Regularized decomposition of stochastic programs: algorithmic techniques and numerical results

Andrzej Ruszczyński

WP-93-21
April 1993

Regularized decomposition of stochastic programs: algorithmic techniques and numerical results

Andrzej Ruszczyński

WP-93-21
April 1993

Regularized decomposition of stochastic programs: algorithmic techniques and numerical results

Andrzej Ruszczyński

Abstract

A finitely convergent non-simplex method for large scale structured linear programming problems arising in stochastic programming is presented. The method combines the ideas of the Dantzig-Wolfe decomposition principle and modern nonsmooth optimization methods. Algorithmic techniques taking advantage of properties of stochastic programs are described and numerical results for large real world problems reported.

Keywords: Stochastic Programming, Decomposition.

1. Introduction

A large class of operations research problems lead to linear programming models of the form

$$\begin{aligned} &\min\ c^T x \\ &Mx = r, \\ &x^{\min} \le x \le x^{\max}. \end{aligned} \tag{1.1}$$

In certain applications, however, some coefficients of the resource/demand vector r or some entries of the matrix M are uncertain. They can be modeled as random variables $r(\omega)$ and $M(\omega)$ with $\omega \in \Omega$, where $(\Omega, \mathcal{B}, P)$ is a probability space, but then the constraints in (1.1),

$$M(\omega)x = r(\omega), \quad \omega \in \Omega,$$

become prohibitively restrictive and usually impossible to satisfy for all realizations of the random entries.

One of modeling approaches to such a situation is the extension of (1.1) to a *stochastic programming problem with recourse.* We introduce recourse decisions (corrective activities) y_ω which can be taken after the realizations of the random entries are known, so as to fulfill the problem constraints.

To be more specific, let us split the constraints into the deterministic and the random parts:

$$M(\omega) = \begin{bmatrix} A \\ T_\omega \end{bmatrix}, \ r(\omega) = \begin{bmatrix} b \\ h_\omega \end{bmatrix},$$

where A is an $m_1 \times n_1$ matrix and $b \in R^{m_1}$, T_ω is an $m_2 \times n_1$ random matrix and d_ω is an m_2 -dimensional random vector over a probability space $(\Omega, \mathcal{B}, P)$. Finally, let $y_\omega \in R^{n_2}$ be the correction vector, $q \in R^{n_2}$ be the correction costs, and let W be an

$m_2 \times n_2$ matrix describing our capabilities of corrections. The problem can be now reformulated as follows:

$$\min \left[c^T x + \int q^T y_\omega P(d\omega)\right]$$

subject to

$$\begin{aligned} Ax &= b \\ T_\omega x + W y_\omega &= d_\omega,\ \omega \in \Omega, \\ x^{\min} \le\ &x \le x^{\max}, \\ y^{\min} \le\ &y_\omega \le y^{\max},\ \omega \in \Omega. \end{aligned} \tag{1.2}$$

In other words, we have to choose x so as to minimize the first stage cost $c^T x$ and the expected correction cost $\int q^T y_\omega P(d\omega)$, under the condition that the correction $W y_\omega$ compensates the shortage/surplus $d_\omega - T_\omega x$.

It should be stressed that the above reformulation is not just a formal trick to pose the problem correctly, but it reflects many real-life situations where shortage/surplus may occur, but they are connected with additional costs.

While there seems to be no doubt as to theoretical advantages of using models of form (1.2), their solution is much more difficult than for underlying deterministic models (1.1). For simplicity we shall focus our attention here on problems with discrete distributions; approximation of general distributions by discrete ones in stochastic programming is discussed in [2] and [8].

Let Ω be finite, $\Omega = \{1, 2, \dots, L\}$, and let the realizations $(T_\omega, d_\omega), w \in \Omega$, be attained with probabilities $p_\omega > 0$, $(\sum_{\omega \in \Omega} p_\omega = 1)$. Then (1.2) can be rewritten as a large linear programming problem

$$\begin{array}{lcccccccccl} \min & c^T x & + & p_1 q^T y_1 & + & p_2 q^T y_2 & + & \dots & + & p_L q^T y_L & \\ & Ax & & & & & & & & & = & b, \\ & T_1 x & + & W y_1 & & & & & & & = & d_1, \\ & T_2 x & & & + & W y_2 & & & & & = & d_2, \\ & \vdots & & & & & & \ddots & & & & \vdots \\ & T_L x & & & & & & & + & W y_L & = & d_L, \end{array} \tag{1.3}$$

$$x^{\min} \le x \le x^{\max},$$

$$y^{\min} \le y_\omega \le y^{\max},\ \omega = 1, 2, \dots, L.$$

There are several reasons for studying (1.3) thoroughly.

First of all, it is the remarkable size that makes this problem difficult from the practical point of view. Stochastic programs are usually extensions of deterministic linear models, so we should think of T having size of a constraint matrix in a typical linear program, and this size is multiplied in (1.3) by the number L of realizations of (T_ω, d_ω). For nontrivial problems with many independent random factors causing the stochasticity of the entries of T_ω and d_ω, L must be sufficiently large to reflect this randomness in our model. As a result, the dimension of (1.3) may go in hundreds of thousands.

Another difficulty is the possibility of ill-conditioning of (1.3). If the number of first stage activities x in the optimal basis exceeds $m_1 + m_2$, then similarity of the realizations T_ω, $\omega \in \Omega$, implies that the columns corresponding to these activities are close to being linearly dependent (for $T_\omega = T$ singularity would occur).

A very rich literature is devoted to solution methods for problems of form (1.3) or their duals. The first group of methods are variants of the simplex method which take advantage of the structure of the constraint matrix of (1.3) to construct compact representations of the basis inverse and to improve pivotal strategies (cf, e.g., [25]). The second group are linear decomposition methods coming down from the famous decomposition principle of Dantzig and Wolfe [5] (see [4, 23]). Finally, there is a possibility of reformulating (1.3) as a nonsmooth optimization problem and applying to it general non-differentiable optimization algorithms.

In the regularized decomposition (RD) method proposed for general large scale structured linear programming problems in [15] we combine the last two approaches: the problem is stated as a nonsmooth optimization problem, but for the purpose of solving it we modify the general bundle method of [9] by taking full advantage of problem's structure. As a result, a finitely convergent non-simplex method for large structured linear programs can be obtained.

The main purpose of our paper is to specialize the regularized decomposition method to stochastic problems with recourse. We present various techniques developed for exploiting specific structural properties of stochastic programs and for mitigating the computational effort and the effects of ill-conditioning. We also describe results of some computational tests showing that the method is capable of solving stochastic programs of considerable size.

