TUTORIALS IN OPERATIONS RESEARCH
INFORMS 2005

© 2005 INFORMS | ISBN 1-877640-21-2
DOI 10.1287/educ.1053.0016

Stochastic Programming: Optimization When Uncertainty Matters

Julia L. Higle
Department of Systems and Industrial Engineering, University of Arizona, Tucson, Arizona 85721, julie@sie.arizona.edu

Abstract Stochastic programming (SP) was first introduced by George Dantzig in the 1950s. Since that time, tremendous progress has been made toward an understanding of properties of SP models and the design of algorithmic approaches for solving them. As a result, SP is gaining recognition as a viable approach for large-scale models of decisions under uncertainty. In this paper, we present an introduction to stochastic programming models and methodology at a level that is intended to be accessible to the breadth of members within the INFORMS community.

Keywords stochastic programming; recourse models; decomposition methods

1. Introduction

Stochastic programming traces its roots to Dantzig [11], where the recourse model is first introduced, and Charnes and Cooper [9], where probabilistically constrained models are introduced. In many ways, a stochastic program (SP) might be thought of as one of the ultimate operations research models. When some of the data elements in a linear program (LP) are most appropriately described using random variables, a stochastic linear program (SLP) results. In this sense, stochastic programs involve an artful blend of traditional (deterministic) mathematical programs and stochastic models. In many cases, solution techniques for stochastic programs rely on statistical estimation and numerical approximation methods. In short, stochastic programming draws upon a wide variety of traditional operations research techniques.

In this paper, we provide a tutorial-level presentation of stochastic programming. Our goal is to provide an overview of the subject that is accessible to most members of the operations research community, with a particular focus on some of the details and terminology that potential users of SP methodology will want to know. We begin with an example illustrating why an SP model might be preferred to a deterministic model. Following that, we identify some of the essential features of a stochastic program that can be exploited by a solution method. Our focus in this paper will be on the class of recourse problems rather than the probabilistically constrained problems. This is primarily because the two classes of problems involve considerably different modeling and solution approaches. For a comprehensive review of probabilistic constraints, the reader is referred to Prékopa [36].

2. Does Uncertainty Matter?

As a general rule, stochastic programs are more difficult to formulate and solve than deterministic mathematical programs. Given the availability of postoptimality analysis, it can be tempting to ease the process by relying on sensitivity analysis to investigate the impact of uncertainty. In many cases, sensitivity analysis is not an appropriate tool for this investigation, as we shall see in the example that follows.

2.1. Sensitivity Analysis

We begin with a simple linear programming problem, which is a variation of a problem that appears in Winston and Venkataramanan [49], and also appears in Higle and Wallace [24].

> The Dakota Furniture Company manufactures desks, tables, and chairs. The manufacture of each type of furniture requires lumber and two types of skilled labor: finishing and carpentry. The cost of each resource and the amount needed to make each type of furniture is given in Table 1.

TABLE 1. Dakota production/resource data.

Resource	Cost ($)	Desk	Table	Chair
Lumber (bd. ft.)	2	8	6	1
Finishing (hrs.)	4	4	2	1.5
Carpentry (hrs.)	5.2	2	1.5	0.5
Demand		150	125	300

> A desk sells for $60, a table sells for $40, and a chair sells for $10. In order to maximize total profit, how much of each item should be produced, and what are the corresponding resource requirements?

There are many ways to determine the production quantities. Perhaps the easiest method is a simple per-item profit analysis. A desk costs $42.40 to produce and sells for $60, for a profit of $17.60 per desk sold. Similarly, a table costs $27.80 to produce and sells for $40, for a profit of $12.20 per table sold. On the other hand, a chair costs $10.60 to produce, and sells for only $10, so that each chair sold represents a net loss of $0.60. Given the objective of maximizing profit, the information provided suggests that Dakota should meet the demand for desks and tables, but decline to produce any chairs, as indicated in Table 2. To determine the resource requirements, we only need to work with the data provided in Table 1. That is, based on the resource requirements per item, production of 150 desks and 125 tables requires the acquisition of 1,950 bd. feet of lumber, 850 labor hours for finishing, and 487.5 labor hours for carpentry. Based on these production and resource quantities, Dakota anticipates a profit of $4,165.

This problem does not require the power of linear programming for solution. Nonetheless, LP models are easily derived. For example, if we define

x_b = number of bd. feet of lumber acquired
x_f = number of labor hours acquired for finishing
x_c = number of labor hours acquired for carpentry
y_d = number of desks produced
y_t = number of tables produced
y_c = number of chairs produced

the problem can be formulated as follows:

$$\begin{aligned}
\text{Max}\quad & 60y_d + 40y_t + 10y_c - 2x_b - 4x_f - 5.2x_c \\
\text{s.t.}\quad & y_d \le 150 \\
& y_t \le 125 \\
& y_c \le 300 \\
& 8y_d + 6y_t + y_c - x_b \le 0 \\
& 4y_d + 2y_t + 1.5y_c - x_f \le 0 \\
& 2y_d + 1.5y_t + 0.5y_c - x_c \le 0 \\
& y_d, y_t, y_c, x_b, x_f, x_c \ge 0.
\end{aligned} \tag{1}$$

Of course, the solution to this problem is identical to the one presented in Table 2.

Table 2. Dakota production/resource solution.

Item	Production cost ($)	Selling price ($)	Profit per item ($)	Demand	Production
Desk	42.40	60.00	17.60	150	150
Table	27.80	40.00	12.20	125	125
Chair	10.60	10.00	−0.60	300	0

Notice that even though the basic data might change, the structure of the problem remains the same. For example, if the selling price for a chair is increased from $10 to $11, it is only necessary to change the appropriate objective function coefficient in the model that is being used. This is the observation that gives rise to the postoptimality investigation of the solution known as *sensitivity analysis*. That is, knowing that the basic structure of the model remains the same, we can investigate the manner in which changes in individual data elements would impact an optimal solution. Notice that if nothing else were to change, and the selling price of a chair increased to $11, the production of chairs would become profitable, and the nature of the solution would change considerably. On the other hand, if the price of the chair were to remain at $10 but the demand for chairs changed from 300 to 200, there would be no impact on the nature of the solution.

Sensitivity analysis is used in an attempt to study the robustness of the solution to a linear programming model. If there is cause for concern regarding the accuracy of the data used, sensitivity analysis is undertaken to determine the manner in which the solution might change if the data were different. When the solution does not change (or when the nature of the solution does not change, as when the basis remains optimal), one breathes a sigh of relief, believing that the proposed solution is appropriate. Unfortunately, no such relief is available if the solution is sensitive to the data. How should you proceed if the solution, or the nature of the solution, varies when the data is changed? In the Dakota example, we saw that a small change in the selling price of a chair took the solution from one extreme, in which no chairs were produced, to the other, in which 300 chairs were produced! Surely, this would be cause for concern and a clear indication that additional effort should be expended toward a better understanding of the selling price of chairs.

