Working Paper

Preconditioned Conjugate Gradients in an Interior Point Method for Two-stage Stochastic Programming

Jacek Gondzio

WP-94-130
December 1994

IIASA

International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria

Telephone: +43 2236 807 □ Fax: +43 2236 71313 □ E-Mail: info@iiasa.ac.at

Preconditioned Conjugate Gradients in an Interior Point Method for Two-stage Stochastic Programming

Jacek Gondzio

WP-94-130
December 1994

International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 807 □ Fax: +43 2236 71313 □ E-Mail: info@iiasa.ac.at

Abstract

We develop a variant of an interior point method for solving two-stage stochastic linear programming problems. The problems are solved in a deterministic equivalent form in which the first stage variables appear as dense columns. To avoid their degrading influence on the adjacency structure AA^T (and the Cholesky factor) an iterative method is applied to compute orthogonal projections. Conjugate gradient algorithm with a structure-exploiting preconditioner is used.

The method has been applied to solve real–life stochastic optimization problems. Preliminary computational results show the feasibility of the approach for problems with up to 80 independent scenarios (a deterministic equivalent linear program has 14001 constraints and 63690 variables).

Key words: interior point method, two-stage stochastic programs, conjugate gradient algorithm.

Preconditioned Conjugate Gradients in an Interior Point Method for Two-stage Stochastic Programming

Jacek Gondzio[1]

1 Introduction

We are concerned with the solution of two-stage stochastic linear optimization problem

$$\begin{array}{lll} \text{minimize} & c_1^T x_1 + E_\omega(c_2^T(\omega)x_2) & \\ \text{subject to} & T_0 x_1 = b_1 & \\ & T(\omega)x_1 + W(\omega)x_2 = b_2(\omega) & \\ & x_1, x_2 \geq 0, & \end{array} \tag{1}$$

where $x_1 \in \mathcal{R}^{n_1}$ and $x_2 \in \mathcal{R}^{n_2}$ are decision vectors at time 1 and 2, respectively; c_1, T_0 and b_1 are cost function, constraint matrix and right hand side vector for time 1 decisions; $c_2(\omega)$, $T(\omega)$, $W(\omega)$, $b_2(\omega)$ are the cost function, constraint matrices and right hand side vector for time 2 decisions which are unknown at time 1. We assume an underlying probability space (Ω, F, P), $\omega \in \Omega$ and we write $T(\omega)$, $W(\omega)$, $b_2(\omega)$ to indicate that they are random and depend on the outcome ω. Further, we assume that Ω is finite, say with S elements. With each element $s \in \{1, \ldots, S\}$, a *scenario* can be associated, i.e. a realization $c_2(\omega)$, $T(\omega)$, $W(\omega)$, $b_2(\omega)$.

Two-stage stochastic programs represent an important class of real–life problems that arise from many different applications from optimal control through financial and network optimization. Their common feature is that we make some immediate decisions (at time 1), subject to first stage constraints, without the knowledge of future random events. After the random variables are realized, further decisions at time 2 are made, subject to additional constraints and incurring additional costs. The overall objective is to minimize costs at time 1 plus expected costs at time 2.

There exist several solution methods for two-stage stochastic linear problems: specializations of the simplex method [8, 9, 13], different variants of decomposition schemes [2, 3, 24, 30, 31, 32] and other computationally attractive approaches [26, 29, 34].

The progress in interior point methods (IPMs) for linear programming [19, 14] makes them also attractive candidates, worth to be tried. An advantageous feature of IPMs is that they converge very rapidly, in 20 to 50 iterations, almost independently of the problem size. Their major disadvantage is sometimes very large computational cost of a

[1]Supported by the Committe for Scientific Research of Poland, grant No PB 8 S505 015 05 and by the Optimization Under Uncertainty Project of IIASA

single iteration that is in practice dominated by an orthogonal projection of a vector onto a null space of the (scaled) linear constraints operator. All efficient general purpose interior point codes [14] compute these orthogonal projections through a direct approach — (Cholesky or Bunch–Parlet) symmetric factorization [7]. There exist structure-exploiting specializations of these direct approaches for stochastic optimization.

The method of [4, 5] handles the whole block of the first stage columns with the Schur complement mechanism. The approach proved to be efficient for problems in which the number of first stage variables, n_1 is not excessive. Additionally, as shown in [16], it parallelises in a scalable way.

The method of [6] applies symmetric, indefinite, Bunch-Parlett decomposition to the augmented least squares system and delays pivots corresponding to dense, first stage columns to prevent excessive fill–in. It parallelises promissingly and generalizes to multi-stage problems.

Let us also mention quite a different approach [20] that reformulates two-stage problems to another deterministic equivalent form with all dense columns split into shorter pieces. However, the method preserves efficiency only for problems with a small number of realizations for which the number of nonanticipativity constraints (that link replicated first stage variables) is not too large.

