Working Paper

Interior Point Methods in Stochastic Programming

Andrzej Ruszczyński

WP-93-8
February 1993

IIASA International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 715210 □ Telex: 079 137 iiasa a □ Telefax: +43 2236 71313

Interior Point Methods in Stochastic Programming

Andrzej Ruszczyński

WP-93-8
February 1993

International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 715210 □ Telex: 079 137 iiasa a □ Telefax: +43 2236 71313

Contents

Interior Point Methods in Stochastic Programming

Andrzej Ruszczyński

1. Introduction

Theory and computational methods of optimization found numerous applications in science, technology and economy. The fundamental linear programming problem

$$\begin{aligned} &\min\ c^T x \\ &Ax = b, \\ &x \geq 0 \end{aligned} \tag{1.1}$$

can be encountered in seemingly remote application areas. The demand for methods capable solving such problems with tens of thousands of variables and constraints stimulates intensive research in the area of linear programming. One of promising approaches to linear programs which may solve problems that are intractable otherwise are so-called *interior point methods*. Their rapid development started from the works of Karmarkar [13], but their main ideas can be traced back to the discovery of Dikin [7] and to the general approach developed by Fiacco and McCormick [9] (see [12, 17, 18]).

There are, however, many application problems for which efficient general-purpose computational methods can hardly be succesful, even if they are so powerful as the latest generation of interior point methods like OSL of [10] or OB1 of [15]. These are primarily problems with imperfect information. When some of the data c, A or b in (1.1) are uncertain we have to use special modeling techniques to correctly formulate the objective and the constraints. Thare are many ways to do that, but the approach that proved successful in many applications is to describe uncertain quantities by random variables. This not only provides us with the access to deep results of the theory of probability, but also has the advantage of using the language that is understood by experts in various fields.

Describing uncertainty by random quantities gives rise to *stochastic programming problems*. They are usually very difficult and even in the simplest case of linear programming problems of form (1.1) and finitely many events (scenarios) they lead to models with very large numbers of variables and constraints. We shall discuss such models in the next section of the paper. We shall see that their size can be made aribrarily large by more and more exact description of uncertainty, but at the same time they possess a special structure that can be exploited by solution methods. In sections 3 and 4 we shall discuss some ways of specializing interior point methods to such problems.

Needless to say, presentation of such broad and important areas within a short paper is not possible without drastic simplifications. Our aim here is to provide an introduction to the subject for the readers who have not yet had the opportunity to work in both areas. We have no doubt that the demands of practice will drive more and more researchers into this field.

2. Stochastic programming models

2.1 Motivation

In stochastic programming we assume that the uncertain parameters in (1.1) are random variables, i.e. $c = c(\omega)$, $A = A(\omega)$, $h = h(\omega)$ with ω denoting an elementary event in a probability space $(\Omega, \mathcal{F}, P)$. For example, Ω may consist of a finite number of events $\omega_1, \omega_2, \ldots, \omega_L$ occuring with probabilities $p_1, p_2, \ldots, p_L$; each ω_l gives rise to a *scenario* (c_l, S_l, h_l) - a particular instance of problem data.

If we knew the realization of the problem data we could just substitute it into (1.1) and solve the (random) problem

$$\begin{aligned} \min \ & c(\omega)^T x \\ & Dx = d, \\ & S(\omega)x = h(\omega), \\ & x \geq 0; \end{aligned} \tag{2.1}$$

we have split here the matrix A and the vector b in (1.1) into deterministic and stochastic parts:

$$A = \begin{bmatrix} D \\ S(\omega) \end{bmatrix}, \quad b = \begin{bmatrix} d \\ h(\omega) \end{bmatrix}.$$

Obviously, the solution $x(\omega)$ of (2.1) would be dependent on the event ω.

However, if we do not know ω and our decision x has to be determined in advance, ther may be no such x that the constraints of (2.1) hold for every $\omega \in \Omega$. The objective in (2.1) may also be substantially different for different ω. We need a new modeling approach to cover such situations.

Since the objective value

$$F_1(x, \omega) = c(\omega)^T x$$

and the error

$$\Delta(x, \omega) = h(\omega) - S(\omega)x \tag{2.2}$$

are random variables, we may use the properties of their distributions as the objective and constraints in our model. There are many ways to do that - we refer the reader to [8] - but we shall focus here on only one approach: two- and multi-stage models.

2.2 Two-stage models

In a two-stage model we assume that an additional cost $F_2(x,\omega)$ is associated with the error (2.2). The cost is modeled as the minimum objective value in an auxiliary optimization problem. In the simplest case it may have the form

$$\begin{aligned} &\min\ q^T y(\omega) \\ &Wy(\omega) = h(\omega) - S(\omega)x, \\ &y(\omega) \geq 0. \end{aligned} \tag{2.3}$$

We call (2.3) the *second stage problem* and its variables - second stage variables. We denote them by $y(\omega)$ to stress that they may be dependent on the event ω; it is already "known" at this stage. So the whole decision process has now the form:

first stage decision x
observation $c(\omega), S(\omega), h(\omega)$
second stage decision $y(\omega)$

The vector q and the matrix W in (2.3) can be used to assign costs to different components of the error $\Delta(\omega)$. For example, using

$$W = [I\ -I]$$

$$q^T = [q_+^T\ q_-^T]$$

we obtain the following version of (2.3):

$$\begin{aligned} &\min\ q_+^T y_+ + q_-^T y_- \\ &y_+ - y_- = h(\omega) - S(\omega)x, \\ &y_+ \geq 0,\ y_- \geq 0. \end{aligned}$$

Then y_+ and y_- represent surplus and shortage with corresponding cost vectors q_+ and q_-.

The minimum second stage cost

$$F_2(x,\omega) = q^T y(x,\omega),$$

where $y(x,\omega)$ is a solution of (2.3), is a random variable again. We can add it to the direct first stage cost $F_1(x,\omega)$ and formulate the problem

$$\min_{x\in X}\ E\left[F_1(x,\omega) + F_2(x,\omega)\right],$$

where $X = \{x \in R^n :\ Dx = d,\ x \geq 0\}$ and E denotes the expected value. In an extended form the problem can be written as follows:

$$\min_{x\in X}\ \left[Ec(\omega)^T x + E\left\{\min\ \{q^T y \mid Wy = h(\omega) - S(\omega)x,\ y \geq 0\}\right\}\right]. \tag{2.4}$$

It should be stressed that the concept of the second stage problem and second stage variables is not just a modeling trick, but it may reflect real-world situations where additional cost results from the necessity to deal with shortages and excesses. Clearly, one can also use nonlinear penalties in (2.3), for example quadratic or piecewise-quadratic functions of y.

