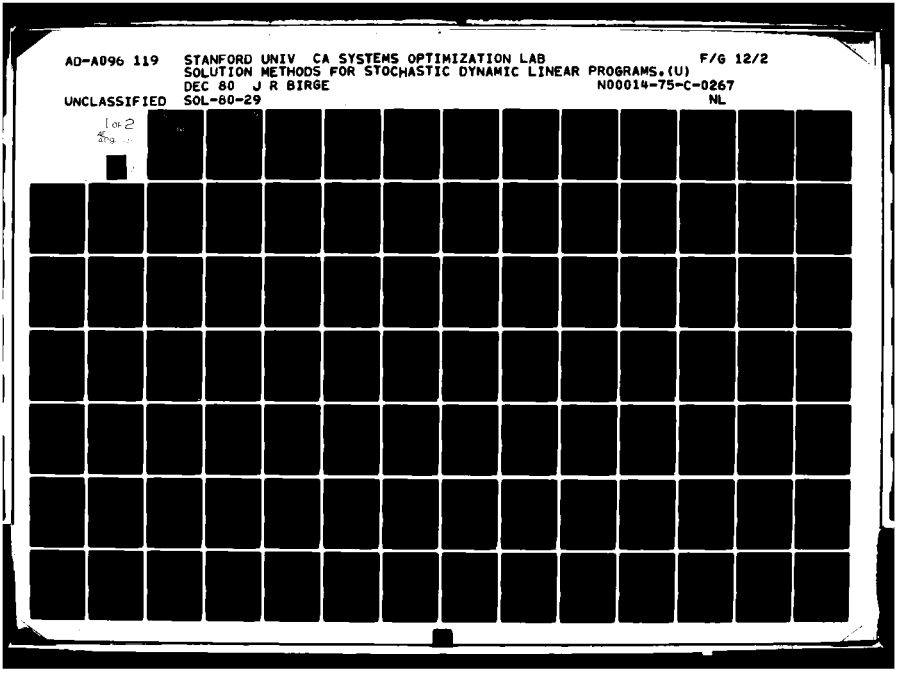

AD-A096 119
STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB
F/G 12/2
SOLUTION METHODS FOR STOCHASTIC DYNAMIC LINEAR PROGRAMS.(U)
DEC 80 J R BIRGE
N00014-75-C-0267
UNCLASSIFIED
SOL-80-29
NL
1 OF 2

Systems
Optimization
Laboratory

LEVEL II

AD A096119

DTIC FILE COPY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

Department of Operations Research
Stanford University
Stanford, CA 94305

81 3 3 042

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

SOLUTION METHODS FOR STOCHASTIC DYNAMIC LINEAR PROGRAMS

by

John R. Birge

TECHNICAL REPORT SOL 80-29

December 1980

Research and reproduction of this report were supported by the Department of Energy Contract DE-AC03-76SF00326, PA# DE-AT03-76ER72018; Office of Naval Research Contract N00014-75-C-0267; National Science Foundation Grants MCS76-81259, MCS-7926009 and ECS-8012974 (formerly ENG77-06761).

PREFACE

Many practical problems in industrial and social planning require optimal decisions to be made periodically through time. Linear programs, called *dynamic linear programs*, can often be formulated to model the requirements of these decision processes. These programs are generally quite large and difficult to solve. The search for efficient methods in finding their optimal solutions has been a major topic in operations research for the past 25 years.

Often the problem modeled by a dynamic linear program involves uncertainties which can complicate and exponentially increase the program's size. There may be several possible outcomes in the future, but deterministic linear programs usually only consider the most likely outcome. In this dissertation, we present methods for solving the *stochastic dynamic linear program*, the dynamic linear program with uncertainties explicitly included.

Our methods take advantage of the program structure. Dynamic linear programs are characterized by a *staircase* structured coefficient matrix, in which non-zero elements appear only in blocks along the diagonal or adjacent to the diagonal. This structure makes many efficient techniques possible. We will show that the stochastic model's specific structure can lead to additional procedures, and that these procedures may improve upon complicated "brute force" solution methods.

We begin in Chapter I by presenting sufficient conditions for a deterministic problem's optimal solution to solve a stochastic problem. The second chapter discusses the difficulties involved in using deterministic solutions in general. We also explore the possibilities for combining separate deterministic solutions and give examples of problems that require the stochastic dynamic

linear program to be solved.

Chapters III, IV, and V present methods for solving the full stochastic program. The first method follows from the decompositions approach to large–scale programming. The next two methods employ different large–scale structured programming techniques, in which, the basis is partitioned but not completely decomposed.

Chapter VI demonstrates that the methods we present are actually dynamic programming approaches. They only differ in their strategies for approximating the optimal state space solution at each stage.

In the final chapter, we present some computational results for our algorithms and discuss potential areas of applications. We also state our conclusions on the use of stochastic dynamic linear programs and suggest areas of future research.

In this dissertation, we use standard mathematical notation. More specific notational conventions are defined in the text. Within each chapter, we refer to equations and propositions by their order of presentation in that chapter (eg.,equation (12)). In reference to equations in other chapters, we prefix the equation by the chapter's roman numeral (eg., (II.12)).

TABLE OF CONTENTS

CHAPTER I

Deterministic Solutions for Stochastic Dynamic Linear Programs

1. Introduction

Dynamic linear program models have been formulated for many different practical situations. When these models involve uncertain quantities, the solution of the resulting stochastic dynamic linear program can be very difficult. Under some circumstances, however, an associated deterministic problem can be stated that is easier to solve and can be used as the "solution" to the stochastic program.

In this chapter, we will state conditions that imply that this *deterministic solution* is in fact optimal for the stochastic dynamic linear program. We call a solution "deterministic" it solves a program that does not allow for any uncertainty in the program parameters. We also will use *myopic solution* to refer to a solution of a program in period t that does not make use of information from periods after t. A solution that considers uncertainty in the future, is called *stochastic*.

In our development of an optimal deterministic solution, we first present the basic multi–stage model and the various approaches taken for its solution. Also, in Section 2, we introduce some terms and notation that appear in the following chapters and discuss the value of having information about the random variables. To do this, we present inequalities that measure the superiority of the stochastic solution over a deterministic solution.

The chapter concludes in Section 3 with the description of conditions

for an optimal deterministic solution. We also give an example of a problem in which a deterministic solution is optimal and illustrate how added complications can necessitate a stochastic solution.

2. The Multi–Stage Problem

The dynamic, or multi–stage, linear program, to which we shall refer, has the following form:

$$
\begin{array}{lrcll}
\min & z_1 = c_1x_1 + c_2x_2 & \cdots & + c_Tx_T & \\
\text{subject to} & A_1x_1 & & & = b_1, \\
& -B_1x_1 + A_2x_2 & & & = \xi_2 \\
& & & & \vdots \\
& & & -B_{T-1}x_{T-1} + A_Tx_T & = \xi_T, \\
& & & x_t & \geq 0 \text{ for all } t,
\end{array}
\qquad (DLP)
$$

where $x_t \in \Re^{n_t}$ (n_t-dimensional Euclidean space), $b_1 \in \Re^{m_1}$, $\xi_t \in \Re^{m_t}$, and the vectors, c_t, and matrices, A_t and B_t, are dimensioned to conform.

For a DLP problem, the right–hand sides, ξ_t, are given. For a situation in which ξ_t is random, the DLP becomes one possible program out of the possible outcomes for ξ_t. In our analysis here, we will not let any of the other quantities be random, so c_t, A_t, B_t, and b_1, will be assumed known. Moreover it is assumed ξ_t and $\xi_{t'}$ are independent $t \neq t'$.

We can also view DLP as an optimal control problem by assuming more structure for the matrices. In this case, we would have $x_t = (y_t, u_t)$, where y_t is a state variable and u_t is a control variable, and we would partition the matrices and vectors as

$$A_t = \begin{pmatrix} I & 0 \\ G_t & D_t \end{pmatrix},$$

$$\xi_t = \begin{pmatrix} 0 \\ \xi_t^2 \end{pmatrix} \tag{1}$$

and

$$B_t = \begin{pmatrix} A_t & B_t \\ 0 & 0 \end{pmatrix}. \tag{2}$$

This formulation includes transitions from state to state according to

$$-A_t y_t - B_t u_t = y_{t+1}$$

and interperiod requirements of

$$G_{t+1} x_{t+1} + D_{t+1} u_{t+1} = \xi_{t+1}.$$

The random vector y_{t+1} represents unknown state to state transitions and future requirements. Although some computational efficiency may be afforded by the special structure of this model, we will restrict our discussion to the general case of DLP.

The first approach we consider for DLP is to solve it for all possible ξ_t and then take the expected value of z_1 as the measure of cost. This approach requires *perfect information* of the outcomes of all future events and is known in the literature as the "wait–and–see" solution (see Madansky [38]). It would be the optimal solution, if one could somehow wait and see until the end of the planning horizon and could beforehand make decisions based on what will occur. Obviously such a perfect information solution is not implementable. When averaged over every possible ξ_t, it provides a measure of the best expected value one could achieve given advanced information about the random variables. We write the expected value of this solution as

$\bar{z}_1$, where,

$$\bar{z}_1 = \mathrm{E}_{\xi_T,\ldots,\xi_2}[z_1(\xi_T,\ldots,\xi_2)], \tag{3}$$

the expectation, "E", is with respect to the random variables, $\xi_2,\ldots,\xi_T$, and z_1 is defined as in DLP with parameters $\xi_2,\ldots,\xi_T$.

An alternative and more realistic approach to the stochastic problem is to consider that during each period t the value ξ_t is known and a decision on x_t must be made without knowledge of the realizations of the future periods' uncertainties. This is known as the "here–and–now" solution because it reflects the need for current decisions. The program can be written as

$$\begin{array}{lrlllll}
\min & z_2 = & c_1x_1 & + \mathrm{E}_{\xi_2}[c_2x_2 & + \mathrm{E}_{\xi_3}[c_3x_3 & ++ \mathrm{E}_{\xi_T}[c_Tx_T]] & \\
\text{subject to} & & A_1x_1 & & & & = b_1, \\
& & -B_1x_1 & +A_2x_2 & & & = \xi_2, \\
& & & & & & = \vdots \\
& & & & -B_{T-1}x_{T-1} & +A_Tx_T & = \xi_T, \\
& & & & & x_t & \geq 0, \\
& & & & & \xi_t & \in \Xi_t,
\end{array}$$

for $t = 2,\ldots,T$. (4)

Problem (4) states that for $t = T$, $\hat{x}_{T-1}$ is given , $\xi_T = \hat{\xi}_T$ is observed and $\hat{x}_T \geq 0$ chosen so that c_Tx_T is minimum. Assuming that $\hat{x}_T$ will be so chosen, $\hat{x}_{T-1}$ is chosen so that, $c_{T-1}\hat{x}_{T-1} + \mathrm{E}_{\xi_T}[c_Tx_T]$ is minimum given $\hat{x}_{T-2}$ and ξ_{T-1} observed, etc.

In general, the decision process for an actual implementation proceeds as follows:

A. A decision, $\hat{x}_1$, is made and implemented.

B. A realization, $\hat{\xi}_2$, of ξ_2 is observed, and a decision, $\hat{x}_2$, is made and

implemented.

C. The process is repeated to find each $\hat{x}_t$ in period t, given the past decision $\hat{x}_{T-1}$ and the outcome of the random vector ξ_{T-1}.

We are going to investigate the effects of using different methods for determining the decisions, $\hat{x}_t$. Not all of these methods exactly solve (4), so we must evaluate the expected value of the objective function for solutions by each method. For a given method, μ, of choosing $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_T$, we first define a function of the random variables, ξ_t, which gives the cost of using those decisions. We write this as

$$z(\hat{x}, \xi \mid \mu) = c_1 \cdot \hat{x}_1(\mu) + c_2 \cdot \hat{x}_2(\hat{x}_1, \hat{\xi}_2 \mid \mu) + \cdots + c_T \cdot \hat{x}_T(\hat{x}_1, \ldots, \hat{x}_{T-1}, \hat{\xi}_2, \ldots, \hat{\xi}_{T-1} \mid \mu), \tag{5}$$

where $\hat{x}_t(\hat{x}_1, \ldots, \hat{x}_{t-1}, \hat{\xi}_2, \ldots, \hat{\xi}_t \mid \mu)$ is the decision chosen by μ given the previous decisions, $\hat{x}_1, \ldots, \hat{x}_{t-1}$, and observations, $\hat{\xi}_2, \ldots, \hat{\xi}_t$.

We can then take expectations over the random vectors to determine the expected cost of the solution found by method μ. We write this as

$$\bar{z}(\mu) = \mathrm{E}_\xi[z(\hat{x}, \xi \mid \mu)]. \tag{6}$$

An exact (but expensive) method for finding an optimal solution to (4) is to proceed by a dynamic programming scheme with backward iteration. We will call this "Method 2". We begin by setting the terminal valuation function:

$$\begin{aligned} z_2^T(x_{T-1}, \xi_T) = \ & \min c_T x_T \\ \text{subject to} \quad & A_T x_T = \xi_T + B_{T-1} x_{T-1}, \\ & x_T \geq 0 \end{aligned} \tag{7}$$

and let $z_2^T(x_{T-1}) = \mathrm{E}_{\xi_T}[z_2^T(x_{T-1}, \xi_T)]$. We further define the recursion on the valuation function as:

$$\begin{aligned} z_2^t(x_{t-1}, \xi_t) = & \min c_t x_t + z_2^{t+1}(x_t) \\ \text{subject to} \quad & A_t x_t = \xi_t + B_{t-1} x_{t-1}, \\ & x_t \geq 0 \end{aligned} \tag{8}$$

and let $z_2^t(x_{t-1}) = \mathrm{E}_{\xi_t}[z_2^t(x_{t-1}, \xi_t)]$. We finally arrive at

$$\begin{aligned} z_2^1(b_1) = & \min c_1 x_1 + z(x_1) \\ \text{subject to} \quad & A_1 x_1 = b_1, \\ & x_1 \geq 0. \end{aligned} \tag{9}$$

Method $\mu = 2$ leads to a sequence of decisions, $\hat{x}_t(\mu = 2)$. We can take expectations of the outcome of these decisions as in (6). This yields

$$\overline{z}_2 = \mathrm{E}_{\xi}[z_2(\hat{x}, \xi \mid \mu = 2)]. \tag{10}$$

We observe that for some outcome $\hat{\xi} = (\hat{\xi}_2, \ldots, \hat{\xi}_T)$,

$$z_2(\hat{x}, \hat{\xi} \mid \mu = 2) = c_1 \hat{x}_1(z_2^1) + c_2 \hat{x}_2(z_2^2(\hat{x}_1, \hat{\xi}_2)) + \cdots + c_T \hat{x}_T(z_2^T(\hat{x}_{T-1}, \hat{\xi}_T))$$

where, for each t, $\hat{x}_t(z_2^t(\hat{x}_{t-1}, \hat{\xi}_t))$ is the optimal solution of (8). Hence, by integrating, we obtain

$$\overline{z}_2 = z_2^1(b_1).$$

So, for Method 2, the result in (9) is the minimum expected cost found by the method.

The expected value, $\overline{z}_2$, is the best possible solution for situations in which decisions must be implemented over time. (For details on this result,

see Chapter VI on dynamic programming.) Enumerating the states, x_{t-1}, in general, can be very difficult, especially when ξ_t can have a continuous distribution. For this reason, we assume that either the distribution of ξ_t is discrete for all t or that we can approximate the continuous distribution by a discrete sample of size $\bar{k}_t$ where $\bar{k}_t$ is not too large. We assume, therefore, that for all t and some $\hat{\xi}_t \in \Re^{m(t)}$,

$$\mathrm{P}(\xi_t = \hat{\xi}_t) = \begin{cases} p_t^1, & \text{if} \quad \xi_t = \xi_t^1; \\ p_t^2, & \text{if} \quad \xi_t = \xi_t^2; \\ ., & \\ p_t^{\bar{k}_t}, & \text{if} \quad \xi_t = \xi_t^{\bar{k}_t}; \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

where by our independence assumptions the probabilities p_t^i do not depend on earlier outcomes.

We have then $\bar{k}_t$ possible outcomes for the random right–hand sides in period t. The outcomes form a tree of possible values (see Figure 1.).

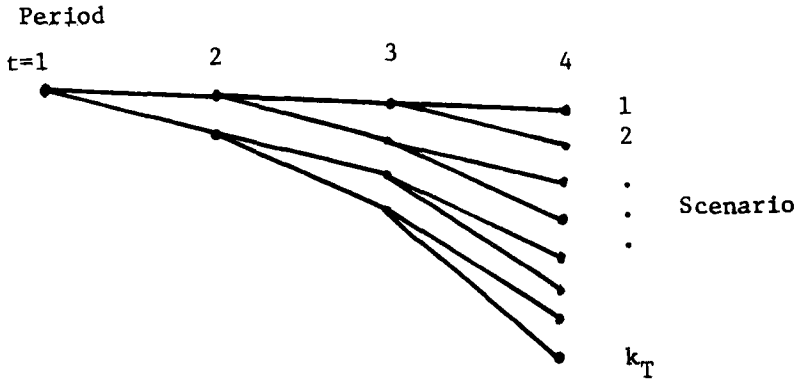


Figure 1. The outcome tree.

We also define k_t as the total number of possible outcomes from period 1 to period t, thus

$$k_t = \Pi_{\tau=1}^{t} \bar{k}_\tau. \tag{12}$$

The tree of outcomes includes k_t nodes in each period t. We call each node a *scenario*. In period t then, a scenario corresponds to a realization of outcomes, $\xi_1 = \hat{\xi}_1, \ldots, \xi_t = \hat{\xi}_t$.

Using this framework, a *descendant* of a scenario j in period t, is defined as any node in periods $t+1$ to T on the branch connected to node j. We adopt the notation $\bar{j}$ for descendants of j. An ancestor of j is then a node on the same branch as j in periods 1 to $t-1$. We denote an ancestor of j as $\hat{j}$. These definitions will apply in this and all subsequent chapters.

Given the discrete distribution, we can write (4) as an explicit linear program. (4) becomes

$$\begin{aligned}
z_3 = \min \quad & c_1x_1 + \textstyle\sum_{j=1}^{k_2} p_2^j c_2 x_2^j \quad + \cdots + \quad \sum_{j=1}^{k_T} p_T^j c_T x_T^j \\
\text{subject to} \\
& A_1x_1 && && = b_1, \\
& -B_1x_1 \quad + A_2x_2^j && && = \xi_2^j, \text{ for all } j, \\
& && && \vdots \\
& && -B_{T-1}x_{T-1}^{\hat{j}} \quad + A_Tx_T^j && = \xi_T^j, \text{ for all } j, \\
& && && x_t \quad \geq 0 \text{ for all } t.
\end{aligned} \tag{SDLP}$$

This program, SDLP, is the primary focus of our presentation. It represents a formulation of the general stochastic dynamic linear program given a discrete distribution. By defining p_t^j as the probability of a node in the tree of

outcomes, we can also incorporate interdependence of the time periods into the model.

SDLP will be analyzed as a structured linear program (presented in Figure 2). Its structure resembles the staircase structure of deterministic dynamic linear programs, but the repetition of the $-B_t$ blocks for each descendant scenario forms spikes below the diagonal. This property makes the strict application of staircase approaches difficult.

Another complication of the stochastic model is that the number of *blocks*, non–zero partitions of the coefficient matrix, grows exponentially with the number of periods, as we see in (12). We will present methods for solving SDLP that reduce the effects of this complication.

The decision process of solving SDLP will be called Method $\mu = 3$. The expected cost found by this method is

$$\bar{z} = \mathrm{E}[z_3(\hat{x}, \xi \mid \mu = 3)], \tag{13}$$

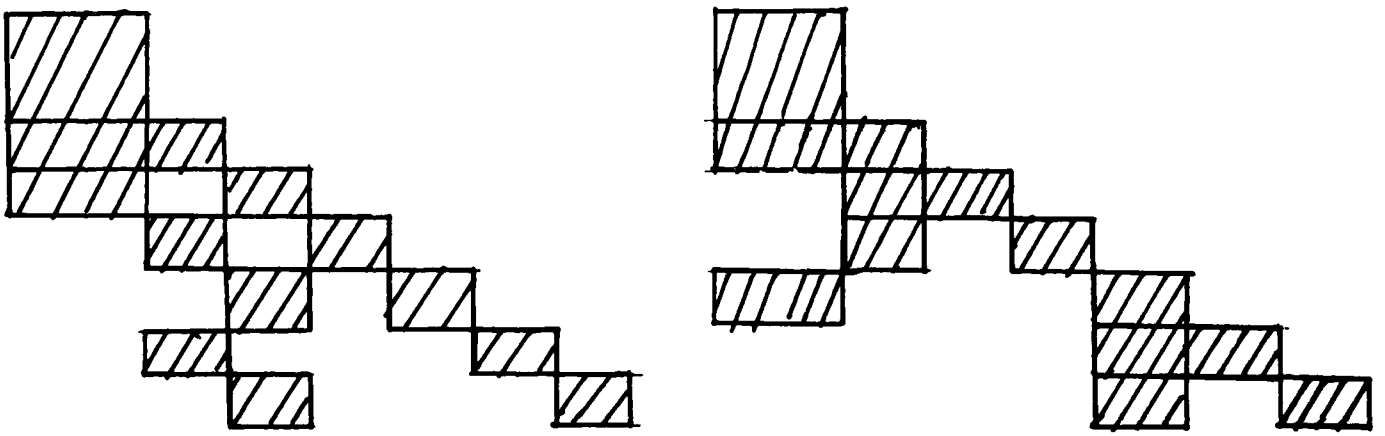


Figure 2. Alternative structures.

An extreme simplification in SDLP would be to consider only one possible outcome for each random variable. A reasonable choice would be the expectation, $\overline{\xi}_t$. The corresponding program has the same form as DLP in which the means, $\overline{\xi}_2, \ldots, \overline{\xi}_T$, appear on the right–hand side. The solution of this problem eliminates the increasing size problem of SDLP and is, therefore, the form usually chosen in practice. In some instances, as we show below, it can even yield the optimal solution to the stochastic program.

We call this deterministic approach of substituting the expectations for the random variables, Method $\mu = 4$. We compute the expected cost of Method 4 then as

$$\overline{z}_4 = \mathrm{E}_\xi[z_4(\hat{x}, \xi \mid \mu = 4)]. \tag{14}$$

The above four approaches to the stochastic dynamic linear program are all related to one another as the following lemma states.

Lemma 1. *The expected cost of the approaches presented above are ordered by*

$$\overline{z}_1 \leq \overline{z}_2 \leq \overline{z}_3, \quad \overline{z}_1 \leq \overline{z}_2 \leq \overline{z}_4, \tag{15}$$

and, for (discrete) distributions without approximations $\overline{z}_1 \leq \overline{z}_2 \equiv \overline{z}_3 \leq \overline{z}_4$.

Proof. Each method can be shown to improve upon the solution by the previous method. For $\overline{z}_1 \leq \overline{z}_2$, we observe that $z_1(x, \xi \mid \mu = 1) \leq z_2(x, \xi \mid \mu = 2)$ since Method 1 chose the optimum solution for each ξ whereas $\mu = 2$ does not in general. Integration preserves the inequality, hence, $\overline{z}_1 \leq \overline{z}_2$.

The next inequality involves either $\mu = 3$, where an incorrect discrete distribution can be used to approximate the correct one, or $\mu = 4$,where the distribution is replaced by one calculated at the expected value. Since

Method 2 optimizes (8) for all ξ_t, $\mathrm{E}_{\xi_t}[z_2^t(x_{t-1}, \xi_t)] \leq \mathrm{E}_{\xi_{t-1}}[z_3^t(x_{t-1}, \xi_{t-1})]$ for all t. Hence, $\overline{z}_2 \leq \overline{z}_3$ or $\overline{z}_2 \leq \overline{z}_4$ follows. ∎

The lemma includes the obvious result that, for the expected value of perfect information, EVPI, where EVPI$= \overline{z}_1 - \overline{z}_2$ if the correct distribution is used or $\overline{z}_1 - \overline{z}_3$ if not,

$$\text{EVPI} \geq 0.$$

This difference represents the maximum amount that one would pay for information about the future. When the EVPI is low there may be little necessity in refining forecasts, but, when it is high, incomplete information about the uncertainties may be costly. In the following chapter, we present an example of this possibility.

A second quantity that we want to examine here is the difference $\overline{z}_2 - \overline{z}_4$ or $\overline{z}_3 - \overline{z}_4$, which we call the *value of the stochastic solution* , VSS. VSS measures the benefit from solving a stochastic program over solving its deterministic approximation. A low VSS indicates that the more complicated SDLP might not be worth the extra effort. VSS can be bound without solving SDLP, as we show in the next chapter.

We note that in Lemma 1 to guarantee that $\overline{z}_3 \leq \overline{z}_4$, we had to assume that the discrete distribution was correct. If the distribution used was only an approximation, then it is possible to make an estimate of the distribution that would lead to $\overline{z}_3 > \overline{z}_4$. This anomaly can occur because some scenarios could lie under sections of the piecewise linear curve, $z_3^t(x_{t-1}, \xi_t)$, that lead to high penalties. We discuss these scenario results more closely in Chapter 2. For our purposes, we assume that the discrete distribution used suffices because more information about the distribution is not available. The consideration

of individual scenarios is the only alternative.

Given these assumptions about the distribution, we would still like to know when SDLP should be solved. To show when SDLP need not be solved, in the next section, we present some conditions that imply VSS=0. In the following chapter, we will give bounds on VSS that also aid in evaluating whether SDLP is worth solving.

3. Optimal Deterministic Solutions

When the numerical costs of solving a stochastic problem are high, a deterministic solution technique is attractive. Since decisions often cannot wait for the detailed analysis of all future possibilities, the method based on assuming some "best guess" of the future environment, is most often the one implemented. In fact, the even simpler policy of using a myopic solution may provide a good basis for decisions. By finding conditions for VSS=0, we are trying to avoid the effort of correctly solving a general stochastic program. With the conditions below, we can check whether the stochastic program need be tried at all. The following lemma, a well-known result from sensitivity analysis, is fundamental in our development.

Lemma 2. *Let B be an optimal basis for DLP with $\xi = (\bar{\xi}_2, \bar{\xi}_3, \ldots, \bar{\xi}_T)$. If B remains feasible for all $\xi \in \Xi$, then B is an optimal basis for DLP for all ξ.*

Proof. Partition the coefficient matrix and cost row according to basic variables, x_B, and non-basic variables, x_N. DLP becomes

$$\begin{aligned} \min \quad & c_B x_B + c_N x_N \\ \text{subject to} \quad & B x_B + N x_N = (b_1, \overline{\xi})', \\ & x_B \geq 0, \\ & x_N \geq 0, \end{aligned} \tag{16}$$

where we use the notation v' to indicate the transpose of v, so $(b_1, \overline{\xi})'$ is the column vector of right–hand sides in DLP.

For B optimal, there exist prices, π, such that

$$\begin{aligned} \pi B &= c_B, \\ \pi N &\leq c_N, \end{aligned} \tag{17}$$

and

$$\begin{aligned} x_B = B^{-1}(b_1, \overline{\xi})^T \geq 0, \\ x_N \geq 0. \end{aligned} \tag{18}$$

If x_B remains feasible for all ξ in (18), (17) still holds, guaranteeing dual feasibility and complementarity. Hence, B is still an optimal basis. ∎

The next problem we might encounter is that of testing whether B is indeed feasible for all values of ξ. An enumeration of all possible ξ is not necessary. Garstka and Rutenberg [25] showed that simple computations for many practical problems, could be performed quickly to find the probability that a given basis is optimal. Their process involves fixing some components in the lattice of discrete values of ξ and then finding the feasible range for the remaining components. This method also proves valuable in the subproblem solutions we investigate in Chapter 3.

To use a basis which satisfies the conditions in Lemma 2 in SDLP, still other conditions must be met. The next lemma helps us find these conditions. For this lemma, we will use a solution from DLP in SDLP. We do this by letting the set of basic activities in DLP, $\{x_t^B : t = 1, \ldots, T\}$, be repeated to

form a basic set in SDLP. This basic set in SDLP is $\{x_t^{B_j} : j = 1, \ldots, k_t; t = 1, \ldots, T\}$, where $x_t^{B_j} = x_t^B$ for all j.

Lemma 3. *Let the set of activities for a feasible basis, B, in DLP be $\{x_1^B, \ldots, x_T^B\}$ where each x_t^B represents activities from period t. Also, let SDLP have at least two distinct new scenarios at each period (i.e., $\overline{k}_t \geq 2$). The activities, $\{x_t^{B_j}\}$, where $x_t^{B_j} = x_t^B$ for all j, form a feasible basis in SDLP if and only if x_t^B consists of $m(t)$ activities for all t.*

Proof. (See Figure 3.) Essentially we shall show that if the count on the number of basic x_t is not $m(t)$ for all t that the corresponding SDLP candidate for basis will be singular. For the necessity of the condition, first let $\mu(t)$ be the number of elements in x_t^B. Assume $x(t)$ does not have $m(t)$ elements for all t. Thus, there exists some $\mu(t) > m(t)$. (If not, since $\sum_{t=1}^T m(t) = m$, $\mu(t) = m(t)$ for all t.)

