LARGE SCALE LINEAR PROGRAMMING TECHNIQUES IN STOCHASTIC PROGRAMMING

Roger J-B. Wets

November 1984
WP-84-90

CONTENTS

LARGE SCALE LINEAR PROGRAMMING TECHNIQUES IN STOCHASTIC PROGRAMMING

Roger J-B. Wets

INTRODUCTION

We study the use of large scale linear programming techniques for solving (linear) recourse problems† whose random elements have discrete distributions (with finite support) more precisely for problems of the type:

(0.1) find $x \in R_+^{n_1}$ such that $Ax = b$

and $z = cx + \mathcal{Q}(x)$ is minimized

where

(0.2) $\mathcal{Q}(x) = \sum_{l=1}^{L} p_l \, Q(x, \xi^l) = E\{Q(x, \xi(\omega))\}$

† The potential use of large scale programming techniques for solving stochastic programs with chance-constraints appears to be less promising and has not yet been investigated. The approximation scheme for chance-constraints proposed by Salinetti, 1983, would, if implemented require a detailed analysis of the structural properties of the resulting (large-scale) linear programs. Much of the analysis laid out in this Section would also be applicable to that case but it appears that further properties – namely the connections between the upper and lower bounding problems – should be exploited.

and for each $l=1,\dots,L$, the *recourse cost* $Q(x,\xi^l)$ is obtained by solving the *recourse problem*:

(0.3) $Q(x,\xi^l) = \inf \{q^l y \mid Wy = h^l - T^l x,\ y \in R_+^{n_2}\}$

where

$$\xi^l = (q^l, h^l, T^l) = (q_1^l,\dots,q_{n_2}^l; h_1^l,\dots,h_{m_2}^l; t_{11}^l,\dots,t_{1n_1}^l,\dots,t_{m_2 n_1}^l)$$

i.e.

$$\xi^l \in R^N \text{ with } N = n_2 + m_2 + m_2 \cdot n_1$$

and

$$p_l = \text{Prob}\,[\xi(\omega) = \xi^l] \quad .$$

The sizes of the matrices are consistent with $x \in R^{n_1}$, $y \in R^{n_2}$, $b \in R^{m_1}$ and for all l, $h_l \in R^{m_2}$; for a more detailed description of the recourse model consult Part I of this Volume. Because W is nonstochastic we refer to this problem as a model with *fixed recourse*. The ensuing development is aimed at dealing with problems that exhibit no further structural properties. Problems with simple recourse for example, i.e. when $W = (I,-I)$, are best dealt with in a nonlinear programming framework, cf. Chapter 4.

Before we embark on the description of solution strategies for the problem at hand, it is useful to review some of the ways in which a problem of this type might arise in practice. First, the problem is indeed a linear recourse model whose random elements follow a known discrete distribution function. In that case either q or h or T is random, usually not all three matrices at once, but the number of independent random variables is liable to be relatively large and even if each one takes on only a moderate number of possible values, the total number L of possible vectors ξ^l could be truly huge , for example a problem with 10 independent random variables each taking on 10 possible values leads us to consider 10 billion $(=L)$ 10-dimensional vectors ξ^l. Certainly not the type of data we want, or can, keep in fast access memory.

Second, the original problem is again a stochastic optimization problem of the recourse type but (0.1) is the result of an approximation scheme, either a discretization of an absolutely continuous probability measure or a coarser discretization of a problem whose "finite" number of possible realizations is too large to contemplate; for more about approximation schemes consult Chapter 2. In this case L, the number of possible values taken on by $\xi(\cdot)$, could be relatively small, say a few hundreds, in particular if (0.1) is part of a sequential approximation scheme, details can be found in Chapter 2, see also Birge and Wets, 1984, for example.

Third, the original problem is a stochastic optimization problem but we have only very limited statistical information about the distribution of the random elements, and $\xi^1, \ldots, \xi^L$ represents all the statistical data available. Problem (0.1) will be solved using the empirical distribution, the idea being of submitting its solution to statistical analysis such as suggested by the work of Dupačova and Wets, 1984. In this case L is usually quite small, we are thinking in terms of L less than 20 or 30.

Fourth, problem (0.1) resulted from an attempt at modeling uncertainty, with no accompanying statistical basis that allows for accurate descriptions of the phenomena by stochastic variables. As indicated in Chapter 1, this mostly comes from situations when there is data uncertainty about some parameters (of a deterministic problem) or we want to analyse decision making or policy setting and the future is modeled in terms of scenarios (projections with tolerances for errors). In this case the number L of possible variants of a key scenario that we want to consider is liable to be quite small, say 5 to 20, and the ξ^l can often be expressed as a sum:

$$\xi^l = \zeta^0 + \eta_{1l}\zeta^1 + \dots + \eta_{Kl}\zeta^k$$

where for $k = 1,\dots,K$, the $\zeta^k \in R^N$ are fixed vectors and $(\eta_1(\cdot),\dots,\eta_K(\cdot))$ are scalar random variables with possible values $\eta_{1l},\dots,\eta_{Kl}$ for $l = 1,\dots,L$. We think of K as being 2 or 3. The typical case being when we have a base projection: $\zeta^0 + \zeta^1$, but we want to consider the possibility that certain factors may vary by as much as 25% (plus or minus). In such a case the model assigns to the (only) random variable $\eta_1(\cdot)$ some discrete distribution on the interval [.75,1.25].

With this as background to our study it is natural to search solution procedures for recourse problems with discrete distributions when there is either only a moderate number of vectors ξ^l to consider (scenarios, limited statistical information, approximation) or there is a relatively large number of possible vectors ξ^l that result from combinations of the values taken on by independent random variables. The techniques discussed further on, apply to both classes of problems, but the tendency is to think of software development that would be appropriate for problems with relatively small L, say from 5 to 1,000. Not just because this class of problems appears more manageable but also because when L is actually very large, although finite, the overall solution strategy would still rely on the solution of approximate problems with relatively small L.

1. RECOURSE MODELS AS LARGE SCALE LINEAR PROGRAMS

Substituting in (0.1) the expressions for $\mathcal{Q}$ and Q, we see that we can obtain the solution by solving the linear program:

(1.1) find $x \in R_+^{n_1}$ and for $l = 1,\ldots,L$, $y^l \in R_+^{n_2}$ such that

$Ax \quad = b$,

$T^l x + Wy^l = h^l$, $\quad l = 1,\ldots,L$

and $z = cx + \sum_{l=1}^{L} p_l q^l y^l$ is minimized.

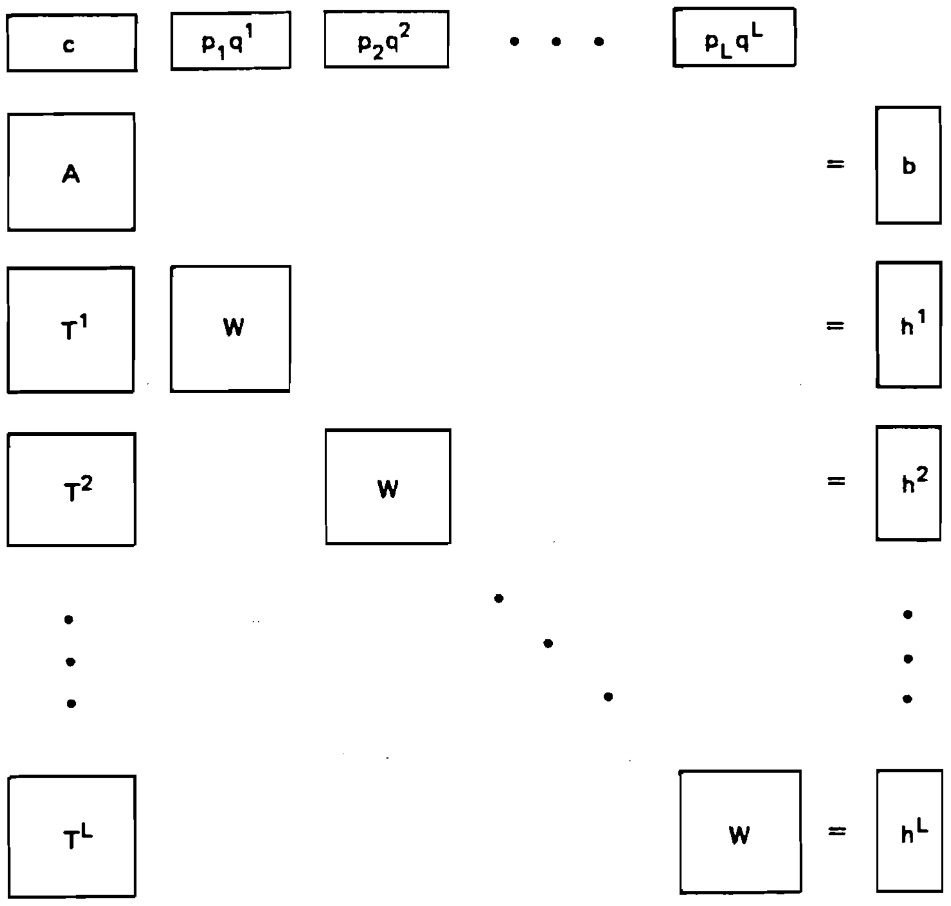
To each recourse decision to be chosen if $\xi(\cdot)$ takes on the value $\xi^l = (q^l, h^l, T^l)$ corresponds the vector of variables y^l. This is a linear program with

$m_1 + m_2 \cdot L$ constraints,

and

$n_1 + n_2 \cdot L$ variables.

