack variables $s_l = y_l - z$, $l \in \Omega$, the linearized dispersion constraints in the (RRSP) model can be replaced by:

$$
\begin{aligned}
& y_l - z - s_l = 0, \quad l = 1, \ldots, N, \\
& -\frac{\lambda}{2} d^k \le s_l \le \frac{\lambda}{2} d^k, \quad l = 1, \ldots, N.
\end{aligned}
$$

Consequently, the constraint new matrix $\bar{A} \in R^{\bar{m}\times\bar{n}}$ ($\bar{m} = m_1 + N(m_2 + n_2)$, and $\bar{n} = n_1 + (2N + 1)n_2$) takes the form:

$$\bar{A} = \begin{array}{c} \begin{matrix} z & x & y_1 & s_1 & y_2 & s_2 & \cdots & y_N & s_N \end{matrix} \\ \begin{pmatrix} & A_0 & & & & & & & \\ & T_1 & W_1 & & & & & & \\ -I & & I & -I & & & & & \\ & T_2 & & & W_2 & & & & \\ -I & & & & I & -I & & & \\ & \vdots & & & & & \ddots & & \\ & T_N & & & & & & W_N & \\ -I & & & & & & & I & -I \end{pmatrix} \end{array}.$$

It is easy to see that, by setting

$$\bar{A}_0 = (0\ A_0),$$

and, for each scenario $l \in \Omega$,

$$\bar{W}_l = \begin{pmatrix} W_l & 0 \\ I & -I \end{pmatrix}, \qquad \bar{T}_l = \begin{pmatrix} 0 & T_l \\ -I & 0 \end{pmatrix},$$

the matrix $\bar{A}$ maintains a dual block-angular structure such as the one of the constraint matrix A in the (SLP) model.

The method developed for solving the (RRSP) model relies on the same primal-dual path-following method used in LOQO [18]. The two methods differ only in the way they solve the *Newton equations* (i.e. the system of equations that represent first-order optimality conditions for the primal and dual logarithmic barrier problems) to determine the dual search direction Δt. The solution of this system represents the major computational effort in any interior point algorithm. While LOQO solves an *augmented system* of Newton equations, our method solves the *normal system* $M\Delta t = \psi$, where $M = \bar{A}D\bar{A}^T$, $D \in R^{\bar{n}\times\bar{n}}$ is a diagonal positive definite matrix depending on the current estimate of the solution and ψ is a specific vector computed from the current iterate.

It is important to note that M is much denser than the original matrix $\bar{A}$ and, thus, a straightforward implementation of an interior point method that directly solves systems with matrix M is quite inefficient. Several approaches have been proposed either to reduce the number of dense columns [15] or to separate them from the other (nondense) columns [2] (see [4] for a detailed comparison). One of the most efficient strategies is based on an explicit factorization of the dual block-angular matrix. The main result in this respect is due to Birge and Qi [6] and it is based on the Shermann-Morrison-Woodbury formula, reported in the following lemma.

Lemma 2.1. *For any matrices S, U, V such that S, and $G = I + V^T S^{-1}U$ are invertible,*

$$(S + UV^T)^{-1} = S^{-1} - S^{-1}UG^{-1}V^T S^{-1}. \tag{9}$$

In the case of the (RRSP) model, the general formula can be exploited to compute M^{-1} as stated in the following theorem.

Theorem 2.1. *Let* $D = \text{diag}\{D_0, D_1, \ldots, D_N\}$, *and* $S = \text{diag}\{S_0, S_1, \ldots, S_N\}$, *where* $D_0 \in R^{(n_1+n_2)\times(n_1+n_2)}$, *corresponds to the iterate of the first-stage variables*, D_l *is the diagonal* $2n_2 \times 2n_2$ *submatrix corresponding to the* l*th block of* D (*i.e. to the iterates of the second-stage variables*), $S_0 = I \in R^{m_1\times m_1}$, *and, for each scenario* $l \in \Omega$, $S_l = \bar{W}_l D_l \bar{W}_l^T \in R^{(m_2+n_2)\times(m_2+n_2)}$.

Furthermore, let

$$
\begin{aligned}
H &= D_0^{-2} + \bar{A}_0^T \bar{A}_0 + \sum_{l=1}^{N} \bar{T}_l^T S_l^{-1} \bar{T}_l, \\
G &= \begin{pmatrix} H & \bar{A}_0^T \\ -\bar{A}_0 & 0 \end{pmatrix}, \qquad \bar{U} = \begin{pmatrix} \bar{A}_0 & I \\ \bar{T}_1 & 0 \\ \vdots & 0 \\ \bar{T}_N & 0 \end{pmatrix}, \qquad \bar{V} = \begin{pmatrix} \bar{A}_0 & -I \\ \bar{T}_1 & 0 \\ \vdots & 0 \\ \bar{T}_N & 0 \end{pmatrix}.
\end{aligned} \tag{10}
$$

If $\bar{A}_0$ *and* $\bar{W}_l$, $l \in \Omega$, *have full row rank, then* M *and* $L = -\bar{A}_0 H^{-1} \bar{A}_0^T$ *are invertible and*

$$
M^{-1} = S^{-1} - S^{-1}\bar{U}G^{-1}\bar{V}^T S^{-1}. \tag{11}
$$

Proof: By setting $\hat{D} := \text{diag}\{D_0, I\}$, $U := \bar{U}\hat{D}$ and $V := \bar{V}\hat{D}$, the matrix M can be written as $M = S + UV^T$. In order to apply Lemma 2.1, we observe that both S and $(I + V^T S^{-1} U)$ must be invertible.