We note that, for $\mu(t) > m(t)$, $t < T$. This is true because, if $\mu(T) \geq m(T)$, then $\sum_{t=1}^{T-1} \mu(t) < \sum_{t=1}^{T-1} m(t)$, which contradicts the fact that $\{x_t^{B_j}\}$ corresponds to a basis. Set $t' = \min \{t : \mu(t) > m(t)\}$. We note that there exists no $t'' < t'$ such that $\mu(t'') < m(t'')$. Again, this would mean the basis was not of full rank. Now, we have $\sum_{t=1}^{t'} \mu(t) - \sum_{t=1}^{t'} m(t) = \overline{k}_{t'} \cdot \delta_t$, where $\delta_t = \mu(t') - m(t')$. But, for $t > t'$, $\sum_{t=t'+1}^{T} \mu(t) - \sum_{t=t'+1}^{T} m(t) = -\overline{k}_{t'} \cdot \overline{k}_{t'+1} \cdot \delta_t$, since each deficiency is repeated $\overline{k}_{t'+1}$ times. Hence, $\sum_{t=1}^T \mu(t) = \overline{k}_{t'}(1 - \overline{k}_{t'+1}) \cdot \delta_t + \sum_{t=1}^T m(t)$, which, by our assumption, implies the columns of the activities, $\{x_t^B\}$, do not form a basis in SDLP. To show sufficiency, first note that if x_t^B has $m(t)$ elements in each period t, then, for all t, there exists a square non-singular partition of the basis, B_t, with columns and rows only in t. (If not, B does not span the row space in period t.) Therefore, in SDLP, the set of columns, $\{B_t(j)\}$, is linearly independent. By construction, the x_t^B

correspond to $m(1) + \sum_{t=2}^{T} k_t m(t)$ columns, so the activities form a basis. ∎

From this lemma, we obtain our result as stated in the following theorem.

Theorem. *If the optimal basis, B, for the program DLP, with $\xi = \bar{\xi}$, is feasible for all $\xi \in \Xi$, and if B has as many columns as there are rows in each period, then the set of activities in B forms an optimal basis in SDLP, and $VSS = 0$.*

Proof. First, to show primal feasibility, let $A_t^{B_j}$ be the square non–singular submatrix associated with the activities x_t^B in scenario j. For all t and j, we have

$$\hat{x}_t^{B_j} = (A_t^{B_j})^{-1}(\hat{\xi}_t^j + B_{t-1}\hat{x}_{t-1}^{B_j}), \tag{19}$$

which is the same value as $\hat{x}_t^B$ in DLP for $\xi_t = \hat{\xi}_t$. Hence, $\hat{x}_T^{B_j} \geq 0$ by Lemma 1 for all j and t.

Let π_t^B be the dual variables in DLP for the basis, B. Next, define $\pi_t^{B_j} = p^j \pi_t^B$. In period T, we obtain

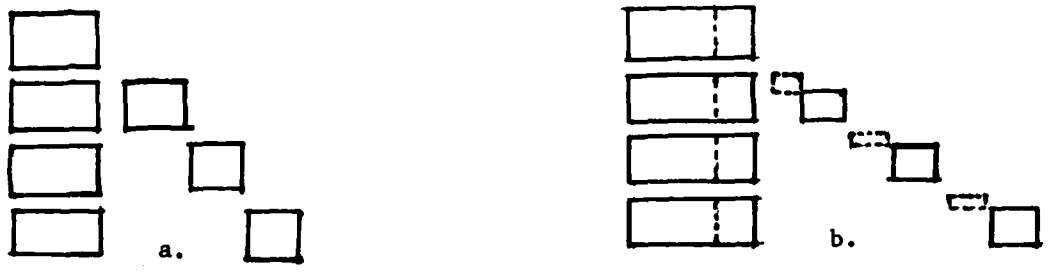


Figure 3. Example of (a) a basis and (b) non–spanning columns for a 2 period example.

$$\pi_t^{B_j} = p^j \pi_t^B \tag{20}$$

and

$$\pi_T^{B_j} A_T^{N_j} \leq p^j c_T^N, \tag{21}$$

where the c_T^N are the non-basic costs in scenario j for $j = 1, \ldots, k_T$. In general, for $\pi_t^{B_j}$ in SDLP, we have

$$-\sum_{\bar{j}=1}^{\bar{k}_{t+1}} \pi_{t+1}^{B_{\bar{j}}} B_t^{B_j} + \pi_t^{B_j} A_t^{B_j} = -p^j \pi_{t+1}^B B_t^B + p^j \pi_t^B A_t^B = p^j c_t^B \tag{22}$$

and

$$-\sum_{\bar{j}=1}^{\bar{k}_{t+1}} \pi_{t+1}^{B_{\bar{j}}} B_t^{N_j} + \pi_t^{B_j} A_t^{N_j} \leq c_t^N. \tag{23}$$

By (22) and (23), for $x_t^{B_j}$ feasible, B is an optimal basis in SDLP. (19) implies that the solution in SDLP given scenario j is the same as in DLP. Hence, $\bar{z}_3 = \bar{z}_4$, and VSS $= 0$. We further note that this also implies the optimal solution is myopic. ∎

This result gives us a method to check for deterministic optimality, but it may be difficult to satisfy these conditions in practical examples. Even when not satisfied, they could be useful, however, in finding an optimal solution with confidence α, where α is the probability of being optimal. The Garstka and Rutenberg procedure mentioned above would be useful for these computations. (This could also be applied to a problem of the form of a *chance-constrained program* as in Charnes and Cooper [14], but we wish to restrict our development to the *recourse problem*, SDLP.)

The next result presents an alternative set of conditions that may prove useful when the conditions in our theorem are not satisfied. We state them as a corollary.

Corollary. *Let $\{B(i)\}$ be a family of bases for DLP, where $B(i)$ is optimal for $\xi_i \in \Xi$. Assume also that, for all periods t and nodes j of scenario i, the set of basic activities, $\{x_t^{B(i(j))}\}$, is the same for all $i(j)$ that include the nodes, $\{\bar{j}_l : l = 1, \ldots, \bar{k}\}$, the descendants of j. If each $B(i)$ additionally has square blocks in each period, then the set of basic activities chosen from $\{B(i)\}$ is optimal in SDLP, and VSS $= 0$.*

Proof. Since $x_t^{B(i(j))}$ is the same for all $\bar{j}$, we can define a set of activities for SDLP as $x_t^{B_j} = x_t^{B(i(j))}$ for all t and j. Now, for primal feasibility, we again have (19) for all t and j, so $\hat{x}_t^{B_j} \geq 0$.

For the dual, define

$$\pi_t^{B_j} = p^j \pi_t^{B(i(j))}. \tag{24}$$

Hence, at period T,

$$\pi_T^{B_j} A_T^{B_j} = p^j c_T^B, \tag{25}$$

and

$$\pi_T^{B_j} A_T^{N_j} \leq p^j c_T^N. \tag{26}$$

For general t, we have

$$\begin{aligned} -\sum_{\bar{j}=1}^{\bar{k}_{t-1}} \pi_{t+1}^{B_j} B_t^B + \pi_t^{B_j} A_t^{B_j} &= \sum_{\bar{j}=1}^{\bar{k}_{t+1}} p^{\bar{j}} (-\pi_{t+1}^{B(i(\bar{j}))} B_t^B + \pi_t^{B(i(j))} A_t^{B_j}), \\ &= p^j c_t^B, \end{aligned} \tag{27}$$

and

$$\sum_{j=1}^{K_{s+1}} \pi_{t+1}^{B_j} B_t^N + \pi_t^{B_j} A_t^{N_j} \leq p^j c_t^N. \tag{28}$$

(25), (26), (27), and (29) give us dual feasibility and complementarity, proving that the set of variables $\{x_t^{B_j}\}$ is optimal. Again, from (19), the values are the same as in solving any deterministic form DLP, so VSS $= 0$. ∎

The corollary gives us more conditions for finding the optimal solution to SDLP without actually solving it. An example of a model which meets these requirements is the Hotelling–Nordhaus model of exhaustible resources (see [34] and [46]) and its extension by Chao [12].

In Chao's model, a dynamic production schedule is chosen to minimize the cost of satisfying an increasing sequence of demand requirements over time. The demands may be satisfied by any of $m-1$ technologies, each using one distinct resource, with finite availability and one "backstop" technology with no resource limits. The program is

$$\min \sum_{i=1}^{m} \sum_{t=0}^{\infty} \beta^t c_i y_{it} + \sum_{i=1}^{m} \sum_{t=0}^{T} \beta^t k_i x_{it} \tag{29}$$

subject to

$$\sum_{t=0}^{\infty} y_{it} \leq R_i, i = 1, 2, \ldots, m,$$

$$\sum_{i=1}^{m} y_{it} = D_t, t = 1, 2, \ldots, T,$$

$$y_{i,t+1} = y_{it} + \sum_{s=0}^{\infty} (\delta_s - \delta_{s-1}) x_{i,t-s},$$

$$y_{it} \geq 0, t = 0, 1, \ldots,$$

$$x_{it} \geq 0, i = 1, 2, \ldots, m,$$

where y_{it} is the amount of the demand, D_t, satisfied by resource i at time t, x_{it} is the amount of resource i committed at t so it may be extracted later, c_i is the current cost of technology i, k_i is the capital cost of i, β is the discount factor, δ_t is the extraction rate, and R_i is the initial availability of the resource used in technology i.

Chao showed that, for this model, a myopic solution is optimal for all future demands and supplies. This solution implies that a family of bases, $\{B(i)\}$, exists that satisfies the conditions of the corollary. Therefore, the stochastic program for (29), in which, D_t and R_i are random, has a deterministic and myopic solution and VSS $= 0$. We note that this very simple model can be modified so that VSS grows. Chao explored the case of price–responsive demands and found that, with a "sufficiently high" discount rate, the optimal decisions are still insensitive to "distant–future" uncertainties. *Our example in Chapter II shows* how near future uncertainties can greatly affect current decisions, also making VSS high.

This chapter has described the value of information in the consideration of decisions made over time. We presented the program, SDLP as a method for incorporating uncertainties into a decision process and explored the possibilities for finding a solution to SDLP without solving the full problem. The value of the stochastic solution is, however, not always low.

Chapter II

The Nature of the Stochastic Solution

1. Introduction

In Chapter I, we presented conditions, under which, the optimal solution to a certain deterministic program is the solution to the stochastic dynamic linear program. The deterministic model used the expected values of the random right–hand sides. . Unfortunately, the great majority of stochastic problems do not meet the certainty equivalence criterion, ie. the sufficient conditions for deterministic optimality. When a model fails the conditions of Chapter I, it would be desirable to solve the stochastic problem directly. In this chapter, we explore the value of that solution and the costs that can arise from not finding the optimal stochastic solution.

We shall concentrate on decisions based upon "risk neutrality", meaning we wish to optimize the expected value of our policy decisions. Alternatively, the decision maker might want to minimize the probability of a catastrophic loss or, otherwise, reduce the variance of his expected utility. These attributes could be reflected in a carefully defined nonlinear utility function or in penalties placed on the less attractive scenarios. In this discussion, we do not consider such specifications. Our consideration of linear models should be, however, sufficiently general to allow for further analysis in this area.

This chapter begins with a discussion of bounds on the expected value of perfect information. It then proceeds in Section 3 to examine uses of the deterministic optimization of different scenarios. In Section 4, the possibilities for combining these solutions and the inherent difficulties in the stochastic program are discussed. Sections 5 then presents examples of these problems.

The chapter concludes in Section 6 with a suggestion for a general strategy to be applied in linear optimization under uncertainty.

2. Bounds on the Expected Value of Perfect Information

We discussed above the expected value of perfect information (EVPI) and value of the stochastic solution (VSS) and showed examples when these quantities might be zero. When the conditions for deterministic optimality are not met, we would like to have simple bounds on the EVPI that may help us determine the worth of solving the stochastic program. If the EVPI and VSS are bounded within a tight range, it may be adequate to use a deterministic approach to the problem instead of following an expensive stochastic method.

The expected value, $\bar{z}$, of the objective function, $z(\xi)$, can be bounded because of its convexity. Madansky [42] and later Huang, Ziemba and Ben-Tal [35] examined this property using the theory of moment spaces to bound the expectation. Their work rests on the following result.

Lemma 1. *The objective function, $z(\xi)$, in (1.3) is a convex and continuous function of ξ, the right-hand side.*

Proof. See Madansky [42]. ∎

This result then allows the application of Jensen's inequality for convex functions. Directly from this, we have

$$\bar{z}_1 = \mathrm{E}[z(\xi)] \geq z(\mathrm{E}(\xi)) = z_4, \tag{1}$$

giving a lower bound on the perfect information solution, $\bar{z}_1$. (In this analysis, we use the definitions of z_1, z_2, z_3, and z_4 from Chapter I.)

An upper bound also can be found for $\bar{z}_1$. We present here only the one

dimensional case. The multidimensional case can be found in Madansky [42]. The bound is known as the *Edmundson–Madansky inequality*, and it states that, for $\xi \in [a,b] \subseteq \Re^1$, $\mu = \mathrm{E}[\xi]$,

$$\overline{z}_1 = \mathrm{E}[z(\xi)] \leq \left(\frac{b-\mu}{b-a}\right) z(a) + \left(\frac{\mu-a}{b-a}\right) z(b), \tag{2}$$

where by definition here $\mu = \mathrm{E}[\xi]$. This is shown easily, since for any $t \in [a,b]$, $t = (a(b-t)+(t-a)b)/(b-a) = \lambda a + \overline{\lambda} b$, and, for $z(t)$ convex, $z(t) \leq \lambda z(a) + \overline{\lambda} z(b)$, which by integrating yields (2). We can now state (1) and (2) as

Theorem 1. *For $\xi \in [a,b] \subseteq \Re^1$, $\mu = E(\xi)$, and z as defined above,*

$$(b-\mu)/(b-a)z(a) + (\mu-a)/(b-a)z(b) \geq \overline{z}_1 \geq z(\mu).$$

Huang, Ziemba and Ben–Tal carry these principles further. They subdivide the interval $[a,b]$ and apply successively finer approximations, which they show approach the expectation. This method makes possible the refinement of the EVPI to whatever level is desired.

Another method for computing bounds on the EVPI was presented by Avriel and Williams [5]. They showed that

$$0 \leq EVPI \leq z_2 - z_4 \leq \overline{z}_4 - z_4, \tag{3}$$

where z_2 is the best stochastic solution and z_4 is the expected value deterministic solution, as in Chapter I. They also show that, for $z(\xi)$ differentiable, the bound using the expected value of ξ (as in z_4) is the tightest possible.

These approaches give us methods for estimates of the benefit we may gain from knowledge about the random variables. We concern ourselves here

with the VSS, the value of solving a stochastic program over solving a deterministic one. The EVPI is used to find the value of additional information, but, in looking at the VSS, we assume that no more information is available. We ask: *what given our present state of knowledge, is the value of solving a large stochastic program?*

Similar results to those above can be found for VSS. We can bound VSS as in the following theorem.

Theorem 2. *The value of the stochastic solution (VSS $= \overline{z}_4 - \overline{z}_3$, as defined in Chapter I), satisfies the inequalities*

$$0 \leq VSS \leq \overline{z}_4 - z_4. \tag{4}$$

Proof. We showed VSS ≥ 0 in Chapter I. It suffices to show $z_4 \leq \overline{z}_3$. We had $\overline{z}_1 \leq \overline{z}_2 \leq \overline{z}_3$. Now, $z_4 = z(\mathrm{E}(\xi))$ and $\overline{z}_1 = \mathrm{E}(z(\xi))$, so, for $z(\xi)$ convex, by Jensen's inequality, $z_4 \leq \overline{z}_1$. The result follows. ∎

This bound can prove useful in estimating the benefit of the stochastic program, but, if it remains high, further analysis may be required. As a first step in solving the stochastic program, we may find other deterministic solutions corresponding to different scenarios or outcomes of the random variables. We describe this approach in the next section.

3. The Scenario Approach

In evaluating the perfect information or "wait–and–see" solution, $\overline{z}_1$, a solution to the program, $z_1(\xi)$, must be found for each possible outcome of the random vector, ξ. The scenario approach, also known as "modified wait–and–see" in Gunderson, Morris, and Thompson [31], involves solving several of these deterministic programs, evaluating the expected cost of using the

strategy that is optimal for each scenario, and choosing the strategy that minimizes this expected cost. In many cases, one of the first period bases dominates the others, and the choice for a decision is clear. As Gunderson, et al, emphasize, this method can be quite responsive to management concerns and may prove very useful since it presents alternative possibilities and risks in a compact and easily understandable form.

In the scenario approach, we first choose a set of possible outcomes for ξ, which we call, $\tilde{\Xi}$, where

$$\tilde{\Xi} = \{\xi_1, \ldots, \xi_k\}. \tag{5}$$

For each $\xi^0 \in \tilde{\Xi}$, we find the optimal solution, x^*, for $z(\xi^0)$. Then, we compute

$$\varsigma(\xi^0) \equiv \mathrm{E}[\varsigma(x^*(\xi^0), \xi)] \tag{6}$$

for each ξ^0, where $\varsigma(x^*(\xi^0), \xi)$ is the resulting objective value from using x^* when ξ actually occurs.

Next, we find

$$\varsigma^* \equiv \min_{\xi^0 \in \tilde{\Xi}} \varsigma(\xi^0). \tag{7}$$

This value represents the least expected cost from using the solution of a deterministic program.

This implies that a deterministic problem other than the expected value problem may result in a better solution. This is because less penalty may be incurred by following a piecewise linear section of the objective function other than that that covers the expected value. (See Figure 1.)

We can also bound ς^* as in the following theorem.

Theorem 3. *For Ξ discrete, the best scenario solution , ς^*, satisfies*

$$\overline{z}_3 \leq \varsigma^* \leq \overline{z}_4. \tag{8}$$

Proof. We know

$$\varsigma^* = \min_{\xi^0 \in \Xi} \mathrm{E}[\varsigma(x(\xi^0)), \xi] \leq \mathrm{E}[\varsigma(x(\bar{\xi}), \xi)] = \overline{z}_4$$

and, for x^* optimal in ς^*,

$$\overline{z}_3 = \min_{\xi \in \Xi} \mathrm{E}[\varsigma(x(\xi)), \xi] \leq \mathrm{E}[\varsigma(x^*), \xi] = \varsigma^*.$$

Hence, the result.∎

The inequalities in (8) show that the scenario approach may be useful in finding a closer approximation to the solution of the stochastic program. ς^* may be especially valuable when $\varsigma^* - z_4$ is small, since it also bounds $\overline{z}_3 - z_4$ and may show that solving the stochastic program is unnecessary.

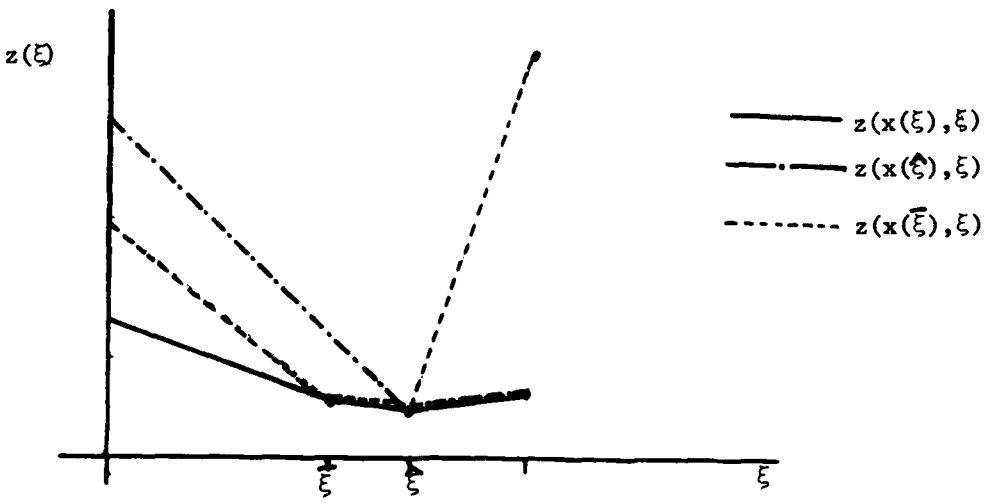


Figure 1. The expected value of $z(x(\bar{\xi}), \xi)$ is greater than that of $z(x(\hat{\xi}), \xi)$.

This outcome is most likely when the optimal decisions in the first period (those that are actually implemented) correspond to the same basis for most of the scenarios.

The scenario approach can also be used to find activities that may be basic in the optimal solution of the stochastic program. If the set of optimal basic activities remains fairly constant for the possible realizations of ξ, then this set of activities may be optimal in the stochastic program. We discuss this possibility in the next section.

4. Combining Scenarios

A natural approach to solving the stochastic program would be to use the optimal solutions of the different scenarios and to combine them in an appropriate manner. The proper combination may, however, be quite difficult to find and may lead to as much effort as solving the stochastic program directly. We present below the problems inherent in combining scenarios and some situations, in which, the optimal combination may be found directly.

In this analysis, we restrict ourselves to a two period case, for which there are only two scenarios considered. The difficulties involved in this example are typical of all stochastic programs, so we present this case because of its simplicity. The results may be easily generalized to more periods and scenarios.

We begin by defining two scenario problems as

$$\begin{aligned} \min\ z(\xi^1) &= c_1x_1 + c_2x_2 \\ \text{subject to}\quad A_1x_1 &= b_1 \\ -B_1x_1 + A_2x_2 &= \xi_2^1 \\ x_1, x_2 &\geq 0, \end{aligned} \tag{S1}$$

and

$$
\begin{aligned}
\min z(\xi^2) &= c_1 x_1 + c_2 x_2 \\
\text{subject to} \quad A_1 x_1 &= b_1 \\
-B_1 x_1 + A_2 x_2 &= \xi_2^2 \\
x_1, x_2 &\geq 0.
\end{aligned}
\tag{S2}
$$

Solving S1 and S2 yields the optimal solutions, $x^{1,*}$ and $x^{2,*}$, and optimal dual prices $(\pi^{1,*}; \sigma^{1,*})$ and $(\pi^{2,*}; \sigma^{2,*})$, where $z^* = \pi^* \cdot b_1 + \sigma^* \cdot \xi$. We also define the following index sets:

$$\beta^1 \equiv \{j : A_1(*, j) \text{ is basic in S1 }\}$$

and

$$\beta^2 \equiv \{j : A_1(*, j) \text{ is basic in S2 }\}. \tag{9}$$

The complements of β^1 and β^2 are defined as $\overline{\beta}^1$ and $\overline{\beta}^2$, respectively.

Now, our actual goal is to solve the following two–period version of SDLP

$$
\begin{array}{lccccc}
\min & c_1 x_1 & +p^1 c_2 x_2^1 & +p^2 c_2 x_2^2 & & \\
\text{subject to} & A_1 x_1 & & & = b_1 & \\
& -B_1 x_1 & +A_2 x_2^1 & & = \xi_2^1 & (SDLP2) \\
& -B_1 x_1 & & +A_2 x_2^2 & = \xi_2^2 & \\
& & & x_1, x_2^1, x_2^2 & \geq 0. &
\end{array}
$$

We would like to use the activities in β^1 and β^2 as the optimal basis of SDLP2. In other words, for $\beta = \{j : A_1(*, j)$ is basic in SDLP2 $\}$, we are looking for $\beta \subseteq \beta^1 \bigcup \beta^2$. Unfortunately, this is not always possible. The difficulty results from the properties of the basis in SDLP2. In order to maintain full rank of the basis in this program, the optimal basic activities

from (S1) and (S2) cannot all be in the optimal basis of SDLP2, unless the square block situation we discussed in Chapter I holds.

In general, when we attempt to combine scenarios, we face a duality gap as the following lemma states.

Lemma 2. *For z^*, the optimal value of SDLP2,*

$$\begin{aligned} \underline{z} &= (p^1\sigma^{1,*} + p^2\sigma^{2,*})b + p^1\pi^{1,*}\xi_2^1 + p^2\pi^{2,*}\xi_2^2 \\ &\leq z^* \\ &\leq c_1(p^1x_1^{1,*} + p^2x_1^{2,*}) + pc_2\tilde{x}_2^{1,*} + p^2c_2\tilde{x}_2^{2,*} = \overline{z}, \end{aligned} \tag{10}$$

where $x_1^{1,*}$ and $x_1^{2,*}$ are the optimal primal solutions for S1 and S2, $\tilde{x}_2^{2,*}$ and $\tilde{x}_2^{1,*}$ are the values of the second period basic variables in S1 and S2 chosen to satisfy $A_2\tilde{x}_2^{1,*} = \xi_2^1 + B_1(p^1x_1^{1,*} + p^2x_1^{2,*})$ and $A_2\tilde{x}_2^{2,*} = \xi_2^2 + B_1(p^1x_2^{1,*} + p^2x_2^{2,*})$, [if $\tilde{x}_2^{1,*}$ or $\tilde{x}_2^{2,*} \leq 0$, set $\overline{z} = +\infty$], and $\pi^{1,*}$, $\pi^{2,*}$, $\sigma^{1,*}$, and $\sigma^{2,*}$ are the optimal dual values for S1 and S2. Furthermore, the duality gap, if $\tilde{x}_2^{1,*}$ and $\tilde{x}_2^{2,*}$ are feasible, is

$$\overline{z} - \underline{z} = (c_1^1 - \sigma^{2,*}A_1^1 + \pi^{2,*}B_1^1)p^2p^1x^{1,*} + (c_1^2 - \sigma^{1,*}A_1^2 + \pi^{1,*}B_1)p^1p^2x^{2,*}, \tag{11}$$

where $c_1^1 = \{c_j : j \in \beta^1 \bigcap \overline{\beta}^2\}$, $c_1^2 = \{c_j : j \in \overline{\beta}^1 \bigcap \beta_2\}$, $A_1^1 = \{a_j : a_j$ is a column of A_1 and $j \in \beta^1 \bigcap \overline{\beta}^2\}$, $A_2^1 = \{a_j : a_j$ is a column of A_1 and $j \in \overline{\beta}^1 \bigcap \beta^2\}$, and B_1^2 and B_1^1 are columns of B_1 corresponding to A_1^1 and A_1^2, respectively.

Proof. First, we show that $(p^1\sigma^{1,*} + p^2\sigma^{2,*}; p^1\pi^{1,*}; p^2\pi^{2,*})$ is a feasible solution to the dual of SDLP2. We have $\pi^{2,*}A_2 \leq c_2$ and $\pi^{1,*}A_2 \leq c_2$, since $\pi^{1,*}$ and $\pi^{2,*}$ solve S1 and S2. This also yields $-\pi^{1,*}B_1 + \sigma^{1,*}A_1 \leq c_1$ and $-\pi^{2,*}B_1 + \sigma^{2,*}A_1 \leq c_2$, so

$$-(p^1\pi^{1,*}+p^2\pi^{2,*})B_1+(p^1\sigma^{1,*}+p^2\sigma^{2,*})A_1\leq c_1, \tag{12}$$

since p^1, $p^2\geq 0$ and $p^1+p^2=1$. Therefore, the solution is dual feasible, hence, by duality, we have $z^*\geq \underline{z}$.

Now, we consider $(p^1x^{1,*}+p^2x^{2,*};\tilde{x}_2^{1,*};\tilde{x}_2^{2,*})$ and observe that $A_1(p^1x^{1,*}+p^2x^{2,*})=b$, that, by definition, $\tilde{x}_2^{1,*}$ and $\tilde{x}_2^{2,*}$ satisfy the second set of inequalities in SDLP2, and, that, if $\tilde{x}_1^{2,*}$ and $\tilde{x}_2^{2,*}$ are not feasible, $\overline{z}=\infty$. Therefore, if $\tilde{x}_2^{1,*}\geq 0$ and $\tilde{x}_2^{2,*}\geq 0$, the solution is feasible and $z^*\leq \overline{z}$.