Although sensitivity analysis offers some sense of security, it is important to recognize that, in many cases, this is really a false sense of security. If there is some uncertainty about the values of some data elements, it should be included in the model. How do we model problems when we *know* that some of the data elements are difficult to predict or estimate? This is precisely the situation for which stochastic programming models are intended.

2.2. Introducing Uncertainty to the LP Model

As a modeling tool, linear programming is intended for deterministic problems. That is, when we formulate a linear programming model, we act as if all data elements are known quantities. For example, in the Dakota example, Table 1 provides everything we need to know about each product's resource requirements. The demand for each product is specified, as are the per-unit selling prices and the per-unit resource costs. It seems a bit presumptuous to assume that we know *all* of these values! Any or all of them may be subject to uncertainty.

It is probably reasonable to assume that Dakota has a good handle on the resource requirements for each of the three items that they produce. However, the selling prices, resource costs, and product demands might all be reasonably assumed to be subject to uncertainty. We have already seen that the solution to the LP model is sensitive to the objective coefficients. To illustrate the power of stochastic programming as a modeling paradigm, we will focus on the demand, and explore the impact of uncertainty in these values. That is, suppose that for each product we are given three potential demand scenarios, which we identify as being the "low," "most likely," and "high" values, as indicated in Table 3.

Table 3. Dakota demand scenarios.

	Low value	Most likely value	High value
Desks	50	150	250
Tables	20	110	250
Chairs	200	225	500
Probability	0.3	0.4	0.3

Notationally, we will denote the probability that the set of low values occurs as p_ℓ, and let p_m and p_h be similarly defined for the most likely and high values, respectively. That is, $p_\ell = 0.3$, $p_m = 0.4$, and $p_h = 0.3$. Notice that the collection of demand scenarios and the corresponding probabilities form a multivariate probability distribution that can be used to describe the demand that Dakota might face in the future.

Now that we have a basic understanding of the deterministic model, and we've been given a collection of demand scenarios and their corresponding probabilities, what should we do next? When faced with this question, most people seem to come up with one or both of the following suggestions:

- Use the *expected demand* and proceed.
- Solve the problem for *each* of the demand scenarios, and somehow combine the solutions.

If we follow this course, we first note that the expected demand is precisely the demand that we used initially: 150 desks, 125 tables, and 300 chairs. Because we have already solved the mean value problem, let us see what happens if we solve the scenario problems. Notice that this merely requires changing the right-hand-side values of the demand constraints in the original problem. The output of this exercise is summarized in Table 4.

There are many observations that one might make regarding Table 4. First, notice that *all* cases indicate that chairs should not be produced. This is to be expected, given our earlier comments regarding the profitability of the various items. Notice that the basic structure of the LP that we have used has not changed, and it still costs more to produce a chair than the revenue that we receive from the sale of a chair. As before, chairs are not profitable, and so none are produced. Next, notice that the resource and production quantities in the first column are simply the expected values of the resource and production quantities in each of the individual scenario columns. Again, this can be traced directly back to the original LP model. Recall that sensitivity analysis of the solution of this LP indicates that the same basis will be optimal for all of these demand configurations. In that case, it is easy to show that this situation must arise (i.e., the expected value of the optimal solutions obtained from the individual scenarios is an optimal solution to the expected demand problem[1]).

These two observations may offer some consolation. The basic structure of all of the solutions is the same—no chairs are produced. Moreover, there is a direct relationship between the collection of scenario solutions and the solution to the problem in which expected values are used. This would seem to suggest that if they are interested in maximizing the expected profit, Dakota should simply use the solution that appears in the first column.

Or should they? This solution would have Dakota produce 150 desks and 125 tables, and purchase exactly enough resource to accomplish this task. Notice that if 150 desks are produced, there is a 30% chance that Dakota will produce more than they are able to sell. Similarly, if 125 tables are produced, there is a 70% chance Dakota will produce more than they are able to sell. Notice that if Dakota plans for the expected demand, and the demand turns out to be low, there will be 100 desks and 105 tables produced that will not be sold.

[1] In the standard notation of linear programming, if $\tilde{b}$ is a right-hand-side vector in which some elements are random variables, and the same basis B is optimal for all possible values of $\tilde{b}$, then $E[B^{-1}\tilde{b}] = B^{-1}E[\tilde{b}]$ is an optimal solution to the problem in which the random vector is replaced by its expected value.

Table 4. Output for individual demand scenarios.

	Demand			
	Expected	Low	Most likely	High
Resource quantities				
Lumber (bd. ft.)	1,950	520	1, 860	3,500
Finishing labor (hours)	850	240	820	1500
Carpentry labor (hours)	487.5	130	465	875
Production Quantities				
Desks	150	50	150	250
Tables	125	20	110	250
Chairs	0	0	0	0
Profit ($)	4,165	1,124	3,982	7,450

Instead of the $4,165 profit that the model suggests they should expect, they will realize a net *loss* of $6,035... and there is a 30% chance that this painful situation will occur![2]

Why does this happen? This is exactly the result of the false sense of security that sensitivity analysis provides. The solutions that we have looked at so far are, on an individual basis, best for one particular demand scenario (i.e., expected, low, etc.). As Dakota embarks on this planning adventure, they do not know what the demand will be, and it is important to obtain a solution that balances the impact of the various scenarios that might be encountered. Because the model that we have used looks at only one demand scenario at a time, it is not able to provide this type of balance. We need a model that explicitly considers the various scenarios... not individually, but collectively. This is precisely the need that stochastic programming models fulfill.

2.3. A Variety of Modeling Choices

In the Dakota problem, there are two types of decisions that are made:

- Production: How many desks, tables, and chairs should be produced?
- Resource: How much lumber, finishing labor, and carpentry labor should be procured?

Between Tables 1 and 3, we have a full description of the data required to develop a more robust model of Dakota's problem. However, there is still a critical piece of information that has not been specified—When do the various decisions have to be made? In complicated problems, decisions and uncertainty are often interwoven over time. That is, we make a decision, we observe what happens, more decisions are made, we observe again, etc. Notice that neither the mean value nor the scenario problems are able to capture this interplay between decisions and uncertainty. Before a stochastic programming model can be developed, the timing of the decisions, relative to the resolution of uncertainty, must be specified.

It is important to recognize that the timing of the decisions is a property of the problem at hand. That is, the modeler must determine when the decisions are made, relative to the resolution of uncertainty. Decisions that can be delayed until after information about the uncertain data is available offer an opportunity to adjust or adapt to the information that is received. We often refer to this adaptation as *recourse*. Models in which some decisions are delayed until after some information about the uncertain data has been obtained are known as recourse models, and decision variables that are permitted to vary with the scenario are sometimes referred to as recourse variables. Decisions either can or cannot be delayed—this is typically beyond our control. However, when it is possible to do so, there is generally value associated with delaying a decision until after additional information is obtained.

[2] To see this, note that the resources will cost $9,835, but the income from the sale of 50 desks and 20 tables is only $3,800.

2.4. A Recourse Model

Recourse models result when some of the decisions must be fixed before information relevant to the uncertainties is available, while some of them can be delayed until afterward. In the Dakota problem, suppose that the resource quantities must be determined fairly early, but production quantities can be delayed until after the demand is known. In some sense, we may think of the production quantities as being *flexible* or *adaptive*, while the resource quantities are not.