In the approach presented in this paper we propose to use an iterative method to compute orthogonal projections, namely, the conjugate gradient algorithm. Clearly, to make the approach competitive we need a good preconditioner that "understands" and exploits well properties of a two-stage problem. Fortunately, such a preconditioner can be found for these problems.

Careful analysis of the special structure of a two-stage stochastic optimization problem (done in Section 3) naturally leads to the choice of a block-diagonal preconditioner, separable with respect to different realizations of the first stage variables. Our preconditioner is described in detail in Section 3.2. To make the paper self-contained, in Section 2 we briefly recall a primal–dual predictor–corrector interior point method and conjugate gradient algorithm for solving linear system of normal equations. The method presented in this paper has been implemented and applied to solve Sen's telecommunication network design problem [33]. Section 4 addresses our preliminary computational experience. Finally, in Section 5 we give our conclusions.

2 Primal–dual method and conjugate gradient algorithm

In this section we recall fundamentals of the (infeasible) predictor–corrector primal–dual method and the conjugate gradient algorithm applied to solve normal equations that arise in orthogonal projections.

2.1 Primal–dual algorithm

The first theoretical results for the primal–dual algorithm come from [21, 18]. Descriptions of its efficient implementations can be found in [19, 22] and in the survey [14].

Let us consider a primal linear programming problem

$$\begin{array}{ll} \text{minimize} & c^T x, \\ \text{subject to} & Ax = b, \\ & x + s = u, \\ & x, s \geq 0, \end{array} \tag{2}$$

where $c, x, s, u \in \mathcal{R}^n, b \in \mathcal{R}^m, A \in \mathcal{R}^{m \times n}$ and its dual

$$\begin{array}{ll} \text{maximize} & b^T y - u^T w, \\ \text{subject to} & A^T y + z - w = c, \\ & z, w \geq 0, \end{array} \tag{3}$$

where $y \in \mathcal{R}^m$ and $z, w \in \mathcal{R}^n$.

Next, let us replace the nonnegativity of constraints in the primal formulation with logarithmic barrier penalty terms in the objective function. It gives the following logarithmic barrier function

$$L(x, s, \mu) = c^T x - \mu \sum_{j=1}^{n} \ln x_j - \mu \sum_{j=1}^{n} \ln s_j. \tag{4}$$

The first order optimality conditions for (4) are

$$\begin{array}{rcl} Ax & = & b, \\ x + s & = & u, \\ A^T y + z - w & = & c, \\ XZe & = & \mu e, \\ SWe & = & \mu e, \end{array} \tag{5}$$

where X, S, Z and W are diagonal matrices with the elements x_j, s_j, z_j and w_j, respectively, e is an n-vector of all ones, $z = \mu X^{-1} e$, and μ is a barrier parameter.

A single iteration of the basic primal–dual algorithm makes one step of Newton's method applied to the first order optimality conditions (5) with a given μ and then μ is updated (usually decreased). The algorithm terminates when infeasibility and complementarity gap are reduced below a predetermined tolerance.

Having an $x, s, z, w \in \mathcal{R}^n_+, y \in \mathcal{R}^m$, Newton's direction is obtained by solving the following system of linear equations

$$\begin{bmatrix} A & 0 & 0 & 0 & 0 \\ I & 0 & I & 0 & 0 \\ 0 & A^T & 0 & I & -I \\ Z & 0 & 0 & X & 0 \\ 0 & 0 & W & 0 & S \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \\ \Delta z \\ \Delta w \end{bmatrix} = \begin{bmatrix} \xi_b \\ \xi_u \\ \xi_c \\ \mu e - XZe \\ \mu e - SWe \end{bmatrix}, \tag{6}$$

where

$$\begin{array}{rcl} & \xi_b & = b - Ax, \\ & \xi_u & = u - x - s, \\ \text{and} & \xi_c & = c - A^T y - z + w, \end{array}$$

denote the violations of the primal and dual constraints, respectively.

The set of linear equations (6) reduces to the *augmented system*

$$\begin{bmatrix} -D^{-2} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r \\ h \end{bmatrix}, \tag{7}$$

or further to the *normal equations system*

$$(AD^2A^T)\Delta y = AD^2r + h, \tag{8}$$

where

$$\begin{aligned} D^2 &= (X^{-1}Z + S^{-1}W)^{-1}, \\ r &= \xi_c - X^{-1}(\mu e - XZe) + S^{-1}(\mu e - SWe) - S^{-1}W\xi_u, \\ h &= \xi_b. \end{aligned} \tag{9}$$

Computing $(\Delta x, \Delta y)$ from (7) or Δy from (8) is usually (when a direct approach [7] is applied) divided into two phases: *factorization* of the matrix to some easily invertible form followed by *solution* that exploits this factorization.