2. The outline of the RD method

It can be readily seen that if x is fixed in (1.3) then minimization with respect to $y_1, y_2, .., y_L$ can be carried out separately. This leads to the following two-stage formulation

$$\min \left[F(x) = c^T x + \sum_{\omega \in \Omega} p_\omega f_\omega(x) \right] \tag{2.1}$$

subject to

$$x \in X_0 = \{x : \; Ax = b, \; x^{\min} \le x \le x^{\max}\}, \tag{2.2}$$

$$x \in X_\omega \quad \text{for} \quad \omega \in \Omega, \tag{2.3}$$

with $f_\omega(x)$ defined as the optimal value of the second stage problem:

$$f_\omega(x) = \min \left\{ q^T y \mid Wy = d_\omega - T_\omega x, \; y^{\min} \le y \le y^{\max} \right\}, \tag{2.4}$$

and with

$$X_\omega = \{x : \; f_\omega(x) < \infty\}.$$

We introduce condition (2.3) explicitly to the problem formulation, because we are going to use separate approximations of f_ω and of their domains X_ω. The functions

f_ω are convex and polyhedral and the sets X_ω are convex closed polyhedra [24]. Thus (2.1)-(2.3) can in principle be solved by a method for piecewise linear problems or by a general algorithm for constrained nonsmooth optimization. Although the pieces of f_ω and the facets of X_ω are not given explicitly, it is possible to extract from the subproblems (2.4) at any x^k information about the piece $(\alpha_\omega^k, g_\omega^k)$ of f_ω active at x^k (an *objective cut*) or information about a constraint $(\bar{\alpha}_\omega^j, \bar{g}_\omega^j)$ defining X_ω and violated at x^k (a *feasibility cut*). The pieces (cuts) collected so far can be used to construct lower approximations of the functions f_ω,

$$f_\omega(x) \geq \bar{f}_\omega(x) = \max\{\alpha_\omega^j + (g_\omega^j)^T x,\ j \in J_\omega\},$$

and outer approximations of the sets X_ω

$$X_\omega \subset \bar{X}_\omega = \{x :\ \bar{\alpha}_\omega^j + (\bar{g}_\omega^j)^T x \leq 0,\ j \in \bar{J}_\omega\};$$

J_ω and $\bar{J}_\omega$ are some selected sets of cuts.

Crucial questions that arise in this respect are the following:

- how the successive points x^k are generated?
- how the cuts at x^k are constructed?
- how the approximations $\bar{f}_\omega$ and $\bar{X}_\omega$ are updated?

The most natural method for generating successive points x^k is to solve the linear approximation of (2.1)-(2.3) constructed on the basis of currently available information:

$$\min_{x \in \bar{X}} \left[\bar{F}(x) = c^T x + \sum_{\omega \in \Omega} p_\omega \bar{f}_\omega(x) \right], \tag{2.5}$$

where

$$\bar{X} = X_0 \cap \left(\bigcap_{\omega \in \Omega} \bar{X}_\omega \right).$$

After solving (2.5) we obtain cuts at the current solution, add them to the sets of cuts used previously, solve (2.5) again, etc..

Instead of constructing separate approximations for all f_ω in (2.5), we can also work with a piecewise linear approximation of their weighed sum $f(x) = sum_{\omega \in \Omega} p_\omega f_\omega(x)$, as it was originally suggested in the L-shaped method of Van Slyke and Wets [23]. This would mean constructing objective cuts for f by averaging (with the weights p_ω) the objective cuts for f_ω.

The cutting-plane approach, however, has well-known drawbacks. Initial iterations are inefficient. The number of cuts increases after each iteration and there is no reliable rule for deleting them. The master problem (2.5) is sensitive with respect to changes in the set of cuts and its conditioning is getting worse when approaching the solution.

For these reasons in the RD method the linear master (2.5) is modified by adding to its objective a quadratic regularizing term:

$$\min_{x \in \bar{X}} \left[\eta(x) = \frac{1}{2\sigma} \|x - \xi^k\|^2 + c^T x + \sum_{\omega \in \Omega} p_\omega \bar{f}_\omega(x) \right]. \tag{2.6}$$

Here ξ^k is a certain regularizing point, and σ is a positive parameter. It turns out that this modification stabilizes the master and makes it possible to delete inactive cuts so that the size of (2.6) is limited. On the other hand, although we replace the linear master by a quadratic one, it is possible to arrange the algorithm for changing the regularizing points ξ^k in such a way that the whole method retains the finite convergence property of the linear approach. We shall present this algorithm in section 3.

Again, we could work here with a convex piecewise linear approximation of the expected second stage cost $f(x) = sum_{\omega \in \Omega} p_\omega f_\omega(x)$, as in general algorithms of [9]. We use here more complicated separate approximations, because aggregation of cuts may slow down convergence of the method, as it was observed in [15] (this idea was also analyzed in [1]). We shall show n section 4 that it is possible to efficiently process separate approximations for each f_ω by exploiting the structural properties of (2.6).

Let us now pass to the question of obtaining objective and feasibility cuts. To discuss this matter in more detail we shall fix our attention on a specific method for solving the subproblems (2.4). Since for a given $x = x^k$ we have to solve (2.4) for all $\omega \in \Omega$ and only the right hand side in (2.4) varies, a reasonable choice is the dual simplex method. Upon termination, two cases may occur: optimality or infeasibility (dual unboundedness).

Objective cuts

Suppose that at $x = x^k$ problem (2.4) is solvable with an optimal basis B and simplex multipliers $\pi^T = q_B^T B^{-1}$. Then for any $x = x^k + \Delta x$, since B remains dual feasible, we get $f_\omega(x) \geq f_\omega(x^k) - \pi^T T_\omega \Delta x$. Hence, in the objective cut

$$\alpha_\omega^k + (g_\omega^k)^T x \leq f_\omega(x) \text{ for all } x \tag{2.7}$$

we have

$$g_\omega^k = -T_\omega^T \pi,\ \alpha_\omega^k = f_\omega(x^k) - (g_\omega^k)^T x^k. \tag{2.8}$$

Feasibility cuts

In case of dual unboundedness (for some $x = x^k$) we stop at a certain dual feasible basis B for which there is a basic variable y_{Br} whose value $\tilde{y}_{Br}$ is out of its bounds (e.g. $\tilde{y}_{Br} > y_{Br}^{\max}$) and which cannot be moved towards the feasibility interval by feasible changes of nonbasic variables. Clearly, $x^k \notin X_\omega$ in this case. On the other hand, for any $x \in X_\omega$ the value of y_{Br} with the basis B must not exceed $y_{Br}^{\max}$, because we shall not be able to decrease it by feasible changes of nonbasics. Denoting by π^T the r-th row of B we obtain the following estimate

$$X_\omega \subset \{x :\ \tilde{y}_{Br} - \pi^T T_\omega (x - x^k) \leq y_{Br}^{\max}\}.$$

If $\tilde{y}_{Br} < y_{Br}^{\min}$ we define π to be the negative of the r-th row of B and obtain

$$X_\omega \subset \{x :\ \tilde{y}_{Br} + \pi^T T_\omega (x - x^k) \geq y_{Br}^{\min}\}.$$

Consequently, at any $x = x^k$ for every $\omega \in \Omega$ for which (2.4) is not solvable we get a feasibility cut

$$\bar{\alpha}_\omega^k + (\bar{g}_\omega^k)^T x \le 0 \quad \text{for all} \quad x \in X, \tag{2.9}$$

where

$$\bar{g}_\omega^k = -T_\omega^T \pi, \quad \bar{\alpha}_\omega^k = \beta_\omega - (\bar{g}_\omega^k)^T x^k, \tag{2.10}$$

and β_ω is the distance to the violated bound.