Therefore, $\underline{z}\leq z^*\leq \overline{z}$. For the expression of the duality gap, we observe that

$$\begin{aligned}
\underline{z} \quad &= (p^1\sigma^{1,*}+p^2\sigma^{2,*})(p^1A_1x^{1,*}+p^2A_1x^{2,*})\\
&+p^1\pi^{1,*}(A_2\tilde{x}_2^{1,*}-p^1B_1x_1^{1,*}-p^2B_1x_1^{2,*})\\
&+p^2\pi^{2,*}(A_2\tilde{x}_2^{2,*}-p^1B_1x_1^{1,*}-p^2B_1x_1^{2,*})\\
&\\
&=c_1^0(p^1x_1^{1,*}+p^2x_1^{2,*})+p^1c_1^1(p^1x_1^{1,*}+p^2x_1^{2,*})\\
&+p^2c_1^2(p^1x_1^{1,*}+p^2x_1^{2,*})+p^2\sigma^{2,*}(p^1A_1^1x_1^{1,*}+p^2A_1^1x_1^{2,*})\\
&+p^2\pi^{2,*}(-p^1B_1^1x_1^{1,*}-p^2B_1^1x_1^{2,*})+p^1\sigma^{1,*}(p^1A_1^2x_1^{1,*}+p^2A_1^2x_1^{2,*})\\
&+p^1\pi^{1,*}(-p^1B_1^2x_1^{1,*}-p^2B_1^2x_1^{2,*})+p^1c_2^1\tilde{x}_2^{1,*}\\
&+p^2c_2^2\tilde{x}_2^{2,*}
\end{aligned} \tag{13}$$

where $c_1^0=\{c_1(j):j\in\beta^1\bigcap\beta^2\}$. We then have

$$\begin{aligned}
\underline{z}=\overline{z}&-p^2c_1^1(p^1x_1^{1,*}+p^2x_1^{2,*})-p^1c_1^2(p^1x_1^{1,*}+p^2x_1^{2,*})\\
&+(p^2\sigma^{2,*}A_1^1-p^2\pi^{2,*}B_1^1+p^1\sigma^{1,*}A_1^2-p^1\pi^{1,*}B_1^2)(p^1x_1^{1,*}+p^2x_1^{2,*})
\end{aligned} \tag{14}$$

which, since $c_1^2\cdot x_1^{1,*}=0$ and $c_1^1\cdot x_1^{2,*}=0$, yields

$$\begin{aligned}
\underline{z}=\overline{z}&-(c_1^1-\sigma^{2,*}A_1^1+\pi^{2,*}B_1^1)(p^2p^1x_1^{1,*})\\
&-(c_1^2-\sigma^{1,*}A_1^2+\pi^{1,*}B_1^2)(p^1p^2x_1^{2,*})
\end{aligned} \tag{15}$$

Hence, the result in (11) follows. ∎

Measuring the duality gap in (11) is another way of finding the value of solving the stochastic problem directly. If the gap is small, then we can use the simple weighted average of the optimal scenario values without incurring a great penalty. A large gap signifies greater variance among the scenario solutions and should lead to a stochastic approach. A large difference can also result from infeasibility of the primal solutions since by definition in this case, $\overline{z} = \infty$. In some instances, the gap may be closed easily as we discuss below.

We first consider the case of $\beta = \beta^1 = \beta^2$ (same first period basic activities in S1 and S2), but where $|\beta| > m_1$, the rank of A_1. This situation, which we described in Chapter I, signifies that the same basic activities cannot exclusively be used to solve SDLP2. They will not span the row space of the matrix. If we have an inequality form, however, the weighted average of optimal first period values may be optimal in SDLP2.

We write the coefficient matrix of SDLP2 as

$$A = \begin{pmatrix} A_1^0 & A_1^1 & A_1^2 & A_1^N & & \\ -B_1^0 & -B_1^1 & -B_1^2 & -B_1^N & A_2^1 & A_2^{1,N} \\ -B_1^0 & -B_1^1 & -B_1^2 & -B_1^N & & A_2^2 \; A_2^{2,N} \end{pmatrix} \tag{16}$$

where $A_1^0 = (A_1(*,j) : j \in \beta^1 \bigcap \beta^2)$, A_1^1, A_1^2 are defined as in Lemma 2, $A_1^N = (A_1(*,j) : j \in \overline{\beta}^1 \bigcap \overline{\beta}^2)$, A_2^1, A_2^2 are the basic second period columns in S1 and S2, respectively, the columns of $A_2^{1,N}$ and $A_2^{2,N}$ are non-basic, and the B_j^i's are defined correspondingly.

Now, we assume that $\beta_1 = \beta_2$ and that we can replace the second equality in SDLP2 with an inequality. SDLP2 becomes

$$
\begin{array}{lccccccc}
\min & c_1^0 y_1^0 & +c_1^1 y_1^1 & +c_1^N y_1^N & +p^1 c_2^1 y_2^1 & +p^2 c_2^2 y_2^2 & +p^1 c_2^{1,N} y_2^{1,N} & +p^2 c_2^{2,N} y_2^{2,N} \\
\text{s.t.} & A_1^0 y_1^0 & +A_1^1 y_1^1 & +A_1^N y_1^N & & & & = b_1 \\
& -B_1^0 y_1^0 & -B_1^1 y_1^1 & -B_1^N y_1^N & +A_2^1 y_2^1 & & +A_2^{1,N} y_2^{1,N} & \geq \xi_2^1 \\
& -B_1^0 y_1^0 & -B_1^1 y_1^1 & -B_1^N y_1^N & & +A_2^2 y_2^2 & +A_2^{2,N} y_2^{2,N} & \geq \xi_2^2 \\
& & & & & & y_1, y_2 & \geq 0,
\end{array}
$$

(SDLP2′)

where the variables (y_1, y_2) replace the variables (x_1, x_2) in SDLP, so that we can compare their values. First, we define a solution to SDLP2′ by the following. Let a solution $(y_1; y_1^1, y_1^2)$ be

$$
(y_1^{0,*}; y_1^{1,*}; y_2^{1,*}) = \begin{pmatrix} A_1^0 & A_1^1 & \\ -B_1^0 & -B_1^1 & A_2^1 \end{pmatrix}^{-1} \begin{pmatrix} b_1 \\ \xi_2^1 \end{pmatrix}. \tag{17}
$$

This solution is the same as $(x_1^{1,*}, x_2^{1,*})$. Next, define

$$
\nu = \{i : x_1^{1,*}(i) \text{is basic in row } j \text{ of S1 for } j > m_1\}, \tag{18}
$$

where m_1 is the number of rows in A_1. Now, partition A_2^1 as

$$
A_2^{1,\nu} = (A_2^1(j,*) : j \in \nu)
$$

and

$$
A_2^{1,\bar{\nu}} = (A_2^1(j,*) : j \notin \nu). \tag{19}
$$

$A_2^{1,\bar{\nu}}$ is non–singular because the basis in (17) is non–singular.

We complete the definition of y^* by

$$
y_2^{2,\bar{\nu}^*} = (A_2^{1,\bar{\nu}})^{-1}(\xi_2^{2,\bar{\nu}} + B_1^{0,\bar{\nu}} y_1^{0,*} + B_1^{1,\bar{\nu}} y_1^{1,*})
$$

and

$$y_2^{2,\nu^*} = \xi_2^{2,\nu} + B_1^{0,\nu} y_1^{0,*} + B_1^{1,\nu} y_1^{1,*}. \tag{20}$$

Here, y_2^{2,ν^*} represents the slack variables. Now, we wish to show that y^* is an optimal solution to SDLP2'. We will have to restrict the solutions of S1 and S2 for this to be true. We first look at the dual of SDLP2'.

Let $\pi^{2,\nu} = (\pi_j^2 : j \in \nu) = 0$ and $\pi^{2,\overline{\nu}} = (\pi_j^2 : j \notin \nu) = p^2 c_2^2 (A_2^{2,\overline{\nu}})^{-1}$, so we have $\pi^2 A_2^2 = (p^2 c_2^2; 0)$ where the objective row coefficients of the slacks y_2^{2,ν^*} are 0.

Next, let

$$\pi^{1,\overline{\nu}} = p^1 c_2^1 (A_2^{1,\overline{\nu}})^{-1} - \pi^{1,\nu} (A_2^{1,\nu})(A_2^{1,\overline{\nu}})^{-1}, \tag{21}$$

and

$$\pi^{1,\nu} = (c_1 - \sigma A_1^{\nu} - B_1^{\overline{\nu}} p^1 c_2^1 (A_2^{1,\overline{\nu}})^{-1})(B_1^{\overline{\nu}} (A_2^{1,\overline{\nu}})^{-1} A_2^{1,\nu} - B_1^{\nu})^{-1},$$

where $B_1^{\nu} = (B^1(*, i) : i$ is basic for row j such that $j \in \nu)$ and $B_1^{\overline{\nu}}$, A_1^{ν} are defined accordingly. Now, let $\hat{B} = (B_1^{1,\overline{\nu}} (A_2^{1,\overline{\nu}})^{-1} A_2^{1,\nu} - B_1^{\nu})^{-1}$ and compute σ as

$$\begin{aligned} \sigma = [&c_1^{\overline{\nu}} + (B_1^{\nu} - B_1^{\overline{\nu}} (A_2^{1,\overline{\nu}})^{-1} A_2^{1,\nu})(\hat{B})^{-1} (c_1^{\nu} - B_1^{\nu} (A_2^{1,\overline{\nu}})^{-1} c_2^1) \\ &+ B_1^{\overline{\nu}} (A_2^{1,\overline{\nu}})^{-1} c_2^1] (A_2^{1,\overline{\nu}} + (B_1^{\nu} - B_1^{\overline{\nu}} (A_2^{1,\overline{\nu}})^{-1} A_2^{1,\nu})(\hat{B})^{-1} A_1^{\nu})^{-1}. \end{aligned} \tag{22}$$

We substitute for $\pi^{1,\nu}$ in (21) and find that, π_{OLD} and σ_{OLD}, the optimal dual prices in S1 (or S2, since the bases and objective functions are the same), satisfy

(i) $\pi^{2,\nu} = 0$,

(ii) $\pi^{2,\mathcal{V}} = p^2(\pi^{\mathcal{V}}_{OLD} + \pi^{\nu}_{OLD} A_2^{2,\nu}(A_2^{2,\mathcal{V}})^{-1})$,

(iii) $\pi^{\nu} = \pi^{\nu}_{OLD}$,

(iv) $\pi^{1,\mathcal{V}} = p^1(\pi^{\mathcal{V}}_{OLD} - (p^2/p^1)\pi^{\nu}_{OLD} A_2^{2,\nu}(A_2^{2,\mathcal{V}})^{-1}$,

and (v) $\sigma = \sigma_{OLD}$.

Hence, we have

$$\begin{aligned} -\pi^{2,\mathcal{V}}B_1 - \pi^{2,\nu}B_1 - \pi^{1,\mathcal{V}}B_1 - \pi^{1,\nu}B_1 + \sigma A_1 \\ -\pi^{\mathcal{V}}_{OLD}B_1 - \pi^{\nu}_{OLD}B_1 + \sigma_{OLD}A_1 \leq c_1, \end{aligned} \tag{23}$$

$$-\pi^{\mathcal{V}}_{OLD}B_1^{\mathcal{V}} - \pi^{\nu}_{OLD}B_1^{\mathcal{V}} + \sigma_{OLD}A_1^{\mathcal{V}} = c_1^{\mathcal{V}}, \tag{24}$$

and

$$-\pi^{\mathcal{V}}_{OLD}B_1^{\nu} - \pi^{\nu}_{OLD}B_1^{\nu} + \sigma_{OLD}A_1^{\nu} = c_1^{\nu}. \tag{25}$$

We also observe that

$$\begin{aligned} \pi^{1,\mathcal{V}}A_2^{1,\mathcal{V}} + \pi^{1,\nu}A_2^{1,\nu} &= p^1(\pi^{\mathcal{V}}_{OLD}A_2^{1,\mathcal{V}} + \pi^{\nu}_{OLD}A_2^{1,\nu}) \\ &\quad - p^2\pi^{\nu}_{OLD}A_2^{1,\nu} + p^2\pi^{\nu}_{OLD}A_2^{1,\nu} \\ &= p^1c_2^1, \end{aligned} \tag{26}$$

and that, similarly,

$$\pi^{2,\mathcal{V}}A_2^{2,\mathcal{V}} + \pi^{2,\nu}A_2^{2,\nu} = p^2c_2^2. \tag{27}$$

We need only that

$$\pi^{1,\mathcal{V}}A_2^{N,\mathcal{V}} + \pi^{1,\nu}A_2^{N,\nu} \leq p^1c_2^{1,N} \tag{28}$$

and

$$\pi^{2,\overline{\nu}} A_2^{N,\overline{\nu}} + \pi^{2,\nu} A_2^{N,\nu} \leq p^2 c_2^{2,N} \tag{29}$$

in order for feasible y^* to be optimal, since we have already shown complementarity in (24), (25), (26), and (27). (23) also shows dual feasibility for the first period constraints.

To ensure that (28) and (29) hold, we need to have the following equality satisfied:

$$\pi^{\nu}_{OLD} A_2^{2,\nu} (A_2^{2,\overline{\nu}})^{-1} A_2^{N,\overline{\nu}} = \pi^{\nu}_{OLD} A_2^{N,\nu}. \tag{30}$$

By the definition of π^1 and π^2, (30) implies (28), since

$$\begin{aligned} \pi^{1,\nu} A_2^{N,\overline{\nu}} + \pi^{1,\nu} A_2^{N,\nu} &= p^1(\pi^{\overline{\nu}}_{OLD} A_2^{N,\overline{\nu}} + \pi^{\nu}_{OLD} A_2^{N,\nu}) \\ &\quad + p^2(\pi^{\nu}_{OLD} A_2^{N,\nu} - \pi^{\nu}_{OLD} A_2^{2,\nu} (A_2^{2,\overline{\nu}})^{-1} A_2^{N,\overline{\nu}}) \\ &\leq p^1 c_1^N, \end{aligned}$$

and (29) holds by

$$\begin{aligned} \pi^{2,\overline{\nu}} A_2^{N,\overline{\nu}} + \pi^{2,\nu} A_2^{N,\nu} &= p^2(\pi^{\overline{\nu}}_{OLD} A_2^{N,\overline{\nu}} + (\pi^{\nu}_{OLD} A_2^{2,\nu} (A_2^{2,\overline{\nu}})^{-1} A_2^{N,\overline{\nu}})) \\ &= p^2(\pi^{\overline{\nu}}_{OLD} A_2^{N,\overline{\nu}} + \pi^{\nu}_{OLD} A_2^{N,\nu}) \\ &\leq p^2 c_2^N. \end{aligned}$$

Hence, we have shown the following theorem.

Theorem 4. *For y^* as defined in (17) and (20), if y^* is a feasible solution to SDLP2$'$ and the optimal dual prices in S1 and S2 are such that (30) is satisfied, y^* is an optimal solution to SDLP2$'$.*

The restriction of the prices in (30) guarantees optimality, but (28) and (29) may be true even if (30) does not hold. In examining a problem of this

type, if (30) fails, one may want to carry out the additional computations in (28) and (29) before solving the stochastic problem.

The difficulty in finding optimality conditions for combining scenarios for even a simple problem such as SDLP2', shows the importance of the stochastic solution. The conditions in (30) can be generalized to allow for penalties in satisfying inequalities in the second period, but the results are more restrictive and direct computation of the dual feasibility conditions becomes more efficient than checking additional inequalities. Our development leads to the following method for combining scenarios.

(A.) Combine the first period scenario solutions, $x_1^1, \ldots, x_1^k$, by a simple weighted average, $y_1^* = \sum_{i=1}^k p^i x_1^i$.

(B.) Follow the branch of worst cases (what we shall call the *catastrophe* branch), that is, the set $\{\xi_t^*\}$ where $\xi_t^* = (\xi_t^*(1), \xi_t^*(2), \ldots, \xi_t^*(m_t))$ such that $\xi_t^*(i) = \sup_j \xi_t^j(i)$ for $i = 1, 2, \ldots, m_t$. Using these right-hand side values, we have that, if

$$-B_{t-1}y_{t-1}^* + A_t y_t^* \geq \xi_t^*,$$

then

$$-B_{t-1}y_{t-1}^* + A_t y_t^* \geq \xi_t^j$$

for all j. This procedure guarantees primal feasibility for y_t^*. We use the non-singular blocks from this scenario to determine new values based on y_1^*.

(C.) Use similar blocks for the other branches and maintain primal feasibility (by, perhaps, paying penalties).

(D.) Compute the dual prices and check for closure of the duality gap.

This approach would be most successful when the different scenarios have nearly identical bases. In these cases, the noncomplementarity would exist in only a few terms. The difficulty of using these optimal linear program solutions, however, is that they all correspond to extreme point solutions and may include very different sets of basic activities. This property could make their combination in a stochastic program most difficult. In the next section, we present small examples of this occurrence.

5. Examples

The extreme point properties of the basis in a linear program are crucial in understanding stochastic program solutions. Critical values of the parameters limit the use of different scenarios. In some cases, the implementation of any deterministic solution may lead to heavy penalties relative to the solution of the stochastic program. One example of this occurrence is the following linear program :

$$
\begin{array}{lrrrrrrl}
\min z = & x_1 & +4x_2 & + \mathrm{E}_\xi[\min & y_1 & +10y_2 & \mid x_1 \text{ and } x_2] & \\
\text{subject to} & x_1 & +x_2 & & & & & = 1 \\
& -x_1 & +2x_2 & +y_1 & +y_2 & & & = \xi \\
& & & & & 0 \le y_1 & & \le 2, \\
& & & & & x_1, x_2, y_2 & & \ge 0, \\
& & & & & & \xi \text{ is Uniform } [0,4]. & \\
& & & & & & & (EX1)
\end{array}
$$

We solve EX1 for $\overline{\xi} = (1,3)$ and find the expected value of using the optimal decisions for these scenarios as

$$\overline{z}_{\overline{\xi}} = \mathrm{E}[z_{\overline{\xi}}(\xi)], \tag{31}$$

where

$$z_{\bar{\xi}}(\xi) = c_1 x(\bar{\xi}) + [\min_y c_2 y | x(\bar{\xi}), \xi]. \tag{32}$$

We define

$$z_I \equiv z(1) \tag{33}$$

and

$$z_{II} \equiv z(3). \tag{34}$$

We want to consider also the perfect information solution

$$z_p \equiv \mathrm{E}_\xi[\min c_1 \cdot x + c_2 \cdot y | \xi], \tag{35}$$

and the stochastic solution, z_s, where we allow $\xi = 1$ or 3 with equal probability.

The function $z_{\bar{\xi}}(\xi)$ for z_I, z_{II}, z_s, and z_p appears in Figure 2. We observe that the optimal basis changes as ξ ranges over [0,4].

For $\xi \leq 1$, x_1 only is in the optimal basic set of variables, for $1 \leq \xi \leq 3$, $\{x_1, x_2\}$ is optimal, for $2 \leq \xi \leq 3$, an alternative optimal set includes x_2 alone, and, for $3 \leq \xi \leq 4$, the only optimal first period activity is x_2. In the stochastic solution, x_1 and x_2 must be in the optimal basis, reducing the expected loss relative to the perfect information solution. We find this from the figure as

$$z_I - z_p = 10.25, \tag{36}$$

$$z_{II} - z_p = 5.50, \text{ and} \tag{37}$$

$$\bar{z}_s - \bar{z}_p = 3.75. \tag{38}$$

The losses in (36), (37), and (38) associated with this problem demonstrate the usefulness of the stochastic solution. By assuming any deterministic value for the right–hand side, a large loss may result. The simple stochastic formulation with two possible ξ values, however, reduces the risk of this situation and enables us to approach the perfect information solution. The stochastic solution, therefore, lends resilience to the result. It provides a rationale for hedging strategies.

To demonstrate further the sensitivity of models to uncertainty, we return to the exhaustible resource model of Chapter I, (I.29). We stated that this model had an optimal deterministic solution, but, by adding an uncertain return from investment in exploration, we again arrive at a situation, in which, every deterministic solution will be associated with losses relative to

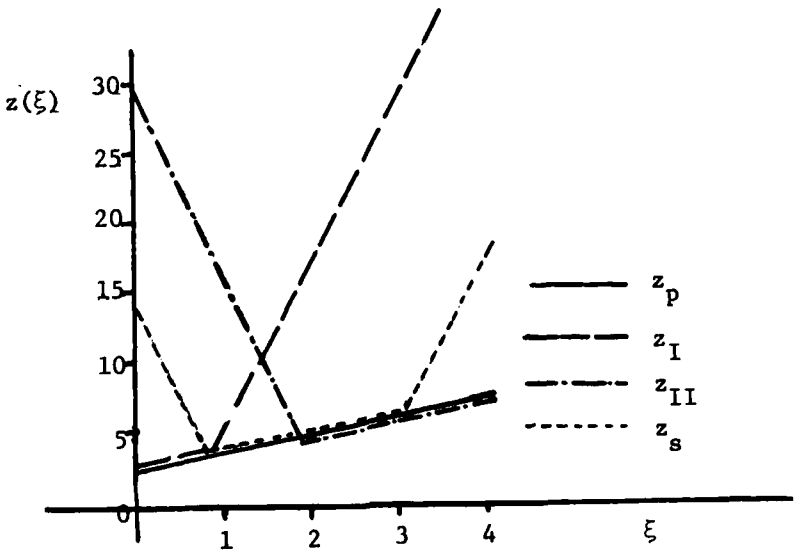


Figure 2. Example costs.

the stochastic solution.

The modified problem is

$$\min \sum_{i=1}^{m} c_i u_{i,0} + \sum_{i=1}^{n} k_i y_{i,0} + \sum_{t=1}^{T} \beta^t \sum_{i=1}^{m} \sum_{j=1}^{k} p^j [c_i u_{i,t}^j + k_i y_{i,t}^j]$$

$$\begin{aligned}
\text{subject to} \quad \xi^0 + u_{i,0} &\leq R_{i,0}; \text{ for } i = 1, \ldots, m, \\
\textstyle\sum_{i=1}^{m} u_{i,0} &\geq D_0; \\
x_{i,t}^j + u_{i,t}^j &\leq R_{i,j}; \text{ for all } i, j, \text{ and } t, \\
\textstyle\sum_{i=1}^{m} u_{i,t}^j &\geq D_t; \text{ for all } j \text{ and } t, \\
x_{i,t-1}^j + u_{i,t-1}^j \quad -\alpha_{i,t}^j y_{i,t-1}^j &= x_{i,t}^j; \text{ for all } i, j, \text{ and t}, \\
\\
x_{i,t}^j \geq 0, u_{i,t}^j &\geq 0, \text{ for all } i, j, \text{ and } t,
\end{aligned} \tag{ERM}$$

where $R_{i,j}$ D_t, $x_{i,t}^j$, and $u_{i,t}^j$ are as defined in (1.29). $y_{i,t}^j$ represents the amount invested at time t under scenario j, and $\alpha_{i,t}^j$ is the return for investment.

ERM includes deterministic investment, but we can formulate an associated stochastic model by allowing $\alpha_{i,t}^j$ to take on several values, $\alpha_{i,t}^{j,1}, \ldots, \alpha_{i,t}^{j,l}$, and restricting investment in $y_{i,t}^j$ to be only one of the $\alpha_{i,t}^{j,q}$ for each scenario. By doing this, we maintain the random variable in the right–hand side as a constraint on $y_{i,t}^j$ for the different scenarios.

By solving ERM for fixed values, the investment decision may quickly swing from one resource to another as a different extreme point in the linear program becomes optimal. The stochastic solution has more basic activities, allowing for hedging against the different possible environments that one may face. To show this property, we consider a two period case of ERM with three alternative resources. We will call them oil, solar power, and some other high

cost backstop technology. We use the following inputs in Table 1.

Resources

	Cost(c_i)	Availability(R_i)
Oil	5	25
Solar	10	10
Backstop	16	∞

Investment

	Cost (k_i)	Return(α_i)
Oil 1	1	1
Oil 2	1	0.1
Solar	1	1

Demand

Period 1	15
Period 2	25

Probabilities

Scenario 1	0.5
Scenario 2	0.5

Discount Rate

$\beta = .80$

Table 1. Two period model inputs.

The only uncertainty in this model is on the return for oil exploration. Investment in solar power can be interpreted as the relatively certain amount of capacity increase from investment. Each unit invested in oil in this model, however, results in either a full unit increase or a tenth of a unit increase in oil availability according to α. These two scenarios are assumed to occur with equal probability.

The deterministic models for "bad luck" ($\alpha = .1$), "good luck"($\alpha = 1$),

and "myopic"(solving only given the first period availabilities and demands) were solved and compared to the stochastic program solution, in which both values of α were considered simultaneously. The results were :

Model	Expected Cost
Myopic	275
Good Luck	239
Bad Luck	245
Stochastic	231

Here, the stochastic solution represents a savings over the deterministic models because its solution involved investment in both oil and solar technologies, while the deterministic scenarios allowed for investment in only one. The relative savings would also increase with a higher cost backstop. It is interesting to note also that, with the addition of investment in exploration, the myopic solution is now far from optimal.

These examples have shown that models can be very sensitive to future uncertainties. The exhaustible resource model demonstrates the possibility of sharp changes in decision–making from near–term uncertainties. We, therefore, want to examine models with a strategy that considers their sensitivity to uncertain parameters. We discuss a general method for dealing with these problems in the following section.

6. A General Strategy

Problems modeled as dynamic linear programs often involve many uncertain assumptions about the parameters involved. In this case, a solution to the stochastic program is desired, but it may be quite costly. We have presented methods for checking whether deterministic solutions may be used.

The strategy resulting from our development in Chapters 1 and 2 follows.

I. Solve the expected value problem and check for basis feasibility and a possible optimal deterministic solution as in Chapter I.

II. If I fails, use the properties presented in 2.3 to bound the VSS and EVPI.

III. If the VSS bound is large, solve different deterministic scenarios and look for a dominant basis. Then, attempt to close the duality gap by combining scenarios.

IV. If the gap persists and if no single basis is indicated, proceed to solve SDLP.

This procedure outlines how we would evaluate the worth of solving successively more complicated problems. As our examples have shown, we may find that the stochastic program is worth solving, and that its solution may result in a substantial savings. We might attempt to solve such problems directly by brute–force methods. Alternatively, the next three chapters present methods that take advantage of the structure of SDLP to reduce the size of these possibly very large linear programs.

CHAPTER III

A Nested Decomposition Algorithm for SDLP

1. Introduction

Stochastic dynamic linear programs have natural subdivisions corresponding to the separate decisions made in each period under various scenarios. These divisions give SDLP a special structure that can lead to improved efficiency in its solution. In the next three chapters, we present three methods employing distinct optimization techniques, each of which exploits the program's structure. The structure is essential for our development. It is the basis of each technique: decomposition, partitioning, and basis factorization.

These methods have been used extensively in large–scale structured programming, but their application in solving stochastic programs has been limited. In our presentation, we demonstrate how these techniques can be applied to the stochastic version of the multi–stage linear program. We will then show that our algorithms may yield substantial savings over straightforward, "brute–force" techniques by using the program structure effectively in reducing the computational cost.

The first method we present is called a *nested decomposition algorithm*

for SDLP or *NDSDLP*, because it decomposes a large problem into successively finer subproblems.

The basic principles of optimization that we employ are *outer linearization* and *inner linearization*, as described in Geoffrion [24]. Through outer linearization we optimize over a larger convex region than is feasible and then, by increasing the restrictions obtained from the subproblems, approach optimality within the true feasible region. To perform inner linearization one optimizes over successive subregions of the feasible region and approaches thereby a global optimum.

The method below applies outer linearization to the primal problem of SDLP. It can also be viewed as applying inner linearization to the dual problem. We then have a master subproblem relationship in the primal as in Benders' method [9] or in the dual as in Dantzig-Wolfe decomposition [19]. We also follow a procedure of passing between periods that is similar to the nested decomposition of primal inner linearization as in Glassey [29] and Ho and Manne [32]. The relationship between the two methods is well–known as Kallio and Porteus showed in [38].

We begin our presentation of the nested decomposition algorithm in Section 2, by examining some properties of objective functions and showing the basic master– subproblem relationship for SDLP under the outer linearization scheme. Section 3 presents the analogous development for inner linearization, or Dantzig-Wolfe decomposition. We follow this in Section 4 with a description of a fundamental problem inherent in SDLP, degeneracy. We present suggestions for its resolution and some of its special difficulties in the stochastic framework. Lastly, in Section 5, we present the full algorithm and its strategy in passing through the scenarios.

2. The Master-Subproblem Relationship

The decomposition algorithm we discuss below relies on certain fundamental properties of the multi-stage program under uncertainty. These properties concern the convexity of the objective function and the representation of the solution set as a convex polyhedron. Wets first observed these attributes in [61] and [62]. These are reviewed below.

We begin by formulating the equivalent convex program to SDLP. We then use this formulation to examine how we can find the set of linear constraints that constitute the solution set. This procedure involves inducing feasibility in the subproblems and finding the conditions for optimality. The section concludes with an explanation of how the cutting planes for these operations are constructed.

Every stochastic dynamic linear program (with continuous or discrete distribution of the random variables) is equivalent to a convex program with linear constraints, as we state in the following theorem of Wets:

Theorem 1 *The stochastic dynamic linear program SDLP, as defined in Chapter 1, is equivalent to a program of the following form:*

$$\begin{aligned} &\min c_1 x_1 + Q_1(x_1) \\ &\textit{subject to} \\ &A_1 x_1 = \xi_1 \quad , \\ &x_1 \in D_1, \\ &x_1 \geq 0, \end{aligned} \tag{ECP}$$

where $Q_1(x_1)$ is a convex function and D_1 is a convex polyhedron.

Proof. See Wets [62]. ∎

The $Q_1(x_1)$ of Theorem 1 is defined as

$$Q_1(x_1) = E_{\xi 2}[Q_1(x_2, \xi_2)] \tag{1}$$

where

$$Q_1(x_2, \xi_2) = \{\min[c_2 x_2 + Q_2(x_2)] | A_2 x_2 = \xi_2 + B_1 x_1, x_2 \in D_2, x_2 \geq 0\}. \tag{2}$$

The subsequent $Q_t(x)$ are defined iteratively. We can solve SDLP by repeated solutions of these convex programs, but the objective function and solution set may be hard to find. We present below methods for finding $Q_t(x_t)$ and D_t without explicitly determining the functions.

ECP above is called the *equivalent convex program* of the stochastic program. We note that this theorem holds when ξ has a continuous or discrete distribution.