Once direction $(\Delta x, \Delta y, \Delta s, \Delta z, \Delta w)$ has been computed, the maximum stepsizes α_P and α_D that maintain nonnegativity of variables in the primal and dual spaces are found. Next, a new iterate is computed using a step reduction factor $\alpha_0 = 0.99995$

$$\begin{aligned} x^{k+1} &= x^k + \alpha_0\alpha_P\Delta x, \\ s^{k+1} &= s^k + \alpha_0\alpha_P\Delta s, \\ y^{k+1} &= y^k + \alpha_0\alpha_D\Delta y, \\ z^{k+1} &= z^k + \alpha_0\alpha_D\Delta z, \\ w^{k+1} &= w^k + \alpha_0\alpha_D\Delta w. \end{aligned} \tag{10}$$

After making the step, the barrier parameter μ is updated and the process is repeated.

2.2 Predictor–corrector technique

Factorization of the matrix (7) or (8) is usually at least an order of magnitude more expensive than the *solution*. The factorizations of systems (6) often take 60% to 90% of the total CPU time needed to solve a problem. It is thus natural to look for a possibility of reducing their number to the necessary minimum, even at the expense of some increase of a cost of a single iteration.

The predictor–corrector technique proposed by Mehrotra [22] decomposes a direction vector $(\Delta x, \Delta s, \Delta y, \Delta z, \Delta w)$ (denoted with Δ) into two parts

$$\Delta = \Delta_a + \Delta_c, \tag{11}$$

where Δ_a and Δ_c denote affine–scaling and centering components, respectively. The term Δ_a is obtained by solving (6) with $\mu = 0$ and Δ_c is the solution of equation (6) with the right hand side vector

$$(0, 0, 0, \mu e - XZe, \mu e - SWe)^T,$$

where $\mu > 0$ is some centering parameter. The term Δ_a is responsible for "optimization" while Δ_c keeps the current iterate away from the boundary.

Let us observe that the affine scaling (predictor) direction Δ_a solves the linear system (6) for the right hand side equal to the current violation of the first order optimality conditions for (2)–(3), i.e. with $\mu = 0$. This direction is usually "too optimistic" — if a full step of length one could be made in it, the LP problem would be solved in one step. Predictor–corrector makes a hypothetical step in this direction. The maximum stepsizes in the primal, α_{Pa} and in the dual, α_{Da} spaces preserving nonnegativity of (x, s) and (z, w), respectively are determined and the predicted complementarity gap

$$g_a = (x + \alpha_{Pa}\Delta x)^T(z + \alpha_{Da}\Delta z) + (s + \alpha_{Pa}\Delta s)^T(w + \alpha_{Da}\Delta w)$$

is computed. It is then used to determine the barrier parameter

$$\mu = \left(\frac{g_a}{g}\right)^2 \frac{g_a}{n}, \tag{12}$$

where $g = x^T z + s^T w$ denotes current complementarity gap.

For such value of μ, the corrector direction Δ_c is computed

$$\begin{bmatrix} A & 0 & 0 & 0 & 0 \\ I & 0 & I & 0 & 0 \\ 0 & A^T & 0 & I & -I \\ Z & 0 & 0 & X & 0 \\ 0 & 0 & W & 0 & S \end{bmatrix} \begin{bmatrix} \Delta_c x \\ \Delta_c y \\ \Delta_c s \\ \Delta_c z \\ \Delta_c w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mu e - \Delta X_a \Delta Z_a e \\ \mu e - \Delta S_a \Delta W_a e \end{bmatrix}, \tag{13}$$

and, finally, the direction Δ of (11) is determined.

2.3 Conjugate gradient algorithm

As mentioned in the previous section, linear system solutions needed in order to compute the predictor (6) and corrector (13) terms of a Newton direction Δ costitute the bulk of work in the primal–dual method. Both these equation systems reduce (see, e.g., [14]) to the symmetric but indefinite augmented system of linear equations (7) and further to the symmetric and positive definite normal equations system (8).

Almost all general purpose primal–dual implementations apply direct approach [7] (symmetric factorization) to one of the abovementioned linear systems. Naturally, the same factorization is used twice when solving systems to obtain predictor and corrector terms.

Iterative methods for solving (8) could never prove really competitive [17, 23] in the context of solution of general large-scale linear programs. The reason for that came always from the difficulty of finding an inexpensive preconditioner. However, in some cases, as e.g., when the LP constraint matrix is specially structured (or when the Cholesky factorization is prohibitively expensive), a good structural preconditioner can be found. This statement applies, in particular, to large scale multicommodity network optimization [28, 27].

It is more natural to apply the conjugate gradient algoritm to the positive definite normal equations system (8) than to the augmented system (although the latter is also possible [10]). Let us consider the following symmetric, positive definite system of linear equations

$$Hp = q, \tag{14}$$

where $H = AD^2A^T$, $A \in \mathcal{R}^{m*n}$, $p, q \in \mathcal{R}^m$. (We assume that the underlying stochastic program has linearly independent constraints, i.e. $rank(A) = m$.)

It is natural to interpret the conjugate gradient algorithm as a method to solve the unconstrained (positive definite) QP problem

$$\text{minimize} \quad (\tfrac{1}{2}p^THp - q^Tp), \tag{15}$$

Equation (14) is the first order necessary optimality condition for (15) and due to positive definitness of H, the sufficient condition as well.