The possibility of solving this problem using standard linear programming software depends very much on L, but even if it were possible to do so, in order to avoid making the solving of (1.1) prohibitively expensive -- in terms of time and required computer memory -- it is necessary to exploit the properties of this highly structured large scale linear program. The structure of the tableau of detached coefficients takes on the form:



1.2 FIGURE: Structure of discrete stochastic program.

We have here a so-called ***dual block angular structure*** with the important additional feature that all the matrices, except for A, along the block diagonal are the same. It is this feature that will lead us to the algorithms that are analysed in Section 3 and which up to now have provided us with the best computational results. It is also this feature which led Dantzig and Madansky, 1961, to suggest a solution procedure for (1.1) by way of the dual. Indeed, the following problem is a dual of (1.1):

(1.3) find $\sigma \in R^{m_1}$, and for $l = 1,\dots,L$, $\pi^l \in R^{m_2}$ such that

$$\sigma A + \sum_{l=1}^{L} p_l \pi^l T^l \leq c,$$

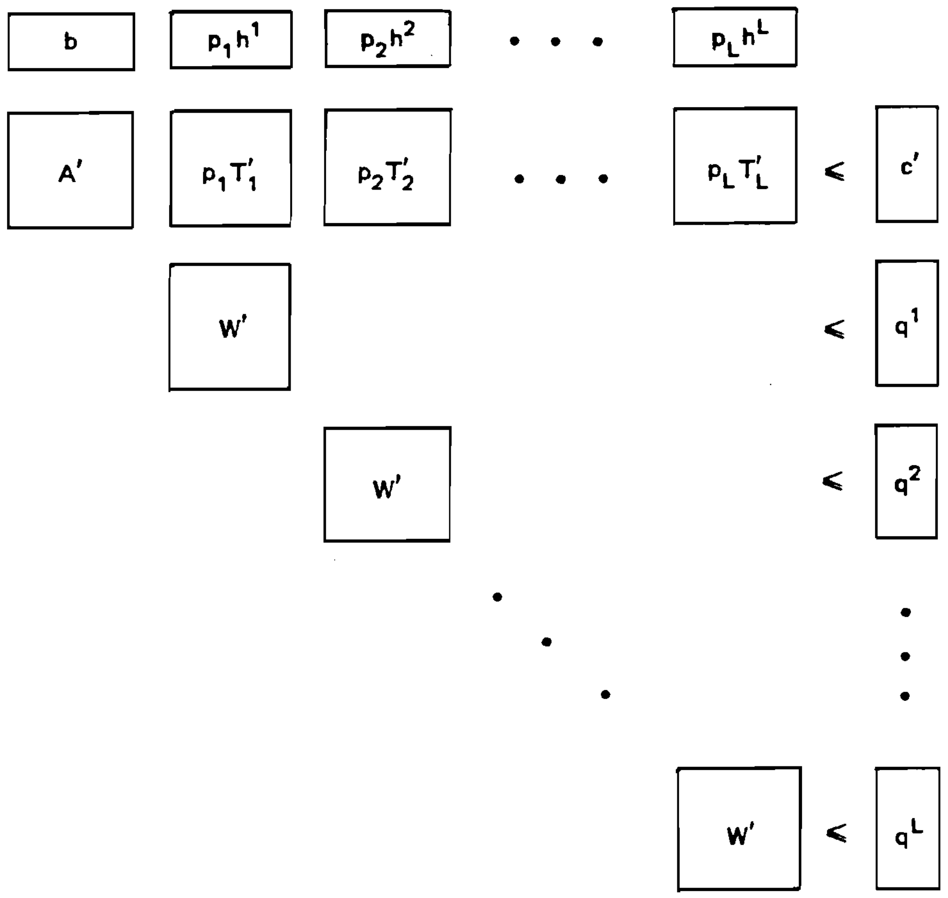
$$\pi^l W \leq q^l, \quad l = 1,\dots,L$$

and $w = \sigma b + \sum_{l=1}^{L} p_l \pi^l h^l$ is maximized.

Problem (1.3) is not quite the usual (formal) dual of (1.1) To obtain the classical linear program dual, set

$$\hat{\pi}^l = p_l \pi^l$$

and substitute in (1.3). This problem has ***block angular structure***, the block diagonal consisting again of identical matrices W. The tableau with detached coefficients takes on the form:



1.4 FIGURE. Structure of dual problem.

Transposition is denoted by $'$, e.g. W' is the transposed matrix of W. Observe that we have now fewer (unconstrained) variables but a larger number of constraints, assuming that $n_2 \geq m_2$, as is usual when the recourse problem (0.3) is given its canonical linear programming formulation. In Section 2 we review briefly the methods that rely on the structure of this dual problem for solving recourse models.

At least when the technology matrix T is nonstochastic, i.e. when $T^l = T$, a substitution of variables, mentioned in Wets, 1966, leads to a linear programming structure that has received a lot of attention in the literature devoted to large scale dynamical systems. Using the constraints of (1.1), it follows that for all $l = 1,\dots,L-1$,

$$Tx = h^l - Wy^l$$

and substituting in the $(l+1)$-th system, we obtain

$$-Wy^l + Wy^{l+1} = h^{l+1} - h^l.$$

Problem (1.1) is thus equivalent to

(1.5) find $x \in R_+^{n_1}$ and for $l = 1,\dots,L$, $y^l \in R_+^{n_2}$ such that

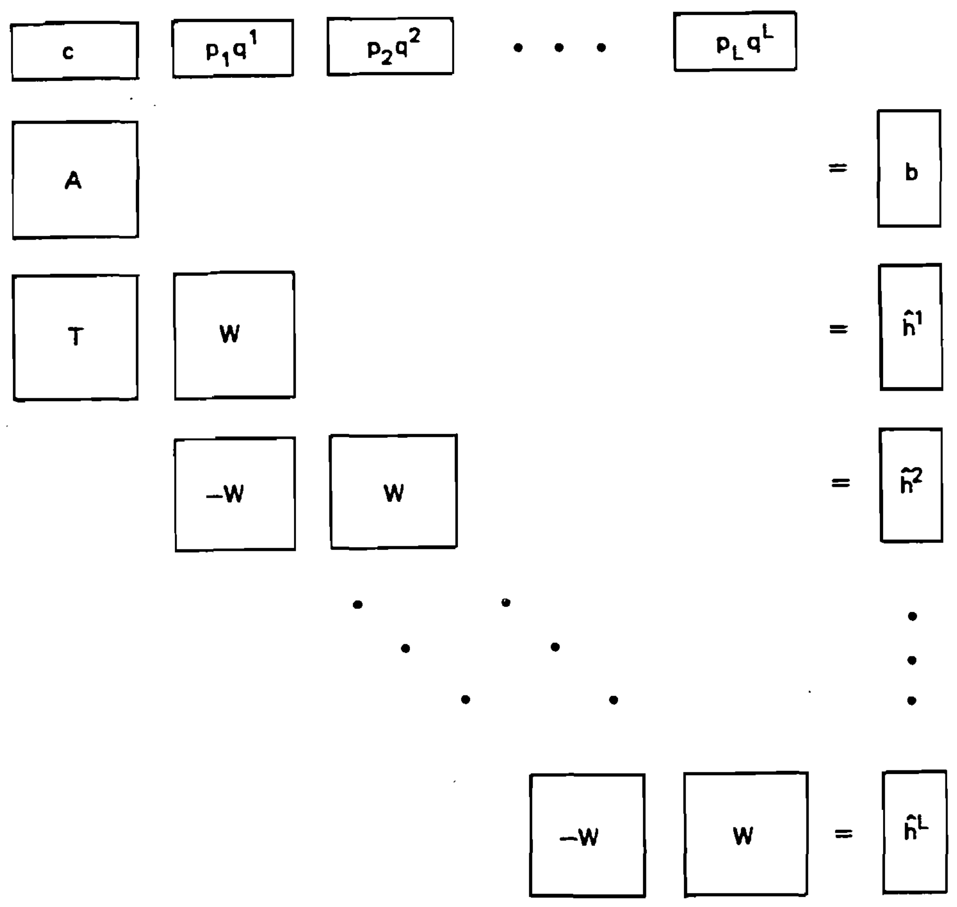
$$\begin{aligned}
&Ax &&= b\\
&Tx + Wy^1 &&= h^1\\
&\quad -Wy^{l-1} + Wy^l &&= h^l - h^{l-1},\ l = 2,\dots,L
\end{aligned}$$

and $z = cx + \sum_{l=1}^{L} p_l q^l y^l$ is minimized.

With $h^0 = 0$ and for $l = 1,\dots,L$,

$$\hat{h}^l = h^l - h^{l-1},$$

the tableau of detached coefficients exhibits a *staircase structure:*



1.5 FIGURE. Equivalent staircase structure.

We bring this to the fore in order to stress at the same time the close relationship and the basic difference between the problem at hand and those encountered in the context of dynamical systems, i.e. discrete version of continuous linear programs or linear control problems. Superficially, the problems are structurally similar, and indeed the matrix of a linear dynamical system may very well have precisely the structure of the matrix that appears in (1.5). Hence, one may conclude that the results and the computational work for staircase dynamical systems, cf. in particular Perold and Dantzig, 1979, Fourer, 1984, and

Saunders, 1983, is in some way transferable to the stochastic programming case. Clearly some of the ideas and artifices that have proved their usefulness in the setting of linear (discrete time) dynamical systems should be explored, adapted and tried in the stochastic programming context. But one should at all times remain aware of the fact that dynamical systems have coefficients (data) that are 1-parameter dependent (time) whereas we can view the coefficients of stochastic problems as being multi-parameter dependent. In some sense, *the gap between* (1.4) *and staircase structured linear programs that arise from dynamical systems is the same as that between ordinary differential equations and partial differential equations.* We are not dealing here with a phenomenon that goes forward (in time) but one which can spread all over R^N (which is only partially ordered)! Thus, it is not so surprising that from a computational viewpoint almost no effort has been made to exploit the structure (1.5) to solve stochastic programs with recourse. However, the potential is there and should not remain unexplored.

2. METHODS THAT EXPLOIT THE DUAL STRUCTURE

Dantzig and Madansky, 1961, pointed out that the dual problem (1.3) with matrix structure (1.4) is ripe for the application of the decomposition principle. It was also the properties of (1.4) that led Strazicky, 1980, to suggest and implement a basis factorization scheme, further analysed and modified by Kall, 1979, Wets, 1983, and Birge in Chapter 12. We give a brief description of both methods and study the connections between these two procedures. We begin with the second one, giving a modified compact version of the original proposal.