In the case of finitely many scenarios $\omega_1, \omega_2, \ldots, \omega_L$ occuring with probabiblities $p_1, p_2, \ldots, p_L$ we can rewrite (2.4) in a more transparent form. Let us observe that we shall have only L different instances of the second stage problem:

$$\begin{aligned} &\min\ q_l^T y_l \\ &W y_l = h_l - S_l x, \\ &y_l \geq 0, \end{aligned} \tag{2.5}$$

with $l = 1, 2, \ldots, L$; each of them occurs with probability p_l. We can therefore look for x and the solutions $y_1, y_2, \ldots, y_L$ within one large optimization problem

$$\begin{array}{lccccccccc} \min & \bar{c}^T x & + & p_1 q^T y_1 & + & p_2 q^T y_2 & + \ \ldots \ + & p_L q^T y_L & & \\ & Dx & & & & & & & = & d, \\ & S_1 x & + & W y_1 & & & & & = & h_1, \\ & S_2 x & & & + & W y_2 & & & = & h_2, \\ & \vdots & & & & & \ddots & & & \vdots \\ & S_L x & & & & & & + \ W y_L & = & h_L, \\ & x \geq 0, & & y_1 \geq 0, & & y_2 \geq 0, & \ldots & y_L \geq 0, & & \end{array} \tag{2.6}$$

with $\bar{c} = Ec(\omega) = p_1 c_1 + p_2 c_2 + \ldots + p_L c_L$. Indeed, for every fixed x the best values of $y_1, y_2, \ldots, y_L$ in (2.6) are solutions of (2.5).

We end this section by noting that we have obtained a very large linear programming problem of a special structure. Its size grows linearly with the number of scenarios L and may easily reach hundreds of thousands of rows and columns in (2.6).

2.3 Multi-stage models

There are many real-world problems where the decisions are made successively with an increasing information about the uncertain parameters. In the two-stage models of the previous subsection we had the extreme case: at the first stage we did not know anything but the distribution of uncertain data; at the second stage the full information was available. We shall generalize this concept into many successive stages $t = 1, 2, \ldots, T$.

Let us assume that our decision vector x is comprised of subvectors corresponding to successive stages

$$x = (x(1), x(2), \ldots, x(T)).$$

Suppose that the constraints are also composed of subgroups relating decisions from neighboring stages

$$S(t)x(t-1) + W(t)x(t) = h(t), \quad t = 1, 2, \ldots, T, \tag{2.7a}$$

$$x(t) \geq 0, \quad t = 1,2,\ldots,T, \tag{2.7b}$$

($D(1)$ and $x(0)$ are empty). The deterministic multi-stage model can be now formulated as the problem of minimizing the objective

$$f(x) = c(1)^T x(1) + c(2)^T x(2) + \ldots + c(T)^T x(T)$$

subject to the constraints (2.7).

Let us now suppose that the data in the above problem are uncertain, i.e. $S(t) = S(t,\omega)$, $W(t) = W(t,\omega)$, $h(t) = h(t,\omega)$, $c(t) = c(t,\omega)$ for $\omega \in \Omega$, where $(\Omega, \mathcal{F}, P)$ is some probability space. The principal question that has to be answered in such a situation is what is the information available when $x(t)$ has to be chosen. In multi-stage stochastic programming problems we assume that at time t we observe

$$\xi(t,\omega) = (S(t,\omega), W(t,\omega), h(t,\omega), c(t,\omega)).$$

Then the decision process has the following form:

observation $\xi(1,\omega)$
decision $x(1,\omega)$
observation $\xi(2,\omega)$
decision $x(2,\omega)$
$\vdots$
observation $\xi(T,\omega)$
decision $x(T,\omega)$

It is natural that decision $x(t)$ may depend on the data $\xi(\tau)$ observed at time instants $\tau = 1,2,\ldots,t$. However, it must not depend on future data $\xi(\tau)$, $\tau > t$. This requirement is called the condition of *nonanticipativity.*

The nonanticipativity condition can be best illustrated for problems with finitely many events, i.e. with $\Omega = \{\omega_1, \omega_2, \ldots, \omega_L\}$. They result in finitely many *scenarios*

$$\xi_l = (\xi_l(1), \xi_l(2), \ldots, \xi_l(T)), \quad l = 1,2,\ldots,L.$$

The scenarios can be put into a tree whose levels correspond to time stages. At level 1 we put $\xi(1)$ - the already known data for stage 1, common for all scenarios. At level 2 we put so many nodes as many different values of $\xi(2)$ may occur. To each such node we attach nodes at level 3, that correspond to different values of $\xi(3)$ that may follow the particular sequence of $\xi(1), \xi(2)$, etc. This is illustrated in Figure 1 in the case of four stages and two possibilities of data values at each stage. A scenario corresponds to a path in this tree. Nonanticipativity then means that there can be only so many versions of decision $x(t)$ as many nodes occur in the scenario tree at level t. For the example of Figure 1 we must have one $x(1)$ common for all scenarios, two versions of $x(2)$ (one for each outcome of the observation of $\xi(2)$), four versions of $x(3)$, etc.

Formally, we can formulate a multi-stage stochastic programming problem as follows:

$$\min \left\{ E \sum_{t=1}^{T} c(t,\omega)^T x(t,\omega) \right\} \tag{2.8a}$$

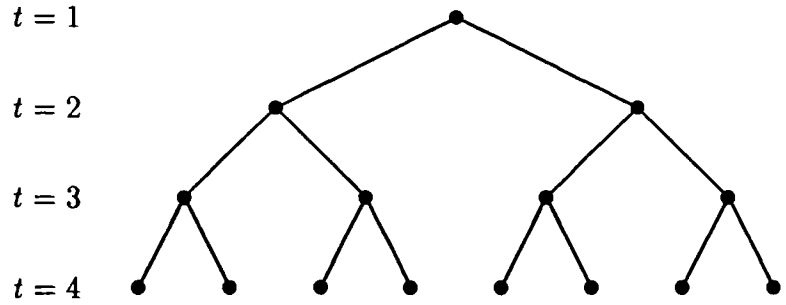

Figure 1: Scenario tree.

subject to the constraints

$$S(t,\omega)x(t-1,\omega) + W(t,\omega)x(t,\omega) = h(t,\omega), \quad t = 1,2,\ldots,T, \quad \omega \in \Omega, \tag{2.8b}$$

$$x(t,\omega) \geq 0, \tag{2.8c}$$

and the additional nonanticipativity constraint that restricts the dependence of $x(t,\omega)$ on ω to the events that can be distinguished on the basis of information collected up to time t.

3. Direct application of interior point methods

3.1 The primal-dual method for quadratic problems

Let us consider the quadratic programming problem

$$\begin{aligned} \min \ & c^T x + \frac{1}{2} x^T Q x \\ & Ax = b, \\ & x \geq 0. \end{aligned} \tag{3.1}$$

Applying the logarithmic barrier function to the inequalities in (3.1) we obtain the approximate problem

$$\begin{aligned} \min \ & c^T x + \frac{1}{2} x^T Q x - \mu \sum_{j=1}^{n} \ln x_j \\ & Ax = b, \end{aligned} \tag{3.2}$$

where $\mu > 0$ is some small penalty parameter. It is well known that if (3.1) is solvable then the accumulation point of the solutions $x(\mu)$ of (3.2) with $\mu \downarrow 0$ are optimal solutions of (3.1) (see [9]).