In each step of the conjugate gradient algorithm [11] we choose the search direction d_k to satisfy

$$d_k^THd_i = 0, \qquad i = 1, 2, ..., k-1, \tag{16}$$

i.e. we force d_k to be H-conjugate to $d_1, d_2, ..., d_{k-1}$. Next, we minimize QP objective (15) along d_k, which leads to the choice of stepsize α_k. A formal presentation of the conjugate gradient algorithm is given in the figure below.

Conjugate gradient algorithm

Input

$H = AD^2A^T$: normal equations matrix,

q: right hand side of (14);

Parameters

ϵ: required accuracy of solution,

k_{max}: the maximum number of c.g. iterations;

Initialize

$p_0 = 0$

$r_0 = q$

Algorithm

for $k = 1, ..., k_{max}$

if $\|r_{k-1}\| \leq \epsilon$ **then**

$p = p_{k-1}$

quit

else

$\beta_k = r_{k-1}^Tr_{k-1}/r_{k-2}^Tr_{k-2}$ $\qquad (\beta_1 = 0)$

$d_k = r_{k-1} + \beta_k d_{k-1}$ $\qquad (d_1 = r_0)$

$\alpha_k = r_{k-1}^Tr_{k-1}/d_k^THd_k$

$p_k = p_{k-1} + \alpha_k d_k$

$r_k = r_{k-1} - \alpha_k Hd_k$

endif

Let us observe that since H is positive definite, there exist at most m H-conjugate (hence linearly independent) directions d_k. Consequently, conjugate gradient algorithm

converges in at most m iterations (if exact arithmetic were used). More precisely, the conjugate gradient algorithm converges in m_0 iterations [11] where m_0 is the number of different eigenvalues of H.

Unfortunately, matrices AD^2A^T in interior point methods show a tendency to be extremely ill conditioned. Hence, pure conjugate gradient algorithm applied to (14) shows hopelessly slow convergence. Additionally, the use of finite precion results in round–off errors which sometimes make it impossible for the algorithm to converge.

To remedy this we use a preconditioner, i.e. a matrix C such that

$$\tilde{H} = C^{-1}HC^{-T} \tag{17}$$

is well conditioned. In such case equation (14) is replaced with

$$\tilde{H}(C^T p) = C^{-1}q, \tag{18}$$

The following conditions are usually imposed on the preconditioner C:

1. effort to compute C is significantly smaller than that of computing direct factorization of H,
2. C is easily invertible,
3. $\tilde{H}$ has small number of different eigenvalues.

The first requirement is a necessary condition to make preconditioned conjugate gradient algorithm (PCG) competitive with the direct approach. The second one aims at reducing the cost of its single iteration. Satisfying the third one ensures fast convergence.

The reader will easily find out that a preconditioned conjugate gradient algorithm presented in the figure below is nothing else than a conjugate gradient algorithm applied to (18).

Preconditioned conjugate gradient algorithm

Input

$\tilde{H} = C^{-1}AD^2A^TC^{-T}$: (preconditioned) normal equations matrix,
$\tilde{q}$: right hand side of (18);

Parameters

ϵ: required accuracy of solution,
k_{max}: the maximum number of c.g. iterations;

Initialize

$\tilde{p}_0 = 0$
$\tilde{r}_0 = \tilde{q}$

Algorithm

for $k = 1, ..., k_{max}$
  if $\|\tilde{r}_{k-1}\| \leq \epsilon$ **then**
    $\tilde{p} = \tilde{p}_{k-1}$
    quit
  else
    $\beta_k = \tilde{r}_{k-1}^T\tilde{r}_{k-1}/\tilde{r}_{k-2}^T\tilde{r}_{k-2}$ $\quad (\beta_1 = 0)$
    $\tilde{d}_k = \tilde{r}_{k-1} + \beta_k\tilde{d}_{k-1}$ $\quad (\tilde{d}_1 = \tilde{r}_0)$
    $\alpha_k = \tilde{r}_{k-1}^T\tilde{r}_{k-1}/\tilde{d}_k^T\tilde{H}\tilde{d}_k$
    $\tilde{p}_k = \tilde{p}_{k-1} + \alpha_k\tilde{d}_k$
    $\tilde{r}_k = \tilde{r}_{k-1} - \alpha_k\tilde{H}\tilde{d}_k$
  endif

Note that an implemented algorithm exploits the decomposition

$$\tilde{H} = \tilde{A}D^2\tilde{A}^T = C^{-1}AD^2A^TC^{-T}, \tag{19}$$

in multiplications $\tilde{H}\tilde{d}_k$. For example, the denominator in the definition of α_k is computed as folows

$$\tilde{d}_k^T\tilde{H}\tilde{d}_k = \tilde{u}_k^T\tilde{u}_k, \tag{20}$$

where

$$\tilde{u}_k = DA^TC^{-T}\tilde{d}_k \in \mathcal{R}^n. \tag{21}$$

In other words, in order to avoid an explicit formulation of the adjacency structure, whenever possible we take advantage of the decomposition of $\tilde{H}$.