It is not difficult to observe that there can be only finitely many objective and feasibility cuts, because the number of possible bases in (2.4) is finite.

3. The logic of the RD method

The method generates two sequences: a sequence ξ^k of regularizing points and a sequence x^k of trial points. Each iteration of the method consists in updating and solving the regularized master problem (2.6), which can be equivalently stated as follows. We introduce variables v_ω , $\omega \in \Omega$, to represent $\hat{f}_\omega(x)$ by inequalities involving objective cuts:

$$(g_\omega^j)^T x + \alpha_\omega^j \le v_\omega, \ j \in J_\omega^k, \ \omega \in \Omega.$$

Using explicit formulations of feasibility cuts (2.9) and putting all the cuts together we can rewrite the master (2.6) in a more compact form

$$\min \left[\frac{1}{2\sigma}\|x - \xi^k\|^2 + c^T x + p^T v\right] \tag{3.1}$$

subject to

$$(G^k)^T x + a^k \le (E^k)^T v. \tag{3.2}$$

The constraints (3.2) (so called *committee*) comprise in general three groups of cuts:

(a) selected direct constraints from (2.2);

(b) selected feasibility cuts (2.9)-(2.10) collected at some previously generated trial points x^j, $j \in \bar{J}_\omega^k \subset \{0, 1, .., k\}$, $\omega \in \Omega$;

(c) selected objective cuts (2.7)-(2.8) collected at some previously generated trial points x^j, $j \in J_\omega^k \subset \{0, 1, .., k\}$, $\omega \in \Omega$.

Thus each column of the matrix E^k in (3.2) is either a null vector, if the cut is of class (a) or (b), or the l-th unit vector if the cut belongs to class (c) and approximates $f_l(x)$. There are never more than $n + 2L$ cuts in the committee.

There are two phases of the method. At Phase 1 we seek a point which satisfies (2.2)-(2.3). It serves then as a starting point for Phase 2, where we aim at solving (2.1)-(2.3). Since the Phase 1 algorithm is in fact a special case of the main Phase 2 method, we shall now describe in detail the latter.

Let ξ^0 be a starting point satisfying (2.2)-(2.3) and let the initial committee be given by

$$G^0 = [g_\omega]_{\omega \in \Omega}, \ a^0 = [\alpha_\omega]_{\omega \in \Omega}, \ E^0 = I,$$

with $(g_\omega, \alpha_\omega)$ describing objective cuts at ξ^0 for $\omega \in \Omega$. The committee may (but need not) contain also some constraints from (2.2) and some feasibility cuts of form (2.9) inherited from Phase 1.

ALGORITHM 1 (the top algorithm of the RD method; $0 < \gamma < 1$)

Step 1. Solve the master at ξ^k getting a trial point x^k and objective estimates v^k and calculate $\hat{F}^k = c^T x^k + p^T v^k$. If $\hat{F}^k = F(\xi^k)$, then STOP (optimal solution found); otherwise continue.

Step 2. Delete from the committee some members inactive at (x^k, v^k) so that no more than $n_1 + L$ members remain.

Step 3. If x^k satisfies (2.2) then go to 4. Otherwise add to the committee no more than L violated constraints, set $\xi^{k+1} = \xi^k$, increase k by 1 and go to Step 1.

Step 4. For $\omega \in \Omega$ solve (2.4) at x^k and :

(a) if the constraints of (2.4) are inconsistent then append to the committee feasibility cut (2.9)-(2.10);

(b) else if $f_\omega(x^k) > v_\omega^k$ then append to the committee objective cut (2.7)-(2.8).

Step 5. If all subproblems were solvable then go to Step 6, else set $\xi^{k+1} = \xi^k$ and got to Step 7.

Step 6. If $F(x^k) = \hat{F}^k$ or $F(x^k) \leq \gamma F(\xi^k) + (1-\gamma)\hat{F}^k$ and exactly $n + L$ members were active at (x^k, v^k) then set $\xi^{k+1} = x^k$; otherwise set $\xi^{k+1} = \xi^k$.

Step 7. Increase k by 1 and go to Step 1.

If the starting point is not feasible we can put into the starting committee artificial cuts $v_\omega \geq -C$, where C is a very large constant, for all the functions $f_\omega(x)$ for which objective cuts are not yet available, and set $F(\xi) = +\infty$.

It follows from the theory developed in [15] that after finitely many steps the method either discovers inconsistency in (2.1)-(2.3) or finds an optimal solution. Our proof for the case $p = [\ 1\ 1\ \dots\ 1\ ]$, $\sigma = 1$ can be trivially extended to arbitrary $p > 0$, $\sigma > 0$.

It is worth mentioning that the finite convergence property does not require any additional non-degeneracy assumptions typical for general cutting plane methods and bundle methods (see [21, 9]).

Few comments concerning implementation of Algorithm 1 are in order. The number of committee members may vary between L and $n_1 + 2L$, but in fact only cuts generated at Step 4 need be stored (see section 4). If the number of linear constraints in (2.2) is large, various strategies can be used at Step 3, similarly to pricing strategies in linear programming. Finally, one can control the penalty coefficient σ on line, increasing it whenever whenever steps are too short, and decreasing σ when $F(x^k) > F(\xi^k)$ (see section 7).

For the purpose of solving the regularized master problem we suggested in [15] a special active set strategy, which we shortly remind below. At each iteration we select a subset of constraints (3.2), defined by some submatrices G, a and E of G^k, a^k and E^k, such that E is of full row rank (at least one cut for each f_ω) and $\begin{bmatrix} G \\ E \end{bmatrix}$ is of full column rank. We treat these cuts as equalities and solve the resulting equality constrained subproblem by solving the system of its necessary and sufficient conditions of optimality

$$E\lambda = p, \tag{3.3}$$

$$E^T v + \sigma G^T G\lambda = G^T(\xi - \sigma c) + a \tag{3.4}$$

(for simplicity we drop the superscript k). The solution is given by

$$x = \xi - \sigma(c + G\lambda). \tag{3.5}$$

In the method we alter the active set by adding or deleting cuts until the solution is optimal for (3.1)-(3.2).

ALGORITHM 2 (solving the RD master)

Step 0. Determine the initial active set by choosing one constraint from each block $\omega \in \Omega$, so that $E = I$ ($L \times L$ identity). Set $\lambda = p$.

Step 1. Recover x and v from (3.4)-(3.5). If (3.2) holds, then stop (solution found). Otherwise choose from (3.2) any violated constraint such that $g^T x + \alpha - e^T v > 0$, and continue.

Step 2. Replace G, E and a with $\begin{bmatrix} G & g \end{bmatrix}$, $\begin{bmatrix} E & e \end{bmatrix}$ and $\begin{bmatrix} a \\ \alpha \end{bmatrix}$. If the new $\begin{bmatrix} G \\ E \end{bmatrix}$ has full column rank, replace λ with $\begin{bmatrix} \lambda \\ 0 \end{bmatrix}$ and go to Step 4; otherwise continue.