Before we proceed with this development, we further note that, since every period of SDLP corresponds to an identically formulated optimization problem, we can form a subproblem in every period, t, and for every scenario, j, that is similar to ECP. Let $\hat{j}$ be the immediate ancestor of j, and $\zeta_t^j = B_{t-1} x_{t-1}^{\hat{j}} + \xi_t^j$, we have

$$z_t^j = \min c_t x_t^j + Q_t(x_t^j)$$

subject to

$$\begin{aligned} A_t x_t^j &= \zeta_t^j, \\ D_t^j x_t^j &\geq \lambda_t^j. \\ x_t^j &\geq 0, \end{aligned} \qquad (ECP(t, j))$$

where we have written the convex polyhedron, D, as $D_t^j x_t^j \geq \lambda_t^j$, and $Q_t(x_t^j)$ is defined as in (1).

ECP(t, j) is used in the subsequent analysis to develop the master subproblem relationships that are encountered in the nested decomposition algorithm. We first want to find a method for constructing D_t^j so that x_t^j will be feasible for its descendant scenarios. Let $\bar{j}$ be a descendant of j in period $t+1$ for a given solution $x_t^{j,0}$ of ECP(t, j). We have

$$\min z_{t+1}^{\bar{j}} = c_{t+1} x_{t+1}^{\bar{j}} + Q(x_{t+1}^{\bar{j}})$$

subject to

$$\begin{aligned} A_{t+1} x_{t+1}^{\bar{j}} &= \xi_{t+1}^{\bar{j}} + B_t x_t^{j,0}, \\ D_{t+1}^{\bar{j}} x_{t+1}^{\bar{j}} &\geq \lambda_{t+1}^{\bar{j}}. \qquad (ECP(T+1, \bar{j})) \\ x_{t+1}^{\bar{j}} &\geq 0, \end{aligned}$$

Now, if ECP$(t+1, \bar{j})$ has no feasible solution, then by Farkas's lemma, there exists a vector $\left[{}_1\sigma_{t+1}^{\bar{j}}, {}_2\sigma_{t+1}^{\bar{j}} \right]$ such that

$${}_1\sigma_{t+1}^{\bar{j}} A_{t+1} + {}_2\sigma_{t+1}^{j} D_{t+1}^{\bar{j}} \leq 0, \tag{3}$$

$$(-{}_2\sigma_{t+1}^{j}) \leq 0, \tag{4}$$

and

$${}_1\sigma_{t+1}^{\bar{j}} (\xi_{t+1}^{\bar{j}} + B_t x_t^{j,0}) + {}_2\sigma_{t+1}^{\bar{j}} \lambda_{t+1}^{\bar{j}} > 0. \tag{5}$$

So, in order for ECP$(t+1, \bar{j})$ to have a feasible solution x_t^j must be chosen such that

$$-({}_1\sigma_{t+1}^{j} B_t) x_t^j \geq + {}_1\sigma_{t+1}^{\bar{j}} \xi_{t+1}^{\bar{j}} + {}_2\sigma_{t+1}^{\bar{j}} \lambda_{t+1}^{\bar{j}}. \tag{6}$$

This implies the following lemma:

Lemma 1. *For every descendant scenario of j, $\bar{j}$, in period $t+1$ and for all ${}_1\sigma_{t+1}^{\bar{j}}$ and ${}_2\sigma_{t+1}^{\bar{j}}$ satisfying (3) and (4) if x_t^j is feasible in SDLP, then it satisfies the inequality (6).*

Using this result, we can add (6) as an additional constraint to ECP(t, j). We repeat this for each descendant $\bar{j}$ of j. We solve ECP(t, j) again after all of these *cuts* have been added and proceed downward again with a new value, $x_t^{j,1}$. In this manner, we iteratively construct the constraint set for x_t^j. From now on, for clarity of exposition, we will write the equations of the form (6)as

$$-(\sigma_{t+1}^{\bar{j},k} B_t) x_t^j \geq \sigma_{t+1}^{\bar{j},k}(\hat{\xi}_{t+1}^{\bar{j}}), \tag{7}$$

where $k = 1, \ldots, p$.

We know how to find the constraint set of ECP(t, j) and we must now find a method of constructing the convex function, $Q(x_t^j)$. To do this, we first observe that ECP(t, j) is equivalent to

$$\min x_t^j = c_t x_t^j + \theta_t^j \tag{8}$$

subject to

$$\begin{aligned} A_t x_t^j &= \xi_t^j, \\ -(\sigma_{t+1}^{\bar{j},k} B_t) x_t^j &\geq \sigma_{t+1}^{\bar{j},k}(\hat{\xi}_{t+1}^{\bar{j}}); k = 1, \ldots, p, \\ Q(x_t^j) &\geq \theta_t^j, \\ x_t^j &\geq 0. \end{aligned}$$

With this formulation, we consider that $x_t^{j,0}$ is again a solution to ECP(t, j) and that ECP$(t+1, \bar{j})$ is feasible for all $\bar{j}$. Next let

$$\pi(x_t^{j,0}), \bar{j} = (\pi_1(x_t^{j,0}, \bar{j}); \pi_2(x_t^{j,0}, \bar{j})) \tag{9}$$

be the optimal dual prices vector for each ECP$(t+1, \bar{j})$, given $x_t^{j,0}$ a solution to ECP(t, j), and define

$$\begin{aligned} \pi_1(x_t^{j,0}) &= \mathrm{E}_{\xi_{t+1}}[\pi_1(x_t^{j,0}, \bar{j})], \\ \rho_1(x_t^{j,0}) &= \mathrm{E}_{\xi_{t+1}}[\pi_1(x_t^{j,0}, \bar{j}) \cdot (\xi_{t+1}^{\bar{j}})] \end{aligned} \tag{10}$$

and

$$\rho_2(x_t^{j,0}) = \mathrm{E}_{\xi_{t+1}}[\sum_{k=1}^{p} \pi_2^k(x_t^{j,0}, \bar{j}) \cdot (\sigma_{t+1}^{\bar{j},k}(\tilde{\xi}_{t+2}^{\bar{j}})]. \tag{11}$$

Now, for $\pi(x_{t+1}^j, \bar{j})$ optimal in ECP$(t+1, \bar{j})$ for any x_{t+1}^j, the following inequality must hold

$$\pi_1(x_t^j, \bar{j})(\xi_{t+1}^{\bar{j}} + B_t x_t^j) + \sum_{k=1}^{p} \pi_2^k(x_t^j, \bar{j})\sigma_{t+2}^{\bar{j},k}(\tilde{\xi}_{t;2}^{\bar{j}}) \geq$$

$$\pi_1(x_t^{j,0}, \bar{j})(\xi_{t+1}^{\bar{j}}; B_t x_t^j) + \sum_{k=1}^{p} \pi_2^k(x_t^j, \bar{j})\sigma_{t+2}^{j,k}(\xi_{t+2}^{\bar{j}}, \tag{12}$$

and, since $\pi(x_t^j, \bar{j})$ is optimal, we have

$$Q(x_t^j, \bar{j}) = \pi_1(x_t^j, \bar{j})(\xi_{t+1}^{\bar{j}}) + B_t x_t^j) + \sum_{k=1}^{p} \pi_2^k(x_t^j, \bar{j}) \cdot \sigma_{t+1}^{\bar{j},k}(\hat{\xi}_{t+1}^{\bar{j}}). \tag{13}$$

Therefore, by taking expectations,

$$Q(x_t^j) \geq \rho_1(x_t^{j,0}) + \rho_2(x_t^{j,0}) + \pi_1(x_t^{j,0}) \cdot B_t x_t^j. \tag{14}$$

Letting $\rho(x_t^{j,0}) = \rho_1(x_t^{j,0}) + \rho_2(x_t^{j,0})$ in (14), we have the following lemma.

Lemma 2. *If (x_t^j, θ_t^j) is a feasible solution to ECP(t, j) written as in (8), then $\theta_t^j \geq \rho(x_t^{j,0}) + (\pi_1(x_t^{j,0})B_t)x_t^j$.*

This lemma enables us to form additional constraints, as

$$(\pi_1^l \cdot B_t)x_t^j + \theta_t^j \geq \rho^l \tag{15}$$

for successive l, where $\pi_1^l = \pi_1(x_t^{j,l})$ and $\rho^l = \rho(x_t^{j,l})$. These cuts are then also added to ECP(t, j) whenever we find that a solution $(x_t^{j,l}, \theta_t^{j,l})$ is such

that

$$\theta_t^{j,\ell} < \rho^\ell + \pi_1^\ell B_t x_t^{j,\ell}. \tag{16}$$

When (15) is solved for all π_1^l, and ρ^l, then we have achieved master-subproblem optimality between ECP(t,j) and its descendants, ECP$(t,\bar{j})$.

We have shown how the master problems and subproblems can be constructed in SDLP by using the fundamental results in Lemma 1 and Lemma 2. We will present below the basic algorithm for finding master-suboptimality. This algorithm follows closely from Benders [9] and has been presented in the two-stage case by Van Slyke and Wets [56]. It is an outer linearization scheme because the feasible regions D_t^j and convex functions Q_t^j are successively approximated by the inequalities in (7) and (15). We call this procedure OLSDLP, for outer linearization of SDLP. This exposition includes Step 2^l, the case of an unbounded solution in ECP(t,j). The justification for this procedure can be found in Van Slyke and Wets [56] for a deterministic problem. We omit details here, because, in general, we will use the algorithm with upper bounded variables and no unbounded solution will be possible.

OLSDLP

Step 1. Solve the current form of ECP(t,j), using Phase I and Phase II of the simplex method:

$$\begin{gathered}
\min c_t x_t^j + \theta_t^j \\
\text{subject to} \\
A_t x_t^j = \xi_t^j, \\
-(\sigma_{t+1}^{\bar{j},k} B_t) x_t^j \geq \sigma_{t+1}^{\bar{j},k}(\hat{\xi}_{t+1}^{\bar{j}}), k = 1,\ldots,p, \\
-(\pi_1^\ell \cdot B_t) x_t^j + \theta_t^j \geq \rho^\ell, \ell = 1,\ldots,q, \\
x_t^j \geq 0,
\end{gathered} \tag{17}$$

where we set $p = q = 0$ initially and let $\theta_t^j = -\infty$, if $q = 0$. If (17) is infeasible, stop. If (17) is feasible and unbounded, go to Step 2'. If (17) is feasible and bounded, go to Step 2.

Step 2. For $(x_t^{j,i}, \theta_t^{j,i})$ a solution of (17), solve the Phase 1 problem of ECP$(t+1, \bar{j})$:

$$\min \omega^{\bar{j}} = ev + eu^+$$

subject to

$$\begin{aligned} A_{t+1}x_{t+1}^{\bar{j}} + Iv &= \xi_{t+1}^{\bar{j}} + B_t x_t^{j,i}, \\ D_{t+1}^{\bar{j}} x_{t+1}^{\bar{j}} + Iu^+ - Iu^- &= \lambda_{t+1}^{\bar{j}} \\ x_{t+1}^{\bar{j}}, v, u^+, u^- &\geq 0. \end{aligned} \tag{18}$$

For each $\bar{j}$, such that $\omega^{\bar{j}} > 0$ in (18), use the resultant multipliers to build a cut of the form in (7). Add these cuts to (17) and increment p. If $\omega^{\bar{j}} > 0$ for any $\bar{j}$, go to Step 1; otherwise, go to Step 3.

Step 2'. From (17), we obtain an unbounded ray, $x_t^{j,l^o} + \lambda y_t^{j,l^o}$, for $\lambda \geq 0$. Now, solve (18) for each $\bar{j}$, but replace $\xi_{t+a}^{\bar{j}} + B_t x_t^{j,l}$ by $B_t y_t^{\bar{j},l^o}$. If $\omega^{\bar{j}} > 0$, for any $\bar{j}$, add cuts as in (7) and return to Step 1. If $\omega^{\bar{j}} = 0$ for all $\bar{j}$, solve ECP$(t+1, \bar{j})$ for all $\bar{j}$ with the same replacement. Let $\bar{z}^l(t+1, \bar{j})$ be the expected value of the objective functions and compute π^l and ρ^l.

Next, solve (18) with $x_t^{j,l}$ for each $\bar{j}$. If $\omega^{\bar{j}} > 0$ for any $\bar{j}$, add the feasibility *cuts* and return to Step 1. If $\omega^{\bar{j}} = 0$ for all $\bar{j}$, then we check if $c_t y_t^{j,l} + \bar{z}^l(t+1, \bar{j}) < 0$. If so, the objective function is unbounded, stop. If $c_t y_t^{j,l} + \bar{z}^l(t+1, \bar{j}) \geq 0$, then we must eliminate $y_t^{\bar{j},l}$ as a feasible direction. We do this by using the π^l and ρ^l found above in forming a constraint of the form (15) and adding it to (17). In this case, we return to Step 1.

Step 3. Solve ECP$(t+1, \bar{j})$ for each $\bar{j}$. Compute $\bar{z}^{\bar{j},i}_{t+1} = E_{\xi_{t+1}}\left[z^{\bar{j}}_{t+1}(x^j_{t+1})\right]$, π^i_1, and ρ^i. If $\bar{z}^{\bar{j},i}_{t+1} \leq \theta^i$, stop. ECP$(t, j)$ is solved. Otherwise, generate a cut of the form (15) add it to (17), and return to Step 1.

This algorithm terminates in a finite number of steps as we state in the following theorem.

Theorem 2. *The algorithm, OLSDLP, for finding the solution of ECP(t, j), results in either an infeasibility criterion, an unbounded solution, or an optimal solution in a finite number of steps.*

Proof. Every iteration of the algorithm results in the addition of a constraint of the form (7) or (15) to the optimization in (17). Since there are a finite number of bases for each ECP$(t+1, \bar{j})$, the number of these constraints is finite. They also cannot be repeated since $x^{j,i}_{t+1}$ would already have had to satisfy that constraint. Therefore, the algorithm terminates after a finite number of steps. ∎

The finiteness of OLSDLP can also be maintained if we delete the cuts in (17) that are slack after each iteration. This is true because the objective function in (17) is monotonically decreasing as new cuts are introduced. We also need only keep at most $m(t+1)+1$ cuts because the solution of ECP(t, j) has at most $m(t) + m(t+1)+1$ basic variables, as Murty showed in [45]. We will return to this important property in more detail in Chapter V in our discussion of the local basis method for SDLP.

We have seen how a master problem at period t in SDLP can be solved by outer linearization using subproblems at period $t+1$. As we stated above, this development is completely analogous to applying inner linearization to the

dual. In the next section, we demonstrate the relationship between OLSDLP and a Dantzig-Wolfe decomposition form of inner linearization.

3. The Relationship to Dantzig-Wolfe Decomposition

Dantzig and Madansky [18] in their fundamental paper on programming under uncertainty first proposed that two-stage stochastic linear programs could be solved by applying Dantzig-Wolfe decomposition to the dual of the stochastic linear program. In the context of our development here, we want to apply this decomposition to the linear subproblem of SDLP at period t and scenario j. We call this program LP(t,j). (Note that in this case we must assume that ξ_{t+1} has a discrete distribution as in SDLP.) The problem we address is then

$$\min z_t^j = c_t x_t^j + p^1 c_{t+1} x_{t+1}^{\bar{j},1} + \cdots + p^k c_{t+1} x_{t+1}^{\bar{j},k}$$

subject to

$$\begin{aligned} A_t x_t^j &= \xi_t^j + B_{t-1} x_{t-1}^{\hat{j}} \\ -B_t x_t^j + A_{t+1} x_{t+1}^{\bar{j},1} &= \xi_{t+1}^{\bar{j},1} \\ -B_t x_t^j + A_{t+1} x_{t+1}^{\bar{j},k} &= \xi_{t+1}^{\bar{j},k} \\ x_t^j, x_{t+1}^{\bar{j},i} &\geq 0, \quad i = 1, \ldots, k. \end{aligned}$$

This is the program for one section of SDLP. We have removed the constraints before period t and after $t+1$.

The dual of LP(t,j) can be written as

$$\max u(\xi_t^j + B_{t-1} x_{t-1}^{\hat{j}}) + p^1 v^1 \xi_{t+1}^{\bar{j},1} + \cdots + p^k v^k \xi_{t+1}^{\bar{j},k}$$

subject to

$$\begin{aligned} uA_t - p^1 v^1 B_t \quad \cdots \quad & p^k v^k B_t \leq c_t, \\ & v^1 A_{t+1} \leq c_{t+1}, \; i = 1, \ldots, k. \end{aligned} \qquad (DP(t,j)$$

Next, let $\{\pi^l\} = \{(\pi^l_1, \ldots, \pi^l_k)\}$, where $l = 1, \ldots, q$, be the set of all possible combinations of k extreme points of $P = \{\pi | \pi A_{t+1} \leq c_{t+1}\}$, and let $\{\sigma^m\}$, where $m = 1, \ldots, p$, be the set of extreme rays of P. Define also $\bar{\pi}^l = \sum_{i=1}^k p^i \pi^l_i$, $\bar{p}^l = \sum_{i=1}^k p^i \pi^l_i \xi_t^{\hat{j},i}$ and $\gamma^{m,i} = \sigma^m \xi_t^{\hat{j},i}$. DP$(j,t)$ can be written in Dantzig-Wolfe decomposition form as:

$$\max u(\xi_t^j + B_{t-1} x_{t-1}^{\hat{j}}) + \sum_{\ell=1}^{q} \lambda^\ell \bar{p}^\ell + \sum_{m=1}^{p} \sum_{i=1}^{k} \nu^{m,i} p^i \gamma^{m,i}$$

subject to

$$\begin{aligned} uA_t - \sum_{\ell=1}^{q} \lambda^\ell \bar{\pi}^\ell B_t - \sum_{m=1}^{p} \sum_{i=1}^{k} \mu^{m,i} p^i \sigma^m B_t &\leq c_t, \\ \sum_{\ell=1}^{q} \lambda^\ell &= 1, \\ \lambda^\ell \geq 0;\ \mu^{m,i} \geq 0, \quad &\text{for all } i, l, \text{ and } m. \end{aligned}$$

(DWD(t,j))

Now, we take the dual of DW(t,j) and use x_t^j and θ as multipliers. The result is

$$\min c_t x_t^j + \theta$$

subject to

$$\begin{aligned} A_t x_t^j &= \xi_t^j + B_{t-1} x_{t-1}^{\hat{j}}; \\ (-\sigma^m B_t) x_t^j &\geq \sigma^m \xi^{j,i}; \quad i = 1, \ldots, k;\ m = 1, \ldots, p; \\ (-\bar{\pi}^\ell B_t) x_t^j + \theta &\geq \bar{p}^\ell; \quad \ell = 1, \ldots, q; \\ x_t^j &\geq 0. \end{aligned} \qquad (DWD(t,j))$$

DWD(t,j) has the same form as (17) except that we have not included the extra constraints that enter into the subproblems of ECP$(t+1, \hat{j})$. The feasibility criteria correspond to the subproblems' proposing an extreme ray to the master problem in DW(t,j), and the optimality cuts on θ correspond

to extreme point proposals. Optimality in the outer linearization corresponds then to the absence of better proposals from the subproblems in the Dantzig-Wolfe approach. We state these results in the following lemma:

Lemma 3. *The outer linearization of the primal problem ECP(t, j) in (17) is equivalent to solving the dual of Dantzig-Wolfe inner linearization as applied to the dual problem of ECP(t, j), DP(t, j).*

Having completed the analysis of this basic algorithm, we would like to show how it is implemented in solving the entire program, SDLP. Further complications enter into OLSDLP because of possible degeneracy in the subproblems. In the next section, we discuss these difficulties and how they relate to stochastic programs. We also propose ways for resolving them.

4. The Degeneracy Problem

One weakness of decomposition techniques is that much of the work to optimize subproblems can be wasted, because the final inputs from the sub to the master differ so much from the initial ones. This can lead to many iterations from master to subproblem that a method with more interaction between the problems might be able to avoid. The next two methods we present have a more unified framework, and hence, fit this description. In this section, we will show how to make OLSDLP more responsive in the subproblems to changes in the master.

One property that might cause unnecessary iterations in decomposition schemes is the fact that excess columns in the basis of the master problem (that is,more than those required to meet the original set of constraints) cause degeneracy in the solution of the subproblems. Dantzig and Abrahamson

[1] first observed (and then proved) this property in their experiments with a dual nested decomposition algorithm for deterministic multi-stage linear programs. They also noticed that repeated sub-optimization changed the basis in the master problem very slightly, and they theorized that some efficiency may be gained by allowing the subproblems to determine some of the values of first period basic variables. This is possible because of the subproblem degeneracy.

To show degeneracy in SDLP, we will again refer to ECP(t, j) as written in (17) and will use ECP$(t+1, \bar{j})$ as

$$\min z_{t+1}^{\bar{j}} = c_{t+1} x_{t+1}^{\bar{j}}$$

subject to

$$\begin{aligned} A_{t+1} x_{t+1}^{\bar{j}} &= \xi_{t+1}^{\bar{j}} + B_t x_t^j, \\ x_t^{\bar{j}} &\geq 0. \end{aligned} \tag{19}$$

where we have dropped the $Q_{t+1}^{\bar{j}}$ and $D_{t+1}^{\bar{j}}$ from ECP$(t+1, \bar{j})$ for the sake of clarity. Equation (19) is the form of ECP$(t+1, \bar{j})$ used before any of its subproblems have been encountered. The degeneracy result is included in the following lemma.

Lemma 4. *If a constraint of the form (7) is binding at the optimal solution $x_t^{j,*}$, to (17), then every feasible primal basic solution of (19) with right-hand side, $\xi_{t+1}^{\bar{j}} + B_t x_t^{j,*}$, is degenerate.*

Proof. For the binding cut, we have

$$(-\sigma_{t+1}^{\bar{j}} B_t) x_t^{j,*} = \sigma_{t+1}^{\bar{j}} \xi_{t+1}^{\bar{j}}. \tag{20}$$

Now, let $A^{\bar{B}}_{t+1}$ be a feasible basis for (19). By applying $z^{\bar{j}}_{t+1}$ to this, we find

$$\sigma^{\bar{j}}_{t+1} A^{\bar{B}}_{t+1} x^{\bar{B}}_{t+1} = \sigma^{\bar{j}}_{t}(\xi^{\bar{j}}_{t+1} + B_t x^{j,*}_t) = 0. \tag{21}$$

We have $\sigma^{\bar{j}}_{t+1} A^{\bar{B}}_{t+1} \leq 0$, but $\sigma^{\bar{j}}_{t+1} A^{\bar{B}}_{t+1} \neq 0$ for $A^{\bar{B}}_{t+1}$ a basis, hence, there exists some $x^{\bar{B}}_{t+1,j} = 0$, proving the result. ∎

This lemma implies that a degeneracy will occur in any subproblem that has forced a tight feasibility cut on the master problem. The constraints in θ which enter the optimality conditions can also cause degeneracies in the subproblems. The difficulty with these degeneracies, however, is that they may enter in any subproblem, and we may not be able to determine which one. We state this degeneracy result in the following lemma.

Lemma 5. *If two constraints of the form (12) are binding at the optimal solution* $(x^{j,*}_t, \theta^{j\,*}_t)$ *to (17), then every solution of (19) for all* $\bar{j}$ *which satisfies the optimality criterion,* $\bar{z}^{\bar{j}}_{t+1} \geq \theta^{j,*}_t$, *includes a degenerate solution for some* $\bar{j}$.

Proof. Let the binding constraints be

$$-(\pi^1 B_t) x^{j,*}_t + \theta^{j,*}_t = \rho^1, \tag{22}$$

and

$$-(\pi^2 B_t) x^{j,*}_t + \theta^{j,*}_t = \rho^2. \tag{23}$$

Let the optimal set of bases for $\{\bar{j}^1, \ldots \bar{j}^k\}$ be $\{A^{B_{1,k}}_{t+1}, \ldots, A^{B_{1,k}}_{t+1}\}$. Associated with these bases are prices $\{\pi^{1,*}, \ldots, \pi^{*,k}\}$. These prices may be the same as those for one of (22) or (23). Without loss of generality assume they are identical with the prices in (23). They must be distinct from (22) because,

in the progression of the algorithm, (23) must have been violated, when (22) was satisfied and before (23) was added to the constraint set of (17). Now, letting

$$\pi^1 = \sum_{i=1}^{k} p^i \pi^{1,i} \text{ and } \rho^1 = \sum_{i=1}^{k} p^i \pi^{1,i} \xi_{t+1}^{\bar{j}i},$$

we have

$$\sum_{i=1}^{k} p^i \pi^{1,i} A_{t+1}^{B_i} x_{t+1}^{\bar{j}i} = \sum_{i=1}^{k} p^i \pi^{1,i} (\xi_{t+1}^{\bar{j}i} + B_t x_t^{j,*}), \tag{24}$$

and

$$\sum_{i=1}^{k} p^i \pi^{i,*} A_{t+1}^{B_i} x_{t+1}^{\bar{j}i} = \sum_{i=1}^{k} p^i \pi^{j,*} (\xi_{t+1}^{\bar{j}i} + B_t x_t^{j,*}). \tag{25}$$

So, by assumption

$$\theta_t^{j,*} = \sum_{i=1}^{k} p^i \pi^{j,*} (\xi_{t+1}^{\bar{j}i} + B_t x_t^{j,*})$$

and $\pi^{i,*} A_{T+1}^{B_1} = c^{B_1}$, and $\pi^{1,i} A_{t+1}^{B_i} \leq c^{B_i}$ for $\pi^{1,i}$ feasible for all i. Therefore, we obtain

$$\sum_{i=1}^{k} p^i \pi^{j,*} A_{t+1}^{Bi} x_{t+1}^{\bar{j}i} = \sum_{i=1}^{k} p^i \pi^{1,i} A_{t+1}^{bi} x_{t+1}^{\bar{j}i} \tag{26}$$

or

$$\sum_{i=1}^{k} p^i (c^{Bi} - \pi^{1,i} A_{t+1}^{Bi}) x_{t+1}^{\bar{j}i} = 0. \tag{27}$$

Now, $p^i \geq 0$, $x_{t+1}^{\bar{j}i} \geq 0$, and $c^{Bi} - \pi^{1,i} A_{t+1}^{Bi} \leq 0$, but we must have $c^{Bi} \neq \pi^{1,i} A_{t+1}^{Bi}$ for $\pi^{i,*}$ unique; therefore, there exists some $\bar{j}i$ such that $x_{t+1}^{\bar{j}i}$ is degenerate. Hence, the result. ∎

The degeneracy we have shown implies that the subproblems are too restricted to alter the direction of the solution to the master problem. Dantzig and Abrahamson [1] have proposed remedying this difficulty in the multi-stage deterministic model by passing columns forward from the master to the

subproblem and allowing the subproblems to determine the weights of these surplus columns in the master problem. We present below an application of this technique to SDLP and discuss its weaknesses in the case of stochastic programs.

We assume an optimal solution, $x_t^{j,*}$, of (17) is partitioned as

$$x_t^{j,*} = (x_t^B, x_t^s) \tag{28}$$

where x_t^B corresponds to a square nonsingular submatrix of A_t, A_t^B, and x_t^s corresponds to A_t^s. We can write x_t^B in terms of x_t^s as

$$x_t^B = (A_t^B)^{-1}(\xi_t^j + B_{t+1}\hat{x_{t+1}^j}) - (A_t^B)^{-1}A_t^s x_t^s. \tag{29}$$

So, we can let x_t^s vary as long as $x_t^B \geq 0$. We then can write (19) for each $\bar{j}$ as

$$\begin{aligned} &\min c_{t+1}\bar{x_{t+1}^j} \\ &\text{subject to} \\ &A_{t+1}\bar{x_{t+1}^j} = \bar{\xi_{t+1}^j} + B_t^B(A_t^B)^{-1}(\xi_t^j + B_{t-1}\hat{x_{t-1}^j}) \\ &\qquad - B_t^B(A_t^B)^{-1}A_t^S x_t^S + B_t^S x_t^S, \\ &\bar{x_{t+1}^j} \geq 0,\ x_t^S \geq 0, \end{aligned} \tag{30}$$

or, letting

$\hat{B}_t^S = B_t^S - B_t^B(A_t^B)^{-1}A_t^S$ and $\hat{\xi}_{t+1}^{\bar{j}} = \bar{\xi_{t+1}^j} + B_t^B(A_t^B)^{-1}(\xi_t^j + B_{t-1}\hat{x_{t-1}^j})$,, as

$$\begin{aligned} &\min c_{t+1}\bar{x_{t+1}^j} \\ &\text{subject to} \\ &A_{t+1}\bar{x_{t+1}^j} - \hat{B}_T^S x_t^S = \bar{\xi_{t+1}^j}, \\ &\bar{x_{t+1}^j} \geq 0,\ x_t^S \geq 0, (x_t^B \geq 0), \end{aligned} \tag{31}$$

where we have added the constraint $x_t^B \geq 0$, parenthetically, because we must check in optimizing (31) that this is not violated. If so, we would fix the variables at zero and proceed.

This problem of checking for feasibility of x_t^B enters both the deterministic and stochastic programs, but, in the stochastic program, any of the subproblems of type (31) may determine x_t^s. Since x_t^s enters the program in which a degeneracy is caused (filling the degenerate variable's position in the basis), we must know for which $\tilde{j}$ the program (31) will have some x_t^s basic. This would be possible if all the x_t^s corresponded to tight *feasibility* (type (7)) cuts, because then the degeneracy would occur in the corresponding scenarios which generated those cuts. For the *optimality* cuts (type (12)), however, degeneracy, where x_t^s is basic, can be in any scenario.