3 Preconditioner for two–stage problem

We are now ready to take a closer look at the structural properties of a deterministic equivalent formulation of a two–stage problem and to derive a specialized preconditioner.

3.1 Special structure of the two–stage problem

In a deterministic equivalent formulation of (1) we associate matrices T_i and W_i with the scenarios $i = 1, 2, ..., S$ and write the following second stage LP constraints

$$T_i x_1 + W_i x_{2i} = b_{2i}. \tag{22}$$

This leads to the deterministic equivalent formulation of the problem

$$\begin{array}{lllll} \text{minimize} & c_1^T x_1 & + \sum_{i=1}^{S} (c_{2i}^T x_{2i}) & & \\ \text{subject to} & T_0 x_1 & & = b_1 & \\ & T_i x_1 & + W_i x_{2i} & = b_{2i}, & i = 1, 2, ..., S, \\ & x_1 \geq 0, & x_{2i} \geq 0, & & i = 1, 2, ..., S. \end{array} \tag{23}$$

Now (23) is a large–scale linear program with a *block–angular* constraint matrix

$$A = \begin{pmatrix} T_0 & & & & \\ T_1 & W_1 & & & \\ T_2 & & W_2 & & \\ & & & \ddots & \\ T_S & & & & W_S \end{pmatrix}. \tag{24}$$

Let us observe that without the first block of columns matrix A becomes block diagonal. Unfortunately, the first stage columns cause dramatic fill in the adjacency structure

$$AA^T = \begin{pmatrix} T_0 T_0^T & T_0 T_1^T & \cdots & T_0 T_S^T \\ T_1 T_0^T & T_1 T_1^T + W_1 W_1^T & \cdots & T_1 T_S^T \\ & & \ddots & \\ T_S T_0^T & \cdots & & T_S T_S^T + W_S W_S^T \end{pmatrix}. \tag{25}$$

This has an immediate negative influence on the Cholesky factor of AA^T that becomes almost completely dense regardless of the sparsity of A.

To avoid this degrading influence [4, 5] suggest special treatment of the whole block of the first stage columns T: it can be seen as a symmetric rank-n_1 (n_1 is the dimension of x_1) corrector to a diagonal (hence very sparse) adjacency structure of the second stage columns. In this approach the first stage columns are taken into account through the Schur complement mechanism. An advantage of it is a possibility of an almost scalable parallelization due to complete separability of computations over independent scenarios. A clear disadvantage is a need to make an assumption that all W_i matrices have full row rank. This assumption is seldom met in practice, which manifests itself in a serious instability of the Schur complement approach. Another disadvantage is a fast growth of computational effort with the number of the first stage columns, n_1.

We hope to be able to overcome most of these difficulties with the use of an iterative method to solve equations with AD^2A^T. (Moreover, we will never formulate this matrix explicitly.)

3.2 Block–diagonal preconditioner

For ease of presentation in this section we shall omit primal–dual scaling matrix D^2 and we shall refer to the adjacency matrix of the form AA^T (instead of AD^2A^T).

Let us partition matrix A into submatrices T and W corresponding to the first stage variables and the second stage variables, respectively

$$A = (T|W), \tag{26}$$

where

$$T = \begin{pmatrix} T_0 \\ T_1 \\ T_2 \\ \cdots \\ T_S \end{pmatrix} \quad \text{and} \quad W = \begin{pmatrix} W_1 & & & \\ & W_2 & & \\ & & \ddots & \\ & & & W_S \end{pmatrix}. \tag{27}$$

With this notation, the adjacency matrix becomes

$$AA^T = WW^T + TT^T. \tag{28}$$

It seems quite natural to require the preconditioner to share block diagonal structure of WW^T

$$C = \begin{pmatrix} C_0 & & & & \\ & C_1 & & & \\ & & C_2 & & \\ & & & \ddots & \\ & & & & C_S \end{pmatrix}. \tag{29}$$

Let us observe that such a preconditioner offers a reasonable compromise for the requirements 1–3 of Section 2.3. It obviously satisfies conditions 1 and 2. Unfortunately, in general it may be far from meeting requiremet 3. Let us now examine more carefully the matrix $\tilde{H}$ of (17)

$$\tilde{H} = C^{-1}AA^TC^{-T} = \tilde{A}\tilde{A}^T,$$

where

$$\tilde{A} = \begin{pmatrix} C_0^{-1}T_0 & & & & \\ C_1^{-1}T_1 & C_1^{-1}W_1 & & & \\ C_2^{-1}T_2 & & C_2^{-1}W_2 & & \\ & & & \ddots & \\ C_S^{-1}T_S & & & & C_S^{-1}W_S \end{pmatrix}. \tag{30}$$

Diagonal blocks C_i of the preconditioner are (sparse) Cholesky factors of some matrices closely related to the diagonal elements of AA^T in (25). Namely,

$$C_0C_0^T = T_0T_0^T \qquad \text{and} \qquad C_iC_i^T = \alpha T_iT_i^T + W_iW_i^T, \;\; i = 1, \ldots, S, \tag{31}$$

where α is a real number from interval $[0, 1]$. Note that this particular form of the preconditioner aims at improving the numerical properties of a single realization (scenario) and it "does not see the problem as a whole". The conjugate gradient algorithm remains responsible for merging information available from scenario preconditioners to get the solution of (14).