For each scenario $l \in \Omega$, the matrix D_l is invertible and, in addition, $\bar{W}_l$ has full rank; consequently, $S_l = \bar{W}_l D_l \bar{W}_l^T$ and, therefore, S is also invertible.

The matrix $I + V^T S^{-1} U = I + \hat{D}\bar{V}^T S^{-1}\bar{U}\hat{D} = \hat{D}(\hat{D}^{-2} + \bar{V}^T S^{-1}\bar{U})\hat{D}$ is invertible if and only if $(\hat{D}^{-2} + V^T S^{-1} U)$ is invertible (since $\hat{D}$ is not singular). We note that $(\hat{D}^{-2} + \bar{V}^T S^{-1}\bar{U})$ is equal to G. This result is an immediate consequence of what follows:

$$
\begin{aligned}
G &= \begin{bmatrix} D_0^{-2} + \bar{A}_0^T \bar{A}_0 + \sum_{l=1}^{N} \bar{T}_l^T S_l^{-1} \bar{T}_l & \bar{A}_0^T \\ -\bar{A}_0 & I - I \end{bmatrix} \\
&= \begin{bmatrix} D_0 & 0 \\ 0 & I \end{bmatrix}^{-2} + \begin{bmatrix} \bar{A}_0^T \bar{A}_0 + \sum_{l=1}^{N} \bar{T}_l^T S_l^{-1} \bar{T}_l & \bar{A}_0^T \\ -\bar{A}_0 & -I \end{bmatrix} \\
&= \hat{D}^{-2} + \bar{V}^T S^{-1} \bar{U}.
\end{aligned}
$$

We can now show that G is not singular. Both D_0^{-2} and $\bar{A}_0^T \bar{A}_0$ are symmetric and definite positive. $\bar{T}_l^T S_l^{-1} \bar{T}_l$ is definite positive for each $l \in \Omega$, since S_l is definite positive. Therefore, H is symmetric and definite positive and, consequently, invertible. The inverse H^{-1} has rank equal to n_1 and can be decomposed into $H^{-1} = H^{-1/2} H^{-1/2}$, where $H^{-1/2}$

is symmetric. We have assumed that $\bar{A}_0$ has full rank and as a consequence the product $\bar{A}_0 H^{-1/2}$ gives a full rank matrix and $L = -\bar{A}_0 H^{-1} \bar{A}_0^T$ is invertible. The matrix G has rank equal to $m_1 + n_1$ and, therefore, is invertible.

Once we have demonstrated that the matrix $(I + V^T S^{-1} U)$ is not singular, we can apply Lemma 2.1 to invert M:

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

□

Directly applying Theorem 2.1 to explicitly compute M^{-1} is not an efficient way to determine Δt. However, the dual step can be obtained as $\Delta t = p - r$, where p is the solution of $Sp = \psi$ and r is obtained by solving the following systems:

$$Gq = V^T p, \tag{12}$$

$$Sr = Uq. \tag{13}$$

The vector p can be computed component-wise by solving $S_l p_l = \psi_l$, for each $l = 0, \ldots, N$. Exploiting even further the special structure of the matrices involved in the definition of S_l in the (RRSP) model we can rewrite $S_l p_l = \psi_l$, for $l = 1, \ldots, N$, as:

$$\begin{pmatrix} W_l D_l' W_l^T & W_l D_l' \\ D_l' W_l^T & D_l' + D_l'' \end{pmatrix} \begin{pmatrix} p_l' \\ p_l'' \end{pmatrix} = \begin{pmatrix} \psi_l' \\ \psi_l'' \end{pmatrix},$$

where $D_l', D_l'' \in R^{n_2 \times n_2}$ are the submatrices of D_l such that $D_l = \mathrm{diag}\{D_l', D_l''\}$. Consequently, the vectors $p_l' \in R^{m_2}$ and $p_l'' \in R^{n_2}$ can be found by solving:

$$S_l' p_l' = \psi_l' - S_l'' \psi_l'', \tag{14}$$

and

$$p_l'' = (D_l' + D_l'')^{-1} \psi_l'' - S_l''' p_l', \tag{15}$$

where the matrices $S_l' \in R^{m_2 \times m_2}$, $S_l'' \in R^{m_2 \times n_2}$, and $S_l''', \in R^{n_2 \times m_2}$ are given by:

$$S_l' = W_l D_l' W_l^T - W_l D_l' (D_l' + D_l'')^{-1} D_l' W_l^T, \tag{16}$$

$$S_l'' = W_l D_l' (D_l' + D_l'')^{-1}, \tag{17}$$

$$S_l''' = (D_l' + D_l'')^{-1} D_l' W_l^T. \tag{18}$$

These matrices can be now used to compute $\bar{T}_l^T S_l^{-1} \bar{T}_l$, and then to form H in order to proceed with the calculation of q and r.

The procedure for doing so is developed next.