We assume that W is of full row rank, if not the recourse problem (0.3) defining Q would be infeasible for some of the values of h^l and T^l unless all belong to the appropriate subspace of R^N in which case a row transformation would allow us to delete the redundant constraints. We also assume that A is of full row rank, (possibly 0 when there are no constraints of that type). Thus with the columns of A' and W' linearly independent (recall that the variables σ and π are unrestricted), and after introducing the slack variables ($s^0 \in R_+^{n_1}$ and $s^l \in R_+^{n_2}$ for $l=1,\dots,L$), we see that each basic feasible solution will include at least n_2 variables of each subsystem

(2.1) $\pi^l W + s^l I = q^l, s^l \geq 0, \quad l = 1,\dots,L$,

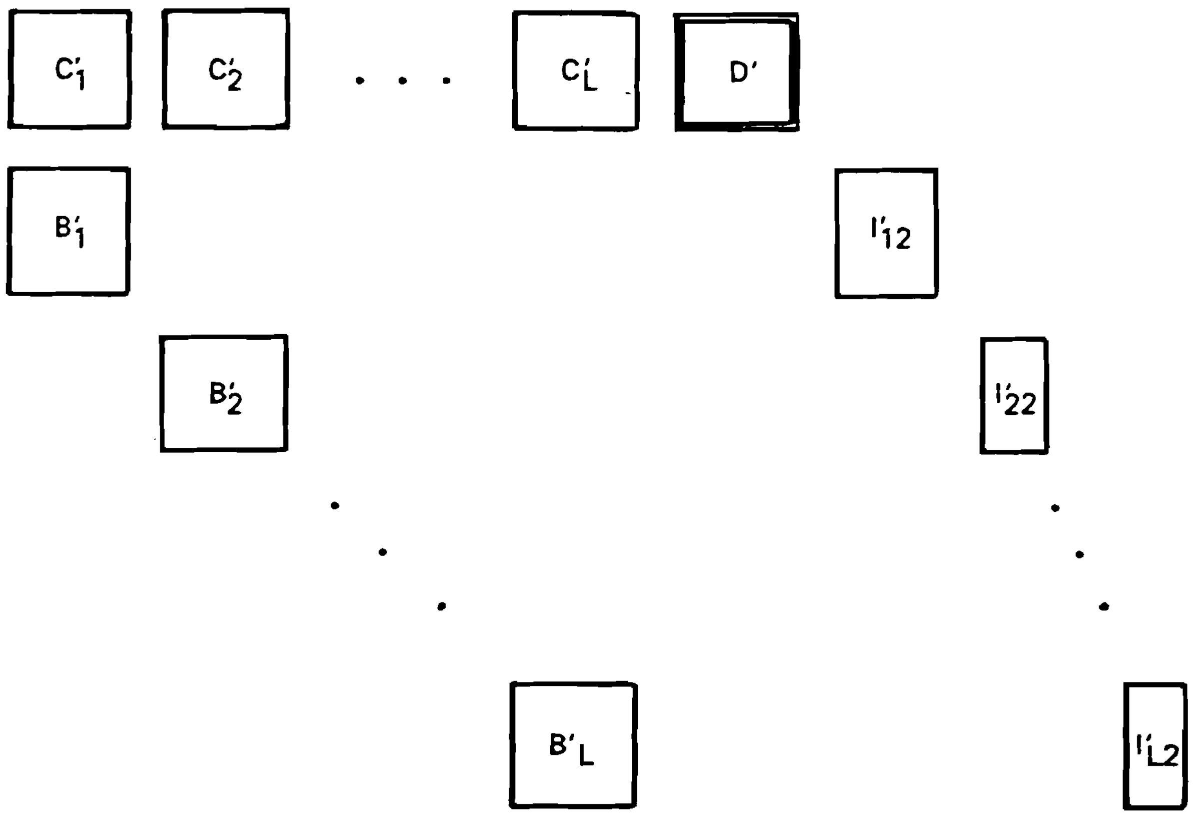
the (unrestricted) m_2 variables π^l and a choice of at least $(n_2 - m_2)$ slack variables (s_j^l, $j=1,\dots,n_2$). Thus the portion of the basic columns that appear in the l-th subsystem can be subdivided into two parts

$$[B_l', I_{l2}'] = [(W', I_{l1}'), I_{l2}']$$

where (W', I_{l1}') is an $(n_2 \times n_2)$ invertible matrix and the extra columns, if any, are relegated to I_{l2}. Thus, schematically and up to a rearrangement of columns, a feasible basis $\hat{B}$ has the structure:

$$\hat{B}' = \begin{bmatrix} C', & D' \\ B', & N' \end{bmatrix},$$

and in a detached coefficient form:



2.2 FIGURE. Basis structure of dual.

The matrix D' corresponding to the columns of (A', I'_{n_1}) that belong to this basis and for $l = 1, \ldots, L$, C_l' is the $n_1 \times m_2$ matrix:

$$C_l' = [p_l T_l', 0]$$

(recall that T'_l is of dimension $n_1 \times m_2$). Each B'_l, after possible rearrangement of row and columns, is of the following type:

$$B'_l = \begin{bmatrix} \boxed{W'_{(l)}} & \boxed{0} \\ \boxed{W'_{(cl)}} & \boxed{\begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix}} \end{bmatrix} = [W', I_{l1}]$$

2.3 FIGURE. Structure of B'_l.

where $W'_{(l)}$ is a $m_2 \times m_2$ invertible submatrix of W', and $W'_{(cl)}$ are the remaining rows of W' that correspond to the rows of the identity that have been included in B'^l (through I'_{l1}). The simplex multipliers associated with this basis $\hat{B}$, of dimension $n_1 + n_2 \cdot L$, are denoted by

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y^1 \\ \cdot \\ \cdot \\ \cdot \\ y^L \end{bmatrix}$$

and are given by the relations

$$\hat{B} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} C & B \\ D & N \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \gamma \\ \beta \end{bmatrix}$$

where $[\gamma', \beta']$ is the appropriate rearrangement of the subvector of coefficients of the objective of (1.4) that corresponds to the columns of $\hat{B}'$, with β' being the subvector of $[b', 0]$ whose components correspond to the columns of D'. This (dual feasible) basis is optimal if the vectors

$$(x, y^l, l = 1,\dots,L)$$

defined through (2.4) are primal feasible, i.e. satisfy the constraints of (1.1). To obtain x and y we see that (2.4) yields

$$y = B^{-1}(\gamma - Cx)$$

$$x = (D - NB^{-1}C)^{-1}(\beta - NB^{-1}\gamma) .$$

Substituting for x this becomes, for $l = 1,\dots,L$,

$$(2.5) \quad y_l = B_l^{-1}(\gamma^l - C_l x)$$

where γ^l is the subvector of $[p_l h^l, 0]$ that corresponds to the columns in B_l'. We have used the fact that $\hat{B}$ is a block diagonal with invertible matrices $(B_l', l = 1,\dots,L)$ on the diagonal. Going one step further and using the properties of N and C, we get the system for x:

$$(2.6) \quad (D - \textstyle\sum_{l=1}^{L} I_{l2} B_l^{-1} C_l)x = \beta - \textstyle\sum_{l=1}^{L} I_{l2} B_l^{-1}\gamma_l$$

The system (2.6) involves n_1 equations in n_1 variables and the L systems (2.5) are of order n_2. Thus instead of calculating the inverse of $\hat{B}$ -- a square matrix of order $(n_1 + n_2 \cdot L)$ -- all that is needed is the inverse of L matrices of order n_2 and a square matrix of order n_1.

Similarly to calculate the values to assign to the basic variables associated to this basis, the same inverses is all that is really required, as can easily be verified. In order to implement this method one would need to work out the updating procedures to show that the simplex method can be performed in this compact form, i.e. that the updating procedures involve only the restricted inverses. But there are other features of which one should take advantage before one proceeds with implementation.

Recall that

(2.7) $$B_l = \begin{bmatrix} W_{(l)} & W_{(cl)} \\ I & 0 \end{bmatrix}$$

where B_l is an invertible matrix of size $m_2 \times m_2$. Then

(2.8) $$B_l^{-1} = \begin{bmatrix} W_{(l)}^{-1}, & -W_{(l)}^{-1} W_{(cl)} \\ 0 & , I \end{bmatrix}$$

Thus it really suffices to know the inverse of $W_{(l)}$, and rather than keeping and updating the $n_2 \times n_2$ – matrix B_l^{-1}, all the information that is really needed can be handled by updating an $m_2 \times m_2$– matrix, relying on sparse updates whenever possible. This should result in substantial savings. The algorithm could even be more efficient by taking advantage of the repetition of similar (sub)bases $W_{(l)}$. We shall not pursue this any further at this time because all of these computational shortcuts are best handled in the framework of methods based on the decomposition principle that we describe next.

The decomposition principle, as used to solve the linear program (1.3), generates the master problem from the equations

$$\sigma A + \sum_{l=1}^{L} \pi^l (p_l T^l) \leq c \, ,$$

by generating extreme points or directions of recession (directions of unboundedness) from the polyhedral regions determined by the L subproblems,

$$\pi^l W \leq q^l \quad .$$

In order to simplify the comparison with the factorization method described earlier, let us assume that

$$\{\pi \mid \pi W \leq 0\} = \{0\} \quad ,$$

i.e. there are no directions of recession other than 0, which means that for all l, the polyhedra $\{\pi^l W \leq q^l\}$ are bounded; feasibility of (1.3) implying that they are nonempty. For $k=1,\dots,\nu$, let

$$\eta^k = (\eta^{1k},\dots,\eta^{lk},\dots,\eta^{Lk})$$

the extreme point generated by the k-th iteration of the decomposition method, i.e.

(2.9) $\eta^{lk} \in \operatorname{argmin}(p_l \pi^l (h^l - T^l x^k) \mid \pi^l W \le q^l)$

where $x^k = (x_j^k, j=1,\dots,n_1)$ are the multipliers associated to the first n_1 linear inequalities of the master problem :

(2.10) find $\sigma \in R^{m_1}$, $\lambda_k \in R_+$, $k = 1,\dots,\nu$ such that

$$\sigma A + \sum_{k=1}^{\nu} \lambda_k \Big(\sum_{l=1}^{L} p_l \eta^{lk} T^l\Big) \le c$$

$$\sum_{k=1}^{\nu} \lambda_k = 1$$

and $w = \sigma b + \sum_{k=1}^{\nu} \lambda_k \big(\sum_{l=1}^{L} p_l \eta^{lk} h^l\big)$ is maximized.