We suppose it is constructive to analyse the influence of parameter α on the preconditioned matrix $\tilde{H}$.

The choice of $\alpha = 0$ corresponds to only preconditioning the second stage contribution to (28). Simple calculations show that in such case

$$\tilde{H} = I + \tilde{T}\tilde{T}^T, \tag{32}$$

with

$$\tilde{T} = C^{-1} \begin{pmatrix} T_0 \\ T_1 \\ T_2 \\ \cdots \\ T_S \end{pmatrix}. \tag{33}$$

Two additional requirements have to be satisfied for such a preconditioner to be successful. First, T_0 and every W_i have to have full row rank to make C_i of (31) well defined. Secondly, to prevent negative influence of the symmetric rank-n_1 corrector $\tilde{T}\tilde{T}^T$ on the conditioning of $\tilde{H}$ one would like the columns of $\tilde{T}$ be orthogonal to each other. If the latter condition were satisfied, then matrix $\tilde{H}$ would have at most n_1+1 different eigenvalues. (The reader can easily check that each column of $\tilde{T}$ is then an eigenvector of $\tilde{H}$.)

Unfortunately, due to ill conditioning of AD^2A^T matrix, the preconditioner of (31) with $\alpha = 0$ is far from satisfying the second requirement.

The choice of $\alpha = 1$ in the preconditioner (31) corresponds to quite a different space transformation. In contrast to the previous preconditioner, this one does not try to cluster all except n_1 eigenvalues of $\tilde{H}$ around the same value of 1. Instead, it introduces variable scaling that brings all diagonal blocks of $\tilde{H}$ to identity matrices and (hopefully) keeps the off–diagonal blocks small. Although the eigenvalues of $\tilde{H}$ are not clustered, the conjugate gradient algorithm is supposed to converge faster due to better scaling of $\tilde{H}$.

In this case the always present ill conditioning of AD^2A^T matrix manifests in too large a contribution of the neglected off-diagonal blocks $C_i^{-1}T_iT_j^TC_j^{-T}$. This, in turn, slows down the convergence of the conjugate gradient algorithm.

We tested experimentally several preconditioners of type (31). Apart from extreme values $\alpha = 0$ and $\alpha = 1$, we tried the preconditioner for many intermediate values of α. The best practical convergence of the algorithm has been observed for $\alpha \in [0.01, 0.1]$. The results presented in the following section have been obtained with $\alpha = 0.01$.

4 Numerical results

The method described in this paper has been verified computationally. The predictor–corrector primal–dual code HOPDM (Higher Order Primal Dual Method) of [1, 12] has been extended to handle two-stage stochastic programs. The results reported in this section have been obtained when it was run on a SUN SPARC 10 workstation with 32 MBytes of memory. HOPDM is written in FORTRAN 77 and has been compiled with the SUN F77 compiler; option -O has been used.

We report on the performance of the algorithm applied to solve the Sen's [33] telecommunication network design problem. The dimensions of the problem are $m_1 = 1$, $n_1 = 90$, $m_2 = 175$ and $n_2 = 705$. Table 1 summarizes the characteristics of the different variants of the problem.

Table 1. Characteristics of Sen's problems.

Problem	Scenarios	Rows of A	Columns of A
SEN.1	1	176	885
SEN.2	2	351	1680
SEN.4	4	701	3270
SEN.8	8	1401	6450
SEN.10	10	1751	8040
SEN.20	20	3501	15990
SEN.30	30	5251	23940
SEN.40	40	7001	31890
SEN.50	50	8751	39840
SEN.60	60	10501	47790
SEN.70	70	12251	55740
SEN.80	80	14001	63690

Table 2 offers some quantitive data regarding the solution process with the method presented in this paper. It contains the number of iterations to reach 8-digit accurate solution, CPU time in seconds and the average and maximum numbers of conjugate gradient iterations to solve the underlying normal equations. We required the precision $\epsilon = 10^{-8}$ of orthogonal projections and we allowed at most $k_{max} = 6 \times n_1 = 540$ PCG iterations to be done.

The analysis of results collected in Table 2 shows that although the method is not very efficient, it is able to solve even nontrivial problems with a considerable number of scenarios in a still acceptable time.

A clear advantage of it are the very small storage requirements that only linearly grow with the number of scenarios. The largest problem solved SEN.80 needed about 20MBytes of memory to be run. Moreover, a dominating term in these requirements was a set of "long" vectors used to handle the logic of the primal–dual method and work arrays needed to implement the conjugate gradient algorithm. The storage for the preconditioner was only 5% of it since every diagonal block C_i had 1650 nonzero elements, so even for an 80-scenarios problem, the preconditioner needed only $80 \times 1650 = 132,000$ double precision numbers (this is equivalent to about 1 MByte of memory).