If we set

$$S_l^{-1} \bar{T}_l = R_l \doteq \begin{pmatrix} R_l' & R_l'' \\ R_l''' & R_l'''' \end{pmatrix},$$

the submatrices $R_l' \in R^{m_2 \times n_2}$ and $R_l'' \in R^{m_2 \times n_1}$ can be found by solving:

$$S_l' R_l' = S_l'', \tag{19}$$

$$S_l' R_l'' = T_l, \tag{20}$$

from which the submatrices $R_l''' \in R^{n_2 \times n_2}$ and $R_l'''' \in R^{n_2 \times n_1}$ are derived:

$$R_l''' = -(D_l' + D_l'')^{-1} - S_l''' R_l', \tag{21}$$

$$R_l'''' = -S_l''' R_l''. \tag{22}$$

Thus, multiplying R_l by the transpose of the matrix $\bar{T}_l$, we get the matrix:

$$\bar{T}_l^T S_l^{-1} \bar{T}_l = \begin{pmatrix} -R_l''' & -R_l'''' \\ T_l^T R_l' & T_l^T R_l'' \end{pmatrix}. \tag{23}$$

These matrices should be summed over all scenarios $l \in \Omega$ and then added to $D_0^{-2} + \bar{A}_0^T \bar{A}_0$ to form H (Eq. (10)).

At this point, the block structure of matrix G can be exploited to find q:

$$Gq = \begin{pmatrix} H & \bar{A}_0^T \\ -\bar{A}_0 & 0 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \begin{pmatrix} \pi' \\ \pi'' \end{pmatrix}, \tag{24}$$

where $\binom{\pi'}{\pi''} = V^T p$, $q_1 \in R^{(n_1+n_2)}$, and $q_2 \in R^{m_1}$. Hence, we get:

$$q_2 = -L^{-1}(\pi'' + \bar{A}_0 H^{-1} \pi'), \tag{25}$$

$$q_1 = H^{-1}\big(\pi' - \bar{A}_0^T q_2\big). \tag{26}$$

Once q is known, r can be computed component-wise by solving the systems $S_l r_l = \bar{T}_l q_1$, $l = 1, \ldots, N$. By exploiting again the structure of the matrix S_l, we have:

$$\begin{pmatrix} W_l D_l' W_l^T & W_l D_l' \\ D_l' W_l^T & D_l' + D_l'' \end{pmatrix} \begin{pmatrix} r_l' \\ r_l'' \end{pmatrix} = \begin{pmatrix} 0 & T_l \\ -I & 0 \end{pmatrix} \begin{pmatrix} q_1' \\ q_1'' \end{pmatrix} = \begin{pmatrix} T_l q_1'' \\ -q_1' \end{pmatrix},$$

where $r_l' \in R^{m_2}$, $r_l'' \in R^{n_2}$, $q_1' \in R^{n_2}$, and $q_1'' \in R^{n_1}$, or equivalently:

$$S_l' r_l' = T_l q_1'' + S_l'' q_1', \tag{27}$$

$$r_l'' = -(D_l' + D_l'') q_1'' - S_l''' r_l'. \tag{28}$$

The implementation of the strategy proposed in Theorem 2.1 requires, at each iteration of the primal-dual path following method, the Cholesky factorization of matrices S_l, $l = 1, \ldots, N$, H, and L. As shown above, the special structure of the matrices involved in the (RRSP) model is further exploited in the computation of the dual step Δt. More specifically, instead of factorizing, for each scenario $l \in \Omega$, the larger matrix $S_l \in R^{(m_2+n_2)\times(m_2+n_2)}$, the smaller matrix $S'_l \in R^{m_2 \times m_2}$ is considered, with a consequent reduction of the computational effort. Problems with a large number of second-stage variables are likely to benefit most from this implementation.

The procedure for solving the system $M\Delta t = \psi$ can be now summarized as follows:

1. (Solve $Sp = \psi$)
   (a) Solve $S_0 p_0 = \psi_0$.
   (b) Form, for each $l \in \Omega$, S'_l, S''_l, and S'''_l (Eqs. (16)–(18)).
   (c) Solve, for each $l \in \Omega$, $S'_l p'_l = \psi'_l - S''_l \psi''_l$ for p'_l (Eq. (14)).
   (d) Compute, for each $l \in \Omega$, p''_l using (15).
2. (Solve $Gq = V^T p$)
   (a) Solve, for each $l \in \Omega$, $S'_l (u_l)_j = (S''_l)_j$ for $(u_l)_j$, $j = 1, \ldots, n_2$ to form R'_l (Eq. (19)) and compute R'''_l (Eq. (21)).
   (b) Solve, for each $l \in \Omega$, $S'_l (v_l)_i = (T_l)_i$ for $(v_l)_i$, $i = 1, \ldots, n_1$, to form R''_l (Eq. (20)) and to obtain R''''_l (Eq. (22)). Thus, compute the columns of the matrix $S_l^{-1}\bar{T}_l$.
   (c) Multiply, for each $l \in \Omega$, $(T_l^T)(u_l)_j$, $i = j, \ldots, n_2$ and $(T_l^T)(v_l)_i$, $i = 1, \ldots, n_1$ to form $\bar{T}_l^T S_l^{-1} \bar{T}_l$ (Eq. (23)). Form H (Eq. (10)). Compute π' and π''.
   (d) Solve $Hu = \pi'$ for u and set $v = \pi'' + \bar{A}_0 u$ (Eq. (25)).
   (e) Form L by solving $(H)w_i = (\bar{A}_0^T)_i$ for w_i, $i = 1, \ldots, m_1$ and by setting $L = -\bar{A}_0[w_1, \ldots, w_{m_1}]$.
   (f) Solve $Lq_2 = -v$ for q_2, then solve $Hq_1 = \pi' - \bar{A}_0^T q_2$ for q_1 (Eqs. (25) and (26)).
3. (Solve $Sr = Uq$)
   (a) Set $r_0 = \bar{A}_0 q_1 + q_2$ and, for each $l = 1, \ldots, N$, solve $S'_l r'_l = T_l q''_1 + S''_l q'_1$ for r'_l (Eq. (27)).
   (b) Compute, for each $l = 1, \ldots, N$, r''_l (Eq. (28)).
4. (Form $\Delta t = p - r$)
   Set $\Delta t_0 = p_0 - r_0$, and, for each $l = 1, \ldots, N$, set $\Delta t_l = p_l - r_l$.