To apply this column passing technique, in general, to SDLP, we can formulate the following alternative form of (17). It includes constraints to keep x_t^B feasible as well as the remaining additional cuts.

$$\min \tilde{c}_t^S x_t^S + \theta_t^j$$

subject to

$$\begin{aligned}
-(A_t^B)^{-1} A_t^s x_t^s &\geq -(A_t^B)^{-1}(\xi_t^j + B_{t-1}\hat{x_{t-1}^j}), \\
-(\tilde{\sigma_t^{j,k}} B_t^s) x_t^s &\geq [\sigma_t^{j,k}(\tilde{\xi_{t+1}^j})], k = 1, \ldots, p, \\
-(\tilde{\pi_1^\ell} B_t^s) x_t^s + \theta_t^j &\geq \tilde{p}^\ell, \ell = 1, \ldots, q, \\
x_t^s &\geq 0,
\end{aligned} \tag{32}$$

where $\tilde{c}_t^s = c_t^B (A_t^B)^{-1} A_t^s + c_t^s$, and the other quantities under tilde are correspondingly defined to reflect the substitution of (29) for x_t^B.

The program (32) is a second master problem that we can use to determine the optimal values of x_t^s given x_t^B. Our proposal then is to follow

OLSDLP with (32) in place of (17). We would do this after solving the subproblems of the form (31), which have generated tight feasibility cuts for (17) and in which we know degeneracies must occur.

The use of this method of passing some columns for tight feasibility cuts and then determining the other x_t^s by using (32) in OLSDLP depends on the difficulty of solving the first master problem. If the repeated solution of (17) has indicated that some variables, x_t^B, are persistent in the basis, then the solution of (32) could obtain the optimal values without involving a reoptimization of (17). Furthermore, if the number of surplus variables, x_t^s is small, (32) may become significantly easier to solve than (17). This alternative method then is one that must be adopted to the individual problem and its requirements. The complications of creating an additional optimization problem in (32) may outweigh the savings in solving this smaller problem.

5. The Complete Methods

We are prepared now to present the algorithm NDSDLP for the entire stochastic problem. This method involves repeated use of the algorithm OLSDLP, and proceeds through the tree of possible scenarios in SDLP in a forward and backward manner. Our presentation here does not include the resolution of the degeneracy problem as discussed in Section 4, but this may be added as a modification to OLSDLP. The algorithm follows.

NDSDLP

Step 0. Set up a problem of the form (17) with no extra constraints for each scenario j in each period t of SDLP.

Step 1. Solve (17) for period T (written 17-1). Use the result x_1^*, and solve

(17) for each node in period 2 ((17-2j) for $j = 1, \ldots, k_2$). If any subproblem is infeasible, add a feasibility constraint of type (7) to (17-1). Solve (17-1) again and proceed to period 2.

After each subproblem at $t = 2$ is feasible, proceed to solve (17-3j) for $j = 1, \ldots, k_3$. Again, for any infeasibilities, pass back cuts. Continue in this manner to period T, until there is a feasible solution for (17-tj) for all t and j. If an infeasible solution to (17-1) ever results, then stop–the problem is infeasible. Else, the result is a feasible primal solution of SDLP.

Step 2. Solve OLSDLP for every scenario in period $T - 1$. This implies master-suboptimality for the last period. Set $T = T - 2$ and go to Step 3.

Step 3. If $t = 1$, go to Step 4. Otherwise, solve OLSDLP for every scenario in period t. For some scenario j at t, this may involve resolving OLSDLP for its descendants $\bar{j}$ at $t + 1$ in order to get optimality for the subproblems. We say a subproblem is "solved", in terms of the algorithm, when OLSDLP applied to it ends in master-suboptimality.

After OLSDLP has been solved for each j at t, set $t = t - 1$ and repeat Step 3.

Step 4. Solve OLSDLP for the original master problem at period 1. This program may again involve resolving the subproblems. If OLSDLP at 1 results in an unbounded or infeasibility termination, then stop–SDLP is accordingly unbounded or infeasible. If OLSDLP ends with master-suboptimality, then stop–the current solution is optimal for SDLP.

This algorithm follows an iterative procedure as in dynamic programming. We pursue this relationship more closely in Chapter VI. NDSDLP also terminates in a finite number of steps as we state in the following theorem.

Theorem 3. *The method, NDSDLP, terminates in a finite number of steps with an optimal solution to SDLP or the unbounded or infeasibility conditions from OLSDLP.*

Proof. From Theorem 2, we know that each implementation of OLSDLP must terminate in a finite number of steps. Since the algorithm proceeds backwards after each period's scenarios are solved by OLSDLP, the terminal conditions in Step 4 must be met in a finite number of steps. ∎

Several improvements can be made to NDSDLP to aid in its efficiency. We have already mentioned the second decomposition possible in OLSDLP as a resolution to the degeneracy problem. We may also want to proceed between the periods without completely satisfying master-suboptimality. This modification may help efficiency, but it must be done carefully in order to avoid any excessive number of iterations among the periods.

Another possibility for speeding the search for a feasible primal solution in Step 1 is that SDLP may have inequality constraints. In this case, the subproblem at period t is

$$\min c_t x_t$$

subject to

$$\begin{aligned} A_t x_t &\geq \xi_t^j + B_{t-1} x_{t-1}, \\ x_t &\geq 0, \end{aligned} \tag{33}$$

for all $\bar{j}$. Hence, to find feasibility for all $\bar{j}$, we need only solve the Phase I problem

$$\min ev$$

$$\text{subject to}$$

$$A_t x_t - Iu + Iv = \alpha_t + B_{t-1} x_{t-1}$$
$$x_t \geq 0, \tag{34}$$

where $\alpha_t = (\alpha_{t,i} \mid \alpha_{t,i} = \max_{\bar{j}} \xi^{\bar{j}}_{t,i})$, thus $\alpha_t \geq \xi^{\bar{j}}_t$ for all $\bar{j}$. A feasible solution to (33) implies that each subproblem $\bar{j}$ at period t is feasible for x_{t-1}.

By solving (33) in Step 2 of OLSDLP, instead of solving the Phase I problem (18) of each $\bar{j}$, we eliminate many unnecessary optimizations. This would greatly aid the efficiency of NDSDLP for SDLP's that have inequality constraints. These programs are quite common in practice and, thus, (33) should prove most valuable.

This modification and our presentation of NDSDLP above demonstrate some of the possibilities for solving SDLP by concentrating on the optimization of smaller subproblems. In Chapter VII, we discuss the computational aspects of the algorithm more carefully. In the next two chapters, we present other methods that rely upon subproblem optimization, but maintain closer ties between the sub- and master problem.

Chapter IV

A Piecewise Strategy

1. Introduction

The decomposition algorithm, NDSDLP, in the last chapter broke the stochastic program SDLP into a sequence of master–subproblem relations. A drawback to NDSDLP could come from wasted effort in optimizing subproblems without affecting the master problems' inputs. The approach in this chapter, the *piecewise strategy* for SDLP, or *PCSDLP*, also separates the program into master and subproblems, but it allows for only one optimization of the subproblems without the master's involvement. This method, which maintains some ties among the separate scenarios, forms a bridge between the decomposition approach in Chapter III and the local basis factorization of Chapter V.

The method is called "piecewise" because it relies on the piecewise linear property of the objective function. Piecewise methods in general (see Geoffrion [27]) follow an optimizing trajectory across the regions of the feasible set. For a convex function, an optimization is performed on each region that leads either to a boundary or interior solution. If the solution is interior, then that point is optimal. If a boundary point is optimal, one optimizes on the adjacent region and repeats the process. (See Figure 1.) If no direction in an adjacent region is improving, then the current point again is optimal.

The piecewise strategy has been applied to large–scale linear programming through a method, called "partitioning", for which, J. B. Rosen [53] has been most responsible. Our use of the strategy in this chapter will be to

exploit the repetitions of the blocks of SDLP and to form an algorithm that can adapt to different scenarios and combine them adequately. In Section 2, we present the basic master–subproblem algorithm for this method and show where efficiencies can be made in its implementation. Section 3 then states the strategy for the full program and presents the difficulties that may occur with PCSDLP.

2. The Master–Subproblem Relationship

The method proceeds by performing two–period optimizations for successively larger problems. In this section, we present the two–period problem, in which, the first, parent, scenario forms the master problem for its direct descendants, the subproblems. Without loss of generality, we assume that this optimization takes place between the first and second periods.

The first period problem we solve is

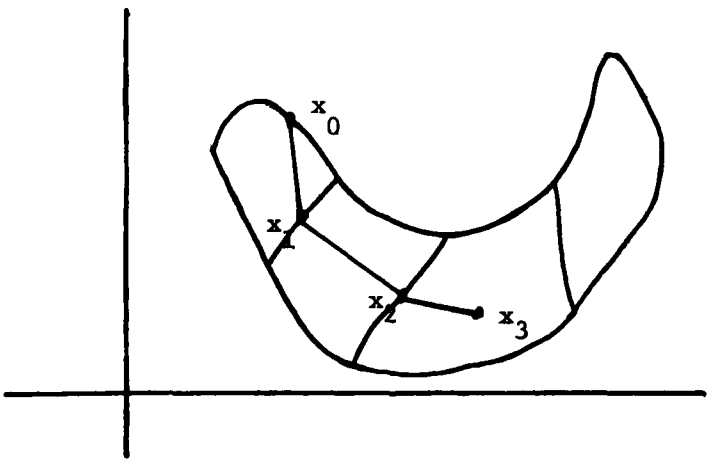


Figure 1. The piecewise path begins at x_0 and leads to x_3.

$$\begin{aligned} &\min \varsigma^0(x_1) = c_1 x_1 \\ &\text{subject to} \quad A_1 x_1 = b_1, \\ &\qquad\qquad x_1 \geq 0. \end{aligned} \tag{1}$$

If (1) is infeasible, the program is infeasible, and we stop. If (1) is unbounded, then we follow Step 2' of NDSDLP to remove this case. If we succeed, we return with the cuts that eliminate the unbounded ray and resolve (1). Now, for x_1^0, an optimal solution to (1), we want to find $Q(x_1^0)$ as ECP of Chapter III, where

$$Q(x_1^0) = \mathrm{E}_{\xi_2}[\min c_2 x_2 \text{ s.t. } A_2 x_2 = \xi_2 + B_1 x_1^0, x_2 \geq 0]. \tag{2}$$

Here, if there exists ξ_2^j such that there is no feasible solution in the jth scenario, then we form a cut as in (3.7) and add it to (1) as part of its constraint matrix. We continue until each subproblem is feasible. Next, associated with each ξ_2^j, there exists an optimal basis, A_2^B, for the problem in $Q(x_1^0)$. We write $Q(x_1^0)$ as

$$Q(x_1^0) = \sum_{j=1}^{k_2} p^j c_2((A_2^{B_j})^{-1}\xi_2^j + (A_2^{B_j})^{-1}B_1 x_1^0). \tag{3}$$

The function $Q(x_1)$ from (3) is linear for all x_2 feasible for $A_2^{B_j}$. Thus,

$$Q(x_1) = \sum_{j=1}^{k_2} p^j c_2((A_2^{B_j})^{-1}\xi_2^j + (A_2^{B_j})^{-1}B_1 x_1) \tag{4}$$

for all x such that

$$x_2 = (A_2^{B_j})^{-1}\xi_2^j + (A_2^{B_j})_1^{-1} x_1 \geq 0, \tag{4a}$$

where we have assumed a single lower bound for all x_2 at 0. In general, and, for most practical purposes, both lower and upper bounds should be

included, augmenting the constraints in (4a). In implementations, this is especially important for artificial variables, where lower and upper bounds coincide.

Now, we can write $\varsigma^0(x_1)$ in (1) as

$$\varsigma^0(x_1) = (c_1 + \sum_{j=1}^{k_2} p^j c_2^{B_j}(A_2^{B_j})^{-1}B_1)x_1 + \sum_{j=1}^{k_2} p^j c_2^{B_j}(A_2^{B_j})^{-1}\xi_2^j, \quad (5)$$

for all x_1 such that $(A_2^{B_j})^{-1}\xi_2^j + (A_2^{B_j})^{-1}B_1x_1 \geq 0$ for all $j = 1, \ldots, k_2$.

We note , for optimal multipliers π^j in subproblem j, that $\pi^j = c_2^{B_j}(A_2^{B_j})^{-1}$, which makes computations in (5) somewhat easier. We also note that each $A_2^{B_j}$ need not be unique, so we may use one basis several times without checking (4a) for each j. We will return to this idea below, but, for the current exposition, we will use constraints (4a) explicitly. Thus, for x_1 restricted as in (4a), we look for an improving direction along the current linear section of $Q(x_1)$ by solving

$$\begin{aligned}
&\min \varsigma^1(x_1) = [c_1 + \textstyle\sum_{j=1}^{k_2} p^j c_2(A_2^{B_j})^{-1}B_1]x_1 \\
&\text{subject to} \\
&\qquad A_1x_1 \qquad\qquad = b_1, \qquad\qquad (6a) \\
&\qquad (A_2^{B_j})^{-1}B_1x_1 \quad \geq \rho^j, j = 1, \ldots, k_2, (6b) \\
&\qquad x \qquad\qquad \geq 0,
\end{aligned} \quad (6)$$

where $\rho^j = (A_2^{B_j})^{-1}\xi_2^j$.

From (6), we obtain an optimal solution, x_1^1, and optimal objective value, $\varsigma^1(x_1^1)$. This is the initial point of our feasible path. We next look at the set of constraints in (6b) which are active. We define

$$T = \{(i,j) : [(A_2^{B_j})^{-1}B_1](i,*) \cdot x_1^1 = \rho_i^j\}, \quad (7)$$

where ρ_i^j is the ith component of the vector ρ^j. $\mathcal{T}$ includes the rows and scenarios that generated a binding cut. It corresponds to the set of degeneracies in the subproblems of NDSDLP and is directly related to the surplus columns described in the next chapter. We next order the pairs $(i,j) \in \mathcal{T}$ lexicographically and write $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_p\}$.

Now, if $\mathcal{T} = \emptyset$, then x_1^0 is a solution of (6), so, by our convexity result in Lemma 2.1, $z(x_1^0) = c_1 x_1^0 + Q(x_1^0) \leq c_1 x_1 + Q(x_1)$ for all feasible x_1. Hence, x_1^0 is an optimal solution of this two-stage SDLP.

For $\mathcal{T} \neq \emptyset$, we consider subproblem j for $\tau_1 = (i,j)$. Here, we have

$$x_2^{B_j} = (A_2^{B_j})^{-1}\xi_2^j + (A_2^{B_j})^{-1}B_1 x_1 \tag{8}$$

and

$$x_2^{B_j}(i) + (A_2^{B_j})^{-1}(i,*)A_2^N x_2^N = 0, \tag{9}$$

where x_2^N is the non-basic partition of the variables, x_2. We now want to force $x_2^{B_j}(i)$ out of the basis, so we can follow a new path in the adjacent feasible region. To maintain optimality, we find the entering variable as in the Dual Simplex Method, (see Dantzig [16])

$$\frac{\bar{c}_2^N(s)}{-\bar{A}_2^N(i,s)} = \min_{-\bar{A}_2^N(i,j)>0} \frac{\bar{c}_2^N(j)}{\bar{A}_2^N(i,j)}, \tag{10}$$

where $\bar{c}_2^N$ and $\bar{A}_2^N$ are the representations of c_2^N and A_2^N relative to the basis, $A_2^{B_j}$.

We can now pivot out $x_2^{B_j}(i)$ and replace it with $x_2^N(s)$. In doing this, we keep $A_2^{B_j}$ and call the new basis $A_2^{B_j,*}$, an *auxiliary basis*, where

$$(A_2^{B_j,*})^{-1} = \eta(i,s)^{-1}(A_2^{B_j})^{-1}, \tag{11}$$

for $\eta(i,s)$, the elementary matrix corresponding to the pivot of $x_2^N(s)$ into the basis in position i. This new basis is used to restrict x_1 in (4a).

We now can formulate our auxiliary problem, for $\mu = 1$,

$$\begin{aligned}
&\min \varsigma^{\mu,*}(x_1) = [c_1 + \textstyle\sum_{j=1}^{k_2} p^j c_2^{B_j} (A_2^{B_j})^{-1} B_1] x_1 \\
&\text{subject to} \\
&\qquad A_1 x_1 = b_1, \qquad (12) \\
&\qquad (A_2^{B_j})^{-1} B_1 x_1 \geq \rho^{j,*}, j = 1, \ldots, k_2, \\
&\qquad x_1 \geq 0.
\end{aligned}$$

In (12), we have changed the objective function from (6) and some of the constraints (6b). The variables x_1^1 still form a feasible basis, but they may no longer be optimal.

We proceed to price out the cost row in (12) with the new parameters and to check optimality. If x_1^1 is still optimal, then we drop $A_2^{B_j,*}$ and return to $\mathcal{T}$ for r_2, again find an entering variable, and form the auxiliary problem (12). As long as we cannot improve on x_1^1 for r_l, we try r_{l+1}. If we find $l+1 > p$, then no direction can improve on x_1^1, hence, it is optimal.

If we find that, for any r_l, x_1^1 is not optimal in (12), then we optimize (12) and obtain x_1^2. We set $\varsigma^2(x_1^2) = \varsigma^{1,*}(x_1^2)$, replace $A_2^{B_j}$ by $A_2^{B_j,*}$, form a new set, $\mathcal{T}$, of the tight constraints, and proceed again with r_1 in search of an improving direction.

We follow the above procedure to obtain a sequence of decreasing objective values, $\varsigma^1(x_1^1) > \varsigma^2(x_1^2) > \cdots > \varsigma^\mu(x_1^\mu)$, until we cannot improve our current solution. The algorithm we have described has the following steps :

PCSDLP(2)

Step 0. Find an optimal bounded feasible solution of (1), or terminate if infeasible or if Step2′ of NDSDLP implies unboundedness.

Step 0'. Find an optimal feasible basis, $A_2^{B_j}$, for each subproblem j in $Q(x_1^0)$ by applying feasibility cuts (3.7) to (1).

Step 1. Form the program (6) using the set of bases, $B = \{A_2^{B_j}\}$. Solve (6) and obtain $\varsigma^1(x_1^1)$ and x_1^1. Form $\mathcal{T}$. If $\mathcal{T} = \emptyset$, stop, x_1^0 is optimal. If $\mathcal{T} \neq \emptyset$, set $l = 1$, $\mu = 1$, and, go to Step 2.

Step 2. If $l > p$, stop, x_1^μ is optimal. For $\tau_l = (i, j)$, find the entering variable s in scenario j by (10). Form the auxiliary basis, $A_2^{B_j,*}$, and the program (12) for $\varsigma^{\mu,*}(x_1)$. Using x_1^μ as a starting solution, solve (12) and obtain x_1^*. If $\varsigma^{\mu,*}(x_1^*) = \varsigma^{\mu,*}(x_1^\mu)$, set $l = l+1$ and return to Step 2. If $\varsigma^{\mu,*}(x_1^*) < \varsigma^{\mu,*}(x_1^\mu)$, go to Step 3.

Step 3. Update $\mathcal{T}$ and B. Set $x_1^{\mu+1} = x_1^*$, $l = 1$, $\mu = \mu + 1$, and go to Step 2.

The following theorem states the finiteness of PCSDLP.

Theorem 1. *The method described above, PCSDLP, terminates in a finite number of steps with an infeasible, unbounded, or optimal solution to the two-stage ($T = 2$) form of the program, SDLP.*

Proof. From Chapter III, we know that Steps 0 and 0' must terminate in a finite number of steps. After Step 2, the solution to (6) and (12) must be feasible in SDLP because primal feasibility of the last solution is maintained. It is bounded because (6) and (12) are more restrictive than (1).

$\mathcal{T}$ is finite since the number of constraints (12b) is finite and each improving solution corresponds to a new set of bases, B. Since there are a finite number of possible basis set combinations, the algorithm must terminate. ∎

3. A Method for Reduced Basis Storage Requirements

Efficiency and storage requirements in the solution of (12) can be significantly improved, if we do not include redundant constraints that occur with duplicated bases, $A_2^{B_j}$. To do this, begin by checking in Step 0' for a repetition of $A_2^{B_j}$. We start with $l = 1$ and increase l, letting each distinct new basis be $A_2^{B_l}$. We obtain $\mathcal{B} = \{A_2^{B_l} | l = 1, \ldots, q\}$. We also define

$$p(l) \equiv \sum_{J(l)} p^j \tag{13}$$

where $J(l) = \{$ all scenarios j with optimal bases, $A_2^{B_l}\}$.

Now, when we construct the constraints (12b), we define

$$\rho^{l,*} = \max_{j \in J(l)} \rho^{j,*}, \tag{14}$$

and, for each component i, we store $\hat{j}(l,i)$ for every l, where $\rho^{\hat{j},*}(i) \geq \rho^{j,*}(i)$, for all $j \in J(l)$. Thus, (12) becomes

$$\begin{aligned}
&\min \varsigma^{\mu,*}(x_1) = [c_1 + \textstyle\sum_{l=1}^{q} p(l) c_2^{B_l,*} (A_2^{B_l,*})^{-1} B_1] x_1 \\
&\text{subject to} \\
&\qquad A_1 x_1 = b_1, \qquad (15a) \\
&\qquad (A_2^{B_l,*})^{-1} B_1 x_1 \geq \rho^{l,*}, \; l = 1, \ldots, q, \qquad (15b) \\
&\qquad x_1 \geq 0.
\end{aligned} \tag{15}$$

Now, in Step 2, write the elements of $\mathcal{T}$ as $\tau_\lambda = (i, \hat{j}(l))$. Each time a new auxiliary basis, $A_2^{B_j,*}$, is investigated, if $A_2^{B_j,*} \in \mathcal{B}$, then we adjust $p(l)$ and possibly $\rho^{l,*}$, but do not change the coefficient matrix. If $A_2^{B_j,*} \notin \mathcal{B}$, then we must add another set of constraints to (15). At Step 3, we update $\mathcal{B}$, the associated probabilities, $p(l)$, and update $\mathcal{T}$ using the pair, $(i, \hat{j}(l))$.

This modification can significantly reduce the number of constraints since a large number of scenarios may have the same basis. We could again use the Garstka–Rutenberg procedure (see Chapter I) to find the probability that each basis is optimal in Step 0' without solving the individual problems. This effect combined with the smaller size of (15) can lead to greater computational efficiencies.

Another efficiency can be gained from using a method similar to the column passing technique of Chapter III. During the algorithm, we may observe that one set of variables, $\{x_1^B\}$, remains in the optimal basic set, $\{x_1^\mu\}$, while the other variables are chosen from a set, $\{x_1^S\}$. If the columns of the set $\{x_1^B\}$ have full rank, we can take a square non-singular submatrix, A_1^B, from (15) and find

$$x_1^B = (A_1^B)^{-1}b_1 - (A_1^B)^{-1}A_1^S x_1^S. \tag{16}$$

We then eliminate x_1^B from (15) and obtain

$$\begin{aligned} \min \varsigma^{\mu,*}(x_1^S) = \quad & \tilde{c}_1^s x_1^S \\ \text{subject to} \quad & \\ & -\tilde{A}_1^s x_1^S \geq -(A_1^{\tilde{B}})^{-1}b_1, \\ & ((A_2^{B_l,*\tilde{}})^{-1}B_i)x_1^S \geq \tilde{\rho}^{l,*}, l = 1,\ldots,q, \\ & x_1^S \geq 0, \end{aligned} \tag{17}$$

where tilde indicates that $\tilde{c}_1^s$, $\tilde{A}_1^s$, $((A_2^{B_l,*\tilde{}})^{-1}B_i)$, and $\tilde{\rho}^{l,*}$ are defined relative to x_1^B through substitution of (16) into the program (12). The definitions are completely analogous to those in (3.32).

The optimization procedure can then continue with the reduced problem (17) to find the optimal values x_1^S, given x_1^B. Using the result of (17) in (15) would then determine optimality. When decisions can be narrowed to

choices among a few variables, this modification may again prove effective in improving efficiency.

The algorithm, PCSDLP(2), is stated for two–period optimizations. It does not require the subproblems to be reoptimized after Step 0'. By maintaining primal feasibility, it is always on a feasible path to the solution and may eliminate the problems of wandering among multiple suboptimal points. In the next section, we present the implementation of PCSDLP(2) for general multi–stage programs.

4. The Complete Solution Strategy

The PCSDLP method follows a procedure very similar to NDSDLP in its passing through the scenarios from period to period. In fact, both of these methods can be seen as local approximations of a dynamic programming scheme, which we present in greater detail in Chapter 6. PCSDLP even begins by finding a feasible primal solution through NDSDLP, but PCSDLP never allows for primal infeasibility or non–optimality in a subproblem after a single optimization.

First, we set up subproblems for each node as in Step 0 of NDSDLP. Next, we find a feasible primal solution by Step 1 of NDSDLP, passing feasibility cuts as we proceed through the periods from 1 to T. After finding this feasible solution, we start by applying PCSDLP(2) to the master–subproblem relations at period $T-1$. Primal feasibility is then maintained throughout the optimization.

For each scenario j in period $T-1$, we solve

$$\begin{aligned}
&\min c_{T-1}x^j_{T-1} + \textstyle\sum_{\bar{j}=1}^{\bar{k}_T} p^{\bar{j}} c_T x^{\bar{j}}_T \\
&\text{subject to} \\
&A_{T-1}x^j_{T-1} \qquad\qquad = \xi^j_{T-1} + B_{T-2}\hat{x}^{\hat{j}}_{T-2}, \\
&-B_{T-1}x^j_{T-1} \qquad +A_T x^{\bar{j}}_T = \xi^{\bar{j}}_T,\ \bar{j} = 1, \ldots, \bar{k}_T, \\
&\qquad\qquad x^j_{T-1} \geq 0, x^{\bar{j}}_T \geq 0,\ \bar{j} = 1, \ldots, \bar{k}_T,
\end{aligned} \tag{18}$$

by using PCSDLP(2).

For a solution to (18), define

$$\beta^{\bar{j}} = \{i : x^{\bar{j}}_T(i) \text{ is basic in subproblem } j\}$$

and

$$\gamma^{\bar{j}} = \{i : i \text{ is basic in row } l \text{ and } (l, \bar{j}) \in \mathcal{T}\}.$$

Now, with the solution from PCSDLP(2), we want to find the optimal basis for the full problem (18). This larger matrix will form the basis of a subproblem for period $T-2$. We first include the set of basic variables in the master problem, $\{x^{B_j}_{T-1}\}$. The basic variables from the subproblems will be chosen as

$$X^j_T = \{x^{\bar{j}}_T(i) : i \in \beta^{\bar{j}} \bigcap \overline{\gamma}^{\bar{j}}\}. \tag{19}$$

This definition eliminates the degenerate variables from the basic set. Since the elements of X^j_T are the only non-zero variables in an optimal feasible solution to (18), if (18) is not degenerate, then the union of $\{x^{B_j}_{T-1}\}$ and X^j_T must form a basic set of variables in (18).

If (18) is degenerate, then we must check whether the columns corresponding to $\{x^{B_j}_{T-1}\}$ and X^j_T span the solution space. If not, we add columns from those corresponding to

$$\overline{X}^j_T = \{x^{\bar{j}}_T(i) : i \in \beta^{\bar{j}} \bigcap \gamma^{\bar{j}}\}. \tag{20}$$

If a column of $x_T^j(i) \in \overline{X}_T^j$ is independent of the columns in the present basis, then it is added until full rank is a achieved. We know that we must obtain a basis since the union of all columns for $\{x_{T-1}^{B_j}\}$, X_T^j, and $\overline{X}_T^j$ spans the space.

We thus obtain a basis for all j in $T-1$. We call this basis, $D_{T-1}^{B_j}$, and we further define

$$D_{T-1} = \begin{pmatrix} A_{T-1} & & \\ -B_{T-1} & A_T & \\ & \ddots & \ddots \\ & -B_{T-1} & A_T \end{pmatrix} \tag{21}$$

$$d_{T-1} = (c_{T-1}, p^1 c_T, \ldots, p^{\bar{k}_T} c_T), \tag{22}$$

$$\psi_{T-1}^j = (\xi_{T-1}^j, \xi_T^1, \ldots, \xi_T^{\bar{k}_T})^T, \tag{23}$$

and

$$y_{T-1}^j = (x_{T-1}^j, x_T^{\bar{j}_1}, \ldots, x_T^{\bar{j}_{\bar{k}_T}}) \tag{24}$$

Now, the problem for PCSDLP(2) is

$$\min \varsigma_{T-2}^{\hat{j}} = c_{T-2} x_{T-2}^{\hat{j}} + \textstyle\sum_{j=1}^{\bar{k}_{T-1}} p^j d_{T-1} y_{T-1}^j \tag{25}$$

subject to

$$A_{T-2} x_{T-2}^{\hat{j}} = \xi_{T-2}^{\hat{j}} + B_{T-1} x_{T-1}^{\hat{j}}, \tag{25a}$$

$$-\tilde{B}_{T-2} x_{T-2}^{\hat{j}} + D_{T-1} = \psi_{T-1}^j, \; j = 1, \ldots, \bar{k}_{T-1}, \tag{25b}$$

$$x_{T-2}^{\hat{j}} \geq 0, y_{T-1}^j \geq 0, \; j = 1, \ldots, \bar{k}_{T-1},$$

where $\tilde{B}_{T-2}$ is the matrix B_{T-2} augmented by zeroes to correspond with D_{T-1}^j.