The necessary conditions for (3.2) can be written as follows

$$-Qx + A^T y + \mu X^{-1} e = c,$$

$$Ax = b,$$

where y is the vector of Lagrange multipliers, $e = [1\ 1\ \ldots\ 1]^T$, and X is a diagonal matrix of dimension n with $X_{jj} = x_j$, $j = 1, 2, \ldots, n$. Denoting the reduced costs in the original problem by z we can rewrite the last system as follows

$$-Qx + A^T y + z = c,$$

$$Ax = b,$$

$$XZe = \mu e.$$

Here, again, Z is a diagonal matrix of dimension n with $Z_{jj} = z_j$, $j = 1, 2, \ldots, n$. This system of nonlinear equations can be iteratively solved by the Newton's method. The linearized system for Newton's directions d_x, d_y and d_z takes on the form

$$\begin{bmatrix} -Q & A^T & I \\ A & & \\ Z & & X \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} = \begin{bmatrix} c + Qx - A^T y - z \\ b - Ax \\ \mu e - XZe \end{bmatrix}. \tag{3.3}$$

In the primal-dual interior point method one solves (3.3) with some μ and makes a step

$$x^{\text{new}} = x + \alpha d_x,$$

$$y^{\text{new}} = y + \alpha d_y,$$

$$z^{\text{new}} = z + \alpha d_z,$$

with $\alpha > 0$ such that x^{new} and z^{new} remain positive. Finally, a smaller μ is calculated and the iteration continues. There are many important technical details concerning the choice of α, μ and of the point at which the linearized equations are solved; we refer the reader to [5, 6, 14, 19].

Computational realization of the primal-dual iterations amounts mainly to repeated solution of the system of linear equations (3.3). Obviously, we can easily eliminate d_z from (3.3) and arrive to the *augmented system*

$$\begin{bmatrix} -(Q + X^{-1}Z) & A^T \\ A & \end{bmatrix} \begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} r \\ s \end{bmatrix}, \tag{3.4}$$

with $r = c - A^T y + Qx - \mu X^{-1} e$ and $s = b - Ax$. Further elimination yields the *normal system*

$$A(Q + X^{-1}Z)^{-1} A^T d_y = s + A(Q + X^{-1}Z)^{-1} r. \tag{3.5}$$

The standard approach is to solve the normal system for d_y and then calculate d_x and d_z from (3.4) and (3.3).

Crucial for the efficiency of the whole approach is that the nonzero pattern of the matrix

$$M = A\Theta^{-1}A^T, \tag{3.6}$$

where $\Theta = Q + X^{-1}Z$, does not depend on x and z. Therefore the sparse Cholesky factorization of M, i.e. selection of a permutation P and a lower triangular matrix L such that

$$PMP^T = LL^T, \tag{3.7}$$

can be carried out in two steps. First, we can compute the pivot ordering P and the nonzero pattern of L by analysing the location of nonzeros in M. This is called *symbolic factorization* and need to be carried out only once. Next, when x and z are known we can just calculate the numerical values of the entries of M and follow the recipes prepared in the symbolic step to fill-in the nonzeros of L. This part of the solution must be repeated at least once per iteration; its efficiency is heavily dependent on the quality of the symbolic factorization and on sparsity of the factor L.

Finally, it is worth noting that when Q is diagonal then the sparsity pattern of M is the same as for linear problems. Thus the computational effort per iteration is for separable quadratic programs almost the same as for the corresponding linear problems.

3.2 The problem of dense columns

When the primal-dual interior point method is applied to stochastic programming problems we encounter difficulties created by the characteristic structure of the constraint matrix.

Let us study this issue in detail on the two-stage problem first. The constraint matrix of (2.6) has the form

$$A = \begin{bmatrix} D & & & & \\ S_1 & W & & & \\ S_2 & & W & & \\ \vdots & & & \ddots & \\ S_L & & & & W \end{bmatrix}. \tag{3.8}$$

Therefore, even for a diagonal or empty Q, the nonzero pattern of M in (3.7) will be the same as the nonzero pattern of

$$AA^T = \begin{bmatrix} DD^T & DS_1^T & DS_2^T & \cdots & DS_L^T \\ S_1D^T & S_1S_1^T + WW^T & S_1S_2^T & \cdots & S_1S_L^T \\ S_2D^T & S_2S_1^T & S_2S_2^T + WW^T & \cdots & S_2S_L^T \\ \vdots & & & \ddots & \\ S_LD^T & S_LS_1^T & S_Ls_2^T & \cdots & S_LS_L^T + WW^T \end{bmatrix}.$$

Needless to say, a significant fill-in takes place during this multiplication; the number of nonzeros of AA^T becomes excessively large and Cholesky factorization (3.7) involves many eliminations. We have to avoid that if we want to solve problems of realistic sizes.

Clearly, the source of these difficulties are the columns corresponding to the first stage variables. Splitting A into

$$A = [A_d \; A_s] \tag{3.9}$$

with

$$A_d = \begin{bmatrix} D \\ S_1 \\ S_2 \\ \vdots \\ S_L \end{bmatrix}, \quad A_s = \begin{bmatrix} W & & & \\ & W & & \\ & & \ddots & \\ & & & W \end{bmatrix} \tag{3.10}$$

we see that

$$\begin{aligned} AA^T &= A_d A_d^T + A_s A_s^T \\ &= \begin{bmatrix} DD^T & DS_1^T & \cdots & DS_L^T \\ S_1 D^T & S_1 S_1^T & \cdots & S_1 S_L^T \\ \vdots & & \ddots & \\ S_L D^T & S_L S_1^T & \cdots & S_L S_L^T \end{bmatrix} + \begin{bmatrix} WW^T & & & \\ & & \ddots & \\ & & & WW^T \end{bmatrix}. \end{aligned}$$

The fill-in created by the columns coreesponding to second stage variables is restricted to the diagonal blocks WW^T. It is clear that we need special techniques to deal with first-stage variables, since otherwise our ability to solve large problems will be very limited.

3.3 Schur complements an rank deficiency

One of general approaches to dense columns in the sparse Cholesky factorization is the use of *Schur complements*. For a diagonal Θ and A given by (3.9) we can write

$$A\Theta^{-1}A^T = A_s\Theta_s^{-1}A_s^T + A_d\Theta_d^{-1}A_d^T. \tag{3.11}$$

Suppose that we know the Cholesky factors for the sparse part

$$A_s\Theta_s^{-1}A_s^T = CC^T; \tag{3.12}$$

for simplicity we neglect here the permutations. Then, applying the Sherman-Morrison formula to the inverse of (3.11) we obtain

$$(A\Theta^{-1}A^T)^{-1} = C^{-T}C^{-1} - C^{-T}C^{-1}A_d(\Theta_d + A_d^T C^{-T}C^{-1}A_d)^{-1}A_d^T C^{-T}C^{-1}. \tag{3.13}$$

Application of formula (3.13) to equation (3.5) requires an additional factorization of a small dense matrix

$$B = \Theta_d + A_d^T C^{-T}C^{-1}A_d. \tag{3.14}$$

This matrix is the *Schur complement* of $A_s\Theta_s^{-1}A_s^T$ in the matrix

$$\begin{bmatrix} A_s\Theta_s^{-1}A_s^T & A_d \\ A_d^T & -\Theta_d \end{bmatrix}. \tag{3.15}$$

Solving the system (3.5) can be then reduced to three systems with CC^T and one system with B.

In our case, however, the straightforward application of the Schur complement technique encounters serious difficulties. Indeed, for the constraint matrix of form (3.8), if we define A_d and A_s by (3.10), the matrix

$$M_s = A_s \Theta_s^{-1} A_s^T$$

will be rank-deficient and the Schur transformation will not be correct (C in (3.12) will not be invertible).

Since it is necessary to have A_s of the full row rank, we need to apply some special tricks to achieve that. One possibility is to include into A_s some columns corresponding to the first-stage variables. This can be easy, if there are sufficiently many first stage variables that occur only in the deterministic constraints (e.g. slack variables for these constraints). Then

$$\begin{bmatrix} D \\ S_1 \\ S_2 \\ \vdots \\ S_L \end{bmatrix} = \begin{bmatrix} D_1 & D_2 \\ S_{11} & \\ S_{21} & \\ \vdots & \\ S_{L1} & \end{bmatrix} \tag{3.16}$$

and if D_2 and W have full row rank we can define

$$A_s = \begin{bmatrix} D_2 & & & & \\ & W & & & \\ & & W & & \\ & & & \ddots & \\ & & & & W \end{bmatrix}. \tag{3.17}$$

The financial planning problems analysed in [16] have such a property.