To model the production quantities as being dependent on the demand scenario, the production variables that have been used thus far, (y_d, y_t, y_c), will be replaced by $\{(y_{d,\omega}, y_{t,\omega}, y_{c,\omega})\}_{\omega\in\{l,m,h\}}$. For example, the variable $y_{t,h}$ will denote the number of tables that will be produced if the demand is "high," and the other variables are similarly defined. On the other hand, the resource variables do not vary with the scenario, and thus will remain as x_b, x_l, and x_c. Notice that in this case, because production quantities are being determined *after* the demand scenario is known, Dakota will not run the risk of producing items that it is not able to sell.

To draw attention to the structure of the decision problem, we may state this version of Dakota's problem as follows:

$$\begin{aligned} \text{Max} \quad & -2x_b - 4x_f - 5.2x_c + E[h(x,\tilde{d})] \\ \text{s.t.} \quad & x_b, x_f, x_c \geq 0, \end{aligned} \tag{2}$$

where $\tilde{d} = \{\tilde{d}_d, \tilde{d}_t, \tilde{d}_c\}$ represents the uncertain demand for desks, tables, and chairs, and for each scenario $\omega \in \{l, m, h\}$,

$$\begin{aligned} h(x, d_\omega) = \text{Max} \quad & 60y_{d,\omega} + 40y_{t,\omega} + 10y_{c,\omega} \\ \text{s.t.} \quad & 8y_{d,\omega} + 6y_{t,\omega} + y_{c,\omega} \leq x_b \\ & 4y_{d,\omega} + 2y_{t,\omega} + 1.5y_{c,\omega} \leq x_f \\ & 2y_{d,\omega} + 1.5y_{t,\omega} + 0.5y_{c,\omega} \leq x_c \\ & y_{d,\omega} \leq d_{d,\omega} \\ & y_{t,\omega} \leq d_{t,\omega} \\ & y_{c,\omega} \leq d_{c,\omega} \\ & y_{d,\omega}, y_{t,\omega}, y_{c,\omega} \geq 0. \end{aligned}$$

Notice that in this case, the subproblem determines the optimal allocation of resources to products after the demand for products is known. Note also that the resource quantities (x_b, x_f, x_c) are inputs to the subproblem and appear on the right-hand side of the constraints. For the sake of completeness, we present the problem in nondecomposed form as follows:

$$\begin{aligned} \text{Max} \quad & -2x_b - 4x_f - 5.2x_c + \sum_{\omega\in\{l,m,h\}} p_\omega * \{60y_{d,\omega} + 40y_{t,\omega} + 10y_{c,\omega}\} \\ \text{s.t.} \quad & -x_b + 8y_{d,\omega} + 6y_{t,\omega} + y_{c,\omega} \leq 0 \\ & -x_f + 4y_{d,\omega} + 2y_{t,\omega} + 1.5y_{c,\omega} \leq 0 \\ & -x_c + 2y_{d,\omega} + 1.5y_{t,\omega} + 0.5y_{c,\omega} \leq 0 \\ & y_{d,l} \leq 50 \\ & y_{d,m} \leq 150 \\ & y_{d,h} \leq 250 \end{aligned}$$

$$
\begin{aligned}
&y_{t,l} \le 20\\
&y_{t,m} \le 110\\
&y_{t,h} \le 250\\
&y_{c,l} \le 200\\
&y_{c,m} \le 225\\
&y_{c,h} \le 500\\
&x_b, x_f, x_c, y_{i,\omega} \ge 0 \quad \forall\, i \in \{d, t, ch\}, \omega \in \{l, m, h\}.
\end{aligned}
$$

So far, we have looked at three different models of the Dakota problem: one in which the random variables are replaced by their expected values, one in which each scenario is considered separately, and the recourse model. To facilitate comparisons and discussions, we summarize the output from each of these models in Table 5.

There are several interesting observations that Table 5 affords. Notice that in terms of the resource quantities, it is difficult to identify a meaningful connection between the solutions to the scenario problems and the solution to the recourse problem. This contrasts sharply with the solution to the mean value problem, which we have already noted is just the expected value of the individual scenario solutions.

The objective values that are reported in Table 5 are especially confusing. These values range from 1,124 for the low-demand scenario to 7,450 for the high-demand scenario. As a result, a naive modeler might be tempted to look favorably on the high-demand scenario. Recognizing that this scenario will occur with probability 0.3, it is important that the objective values that appear in Table 5 be understood within the context from which they are obtained. For example, the scenario solutions are based on the assumption of perfect information. If we knew for sure that demand would be low, Dakota could obtain a maximum profit of 1,124. Similarly, if we knew for sure that demand would be high, Dakota could obtain a maximum profit of 7,450. The scenario problems consider only one scenario at a time. They do not consider the impact of planning for one scenario and having another scenario materialize. Similarly, the mean value solution plans only for the average demand, and does not consider its impact under the various scenarios. Only the recourse model is designed to encourage a solution that balances the potential impact of the various scenarios. That is, the objective values for the mean value problem and the scenario problems are unrealistically optimistic—they do not include considerations of the impact of variability in

TABLE 5. Output for Dakota models.

		Scenario solutions					
	Mean value	low	m. likely	high	Recourse		
Resource quantities							
Lumber (bd. ft.)	1,950	520	1,860	3,500	1,300		
Fin. labor (hrs.)	850	240	820	1,500	540		
Car. labor (hrs.)	487.5	130	465	875	325		
					Demand		
Production quantities					low	m. likely	high
Desks	150	50	150	250	50	80	80
Tables	125	20	110	250	20	110	110
Chairs	0	0	0	0	200	0	0
Obj. value	4,165	1,124	3,982	7,450	1,730		

Table 6. Expected profit associated with scenario solutions.

Demand scenario	Profit suggested by scenario problem	Expected profit
Expected value	4,165	1,545
Low	1,124	1,124
Most likely	3,982	1,702
High	7,450	−2,050

*Recall that the optimal expected profit is 1,730.

the demand. Consequently, they fail to provide a meaningful representation of the expected profit.

Unlike the mean value and scenario problems, the objective values of the recourse problems correspond to *expected profit*. The flexibility that the recourse model affords is nicely illustrated by the structure of the solution. Notice that it is the only model that suggests that chairs should be made, and then only when the demand for Dakota's products is low. The solution to the recourse model purchases sufficient resources to satisfy demand at some of the higher levels. That is, enough lumber and labor are procured to enable Dakota to address, at least partially, the higher-demand scenarios. If it turns out that the demand is low, Dakota is able to recover much of their resource expense by producing chairs. In the low-demand scenario, some of the lumber and carpentry labor will go unused, but Dakota will make maximum use of the resources that they have purchased. In all cases, Dakota will be able to sell all items produced. Thus, we see that, in reality, chairs are useful as a fallback item. When demand is low, Dakota can cut its losses by producing and selling chairs, which is preferable to producing and *not* selling desks and tables!