The basis associated to the master problem is $(n_1 \times n_1)$, whereas the basis for each subproblem is exactly of order n_2. In the process of solving the subproblems the iterations of the simplex method bring us from one basis of type (2.7) to another one of this type (all transposed, naturally) with inverses given by (2.8). Here again, the implementation should take advantage of this structural property, and updates should be in terms of the $m_2 \times m_2$ submatrices $W_{(l)}$. But we should also take advantage of the fact that all these subproblems are identical except for the right-hand sides and/or the cost coefficients, and this, in turn, would lead us to the use of bunching and sifting procedures of Section 4.

It is remarkable and important to observe that the basis factorization method *with the modifications* alluded to earlier and the decomposition method applied to the dual, as proposed by Dantzig and Madansky, 1961, require the same computational effort; J. Birge gives a detailed analysis in Chapter 12, independently B. Strazicky arrived at similar results. In view of all of this it is appropriate to view the method relying on basis factorization as a very close parent of the decomposition method as applied to the dual problem (1.3), but it does not give us the organizational flexibility provided by this latter algorithm. On conceptual ground, as well as in terms of computational efficiency, it is the decomposition based algorithm that should be retained for potential software implementation. In fact, this is essentially what has occurred, but it is a "primal" version of this decomposition algorithm, which in this class of (essentially) equivalent methods appears best suited for solving linear stochastic programs with recourse. It is a primal method -- which means that we always have a feasible $x \in R_+^{n_1}$ at our disposal -- and it allows us to take advantage in the most straightforward manner of some of the properties of recourse models to speed up computations.

3. METHODS THAT ARE PRIMAL ORIENTED

The great difference between the methods that we consider next and those of Section 2 is that finding x that solves the stochastic program (0.1) is now viewed as our major, if not exclusive, concern. Obtaining the corresponding recourse decisions $(y^l, l=1,\ldots,L)$ or associated dual multipliers $(\pi^l, l=1,\ldots,L)$ is of no real interest, and we only perform some of these calculations because the search for an optimal solution x requires knowing some of these quantities, at least in an amalgamated form. On the other hand, in the methods of Section 2 all the variables $(\sigma,\pi^1,\ldots,\pi^L)$ are treated as equals; to have the optimality criterion fail for some variable in subsystem l (even when p_l is relatively small) is handled with the same concern as having the optimality criteria fail for some of the $(\sigma_i, i=1,\ldots,m_1)$ variables.

Another important property of these methods is their natural extension to stochastic programs with arbitrary distribution functions. In fact, they are particularly well-suited for use in a sequential scheme for solving stochastic programs by successive refinement of the discretization of the probability measure, each step involving the solution of a problem of type (0.1), cf. Chapter 2.

We stress these conceptual differences, because they may lead to different, more flexible, solution strategies; although we are very much aware of the fact that if at each stage of the algorithm all operations are carried out (to optimality), it is possible to find their exact counterpart in the algorithms described in Section 2; for the relationship between the L-shaped algorithm described here and the decomposition method applied to the dual, see Van Slyke and Wets, 1969; between the above and the basis factorization method see Chapter 12; consult also Ho, 1983, for the relationship between various schemes for piecewise linear functions which are widely utilized for solving certain classes of stochastic programming problems, and Chapter 4.

The *L-shaped algorithm*, which takes its name from the matrix layout of the problem to be solved, was proposed by Van Slyke and Wets, 1969. It can be viewed as a cutting hyperplane algorithm (outer linearization) but to stay in the framework of our earlier development, it is best to interpret it here as a *partial* decomposition method. We begin with a description of a very crude version of the algorithm, only later do we elaborate the modifications that are *vital* to make the method really efficient. To describe the method it is useful to consider the problem in its original form (0.1) which we repeat here for easy reference:

(3.1) find $x \in R_+^{n_1}$ such that $Ax = b$,

and $z = cx +$ (x) is minimized

We assume that the problem is feasible and bounded, implementation of the algorithm would require an appropriate coding of the initialization step relying on the criteria for feasibility and boundedness such as found in Wets, 1972. The method consists of three steps that can be interpreted as follows. In Step 1, we solve an approximate of (3.1) obtained by replacing by an outer-linearization, this brings us to the solving of a linear programming whose constraints are $Ax = b$, $x \geq 0$ and the additional constraints (3.2) and (3.3) that come from:

(i) induced feasibility cuts generated by the fact that the choice of x must be restricted to those for which $\mathcal{Q}(x)$ is finite, or equivalently for which $Q(x, \xi^l) < +\infty$ for all $l = 1, \ldots, L$, or still for which there exists $y^l \in R_+^{n_2}$ such that $Wy^l = h^l - T^l x$ for all $l = 1, \ldots, L$.

(ii) linear approximations to on its domain of finiteness.

These constraints are generated systematically through Steps 2 and 3 of the algorithm, when a proposed solution x^ν of the linear program in Step 1 fails to satisfy the induced constraints, i.e. $(x^\nu) = \infty$ (Step 2) or if the approximating problem does not yet match the function at x^ν (Step 3). The row-vector generated in Step 3 is actually a subgradient of at x^ν. The convergence of the algorithm under the appropriate nondegeneracy assumptions, to an optimal solution of (3.1), is based on the fact that there are only a finite number of constraints of type (3.2) and (3.3) that can be generated by Steps 2 and 3 since each one corresponds to some basis of W and a pair (h^l, T^l) or to a basis of W and to one of a finite number of weighted averages of the $(q^l, l=1, \ldots, L)$ and

$((h^l, T^l), l = 1,\ldots L)$.

Step 0. Set $\nu = r = s = 0$.

Step 1. Set $\nu = \nu + 1$ and solve the linear program

find $x \in R_+^{n_1}$, $\vartheta \in R$ such that

$$
\begin{array}{llll}
 & Ax & = b & \\
(3.2) & D_k x & \geq d_k, & k = 1,\ldots,r, \\
(3.3) & E_k x + \vartheta & \geq e_k, & k = 1,\ldots,s, \text{ and} \\
 & cx \quad + \vartheta & = z & \text{is minimized.}
\end{array}
$$

Let (x^ν, ϑ^ν) be an optimal solution. If there are no constraints of type (3.3), the variable ϑ is ignored in the computation of the optimal x^ν, the value of ϑ^ν is then fixed at $-\infty$.

Step 2. For $l = 1,\ldots,L$ solve the linear programs

(3.4) find $y \in R_+^{n_2}$, $v^+ \in R_+^{m_2}$, $v^- \in R_+^{m_2}$ such that

$$Wy + Iv^+ - Iv^- = h^l - T^l x^\nu \text{ and}$$

$$ev^+ + ev^- = v^l \text{ is minimized}$$

(here e denotes the row vector $(1,1,\ldots,1)$), until for some l the optimal value $v^l > 0$. Let σ^ν be the associated simplex multipliers and define

$$D_{r+1} = \sigma^\nu T^l$$

and

$$d_{r+1} = \sigma^\nu h^l$$

to generate an induced feasibility cut. Return to Step 1 adding this new constraint of type (3.2) and set $r = r + 1$. If for all l, the optimal value of the linear program (3.4) $v^l = 0$, go to *Step* 3.

Step 3. For every $l = 1,\ldots,L$, solve the linear program

(3.5) find $y \in R_+^{n_2}$ such that

$$Wy = h^l - T^l x^\nu, \text{ and}$$

$$q^l y = w^l \text{ is minimized.}$$

Let $\pi^{l\nu}$ be the multipliers associated with the optimal solution of problem l. Set $t = t + 1$ and define

$$E_t = \sum_{l=1}^{L} p_l \pi^{l\nu} T^l,$$

$$e_t = \sum_{l=1}^{L} p_l \pi^{l\nu} h^l,$$

and

$$w^\nu = \sum_{l=1}^{L} p_l \pi^{l\nu} (h^l - T^l x^\nu) = e_t - E_t x^\nu \quad .$$

If $\vartheta^\nu \geq w^\nu$, we stop; x^ν is the optimal solution. Otherwise, we return to *Step* 1 with a new constraint of type (3.3).

An efficient implementation of this algorithm, whose steps can be identified with those of the decomposition method applied to the dual problem (see Section 2), depends very much on the acceleration of Steps 2 and 3. This is made possible by relying on the specific properties of the problem at hand (3.1), and it is in order to exploit these properties that we have separated Steps 2 and 3 which are the counterparts of Phase I and Phase II of the simplex method as applied to the recourse problem (0.3). In practice one certainly does not start from scratch when solving the L linear programs in Step 3; Section 4 is devoted to the analysis of Step 3, i.e. how to take advantage of the fact that the L linear programs that need to be solved have the same technology matrix W as well as from the fact that the $\xi^l = (q^l, h^l T^l)$ are the realizations of a random vector. Here we concern ourselves with the improvements that could be made to speed up Step 2, and we see that in many instances, dramatic gains could be realized.

First and for all, Step 2 can be skipped altogether if the stochastic program is with *complete recourse*, i.e. when

(3.6) $\text{pos} W := \{t \mid t = Wy, y \geq 0\} = R^{m_2}$,

a quite common occurrence in practice. This means naturally that no induced feasibility constraints (3.2) need to be generated. This will also be the case if we have a problem with ***relatively complete recourse*** i.e. when for every x satisfying $Ax = b, x \geq 0$, and for every $l = 1,\ldots,L$, the linear system

$$Wy = h^l - T^l x, y \geq 0 ,$$

is feasible. This weaker condition is much more difficult to recognize, and to verify it would precisely require the procedure given in Step 2.

Even in the general case, it may be possible to substitute for Step 2: for some (h^ν, T^ν)

Step 2'. Solve the linear program

(3.7) find $y \in R_+^{n_2}, v^+ \in R_+^{m_2}, v^- \in R_+^{m_2}$ such that

$$Wy + Iv^+ - Iv^- = (h^\nu - T^\nu x^\nu)$$

and $ev^+ + ev^- = v^\nu$ is minimized.

Let σ^ν be the associated simplex multipliers and if the optimal value of $v^\nu > 0$, define

$$D_{r+1} = \sigma^\nu T^\nu ,$$

and

$$d_{r+1} = \sigma^\nu h^\nu$$

to generate an induced feasibility cut of type (3.2). Return to *Step* 1 with $r = r+1$. If the optimal value of $v^{\nu} = 0$, go to Step 3.