Fast growth of the number of PCG iterations with an increase of the number of scenarios is a discouraging result. We hoped to keep this number around $n_1 = 90$ but this was the case only for the number of scenarios that did not exceed 20. Unfortunately, for a larger number of scenarios conjugate gradient algorithm revealed slower convergence showing the limits of the block–diagonal preconditioner presented in this paper. The required accuracy $\epsilon = 10^{-8}$ could not have been obtained even within k_{max} PCG iterations. Due to this lack of precision in the calculation of orthogonal projections, in case of two largest problems SEN.70 and SEN.80 computations were terminated with only four and three exact digits of optimum, respectively.

Finally, we would like to comment on a surprizing fenomenon that we observed. The number of PCG iterations changes always in the same manner in seccessive iterations of the primal–dual method. It is small in the beginning (definitely smaller than n_1). It increases to its maximum in the "middle" of optimization, say between getting the first accurate digit of optimal result and getting about 4–5 digits accurate. In the following, final phase of optimization, this number rapidly drops to less than $n_1/2$. This means that the block–diagonal preconditioner lacks accuracy in the middle of optimization but continuously improves with the progress towards optimum.

Table 2. Solution statistics for Sen's problems.

Problem	Solution		Preconditioned c.g. iterations	
	Iters	CPU time	Average	Maximum
SEN.1	11	18.2	28	41
SEN.2	12	50.4	39	62
SEN.4	15	145.5	50	65
SEN.8	21	491.7	61	79
SEN.10	22	729.1	67	91
SEN.20	30	2877.0	98	193
SEN.30	37	7383.0	223	540
SEN.40	43	13276.6	240	540
SEN.50	51	26960.8	256	540
SEN.60	54	29123.8	264	540
SEN.70	55*	42854.8	293	540
SEN.80	56*	51171.3	309	540

$*$ - only four and three digit optimum attained, respectively.

A natural explanation of such behavior is that the closer to the optimum, the clearer separation into active and inactive constraints and bounds can be observed. A consequence of it is a good separation of diagonal elements of D^2 into those which converge to zero and those which remain large (see, e.g., [14], Section 5). The contribution of small elements of D^2 to AD^2A^T becomes negligible and the effective rank of (first stage) columns that the preconditioner has to take into account decreases very fast.

5 Conclusions

We have presented a new approach to the solution of two–stage stochastic optimization problems that applies preconditioned conjugate gradient algorithm to compute projections in the interior point method. The particular block–angular structure of the two–stage problem has been exploited to define an efficient block–diagonal preconditioner. The method has been shown to be practicable when applied to solve a real–life optimization problem known from the literature.

Although the method does not offer fast convergence, it has several advantages e.g.: storage efficiency and a straightforward generalization to multi–stage stochastic programs. Let us also observe that almost all linear algebra operations in it can be separated into tasks that depend only on a single scenario and interact very little with each other. Thus the method proposed can probably be parallelized in a scalable way (see, e.g., [15]).

Acknowledgements
This paper grew out of the author's discussion with Professor Andrzej Ruszczynski from the International Institute for Applied Systems Analysis, who should share credit for the main idea. The author is grateful to Artur Swietanowski for reading this paper very carefully.

References

[1] Altman A. and J. Gondzio (1993) An efficient implementation of a higher order primal–dual interior point method for large sparse linear programs, *Archives of Control Sciences* 2, No 1/2, pp. 23-40.

[2] Bahn O., O. du Merle, J.-L. Goffin and J.-P. Vial (1994) A cutting plane method from analytic centers for stochastic programming, Technical Report, Dept. of Management Studies, University of Geneva, 1211 Geneva, Switzerland, December 1992, revised January 1994, *Mathematical Programming* (to appear).

[3] Birge J. (1985) Decomposition and partitioning methods for multistage stochastic linear programs, *Operations Research* 33, pp. 989-1007.

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

[5] Birge J. and L. Qi, (1988). Computing block-angular Karmarkar projections with applications to stochastic programming, *Management Science* 34, No 12, pp. 1472-1479.

[6] Czyzyk J., R. Fourer and S. Mehrotra (1994) Parallel solution of multi-stage stochastic linear programs by interior point method, Technical Report, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois 60208–3119, USA, August 1994.

[7] Duff I.S., A.M. Erisman and J.K. Reid (1987) *Direct methods for sparse matrices*, Oxford University Press, New York, 1987.

[8] Ermoliev Yu. and R.J.-B. Wets (eds.) *Numerical Techniques for Stochastic Optimization*, Springer-Verlag, Berlin, 1988.

[9] Fourer R. (1984). Staircase matrices and systems, *SIAM Review* 26, pp. 1-70.