We have already noted that the objective values associated with the scenario problems are overly optimistic. These problems are only appropriate if one truly has the ability to adapt fully to the demand once it is known. If they are used inappropriately (e.g., if demand is uncertain at the time that the resource decisions must be made), a mismatch between production and demand is likely to result, and the actual objective values will differ substantially from those that are suggested! In Table 6, we compare the objective value associated with the individual scenario problems and the corresponding expected profits in the recourse setting. Notice the extent to which the scenario objective values overestimate the expected profits.

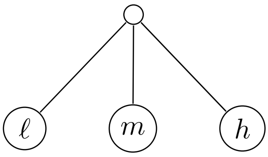
3. Recourse Problems

The formulation produced in §2.4 is known as a recourse problem. The term "recourse" refers to the opportunity to adapt a solution to the specific outcome observed. That is, x corresponds to the resource levels to be acquired, and ω corresponds to a specific data scenario—in this case, the actual demands for the various products. The decision y adapts to the specific combination of x and ω obtained. In the event that the initial decision x is coupled with a bad outcome, the variable y offers an opportunity to recover to the fullest extent possible. Recourse problems are always presented as problems in which there are two or more decision stages.

3.1. Components of a Recourse Problem

Each recourse problem can be characterized by its scenario tree, its scenario problems, and its nonanticipativity constraints. A *scenario* is one specific, complete realization of the stochastic elements that might appear during the course of the problem (e.g., a possible sequence of demand over the duration of the problem). The *scenario tree* is a structured

Figure 1. Scenario tree for the Dakota problem.



distributional representation of the stochastic elements and the manner in which they may evolve over the period of time represented in the problem. A *scenario problem* is associated with a particular scenario and may be looked upon as a deterministic optimization problem (such as those whose solutions are displayed in Table 4). The scenario tree for the problem presented in §2.4 is very simple, involving only a root node and the three individual demand scenarios that are considered. We include it in Figure 1. In general, the nodes in the scenario tree correspond to a model of the manner in which information might become available over time. Although they are not typically depicted as such, nodes in a scenario tree are associated with a subset of decisions. The root node corresponds to the initial decision stage, during which no specific information regarding the random variables has been obtained. The leaf nodes correspond to the final decisions required, which are made after all available information has been obtained. In the Dakota problem, the resource decisions are associated with the root node while the production decisions are associated with the leaf nodes.

Depending on the manner in which the problem is formulated, it may be necessary to include specific conditions to ensure that the decision sequence honors the information structure associated with the scenario tree. These conditions are known as the *nonanticipativity constraints*, and impose the condition that scenarios that share the same history (of information) until a particular decision epoch should also make the same decisions. In reality, the nonanticipativity constraints ensure that the solutions obtained are implementable, i.e., that actions that must be taken at a specific point in time depend only on information that is available at that time. For that reason, the terms nonanticipativity and implementability are sometimes used interchangeably. These nonanticipativity constraints, which are derived from the scenario tree, are a distinguishing characteristic of stochastic programs—solution methods are typically designed to exploit their structure. The recourse formulation used to describe the Dakota problem implicitly honors the information structure. This is because the resource variables (which are associated with the root node in the scenario tree) do not vary with the scenario, while the production variables (which are associated with the leaf nodes in the scenario tree) are permitted to vary with the scenario. In §3.7 we will discuss formulations in which nonanticipativity constraints are specifically included.

Throughout this paper, $\tilde{\omega}$ will represent the uncertain data associated with the decision problem. In this sense, $\tilde{\omega}$ is a random variable defined on a probability space $(\Omega, \mathcal{A}, \mathcal{P})$. When $\tilde{\omega}$ is a discrete random variable, $p_\omega = \mathcal{P}\{\tilde{\omega} = \omega\}$ for each scenario $\omega \in \Omega$. Decision variables, matrices, etc, are defined in the same manner as for traditional linear programming problems. However, elements that might vary with the scenario will be identified by a subscript ω. For example, a set of constraints

$$T_\omega x + W_\omega y_\omega = r_\omega$$

would indicate that the matrices (T_ω and W_ω) and right-hand-side vector (r_ω) contain elements that are specific to the scenario ω. Additionally, the vector y_ω might vary with the specific scenario, while the vector x cannot.

Finally, it is important to note that, in general, *randomness* within the data tends to be very sparse. That is, more often than not only a small number of elements in the matrices and vectors that are subject to uncertainty are represented as random variables—most elements are typically fixed. For example, in a max-flow problem, it may be the case that arc capacities are modeled as random variables. However, the right-hand sides in flow conservation constraints remain fixed at zero and the data used to describe the flows (i.e., the incidence matrix) are similarly fixed.

3.2. Two-Stage Recourse Problems

The SLP formulated in §2.4 is an example of a two-stage problem, which has the general form:

$$\begin{aligned} \text{Min} \quad & cx + E[h(x,\tilde{\omega})] \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0 \end{aligned} \tag{3}$$

$$\text{where} \quad h(x,\omega) = \begin{aligned} \text{Min} \quad & g_\omega y \\ \text{s.t.} \quad & W_\omega y \geq r_\omega - T_\omega x \\ & y \geq 0. \end{aligned} \tag{4}$$

In this case, x is known as a first-stage decision (also known as the *here and now* decision). Note that in this model, x does not respond to ω—it is effectively determined before any information regarding the uncertain data has been obtained. On the other hand, y—the second-stage variable—is determined after observations regarding $\tilde{\omega}$ have been obtained. The problem in (4) is known variously as the second-stage problem, the subproblem, or the recourse subproblem. In essence, the goal of a two-stage model is to identify a first-stage solution that is well positioned against all possible observations of $\tilde{\omega}$. An optimal solution will tend to have the property that x leaves the second-stage decision in a position to exploit advantageous outcomes of $\tilde{\omega}$ without excessive vulnerability to disadvantageous outcomes. Note that in this case, careful attention to the specific outcomes used to model the uncertainty is necessary. An overly coarse model may result in a failure to adequately represent outcomes that should influence the first-stage decision, leaving the second stage in an unmodeled state of vulnerability. An excessively fine model may result in increased computational burden.

As presented, without assuming any additional properties or structure on (4), we would describe the problem as having *general recourse.* In many cases, there is specific structure in the recourse subproblem that can be exploited for computational advantage. Some of the more commonly occurring types of structure are described in the following sections.

3.3. Simple Recourse

A special case of the recourse model, known as the simple recourse model, arises when the constraint coefficients in the second-stage problem, W, form an identity matrix so that

$$\begin{aligned} h(x,\omega) = \text{Min} \quad & g_\omega^+ y^+ + g_\omega^- y^- \\ \text{s.t.} \quad & I y^+ - I y^- = r_\omega - T_\omega x \\ & y^+, y^- \geq 0. \end{aligned} \tag{5}$$

For any right-hand-side vector, $r_\omega - T_\omega x$, a feasible solution to (5) is easily determined by simply noting the sign of the right-hand side of each constraint and setting y^+ and y^- accordingly. Furthermore, provided that the ith component of $g_\omega^+ - g_\omega^-$ satisfies $g_{\omega i}^+ - g_{\omega i}^- > 0$ for each i, this feasible solution is also optimal.