Another technique for correcting the rank of $A_s \Theta_s^{-1} A_s^T$ was suggested in [4]. It is based on the transformation

$$\begin{aligned} AA^T &= \begin{bmatrix} DD^T & DS_1^T & \cdots & DS_L^T \\ S_1 D^T & S_1 S_1^T & \cdots & S_1 S_L^T \\ \vdots & & \ddots & \\ S_L D^T & S_L S_1^T & \cdots & S_L S_L^T \end{bmatrix} + \begin{bmatrix} & WW^T & & \\ & & \ddots & \\ & & & WW^T \end{bmatrix} \\ &= \begin{bmatrix} DD^T - I & DS_1^T & \cdots & DS_L^T \\ S_1 D^T & S_1 S_1^T & \cdots & S_1 S_L^T \\ \vdots & & \ddots & \\ S_L D^T & S_L S_1^T & \cdots & S_L S_L^T \end{bmatrix} + \begin{bmatrix} I & & & \\ & WW^T & & \\ & & \ddots & \\ & & & WW^T \end{bmatrix} \end{aligned} \tag{3.18}$$

The sparse part has a full row rank (if W has) and

$$\begin{bmatrix} DD^T - I & DS_1^T & \cdots & DS_L^T \\ S_1 D^T & S_1 S_1^T & \cdots & S_1 S_L^T \\ \vdots & & \ddots & \\ S_L D^T & S_L S_1^T & \cdots & S_L S_L^T \end{bmatrix} = UV^T$$

with

$$U = \begin{bmatrix} D & I \\ S_1 & 0 \\ \vdots & \\ S_L & 0 \end{bmatrix}, \quad V = \begin{bmatrix} D & -I \\ S_1 & 0 \\ \vdots & \\ S_L & 0 \end{bmatrix}$$

We can now apply the asymmetric Sherman-Morrison formula

$$(AA^T)^{-1} = C^{-T}C^{-1} - C^{-T}C^{-1}UB^{-1}V^TC^{-T}C^{-1}.$$

with the "Schur complement":

$$B = \begin{bmatrix} B_1 & D^T \\ -D & \end{bmatrix}$$

$$B_1 = I^2 + D^TD + \sum_{l=1}^{L} S_l^T(W_lW_l^T)^{-1}S_l$$

Some encouraging results have already been obtained in [2].

Carpenter [5] reports computational experience with the Schur complement method for a special class of two-stage problems where the form (3.16)-(3.17) could be obtained with a full-rank A_s. She observed a dramatic (five- to ten-fold) improvement in solution time over the straigthforward approach. However, even in such medium-size problems a loss of numerical accuracy has been observed. This has also been noted for general LP problems in [6] and [15].

Generally, the loss of numerical accuracy and instability for larger problems is the main drawback of the Scur complement approach. There are two sources of it:

- ill-conditioning or rank deficiency of the sparse part;

- difficulties in the factorization of the dense Schur complement matrix.

Both of them occur in stochastic programming.

For multi-stage stochastic programs the situation is more difficult. Let us illustrate the problem on the example from Figure 1. The constraint matrix has the form

$$A = \begin{bmatrix}
W & & & & & & & & & & & & & & \\
S & W & & & & & & & & & & & & & \\
S & & W & & & & & & & & & & & & \\
 & S & & W & & & & & & & & & & & \\
 & S & & & W & & & & & & & & & & \\
 & & S & & & W & & & & & & & & & \\
 & & S & & & & W & & & & & & & & \\
 & & & S & & & & W & & & & & & & \\
 & & & S & & & & & W & & & & & & \\
 & & & & S & & & & & W & & & & & \\
 & & & & S & & & & & & W & & & & \\
 & & & & & S & & & & & & W & & & \\
 & & & & & S & & & & & & & W & & \\
 & & & & & & S & & & & & & & W & \\
 & & & & & & S & & & & & & & & W
\end{bmatrix},$$

where for simplicity we use S and W to denote all $S_l(t)$ and $W_l(t)$. To follow the Schur complement approach described here we would have to incorporate into A_d the columns corresponding to all stages but the last one. This would result in a rank-deficient A_s and would call for some special correcting devices, such as that shown in (3.18). But also, for nontrivial problems, we would obtain a very large "dense" matrix B defined by (3.14); much larger than any dense factorization method can handle. So there is a need to carefully select the columns to be involved into A_d so that A_s will be sparse enough and B small.

Such a technique can be based on the augmented system (3.4). Instead of eliminating analytically d_x and arriving to (3.5) we can deal directly with (3.4) and try to construct the factorization

$$U^T U = P \begin{bmatrix} -\Theta & A^T \\ A & \end{bmatrix} P^T \tag{3.19}$$

with a sparse, upper-triangular U and some permutation P. Let us observe that if the permutation P is such that the first pivots are from Θ, then (3.19) will become equivalent to (3.5). However, when we delay processing some of the diagonals of Θ (and the corresponding columns of A), we can obtain different factors. To illustrate that, suppose that we split A as in (3.9) and we choose P in (3.19) such that the columns from A_d appear at the last positions (last pivots will be from Θ_d). We would then obtain the matrix

$$\begin{bmatrix} -\Theta_s & A_s^T & \\ A_s & & A_d \\ & A_d^T & -\Theta_d \end{bmatrix}. \tag{3.20}$$

Carrying out the elimination for the first pivots (in Θ_s) yields the system (3.15). Therefore the Schur complement technique is algebraically equivalent to a particular pivot selection in the augmented system (3.19). However, the advantage of (3.19) is that the pivot ordering (or division into sparse and dense parts in (3.20)) need not be specified in advance, but is determined dynamically on-line to satisfy sparsity and stability requirements. Encouraging computational results with this approach have been reported in [11, 26, 27].

3.4 Split-variable formulation

There is an explicit way of formulating the non-anticipativity constraints in stochastic programming models that provides us with an additional insight into their properties and allows for the development of new computational techniques. Namely, with every scenario

$$\xi_l = (\xi_l(1), \xi_l(2), \ldots, \xi_l(T))$$

we can associate the corresponding sequence of decisions

$$x_l = (x_l(1), x_l(2), \ldots, x_l(T))$$

in the problem (2.8). The sequences x_l, $l = 1, 2, \ldots, L$, however, cannot be independent. Nonanticipativity requires that

$$x_l(t) = x_k(t) \quad \text{if} \quad \xi_l(\tau) = \xi_k(\tau) \quad \text{for} \quad \tau = 1, 2, \ldots, t. \tag{3.21}$$

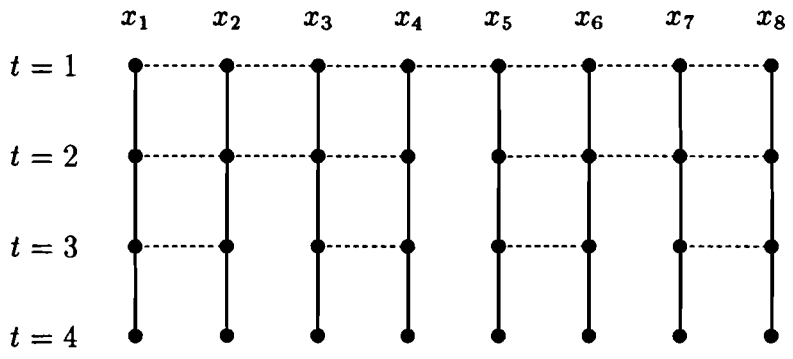

Figure 2: Sequences of decisions and nonanticipativity.