The procedure described above is well suited for parallel implementation since most of the steps involve individual scenarios, and, thus, the computational workload can be easily splitted among processors by using trivial scenario based decomposition.

3. The effects of restriction on recourse solution

In this section, we illustrate the effect of the introduction of robustness conditions on the variability of the recourse solutions across scenarios. In particular, we show how restriction

Table 1. Size charateristics of test problems.

	Original problem (SLP)				Extended problem (RRSP)			
	First stage		Second stage		First stage		Second stage	
Problem	Rows	Columns	Rows	Columns	Rows	Columns	Rows	Columns
scagr7	15	20	38	40	15	60	78	80
scsd8	10	70	20	140	10	210	160	280
sctap1	30	48	60	96	30	144	156	192
scrs8	28	37	28	38	28	75	66	76

limits the dispersion of the recourse solutions, while it causes an increase in the objective function value. It is beyond the scope of this paper to investigate how to identify the acceptable parameters, since their choice is problem dependent. Our objective is to illustrate the corresponding increase in the optimal objective value when a considerable increase is imposed on the dispersion of the recourse variables (i.e. very small value of ϵ).

A detailed assessment of restricted recourse models is given in [20]. Here we illustrate the behaviour on a set of test problems from the SLP library of Holmes [12]. The same problems are used in the next section to analyze the computational efficiency of our matrix factorization technique. In these problems the uncertain parameters are the right-hand-side vectors h of the second-stage constraints.

In the following tables we report the size characteristics of the test problems. In particular, Table 1 describes the size of the first- and second-stage problems for both the original SLP and the extended version of RRSP (after the cluster vector z and the slack variables s_l, $l \in \Omega$ are added).

Table 2 summarizes the sizes of the deterministic equivalent programs of the original and the extended versions of each test problem with an increasing number of scenarios.

In Table 3 we report the values of the recourse dispersion ρ and the euclidean norm of the virtual range ω before and after imposing robustness conditions. In addition, we report the optimal objective values of the objective function $f_{(SLP)}$ and $f_{(RRSP)}$ in both the original and the restricted recourse cases.

On the basis of these values, the relative increase ϵ_f in the objective value is computed as follows:

$$\epsilon_f = \frac{f_{(RRSP)} - f_{(SLP)}}{f_{(SLP)}}.$$

We also report the relative reduction ϵ_ρ in the recourse dispersion defined as:

$$\epsilon_\rho = \frac{\rho_{(SLP)} - \rho_{(RRSP)}}{\rho_{(SLP)}}.$$

The results of Table 3 have been collected by using LOQO as optimizer (but any code could be used for this purpose) with a fixed maximum number of iterative restrictions $k = 50$ and a value of $\lambda = 1$.

Table 2. Size characteristics of the deterministic equivalent of the SLP and RRSP problems.

		Original problem		Extended problem	
Problem	Scenarios	Constraints	Variables	Constraints	Variables
scagr7.4	4	167	180	327	380
scagr7.8	8	319	340	639	700
scagr7.16	16	623	660	1263	1340
scagr7.32	32	1231	1300	2511	2620
scagr7.64	64	2447	2580	5007	5180
scsd8.4	4	90	630	650	1330
scsd8.8	8	170	1190	1290	2450
scsd8.16	16	330	2310	2570	4690
scsd8.32	32	650	4550	5130	9170
scsd8.64	64	1290	9030	10250	18130
sctap1.4	4	270	432	654	912
sctap1.8	8	510	816	1278	1680
sctap1.16	16	990	1584	2526	3216
sctap1.32	32	1950	3120	5022	6288
sctap1.64	64	3870	6192	10014	12432
scrs8.4	4	140	189	292	379
scrs8.8	8	252	341	556	683
scrs8.16	16	476	645	1084	1291
scrs8.32	32	924	1253	2140	2507
scrs8.64	64	1820	2469	4252	4939

As shown in Table 3 and in figures 1–4, the reduction in the dispersion of recourse solutions and the consequent increase in the objective value depend on the problem and its flexibility to achieve robust solutions. Three cases can be distinguished here.

In the first case (such as `scagr7`, `sctap1`), we observe that the restricted recourse solution is characterized by a very limited increase of the objective value and a considerable reduction of the recourse dispersion. In the second case (see, for example, `scrs8`), the reduction of the recourse dispersion is payed for by a considerable increase of the objective value. In this case, it is up to the user to balance the required dispersion reduction with the acceptable increase of objective value. In the last case, the problem exhibits inflexibility in its solution leading to infeasibility when robustness conditions are introduced in the model (this behaviour was observed in some problems such as `scfmx1` [12]).