It is worth noting that when a problem has simple recourse, the subproblem (5) can be equivalently represented as

$$h(x,\omega) = \sum_i h_i(x,\omega)$$

where

$$\begin{aligned} h_i(x,\omega) = \text{Min} \quad & g^+_{\omega i} y^+_i + g^-_{\omega i} y^-_i \\ \text{s.t.} \quad & y^+_i - y^-_i = r_{\omega i} - \{T_\omega x\}_i \\ & y^+_i, y^-_i \geq 0. \end{aligned}$$

That is, the second-stage problem is separable by row. As a result, only the marginal distributions of the right-hand-side vector, $r_{\tilde{\omega}} - T_{\tilde{\omega}} x$, are necessary to calculate the expected value of the second-stage objective function values, which eases their calculation considerably.

Simple recourse problems arise in numerous situations. For example, when *target values* can be identified, and a primary concern involves minimizing deviations from these target values (although these might be weighted deviations), a simple recourse problem results.

3.4. Fixed Recourse

Another case that often arises is a property that is known as *fixed recourse.* A fixed recourse problem is one in which the constraint matrix in the recourse subproblem is not subject to uncertainty (i.e., it is fixed). In this case, the recourse subproblem is given by:

$$\begin{aligned} h(x,\omega) = \text{Min} \quad & g_\omega y \\ \text{s.t.} \quad & Wy \geq r_\omega - T_\omega x \\ & y \geq 0. \end{aligned}$$

Note that the simple recourse problem has fixed recourse. This representation of $h(x,\omega)$ is apparently not much different from (4). However, when the second-stage objective coefficients are also fixed, the dual representation of the recourse subproblem is given by

$$\begin{aligned} h(x,\omega) = \text{Max} \quad & \pi^\top (r_\omega - T_\omega x) \\ \text{s.t.} \quad & \pi^\top W \leq g^\top \\ & \pi \geq 0. \end{aligned} \tag{6}$$

In this case, the set of dual feasible solutions is fixed (i.e., does not vary with ω), a property that can be exploited computationally while designing a solution method.

3.5. Complete Recourse

So far, our focus has been on properties that arise from the recourse problem data. The reader will note that our presentation of the recourse problems suggests a decomposition of the problem into a first and second-stage problem. Indeed, many solution procedures exploit this opportunity for decomposition. In this setting, a question arises that involves feasibility of a particular first-stage vector x. That is, what assurances are there that the recourse function $h(x,\omega)$ is necessarily finite?

Note that $E[h(x,\tilde{\omega})] < \infty$ as long as the recourse subproblem (4) is feasible for all x. A problem for which $Y(\omega,\chi) = \{y \mid W_\omega y \geq \chi\}$ is nonempty for any value of χ is said to have complete recourse. If a problem has complete recourse, the recourse function is necessarily finite. A slightly less strenuous property, which leads to the same result, is known as "relatively complete recourse." Relatively complete recourse results if $Y(\omega,\chi)$ is nonempty for all $\chi \in \{r_\omega - T_\omega x \mid (\omega, x) \in \Omega \times X\}$. That is, relatively complete recourse merely restricts the statement of complete recourse to those values of the right-hand-side vector that can be encountered.

Complete recourse and relatively complete recourse may sound like extremely difficult properties to ensure, but in fact it is quite easy to guarantee their existence while a model is formulated. For example, by penalizing deviations from feasibility in the recourse subproblem as follows:

$$\begin{aligned} h(x,\omega) = \text{Min} \quad & g_\omega y + Me^\top z \\ \text{s.t.} \quad & Wy + z \geq r_\omega - T_\omega x \\ & y, z \geq 0 \end{aligned}$$

(where M is a large constant and e is an appropriately dimensioned vector of ones), the problem has complete recourse. This type of modeling technique is commonly employed by stochastic programmers. Note that penalizing deviations from the original model in (4) tends to promote feasibility in the first-stage decision. Perhaps more importantly, a formulation such as this does not promote solutions that are overly influenced by rare events with extreme values.

3.6. Scenario Formulations

There are several ways to formulate an SLP. Thus far, our focus has been on formulations that explicitly represent the information process (as modeled by the scenario tree) within the sequence of decisions that are made. An alternate, but equally valid, representation of the problem is one in which a problem is formulated for each possible scenario and constraints are added to ensure the information structure associated with the decision process is honored. In this case, we begin by representing all decision variables as if they were permitted to depend on the specific scenario encountered, which leads to the scenario problems for each $\omega \in \Omega$:

$$\begin{aligned} \text{Min} \quad & cx_\omega + g_\omega y_\omega \\ \text{s.t.} \quad & T_\omega x_\omega + W_\omega y_\omega \geq r_\omega \qquad (7) \\ & x_\omega, y_\omega \geq 0. \end{aligned}$$

Without the introduction of additional constraints, we obtain a situation in which $\{(x_\omega, y_\omega)\}_{\omega\in\Omega}$ vary freely in response to each specific scenario. This runs contrary to the notion that some decisions can respond to the specific scenario, while others cannot. We can remedy this by including constraints that ensure that the decision sequence honors the information structure present in the problem as follows:

$$\begin{aligned} \text{Min} \quad & \sum_{\omega\in\Omega} (cx_\omega + g_\omega y_\omega) p_\omega \qquad (8) \\ \text{s.t.} \quad & T_\omega x_\omega + W_\omega y_\omega \geq r_\omega \\ & x_\omega - x = 0 \quad \forall \omega \in \Omega \qquad (9) \\ & x_\omega, y_\omega \geq 0. \end{aligned}$$

Recall that for each $\omega \in \Omega$, $p_\omega = P\{\tilde{\omega} = \omega\}$ so that the objective in (8) represents the expected value as in (3). Constraints such as (9) are known as nonanticipativity constraints and ensure that decisions honor the information structure of the problem. Note that in (9) we have used a free variable, x, to constrain the scenario-dependent, first-stage variables $\{x_\omega\}_{\omega\in\Omega}$ to be equal. There are numerous ways in which these constraints might be represented. For example, in (9) we might replace x with $E[x_{\tilde{\omega}}] = \sum_{\omega\in\Omega} p_\omega x_\omega$, as in Dempster [12] and in Rockafellar and Wets [40]. Alternatively, one might consider a more sophisticated representation that results in sparser constraints, such as one finds in Mulvey and Ruszczyński [32]. In general, the precise manner in which the nonanticipativity constraints are represented depends on the analysis and/or solution methodology to be undertaken. We note that when an SP is explicitly presented in its full form, as in (8), it is sometimes referred to as the deterministic equivalent problem (DEP). Properties and characteristics of the DEP are discussed in Wets [48].