For the example of Figure 1 the relations between the sequences are illustrated in Figure 2; the horizontal dotted lines denote equality. Such an explicit approach to nonanticipativity allows us to formulate multi-stage stochastic programming problems as follows

$$\min \sum_{l=1}^{L} p_l \langle c_l, x_l \rangle \tag{3.22a}$$

$$x_l \in X_l, \quad l = 1, 2, \ldots, L, \tag{3.22b}$$

$$x_l(t) = x_k(t) \quad \text{for some} \quad l, k, t. \tag{3.22c}$$

Here $c_l = (c_l(1), c_l(2), \ldots, c_l(T))$, the set X_l represents the dynamics of the system in scenario l given by the constraints

$$S_l(t)x_l(t-1) + W_l(t)x_l(t) = h_l(t), \quad t = 1, 2, \ldots, T, \tag{3.23a}$$

$$x_l(t) \geq 0 \tag{3.23c}$$

and (3.22c) represents the nonanticipativity constraint (3.21). It is obvious that constraints (3.21) are redundant and that we can work with a carefully selected subset of them. One of possibilities is to arrange some ordering among the scenarios and to relate $x_l(t)$ to only one other scenario in its equivalence group. We can generate for every t a permutation

$$\nu(l, t), \; l = 1, 2, \ldots, L$$

such that $\nu(l, t)$ points to a *sibling* of scenario l at stage t defined as follows. It is the next scenario $l+1$, if l and $l+1$ have the common past up to t; otherwise it is the first scenario among those that have common past with l up to stage t. We assume that the scenarios are ordered in such a way that each of them is followed by the one that has as much in common with it as possible from the scenarios having larger numbers. For the example of Figure 1 such a mapping would have the form shown in Table 1.

The nonanticipativity constraint can be then written as

$$x_l(t) = x_{\nu(l,t)}(t), \quad l = 1, 2 \ldots, L, \quad t = 1, 2 \ldots, T. \tag{3.24}$$

Time stage	Scenario							
	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	1
2	2	3	4	1	6	7	8	5
3	2	1	4	3	6	5	8	7
4	1	2	3	4	5	6	7	8

Table 1: Siblings of scenarios.

There is a number of advantages of the split-variable formulation (3.22). The most visible one is the change of the structure of the constraint matrix of a stochastic problem. Let us illustrate this on the case of a two-stage problem (2.6). In the split-variable formulation it takes on the form:

$$\begin{array}{lccccccccc} \min & p_1 c_1^T x_1 & +p_2 c_2^T x_2 & +\ldots & +p_l c_l^T x_L & +p_1 q^T y_1 & +\ldots & +p_l q^T y_L & & \\ & D x_1 & & & & & & & = & d, \\ & x_1 & -x_2 & & & & & & = & 0, \\ & & x_2 & -x_3 & & & & & = & 0, \\ & & & \ddots & & & & & & \vdots \\ & & & x_{L-1} & -x_L & & & & = & 0, \\ & S_1 x_1 & & & & +W y_1 & & & = & h_1, \\ & & \ddots & & & & \ddots & & & \vdots \\ & & & & S_L x_L & & & +W y_L & = & h_L, \\ & x_1 \geq 0, & x_2 \geq 0, & \ldots & x_L \geq 0, & y_1 \geq 0, & \ldots & y_L \geq 0. & & \end{array} \tag{3.25}$$

The constraint matrix has the form

$$A = \begin{bmatrix} D & & & & & & \\ I & -I & & & & & \\ & \ddots & & & & & \\ & & I & -I & & & \\ S_1 & & & & W & & \\ & \ddots & & & & \ddots & \\ & & & S_L & & & W \end{bmatrix}.$$

There are no more dense columns in this formulation (at most three blocks in each column), so the Cholesky factorization of AA^T may be much sparser than in the original formulation. The experiments of [16] show that the split-variable formulation is always better than the compact formulation; the factors have less nonzeros and the cost of solution can be decreased 2-3 times. The computation time is still higher than in the Schur complement approaches, but no stability difficulties have been observed. It should be stressed, however, that the decrease of the number of nonzeros in the Cholesky factors is not so high as the decrease of the number of nonzeros in AA^T; the

additional nonanticipativity constraints create a significant fill-in during the elimination process.

Summing up, splitting variables is a simple and general technique for dealing with dense columns in stochastic programming. It is reliable and always leads to moderate gains over the straightforward formulation.

3.5 Solving the dual problem

Since the constraint matrix of the dual problem is the transpose of the matrix of primal contstraints, it seems that a simple remedy to our problems is formulating and solving the dual of (2.6):

$$\begin{aligned} &\max\ b^T y \\ &A^T y \leq c, \end{aligned}$$

with A of form (3.8). However, y is not sign-constrained, so in order to avoid getting the system (3.3) again, we have to split y into

$$\begin{aligned} y &= y^1 - y^2, \\ y^1 &\geq 0,\ y^2 \geq 0. \end{aligned} \tag{3.26}$$

This yields the problem

$$\begin{aligned} &\max\ b^T y^1 - b^T y^2 \\ &A^T y^1 - A^T y^2 + z = c, \\ &y^1 \geq 0,\ y^2 \geq 0,\ z \geq 0. \end{aligned} \tag{3.27}$$

We can now formulate the logarithmic barrier function and arrive to a new primal-dual method for the dual problem (3.27), in a similar way to section 3.1. This idea was analysed in [25] and [3]. However, passing to the dual and splitting variables by (3.26) increases the size of the problem considerably. Even more dangerous is that the transformation (3.26) introduces an inherent ill-conditioning into direction-finding systems due to the non-uniqueness of the dual solution: usually both y_j^1 and y_j^2 grow and only their difference is convergent to y_j (see [25]).

4. Decomposition

4.1 The augmented Lagrangian

Let us recall the split-variable formulation of multi-stage problems. For each scenario l the feasible set has the form

$$X_l = \{x_l :\ A_l x_l = h_l,\ x_l \geq 0\}, \quad l = 1, \ldots, L,$$

where A_l is a staircase matrix resulting from the constraints (3.23a):

$$A_l = \begin{bmatrix} W_l(1) & & & \\ S_l(2) & W_l(2) & & \\ & \ddots & & \\ & & S_l(T) & W_l(T) \end{bmatrix} \tag{4.1}$$

The non-anticipativity constraints (3.24) can be put together into some joining constraint $x = Ux$, which, although simple, is nonseparable and multidimensional. Thus, the whole problem has the following structure:

$$\min \sum_{l=1}^{L} p_l \langle c_l, x_l \rangle \tag{4.2a}$$

subject to

$$x - Ux = 0, \tag{4.2b}$$

$$x_l \in X_l, \quad l = 1, \ldots, L. \tag{4.2c}$$

The well-known *Dantzig-Wolfe decomposition method* is the classical approach to such problems. It may be viewed as a dual method based on the *Lagrangian function*

$$\begin{aligned} L(x\ , \pi) &= \sum_{l=1}^{L} p_l \langle c_l, x_l \rangle + \langle \pi, x - Ux \rangle \\ &= \sum_{l=1}^{L} \sum_{t=1}^{T} p_l \langle c_l(t), x_l(t) \rangle + \sum_{l=1}^{L} \sum_{t=1}^{T-1} \langle \pi_l(t), x_l(t) - x_{\nu(l,t)}(t) \rangle. \end{aligned}$$