To begin PCSDLP(2) for (25), we substitute constraints of the form in (6b) for (25b) and enter Step 1 in PCSDLP(2). We never reoptimise the subproblems, but look for feasibility maintaining pivots in both periods $T-1$

and T. In general, after solving the two-stage problem for each scenario j in period t, we would again find the bases, $D_t^{B_j}$, and construct a two-stage problem for $t-1$ as in (25) by combining x_t^j with $(y_{t+1}^{\vec{j}_1}, \ldots, y_{t+1}^{\vec{j}_{\bar{k}_{t+1}}})$. PCSDLP(2) would begin by optimizing

$$\min \varsigma^1(x_{t-1}^{\hat{j}}) = [c_{t-1} + \textstyle\sum_{j=1}^{\bar{k}_t - 1} p^j d_t^{B_j} (D_t^{B_j})^{-1} B_1] x_{t-1}^{\hat{j}} \tag{26}$$

subject to

$$A_{t-1} x_{t-1}^{\hat{j}} = \xi_{t-1}^{\hat{j}} + B_{t-2} x_{t-2}^{\hat{i}},$$
$$(D_t^{B_j})^{-1} \tilde{B}_{t-1} x_{t-1}^{\hat{j}} \geq \rho^j, \; j = 1, \ldots, \bar{k}_{t-1},$$
$$x_{t-1}^{\hat{j}} \geq 0.$$

The dual pivoting operations could then be performed in any of periods t through T.

PCSDLP continues by combining master problems with subproblems and following these iterations back to period 1. This process uses the basis structure depicted in Figure 2b. of Chapter I, in which, we view each scenario as starting a new problem. The steps we have described for PCSDLP follow.

PCSDLP

Step 1. Follow Step 0 and Step 1 of NDSDLP to obtain a feasible solution to SDLP. Set $t = T - 1$ and set up a program of the form (12) for each scenario j in $T - 1$. Set $j = 1$.

Step 2. Follow Steps 1, 2, and 3 of PCSDLP(2) for the problem at node (j, t). If $j = k_t$, go to Step 3. If $j < k_t$, set $j = j + 1$ and return to Step 2.

Step 3. If $t = 1$, stop, x_1^{μ} is optimal. If $t > 1$, combine the master and subproblems of each scenario j at period t and form programs as in (26) to initiate PCSDLP(2). Set $t = t - 1$ and go to Step 3.

The finite termination of this method is guaranteed by the finiteness of

PCSDLP(2) and our passing back one period in each encounter with Step 3. We state this as a Corollary to Theorem 1.

Corollary. *PCSDLP terminates in a finite number of steps with an infeasible or unbounded solution from the procedures of NDSDLP or with an optimal bounded feasible solution to SDLP.*

PCSDLP's greatest potential improvement over NDSDLP is, as we have emphasized, its maintenance of primal feasibility and subproblem optimality. This advantage over the possible suboptimizations and infeasibilities in NDSDLP must be discounted, however, by the growth of the bases, D_t^B, in the subproblems. Their larger size may lead to a greater number of computations in performing pricing and the minimum ratio test in (1). We can, however, gain efficiency with a compact factorization of D_t^B. The following chapter describes such a technique and its application to the full problem, SDLP. This local basis method could then be used in conjunction with PCSDLP to gain still greater efficiency.

CHAPTER V

A Local Basis Simplex Method

1. Introduction

The two methods presented above, NDSDLP and PCSDLP, require the optimization of subproblems essentially independent of the main problem. This practice can be costly, leading to a great number of iterations. We describe below an adaptation of the Simplex Method for linear programs with stochastic structure. This method reduces the complications of stochastic linear programs by taking advantage of some fundamental properties of the basis. We call this approach a *local basis simplex method, LBSMPX* , because it relies on the near *square block triangularity* of the bases for these problems.

Block triangular linear programs, in general, have the form:

$$\begin{aligned} &\min \quad c_1x_1 + c_2x_2 + \cdots + c_Tx_T \\ &\text{subject to} \\ &\qquad A_{t1}x_1 + \cdots + A_{tt}x_t = b_t, t = 1, \ldots, T, \\ &\qquad\qquad x_t \geq 0, t = 1, \ldots, T. \end{aligned} \tag{1}$$

where $x_t \in R^{n_t}, b_t \in R^{m_t}, c_t \in R^{n_t}$, and the matrices A_{ij} are dimensioned accordingly.

The detached coefficient matrix is then:

$$A = \begin{pmatrix} A_{11} & & & \\ A_{21} & A_{22} & & \\ \vdots & \vdots & & \\ A_{T1} & A_{T2} & \cdots & A_{TT} \end{pmatrix} \tag{2}$$

and has dimension,$m \times n$, where $m = \sum_{t=1}^{T} m_t$ and $n = \sum_{t=1}^{T} n_t$.

Dantzig [15] introduced the idea of using an *artificial basis* for these programs in which square ($m_t \times m_t$) basis blocks appeared along the diagonal. The *true basis* could be derived from the artificial basis by a set of *side conditions*. He observed that, because of the *persistence* property in dynamic linear programs, the additional computations would be few.

This concept of basis factorization and the use of *pseudo- basic* variables, as in Beale[6], have been applied in a variety of examples. Recent implementations are found in Kallio and Porteus[38],Perold and Dantzig[48],Fourer[22], and Propoi and Krivonozhko[52]. This last approach is close to the development here, using local bases as a factorization for multi-stage linear programs. A thorough and unified presentation of the relationships among basis factorization, partitioning, and decomposition can also be found in Winkler[67].

2. The Structure of the SDLP Basis

The program,SDLP, is another member of the class of block triangular linear programs as the following shows.

Lemma 1. *The program, SDLP, has the structure of a block triangular linear program, as in (1).*

Proof. Define

$$A_{tt} = \begin{pmatrix} A_t & & & \\ & A_t & & \\ & & \ddots & \\ & & & A_t \end{pmatrix} \tag{3}$$

where A_t is repeated k_t times to correspond with each scenario in period t.

Define also

$$A^i_{t-1,t} = \begin{pmatrix} -B_{t-1} & & & \\ & -B_{t-1} & & \\ & & \ddots & \\ & & & -B_{t-1} \end{pmatrix} \tag{4}$$

where k_{t-1} repetitions of the matrix, $-B_{t-1}$, correspond to the possible outcomes of period $t-1$.

Then, we define $A_{t-1,t}$ as

$$A_{t-1,t} = \begin{pmatrix} A^1_{t-1,t} \\ A^2_{t-1,t} \\ \vdots \\ A^{k_t}_{t-1,t} \end{pmatrix}$$

The other A_{ij} matrices are void, so SDLP has the desired structure. ∎

The SDLP has other advantages that it shares with general multi-stage linear programs. In these programs, the number of additional columns required in finding the true basis from the artificial basis is limited. This bound means that storage requirements should not grow excessively with the problem size. This well-known property is stated in the following lemma:

Lemma 2. *For a dynamic linear program (a block triangular linear program where $A_{st} = 0$ for all $s > t+1$, ie. a staircase structure), the number of surplus and deficient columns in each block of the basis, B_{tt}, is bounded by:*

(i) $0 \leq l_1 \leq m_2$,

(ii) $-m_t \leq l_t \leq m_{t+1}, t = 2, \ldots, T-1$,

(iii) $l_T \geq -m_T$.

where $l_t + m_t$ is the number of basic columns in the basis from period t,and the basis has the form:

$$B = \begin{pmatrix} B_{11} & & & & \\ B_{21} & B_{22} & & & \\ 0 & B_{32} & B_{33} & & \\ & & & & \\ 0 & 0 & 0 & B_{TT-1} & B_{TT} \end{pmatrix}. \tag{5}$$

Proof. (ia). In order for B to be nonsingular B_{11} must have full rank. Therefore, $l_1 \geq 0$.

(ib) and (iia). $l_t \leq m_{t+1}$. Assume $l_t > m_{t+1}$, then B has greater than $m_t + m_{t+1}$ independent columns in period t, but the row rank of this block is less than or equal to $m_t + m_{t+1}$, a contradiction. Hence, $l_t \leq m_{t+1}$.

(iib) and (iii). $l_t \geq -m_t$. Assume $l_t < -m_t$, then there are greater than $\sum_{s=1}^{t} m_s$ independent columns in periods $1, 2, \ldots, t-1$, which again contradicts row rank, so $l_t \geq -m_t$.∎

The direct application of this lemma to the stochastic linear program, SDLP, would imply that for any period, t, there are at most $k_t \cdot m_{t+1}$ surplus columns in the basis. Because of the highly structured nature of SDLP, however, a much tighter bound can be found. We show this below in the following lemma. Murty [45] first observed this property for the two-stage case of programming under uncertainty with continuous distributions of the random variables in 1968.

Lemma 3. *For SDLP and for each scenario j in period t*

(i) $0 \leq l_1 \leq m_2$,

(ii) $-m_t \leq l_t^j \leq m_{t+1}, j = 1, \ldots, k_t; t = 2, \ldots, T-1$,

(iii) $-m_t \leq l_t^j; j = 1, \ldots, k_T$.

where l_t^j is the number of surplus columns in the basis for period t and scenario j, and m_t is the number of rows for a single scenario in period t.

Proof. (ia) This follows from Lemma 2.

(ib) and (iia). Assume that $l_t^j > m_{t+1}$. This implies that there are greater than $m_t + m_{t+1}$ independent columns) in the matrix :

$$D_t = \begin{pmatrix} A_t \\ -B_t \\ -B_t \\ \vdots \\ -B_t \end{pmatrix} \tag{6}$$

but, again, D_t has row rank $m_t + m_{t+1}$, a contradiction. Therefore, $l_t^j \leq m_{t+1}$.

(iib) and (iii). As before, the number of independent columns cannot exceed the row rank of the submatrix including that scenario, and the result follows.∎

From this result, we proceed to find an efficient implementation of the Simplex Method to the problem, SDLP, in which, a limited number of additional computations are used to perform the simplex routines of finding the true basis representation of a column, the cost row, and prices.

3. Finding the true basis representation of a column

To find the column representation, first define a square block triangular artificial basis for SDLP as

$$U = \begin{pmatrix} U_1 & & & & & \\ V_1 & U_2^1 & & & & \\ V_1 & & U_2^2 & & & \\ & V_2^1 & & \cdot & & \\ & V_2^2 & & & \cdot & \\ & & & & \vdots & \\ & & & & V_{T-1}^{k_T-1} & U_T^{k_T} \end{pmatrix} \tag{7}$$

where U_t^j consists of the rows of A_t^j for m_t independent columns in each period t, and V_t includes the corresponding rows in $-B_t$. For a constraint matrix, A, we also define

$$\alpha \equiv \{a_j \in A : a_j \text{ is a column in the artificial basis}\}.$$

We further define

$$\tau \equiv \{a_j \in A : a_j \text{ is a column in the true basis }\}.$$

The complements of α and τ are defined as $\overline{\alpha}$ and $\overline{\tau}$.

For every U_t, there also is a partition

$$U_t^j = \begin{pmatrix} P_t^j & \tilde{T}_t^j \\ \tilde{P}_t^j & T_t^j \end{pmatrix} \tag{8}$$

where P_t^j and T_t^j are square and non-singular, the columns in P_t^j are *pseudo-basic*, chosen from $\alpha \bigcap \overline{\tau}$, and T_t^j contains columns from the true basis. We define here

$$\psi \equiv \{k : a_j \in \alpha \bigcap \overline{\tau} \text{ where } a_j \text{ is the basic column for row } k\}.$$

To replace these columns in the basis at period t, there must be *surplus* basic variables from period $t-1$. We write these columns in terms of the

artificial basis as

$$U^{-1}\begin{pmatrix} A^j_{t-1} \\ -B^j_{t-1} \end{pmatrix} = \begin{pmatrix} P^j_{t-1} \\ Q^j_{t-1} \\ R^j_{t-1} \end{pmatrix} \tag{9}$$

where P^j_{t-1} consists of rows from A^j_{t-1} and is $m_{t-1} \times s^j_{t-1}$ (the number of surplus columns in period $t-1$ under scenario j). The rows corresponding to $-B^j_{t-1}$ are partitioned between Q^j_{t-1}, an $s^j_{t-1} \times s^j_{t-1}$ matrix, and R^j_{t-1}, a $\left(k_t \cdot m_t - s^j_{t-1}\right) \times s^j_{t-1}$ matrix.

The coefficient matrix,A, is then

$$U^{-1}A = \begin{pmatrix} P_1 & & & & & & \\ & P_2^1 & & & & & \\ Q_1 & & \cdot & & & & \\ & & & P_2^{k_2} & & & \\ & P_2^1 & & & & & \\ R_1 & & \cdot & & & & \\ & & & P_2^{k_2} & & & \\ & Q_2^1 & & & & & \\ I & & \cdot & & & & U^{-1}\tilde{A} \\ & & & Q_2^{k_2} & & & \\ & R_2^1 & & & & & \\ & & \cdot & & & & \\ & & & R_2^{k_2} & & & \\ & & & \vdots & & & \\ & & & & \cdot & Q_{T-1}^{k_T} & \\ & & & & \cdot & & \\ & & & & & R_{T-1}^{k_T} & \end{pmatrix} \tag{10}$$

which, by row and column permutation, is

$$
\begin{pmatrix}
 & P_2^1 & & & & & & & \\
Q_1 & & \cdot & & & & & & \\
 & & & P_2^k & & & I & & \\
 & Q_2^1 & & & & & & & \\
 & & \cdot & & & & & & \\
 & & & Q_2^{k_2} & & & & & \\
 & & & & \cdot & & & & \\
 & & & & & P_{T-1}^{k_{T-1}} & & & \\
 & & & & & \vdots & & & \\
 & & & & & Q_{T-1}^{k_{T-1}} & & & \\
P_1 & & & & & & & U^{-1}\tilde{A} & \\
R_1 & & & & & & & & \\
 & & \cdot & & & & I & & \\
 & & & \cdot & & & & & \\
 & & & & & P_{T-1}^{k_{T-1}} & & & \\
 & & & & \cdot & & & & \\
 & & & & & R_{T-1}^{k_{T-1}} & & &
\end{pmatrix}
\tag{11}
$$

The following allows us to use this partitioning for computations.

Lemma 4. *Relative to the artificial basis, U, the true basis, T, of SDLP has columns partitioned as in (11), where each Q_t^j is square and nonsingular.*

Proof. By the definition of the surplus columns in (9) and suitable permutations, we arrived at (11).

Next, define Q as the matrix, relative to the artificial basis, of surplus columns with rows in ψ, that is

$$Q = \begin{pmatrix} & P_2^1 & & \\ Q_1 & & \cdot & \\ & Q_2^1 & & \cdot \\ & & \cdot & P_{T-1}^{k_{T-1}} \\ & & & \cdot \\ & & & Q_{T-1}^{k_{T-1}} \end{pmatrix}$$

which is clearly nonsingular because U^{-1} and T are both nonsingular.

Now, index Q_t^j for all j and t along the diagonal as $Q_1, \ldots, Q_k$. Let Q_p be the first singular matrix in the list. Its row rank is $r_p < s_p$, the number of columns. In order for Q to be nonsingular, however, the row rank of the submatrix of the remaining columns, $\bar{r}_{p+1}$, is

$$\bar{r}_{p+1} > \sum_{j=p+1}^{k} s_j.$$

This violates the column rank, therefore, each Q_t^j is nonsingular.∎

Given that the Q_t^j's are nonsingular, we can proceed to eliminate the pseudo–basic columns from the basis. This procedure involves premultiplying $U^{-1}A$ by a product of matrices, $F_1, F_2^1, \ldots, F_{T-1}^{K_{T-1}}$. We first define

$$F_1 = \begin{pmatrix} Q_1^{-1} & & \\ & \cdot & \\ & & I \\ P_1 Q_1^{-1} & & \\ R_1 Q_1^{-1} & & \end{pmatrix} \tag{12}$$

and observe that

$$F_1U^{-1}A = \begin{pmatrix} I & \overline{P}_2^1 & & & Q_1^{-1} & \\ & Q_2^1 & & & & I \\ & & Q_{T-1}^{k_{T-1}} & & & \\ & & & & -P_1Q_1^{-1} & \\ & & & & -R_1Q_1^{-1} & \\ & & & I & & \end{pmatrix} \tag{13}$$

We next define

$$F_2^1 = \begin{pmatrix} I & -\overline{P}_2^1(Q_2^1)^{-1} & \\ & (Q_2)^{-1} & \\ & -\overline{R}_2^1(Q_2^1)^{-1} & I \end{pmatrix}$$

and, in general,

$$F_t^l = \begin{pmatrix} I & -\overline{P}_t^l(Q_t^l)^{-1} & \\ & (Q_t^l)^{-1} & \\ & -\overline{R}_t^l(Q_t^l)^{-1} & I \end{pmatrix} \tag{14}$$

where $\overline{R}_t^l$ and $\overline{P}_t^l$ are the partitions of the surplus columns, relative to the previous F_s^j's. By repeated multiplication, the matrix relative to the true basis is found:

$$F_{T-1}^{k_{T-1}} \cdots F_1U^{-1}A = \begin{pmatrix} I & & \\ & F_{T-1}^k \cdots F_1 & F_{T-1}^{k_{T-1}} \cdots F_1U^{-1}\tilde{A} \\ & I & \end{pmatrix} \tag{15}$$

In order to facilitate finding the matrices, F_s^j, we note that the growth of nonzeroes in every period and scenario is limited. The following lemma states this.

Lemma 5. *After multiplication by $F_t^{l-1}, F_t^{l-2}, \ldots, F_t^1, \ldots, F_1$, the additional nonzero blocks in the surplus columns of scenario l in period t occur only in rows, for which, l's ancestor scenarios are basic.*

Proof. We proceed by induction on p, where we order all the blocks of surplus columns with $p = 1, \ldots, k$. For $p = 1$, the result is true trivially since no new nonzero blocks are added.

Assume this is true for all $p \leq n$. We look at the surplus columns in $n+1$, and assume, without loss of generality, that this is scenario l in period t. We then define

$$F_n \cdots F_1 \cdot \begin{pmatrix} P_t^l \\ Q_t^l \\ R_t^l \end{pmatrix} \equiv S_t^l \tag{16}$$

Next observe that every F_p in periods $1, \ldots, t-2$ has only identities in columns corresponding to P_t^l, Q_t^l, and R_t^l, so that only the F_{t-1}^l need be considered. If the scenarios are not ancestors of l at period t, then again by the hypothesis, they have no block entries that correspond to R_t^l, Q_t^l, or P_t^l, and so do not alter S_t^l.

Therefore, we have

$$S_t^l = F_{t-1}^l \cdot \begin{pmatrix} P_t^l \\ Q_t^l \\ R_t^l \end{pmatrix} = \begin{pmatrix} I & -P_{t-1}^l (Q_{t-1}^l)^{-1} & \\ & (Q_{t-1}^l)^{-1} & \\ & -R_{t-1}^l (Q_{t-1}^l)^{-1} & I \end{pmatrix} \begin{pmatrix} P_t^l \\ Q_t^l \\ R_t^l \end{pmatrix} \tag{17}$$

and the only additional blocks occur where $-P_{t-1}^l (Q_{t-1}^l)^{-1} P_t^l$ is nonzero. By the hypothesis, the only nonzero blocks in S_{t-1}^l are in its ancestor rows from previous periods, so the only additional nonzero blocks in S_t^l will occur in these rows and the rows where columns are surplus in $t-1$, scenario l. This completes the induction. ∎

This lemma proves valuable in computing the columns relative to the true basis . Multiplications and storage are limited since only sections of each surplus block need to be considered. We next wish to compute the value of a column, z_i, where $z_i \in \overline{r}(t,l)$, the non -basic columns in period t and under scenario l.

We first observe that

$$F_n \cdots F_1 U^{-1} z_i = F_t^l \cdot F_{t-1}^{\bar{l}} \overline{z_i} \equiv \overline{z_i^*}, \tag{18}$$

where $\bar{l}$ is the first ancestor scenario of l, since for every F_j^p such that $j \notin \{t, t-1\}$ or $p \neq l$

$$F_j^p \overline{z_i} = \overline{z_i} \tag{19}$$

by the definition of F_j^p.

Next partition the components of the vector $\overline{z}_i$ as

$$\overline{a}_i^p \equiv (\overline{z}_{ji} : j \in \psi(t))$$

$$\overline{b}_i^p \equiv (\overline{z}_{ji} : j \in \psi(t+1))$$

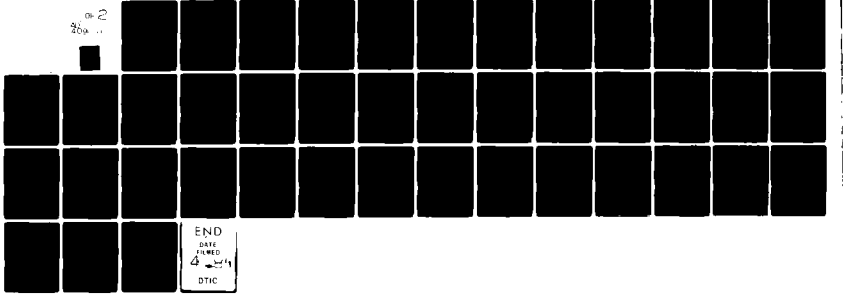
$$\overline{a}_i^o = (\overline{z}_{ji} : j \in \overline{\psi}(t))$$

$$\overline{b}_i^o = (\overline{z}_{ji} : j \in \overline{\psi}(t+1)). \tag{20}$$

Then, $\overline{z}_i^*$ can be written as

$$\overline{z}_i^* = \begin{pmatrix} -P_{t-1}^l (Q_{t-1}^l)^{-1} \\ (Q_{t-1}^l)^{-1} \\ \\ -R_{t-1}^l (Q_{t-1}^l)^{-1} \end{pmatrix} \cdot \overline{a}_i^p + \begin{pmatrix} -P_t^l (Q_t^l)^{-1} \\ (Q_t^l)^{-1} \\ \\ -R_t^l (Q_t^l)^{-1} \end{pmatrix} \cdot \overline{b}_i^p + \begin{pmatrix} \overline{a}_i^o \\ \overline{b}_i^o \end{pmatrix} \tag{21}$$

AD-A096 119 STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB F/G 12/2
SOLUTION METHODS FOR STOCHASTIC DYNAMIC LINEAR PROGRAMS.(U)
DEC 80 J R BIRGE N00014-75-C-0267
UNCLASSIFIED SOL-80-29 NL



END
DATE FILMED
4-81
DTIC

With this representation, we form the following procedure, called **LBFTRN** (for local basis forward transformation) to find $\overline{z}_i^*$.

LBFTRN

Step 0. Identify an incoming column z_i. Find

$$\overline{z}_i = ((U_t^l)^{-1}a_i; (U_{t+1}^l)^{-1}(V_t^l((U_t^l)^{-1}a_i + b_i)) = (\overline{a}_i; \overline{b}_i) \text{as defined in (20).} \tag{22}$$

Step 1. Identify $\overline{a}_i^p$ and $\overline{b}_i^p$. Find

$$\begin{aligned} \overline{w} &\equiv (Q_{t-1}^l)^{-1}\overline{a}_i^p, \\ \overline{v} &\equiv (Q_{t-1}^l)^{-1}\overline{b}_i^p. \end{aligned} \tag{23}$$

Step 2. For all $j \in \bigcup_{s=1}^{t-2} \psi(s)$ (all preceding pseudo–basic columns) find

$$\overline{z}_{ji}^* = -\overline{P_{t-1}^l}(j,*) \cdot \overline{w} - \overline{P_t^l}(j,*) \cdot \overline{v} \tag{24}$$

for $j \in \psi(t-1)$,

$$\overline{z}_{ji}^* = \overline{w} - \overline{P_t^l}(j,*)\overline{v}, \tag{25}$$

for $j \in \psi(t)$,

$$\overline{z}_{ji}^* = \overline{v}, \tag{26}$$

for $j \in \bigcup_{s=1}^{t-2} \overline{\psi}(s)$,

$$\overline{z}_{ji}^* = -R_{t-1}^l(j,*)\overline{w} - R_t^l(j,*)\overline{v}, \tag{27}$$

and for $j \in \overline{\psi}(t-1)$,

$$\overline{z}_{ji}^* = -R_t^l(j,*)\overline{v}. \tag{28}$$

Step 3. For $j \in \overline{\psi}(t,l)$,

$$\overline{z}_{ji}^* = \overline{z}_{ji} + \overline{a}_{ji}^o, \tag{29}$$

and for $j \in \overline{\psi}(t+1, l)$

$$z^*_{ji} = z_{ji} + b^o_{ji}. \tag{30}$$

LBFTRN results in the updated vector, $\bar{z}^*_i$. Therefore, from (21) and our previous results, the following proposition holds.

Proposition 1. *The representation of an incoming variable, z_i, relative to the current basis in SDLP can be found as $\bar{z}^*_i$ from the above procedures in LBFTRN.*

This routine allows for quick computations of an incoming column when there are few surplus columns. It requires current information of which rows belong to $\psi(t)$ and $\psi(t+1)$, the pseudo–basic columns in period t and period $t+1$, the representations of P^l_{t-1}, P^l_t, R^l_{t-1}, R^l_t, and the inverses of Q^l_{t-1} and Q^l_t. This storage requirement would be small, relative to the storage of the entire matrix, if few surplus columns are present.

4. Finding the Dual Prices and Pricing

The backwards transformation involved in computing the dual prices for SDLP can also be done more efficiently with the use of a square block triangular artificial basis. A savings here could be substantial because the pricing operation aften requires a majority of the effort in linear programming codes (viz., Fourer [22]). The method presented below requires only the present and previous scenario blocks' surplus column updates for computation of the reduced costs.

We assume again that we are in period t and at scenario l. The dual prices relative to the artificial basis are computed first, by transformations from the last period to the beginning. We define

$$\pi^l_T = c^u_T (U^l_T)^{-1} \text{for all } l = 1, \ldots, k_T. \tag{31}$$

and define π_t^l by the general recursion:

$$\pi_t^l = \left(c_t^u - [\sum_{l(t+1)} p_{t+1}^{l(t+1)} \pi_{t+1}^{l(t+1)}] V_t^l \right) (U_t^l)^{-1} \tag{32}$$

for all $l = 1, \ldots, k_t$, where $l(t+1)$ denotes all descendants of $l(t)$ at $t+1$ and $p_{t+1}^{l(t+1)}$ is the probability of each descendant.

Now, to find the prices relative to the true basis, we look for ρ such that

$$0 = c_j - \rho \cdot a_j \text{ for all } j \in \tau, a_j \in A. \tag{33}$$

We have from π defined in (32) and (33) that

$$0 = c_j - \pi \cdot a_j \text{ for all } j \in \tau \bigcap \alpha, \tag{34}$$

or, since $\pi = cU^{-1}$,

$$\begin{aligned} 0 &= c_j - c(U^{-1} a_j) \\ &= c_j - c_j \cdot I_j, \end{aligned} \tag{35}$$

where I_j is a unit column with identity in row j.

Thus, for c partitioned as

$$c^Q = (c_j : j \in \tau \bigcap \alpha), \text{and}$$

$$c^R = (c_j : j \in \tau \bigcap \alpha), \tag{36}$$

we have from (35) and the definition of Q and R in (9),

$$\delta_j = c_j - \sum_{i \neq j} c^{Q_i} \cdot P_i - c^{Q_j} \cdot Q_j - c^R \cdot R_j \text{ for all } j \in \tau \bigcap \bar{\alpha}. \tag{37}$$

We seek next σ such that

$$0 = c_j - \sum_{i \neq j} (c_i^Q + \sigma_i) P_j - (c^{Q_j} + \sigma_j) Q_j - c^R R_j \text{ for all } j \in \tau \bigcap \bar{\alpha}. \tag{38}$$

We do this iteratively by finding first

$$\sigma_1 = \delta_1 Q^{-1} \tag{39}$$

and, in general,

$$\sigma_t^{l(t)} = [\delta_t^l - \sigma_t^{l(t-1)} P_t^l](Q_t^l)^{-1} \text{ for all } l = 1, \ldots, k_t. \tag{40}$$

We define ρ as

$$\rho = \begin{cases} c_j + \sigma_j, & \text{if } j \in \psi \\ c_j, & \text{if } j \in \overline{\psi}. \end{cases} \tag{41}$$

Hence, (33) holds by the construction of σ in (39) and (40) and the preservation of $\overline{c}^R = 0$.

The computation of ρ, using σ as defined in (39), allows pricing in an individual period t to involve only the inverse of Q_t^l. If pricing proceeds from period t to $t+1$, then, for a constant artificial basis, the previous prices need not be recomputed. This strategy may result in substantial savings by eliminating unnecessary multiplications for the other periods. It also saves on storage, since extraneous surplus block inverses can be ignored.