3.7. Multistage Recourse Problems

Our focus thus far has been on two-stage problems with recourse, in which an initial decision is made while the specific scenario to be obtained is unknown, followed by another decision that is made after this information is available. It is not difficult to envision situations in which this "decide-observe-decide..." pattern is repeated several times. This leads to a multistage recourse problem. Formulating a multistage recourse problem can become a delicate operation due to the manner in which decisions and observations are interspersed. In this section, we will simply introduce a scenario formulation and indicate a method for identifying the nonanticipativity constraints.[3]

To begin, for each scenario $\omega \in \Omega$, let c_ω represent the objective function coefficients corresponding to the scenario and let $X(\omega)$ denote the set of solutions that are feasible for the scenario. That is, if there were exactly one data scenario to consider, the problem would be represented as:

$$\begin{aligned} \text{Min} \quad & c_\omega x_\omega \\ \text{s.t.} \quad & x_\omega \in X(\omega). \end{aligned} \tag{10}$$

In general, the scenario constraints (10) are represented as multistage constraints:

$$\sum_{j=1}^{t} A_{tj} x_j = b_t \quad t = 1, \ldots, T$$

so that the actions taken at stage t are constrained by actions taken earlier in the process. If $\mathcal{N}$ denotes the set of nonanticipative solutions, then a multistage problem can be expressed as:

$$\text{Min} \quad \sum_\omega p_\omega c_\omega x_\omega \tag{11}$$

$$\begin{aligned} \text{s.t.} \quad & x_\omega \in X(\omega) \quad \forall\, \omega \in \Omega \\ & \{x_\omega\}_{\omega\in\Omega} \in \mathcal{N}. \end{aligned} \tag{12}$$

As we have mentioned previously, the nature of the nonanticipativity constraints in (12) depends on the specific structure of the scenario tree. Suppose that we have a scenario tree as depicted in Figure 2. Note that in this case we have depicted a tree for a four-stage problem. In general, each node in the scenario tree corresponds to a collection of scenarios at a specific stage. Consider the node marked n in the scenario tree, and note that it corresponds to a stage in the problem, $t(n)$.[4] Let the set of scenarios that pass through node n be denoted as $\mathcal{B}(n)$, as depicted by the darkened scenarios in Figure 2. In the second stage, these scenarios cannot be distinguished from each other—while it is possible to recognize that the data indicates that it corresponds to node n, it is not possible to recognize which of the scenarios in $\mathcal{B}(n)$ will ultimately result. For solutions to the problem to be implementable (i.e., a.k.a. nonanticipative), we must ensure that decision variables that are associated with node n produce identical values. One way to do this is to include constraints of the following form:

$$x_{t(n)\omega} - x_n = 0 \quad \forall\, \omega \in \mathcal{B}(n).$$

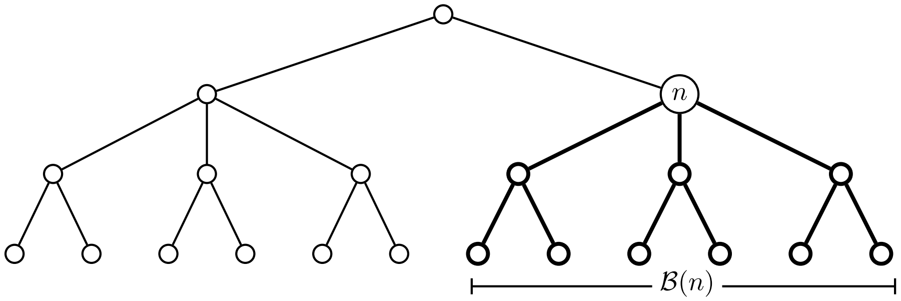
Note the similarity between this form of the constraint and (9). If we let N denote the set of nonleaf nodes in the scenario tree, then we may represent the set of nonanticipative solutions as:

$$\mathcal{N} = \big\{\{x_\omega\}_{\omega\in\Omega} \mid x_{t(n),\omega} - x_n = 0 \;\; \forall\, \omega \in \mathcal{B}(n), \;\; \forall\, n \in N\big\}.$$

[3] If we adopt a decision-stage formulation similar to (3), then $h(x,\omega)$ includes the expected "cost-to-go" function associated with later decision stages.

[4] In this case, $t(n) = 2$.

FIGURE 2. Bundles within a scenario tree.



Finally, as previously noted, the representation of the nonanticipativity constraints is not unique—there are any number of choices available. The specific choice selected is typically guided by the solution method to be used.

3.8. Solutions to Recourse Problems

Finally, it is necessary to comment on the nature of a solution to these problems, which involve multiple (i.e., two or more) decision stages. In deterministic linear programming, we are accustomed to specifying the entire solution vector—indicating a value (zero or otherwise) for each individual variable. If we consider this within the context of a two-stage problem, that would require reporting values for x as well as for $\{y_\omega\}_{\omega \in \Omega}$—a task that can quickly become daunting. Note that if there are only 10 random variables within the data elements, and these are modeled as independent random variables with only three possible outcomes each (corresponding to high, medium, and low values), then Ω contains $3^{10} = 59{,}049$ separate data scenarios. For this reason, the reporting of stochastic programming solutions is typically restricted to the first-stage variables. Note that this is especially appropriate when considering the fact that this is the action that requires immediate commitment—all other decisions can be delayed until further information is postponed.

4. Does Uncertainty Matter?—A Quick Check

The Dakota example in §2.1 illustrates some of the ways in which deterministic models combined with investigations of solution sensitivity do not adequately represent opportunities to adapt to information obtained at intermediate stages of the decision sequence. The example also illustrates the manner in which a stochastic linear programming formulation might differ from a related deterministic linear programming formulation. For one thing, we see that the size of the problem increases, and we can easily imagine that solution difficulties increase as well. In fact, as the number of scenarios that must be considered increases, hopes of solving the resulting problem using general purpose, off-the-shelf LP solvers are quickly abandoned in favor of specialized solution methods. Prior to solving an SLP, it is useful to investigate the quality of the solution that can be obtained via the more easily solved deterministic LP.

We return to the general structure of the recourse problem (3)–(4),

$$\begin{aligned} \text{Min} \quad & cx + E[h(x,\tilde{\omega})] \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0 \end{aligned}$$

$$\text{where} \quad h(x,\omega) = \begin{aligned} \text{Min} \quad & g_\omega y \\ \text{s.t.} \quad & W_\omega y \geq r_\omega - T_\omega x \\ & y \geq 0. \end{aligned}$$

Note that the function, $h(x,\omega)$, is defined as the value function of the second-stage linear program that appears in (4), and that the vector x appears on the right-hand side of this minimization problem. The dual to (4) is given by

$$\begin{aligned} h(x,\omega) = \text{Max} \quad & \pi^\top(r_\omega - T_\omega x) \\ \text{s.t.} \quad & \pi^\top W_\omega \le g_\omega^\top \\ & \pi \ge 0. \end{aligned}$$

Using this dual representation of $h(x,\omega)$, it is a relatively simple exercise to verify that it is a piecewise linear convex function of the variable x. If the sample space of $\tilde{\omega}$ is countable, the expected value of this function, which appears in the objective of (3), is simply

$$E[h(x,\tilde{\omega})] = \sum_{\omega\in\Omega} h(x,\omega)p_\omega. \tag{13}$$

Convexity is preserved through this operation. In general, piecewise linearity is preserved as well.[5] When the problem has fixed recourse, so that uncertainty is not present in the second-stage constraint matrix, W, and the second-stage objective coefficients, g, are fixed as well, similar arguments will ensure that $h(x,\omega)$ is also a convex function of ω.