The Lagrangian function is separable into terms dependent on x_l, $l = 1, \ldots, L$:

$$L(x\ , \pi) = \sum_{l=1}^{L} \langle \bar{c}_l, x_l \rangle$$

with

$$\bar{c}_l(t) = \begin{cases} p_l c_l(t) + \pi_l(t) - \pi_{\nu^{-1}(l,t)}(t) & \text{if } t < T, \\ p_l c_l(t) & \text{if } t = T, \end{cases} \tag{4.3}$$

and can be minimized subject to (4.2c) independently for each x_l. However, updating multipliers π requires solution of a linear *master problem* which has the number of rows equal to the number of rows in (4.2b) and unspecified number of columns. This makes the Dantzig-Wolfe method hard to implement for multi-stage stochastic programming problems.

An alternative dual approach to (4.2) is the use of the *augmented Lagrangian function*

$$\begin{aligned} \Lambda_r(x\ , \pi) &= \sum_{l=1}^{L} p_l \langle c_l, x_l \rangle + \langle \pi, x - Ux \rangle + \frac{1}{2} r \parallel x - Ux \parallel^2 \\ &= \sum_{l=1}^{L} \sum_{t=1}^{T} p_l \langle c_l(t) x_l(t) \rangle + \sum_{l=1}^{L} \sum_{t=1}^{T-1} \langle \pi_l(t), x_l(t) - x_{\nu(l,t)}(t) \rangle \\ &\quad + \frac{1}{2} r \sum_{l=1}^{L} \sum_{t=1}^{T-1} \parallel x_l(t) - x_{\nu(l,t)}(t) \parallel^2 . \end{aligned} \tag{4.4}$$

Here $r > 0$ is a penalty parameter. The *augmented Lagrangian method* (see, e.g., [1]) can be stated for (4.2) as follows.

Algorithm 1

1 For fixed multipliers π^k solve the problem

$$\min \ \Lambda_r(x\ ,\pi^k) \text{ subject to } x \in X_1 \times \ldots \times X_L. \tag{4.5}$$

Let $x^k = (x_1^k\ , x_2^k, ..., x_L^k)$ be the solution of (4.5).

2 If $x_l^k(t) = x_{\nu(l,t)}^k(t)$ for all l and t, then stop (optimal solution found); otherwise set for $l = 1, \ldots, L$ and $t = 1, \ldots, T-1$

$$\pi_l^{k+1}(t) = \pi_l^k(t) + r(x_l^k(t) - x_{\nu(l,t)}^k(t)), \tag{4.6}$$

increase k by 1 and go to **1**.

There is a number of general advantages of the augmented Lagrangian approach over usual dual methods: simplicity and stability of multiplier iterations and possibility of starting from arbitrary π^0 are among the most important ones. It is also well known that if (4.2) has a solution, then Algorithm 1 is finitely convergent.

One of the "disadvantages" of the augmented Lagrangian approach is that the objective in (4.5) is no longer linear, but quadratic. But for interior point methods it is not a disadvantage any more, especially in our case, where both the constraint matrix A and the Hessian Q in (3.1) are sparse. Indeed for (4.5) we have

$$A = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_L \end{bmatrix} \tag{4.7}$$

with each A_l of form (4.1). Next, since the quadratic terms in (4.4) relate only neigboring scenarios we shall have Q with at most 3 nonzeros in each row and column: one diagonal entry with value $2r$ and two off-diagonals with value $-r$. Therefore Cholesky factorization of (3.5), and especially of (3.4), can be quite successful. Nevertheless the presence of off-diagonal entries in Q creates a substantial fill-in in the factors; it is analogous to the effect of linking constraints in the split-variable formulation (3.25).

In the next section we shall present an approach based on augmented Lagrangians, in which we shall remove off-diagonal elements of Q. It will make the factorization much easier and it will even allow a parallel distributed solution of the problem (for another approach to augmented Lagrangian decomposition see [22]).

4.2 Diagonal quadratic approximation

Clearly, nonseparability of (4.4) is due to the quadratic terms

$$\| x_l(t) - x_{\nu(l,t)}(t) \|^2 \tag{4.8}$$

which contain cross-products $\langle x_l(t), x_{\nu(l,t)}(t)\rangle$. Suppose that x belongs to a neighborhood of some reference point $\tilde{x}$. As noted in [20], we can use then the technique of Stephanopoulos and Westerberg [24] and approximate the cross-products locally by

$$\langle x_l(t), x_{\nu(l,t)}(t)\rangle \approx \langle x_l(t), \tilde{x}_{\nu(l,t)}(t)\rangle + \langle \tilde{x}_l(t), x_{\nu(l,t)}(t)\rangle - \langle \tilde{x}_l(t), \tilde{x}_{\nu(l,t)}(t)\rangle \qquad (4.9)$$

with an error of order $O(\| x - \tilde{x} \|^2)$. Using (4.9) in (4.8) and then in (4.4) we see that problem (3.5) can be in the neighborhood of $\tilde{x}$ approximated by L subproblems

$$\begin{aligned} \min \tilde{\Lambda}_r^l(x_l, \pi; \tilde{x}) \;=\; & \sum_{t=1}^{T} \langle \bar{c}_l(t), x_l(t)\rangle + \\ & \frac{1}{2} r \sum_{t=1}^{T-1} \{ \| x_l(t) - \tilde{x}_{\nu(l,t)}(t) \|^2 + \| x_l(t) - \tilde{x}_{\nu^{-1}(l,t)}(t) \|^2 \} \end{aligned} \qquad (4.10a)$$

subject to

$$x_l \in X_l, \qquad (4.10b)$$

with $\bar{c}_l(t)$ defined by (4.3).

The approximation point $\tilde{x}$ can be iteratively updated by the following algorithm (see [21, 23]).

Algorithm 2

1 Set $\pi = \pi^k$, $\tilde{x}^{k,m} = x^{k-1}$ *and* $m = 1$.

2 For $l = 1, \ldots, L$ solve (4.10) with $\tilde{x} = \tilde{x}^{k,m}$ obtaining new points $x_l^{k,m}$.

3 If $\| x_l(t) - \tilde{x}_l(t) \| \leq \epsilon$ for all l and $t < T$, where $\epsilon > 0$ is some prescribed accuracy, then stop; otherwise set for $l = 1, \ldots, L$, $t = 1, \ldots, T$

$$\tilde{x}_l^{k,m+1}(t) = \tilde{x}_l^{k,m}(t) + \alpha(x_l^{k,m}(t) - \tilde{x}_l^{k,m}(t)), \qquad (4.11)$$

increase m by 1 and go to **2**.

We can use this approximation in two ways.