4. Computational results

We have implemented a *C* code (hereafter referred to as RRSP) for the restricted recourse procedure that takes advantage of both the dual block-angular structure of the constraint matrix and of the structure of the matrices $\bar{W}$ and $\bar{T}$ arising in the (RRSP) model.

Table 3. Effect of the restricted recourse on the objective function and the dispersion value.

	Problem SLP			Problem RRSP				
Problem	$\|\omega\|$	$\rho_{(SLP)}$	$f_{(SLP)}$	$\|\omega\|$	$\rho_{(RRSP)}$	$f_{(RRSP)}$	ϵ_f (%)	ϵ_ρ (%)
sctap1.4	2.77	1.36	280.50	0.011	0.005	299.77	6.86	99
sctap1.8	27.07	13.43	360.50	0.308	0.130	393.38	9.12	99
sctap1.16	27.65	13.80	359.00	0.414	0.158	389.59	8.52	98
sctap1.32	27.88	13.91	354.00	0.583	0.208	382.84	8.14	98
sctap1.64	42.09	14.03	344.00	1.141	0.347	370.53	7.71	97
scagr7.4	800.72	338.82	−832738.00	5.724	2.447	−832051.70	0.08	98
scagr7.8	801.86	335.61	−832737.80	5.567	2.372	−832051.60	0.08	99
scagr7.16	800.87	306.15	−832738.10	5.567	2.118	−832051.60	‘0.08	98
scagr7.32	808.61	237.46	−832737.80	5.463	2.027	−832055.80	0.08	97
scagr7.64	802.93	222.09	−832737.80	5.424	2.016	−832055.70	0.08	97
scrs8.4	141.71	70.55	690.21	4.370	2.185	1296.97	87.90	96
scrs8.8	141.71	70.02	1122.97	4.452	2.093	1674.46	49.10	97
scrs8.16	141.71	52.83	879.22	4.452	1.771	1669.81	89.91	96
scrs8.32	141.71	52.01	834.61	4.452	1.591	1597.57	91.40	96
scrs8.64	141.71	52.07	669.54	4.453	1.668	1437.63	114.71	96
scsd8.4	0.70	0.35	15.50	0.417	0.184	17.94	15.74	47
scsd8.8	1.00	0.50	16.00	0.595	0.253	18.06	12.87	49
scsd8.16	1.00	0.47	15.99	0.582	0.241	17.86	11.69	48
scsd8.32	1.00	0.43	15.99	0.596	0.231	17.69	12.32	46
scsd8.64	1.00	0.36	15.84	0.615	0.206	17.16	8.33	42

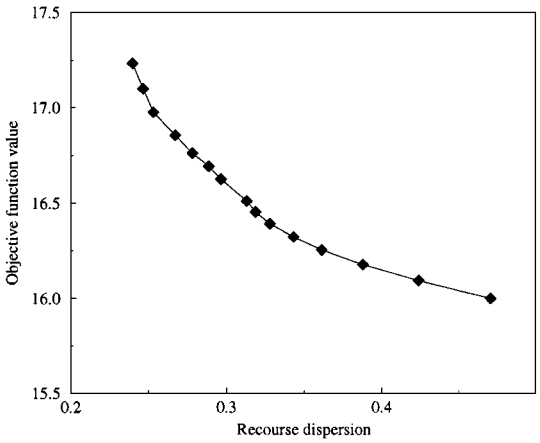

Figure 1. Effect of restricted recourse on the objective value: problem `scsd8`.

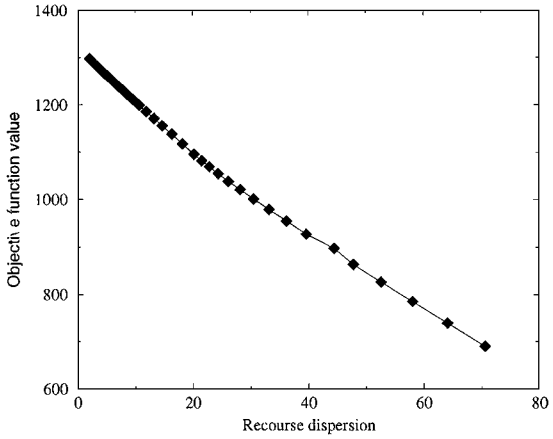

Figure 2. Effect of restricted recourse on the objective value: problem `scr8`.

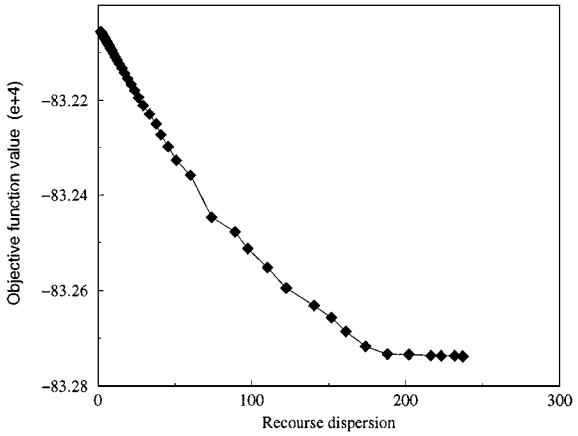

Figure 3. Effect of restricted recourse on the objective value: problem `scagr7`.