Step 3. Find multipliers z such that $\begin{bmatrix} G \\ E \end{bmatrix} \begin{bmatrix} z \\ -1 \end{bmatrix} = 0$ (i.e. the new cut $\begin{bmatrix} g \\ e \end{bmatrix}$ is a linear combination of the previous active constraints with coefficients z). Replace λ with $\begin{bmatrix} \lambda - \theta z \\ \theta \end{bmatrix}$ finding the largest $\theta \geq 0$ for which λ remains nonnegative, and an index i such that $\theta = \lambda_i / z_i$. If such a finite θ does not exist, then stop (inconsistent constraints). Otherwise delete the i-th active constraint.

Step 4. Solve (3.3)-(3.4) getting multipliers $\bar{\lambda}$. If $\bar{\lambda} \geq 0$, replace λ by $\bar{\lambda}$ and go to Step 1. Otherwise continue.

Step 5. Replace λ with $\lambda + \theta(\bar{\lambda} - \lambda)$, finding the largest $0 \leq \theta < 1$ for which λ remains nonnegative, and an index i such that $\theta = \lambda_i / (\lambda_i - \bar{\lambda}_i)$. Delete the i-th active constraint and go to Step 4.

Step 0 (cold start) need be carried out only at the first iteration of Algorithm 1. At subsequent iterations we start from the active set that was optimal at the previous iteration (i.e. from Step 1 of Algorithm 2).

4. Critical scenarios and reduced cuts

The system of necessary conditions of optimality (3.3)-(3.4) must be solved any time the active set is altered or ξ is changed by Algorithm 1. The number of equations in (3.3) is equal to the number of blocks L, whereas the size of (3.4) is equal to the number of active cuts, which is $m + L$ with some $0 \leq m \leq n_1$. Thus the total size is $2L + m$: quite a large number when many realizations (scenarios) are taken into account. We need a special approach to this problem if we want to make our method competitive with standard techniques.

The key observation is that there must be at least one cut for each f_ω in the active set (E has full row rank). With the number of active cuts bounded by $L + n_1$, there may be at most m ($m \leq n_1$) scenarios that are represented in the active set by more than one cut. We shall call them *critical scenarios.* The scenarios that are not critical have purely linear approximations. We shall exploit it in the numerical procedure to achieve substantial simplifications.

Formally, we select for each scenario one active cut and call it a *basic cut.* The basic cuts form the system

$$G_B^T x + a_B = v \tag{4.1}$$

of dimension L. Other active cuts, which occur only for critical scenarios will be called *nonbasic.* Rearranging the order of cuts so that the basic cuts appear first we shall have $E = \begin{bmatrix} I & N \end{bmatrix}$. The nonbasic cuts form the system

$$G_N^T x + a_N = N^T v \tag{4.2}$$

of dimension m. Subtracting (4.1) multiplied by N^T from (4.2) yields *reduced cuts*:

$$\hat{G}^T x + \hat{a} = 0, \tag{4.3}$$

where

$$\hat{G} = G_N - G_B N, \tag{4.4}$$

$$\hat{a} = a_N - N^T a_B. \tag{4.5}$$

In other words, each critical scenario is represented by the differences between its nonbasic cuts and its basic cut.

Next, partitioning λ into $\begin{bmatrix} \lambda_B \\ \lambda_N \end{bmatrix}$, we can use (4.1) to eliminate v and λ_B from (3.3)-(3.4), which yields

$$\sigma \hat{G}^T \hat{G} \lambda_N = \hat{G}^T x_B + \hat{a}, \tag{4.6}$$

where x_B is the solution implied by basic cuts alone,

$$x_B = \xi - \sigma(c + G_B p). \tag{4.7}$$

The other unknowns in (3.3)-(3.4) are defined by

$$\lambda_B = p - N\lambda_N,$$

$$x = x_B - \sigma \hat{G} \lambda_N. \tag{4.8}$$

In this way the large system of necessary and sufficient conditions of optimality has been reduced to a relatively small system (4.6) whose order m never exceeds the number of first stage variables n_1, independently of the number of realizations taken into account. This is a substantial improvement over the LP formulation (1.3).

However, the other difficulty typical for stochastic programs, ill-conditioning, still remains. Its effect on our approach is that the reduced cuts (4.3) can be almost linearly dependent. Indeed, from (2.8) we see that a reduced objective cut is of the form

$$\hat{g}_\omega = T_\omega^T(\pi_{B\omega} - \pi_\omega), \tag{4.9}$$

where $\pi_{B\omega}$ and π_ω are the vectors of simplex multipliers in block ω that generated the two cuts forming $\hat{g}_\omega$. The sets of possible multiplier vectors (basic solutions of duals to (2.4) are the same for each ω, so $\pi_{B\omega} - \pi_\omega$ may be identical for many reduced cuts. Then, by (4.9), the similarity of T_ω will cause similarity of the corrsponding reduced cuts.

An established approach to such difficulties is the use of orthogonal factorization (cf. e.g. [3])

$$\hat{G} = QR, \tag{4.10}$$

where $Q^TQ = I$ and R is upper triangular. Then, defining an auxiliary vector w by

$$R^Tw = \hat{G}^Tx_B + \hat{a} \tag{4.11}$$

we can reduce (4.6) to a triangular system

$$\sigma R\lambda_N = w. \tag{4.12}$$

We can also use (4.10)-(4.12) to update the solution of (4.6) each time the active set is revised or ξ is changed by Algorithm 1. This can be carried out by appropriate modifications of R, w, x_B and x, without storing Q explicitly. Since the set of critical scenarios may change, we need to develop a number of special transformations to take full advantage of our reduced formulation. We shall describe all these operations in detail in the next section.

5. Elementary transformations

Efficient solution of the master problem is the key to the good performance of the whole method. In this section we show how we can quickly update the solution of (4.12) when the active set changes.

Let us at first observe that we can decrease the dimension of (4.12) by treating in a special way simple bounds on variables appearing in the active set. Assuming for simplicity that bounds on variables $x_1, x_2, .., x_k$ are active and putting them in front of $\hat{G}$ and $\hat{a}$ we see that

$$\hat{G} = \begin{bmatrix} J & \hat{G}_1 \\ & \hat{G}_2 \end{bmatrix}, \quad \hat{a} = \begin{bmatrix} -J\beta \\ \hat{a}_2 \end{bmatrix}, \tag{5.1}$$

where β is the vector of active bounds and J is a diagonal matrix with $j_{ii} = 1$ if $\beta_i = x_i^{\max}$ and $j_{ii} = -1$ if $\beta_i = x_i^{\min}$. Other factorization elements can be split in a similar fashion:

$$Q = \begin{bmatrix} I & \\ & Q_2 \end{bmatrix}, \quad R = \begin{bmatrix} J & \hat{G}_1 \\ & R_2 \end{bmatrix}, \tag{5.2}$$

$$x_B = \begin{bmatrix} x_{B1} \\ x_{B2} \end{bmatrix}, \quad w = \begin{bmatrix} x_{B1} - \beta \\ w_2 \end{bmatrix} \tag{5.3}$$

with

$$\hat{G}_2 = Q_2 R_2, \tag{5.4}$$

and

$$R_2^T w_2 = (\hat{G}_2^T x_{B2} + \hat{a}_2 + \hat{G}_1^T \beta). \tag{5.5}$$

By splitting

$$\lambda_N = \begin{bmatrix} \lambda_{N1} \\ \lambda_{N2} \end{bmatrix}$$

we can reduce (4.12) to the system

$$\sigma R_2 \lambda_{N2} = w_2; \tag{5.6}$$

other components of λ_N follow from the form of R and w in (5.2) and (5.3):

$$\lambda_{N1} = J(x_{B1} - \hat{G}_1 \lambda_2 - \beta).$$

It is sufficient to store and transform x_{B2}, R_2 and w_2, which by many active bounds leads to substantial savings in both storage and computational expenses.