The local basis simplex procedures for finding dual prices are then:

LBBTRN

Step 0. For period t and scenario l, $c_\tau = (c_j(t,l) : j \in \tau \bigcap \overline{\alpha})$.

Step 1. Find $\delta_j(t,l)$ as defined in (37) for all $j \in \tau(t,l) \bigcap \overline{\alpha}$.

Step 2. Compute σ_t^l as in (40).

Step 3. Form ρ as in (41).

This procedure is then followed by LBPRCE which finds

$$\overline{c}_j = c_j - \rho \cdot a_j \tag{42}$$

for all $j \in \tau(t,l)$. The incoming variable in this strategy is then chosen as the variable with the most negative reduced cost in that period and scenario. We define this as

$$\bar{c}_s(t,l) = \min_{j \in \tau(t,l)} c_j - \rho \cdot a_j. \tag{43}$$

If $\bar{c}_s(t,l) \geq 0$, then the algorithm would proceed to the next scenario in period t, or, if $l = k_t$, the first scenario in period $t+1$ would be considered.

5. Updating the Pseudo-Bases and Surplus Blocks

After the incoming column is chosen in (43) a leaving column is selected by the minimum ratio criterion of the standard simplex method. This procedure, called CHUZR, finds

$$\frac{\bar{z}^*_{rs}}{\bar{\xi}_r} = \min_{\bar{z}^*_{is} > 0} \left(\frac{\bar{z}^*_{is}}{\bar{\xi}_i}\right), \tag{44}$$

where $\bar{z}^*_s$ is the representation of the incoming column found by LBFTRN and $\bar{\xi}_i$ is the current value of the right hand side in the i^{th} row.

The computations in CHUZR can also be reduced because nonzero entries in $\bar{z}^*_s$ are restricted to certain blocks as we discussed in Section 3. For updating the basis, s can be pseudo– basic or non–basic in period t (ie.,$s(t,l) \in \alpha \bigcap \tau(t,l)$ or $s(t,l) \in \bar{\alpha} \bigcap \tau(t,l)$), and s can enter the true artificial basis in period t or become surplus basic in period $t+1$ (that is, $r \in \tau(t-1,l) \bigcap \bar{\alpha}, \tau(t+1,l) \bigcap \alpha, \tau(t+1,l) \bigcap \bar{\alpha}$,or $\tau(t+1,l) \bigcap \alpha$). We discuss these cases below :

U1. $r \in \tau(t-1,l)$.

In this case s replaces a surplus column from the previous period. To update, we remove a column and row from $Q(t-1,l)$ and update the corresponding list of pseudo–vectors in period t. We also update $U(t,l)^{-1}$ by replacing the pseudo–basic column, $\psi(r)$, that corresponded to r with s.

This is done by adding an elementary matrix to the eta file of $U(t,l)^{-1}$ as in standard simplex codes. (Note that if $s \in \alpha$, the artificial basis remains unchanged.)

U2. $r \in \tau(t,l) \bigcap \alpha$.

Here, s replaces a currently true basic column in the artificial basis. If $s \in \alpha$, then we must replace the row in $Q(t,l)$ for which s is basic with the row for which r was basic. The artificial basis is unchanged. If $s \in \overline{\alpha}$, then we need only update the artificial basis, $U(t,l)$, as in **U1.**

U3.$r \in \tau(t+1,l) \bigcap \overline{\alpha}$.

In this case, s replaces a surplus column which is basic in the next period. We maintain the same artificial basis and update the surplus block,$Q(t+1,l)$. To do this, the column occupied by r in $Q(t+1,l)$ is replaced by the corresponding row entries of s. This is performed easily by premultiplying by an elementary matrix for this pivot.

U4. $r \in \tau(t+1,l) \bigcap \alpha$.

In this case, s replaces a basic column in the next period. This involves adding a column and row to $Q(t+1,l)$. The new row has entries from each of the surplus basic columns. The list of pseudo–basic variables in the next period must also be updated with the addition of the row for which r was basic. Again, the artificial basis remains unchanged.

The updating procedure can be confined to the current local artificial bases and current and following pseudo–bases. This property enables us to store only the present and following bases for updating purposes.

6. The Algorithm

The previous sections have presented the basic routines for a local basis simplex method. We discuss below the method's basic strategy and im-

plementation. The algorithm is similar to the nested decomposition approach of Chapter III in that it alternately follows a forward and backward procedure through the periods. This strategy is also similar to dynamic programming since the algorithm proceeds from one local optimum to the next. The basic method, call LBSMPX, follows.

LBSMPX

Step 0. Find an artificial basis. Do this by solving first

$$\begin{aligned} &\min\ c_1 x_1 \\ &\text{s. t. } A_1 x_1 = \xi, \\ &\qquad x_1 \geq 0, \end{aligned} \tag{45}$$

and proceeding to solve

$$\begin{aligned} &\min\ c_t x_t^l \\ &\text{s.t. } A_t x_t^l = \xi_t^l + B_{t-1} x_t^{l(t-1)}, \\ &\qquad x_t^l \geq 0, \end{aligned} \tag{46}$$

for all $(t,l), t = 1,\ldots,T, l = 1,\ldots,k_t$. If any of these programs has no feasible solution, use the last infeasible solution basis as U_t^l. Otherwise, we store each locally optimal basis as U_t^l and form the artificial basis from these matrices.

Step 0. Set $t = T-1, l = 1, NDRCTN = 'BACK', CFLAG = 'NO', MSTAT = 'YES', ITNO = 0$.

Step 1. Invert the current basis and find all basic variables values. This procedure, called INVERT, is equivalent to performing LBFTRN on the right hand side. If all variables have feasible values, $MSTAT = 'YES'$. Else, $MSTAT = 'NO'$.

*Step 2.*Form a Phase I objective row, if $MSTAT = 'NO'$. Else, use the objective row,c_t. This routine is FORMC.

Step 3. Call LBBTRN for (t,l) to find the current prices. Set $ITNO = ITNO + 1$.

Step 4. Call PRICE for (t,l). If $\bar{c}_s \geq 0$, check basic variable values. If there exists $x(t', l')$ infeasible, set $MSTAT = 'NO'$, and

(a) If $t = T, l = k_T, NDRCTN = 'FORE'$,and $CFLAG = 'NO'$, if $MSTAT = 'YES'$, the solution is optimal,stop. If $MSTAT = 'NO'$ and the objective value, $x(1,1) > 0$,then the solution is infeasible, stop. If $x(1,1) = 0$, then, set $MSTAT = 'YES'$, and go to Step 2.

(b) If $t = 1, l = 1, NDRCTN = 'BACK'$, and $CFLAG = 'NO'$, then if $MSTAT = 'YES'$, the solution is optimal, stop. Else, if the objective $x(1,1) > 0$, then there is no feasible solution, stop. If $x(1,1) = 0$, then set $MSTAT = 'YES'$ and go to Step 2.

(c) If $t = T, l = k_T, NDRCTN = 'FORE'$, and $CFLAG = 'YES'$,set $t = T-1, l = 1, NDRCTN = 'BACK', CFLAG = 'NO'$,set $MSTAT = YES$. If $ITNO < MAXIT$(the maximum number of iterations between reinversions),go to Step 2, else go to Step 1.

(d) If $t = 1, l = 1, NDRCTN = 'BACK'$, and $CFLAG = 'YES'$, then set $t = 2, l = 1, NDRCTN = 'BACK', CFLAG = 'NO', MSTAT = 'YES'$. If $ITNO < MAXIT$, go to Step 2. Else, go to Step 1.

(e) If $t < 1$ and $l < k_t$, then set $l = l + 1$. If $ITNO < MAXIT$, go to Step 2,else go to Step 1.

(f) If $1 < t < T, l = k_t, NDRCTN = 'BACK'$, set $t = t-1, l = 1$, go to Step 2.

(g) If $1 < t < T, l = k_t, NDRCTN = 'FORE'$, set $t = t+1, l = 1$, go to Step 2.

If $\bar{c}_s < 0$, set $CFLAG = 'YES'$.

Step 5. Call LBFTRN.

Step 6. Call *CHUZR*. If no r is found (ie., all $\bar{z}^*_{is} \leq 0$), then the solution is unbounded, stop.

Step 7. Call UPDATE to perform U1, U2, U3, or U4, depending on r, go to Step 2.

The preceding method leads to an optimal, unbounded, or infeasible solution as the following theorem states.

Theorem 1. *Assuming nondegeneracy (or suitable resolutions by choice of outgoing variable in CHUZR (see Dantzig [17])), the method, LBSMPX, terminates in a finite number of steps with an optimal solution, an unbounded feasible solution or an infeasibility criterion.*

Proof. Each pass through Step 7 results in a decreased objective value under nondegeneracy. Since there are a finite number of bases, the algorithm can pass Step 7 at most a finite number of times. The method would, therefore, terminate as in the simplex method as long as it takes at most a finite number of steps between passes of Step 7.

The algorithm does not return to Step 7 for each iteration that Step 4 results in (c), (d), (e), (f), or (g). However, in these cases, the algorithm progresses to the next scenario or period in either the forward or backward phase. Therefore, conditions (a) or (b) in Step 4 must be met after each phase.

When (a) or (b) are encountered in Step 4, the algorithm proceeds in the opposite direction, if Step 7 was ever encountered in the last pass through every scenario (the case, $CFLAG = 'YES'$). If no improvement was made, that is, if Step 7 was not passed in the last phase, the algorithm terminates. Therefore, LBSMPX always terminates in a finite number of steps. ∎

As in the Simplex Method, variations in LBSMPX may be used. The

most negative pricing strategy in PRICE, for instance, can be altered. We can also choose to proceed to the next period with a criterion other than $\bar{c}_s(t,l) \geq 0$. $\bar{c}_s(t,l) \geq -\epsilon$ could be used with ϵ decreasing in size as the optimal solution is approached.

LBSMPX uses the standard simplex techniques by efficiently partitioning the basis. As in the nested decomposition approach, NDSDLP, and the piecewise method, PCSDLP, LBSMPX concentrates on different sections of the basis one at a time. It differs with the previous methods, however, by continuously reflecting changes in the entire problem. The advantage of this property is that the global solution reflects the local optimization more quickly. The disadvantages, however, are that the method requires more computational effort in maintaining these instantaneous changes. These relative computational requirements are, again, discussed more closely in Chapter VII.

Chapter VI

The Relationship to

Dynamic Programming

1. Introduction

In Chapter I, the dynamic programming formulation for the general stochastic dynamic linear program, SDLP, was presented. In principle, SDLP could be solved exactly by this method. Since the random vectors, ξ_t, were allowed to have continuous distributions, the program at stage t was an infinite dimensional optimization problem. To avoid this complication, we approximated the distribution with discrete valued random vectors, ξ_t^i, and formulated a linear program, the solution of which we discussed above.

We considered linear programming techniques, but we could have used dynamic programming to solve this discrete distribution problem. In a production example, Beale, Forrest, and Taylor [7] estimated that four state variables in a time period could be handled. For larger problems, they concluded, an approximation must be used.

We chose the linear programming formulation because of the historically good performance of the Simplex Method and simplex–based algorithms. Using these algorithms, large problems with over a thousand rows and columns can be solved easily (see White [66] for a discussion of computational experience with large–scale linear optimization). Dynamic programming techniques using the standard computational procedure are generally limited to problems with six state variables and six decision variables (see Larson and Casti [41]). Advanced techniques can, however, be implemented

for larger problems in order to approximate and then refine the state and policy space. These methods prove very valuable for general transition equations and objective functions (see Larson [39]), but linear programming methods are most commonly used for problems with linear constraints and objectives. They do not require quantization of the state space or knowledge of the range of state variable values along the optimal path. Linear programming is, therefore, often more efficient than dynamic programming–type algorithms.

Our methods in Chapter III and IV actually combine these two techniques, although we have presented them as linear optimization strategies. In the following sections, a standard dynamic programming formulation of the stochastic linear program will be presented and its advantages will be discussed. We will then show that the piecewise method, PCSDLP, and the nested decomposition approach, NDSDLP, are simply different versions of general dynamic programming.

2. The Quantized State Space Approach

A standard simplification in dynamic programming is *quantization*. SDLP can be formulated in this manner by first creating a discrete approximation of the state space. This will enable us to form a recursion at every stage t and for every state y_t. We first define

$$y_t \equiv B_t x_t. \tag{1}$$

and quantize this vector as $\{y_t^1, y_t^2, \ldots, y_t^{l_t}\}$. The random right–hand sides will again have discrete values, $\{\xi_t^1, \xi_t^2, \ldots, \xi_t^{k_t}\}$.

Now, the following backward algorithm, BDP, can be used.

BDP

Step 0. For all y_T^i, find

$$z_T(y_{T-1}^i) = \mathrm{E}_{\xi_T}[z_T(y_{T-1}^i, \xi_T)], \tag{2}$$

where

$$\begin{aligned} z(y_{T-1}^i, \xi_T) = \ & \min \ c_T x_T \\ & \text{subject to} \\ & A_T x_T = y_{T-1}^i + \xi_T, \\ & x_T \geq 0. \end{aligned} \tag{3}$$

Set t=T. Go to Step 1.

Step 1. Set $t = t - 1$. For all y_{t-1}^j, find

$$z_t(y_{t-1}^j) = \mathrm{E}_{\xi_t}[z_t(y_{t-1}^j, \xi_t)], \tag{4}$$

where $z_t(y_{t-1}^j, \xi_t) = \min_{y_t^k}\{z_t((y_{t-1}^j, \xi_t), y_t^k)\}$, where

$$\begin{aligned} z_t((y_{t-1}^j, \xi_t), y_t^k) = \ & \min \ c_t x_t + z_{t+1}(y_t^k) \\ & \text{subject to} \\ & A_t x_t = \xi_t + y_{t-1}^j, \\ & -B_t x_t = y_t^k, \\ & x_t \geq 0. \end{aligned} \tag{5}$$

Step 2. If t=2, go to Step 3. Else, go to Step 1.

Step 3. Find $z_1 = z_1(0, b_1) = \min_{y_1^j} z_1((0, b_1), y_1^j)$, where $z_1((0, b_1), y_1^j)$ is as defined in (5) for $y_0 = 0$, $\xi_1 = b_1$. Stop.

Proposition. *The algorithm, BDP, converges in a finite number of steps to an optimal solution, $x_t^*(y_{t-1}^j)$, for $z_t(y_{t-1}^j)$, at every stage t of the program SDLP, under the following assumptions :*

I. The support of the random vector, ξ_t, is $\Xi_t = \{\xi_t^1, \xi_t^2, \ldots, \xi_t^k\}$.

II. The values of the inventories from period t are $B_t x_t$, where $B_t x_t \in Y_t = \{y_t^1, y_t^2, \ldots, y_t^l\}$.

Proof. The result follows directly from Bellman's Principle of Optimality and the finiteness of the Simplex Method for linear programming. ∎

BDP requires the solution of a great number of linear programs in Step 1. The storage requirements may be prohibitive for a large or detailed state space, Y_t. BDP does, however, have the advantage of solving small subproblems and of following a single pass to optimality recursively from period t back to 1. The method may even be implementable when one has a great deal of advance information about the admissible states in the solution. For more general problems, however, more efficient ways to characterize the state space are necessary. In the next section, we show that the algorithms, PCSDLP and NDSDLP, of Chapters III and IV are such methods, employing approximations for a continuous state space.

3. Relation to the Nested Decomposition Method

In this analysis, we consider the optimization problem, DP(t), at some time t :

$$z_t(y_{t-1}) = \mathrm{E}_{\xi_t}[z_t(y_{t-1}, \xi_t)], \tag{6}$$

where $z_t(y_{t-1}, \xi_t) = \min_{y_t} z_t((y_{t-1}, \xi_t), y_t)$ and

$$z_t((y_{t-1}, \xi_t) = \min c_t x_t + z_{t+1}(y_t)$$
subject to

$$\begin{aligned} A_t x_t &= y_{t-1} + \xi_t, \\ -B_t x_t &= y_t, \\ x_t &\geq 0. \end{aligned} \tag{7}$$

Here, we would like to have an explicit representation of $z_{t+1}(y_t)$, which we approximated with the quantizations above. One way to find $z_{t+1}(y_t)$ is to exploit its convexity properties. The nested decomposition and piecewise approaches do this, as we show below.

For the nested decomposition approach, the following problem at stage t is solved instead of DP(t) :

$$\begin{aligned} &\min c_t x_t \quad + \theta_t \\ &\text{subject to} \\ &\quad A_t x_t \qquad = y_{t-1} + \xi_t, \\ &-(\pi_{t+1}^k B_t) x_t + \theta_t \geq \rho_{t+1}^k, k = 1, \ldots, p, \qquad (ND(t)(a)) \\ &(-\sigma_{t+1}^l B_t) x_t \quad \geq (\sigma_{t+1}^l \tilde{\alpha}_{t+1}), l = 1, \ldots, q, (ND(t)(b)) \\ &\qquad\qquad x_t \geq 0, \end{aligned} \tag{ND(t)}$$

where the constraints ND(t)(b) keep x_t feasible and ND(t)(a) forms an outer linearization of the convex function, $Q(x_t)$, as we showed in Chapter 3. The following lemma then shows the equivalence of DP(t) and ND(t).

Lemma 1. *For all t, $1 \leq t \leq T$, x_t^* is the optimal solution of $z_t(y_{t-1}, \xi_t)$ if and only if x_t^* solves:*

$$\begin{aligned} \min \quad & c_t x_t + Q(x_t) \\ \text{subject to } & A_t x_t = y_{t-1} + \xi_t, \\ & x_t \geq 0. \end{aligned} \tag{8}$$

Proof. For $t = T$, we have

$$\begin{aligned} Q_T(x_{T-1}) &= \mathrm{E}[Q_T(x_{T-1}, \xi_T)] \\ &= \mathrm{E}_{\xi_T}[\min c_T x_T | A_T x_T = \xi_T + B_{T-1} x_{T-1}, x_T \geq 0] \\ &= z_T(B_{T-1} x_{T-1}). \end{aligned} \tag{9}$$

At $t = T - 1$,

$$Q_{T-1}(x_{T-2}, \xi_{T-1}) = \min c_{T-1} x_{T-1} + Q_T(x_{T-1}) \tag{10}$$

subject to

$$\begin{aligned} A_{T-1} x_{T-1} &= \xi_{T-1} + B_{T-2} x_{T-2}, \\ x_{T-1} &\geq 0, \end{aligned}$$

which, by (9), is equivalent to :

$$\begin{aligned} \min \quad & c_{T-1} x_{T-1} + z_T(y_{T-1}) \\ \text{subject to} \quad & \\ & A_{T-1} x_{T-1} = \xi_{T-1} + B_{T-2} x_{T-2}, \\ & B_{T-1} x_{T-1} = y_{T-1}, \\ & x_{T-1} \geq 0. \end{aligned} \tag{11}$$

Therefore, the functions $z_{T-1}(y_{T-2}, \xi_{T-1})$ and $Q_{T-1}(x_{T-2}, \xi_{T-1})$ are identical and have correspondingly the same optimal solutions, x^*_{T-1}. The result follows by induction on t.∎

Lemma 1 shows that the cutting plane method used in the nested decomposition approach is equivalently a method for finding the function, $z_t(y_{t-1})$, in DP(t). In doing this, the constraints in ND(t)(a) linearize the convex function in the neighborhood of y_{t-1}.

In other words, at stage $t + 1$, for a given y_t^j, we find

$$
\begin{aligned}
z_{t+1}(y_t^j) &= \mathrm{E}_{\xi_t}[z_{t+1}(y_t^j, \xi_{t+1})] \\
&= \mathrm{E}_{\xi_t}[\pi_{t+1}^j(\xi_{t+1})(y_t^j + \xi_{t+1})], \\
&= \pi_{t+1}^j y_t^j + \rho_{t+1}^j, \\
&= (\pi_{t+1}^j B_t)x_t^j + \rho_{t+1}^j.
\end{aligned} \tag{12}
$$

By convexity and for $(\pi_{t+1}^j B_t)x_t + \rho_{t+1}^j$ a support of $z_{t+1}(B_t x_t)$, we obtain

$$
z_{t+1}(B_t x_t) \geq (\pi_{t+1}^j B_t)x_t + \rho_{t+1}^j. \tag{13}
$$

Now, we write DP(t) as

$$
\begin{aligned}
\min \quad & c_t x_t + \theta_t && (DPP(t)) \\
& A_t x_t = y_{t-1} + \xi_t, && (DPP(t)(a)) \\
& z_{t+1}(B_t x_t) \leq \theta_t, && (DPP(t)(b)) \\
& x_t \geq 0,
\end{aligned}
$$

and observe that, if (x_t, θ_t) is such that

$$
\theta_t < (\pi_{t+1}^j B_t)x_t + \rho_{t+1}^j, \tag{14}
$$

then (x_t, θ_t) is infeasible in DPP(t). This corresponds (see Figure 1.) to finding a point within the feasibility space of the previously generated cutting planes that is not feasible for z_{t+1}. Another plane is added at the new x_t value and DPP(t) is solved again. The process is repeated until (x_t^*, θ_t^*) is found such that $\theta_t^* = z_{t+1}(B_t x_t^*)$. The following lemma results:

Lemma 2. *If (x_t^*, θ_t^*) is an optimal solution to ND(t) and $\theta_t^* = z_{t+1}(B_t x_t^*)$, then x_t^* is an optimal solution of DP(t).*

Proof. (x_t^*, θ_t^*) is optimal for the reduced constraints in ND(t) and x_t^* is feasible for DP(t) since $\theta_t^* = z_{t+1}(B_t x_t^*)$, therefore, x_t^* is optimal for DP(t).∎

From this lemma, NDSDLP can be seen as a method for finding an optimal solution to the dynamic program, DP(t). We further observe that the discreteness of the ξ_t distribution was not necessary for this development. The convexity of $z_{t+1}(y_t)$ is sufficient for the outer linearization to properly bound the objective function. The next section describes how PCSDLP uses the piecewise linearity of $z_{t+1}(y_t)$ to make local definitions of the dynamic programming recursion.

4. Relation to the Piecewise Method

For $\Xi_t = \{\xi_t^1, \xi_t^2, \ldots, \xi_t^k\}$, from Lemma 1, it follows that

$$\begin{aligned} z_{t+1}(B_t x_t) &= \sum_{j=1}^{k} p^j z_{t+1}(B_t x_t, \xi_t^j) \\ &= \sum_{j=1}^{k} p^j Q(x_t, \xi_t^j) \\ &= Q(x_t). \end{aligned} \tag{15}$$

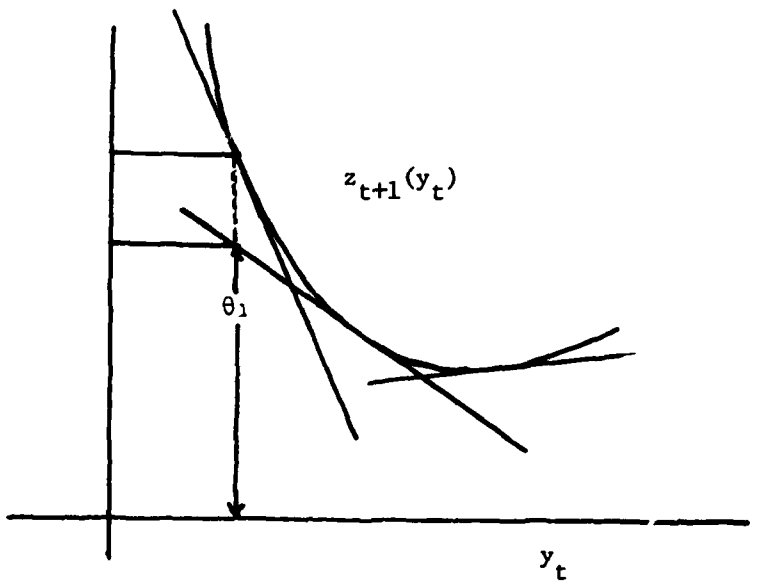


Figure 1. Outer linearizations.

Hence, z_t and Q have the same properties as functions of x_t. This implies that z_t is piecewise linear in x_t, as we showed for $Q(x_t)$ in Chapter 4. Therefore, we can write DP(t) as

$$\begin{aligned}\min\ c_t x_t \quad &+\sum_{j=1}^{k} p^j c_{t+1}[(A_{t+1}^{B_j})^{-1}(B_t x_t+\xi_{t+1}^j)] \\ &+z_{t+2}(B_{t+1}[(A_2^{B_j})^{-1}(B_t x_t+\xi_{t+1}^j)])\end{aligned} \qquad (PW(t))$$

subject to

$$\begin{aligned} A_t x_t &= y_{t-1}+\xi_t, \\ (A_2^{B_j})^{-1}B_t x_t &\geq -(A_2^{B_j})^{-1}\xi_{t+1}^j, j=1,\ldots,k, \\ x_t &\geq 0. \end{aligned}$$

We proceed as in Chapter 4 to write PW(t) as

$$\begin{aligned} \min \quad & \tilde{c}_t x_t + k' \\ \text{subject to} \quad & \\ & A_t x_t = y_{t-1}+\xi_t, \\ & -\tilde{B}_t^j x_t \geq \tilde{\xi_{t+1}}^j, j=1,\ldots,k, \\ & x_t \geq 0, \end{aligned} \qquad (PWW(t))$$

where k' is a constant, and $\tilde{c}_t$, $\tilde{B}_t^j$, and $\tilde{\xi_{t+1}}^j$ are the values relative to the subproblem bases as in (3.12) and (3.15).

Here, the function $z_t((y_{t-1},\xi_t),y_t)$ is limited to the linear piece, for which, $-\tilde{B}_t^j x_t \geq \tilde{\xi_{t+1}}^j$ for all j. The piecewise algorithm proceeds by looking for improvements from x_t^k, the kth optimal solution found to PWW(t). A sequence,

$$z_t((y_{t-1},\xi_t),y_t^1) > z_t((y_{t-1},\xi_t),y_t^2) > \cdots > z_t((y_{t-1},\xi_t),y_t^k), \qquad (16)$$

is found, such that, if $z_t((y_{t-1},\xi_t),y_t^{k+1}) = z_t((y_{t-1},\xi_t),y_t^k)$ for all feasible directions from y_t^k, then the x_t^k corresponding to $B_t x_t^k = y_t^k$ is optimal for

DP(t), and we can proceed to find $z_t(y_{t-1})$. (See Figure 2, and note that the optimum is determined for z_t only, so that z_{t+1} need not be at a minimum.)

From this development, we have

Lemma 3. *If the piecewise linear approach, PCSDLP, terminates with* $z_t((y_{t-1}, \xi_t), y_t^{k+1}) = z_t((y_{t-1}, \xi_t), y_t^k)$, *then the associated* x_t^k *is an optimal solution to* $z_t((y_{t-1}, \xi_t))$ *in DP(t).*

4. Conclusion

In Sections 3 and 4, we showed that the piecewise and nested decomposition methods found values of x_t that minimize the function z_t for a given state y_{t-1}. The following theorem, which follows directly from Lemmas 3 and 4, states this result.

Theorem 1. *The nested decomposition and piecewise methods presented in Chapters 3 and 4 obtain optimal values of* $z_t(y_{t-1}, \xi_t)$ *at each stage t of the*

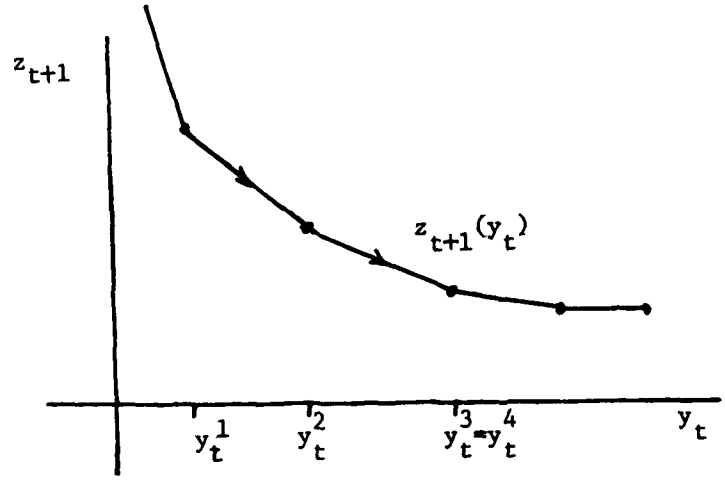


Figure 2. At y_t^3, $z_t((y_{t-1}, \xi_t), y_t^3) = z_t((y_{t-1}, \xi_t), y_t^4)$.

dynamic programming procedure.

PCSDLP and NDSDLP, when applied to this backward procedure, must each start from some given state, y_t. The dynamic programming approach of enumerating these states may be valuable here. If we let y_t take on representative values and evaluate each of the associated $z_t(y_{t-1})$ separately before proceeding to stage $t-1$, some computational effort may be saved in both the nested decomposition and piecewise approaches.

CHAPTER VII

Computational Results

and

Conclusions

1. Introduction

In this chapter, we synthesize our previous development of alternative methods for reducing the complexity of problems under uncertainty. We also present some results from solving these stochastic dynamic linear programs. In this presentation, we wish to show that the difficulties in solving complex stochastic programs are not so great that one should ignore uncertainties. We will demonstrate that the stochastic solution can be evaluated without prohibitive complications and that our techniques may prove beneficial in this evaluation.