The restricted recourse procedure iteratively reduces the dispersion bounds on the recourse variables (by reducing the hypercube d) and calls the primal-dual path-following interior point solver to obtain a solution of the restricted recourse program. Thus, the RRSP procedure can be seen as a sequence of outer iterations in which we solve the (RRSP) problem by a sequence of interior point (inner) iterations. The number of outer iterations depends on the problem to be solved and the choice of the tolerance parameter ϵ.

In order to evaluate the performance of the RRSP code, we report in Table 4 the average number of inner iterations, and the average CPU time needed per outer iteration (with a small value of ϵ and $k = 50$). We also report in the same table, the results collected by using ROBOpT [21] and LOQO [19] as solvers; RRSP uses the same default accurancy parameters as defined in LOQO and ROBOpT.

Table 4. Average interior point iterations and solution time (in CPU seconds) per outer restriction iteration.

	ROBOpT		*RRSP*		*LOQO*	
Problem	Iterations	Time	Iterations	Time	Iterations	Time
scagr7.4	20.0	7.84	20.0	4.60	19.5	1.00
scagr7.8	22.0	15.53	20.4	7.92	20.5	3.27
scagr7.16	22.4	30.05	22.5	15.50	23.8	28.41
scagr7.32	23.0	62.40	25.0	33.50	26.4	954.17
scagr7.64	27.0	140.36	31.0	78.38	28.6	22.55
scsd8.4	13.0	71.20	13.0	14.85	17.9	3.33
scsd8.8	14.0	148.41	14.0	23.70	17.6	10.03
scsd8.16	13.0	268.41	13.0	37.70	17.1	58.86
scsd8.32	14.0	574.18	14.0	73.40	18.4	57.15
scsd8.64	15.0	1217.93	15.0	151.33	21.5	128.69
sctap1.4	13.5	41.64	12.8	14.93	16.5	1.71
sctap1.8	14.9	86.93	14.0	25.96	17.7	6.80
sctap1.16	16.0	181.60	16.0	50.32	19.8	62.09
sctap1.32	17.6	389.64	17.0	98.76	24.7	27.61
sctap1.64	21.1	879.48	20.0	227.22	27.3	61.45
scrs8.4	16.0	7.48	16.0	4.66	18.0	0.72
scrs8.8	18.0	14.08	18.0	7.66	18.3	1.92
scrs8.16	19.0	26.13	19.0	13.56	19.4	10.98
scrs8.32	23.0	59.55	23.0	28.10	21.6	5.48
scrs8.64	27.0	135.36	26.0	59.80	22.3	12.94

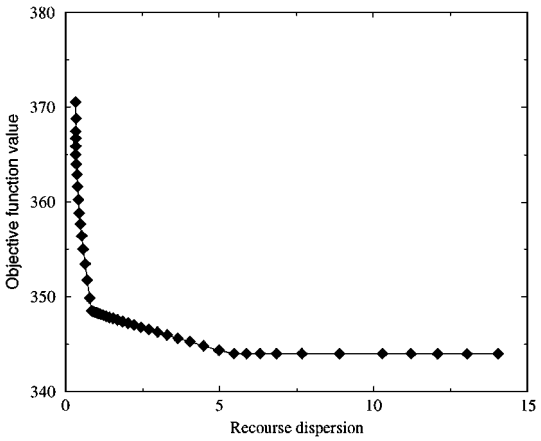

Figure 4. Effect of restricted recourse on the objective value: problem `scatp1`.

All these codes are implementations of the same interior point algorithm, and they differ only in the way they calculate the dual step. In particular, LOQO is a general purpose code and does not exploit the block structure of the constraint matrix. ROBOpT exploits the dual block-angular structure of SLP, while our code goes a step further than ROBOpT and it exploits the special structure of the blocks that is observed in the (RRSP) model.

All the computational experiments have been carried out on a DEC 4000 Alpha workstation with a 175 MHz Alpha processor and 64 Mb RAM. The `cc` compiler with the default optimization level was used.

The results in Table 4 and in figures 5–8, show that RRSP is competitive with LOQO and significantly faster than ROBOpT.

The results deserve some interpretation. LOQO is a general purpose solver that exploits sparsity, but it does not take advantage of the block structure.

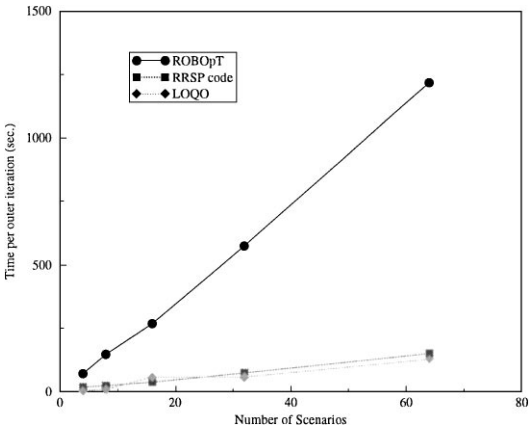

Figure 5. Solution time with increasing number of scenarios: problem `scsd8`.