[10] Gill P.E., W. Murray, D. B. Ponceleón and M.A. Saunders (1992) Preconditioners for Indefinite Systems Arising in Optimzation, *SIAM Journal on Matrix Analysis and Applications* 13, No 1, pp. 292-311.

[11] Golub G.H. and C. Van Loan, *Matrix Computations*, (2nd ed.) The Johns Hopkins University Press, Baltimore and London, 1989.

[12] Gondzio, J. (1994) Presolve Analysis of Linear Programs Prior to Applying the Interior Point Method, Technical Report 1994.3, Department of Management Studies, University of Geneva, Switzerland.

[13] Gondzio J. and A. Ruszczynski (1992). A sensitivity method for basis inverse representation in multistage stochastic linear programming problems, *Journal of Optimization Theory and Applications* 74, pp. 221-240.

[14] Gondzio J. and T. Terlaky (1994) A computational view of interior point methods for linear programming, in: *Advances in Linear and Integer Programming*, J. Beasley ed., Oxford University Press, Oxford, England, 1994 (to appear).

[15] Gupta A., V. Kumar and A. Sameh (1994). Performance and scalability of preconditioned conjugate gradient methods on parallel computers, Technical Report 92-64, Department of Computer Science, University of Minnesota, Mineapolis, November 1992, revised in April 1994.

[16] Jessup E.R., D. Yang and S.A. Zenios (1993) Parallel factorization of structured matrices arising in stochastic programming, Technical Report 93-02, Operations and Information Management Department, University of Pennsylvania, Philadelphia, *SIAM Journal on Optimization* (to appear).

[17] Karmarkar N.K. and K.G. Ramakrishnan (1991) Computational results of an interior point algorithm for large scale linear programming, *Mathematical Programming* 52, pp. 555-586.

[18] Kojima, M., S. Mizuno and A. Yoshise (1989) A Primal–Dual Interior Point Algorithm for Linear Programming, in N. Megiddo, ed., *Progress in Mathematical Programming: Interior Point and Related Methods*, Springer–Verlag, New York, pp. 29–48.

[19] Lustig, I. J., R. E. Marsten and D. .F. Shanno (1994) Interior Point Methods for Linear Programming: Computational State of the Art, *ORSA Journal on Computing* 6, pp. 1–14.

[20] Lustig I., J. Mulvey and T. Carpenter (1991) Formulating two-stage stochastic programs for interior point methods, *Operations Research* 39, pp. 757-770.

[21] Megiddo, N. (1989) Pathways to the Optimal Set in Linear Programming, in N. Megiddo, ed., *Progress in Mathematical Programming: Interior Point and Related Methods*, Springer–Verlag, New York, pp. 131–158.

[22] Mehrotra, S. (1992) On the Implementation of a Primal–Dual Interior Point Method, *SIAM Journal on Optimization* 2, pp. 575–601.

[23] Mehrotra, S. (1992) Implementations of affine scaling methods: approximate solutions of systems of linear equations using preconditioned conjugate gradient methods, *ORSA Journal on Computing* 4, No 2, pp. 103-118.

[24] Mulvey J. and A. Ruszczynski (1994) A new scenario decomposition method for large scale stochastic optimization, *Operations Research* (to appear).

[25] Munksgaard N. (1980) Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients, *ACM Transactions on Mathematical Software* 6, No 2, pp. 206–219.

[26] Nielsen S.S. and S.A. Zenios (1993) A massively parallel algorithm for nonlinear stochastic network problems, *Operations Research* 41, No 2, pp. 319–337.

[27] Portugal L., F. Bastos, J. Judice, J. Paixõ and T. Terlaky (1993) An Investigation of Interior Point Algorithms for the Linear Transportation Problems, Technical Report No. 93–100, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Nederlands.

[28] Resende, M.G.C. and Veiga, G. (1992) An Efficient Implementation of a Network Interior Point Method, Technical Report, February 1992, AT&T Bell Laboratories, Murray Hill, NJ, USA.

[29] Rockafellar T. and R. J.-B. Wets, Scenarios and policy aggregation in optimization under uncertainty, *Mathematics of Operations Research* 16, No 1, pp. 119-147.

[30] Ruszczynski A. (1985). A regularized decomposition method for minimizing a sum of polyhedral functions, *Mathematical Programming* 33 pp. 309-333.

[31] Ruszczynski A., (1993) Parallel decomposition of multistage stochastic programs, *Mathematical Programming* 58, pp. 201-228.

[32] Swietanowski A. (1994) Efficient Solution Techniques for Two Stage Stochastic Linear Problems, to appear as a Working Paper of the International Institute for Applied Systems Analysis, Laxenburg, Austria.

[33] Sen S., R.D. Doverspike and S. Cosares (1992) Network planning with random demand, Technical Report, Department of Systems and Industrial Engineering, University of Arizona, Tucson, 1992.

[34] Wallace S. and T. Helgason (1991) Structural properties of progressive hedging algorithm, *Annals of Operations Research* 31, pp. 445–456.