First, we can just apply the interior point method of section 3.1 to the collection of problems (4.10) treated as one big problem - similarly to (4.5). Then, owing to the absence of off-diagonal elements in the Hessian matrix, the nonzero pattern of (3.6) will be the same as of

$$AA^T = \begin{bmatrix} A_1 A_1^T & & & \\ & A_2 A_2^T & & \\ & & \ddots & \\ & & & A_L A_L^T \end{bmatrix}.$$

This will not only make the factorization easier, but allow a block-wise factorization of (3.7). The experiments reported in [20] show that removing off-diagonal entries of Q dramatically improves the quality of Cholesky factorization: the number of nonzeros in the factors (3.7) decreases at least three times.

Secondly, we can deal with the problems (4.10) in parallel and have an independent interior point algorithm running for each of them. We can run them in parallel and exchange only the information that is necessary to continue computation.

Expanding the quadratic penalty term in (4.10a) and neglecting terms that do not depend on x_l we arrive to the following formulation

$$\min \ \tilde{c}_l^T x_l + \frac{1}{2} x_l^T Q_l x_l \tag{4.12a}$$

subject to

$$A_l x_l = b_l \ , \tag{4.12b}$$

$$x_l \geq 0 \ , \tag{4.12c}$$

where

$$\tilde{c}_l(t) = \begin{cases} c_l(t) + \pi_l(t) - \pi_{\nu^{-1}(l,t)}(t) - r\tilde{x}_{\nu(l,t)}(t) - r\tilde{x}_{\nu^{-1}(l,t)}(t) & \text{if } t < T, \\ c_l(t) & \text{if } t = T, \end{cases} \tag{4.13}$$

and Q_l is a diagonal matrix with the diagonal elements equal to $2r$ for variables which have siblings, and 0 otherwise.

Problem (4.12) is a separable quadratic programming problem, so, identically as in section 3.1 we can use logarithmic barrier functions for nonnegativity constraints. We can solve (in one way or another) the system (3.3) for each l and make steps

$$x_l^{\text{new}} = x_l + \alpha_l d_{xl}, \tag{4.14a}$$

$$y_l^{\text{new}} = y_l + \alpha_l d_{yl}, \tag{4.14b}$$

$$z_l^{\text{new}} = z_l + \alpha_l d_{zl}, \tag{4.14c}$$

with stepsize α_l and barrier coefficient μ_l calculated by appropriate rules [6, 16, 14].

In this way we can proceed until the solution of (4.12) will be found. However, we may stop after a small number of iterations (or just one) and update the approximation point $\tilde{x}$ by (4.11).

In general, change of $\tilde{x}$ results in the change of $\tilde{c}_l$ given by (4.13) which in turn causes the loss of dual feasibility. But we are still in an interior point so it is possible to continue the iterations without essential difficulties. It is even better to change $\tilde{x}$ more frequently to modify the trajectory further from the boundary.

Summing up, the iterative character of barrier methods and their ability to start from arbitrary interior points makes it possible to integrate the algorithm for solving (4.10) with the updates of the approximation point (4.11).

4.3 A parallel distributed implementation

The Diagonal Quadratic Approximation method is a flexible theoretical scheme which can be implemented in many ways, in particular, in a parallel distributed environment.

Let us assume that we have a sufficiently large number of processors and let us assign each scenario to a different processor. Then it is clear that iterations (4.14) can

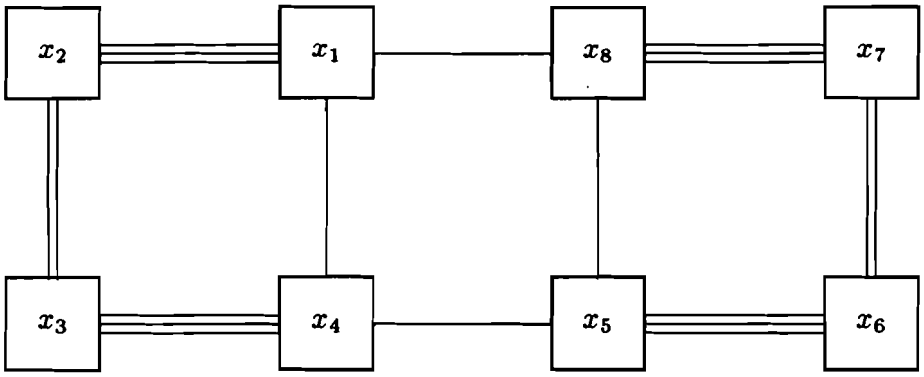

Figure 3: Communication between subproblems by full decomposition.

be carried out (for a fixed $\tilde{x}$) independently in each processor. It is also true for the algorithm for updating $\tilde{x}$ by (4.11).

Although the change of $\tilde{x}$ causes changes in $\tilde{c}$ in (4.12), it follows from (4.13) that the interactions occur only between siblings at a given stage: $\tilde{x}_l(t)$ influences the cost $\tilde{c}_j(t)$ in scenarios $j = \nu(l,t)$ and $j = \nu^{-1}(l,t)$, and vice versa. So, we derive a communication structure between processors, in which messages need to be passed between nearest neighbors. For the example of Figure 2 and Table 1 the communication structure is depicted in Figure 3. The number of lines joining subproblems in Figure 3 denotes the number of stages at which the siblings need to exchange data.

The multiplier iteration (4.6) can also be carried out in a distributed fashion. Indeed, when Algorithm 2 stops, we have $x_l(t) = \tilde{x}_l(t)$, so $x_l(t)$ is known to the sibling $\nu(l,t)$, while $x_{\nu(l,t)}$ is known to subproblem i. Thus, both i and $\nu(l,t)$ can calculate $\pi(l,t)$ by (4.6). Clearly, the same argument applies to the sibling $\nu^{-1}(l,t)$. Thus, all subproblems can update their costs (4.13) in a distributed fashion.

It is worth stressing that in this way we arrived to a decomposition method without any coordinating unit. What once was a large master problem in the Dantzig-Wolfe method became in our approach a simple shift of costs (4.13) distributed among subproblems.

Assigning each scenario to a different processor is not the only possibility to distribute the DQA method. We can form larger subproblems comprising multiple scenarios. Then, obviously, the non-anticipativity links between scenarios included into one subproblem should be treated directly as constraints in the subproblem solver. The only non-anticipativity constraints that need to be coordinated by the DQA method are those that link scenarios assigned across subproblems.

To describe it formally, let $I_1 = \{1,\dots,i_1\}$, $I_2 = \{i_1+1,\dots,i_2\},\dots$, $I_K = \{i_{K-1}+1,\dots,L\}$ be subsets of scenarios assigned to subproblems $1,2,\dots,K$. Let us define the *outgoing boundary* of subproblem k by

$$\partial^+ I_k = \{(l,t) \in I_k \times \{1,\dots,T\} :\ \nu(l,t) \notin I_k\}.$$

In a similar way we define the *ingoing boundary*:

$$\partial^- I_k = \{(l,t) \in I_k \times \{1,\ldots,T\} : \ \nu^{-1}(l,t) \notin I_k\}.$$

Then we can formulate subproblem k as follows

$$\min \sum_{i \in I_k} \sum_{t=1}^{T} \langle \tilde{c}_l(t), x_l(t) \rangle + \frac{1}{2} r \sum_{(l,t) \in \partial^+ I_k} \|x_l(t)\|^2 + \frac{1}{2} r \sum_{(l,t) \in \partial^- I_k} \|x_l(t)\|^2 \qquad (4.15a)$$

subject to

$$x_l \in X_l, \quad i \in I_k, \qquad (4.15b)$$

$$x_l(t) = x_{\nu(l,t)}(t), \quad i \in I_k, \quad \nu(l,t) \in I_k, \qquad (4.15c)$$

with the modified costs defined by:

$$\tilde{c}_l(t) = \begin{cases} p_l c_l(t) + \pi_l(t) - r\tilde{x}_{\nu(l,t)}(t) & \text{if } (l,t) \in \partial^+ I_k, \\ p_l c_l(t) - \pi_{\nu^{-1}(l,t)}(t) - r\tilde{x}_{\nu^{-1}(l,t)}(t) & \text{if } (l,t) \in \partial^- I_k, \\ p_l c_l(t) & \text{otherwise.} \end{cases}$$