Adding a cut

Suppose that a new cut $g^T x + \alpha \le e^T v$ has to enter the active set. We compute the reduced cut

$$\begin{aligned} \hat{g} &= g - G_B e, \\ \hat{\alpha} &= \alpha - a_B^T e, \end{aligned}$$

and we split it into

$$\hat{g} = \begin{bmatrix} \hat{g}_1 \\ \hat{g}_2 \end{bmatrix}.$$

Next, we orthogonalize $\hat{g}_2$ with respect to Q_2 (which is not stored) by calculating

$$r = Q_2^T \hat{g}_2 = R_2^{-T} \hat{G}_2^T \hat{g}_2, \tag{5.7}$$

$$z = R_2^{-1} r, \tag{5.8}$$

$$q = \hat{g}_2 - Q_2 r = \hat{g}_2 - \hat{G}_2 z, \tag{5.9}$$

$$\rho = \|q\|. \tag{5.10}$$

Then

$$R_2^{\text{new}} = \begin{bmatrix} R_2 & r \\ & \rho \end{bmatrix}, \tag{5.11}$$

which corresponds to $Q_2^{\text{new}} = \begin{bmatrix} Q_2 & \frac{1}{\rho} q \end{bmatrix}$ (which is, of course, not stored).

By straightforward calculation of w_2 by (5.5) we get

$$w_2^{\text{new}} = \begin{bmatrix} w_2 \\ \nu \end{bmatrix} \tag{5.12}$$

with

$$\nu = \frac{1}{\sigma}(\hat{g}_2^T x_{B2} + \hat{\alpha} + \hat{g}_1^T \beta - r^T w).$$

By (5.11), (5.12) and (5.8) we can also update the solution of (5.6):

$$\lambda_{N2}^{\text{new}} = \frac{1}{\sigma}(R_2^{\text{new}})^{-1} w^{\text{new}} = \frac{1}{\sigma}\begin{bmatrix} R_2^{-1} w \\ 0 \end{bmatrix} - \frac{\nu}{\sigma\rho}\begin{bmatrix} R_2^{-1} r \\ -1 \end{bmatrix} = \begin{bmatrix} \lambda \\ 0 \end{bmatrix} - \frac{\nu}{\sigma\rho}\begin{bmatrix} z \\ -1 \end{bmatrix}.$$

Finally, the primal solution can also be updated; from (4.8), (5.4), (4.12) and (5.12) we get

$$x_2^{\text{new}} = x_{B2} - \sigma \hat{G}_2^{\text{new}} \lambda_{N2}^{\text{new}} = x_{B2} - Q_2^{\text{new}} w_2^{\text{new}} = x_2 - \frac{\nu}{\rho} q.$$

Consequently, the only time-consuming operation is the orthogonalization by (5.7)-(5.10).

Adding a bound

Suppose that a bound $x_j = x_j^{\max}$ has to be added to the active set. The first operations are essentially the same as in the case of adding a cut, but (5.7) simplifies a little because $\hat{g}$ is a unit vector:

$$r = R_2^{-T} s,$$

where s^T is the j-th row of $\hat{G}$. The other operations are identical.

After carrying out the whole augmentation procedure, we have now to re-arrange the order of active constraints so that the new bound will appear before the cuts, as in (5.1). We can do it by moving the last column of the new R_2 to the front (the bound before the cuts) and the last row of the new R_2 to the first position (the column $\frac{1}{\rho} q$ to the front of Q), which yields the matrix

$$\tilde{R}_2 = \begin{bmatrix} \rho & \\ r & R_2 \end{bmatrix}. \tag{5.13}$$

The upper triangular form of $\tilde{R}_2$ can be now restored by a sequence of Givens operations $P = P_{1,2} \cdots P_{1,m} P_{1,m+1}$ that act on rows 1 and i of $\tilde{R}_2$, for $i = m+1, m, \ldots, 2$ to annulate r:

$$P \begin{bmatrix} \rho & \\ r & R_2 \end{bmatrix} = \begin{bmatrix} 1 & s^T \\ 0 & R_2^{\text{new}} \end{bmatrix}. \tag{5.14}$$

(That s^T in (5.14) is the j-th row of $\hat{G}$ can be seen from (5.2)). It follows from (5.5) and the orthogonality of P that the new w can be calculated in parallel:

$$P \begin{bmatrix} \nu \\ w \end{bmatrix} = \begin{bmatrix} x_{Bj} - x_{Bj}^{\max} \\ w^{\text{new}} \end{bmatrix}, \tag{5.15}$$

because

$$\begin{bmatrix} \rho & \\ r & R_2 \end{bmatrix}^T \begin{bmatrix} \nu \\ w \end{bmatrix} = \begin{bmatrix} \rho & \\ r & R_2 \end{bmatrix}^T P^T P \begin{bmatrix} \nu \\ w \end{bmatrix} = \begin{bmatrix} 1 & s^T \\ 0 & R_2^{\text{new}} \end{bmatrix}^T P \begin{bmatrix} \nu \\ w \end{bmatrix}. \tag{5.16}$$

As a result, at a little extra cost of one sweep of Givens operations we avoided increasing the size of R_2.

Deleting a nonbasic cut

Let the dimension of R_2 be m. Deleting the j-th (say) cut of G_N is equivalent to deleting the j-th column of $\hat{G}_2$, the j-th element of $\hat{a}_2$ and the j-th column of R_2, which yields $\hat{G}_2^{\text{new}}$, $\hat{a}_2^{\text{new}}$ and an $m \times (m-1)$ matrix $\tilde{R}_2$. The upper triangular form of $\tilde{R}$ can now be restored by a sequence of Givens operations $P = P_{m-1,m} \cdots P_{j+1,j+2} P_{j,j+1}$ that act on rows i and $i+1$, for $i = j, \dots, m$, to annulate the subdiagonal entries $\tilde{r}_{i+1,i}$ of $\tilde{R}$. Analogously to (5.16) we can update in parallel w, getting:

$$P \begin{bmatrix} \tilde{R} & w \end{bmatrix} = \begin{bmatrix} R_2^{\text{new}} & w^{\text{new}} \\ & \nu \end{bmatrix}.$$

Deleting a basic cut

Suppose that a basic cut g_B^l is to be deleted from G_B, $1 \le l \le L$. To restore the form of E we have to replace it by an nonbasic cut g_N^j corrsponding to the l-th function. Assume that we can find such g_N^j in G_N. Then we have to delete $\hat{g}^j$ from $\hat{G}$ and update the columns of $\hat{G}$ that correspond to the l-th block, so that (4.4)-(4.5) will hold for with G_B^{new} and N^{new}. This is equivalent to subtracting $\hat{g}^j$ from these columns, which yields G_N^{new}.