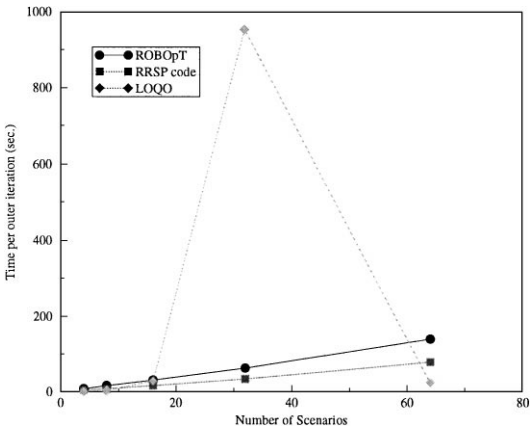

Figure 6. Solution time with increasing number of scenarios: problem `scagr7`.

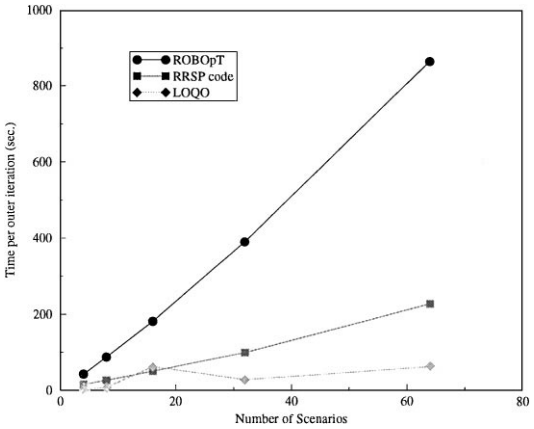

Figure 7. Solution time with increasing number of scenarios: problem `sctap1`.

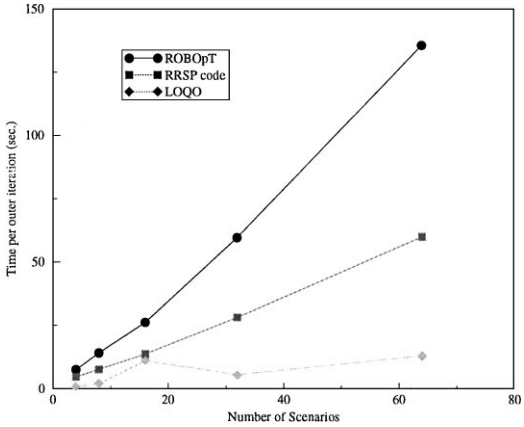

Figure 8. Solution time with increasing number of scenarios: problem `scrs8`.

From the results reported here it appears that the exploitation of sparsity by LOQO is efficient, and our method does not have remarkable advantage. However, our method enjoyes the appealing feature to be well suited for parallel implementation [3]. In addition, references [13] and [21] report very promising results with parallel implementations of the special matrix factorization procedure implemented in ROBOpT (i.e. exploiting the dual block-angular structure of two-stage stochastic SLP problems).

The advantage of RRSP over ROBOpT can be explained in the way they exploit structure. ROBOpT employs a matrix factorization technique similar to ours, but does so for two-stage problems, assuming that the first-stage coupling variables and the scenario blocks are relatively dense. The restricted recourse reformulation (RRSP model) creates larger blocks for both first- and second-stage constraints that are very sparse. ROBOpT does not exploit the added sparsity, while it suffers from the enlarged problem size. Our code exploits the

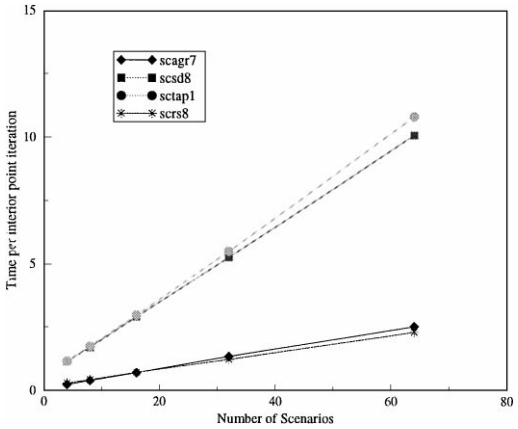

Figure 9. Computational performance of the RRSP code.

internal structure of the blocks in addition to the overall dual block angular structure of the problem, and therefore it is much more efficient than ROBOpT.

As shown in [21] one of the most attractive features of ROBOpT concerns the ability to implement a very efficient parallel version when solving large scale problems. A key charecteristic of parallel ROBOpT is its *scalability*.

The scalability of a parallel code is its ability to maintain efficiency at a fixed value by simultaneously increasing in constant proportion the number of processors and the size of the problem. Jessup et al. [13], show that this feature derives from the Birge-Qi factorization of the dual block-angular structure of the constraint matrix. Hence, a parallel implementation of RRSP seems very promising since it computes the dual step by using the same factorization procedure in ROBOpT taking further advantage of the special block structure of the matrices in the (RRSP) model.

Furthermore, it is clear from Table 4 that the number of interior point iterations is only marginally affected by the increasing problem size (i.e. the number of scenarios). The above observation is made even more relevant by the results in figure 9, which shows how the average execution time of an inner iteration in RRSP increases linearly with the number of scenarios.

For all test problems considered here, the RRSP code compares favorably especially with LOQO, which exhibits fluctuations in solution time for all the test problems (see figures 5–8).