This means that we have replaced solving L linear programs by just solving 1 of them. In some other cases it may be necessary to solve a few problems of type (3.7) but the effort would in no way be commensurate with that of solving all L linear programs of Step 2. In Section 5 of Wets, 1974, one can find a detailed analysis of the cases when such a substitution is possible, as well as some procedures for the choice or construction of the quantities h^{ν} and T^{ν} that appear in the formulation of (3.7). Here we simply suggest the reasons why this simplification is possible and pay particular attention to the case when the matrix T is nonstochastic.

Let $<$ be the partial ordering induced by the closed convex polyhedral cone pos W, see (3.6), i.e. $a^1 < a^2$ if $a^2 - a^1 \in \text{pos} W$. Then for given $x \in R^{n_1}$ and for every $l = 1,\ldots,L$, the linear system

(3.8) $Wy = h^l - T^l x^{\nu}, \quad y \geq 0$

is feasible, if there exists $a^{\nu} \in R^{m_2}$ such that for all $l = 1,\ldots,L$,

(3.9) $a^{\nu} < h^l - T^l x^{\nu}$,

and the linear system

(3.10) $Wy = a^{\nu}, y \geq 0$

is feasible -- or equivalently $a^{\nu} \in \text{pos} W$. There always exists a^{ν} that satisfies (3.9), recall L is finite. If in addition, a^{ν} can be chosen so that

(3.11) $a^{\nu} = h^{\nu} - T^{\nu} x$

for $\nu \in \{1,\ldots,L\}$, then (3.8) is feasible for all l *if and only if* (3.10) is feasible with a^{ν} as defined by (3.11). Although in general such an a^{ν} does not exist, in practice, at most a few extreme points of the set

$$S^\nu = \{\alpha \mid \alpha = h^l - T^l x^\nu,\ l=1,\dots,L\}\ ,$$

need to be considered in order to verify the feasibility of *all* the linear systems (3.8). Computing lower bounds of S^ν with respect to $<$ may require more work than we bargained for, but it really suffices, cf. Theorem 4.17 of Wets, 1974, to construct lower bounds of S^ν with respect to any closed cone contained in pos W, and this could be, and usually is taken to be, an orthant. In such a case obtaining α^ν is effortless.

Let us consider the case when T is nonstochastic and assume that pos W contains the positive orthant, if it contains another orthant simply multiply some rows by -1 making the corresponding adjustments in the vectors $(h^l,\ l = 1,l,\dots,L)$. This certainly would be the case if slack variables are part of the y-vector, for example.

For $i = 1,\dots,m_2$, let

$$\alpha_i = \min_l\ h_i^l$$

If $\alpha = h^\nu$ for some $\nu \in \{1,\dots,L\}$, which would always be the case if the $(h_i(\cdot),\ i = 1,\dots,m_2)$ are independent random variables, then it follows from the above that for $l=1,\dots,L$, the linear systems

$$Wy = h^l - Tx^\nu,\ \ y \geq 0$$

are feasible if and only if

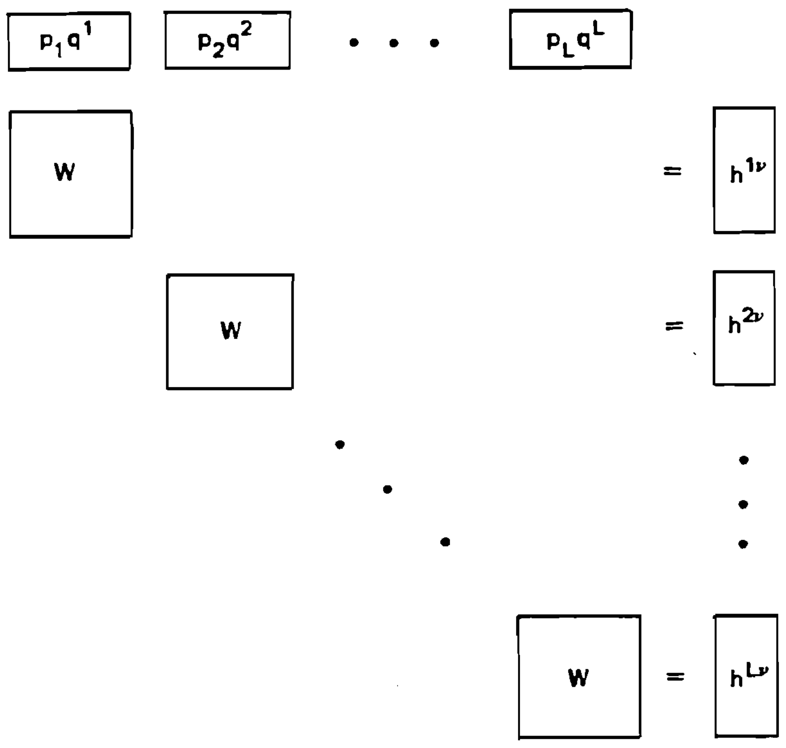
$$Wy = \alpha - Tx^\nu,\ \ y \geq 0.$$

Note that in this case the lower bound

$$\alpha^\nu = \alpha - Tx^\nu$$

is a simple function of x^ν.

In our description of the L-shaped algorithm the connections to large scale linear programming may have been somewhat lost, if anything it is how to deal with the "nonlinearity" of $\mathcal{Q}$ which has played center stage. To regain maybe a more *linear* programming perspective it may be useful to view the algorithm in the following light. Let us return to the dual block angular structure (1.2) from which it is obvious that if we can adjust the simplex method so that it operates separately on the x-variables and the (y_l-variables, $l=1,\dots,L$), it will be possible to take advantage of the block diagonal structure of the problem with respect to the (y^l-variables, $l=1,\dots,L$). Given that some x^ν is known which satisfy the constraint $x \geq 0$, $Ax = b$, then finding the optimal solution of (1.2), with the additional constraint $x = x^\nu$ leads to solving a linear program, whose tableau of detached coefficients has the structure:



3.12 FIGURE. Structure of the y-problem.

where for $l=1,\ldots,L$, $h^{l\nu} = h^l - T^l x^\nu$. Clearly, when confronted with such a problem we want to take advantage of its separability properties and this is precisely what is done in Steps 2 and 3 of the L-shaped algorithm.

The structure of (3.12), with the same matrix W on the block diagonal, suggests that of a distributed system. A continuous version would take the form:

(3.13) find $y : \Omega \to R^{n_2}$ such that for all $w \in \Omega$

$$y(w) \in \operatorname{argmin} [q(w)y \mid Wy = h^\nu(w), y \in R_+^{n_2}] \quad .$$

Because of the linearity of the objective function, the trajectory $w \mapsto y(w)$ will be linear with respect to h^ν if the same basis of W remains optimal. The main task in solving (3.13) would be to decompose Ω in regions of linearity of $y(\cdot)$. Once this decomposition is known the remainder is rather straightforward. Finding this decomposition is essentially the subject of Section 4, which concerns itself with the organization of the computational work so as to bring the effort involved to an acceptable level. Problem (3.13) again brings to the fore the connections between this work and that on dynamical systems (continuous linear programming). With not too much difficulty it should be possible to formulate a bang-bang principle for systems with distributed parameters space (here R^{m_2}) that would correspond to our scheme for decomposing Ω.

To conclude our discussion of the L-shaped algorithm, let us record a further modification suggested by L. Nazareth. When the matrix T is nonstochastic, say $T^l = T$ for all l, then the linear program in Step 1 may be reformulated as

(3.14) find $x \in R_+^{n_1}$, $\chi \in R^{m_2}$, $\vartheta \in R$ such that

$$\begin{array}{llll}
Ax & & = b & \\
Tx & -\chi & = 0 & \\
 & F_k\chi & \geq f_k, & k = 1,\dots,r \\
 & G_k\chi + \vartheta & \geq g_k, & k = 1,\dots,s, \text{ and} \\
cx & \quad + \vartheta & = z & \text{is minimized}
\end{array}$$

The induced feasibility constraints are generated as earlier in Step 2 with

$$F_{r+1} = -\sigma^\nu \;,\; f_{r+1} = \sigma^\nu h^l$$

The optimality cuts (approximation cuts) are generated in Step 3 with

$$G_t = -\textstyle\sum_{l=1}^L p_l \pi^{l\nu} \;,$$
$$g_t = \textstyle\sum_{l=1}^L p_l \pi^{l\nu} h^l \quad .$$

The linear program that generates the σ^ν and $\pi^{l\nu}$ as (optimal) simplex multipliers of Phases I and II respectively, is given by

find $y \in R_+^{n_2}$ such that

$Wy = h^l - \chi^\nu$, and

$qy = w^l$ is minimized.

Note that now the "nonlinearity" is handled in a space of dimension m_2 which is liable to be much smaller than n_1, and we should reap all the advantages that usually come from a reduction in the number of non-linear variables.

All of these simplifications come from the fact that when T is non-stochastic we can interpret the search for an optimal solution, as the search for an optimal χ^*, "the certainty equivalent". It is easy to see that knowing χ^* would allow us to solve the original problem by simply solving

(3.15) find $x \in R_+^n$ such that $Ax = b$, $Tx = \chi^*$,

and $z = cx$ is minimized .

The sequence $\{\chi^\nu, \nu = 1,\ldots\}$ generated by the preceding algorithm can be viewed as a sequence of tenders (to be "bet" against the uncertainty represented by h). This then suggests other methods based on finding χ^* by considering the best possible convex combination of the tenders generated so far; these algorithms based on generalized linear programming, see Nazareth and Wets, 1984, and Chapter 4 of this Volume. However, this approach does not appear to be very promising for the general class of problems considered here, not even when T is nonstochastic. Indeed, the algorithm would proceed as follows:

Step 0. Find a feasible $x^0 \in R_+^{n_1}$ such that $Ax^0 = b$

Set $\chi^0 = x^0$

Choose $\chi^1, \ldots, \chi^\nu$, potential tenders, $\nu \geq 0$.

Step 1. Find $(\sigma^\nu, \pi^\nu, \vartheta_\nu)$ the (optimal) simplex multipliers associate with the solution of the linear program:

$$
\begin{array}{lll}
\text{minimize } cx + \sum_{l=0}^{\nu} \lambda_l \, Q(\chi^l) & & \\
\quad Ax & = b & :\sigma^\nu \\
\quad Ix - \sum_{l=0}^{\nu} \lambda_l \chi^l & = 0 & :\pi^\nu \\
\quad \sum_{l=0}^{\nu} \lambda_l & = 1 & :\vartheta_\nu \\
\quad x \geq 0, \lambda_l \geq 0 & &
\end{array}
$$

Step 2. Find $\chi^{\nu+1} \in \mathrm{argmin}[\,\mathcal{Q}(\chi) + \pi^{\nu}\chi]$

If $\mathcal{Q}(\chi^{\nu+1}) + \pi^{\nu}\chi^{\nu+1} \geq \vartheta_{\nu}$, stop: optimal.