Therefore, in our reduced factorization (5.4) we only need to subtract the j-th column r^j of R_2 from all other columns that correspond to the same block l. To avoid introducing subdiagonal nonzeros to R_2 we choose j to be the smallest among the candidates.

To update the other data, let us note that (4.7) yields

$$x_B^{\text{new}} = x_B + \sigma p_l (g_B^l - g_N^j) = x_B - \sigma p_l \hat{g}^j. \tag{5.17}$$

This has an influence on the right side of (5.5) and implies

$$w_2^{\text{new}} = w_2 - \sigma p_l R_2^{-T} G_2^T \hat{g}_2^j = w_2 - \sigma p_l Q_2^T \hat{g}_2^j = w_2 - \sigma p_l r^j. \tag{5.18}$$

Subtracting the j-th columns of $\hat{G}_2$ and R_2 from the related ones does not have any influence on the solution of (5.5) because it only combines the equations there.

After performing the above exchange of a basic and a nonbasic cut we carry out the procedure for deleting a nonbasic cut.

If g_B^l is the only gradient in block l, the need to delete it may arise only when we tried to append a new gradient g from block l and linear dependence occured. Then the new gradient simply replaces g_B^l leaving the set of nonbasic gradients unchanged. Similarly to (5.17),

$$x_B^{\text{new}} = x_B + \sigma p_l(g_B^l - g) = x_B + \sigma p_l \hat{g}.$$

Using it in (5.5), analogously to (5.18), we get

$$w_2^{\text{new}} = w_2 - \sigma p_l Q_2^T \hat{g}_2 = w_2 - \sigma p_l r,$$

with r given by (5.7).

Deleting a bound

Suppose that bounds on variables $x_1, x_2, \dots, x_k$ are active and that we have to delete a bound $x_j = x_j^{\max}$, $j \le k$. We have at first to put the bound to the end of $\hat{G}$ and $\hat{a}$ in (5.1) and the corresponding column to the end of Q in (5.2), which yields a matrix

$$\tilde{R}_2 = \begin{bmatrix} R_2 & \\ s^T & 1 \end{bmatrix},$$

where s^T is the j-th row of $\hat{G}_1$ (this is essentially inverting the procedure for adding a bound, cf. (5.13)). Now we can restore the upper triangular form of $\tilde{R}_2$ by a sequence of Givens operations $P = P_{m,m+1} \cdots P_{2,m+1} P_{1,m+1}$ that act on rows i and $m+1$ of $\tilde{R}_2$, $i = 1, \dots, m$ to annulate s^T:

$$P \begin{bmatrix} R_2 & \\ s^T & 1 \end{bmatrix} = \begin{bmatrix} R_2^{\text{new}} & r \\ & \rho \end{bmatrix}.$$

Again, this is an inverse operation to (5.14). Likewise, we update w by an inverse of (5.15),

$$P \begin{bmatrix} w \\ x_{Bj} - x_{Bj}^{\max} \end{bmatrix} = \begin{bmatrix} w^{\text{new}} \\ \nu \end{bmatrix}.$$

6. Data structures

The specific form of our problem and its large size create a need for designing data structures which on the one hand would use the computer memory in an economic way and on the other hand allow for efficient execution of the operations appearing in the sub-algorithms of the method. There are three groups of problems related to this subject:

(a) original problem data,

(b) data related to the procedure for solving the subproblems,

(c) data related to the master algorithm.

Let us discuss these issues in more detail.

Original problem data

It is a matter of routine in linear programming systems that problem data are stored as sparse matrices, usually as files of packed columns. However, storing A, W and T_ω for $\omega \in \Omega$ in this way does not fully exploit characteristic features of stochastic programs, because in practice only some of the entries of the technology matrix are random, if any. Each T_ω can be thus expressed as

$$T_\omega = \bar{T} + \Delta T_\omega \tag{6.1}$$

where $\bar{T}$ is a certain deterministic matrix and ΔT_ω has only few nonzero entries. Furthermore, the sparsity pattern of ΔT_ω is the same for all $\omega \in \Omega$. Similarly,

$$d_\omega = \bar{d} + \Delta d_\omega, \ \omega \in \Omega. \tag{6.2}$$

This suggests the use of a structure in which we store the following data:

(a) the deterministic entries of A, $\bar{T}$, b, $\bar{d}$ and W together with their positions;

(b) the positions of random entries in T_ω and d_ω;

(c) the realizations of the entries of ΔT_ω , Δd_ω for $\omega \in \Omega$.

This structure allows for easy recovery of any data and its storage requirements exceed the deterministic formulation only by LN_r , where N_r is the number of random entries and L is the number of realizations. It is also clear that A and $\bar{T}$ may be input as one constraint matrix, and their separation can be performed on the basis of information about positions of random entries and nonzero entries of W.

Subproblem data

Subproblems (2.4) differ only in the right hand sides

$$\tilde{d}_\omega = d - T_\omega x$$

which on the basis of (6.1) and (6.2) can be fast calculated by

$$\tilde{d}_\omega = \tilde{d}_\omega + \Delta\tilde{d}_\omega$$

with

$$\tilde{d} = \bar{d} - \bar{T}x,$$
$$\Delta\tilde{d}_\omega = \Delta d_\omega - (\Delta T_\omega)x.$$

As mentioned in section 2, a reasonable method for solving (2.4) is the dual simplex method: we can successively process all the subproblems by starting each from the optimal basis obtained for the previous one. To speed up the reoptimization procedure we can reorder $\tilde{d}_\omega$, $\omega \in \Omega$, in such a way that subsequent realizations would be close one to another, as in bunching procedures (see [7, 25]). One of possibilities, which we used in our code, is to base bunching on the sign pattern of $\Delta\tilde{d}_\omega$. More advanced techniques that store a search tree of possible bases, such as trickling down (see, e.g., [7]), are very hard to implement for large problems, especially when Bartels-Golub factorization is used to represent the basis inverse.

Upper level data

The most difficult problem concerning data storage is the way for storing and using objective and feasibility cuts (2.8) and (2.10). Since g_ω and $\bar{g}_\omega$ are computed vectors and are usually dense, the simplest approach is to store them in an $n \times M$ matrix, where M is the number of computed cuts in (3.2), $L \leq M \leq n + 2L$. Together with this, we can store three integer arrays indicating the numbers of blocks and pointing basic and nonbasic cuts. The reduced cuts (4.4) may overwrite G_N.