Jensen's inequality, which involves convex functions of random variables, applies in this case and offers a simple method for bounding the objective value improvement that might be obtained via solution as an SLP. Jensen's inequality ensures that when $h(x,\omega)$ is convex in ω and $\tilde{\omega}$ is a random variable, then

$$h(x,E[\tilde{\omega}]) \le E[h(x,\tilde{\omega})].$$

Note that if $X = \{x : Ax \ge b,\ x \ge 0\}$ (i.e., all x that satisfy the first-stage constraints in (3)), then

$$\begin{aligned} & cx + h(x,E[\tilde{\omega}]) \le cx + E[h(x,\tilde{\omega})] \quad \forall x \in X \\ \Rightarrow \quad & \operatorname*{Min}_{x\in X}\{cx + h(x,E[\tilde{\omega}])\} \le \operatorname*{Min}_{x\in X}\{cx + E[h(x,\tilde{\omega})]\}. \end{aligned} \tag{14}$$

Equation (14) indicates an ordering in the optimal objective function values for two distinct, yet related, problems. On the left-hand side, we have the case in which all random elements are replaced by their expected values—the so-called *mean value problem.* On the right-hand side, we have the SLP. Note that (14) indicates that the optimal objective value associated with the SLP is bounded by the optimal value of the mean value problem. Let

$$\begin{aligned} \bar{x} &\in \arg\min\{cx + h(x,E[\tilde{\omega}]) \mid x \in X\} \\ x^* &\in \arg\min\{cx + E[h(x,\tilde{\omega})] \mid x \in X\} \end{aligned}$$

and note that $c\bar{x} + h(\bar{x},E[\tilde{\omega}]) \le cx^* + E[h(x^*,\tilde{\omega})]$. Note also that because $\bar{x} \in X$, we have that $cx^* + E[h(x^*,\tilde{\omega})] \le c\bar{x} + E[h(\bar{x},\tilde{\omega})]$. In combination, this yields

$$c\bar{x} + h(\bar{x},E[\tilde{\omega}]) \le cx^* + E[h(x^*,\tilde{\omega})] \le c\bar{x} + E[h(\bar{x},\tilde{\omega})]. \tag{15}$$

The inequalities in (15) suggest a fairly straightforward method for quickly determining whether or not solving the problem as an SLP is worth the effort required:

Step 1. Solve $\text{Min}_{x\in X}\{cx + h(x,E[\tilde{\omega}])\}$ to obtain $\bar{x}$.

Step 2. Evaluate $E[h(\bar{x},\tilde{\omega})]$.

Step 3. If $E[h(\bar{x},\tilde{\omega})] - h(\bar{x},E[\tilde{\omega}])$ is sufficiently small, accept $\bar{x}$ as an acceptable solution.

[5] In general, the expectation is calculated via integration. In some special cases when the random variables are absolutely continuous, the function is smooth rather than piecewise linear. However, convexity in x is preserved nonetheless.

The gap identified in Step 3 is an upper bound on the loss of optimality associated with using $\bar{x}$ in lieu of identifying x^*. When this gap is sufficiently small, there is no need to invest further effort in pursuit of an optimal solution. Note that the precise evaluation of the expected value indicated in Step 2 may be difficult to undertake. In this case, statistical estimation techniques are suggested. For example, suppose that $\{\omega^t\}_{t=1}^N$ is a large number of randomly generated observations of $\tilde{\omega}$. Then $E[h(\bar{x},\tilde{\omega})]$ can be approximated using the sample mean, $(1/N)\sum_{t=1}^N h(\bar{x},\omega^t)$, and confidence statements regarding the accuracy of the estimated value are readily obtained.

For additional methods that can be used to estimate the potential value associated with solving the stochastic program (i.e., as compared to simply using the solution to the mean value problem), the reader is referred to Birge [4]. Note that (15) makes use of upper and lower bounds on the optimal SLP objective function value. The topic of upper and lower bounds in SLP has been extensively studied. The reader is referred to Birge and Wets [6], Edirisinghe and Ziemba [14], and Frauendorfer [15] for further comments on more involved bounding techniques.

5. Solution Approaches

By now, it is probably clear that when $\tilde{\omega}$ involves discrete random variables, a stochastic linear program is really a specially structured linear program. If the number of scenarios is small enough, the SLP can be solved using an off-the-shelf linear programming solver. In general, however, the number of scenarios can become explosively large. For example, when the random variables are independent, the number of scenarios is the product of the number of possible outcomes for each marginal random variable, which can lead to an explosive number of possible outcomes. When this occurs, it is necessary to use solution methods that are specifically designed to exploit the structural properties of the stochastic program. These methods typically involve a decomposition of the problem, and increasingly often use statistical estimation methods as well.

5.1. Decomposition

The first solution procedure proposed for two-stage stochastic linear programs with recourse is the L-shaped method (van Slyke and Wets [47]). The L-shaped method decomposes the problem by stage—the first-stage problem leads to a master problem and the second-stage problem leads to a subproblem. In reality, the method is simply an adaptation of Benders' decomposition [2] to the structure of the second-stage problem. Beginning with the problem statement as in (3)–(4), the second-stage objective function $E[h(x,\tilde{\omega})]$ is approximated using a piecewise linear convex function $\nu(x)$, where

$$\nu(x) = \mathrm{Max}\{\alpha_t + \beta_t x \mid t = 1,\dots,k\}.$$

The approximation is developed iteratively, and $\nu(x)$ is typically represented in a master program using a cutting-plane approximation:

$$\begin{aligned} \text{Min} \quad & cx + \nu \\ \text{s.t.} \quad & Ax \ge b \\ & \nu \ge \alpha_t + \beta_t x \quad t = 1,\dots,k \\ & x \ge 0. \end{aligned} \tag{16}$$

The coefficients on these cutting planes are obtained from dual solutions to (4). That is,

$$\begin{aligned} h(x,\omega) &= \mathrm{Min}\big\{g_\omega y \mid W_\omega y \ge r_\omega - T_\omega x,\ y \ge 0\big\} \\ &= \mathrm{Max}\big\{\pi^\top (r_\omega - T_\omega x) \mid \pi^\top W \le g^\top,\ \pi \ge 0\big\}. \end{aligned}$$

Let $\Pi = \{\pi \mid \pi^\top W \le g^\top, \pi \ge 0\}$, and note that for each $\pi \in \Pi$,

$$h(x,\omega) \ge \pi^\top (r_\omega - T_\omega x)$$

with equality holding when $\pi \in \arg\max\{\pi^\top (r_\omega - T_\omega x) \mid \pi \in \Pi\}$. Consequently, if x^k is a solution to (16) obtained in the kth iteration, and $\pi(x^k,\omega) \in \arg\max\{\pi^\top (r_\omega - T_\omega x^k) \mid \pi \in \Pi\}$, then the next cut to be added to the piecewise linear convex approximation of $E[h(x,\tilde{\omega})]$ is given by:

$$\alpha_{k+1} + \beta_{k+1} x = \sum_{\omega \in \Omega} \pi^\top (x^k,\omega)^\top (r_\omega - T_\omega x) p_\omega.$$