Since internal non-anticipativity constraints (4.15c) are treated directly, we only need to apply augmented Lagrangian terms to the links between boundaries of subproblems:

$$x_l(t) = x_{\nu(l,t)}(t), \quad (l,t) \in \partial^+ I_k, \quad k = 1,\ldots,K.$$

and only for these (l,t) we carry out iteration (4.6). So, similarly to the full decomposition scheme, exchange of information between subproblems occurs only along coordinated links $(l,t) \leftrightarrow (\nu(l,t),t)$ with $(l,t) \in \partial^+ I_k$, $k = 1,\ldots,K$.

In the example of Figure 2, assuming that subproblem 1 contains scenarios 1, 2, 3 and 4, subproblem 2 contains scenarios 5 and 6, subproblem 3 contains scenario 7 and subproblem 4 contains scenario 8, only links shown in Table 2 need be coordinated. The resulting communication structure of subproblems is shown in Figure 4. It could also be obtained by clustering subgroups of nodes of Figure 3.

Time	I_1				I_2		I_3	I_4
stage	1	2	3	4	5	6	7	8
1	-	-	-	5	-	7	8	1
2	-	-	-	-	-	7	8	5
3	-	-	-	-	-	-	8	7
4	-	-	-	-	-	-	-	-

Table 2: Coordinated links by partial decomposition.

The experiments reported in [21] show that in this way we can solve problems of remarkable sizes - with hundreds of thousands of variables and constraints by using standard workstations connected in a network. We assign as big subproblems as possible to the workstations; owing to the use of interior point methods they can be quite large. Messages about the approximation points $\tilde{x}_l$ and multiplier iterations are passed via the network. The stability of the multiplier method makes coordination of the subproblems possible, even in the presence of numerical errors.

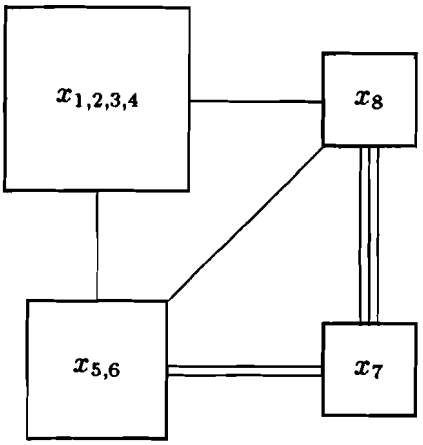

Figure 4: Communication between subproblems by partial decomposition.

5. Conclusions

Modeling uncertainty in linear programming problems by stochastic quantities gives rise to very large models with special structures. Their size can easily become so large that a straightforward application of general-purpose solvers can hardly be succesful. We have shown two ways of overcoming these difficulties within the primal-dual interior point methods.

One way is to exploit the problem structure within the linear algebra of the method. These are mainly special tricks for dealing with dense columns; in stochastic programming they lead to highly structured procedures with a potential of parallelization.

The second approach is the decomposition of the whole problem into smaller parts and coordination of them. Here, the availability of interior point methods that can solve quadratic programs as easily as linear ones is a breakthrough, because it allows for using stable and simple augmented Lagrangian techniques.

References

[1] D.P. Bertsekas,*Constrained Optimization and Lagrange Multiplier Methods* (Academic Press,1982).

[2] J.R. Birge and D. Holmes, "Efficient solution of two stage stochastic linear programs using interior point methods," technical report 92-8, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, 1992.

[3] J.R. Birge, R.M. Freund and R. Vanderbei, "Prior reduced fill-in in solving equations in interior point algorithms", *Operations Research Letters* 11(1992) 195-198.

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

[5] T.J Carpenter, *Practical Interior Point Methods for Quadratic Programming*, PhD Dissertation, Department of Civil Engineering and Operations Research, Princeton University, Princeton, 1992.

[6] I.C. Choi, C.L. Monma and D.F. Shanno, "Further developments of a primal-dual interior point method," *ORSA Journal on Computing* 2(1990) 304-311.

[7] I.I. Dikin, "Iterative solution of problems of linear and quadratic programming", *Soviet Mathematics Doklady* 8(1967) 674-675.

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

[9] A.V. Fiacco and G.P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques* (John Wiley and Sons, New York, 1967).

[10] J.J.H. Forrest and J.A. Tomlin, "Implementing interior point linear programming methods in the Optimization Subroutine Library", *IBM Systems Journal* 31(1992) 26-38.

[11] R. Fourer and S. Mehrotra, "Performance of an augmented system approach for solving least-squares problems in an interior point method for linear programming", *Mathematical Programming Society Newsletter* 19(1991) 26-31.

[12] P.E. Gill, W. Murray, M.A. Saunders, J.A. Tomlin and M.H. Wright, "On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method", *Mathematical Programming* 36(1986) 183-209.

[13] N.K. Karmarkar, "A new polynomial time algorithm for linear programming", *Combinatorica* 4(1984) 373-395.

[14] I.J. Lustig, R.E. Marsten and D.F. Shanno, "On implementing Mehrotra's predictor-corrector interior point method for linear programming", Report SOR-90-03, Princeton University, Department of Civil Engineering and Operations Research, Princeton, 1990 (to appear in *SIAM Journal on Optimization*).

[15] I.J. Lustig, R.E. Marsten and D.F. Shanno, "Computational experience with a primal-dual interior point method for linear programming," *Linear Algebra and its Applications* 152(1991) 191-222.

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

[17] R.D.C. Monteiro and I. Adler, "Interior path following primal dual algorithms. Part 1: Linear programming", *Mathematical Programming* 44(1989) 27-42.

[18] R.D.C. Monteiro and I. Adler, "Interior path following primal dual algorithms. Part 2: Convex quadratic programming", *Mathematical Programming* 44(1989) 43-66.

[19] S. Mehrotra, "Generalized predictor-corrector methods and their performance", technical report, Department of Industrial Engineering and Management Science, Northwestern University, Evanston 1992.

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

[21] J.M. Mulvey and A. Ruszczyński, "A new scenario decomposition method for large scale stochastic optimization," technical report SOR 91-19, Department of Civil Engineering and Operations Research, Princeton University, Princeton 1991.

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

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

[24] G. Stephanopoulos and W. Westerberg, "The use of Hestenes' method of multipliers to resolve dual gaps in engineering system optimization", *Journal of Optimization Theory and Applications*, 15(1975), pp. 285-309.

[25] R. J. Vanderbei, "ALPO: Another linear programming optimizer", technical report, AT&T Bell Laboratories, 1990.

[26] R. J. Vanderbei, "Symmetric quasi-definite matrices", technical report SOR 91-10, Department of Civil Engineering and Operations Research, Princeton University, Princeton 1991.

[27] R. J. Vanderbei and T.J. Carpenter, "Symmetric indefinite systems for interior point methods", technical report SOR 91-7, Department of Civil Engineering and Operations Research, Princeton University, Princeton 1991.