Still, for a very large number of realizations, or when the number of rows of T is much smaller than the dimension of x, this data structure may suffer from excessive storage requirements. On the other hand, from (2.8) and (2.10) we see that each computed cut is defined by the corresponding vector of simplex multipliers π and the original problem data. Hence, it is sufficient to store a file of multipliers Π defining the cuts and to recover from them the subgradients necessary for each transformation. Next, noting that the same basis in (2.4) may be optimal for many right hand sides, we see that many cuts may have the same multiplier vector π, so the number of columns of Π may be significantly smaller than the number of computed cuts. This is similar to the bunching procedure (see [25, 6, 7]) but we group here the cuts, not the subproblems. While there is no doubt as to storage savings due to the implicit data structure, we have to carefully examine the computational overhead implied by the necessity of recovering cuts from multipliers. The most time-consuming operations here are multiplications by $\hat{G}$ and $\hat{G}^T$, but the specific structure (6.1) used for storing T_ω may allow for mitigating this overhead considerably.

7. Numerical results

A specialized code has been developed on the basis of the techniques described in this paper. For the purpose of solving lower level subproblems (2.4) we used subroutines from the XMP package of [10] and the LA05 routines from Harwell library for basis inverse management (see [14]). All computations were carried out on a Sun Sparcstation 1. The time reported is always in seconds.

The first series of experiments were carried out on a network planning problem with a random demand (see [20]). The original problem has 82 independent random variables, each taking 5-10 possible values, which results in an astronomical number of all possible scenarios. Therefore, we sampled from this distribution a smaller number of scenarios. The sizes of the original problem and the resulting LP approximations are given in tables 1 and 2.

Four versions of the method were run on these problems. The first one was our basic RD method described in this paper, with the penalty parameter σ controlled on-line as follows ($0.5 < \gamma < 1.0$):

- if $F(x^k) > \gamma F(\xi^k) + (1-\gamma)\hat{F}^k$ ("null step") then decrease σ;
- if $F(x^k) < (1-\gamma)F(\xi^k) + \gamma\hat{F}^k$ ("exact step") then increase σ;
- otherwise ("approximate step") keep σ unchanged.

The second version had a very large parameter σ in (2.6), which practically disabled the effect of the regularizing term (we used $\sigma = 10^6$). In this way we aimed at assessing the effect of regularization on performance of the method. We still kept, however, separate cuts for each subproblem, as in the *multicut method* of [1].

The third version was the regularized method again, but with the cuts generated for the expected second stage cost

$$f(x) = \sum_{\omega\in\Omega} f_\omega(x)$$

instead for each f_ω separately. We used the same code again, but the objective cuts from subproblems were averaged before being passed to the master:

$$g^j = \sum_{\omega\in\Omega} g^j_\omega,$$

$$\alpha^j = \sum_{\omega\in\Omega} \alpha^j_\omega.$$

As a result, we obtained the bundle-type method of [9].

The fourth version also accumulated aggregate cuts, as the previous one, but had the regularizing term disabled by $\sigma = 10^6$. So, the method was practically equivalent to the L-shaped method of [23].

The results are summarized in tables 3 and 4.

Our RD method clearly outperforms all the other versions for larger problems. Comparing the performance of the regularized and the linear version (table 3) we see that the use of regularization substantially decreases the number of iterations of the RD method and, consequently, the number of dual simplex steps in subproblems. However, regularization alone, without the decomposition of cuts, does not help much, as the results for the bundle-type method in table 4 show. It turns out that it is much better to work with our large master having separate cuts for each f_ω than with a bundle-type master of [9]. Both methods with aggregate cuts - the bundle method and the L-shaped method - were very slow near the solution: they had increasing difficulties in identifying attractive sets of cuts and good dirctions. Presumably, directional minimization for the bundle method might help a little, but in our case it would be prohibitively expensive. It is much better to have a more accurate approximation of the recourse function, as in the RD method, even if it requires solving a "large" master.

Apart from a much slower convergence, the methods with averaged cuts (table 4) encountered numerical difficulties in the neigborhood of the solution: the cuts did not support the graph of the recourse function with the required accuracy. So, also from the numerical point of view it is better to keep cuts for subproblems separately and to use probabilities explicitly in the master (see (4.7)), than to have a smaller number of averaged cuts.

Since the size of the master could be the only possible argument against using separate cuts for the subproblems, we can safely discard methods with averaged cuts as clearly inferior (for two-stage stochastic programs).

The next example is a stochastic scheduling and transportation problem described in detail in [11]. Again, in tables 5 and 6 we present problem statistics for various numbers of scenarios. In table 7 we summarize the performance of the RD method and of its linear counterpart (but still with decomposed cuts) and in tables 8 and 9 we give details of the runs for the two largest problems. Again, for these large problems our RD method is a clear winner, mainly because of a smaller number of null steps - long trial steps to points which are much worse than the current one. The method without regularization was not able to solve the largest 1000-scenario problem in an acceptable time; the number of fruitless null steps was too large.

There seems to be no doubt as to efficiency of our approach and to usefulness of regularization. We were able to solve successfully problems that have deterministic equivalents of enormous sizes. This is mainly due to the use of regularization and cut decomposition in the master and to our specialized technique for solving the master problem, which boils it down to a size that is practically independent on the number of scenarios.

Straightforward LP techniques fail on these problems already for moderate numbers of scenarios. As an illustration, in table 10 we present available computation times (on an identical computer) of a simplex code MINOS 5.3 of [13], an interior point code LOQO of [22] and a Benders decomposition code MSLIP of [6] for our second example. The methods are all much slower than RD, and they could not, so far, solve problems with larger numbers of scenarios. The method that was able to solve problems of comparable sizes was the Diagonal Quadratic Approximation method (DQA) of [11, 12]; it was also slower than RD (although it was run on a network of workstations rather than just one), but it has a potential of solving a broader class of problems (see [19]).

8. Conclusions

The regularized decomposition method appears to be a rather efficient tool for solving large scale two-stage stochastic programming problems. Its efficiency is due to the following features.

1. The quadratic regularizing term stabilizes the master problem and helps to avoid aimless steps.

2. The use of separate approximations for scenario subproblems instead of aggregate (averaged) cuts speeds up convergence owing to a better description of the recourse function.

3. The special algorithm for solving the master problem based on dynamic selection of critical scenarios reduces it to a small numerical core whose size does not depend on the number of scenarios.

4. The use of the dual simplex method allows for rapid re-optimization of the subproblems.

There is also a disadvantage associated with our approach: the work with the reduced master requires a number of involved operations, so implementation of the method is not easy. But this is one of the keys to the good performance, because we use many closed-form formulae.

The method allows for a number of generalizations and extensions: its dual form turns out to be a decomposition method for augmented Lagrangians (see [16, 17]); it can also be generalized to an asynchronous multistage version (see [18]). We hope to be able to present for these versions results for equally difficult examples as the ones discussed here.

References

[1] Birge J.R. and Louveaux F.V.,"A multicut algorithm for two-stage stochastic linear programs", *European Journal of Operations Research* 34(1988) 384-392.

[2] Birge J.R. and Wets R.J.-B., "Designing approximation schemes for stochastic approximation problems, in particular for stochastic programs with recourse", in: *Stochastic Programming 1984*, A. Prekopa and R.J.-B. Wets (eds.), *Mathematical Programming Study* 27(1986) 54-102.