We have mentioned that our code should be better suited for solving problems in which the number of second-stage columns is greater than the number of rows. This observation is confirmed by the computational results. RRSP exhibits its best relative performance in the solution of problem `scsd8` which has the largest proportion of second-stage variables. In this case, RRSP is very competitive with LOQO and is up to 8 times faster than ROBOpT.

5. Conclusion

We have proposed an algorithm to solve restricted recourse stochastic programs. The model is a classical two-stage stochastic programming problem that, in order to ensure robustness,

includes linear constraints on recourse decisions. We have shown that the considerable increase in the size of the problem, caused by the inclusion of the linearized dispersion constraints, is balanced by a suitable factorization that exploits both the block-angular structure of the constraint matrix and the special structure of the matrices $\bar{W}$ and $\bar{T}$ arising in the (RRSP) model. The extension of the matrix factorization method to exploit the inner structure of the blocks represents the main contribution of this paper with respect to previous works.

The computational results show that RRSP is significantly faster than the serial implementation of ROBOpT and is competitive with the state-of-the-art optimization software LOQO. A parallel implementation of the RRSP code seems to be very promising for the solution of large scale problems with a very large number of scenarios.

Acknowledgment

We would like to thank Stavros Zenios and Hercules Vladimirou for constructive suggestions and for providing the codes that have been the basis for our algorithmic developments.

References

1. E.D. Andersen, J. Gondzio, Cs. Mészáros, and X. Xu, "Implementation of interior point methods for large scale linear programming," in Interior Point Methods in Mathematical Programming, T. Terlaky (Ed.), Kluwer Academic Publishers, 1996, Ch. 6, pp. 189–252.
2. K.D. Andersen, "A modified Schur complement method for handling dense columns in interior point methods for linear programming," Technical Report, Dash Associates Ltd, Quinton Lodge, Binswood Avenue, Leamington Spa, Warwickshire CV32 5TH, UK, 1995.
3. P. Beraldi, L. Grandinetti, R. Musmanno, and C. Triki, "A parallel algorithm for solving two-stage stochastic linear programs with restricted recourse," Technical Report 06-98, Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, 87036 Rende, Italy, 1998. (Submitted for publication on Parallel Computing).
4. J.R. Birge and D.F. Holmes, "Efficient solution of two-stage stochastic linear programs using interior point methods," Computational Optimization and Applications, vol. 1, pp. 245–276, 1992.
5. J.R. Birge and F.V. 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, vol. 34, no. 12, 1990.
7. J. Czyzyk, R. Fourer, and S. Mehrotra, "Parallel solutions of multi-stage stochastic linear programs by interior-point methods," Technical Report, Department of Industrial Engineering and Management Sciences, Northwerstern University, 1994. (Draft).
8. G.B. Dantzig, "Planning under uncertainty using parallel computing," Annals of Operations Research, vol. 14, pp. 1–16, 1988.
9. G.B. Dantzig and P.W. Glynn, "Parallel processors for planning under uncertainty," Annals of Operations Research, vol. 22, pp. 1–21, 1990.
10. A. De Silva and D. Abramson, "A parallel interior point method for stochastic linear programs," Technical Report TR94-4, Shcool of Computing and Information Technology, Griffith University, Nathan Qld 4111, Australia, 1994.
11. A. De Silva and D. Abramson, "Parallel algorithm for solving stochastic linear programs," in Handbook of Parallel and Distributed Computing, A. Zomaya, (Ed.), McGraw Hill, 1996, pp. 1097–1115.
12. D. Holmes, "A collection of stochastic programming problems," Technical Report 94-11, Department of Industrial and Operations Engineering, University of Michigan, 1994.
13. E.R. Jessup, D. Yang, and S.A. Zenios, "Parrallel factorization of structured matrices arising in stochastic programming," SIAM J. on Optimization, vol. 4, no. 4, pp. 833–846, 1994.

14. I.J. Lustig, R.E. Marsten, and D.F. Shanno, "Interior point methods for linear programming," ORSA J. on Computing, vol. 6, pp. 1–14, 1994.
15. I.J. Lustig, J. Mulvey, and T.J. Carpenter, "Formulating stochastic programs for interior point methods," Operations Research, vol. 39, no. 5, pp. 757–770, 1991.
16. J.M. Mulvey, R.J. Vanderbei, and S. Zenios, "Robust optimization of large-scale systems," Operations Research, vol. 43, pp. 264–281, 1995.
17. A. Ruszczyński, "Interior point methods in stochastic programming," Working paper WP-93-8, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1993.
18. R.J. Vanderbei, "LOQO: An interior point code for quadratic programming," Technical Report SOR-94-15, School of Engineering and Applied Science, Department of Civil Engineering and Operations Research, Princeton University, 1994.
19. R.J. Vanderbei, "LOQO user's manual—version 2.27," Technical Report SOR-96-07, School of Engineering and Applied Science, Department of Civil Engineering and Operations Research, Princeton University, 1996.
20. H. Vladimirou and S.A. Zenios, "Stochastic linear programs with restricted recourse," European J. of Operations Research, vol. 101, pp. 172–192, 1997.
21. D. Yang and S.A. Zenios, "A scalable parallel interior point algorithm for stochastic linear programming and robust optimization," Computational Optimization and Application, vol. 7, pp. 143–158, 1997.