Otherwise return to *Step* 1 with $\nu = \nu + 1$

The attractiveness of this approach rests on the fact that the algorithm allows for the choice of a number of tenders (trial solutions) which would provide an excellent initial approximate solution to the problem as a whole just after 1 passage through Step 1, assuming of course that the tenders $\chi^1,\ldots,\chi^{\nu}$ are chosen by an informed problem solver. Note, however, that for each tender $\chi \in R^{n_1}$ we need to find the value of $\mathcal{Q}(\chi) = \sum_{l=1}^{L} p_l\, Q(\chi,\xi^l)$, i.e. solve the L linear programs

find $y \in R_+^{n_2}$ such that $Wy = h^l - T^l\chi$,

and $w^l = q^l y$ is minimized.

Of course in order to do so, we can take advantage of the techniques described in the next section.

However, our enthusiasm for this approach must be tempered by the sobering realization that performing Step 2 requires essentially the same amount of work as solving the original problem (0.1). Indeed to find $\chi^{\nu+1}$ we need to solve the linear program:

(3.16) find $\chi \in R^{n_1}$, $y^l \in R_+^{n_2}$, $l = 1,\ldots,L$, such that

$T^l\chi + Wy^l = h^l$, and

$\pi^{\nu}\chi + \sum_{l=1}^{L} p_l\, q^l y^l$ is minimized.

The only differences with the structure of the original problem is that there are no constraints $Ax = b$ and χ must not necessarily be nonnegative. There may be some advantages here, but certainly not enough to warrant solving the stochastic program (0.1) by solving a sequence of programs of the type (3.16), unless we were dealing with stochastic programs that exhibit further structural properties, such as stochastic programs with simple recourse, see the end of Section 6 of Nazareth and Wets, 1984, and in particular the implementation described by Nazareth in Chapter 13.

4. SIFTING, BUNCHING AND BASES UPDATES.

In the final analysis, Step 3 of the L-shaped algorithm boils down to the calculation of the value of Q and of its gradient at x^ν. What it involves is solving a large number of similar linear programs, or if you prefer one linear program with matrix structure, Figure 3.12. The same type of operations would be required for the actual carrying out of *Step* 2 of the algorithm based on the generation of tenders. The extent to which we are able to speed up these computations will determine the level of "stochasticity" that we are able to handle. This Section raises the question of how to organize the work so as to minimize the computational effort involved. We consider only the case of multiple right-hand sides, resulting, as the case may be, from h and/or T random; by duality, the analysis also applies to the case when only q is random (and h and T are nonstochastic). When both the cost coefficients and the right-hand sides of the recourse problem (0.3) include random variables a further refinement of the methods suggested here would be required. We shall not be concerned with special cases such as simple recourse $W = (I, -I)$, or network-structured problems when specific computational shortcuts are possible, e.g. Midler and Wollmer, 1969, Wallace, 1984, and Qi, 1984.

In its simplest form, the problem that we are concerned with is finding an efficient procedure for solving L linear programs with variable right-hand sides: For $l = 1,\dots,L$, (4.1) find $y \in R_+^{n_2}$ such that

$Wy = t^l$,

$qy = w^l$ is minimized.

The vectors $\mathcal{T} = \{t^l \ , \ l=1,\ldots,L\}$ come from $t^l = h^l - T^l x^\nu$ or $t^l = h^l - \chi^\nu$, cf. Section 3.

For all l, (4.1) is feasible, i.e.

(4.2) $t^l \in \text{pos } W = \{t \mid t = Wy, \ y \geq 0\}$,

(this comes from the fact that x^ν or χ^ν satisfies the induced feasibility constraints). Moreover, by assumption we have that (4.1) is bounded, and hence for all l, (4.1) is solvable. We shall denote the optimal solution by y^l, and the associated simplex multipliers by π^l. We have that

$\pi^l W \leq q$,

and

$qy^l = \pi^l t^l.$

The methods that we study can be divided into *sifting* (discrete parametric analysis) and *bunching* (basis by basis analysis) procedures. We begin with a description of a very crude bunching procedure, which nonetheless would be much more efficient than solving separately all L linear programs (4.1). This technique is easily modified to also take care of the case of multiple cost coefficients vector, cf. Wets 1983, p.587.

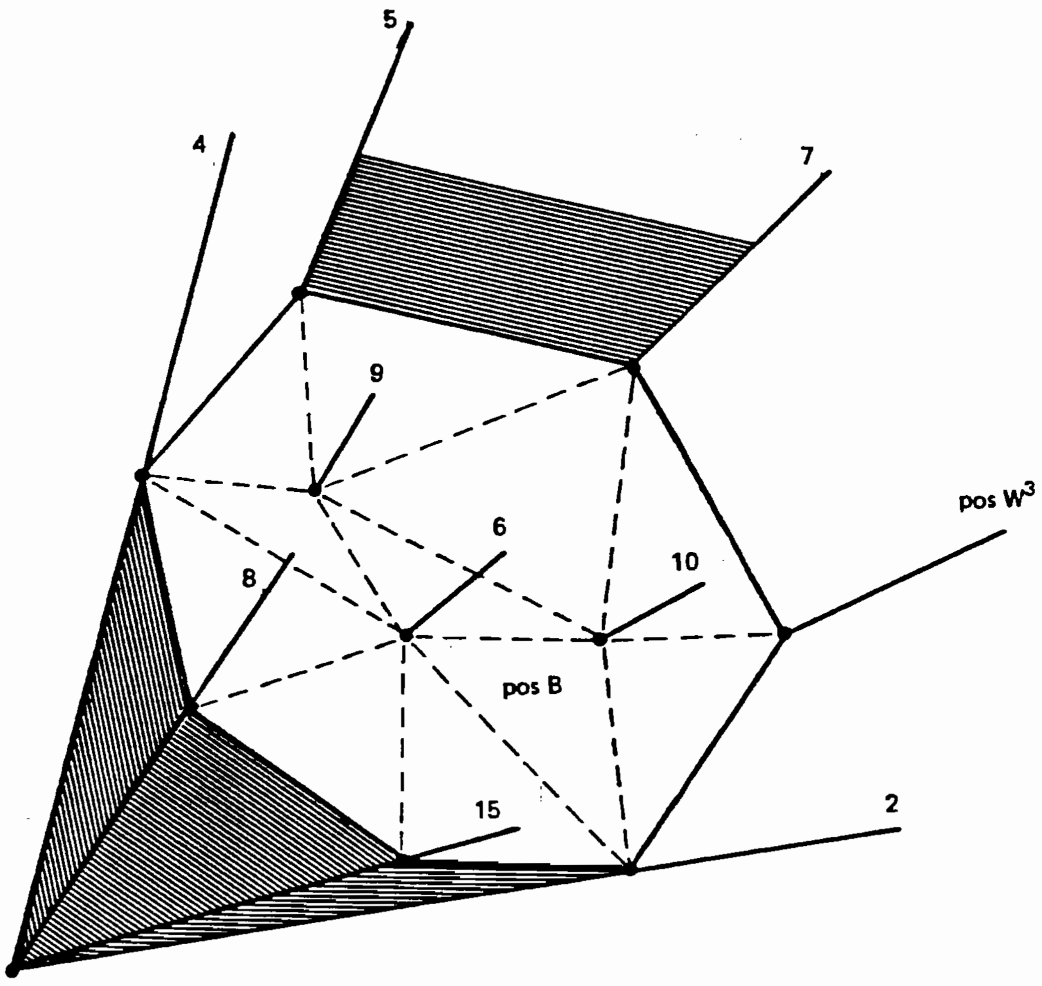
Let B be an $m_2 \times m_2$ invertible matrix of W with $\gamma B^{-1} W \leq q$ where γ is the subvector of q that corresponds to the columns of W in B; recall that W is assumed to be of full row rank. Then from the optimality conditions for linear programming, it follows that this basis B is optimal for any vector $t \in R^{m_2}$ such that

(4.3) $B^{-1}t \geq 0$

and then the optimal simplex multipliers are given by

$$\pi = \gamma B^{-1} \quad .$$

This means that pos W is decomposable in a number of simplicial cones of the type pos $B = \{t \mid B^{-1}t \geq 0\}$, such that whenever $t \in$ pos B then B is an optimal basis for the linear program: find $y \in R_{+}^{n_2}$ such that $W_y = t$ and $w = qy$ is minimized. Moreover, on pos B, the (optimal) simplex multipliers remain constant. All of these observations can be rendered very precise and are summarized in the Basis Decomposition Theorem, Walkup and Wets, 1969. The figure below illustrates such a decomposition:



4.4 FIGURE: Decomposition of pos W.

Now suppose that we solve the linear program (4.1) for some l, and $B_{(1)}$ is the corresponding optimal basis. Since $B_{(1)}^{-1}$ is readily available, finding the bunch of vectors t^l for which $B_{(1)}$ is the optimal basis is relatively easy since all we need to do is to verify if

(4.5) $B_{(1)}^{-1} t^l \geq 0$.