[3] Daniel J.W. et al., "Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization", *Mathematics of Computation* 30(1976) 772-795.

[4] Dantzig G. and Madansky A., "On the solution of two-stage linear programs under uncertainty", in *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, University of California Press, Berkeley 1961, pp. 165-176.

[5] Dantzig G. and Wolfe P., "Decomposition principle for linear programs", *Operations Research* 8(1960) 101-111.

[6] Gassmann H.I.,"MSLiP: A computer code for the multistage stochastic linear programming problem", *Mathematical Programming* 47 (1990) 407-423.

[7] Haugland D. and Wallace S.W., "Solving many linear programs that differ only in the righthand side", *European Journal of Operations Research* 37(1988) 318-324.

[8] Kall P., Ruszczyński A. and Frauendorfer K., "Approximation techniques in stochastic programming", in: *Numerical Techniques for Stochastic Optimization* (Y. Ermoliev and R. Wets, eds), Springer-Verlag, Berlin 1988, pp. 33-64.

[9] Kiwiel K.C., *Methods of Descent for Nondifferentiable Optimization*, Springer-Verlag, 1985.

[10] Marsten R., "The design of the XMP linear programming library", *ACM Transactions of Mathematical Software* 7(1981) 481-497.

[11] Mulvey J.M. and Ruszczyński A., "A new scenario decomposition method for large-scale stochastic optimization", technical report SOR-91-19, Deapartment of Civil Engineering and Operations Research, Princeton University, Princeton 1991 (to appear in *Operations Research*).

[12] Mulvey J.M. and Ruszczyński A., "A diagonal quadratic approximation method for large-scale linear programs", *Operations Research Letters* 12(1992) 205-215.

[13] Murtagh B.A. and Saunders M.A., "MINOS user's guide", Technical report SOL 83-20R, Systems Optimization Laboratory, Stanford University, 1983 (revised 1987).

[14] Reid J., "A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases", *Mathematical Programming* 24(1982) 55-69.

[15] Ruszczyński A., "A regularized decomposition method for minimizing a sum of polyhedral functions", *Mathematical Programming* 35(1986) 309-333.

[16] Ruszczyński A., "An augmented Lagrangian decomposition method for block diagonal linear programming problems", *Operations Research Letters* 8(1989) 287-294.

[17] Ruszczyński A., "Regularized decomposition and augmented Lagrangian decomposition for angular linear programming problems", in: Aspiration Based Decision Support Systems (A. Lewandowski and A. Wierzbicki, eds.), Lecture Notes in Economics and Mathematical Systems, Springer Verlag, Berlin 1989, pp. 80-91.

[18] Ruszczyński A., "Parallel decomposition of multistage stochastic programming problems", *Mathematical Programming* (forthcoming).

[19] Ruszczyński A., "Augmented Lagrangian decomposition for sparse convex optimization", WP-92-75, IIASA, Laxenburg, 1992.

[20] Sen S., Doverspike R.D. and Cosares S., "Network planning with random demand", technival report, Department of Systems and Industrial Engineering, University of Arizona, Tucson, 1992.

[21] Topkis J.M., "A cutting plane algorithm with linear and geometric rates of convergence", *Journal of Optimization Theory and Applications* 36(1982) 1-22.

[22] Vanderbei R. J., "LOQO user's manual", technical report SOR-91-10, Deapartment of Civil Engineering and Operations Research, Princeton University, Princeton 1991.

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

[24] Wets R.J.-B., "Stochastic programs with fixed recourse: the equivalent deterministic program", *SIAM Review* 16(1974) 309-339.

[25] Wets R.J.-B., "Large scale linear programming techniques", in: *Numerical Techniques for Stochastic Optimization* (Y. Ermoliev and R. Wets, eds), Springer-Verlag, Berlin 1988, pp. 65-94.

	Rows	Columns
1st stage	1	89
2nd stage	175	706

Table 1: Dimensions of Example 1.

Scenarios	Rows	Columns
1	176	795
10	1751	7149
20	3501	14209
50	8751	35389
100	17501	70689
200	35001	141289

Table 2: Dimension of equivalent LP formulations for Example 1.

Scen.	Regularized Decomposition			Linear Version		
	Iterations		CPU	Iterations		CPU
	Main	Subs	Time	Main	Subs	Time
10	14	7241	128	14	7241	125
20	19	20523	361	24	25606	455
50	41	115711	2137	56	157782	2898
100	40	231074	4191	67	389453	6928
200	39	453346	7803	87	1011807	17442

Table 3: Performance of the methods with decomposed cuts for Example 1.

Scen.	Bundle Method			L-shaped Method		
	Iterations		CPU	Iterations		CPU
	Main	Subs	Time	Main	Subs	Time
10	18	9509	168	16	8356	151
20	37	39361	692	36	38620	700
50	217*	636050	11461	127*	367957	6888
100	244*	1452821	26026	117	694542	12920
200	149*	1757725	31476	140*	1653503	30204

* - final accuracy 10^{-6} not obtained

Table 4: Performance of the methods with aggregate cuts for Example 1.

	Rows	Columns
1st stage	467	121
2nd stage	118	1259

Table 5: Dimensions of Example 2.

Scenarios	Rows	Columns
1	585	1380
10	1647	12711
50	6376	63071
100	12267	126021
200	24067	251921
1000	118467	1259121

Table 6: Dimension of equivalent LP formulations for Example 2.

Scen.	Regularized Decomposition			Linear Version		
	Iterations		CPU	Iterations		CPU
	Main	Subs	Time	Main	Subs	Time
10	23	4628	132	36	5674	161
50	31	22282	620	42	28526	789
100	25	35126	987	63	82953	2483
200	23	61614	1729	120	344686	9502
1000	50	631380	17795	200*	2266247	62467

* - final accuracy 10^{-6} not obtained

Table 7: Performance of the method for Example 2.

	Regularized Decomposition	Linear Version
Main algorithm		
Iteration	23	120
Exact steps	15	1
Approximate steps	3	19
Null steps	5	100
Master		
Adding a cut	1007	2330
Adding a bound	99	414
Deleting a cut	950	1196
Deleting a bound	6	165
Dependence	64	1351
Relaxation	892	10

Table 8: Operations in the 200-scenario version of Example 2.

	Regularized Decomposition	Linear Version
Main algorithm		
Iteration	50	200
Exact steps	17	0
Approximate steps	15	8
Null steps	18	192
Master		
Adding a cut	4764	9106
Adding a bound	99	1280
Deleting a cut	4675	4828
Deleting a bound	6	599
Dependence	308	5396
Relaxation	4373	31

Table 9: Operations in the 1000-scenario version of Example 2.

Scenarios	MSLIP	MINOS	LOQO	DQA*	RD
1	701	95	9	11	38
5	3502	1606	221	225	75
10	6401	5344	1584	1625	132
15	-	-	5025	9076	186
50	-	-	-	4251	620
100	-	-	-	15562	987
200	-	-	-	25202	1729
1000	-	-	-	-	17795

*- elapsed time on a network of workstations.

Table 10: Execution times of different methods for Example 2.