Our strategy has been first to consider deterministic problems and then to extend them to the SDLP form. Our initial step in this approach dealt with the properties of the basis for different scenarios. In Chapter I, we showed that this analysis may result in finding an optimal deterministic solution. Beyond this, in Chapter 2, we showed that a bound on the value of the stochastic solution could be obtained before proceeding to solve SDLP. We next gave three algorithms for solving this problem. In the following sections, we present the computational properties of these algorithms in solving some examples of SDLP.

Section 2 describes our computational results on representative test problems and compares the algorithms' performance with standard linear pro-

gramming. In evaluating these methods, we also consider their usefulness in models of actual phenomena. In Section 3, we discuss areas in which stochastic linear programming has been applied, and we show that the solution of uncertainty models can lead to prudent decision–making. Following this analysis, in Section 4, we state our conclusions on solving stochastic linear programs and present directions for future research in this area.

2. Computational Results

The algorithms of Chapters III, IV, and V have been programmed in FORTRAN on the SCORE DECSystem 20 computer at Stanford University. This system is useful for observing the properties of algorithms, but it is not designed for solving the extremely large problems modeled in some stochastic dynamic linear programs. Our goal was not to draw definitive conclusions about the performance of each algorithm, but, instead, to observe some of the algorithms' properties, so that, on systems with larger processors, they may be implemented with a better understanding of their capabilities.

Since current linear programming packages take great advantage of the *sparsity* (the relative number of zero entries in the coefficient matrix), we compared our algorithms' performance with that of LPM–1, a program written by J. A. Tomlin and revised by G. Kochman at the Systems Optimization Laboratory. LPM–1 employs compact storage of the non–zero entries in the coefficient matrix, performs an LU–decomposition of the basis, retains the basis inverse in product form, and uses a merit counting sort for maintaining sparsity in the inverse (see Pfefferkorn and Tomlin [49]). This procedure has proved efficient in solving linear programs because the coefficient matrices are characteristically very sparse. Since staircase linear programs (and SDLP's) have considerable sparsity in their structure, it has been conjectured [47]

that efficient handling of these elements implicitly uses the structure and that, except on very large problems beyond the capacity of standard linear programming codes, no improvement can be made upon the packaged codes. Our procedures have an advantage over these codes because they only involve small sections of the program during an optimization. They do not require storage of the entire program parameters and, thus, may handle larger problems. Apart from this advantage, we want to observe our algorithms' performance on smaller problems that simplex codes handle easily.

The nested decomposition method, NDSDLP, was programmed in FORTRAN as NDST1. This program includes the basic algorithm presented in Chapter III, but does not include modifications, such as column passing. For solving individual linear programs at each node, NDST1 uses the procedures of LPM–1. It saves lists of the current basic variables at each node, but it does not maintain the LU–decomposition for each node. Instead, it keeps only one *ETA file* (the elementary matrices in the product form of the basis inverse) for the current node and reinverts the basis each time it encounters a new node. NDST1 also includes lower and upper bound capabilities on each variable and, hence, does not require Step 2′ of NDSDLP.

The piecewise approach, PCSDLP, was programmed as PCST1. It includes the feasibility routines of NDST1 for finding a primal feasible solution of SDLP, and it follows the procedures of PCSDLP given in Chapter IV. It does not, however, have capabilities in saving lists of subproblem bases. Alternatively, it considers each subproblem basis individually and, therefore, may be forced to solve larger than necessary master problems. PCST1 also uses lower and upper bounds on all variables and checks for possible upper bound violations of the equation (IV.4a), as we mentioned above. For the case of artificial variables in the subproblem basic sets, the upper bounds are

always included as additional constraints.

Storage in PCST1 is larger than that in NDST1 because of the need to maintain both the last solution for ς^{μ} and the solution for the current auxiliary problem of $\varsigma^{\mu,*}$. For this reason, ETA files are maintained for both problems.

LBSMPX was implemented in the program LBS1. This program uses the basic procedures of LPM–1 for the artificial basis, with updated transformations for the surplus basis blocks. It follows the cyclical pricing strategy described in Chapter V and uses a "most negative" pricing criterion in each period. Pricing is restricted to that period as long as the least reduced cost is negative. Updates of the surplus basis blocks, Q_t^j, are performed according to U1, U2, U3, and U4 in Chapter V, and the inverse of this matrix is stored explicitly.

We have chosen a set of representative problems to show the algorithms' computational properties. The data for the test cases appear in Table 1.

TABLE 1

Program Parameters

	Number of Constraints (m)	Number of Variables (n)	Number of Nodes (k)	Density (ρ^*)
PA1	10	22	3	.15
RD1	10	25	3	.25
NG1	19	35	3	.28
PA2	22	50	7	.09
RD2	22	57	7	.14
PA3	46	106	15	.04
RD3	46	121	15	.07

* (number of non-zero elements)/(total number of elements)

These examples were chosen to represent the types of problems found in applications.

The first cases, RD1, RD2, and RD3, were selected from a control problem in Davis [20] and modified to reflect a production planning model as in Beale *et al* [7]. In this format, the expected present value of profits is maximized for a firm planning the production and storage of a number of products that require common resources. Their sales are determined by an uncertain demand. The decision variables in this model are :

$x_{i,t}$ — the amount of product i sold in period t,

$y_{i,t}$ — the amount of i produced in period t,

$s_{i,t}$ — the amount of product i in stock at the end of period t.

The linear program for this problem is

$$
\begin{aligned}
&\max \textstyle\sum_{t=1}^{T} \sum_{i=1}^{q} (p_{i,t}x_{i,t} - c_{i,t}y_{i,t} - k_{i,t}s_{i,t}) \\
&\text{s. t.} \qquad \textstyle\sum_{i=1}^{q} y_{i,t} \leq R_t, \text{ for all } t, \\
&\qquad\qquad s_{i,t-1} + y_{i,t} - x_{i,t} = s_{i,t}, \text{ for all } t \text{ and } i, \\
&\qquad\qquad 0 \leq x_{i,t} \leq d_{i,t}, \text{ for all } t \text{ and } i,
\end{aligned} \tag{1}
$$

where R_t is the total capacity in period t, $p_{i,t}$ is the selling price of product i, $c_{i,t}$ is the production cost of product i, $k_{i,t}$ is the holding cost of stock in i, T is the planning horizon, and q is the number of products. In this example, we included random uncertainties in $d_{i,t}$ and formulated (1) as in SDLP for optimizing the expected value of profits over the planning horizon.

The examples, PA1, PA2, and PA3, were contributed by P. Abrahamson [1] and represent the discrete approximation of a continuous time linear program. This program represents the optimal control of a particle subject to state space constraints. It is written as

$$
\begin{aligned}
&\min \qquad \int_0^3 u(t)e^{\alpha t}dt \\
&\text{s.t.} \quad x(t) - \textstyle\int_0^t (u(s) - x(s))ds = 1, \\
&\qquad\qquad u(t) \leq 1, \\
&\qquad\qquad x(t) \leq .7 - .2t, \\
&\qquad\qquad x(t), u(t) \geq 0,
\end{aligned} \tag{2}
$$

where we considered uncertainty in the state to state transitions for $x(t)$. The application of uncertainty to continuous problems, as suggested by this model, is an important area for possible future applications.

The last example, NG1, is derived from the exhaustible resource model of Chapter I. It includes uncertainty in investment in different technologies. This example is included, because the basis must be changed significantly from the myopic first period solution to the optimal stochastic solution. This process requires more surplus basic variables in the first period and demonstrates

TABLE 2

Sample Times and Iterations

	LPM-1		NDST1		PCST1		LBS1	
	Time (CPUs)	No. of Iter.	Time (CPUs)	No. of Iter.	Time (CPUs)	No. of Iter.	Time (CPUs)	No. of Iter.
PA1	.18	12	.14	12	.19	12	.20	11
RD1	.30	15	.23	12	.42	13	.33	15
NG1	.48	18	.83	29	1.01	23	.56	17
PA2	.94	29	1.71	29	2.41	30	1.05	28
RD2	1.59	31	.91	32	1.34	33	1.73	31
PA3	4.01	61	3.39	68	5.61	79	4.96	64
RD3	7.13	65	4.89	67	7.64	71	6.72	59

some of the complications of stochastic programs.

The solution times and number of simplex iterations required for the optimal solution of these problems are presented in Table 2. The times are in CPU seconds for solutions from a "cold start"(an infeasible basis), and represent computing time excluding time for data input. The results show that, on small programs, our algorithms may be competitive with codes using sparsity techniques alone, such as LPM-1. This is especially significant because NDST1, PCST1, and LBS1 are experimental codes and are not programmed for maximum computing efficiency. Small problems, however, are not representative of actual applications, and experience with larger programs is mandatory for true comparisons. Again, we do not intend to prove the superiority of our methods, but only to show how they perform.

TABLE 3

Sample Times per Iteration

	LPM–1 Time/No. of Iterations	NDST1 Time/No. of Iterations	PCST1 Time/No. of Iterations	LBS1 Time/No. of Iterations
PA1	.015	.012	.016	.018
RD1	.020	.019	.032	.022
NG1	.027	.028	.044	.033
PA2	.032	.059	.080	.037
RD2	.051	.028	.041	.056
PA3	.066	.049	.071	.078
RD3	.110	.073	.108	.114

NDST1 performs best overall on these examples, but, where it performs poorly we can observe some of its properties. In NG1, a large number of cuts were required on the first period and many suboptimizations were performed, slowing its convergence to an optimum. We also see here that these suboptimizations required NDST1 to perform 25 percent more iterations than PCST1, which maintains subproblem optimality. This difficulty also appears in PA2 where the added effort of reinversion at each pass from node to node led to NDST1's having a larger average time per iteration than LPM–1 (see Table 3.). We note that the small growth of non–zeroes in the basis inverse of PA2 is a good example of LPM–1's advantage in performing backward and forward transformations very quickly. On the other examples, because NDST1 solves the smallest problems, it requires the least amount of time per

iteration despite reinversion.

An interesting property observed in the solutions by NDST1 concerns the set of basic variables in the master problem as cutting planes are added from the subproblems. In our examples, we often saw that one set of basic variables would remain constant in the master problem from iteration to iteration and that the additional basic elements, would be chosen from another small set of variables. The algorithm would choose a member of the additional set on one iteration, replace it with another member in the next iteration, and then bring the first element back into the basis at master–subproblem optimality. For X_B, the constant set of basic variables, and $\{x_1, x_2\}$, the additional basic variables, the basis in the master problem would follow the pattern:

Iteration	Basic Set
1	$\{X_B\}$
2	$\{X_B, x_1\}$
3	$\{X_B, x_2\}$
4	$\{X_B, x_1, x_2\}$– optimal.

It has been observed [1] that the deterministic problem often brings additional variables into the basis and then adjusts them in subsequent iterations without dropping them from the basis. This observation led to the column passing technique we discussed above. Our observations indicate that the set of columns passed forward to the subproblems should include columns that were in the basis but were deleted. This would allow for the entire set of columns to obtain the optimal weights in a single pass.

The behavior we have observed may be indicative of the hedging effect in stochastic problems. In our example above, x_1 may be included to optimize the first descendant scenario and x_2 may correspond to optimising with

TABLE 4

Number of First Period Optimizations

	NDST1 Number of Passes	PCST1 Number of Passes
PA1	6	4
RD1	8	7
NG1	14	11
PA2	8	6
RD2	9	7
PA3	12	10
RD3	15	13

the second subproblem. The algorithm tries each of these activities before deciding on the optimal combination in Iteration 4. By including each of these potentially basic variables in the set corresponding to the columns passed forward, we may avoid excessive optimization.

PCST1 performed consistently worse than NDST1 based on CPU time, but its strength of not requiring suboptimization appears in Table 4. This chart displays the number of times the master problem in period 1 was solved. PCST1 requires fewer of these optimizations and passes to the subproblems. It thus decreases the possibility of lengthy suboptimization. PCST1 solves, however, a larger master problem than NDST1, because it includes the degeneracies associated with repeated subproblem bases. The basis list approach presented in Chapter 4 should alleviate some of the

difficulty here.

LBS1 also maintains more information than is necessary and its use of the triangular basis differs only slightly from the factorization for sparsity in LPM–1. LBS1's additional bookkeeping is most evident on smaller programs and begins to be efficient for larger problems as we start to see in RD3.

These examples presented some of the properties of NDST1, PCST1, and LBS1. Knowledge of these attributes should be helpful in determining what method to use for a specific problem. To determine this, as we stated above, the deterministic scenarios should be evaluated first and then the stochastic problem should be solved. Because of its simplicity, we would recommend NDST1 as a first method to try, followed by PCST1, if convergence to optimality is slowed by excessive suboptimization. LBS1's implementation is more determined by efficient coding and its evaluation should be made only once its procedures have been efficiently coded for larger problems on faster processors.

3. Areas of Application

We have mentioned that stochastic dynamic linear programs may be found in many real–world contexts. In this section, we discuss some actual applications of these problems in decision–making. We mention these examples to demonstrate the range of possible implementations of our techniques.

The first use of stochastic linear programs was by Dantzig and Ferguson [21] on a problem of airline scheduling with uncertain passenger demand. Using a modification of the transportation simplex method, they demonstrated that the net expected costs for meeting the carrier's demand could be reduced from $1,666,000 to $1,524,000 by considering the complete distribution of

demand instead of only the expected values of those demands.

More recent uses of stochastic linear programs include Manne's analysis in [44] of the decision to develop breeder fission technology. Another analysis of nuclear fuel choices appears in Avi-Itzhak and Connally [4], wherein, they use the scenario approach of assuming decisions based on deterministic solutions and evaluating the expected cost of these decisions under different future scenarios.

Other areas of application are found in the investment management of bank portfolios. Aghili *et al* [2] evaluated the decisions of a small midwestern bank in allocating their assets among a set of securities and loan positions. They used the bank management's criteria for constraints and formulated a two-stage stochastic linear program with uncertain interest rates and bond yields. Their results using only financial and accounting constraints reflected the extreme point property of optimal linear program solutions. Their model would have placed almost all of the bank's portfolio into municipal bonds, forgoing all commercial loans and mortgages.

The difficulty Aghili *et al*, encountered arose from their use of the risk neutral linear objective function. To improve their results, they incorporated management's concerns into additional policy constraints and obtained solutions that were much closer to the bank's own decisions. From this analysis, management became aware of the need to consider different future economic environments. Previously, they had used only one forecast for the future but were now able to consider several scenarios. This openness to changing money conditions gave the bank management more options to consider in their policy decisions and led them to reshape some of their ideas on the bank's operations.

The bank balance sheet model is a good example of decision makers'

use of a stochastic model. Difficulties often arise in the implementation of a model's optimal solution, but, in Aghili *et al*'s experience, the model served to inform management, and it became an important policy instrument. Another example of this utility of stochastic linear models appears in Gaither's study [24] of commercial fishing seasons. That research demonstrated that substantial savings could be accrued by proper fishing regulation in Alaska, but modifying those regulations required difficult policy decisions that would not be immediately forthcoming. Despite these problems, Gaither's model also played an important role in demonstrating to the decision makers how they should structure their strategy and what concerns they should have.

4. Conclusion

We have presented several approaches for solving linear models that involve decisions made over time in uncertain environments. We have showed that the resulting stochastic dynamic linear programs can be analyzed through the use of deterministic scenario solutions and the subproblem solution methods of nested decomposition, piecewise path following, and local basis factorization. This analysis rests upon the fundamental properties of the basis in linear programming, the understanding of which is crucial in understanding the characteristics of a stochastic solution.

The stochastic solution succeeds because it allows for a regularization of the extreme point solutions that occur in different deterministic scenarios. This property should be carefully considered in evaluating any stochastic solution because it can be a powerful stabilizing force in determining optimal decisions. Within the framework of the stochastic model, a single decision can be chosen that will reflect each of the future scenarios included in the model.

The methods we presented for solving these SDLP's demonstrated credible performance on smaller examples. These results are encouraging for future implementations on large–scale problems. This experimentation on large models should be the major direction for future research.

The true test of our algorithms' benefits and of the usefulness of the SDLP program, in general, must come from actual policy models. Our presentation of examples from production planning, exhaustible resources, and portfolio management demonstrate that many applications of SDLP are possible. The inclusion of uncertainty into a linear model extends the range of decision factors and helps stabilize policy strategies in dynamic settings. These attributes may bring significant gains to the use of models in decision–making.

Our primary goal has been to show that, by exploiting the stochastic linear program's special structure, models may be able to include uncertain parameters without becoming hopelessly complex. By proceeding from deterministic scenarios through the stochastic solution, the modeler can be aware of the value of proceeding to more complex stages and can develop a better understanding of the nature of the deterministic linear programming solution and its relation to the stochastic strategy. To complete this analysis most effectively, the computational costs of the stochastic program must be reduced. We hope that our work will help make these reductions possible, thereby, enabling models to become more powerful tools in affecting current decisions in an uncertain world.

BIBLIOGRAPHY

[1] Abrahamson, P. G., "A Nested Decomposition Approach for Solving Staircase Structured Linear Programs", **Proceedings of the Workshop on Large–Scale Linear Programming**, IIASA, Laxenburg, Austria, June 2–6, 1980.

[2] Aghili, P., R.H., Cramer and H.W. Thompson, "On the applicability of two-stage programming models to small bank balance sheet management", *J. Bank Res. J.*, pp. 246-256.

[3] Arrow, K.J., S. Karlin and H. Scarf, **Studies in the Mathematical Theory of Inventory and Production**, Stanford University, Stanford, CA, 1958.

[4] Avi-Itzhak, B. and T.J. Connolly, "The plutonium issue", Technical Report SOL 78-24, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, CA, September 1978.

[5] Avriel, M. and A.C. Williams, "The value of information and stochastic programming", *OR* 18, No. 5, Sept.-Oct. 1970.

[6] Beale, E.M.L., "The simplex method using pseudo-basic variables for structured linear programming problems", pp. 133-148 in *Recent Advances in Math. Prog.*, R.L. Graves and P. Wolfe (eds.), McGraw-Hill, New York, 1962.

[7] Beale, E., J.J.H. Forrest and C.J. Taylor, "Multi-time period stochastic programming", in *Stochastic Programming*, M.A.H. Dempster (ed.), Academic Press, New York, 1980.

[8] Bellman, R., Dynamic Programming, Princeton University Press, Princeton, NJ, 1957.

[9] Benders, J.F., "Partitioning procedures for solving mixed-variables programming problems", *Numerische Mathematik* 4, pp. 238-252, 1962.

[10] Bereanu, B., "Some numerical methods in stochastic linear programming under risk and uncertainty", in *Stochastic Programming*, M.A.H. Dempster (ed.), Academic Press, New York, 1980.

[11] Bisschop, J. and A. Meeraus, "Matrix augmentation and partitioning in the

updating of the basis inverse", *Math. Prog.* **13**, No. 3, pp. 241-254.

[12] Chao, H.P., "Exhaustible resource models: the value of information", Technical Report, Stanford University, August 1979.

[13] Chao, H.P., Private communication, May, 1980.

[14] Charnes, A. and W.W. Cooper, "Chance-constrained programming", *Management Science* **6**, No. 1, October 1959.

[15] Dantzig, G.B., "Upper bounds secondary constraints and block triangularity in linear programming", *Econometrica* **23**, pp. 174-183, April 1955.

[16] Dantzig, G.B., *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, 1963.

[17] Dantzig, G.B., "Solving staircase systems", draft of September 12, 1980.

[18] Dantzig, G.B. and A. Madansky, "On the solution of two-stage linear programs under uncertainty",*Proceedings, 4th Berkeley Symp. on Mathematical Statistics and Probability* , U.C. Press, Berkeley, pp. 165-176, 1961.

[19] Dantzig, G.B. and P. Wolfe, "Decomposition principle for linear programs", *OR* 8, No. 1, pp. 101-111, Jan.-Feb. 1960.

[20] Davis, R., "New jump conditions for state constrained optimal control problems", Ph.D. dissertation, Stanford University, 1980.

[21] Ferguson, A. and G.B. Dantzig, "The allocation of aircraft to routes: an example of linear programming under uncertain demands", *Management Science* **3**, pp. 45-73, 1956.

[22] Fourer, R., "Solving staircase l.p.'s by the simplex method", Technical Report SOL 79-18, Systems Optimization Laboratory, Stanford University, Stanford, CA, November 1979.

[23] Gabbay, H., "Multi-stage production planning", *Management Science* **25**, No. 11, pp. 1138-1148, Nov. 1979.

[24] Gaither, N., "A stochastic constrained optimization model for determining commercial fishing seasons", *Management Science* **26**, No. 2, Feb. 1980.

[25] Garstka, S. and D. Ruthenberg, "Computations in discrete stochastic programs without recourse", *OR* **21**, PP. 112-122, 1973.

[26] Garstka, S. and R.J.B. Wets, "On decision rules in stochastic programming",

Math. Prog. **7**, No. 2, pp. 117-143, 1974.

[27] Geoffrion, A.M., "Elements of large-scale mathematical programming", *Management Science* **16**, No. 11, pp. 652-675, July 1970.

[28] Glassey, C.R., "Dynamic linear programs for production scheduling", *OR* **19**, pp. 45-56, 1971.

[29] Glassey, C.R., "Nested decomposition and multi-stage linear programs", *Management Science* **20**, pp. 282-292.

[30] Grinold, R.C., "A new approach to multi-stage stochastic linear programs", pp. 19-29 in *Stochastic systems: Modelling, Identification and Optimisation*, R.J.B. Wets (ed.); also *Management Science* 8, 1975.

[31] Gunderson, H.S., J.G. Morris, and H.E. Thompson, "Stochastic programming without recourse: a modification from an applications viewpoint", *JORS* **29**, No. 8, pp. 769-778, August 1978.

[32] Ho, J.K. and A.S. Manne, "Nested decomposition for dynamic models", *Math. Prog.* **6**, pp. 121-140, 1974.

[33] Holt, C., F. Modigliani, J. Muth, and H. Simon, *Planning Production, Inventories, and Work Force*, Prentice-Hall, Englewood Cliffs, NJ, 1960.

[34] Hotelling, H., "The economics of exhaustible resources", *Journal of Political Economy* **39**, pp. 137-175, 1931.

[35] Huang, C.C., W.T. Ziemba and A. Ben-Tal, "Bounds on the expectation of a convex function of a random variable with applications to stochastic programming", *OR* **25**, pp. 315-325, 1977.

[36] Huang, C.C., I. Vertinsky, W.T. Ziemba, "Sharp bounds on the value of perfect information", *OR* **25**, pp. 128-139, 1977.

[37] Kall, P., 'Computational methods for solving two-stage stochastic linear programming problems", *Journal of Appl. Math. and Physics* **30**, pp. 261-271, 1979.

[38] Kallio, M. and E.L. Porteus, "Decomposition of arborescent linear programs", *Math. Prog.* **13**, No. 3, pp. 348-355, Dec. 1977.

[39] Larson, R.E., "A survey of dynamic programming computational procedures", *IEEE Trans. on Automatic Control* **AC-12**, no. 6, pp. 764-774, 1967.

[40] Larson, R.E. and W.G. Keckler, "Application of dynamic programming to the control of water resource systems", *Automatica* 5, No, 1, pp. 15-26, January 1969.

[41] Larson, R.E. and J.L. Casti, *Principles of Dynamic Programming*, Marcel Dekker, Inc., New York, 1978.

[42] Madansky, A., "Inequalities for stochastic linear programming problems", *Management Science* 6, pp. 197-204, 1960.

[43] Manne, A.S., "Waiting for the breeder", *Review of Economic Studies Symposium*, pp. 47-65, 1974.

[44] Manne, A.S., and R.G. Richels, 'A decision analysis of the U.S. breeder reactor program", *Energy*, 3, pp. 747-767, 1978.

[45] Murty, K.G., "Two-stage linear program under certainty: a basic property of the optimal solution", *Zeitschrift fuer Wahrscheinlicheitstheorie und Verwandte Gebiete* 10, pp. 284-288, 1968.

[46] Nordhaus, W.D., "The allocation of energy resources", Brooklyn Paper on Economic Activity, 3, pp. 529-576, 1973.

[47] Perold, A., Private communication, February 1980.

[48] Perold, A.F. and G.B. Dantzig, "A basis factorization method for block triangular linear programs", Technical Report SOL 78-7, Systems Optimization Laboratory, Stanford University, April 1978.

[49] Pfefferkorn, C.E. and J.A. Tomlin, "Design of a linear programming system for ILLIAC IV", Technical Report SOL 76-8, Systems Optimization Laboratory, Stanford University, April 1976.

[50] Phillips, L.D. and H.J. Otway, "Limitations to human judgment: implications for system modellers", *Energy systems Analysis International Conference*, Dublin, Ireland, October 9-11, 1979.

[51] Propoi, A., "Models for educational and manpower planning: a dynamic linear programming approach", RM-78-20, IIASA, Laxenburg, Austria, May, 1978.

[52] Propoi, A. and V. Krivonozhko, 'The simplex method for dynamic linear programs", RR-78-14, IIASA, Vienna, Austria, September 1978.

[53] Rosen, J.B., "Convex partition programming", in Recent Advances in Mathematical *Programming*, R.L. Graves and P. Wolfe (eds.), McGraw-Hill, New York,

1963.

[54] Samuelson, P.A., "Lifetime portfolio selection by dynamic stochastic programming", *Rev. Econ. Stat.* **51**, pp. 239-246, 1969.

[55] Stancu-Minasian, S. and M.J. Wets, "A research bibliography in stochastic programming", *OR* **24**, No. 61, pp. 1078-1119, 1976.

[56] Van Slyke, R. and R. Wets, "L-shaped linear programs with applications to optimal control and stochastic linear programs", *SIAM J. of Appl. Math.* **17**, pp. 638-663, 1969.

[57] Van Slyke, R. and R. Wets, "Programming under uncertainty and stochastic control", *J. SIAM Control* **4**, No. 1, pp. 179-193, 1966.

[58] Walkup, D. and R. Wets, "Stochastic programs without recourse", *SIAM J. of Appl. Math.* **15**, pp. 1299-1314, 1967.

[59] Walkup, D. and R. Wets, "Stochastic programs without recourse, II: on the continuity of the objective", *SIAM J. of Appl. Math.* **17**, pp. 98-103, 1969b.

[60] Wets, R., "Programming under uncertainty: the complete problem", *Z. Wahrscheinlicheitstheorie und Verwandte Gebiete* **4**, pp. 319-339, 1966.

[61] Wets, R., "Programming under uncertainty: the solution set", *SIAM J. of Appl. Math.* **14**, pp. 1143-1151, 1966.

[62] Wets, R., "Programming under uncertainty: the equivalent convex program", *SIAM J. of Appl. Math.* **14**, pp. 89-105, 1966.

[63] Wets, R., "Characterization theorems for stochastic programs", *Math. Prog.* **2**, pp. 166-175, 1972.

[64] Wets, R., 'Stochastic programs with fixed recourse: the equivalent convex program", *SIAM Review* **16**, No. 3, July 1974.

[65] Wets, R., "Stochastic programs with recourse: a basic theorem for multi-stage problems", *Zeitschrift fuer Wahrscheinlicheitstheorie und Verwandte Gebiete* **21**, pp. 201-261, 1978.

[66] White, W.W., " A status report on computing algorithms for mathematical programming", *ACM Computing Survey* **5**, No. 3, September 1973.

[67] Winkler, C., "Basis factorization for block-angular linear programs", Technical Report SOL 74-19, Systems Optimization Laboratory, Stanford University,

December 1974.

[68] Yudin, D.B., "Mathematical methods of control under uncertainty problems and methods of stochastic programming", Soviet Radio, Moscow, 1974.

[69] Ziemba, W.T., "Transforming stochastic dynamic programming problems with nonlinear programs", *Management Science* 1, pp. 450-462, 1971.

[70] Ziemba, W.T., "Computational algorithms for convex stochastic programs with simple recourse", *OR* 8, pp. 414-431, 1970.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) SOL-80-29	2. GOVT ACCESSION NO. AD-A096	3. RECIPIENT'S CATALOG NUMBER 119
4. TITLE (and Subtitle) SOLUTION METHODS FOR STOCHASTIC DYNAMIC LINEAR PROGRAMS,		5. TYPE OF REPORT & PERIOD COVERED (9) Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(a) (10) John R. BIRGE		8. CONTRACT OR GRANT NUMBER(a) (15) N00014-75-C-0267 DE-AC03-76SF00326
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research - SOL Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research - Dept. of the Navy 800 N. Quincy Street Arlington, VA 22217		(11) 12. REPORT DATE December 1980 13. NUMBER OF PAGES 131
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) (12) 138		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

large-scale linear programming
stochastic programming
dynamic linear programs
decomposition
expected value of perfect information
basis factorization

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SEE ATTACHED

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD A096 408765

SOL 80-29: Solution Methods for Stochastic Dynamic Linear Programs,
John R. Birge

Linear programs have been formulated for many practical situations that require decisions made periodically through time. These dynamic linear programs often involve uncertainties. Deterministic solutions of these problems may lead to costly incorrect decisions, and, when a stochastic solution is attempted the problem may become too large. In this report, we present methods for reducing the computational cost of these stochastic programs, and we show conditions under which the stochastic program need not be solved. Our methods are based on the large-scale programming techniques of decomposition, partitioning, and basis factorization.


LME