Let B_1 be the family of all such vectors, $\pi_{(1)}$ be the corresponding simplex multipliers and the probability mass associated with B_1 given by

$$p_{(1)} = \sum_{t^l \in B_1} p_l \quad .$$

All vectors t^l that have failed the nonnegativity test (4.5) are in $\mathcal{T}_1 = \mathcal{T} \setminus B_1$. We are now in the same situation as at the outset. Picking a vector in $\mathcal{T}_1$, we obtain a new basis $B_{(2)}$, the corresponding vector $\pi_{(2)}$ the bunch B_2 and associated probability mass $p_{(2)}$. This process is continued until all $t^l \in \mathcal{T}$ have been bunched. The expected value of these linear programs -- the quantity that would correspond to $\mathcal{Q}(x^\nu)$ or $\Psi(\chi^\nu)$ -- is given by:

$$\sum_k \pi_{(k)} \sum_{t^l \in B_k} p_l t^l \quad .$$

The expected simplex multipliers - a quantity used in the construction of feasibility and optimality cuts -- is given by :

$$\sum_k p_{(k)} \pi_{(k)} \quad .$$

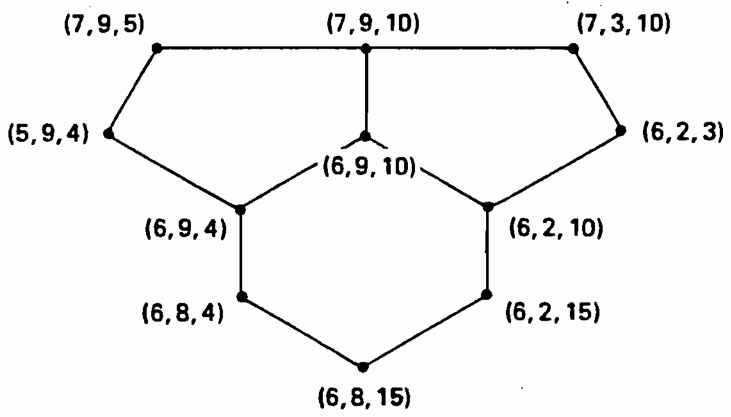
A number of computational shortcuts come immediately to mind as suggested by the decomposition of pos W. First, note that $\mathcal{T}$ or even co$\mathcal{T}$ (the convex hull of $\mathcal{T}$), is a subset of pos W that meets some -- and usually only a few -- of the simplicial cones that are part of this decomposition. Moreover, most of the vectors in $\mathcal{T}$ will be found in adjacent cells, thus instead of just picking any vector t^l that failed the (nonnegativity) test (4.5), we could choose a vector $\hat{t}$ in $\mathcal{T}_1$ such that $\hat{t}$ belongs to a neighboring cell, which necessarily means that $B_{(1)}^{-1}\hat{t}$ has exactly 1 negative entry; note that $B_{(1)}^{-1}t$ having 1 negative entry does not automatically imply that t belongs to an adjacent cell of pos$B_{(1)}$. Passing from pos $B_{(1)}$ to a neighboring cell requires just one (dual) pivot step. It is clear that substantial computational savings could be realized by a systematic organization of the work.

One way is to proceed as suggested in Wets, 1983: pick a vector $t \in \mathcal{T}$, say t^1, and solve the linear program (4.1) with $l = 1$. Let $B_{(1)}$ be the optimal basis. Multiply each vector t in $\mathcal{T}$ by $B_{(1)}^{-1}$. The bunch B_1 is the collection of all vectors t such that

$$\bar{t}^l_{(1)} = B_{(1)}^{-1}t^l \geq 0 \quad .$$

For each vector $\bar{t}^l_{(1)} \in \mathcal{T}_1 = \mathcal{T} \setminus B_1$, with necessarily at least 1 negative element, we record the actual number of negative entries as well as m_l the magnitude of the most negative element. Now choose a vector t in $\mathcal{T}_1$ with a minimal number of negative entries and among them one with m_l as small as possible. Pivot, relying on the criteria provided by the dual simplex method, to obtain the next (optimal) basis $B_{(2)}$, the associated multipliers $\pi_{(2)}$ and construct $\mathcal{T}_2$; and then continue in a similar manner.

What all of this comes down to is that we build a partitioning of that portion of pos W that covers $\mathcal{T}$ (or co $\mathcal{T}$). What we need is the sublattice structure of the cells that contain $\mathcal{T}$. In certain cases it may be possible to work out the complete decomposition of pos W and then use it whenever we enter Step 3 of the L-shaped algorithm. Each subbasis of W that generates a cell of the decomposition is recorded with labels that point to the neighboring cells. The lattice generated by the decomposition in Figure 4.4, would take the graph structure given in Figure 4.6. The labelling of the nodes could be the indices of the columns in the basis.



4.6 FIGURE: Lattice of the decomposition of pos W.

The pointers would correspond to the pivot step required to pass from one basis to a neighboring one. Here this is a planar graph but that would not necessarily be the case if $m_2 > 3$. In general, working out the complete decomposition of pos W may be a serious undertaking, the number of cells could increase exponentially as a function of m_2 (for n_2 sufficiently large). Even for problems whose recourse matrix W have a network structure, the number of components in a complete decomposition of pos W may become unmanageable even for relatively "small" problems, see Wallace, 1984.

Short of first working out a complete decomposition and then finding a good path through the lattice, so as to minimize the number of operations, What could be done? What appears the most efficient approach to date is to bunch the elements of $\mathcal{T}$ by a *trickling down procedure* that we describe next. Unless there are some good reasons for proceeding otherwise -- for example the inverse of a "good" subbasis of W is available -- we would start by finding the cell associated with $\bar{t}$, where

$$\bar{t} = \sum_{t^l \in \mathcal{T}} p_l t^l$$

is the mean of the vectors in $\mathcal{T}$, geometrically: the centroid of $\mathcal{T}$. We have to solve the linear program:

find $y \in R_+^{n_2}$ such that $Wy = \bar{t}$,

and qy is minimized.

This yields an optimal basis $B_{(1)}$, its inverse $B_{(1)}^{-1}$ and associated multiplier $\pi_{(1)}$. We assume that $B_{(1)}^{-1}$ is stored as an explicit dense matrix. Now consider t^1 and sequentially perform the multiplications

$$[B_{(1)}^{-1}]_i t^1 = \hat{t}_i^1 \quad .$$

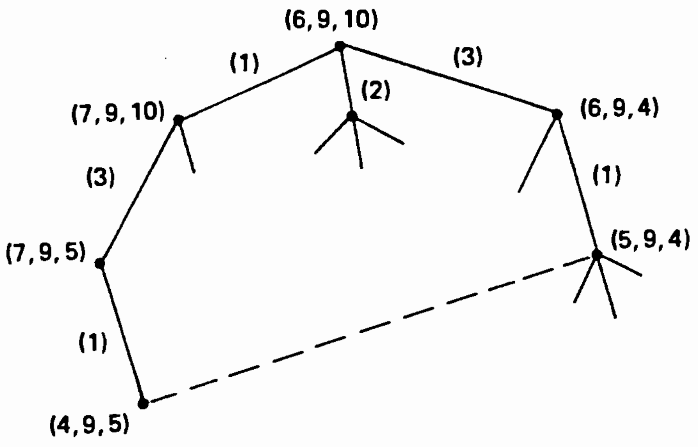
If $\hat{t}_i^1 \geq 0$ for all i, place t^1 in bunch 1, otherwise stop as soon as for some index i, $\hat{t}_i^1 < 0$. Perform *one* dual simplex step, with pivot in row i. In doing so we create a new basis $B_{(2)}$ with

$$[B_{(2)}^{-1}]_i t^1 \geq 0$$

(preserving dual feasibility). The branching from $B_{(1)}$ occurred on i. Repeat the same procedure with $B_{(2)}$ instead of $B_{(1)}$, branching if necessary (recording the branching index), otherwise assigning t^1 to bunch 2. If branching did occur, then continue until a basis $B_{(k)}$ is found such that $B_{(k)}^{-1} t^1 \geq 0$. This will necessarily take place since $t^1 \in \mathcal{T} \subset \text{pos } W$ by assumption, and the pivot path is a simplex path for the dual problem with the pivot choice determined by the first negative entry; degeneracy could be resolved by a random selection rule or Bland's rule. This procedure creates a tree, rooted at $B_{(1)}$, whose nodes correspond to the bases (associated with the cells of the decomposition of pos W), the branches being determined by the first negative entry encountered when multiplying t by $B_{(l)}^{-1}$. Figure 4.7 gives part of such a tree for the decomposition of Figure 4.4 assuming that $\mathcal{T}$ covers pos W, and that

$$\bar{t} \in \text{pos}(W^9, W^6, W^{10}) \quad .$$

The number on the branches indicating branching on the i-th entry that leads to the subsequent basis.



4.7 FIGURE: Tree generated by trickling down procedure.

Note that the same cell may be discovered on different branches of the tree. No effort would be made to recognize that this is taking place, since too much computational effort would be involved in trying to identify such a situation, and only marginal gains could be reaped as will be clear from the subsequent development that concerns updates, i.e. the information necessary to pass from one node of the tree to the next.

It is clear that a great amount of calculations are bypassed by the trickling down procedure, by comparison to the "rough" version of the bunching procedure described at the beginning of this Section. However, it may appear that the storage of all inverse bases (corresponding to the nodes of the tree) as well as keeping track of pointers may negate all the advantages that may be gained from this bunching technique. This, however, can be overcome by relying on Schur-complement updates for the bases $B_{(k)}$. Updates of this type in the context of linear programming were first suggested by Bisschop and Meeraus, 1977, 1980. Suppose $B_{(k)}$ is obtained from $B_{(0)}$ by adding k columns – without loss of generality assume they are $W_{(k)} = [W^{j_1},\dots,W^{j_k}]$ -- and by pivoting out k columns. The equation

$$B_{(k)}y' = t$$

where $y' \in R^{m_2}$ can also be rewritten as

$$\begin{bmatrix} B_{(0)}, & W_{(k)} \\ I_{(k)} & 0 \end{bmatrix} \begin{bmatrix} y' \\ z \end{bmatrix} = \begin{bmatrix} t \\ 0 \end{bmatrix}$$

where $I_{(k)}$ is part of an identity matrix with rows having their entry 1 corresponding to the columns that have to leave the basis when passing from $B_{(0)}$ to $B_{(k)}$. This matrix of coefficients can be written as a block LU product

$$\begin{bmatrix} B_{(0)}, & W_{(k)} \\ I_{(k)}, & 0 \end{bmatrix} = \begin{bmatrix} B_{(0)}, & 0 \\ I_{(k)}, & C_{(k)} \end{bmatrix} \begin{bmatrix} I, & Y_{(k)} \\ 0, & I \end{bmatrix}$$

where the I's in the last matrix are $m_2 \times m_2$ and $k \times k$ indentity matrices. We have that

$$Y_{(k)} = B_{(0)}^{-1} W_{(k)} \;,$$

$$C_{(k)} = -I_{(k)} Y_{(k)} \;,$$

and thus

$$C_{(k)} = -I_{(k)} B_{(0)}^{-1} W_{(k)} \;.$$

This matrix is $k \times k$ and is the only information that is needed to reconstruct all that is needed at the node associated with $B_{(k)}$, in addition to $B_{(0)}^{-1}$ which is supposed to be available (in an LU form, for example). This means that at depth 1 in the tree, only 1×1 updates are necessary; at depth 2, 2×2 updates. Since we reasonably expect to find the largest number of points of $\mathcal{T}$ in the immediate neighborhood of $\bar{t}$ we do not expect to have to construct very long (deep) trees, and the updating information should be of manageable size.

Bunching by the trickling down procedure appears to minimize the amount of operations needed to assign a given $t \in \mathcal{T}$ to its bunch, and by relying on Schur-complement updates the amount of information required at each node is kept very low. When k -- the number of bunches -- gets to be too large it may be necessary to start a tree with a new root. This approach to bunching can even be used effectively in specially structured problems such as worked out in Wallace, 1985, in the case of networks.

The *sifting procedure*, a sort of discrete parametric analysis, has been proposed by Gartska and Rutenberg, 1973. It is designed for handling the case when the points in $\mathcal{T}$ are the possible realizations of m_2 independent random variables, for example when T is nonstochastic and the $h_i(\cdot)$, $i=1,\ldots,m$, are independent random variables. We assume that the vectors in $\mathcal{T} \subset pos\ W$ are obtained by setting for every $i = 1,\ldots,m_2$,

$$t_i = \tau_{il}$$

for some $l \in \{1,\ldots,k_i\}$ where we have ordered the τ_{il} i.e.,

$$\tau_{i1} < \tau_{i2} < \ldots < \tau_{ik_i} \quad .$$

We have thus a doubly indexed array:

$$(4.8) \quad \begin{array}{l} \tau_{11} < \tau_{12} \quad < \ldots\ldots\ldots\ldots\ldots < \tau_{1k_1} \quad , \\ \tau_{21} \quad < \tau_{22} \quad < \ldots\ldots\ldots < \tau_{2k_2} \quad , \\ \ldots\ldots \\ \tau_{m_2,1} < \tau_{m_2,2} < \ldots\ldots\ldots\ldots\ldots < \tau_{m_2,k_m} . \end{array}$$

We sift through this array in the following manner: let

$$t^1 = (\tau_{11},\tau_{21},\ldots,\tau_{m_2,1}),$$

and solve the linear program

find $y \in R_+^{n_2}$ such that $Wy = t^1$,

and qy is minimized.

Suppose that $B_{(1)}$ is the associated optimal basis, with

$$[B_{(1)}^{-1}] = [\beta^1,\beta^2,\ldots,\beta^{m_2}] \quad .$$

Recall that $t \in pos\ B_{(1)}$ as long as $[B_{(1)}^{-1}]t \geq 0$. Hence to find out which subset of vectors belong to pos $B_{(1)}$ we study systematically the range of values of τ that satisfy:

$$\left[\sum_{\substack{j=1 \\ j\neq l}}^{m_i} \beta^j \tau_{j,e_j}\right] + \beta^l \tau \geq 0$$

for some fixed $\tau_{j,e_j} \in (\tau_{j1},\ldots,\tau_{j,k_j})$ and record those values of $\tau_{l,q}$ that belong to that range; the corresponding t-vectors are then in pos $B_{(1)}$. More specifically, identify first the largest index k such that

$$\left(\sum_{j=1}^{m_2-1} \beta^j \tau_{j1}\right) + \beta^{m_2}\tau_{m_2,k} \geq 0 \quad .$$

All vectors $(\tau_{11}, \tau_{21}, \ldots, \tau_{m_2-1,1}, \tau_{m_2,l})$ with $l = 1, \ldots, k$ are recorded as being in pos $B_{(1)}$. We then "move" $\tau_{m_2-1,1}$ to $\tau_{m_2-1,2}$ and repeat the same analysis on the last coordinate of t. If

$$\{\tau \mid (\textstyle\sum_{j=1}^{m_2-2} \beta^j \tau_{j1}) + \beta^{m_2-1}\tau_{m_2-1,2} + \beta^{m_2}\tau \geq 0\} \cap [\tau_{m_2,1}, \tau_{m_2,k_2}] = \phi,$$

we return the (m_2-1)-th coordinate of t to $\tau_{m_2-1,1}$ and increase the preceding element of t to its next higher value, otherwise it is the (m_2-1)-th coordinate which is increased (discretely) to its next higher value, if possible; if not it is again the (m_2-2)-th coordinate which is pushed to its next value. This is continued, systematically, until the search with $B_{(1)}$ is exhausted. We now restart the procedure with the "lowest " vector

$$(\tau_{1j_1}, \tau_{2j_2}, \ldots, \tau_{m_2,j_m})$$

which has not been included in the first bunch, i.e. with the j_l as low as possible. The procedure is repeated until all possible vectors generated by the array have been assigned to a given bunch. Further details can be found in Gartska and Rutenberg, 1973, who also report computational experience which would favor this approach with respect to the coarse bunching procedure described at the beginning of this section. However, to rely on this procedure we must be in this specific situation, i.e. when the vectors in $\mathcal{T}$ can be given the array representation (4.8) and this is not always the case, we often deal with dependent random variables and if (0.1) is the result of an approximation scheme then the chosen discretization will usually not be of this type.

5. CONCLUSION

At this stage of algorithmic development for (linear) stochastic programs with recourse, decomposition-type methods aided by a number of shortcuts made possible by the structural properties of the problem, appear as the clear cut favorites. Of course, this is mostly due to the fact that they allow us to exploit to the fullest these structural properties, see Section 4, but there may be some other justification for using decomposition-type methods. Experiments, cf. Beer, 1981, have shown that with the decomposition method, a value near the optimum -- Beer speaks of an error of no more than 3% -- is reached at an early stage of the computation. Given on one hand the stability of the solution to stochastic programs -- see Dupačova, 1984, Wang, 1984 -- and on the other hand our limitations in the (precise) description of stochastic phenomena or other sources of uncertainties, as mentioned in Section 1, a rapid convergence to an approximate solution is all that is expected and required. If solving the discrete stochastic program (0.1) is part of a sequential scheme for solving a stochastic program with continuous probability distribution or with a discrete distribution involving many more points than L , then it would *not* be necessary to solve up to optimality before a further refinement is introduced. Again decomposition-type methods that exhibit rapid convergence to nearly optimal solutions would be ideally suited in such a scheme.

REFERENCES

K. Beer, 1981. Solving linear programming problems by resource allocation methods, in *Large-scale Linear Programming* eds. G. Dantzig, M. Dempster, and M. Kallio, IIASA Collaborative Proceedings Series, Laxenburg, 409-424.

J. Birge and R. Wets, 1984. Designing approximation schemes for stochastic optimization problems, in particular stochastic programs with recourse, *Mathematical Programming Study*

J. Bisschop and A. Meeraus, 1977. Matrix augmentation and partitioning in the updating of the basis inverse, *Mathematical Programming*, 13, 241-254

J. Bisschop and A. Meeraus, 1980. Matrix augmentation and structure preservation in linearly constrained control problems, *Mathematical Programming*. 18, 7-15.

G. Dantzig and A. Mandansky, 1961. On the solution of two-stage linear programs under uncertainty , *Proc. Fourth Berkeley Symposium on Mathematical Statistics and Probability, Vol.1* Univ. California Press, Berkeley, 1961. 165-176

J. Dupačova, 1984. Stability in stochastic programming with recourse - estimated parameters, *Mathematical Programming*, 28, 72-83.

J. Dupačova and R. Wets, 1984. On the asymptotic behaviour of constrained estimates and optimal decision . Manuscript, IIASA, Laxenburg (forthcoming).

R. Fourer, 1984. Staircase matrices and systems, *SIAM Review*, 26, 1-70.

S. Gartska and D. Ruthenberg, 1973. Computation in discrete stochastic programs with recourse, *Operations Research*, 21, 112-122

J. Ho, 1983. Equivalent piecewise linear formulations of separable convex programs, Manuscript, Univ. Tennessee, Knoxville.

P. Kall, 1979. Computational methods for solving two-stage stochastic linear programming problems, *Z. Angew. Math. Phys.* 30, 261-271.

J. Midler and R. Wollmer, 1969. Stochastic programming models for scheduling airlift operations, *Naval Res. Logist. Quat.* 16, 315-330

L. Nazareth and R. Wets, 1984. Algorithms for stochastic programs: the case of nonstochastic tenders, *Mathematical Programming Study*.

A. Perold and G. Dantzig, 1979. A basis factorization method for block triangular linear programs in *Sparse Matrix Proceedings, 1978*, eds I. Duff and G. Stewart, SIAM Publications, Philadelphia, 283-312.

L. Qi, 1984. Forest iteration method for stochastic transportation problem, *Mathematical Programming Study*.

G. Salinetti, 1983. Approximations for chance-constrained programming problems, *Stochastics*.

M. Saunders, 1983. Private Communication.

R. Van Slyke and R. Wets, 1969. L-shaped linear programs with applications to optimal control and stochastic programming, *SIAM J. Appl.Math.* 17, 638-663

B. Strazicky, 1980. Some results concerning an algorithm for the discrete recourse problem, in *Stochastic Programming* ed. M. Dempster, Academic Press, London. 263-274.

D. Walkup and R. Wets, 1969. Lifting projections of convex polyhedra, *Pacific J. Mathem.* 28, 465-475.

S. Wallace, 1984. Decomposing the requirement space of a transportation problem into polyhedral cones, *Mathematical Programming Study*.

S. Wallace, 1985. On network structured stochastic optimization problem, *Report* no. 842555-8, Chr. Michelsen Institute, Bergen Norway.

J.Wang, 1984. Distribution sensitivity analysis for stochastic programs with recourse, *Mathematical Programming*.

R. Wets, 1966. Programming under uncertainty: the equivalent convex program, *SIAM J. Appl. Math.* 14, 89-105

R. Wets, 1972. Characterization theorems for stochastic programs, *Mathematical Programming*, 2, 166-175.

R. Wets, 1974. Stochastic progrograms with fixed recourse: the equivalent deterministic program, *SIAM Review*, 16, 309-339

R. Wets, 1983. Stochastic programming: solution techniques and approximation schemes, in *Mathematical Programming: The State of the Art, 1982*, eds. A. Bachem, M. Grotsched and B. Korte, Springer Verlang, 566-603