In representing the cuts in this manner, a property of separability in the subproblem has been exploited. Formally, Benders' decomposition would define the subproblem as $E[h(x,\tilde{\omega})]$, a single problem involving all possible scenarios. However, because

$$\begin{aligned} E[h(x,\tilde{\omega})] &= \mathrm{Min}\left\{ \sum_{\omega \in \Omega} p_\omega g_\omega y_\omega \,\middle|\, W_\omega y \ge r_\omega - T_\omega x,\ y_\omega \ge 0\ \forall\ \omega \in \Omega \right\} \\ &= \sum_{\omega \in \Omega} p_\omega \mathrm{Min}\left\{ g_\omega y_\omega \,\middle|\, W_\omega y \ge r_\omega - T_\omega x y_\omega \ge 0 \right\}, \end{aligned}$$

the L-shaped method is able to define cutting-plane coefficients from the individual (scenario) subproblems. This results in a substantial reduction in the size of the subproblems that are solved (although it increases the number of such problems that are solved in any iteration).

If the problem has either complete or relatively complete recourse, this presentation of the L-shaped method is sufficient to ensure that an optimal solution to the SLP can be identified. If the problem is not "blessed" with one of these useful properties, it is possible that for some values of x^t one or more of the scenario subproblems are infeasible. This means that x^t is not a feasible solution in the first place, and constraints must be added to the master program to eliminate it from further consideration. This is accomplished using extreme rays from the subproblem dual as in Benders' decomposition.

In many ways, the L-Shaped method is by now a classic technique for solving two-stage stochastic programs. Although it is well suited for an introduction to SLP solution methodology, it is no longer computationally effective for large-scale problems.[6] One of the first major improvements to this basic methodology involved the introduction of a regularizing term (a quadratic proximal term added to the objective, $\|x - x^k\|^2$, for example) in Ruszczyński [42]. The inclusion of this term imparts mathematical properties on the master problem that can be exploited computationally. Simultaneously, Ruszczyński [42] suggests a multicut representation of the second-stage objective approximation (i.e., one cut per scenario, rather than a single cut on the expected objective function), an adaptation of the basic cutting-plane methodology later investigated in Birge and Louveaux [5].

5.2. Statistically Based Methods

One of the major handicaps of the L-shaped method is the need to solve a subproblem for each scenario (i.e., for all $\omega \in \Omega$) for each cutting plane. In large-scale problems, the number of scenarios is much too high for this to be a reasonable approach, and the possibility of using statistical estimations of the recourse function becomes computationally attractive.

Conceptually, the simplest method for incorporating statistical approximations in the solution procedure is to replace the recourse function, $[h(x,\tilde{\omega})]$, by a sample mean approximation.

[6] In this sense, "large scale" typically refers to the number of scenarios in Ω.

That is, if $\{\omega^t\}_{t=1}^n$ is a collection of independent and identically distributed observations of $\tilde{\omega}$, then one might consider undertaking the solution of *the sample mean problem*:

$$\begin{aligned} \text{Min} \quad & cx + \frac{1}{n}\sum_{t=1}^{n} h(x,\omega^t) \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0. \end{aligned} \tag{17}$$

Notice that (17) merely replaces the original distribution of $\tilde{\omega}$ with the empirical distribution associated with the sample. In other words, the sample mean problem is a stochastic program with an alternate distribution! When the sample size is significantly smaller than the number of scenarios in the original distribution, (17) will be much easier to solve than (3).

On the surface, this approach is quite appealing. However, we note that the solution obtained is dependent on the specific sample that was drawn. Consequently, it is subject to error in the same manner that the mean value solution is subject to error. In the absence of relatively complete recourse, it is possible that the solution obtained is actually infeasible! That is, there may be some scenarios that are not represented in the sample for which the solution to (17) is not feasible. For this reason, relatively complete recourse is a critical property for problems that are to be solved using statistical approximation schemes. A more subtle difficulty with this approach resides in the objective function values obtained from (17). In §4 we use Jensen's inequality to establish a lower bound on the optimal objective value. In [30], Mak et al. use similar logic to establish that the optimal value for (17) provides a biased estimator for the optimal value for (3). That is, the optimal value for (17) is *expected* to be lower than the optimal solution to (3). The reason for this may be fairly clear—solution of the mean value problem permits an optimal adjustment to a small subset of the scenarios in Ω—while solution of the stochastic program requires that the entire set of scenarios be considered.

Based on results in Mak et al. [30], efforts have been made to eliminate this bias and potential error in the solution of the mean value problem. In essence, the suggestion is to undertake the solution of the mean value problem a number of times using a series of independent samples. That is, if M separate samples are obtained (each with n observations), then M separate solutions are obtained, denoted as $\{x_{mv}^m\}_{m=1}^M$. These M solutions are candidate solutions, from which the "apparently best" candidate is selected. To compare solutions, $cx_{mv}^m + E[h(x_{mv}^m, \tilde{\omega})]$ is estimated for $m = 1, 2, ..., M$. Of course, given that Ω involves a large collection of scenarios, these objective function values are estimated statistically based on a random sample that is independent from the samples used to obtain the candidate solutions. This process is referred to as the *sample average approximation method* (Shapiro [44] and Kleywegt et al. [28]), although it is related to *sample path optimization* (Robinson [37] and Fu et al. [35]). There is some evidence that when the objective function values are easy to estimate with only a small number of observations (i.e., when the variability in the objective function values is rather low), this replication can be undertaken without excessive computational burden—otherwise, the replication can become somewhat cumbersome (see, e.g., Higle and Zhao [25]).

5.3. Stochastic Decomposition

One method that specifically integrates elements of decomposition techniques and statistical approximation techniques is stochastic decomposition (SD) (Higle and Sen [21, 22]). Unlike the sample mean optimization in (17), which operates with a fixed sample size, SD operates with an adaptive sample size—increasing the sample size as iterations progress. Unlike the L-shaped method, which solves a subproblem for each scenario for each cutting plane constructed, SD uses recursive approximation methods based on previously solved problems to bypass the solution of the vast majority of the subproblems that would otherwise be solved. The combination of adaptive sampling and subproblem approximations has proven to be quite powerful, especially when a regularized master program (as described in Higle and Sen [20]) is used (see, e.g., Higle and Zhao [25]).

5.4. Methods for Multistage Problems

It is probably obvious in §3.7 that solving multistage problems can be substantially more complex than solving two-stage problems. With multistage problems, it is especially important to exploit the opportunity to decompose the problem.

Early methods for solving multistage problems decomposed the problem by decision stage, bearing a strong similarity to the computation of the cost-to-go-function commonly employed in dynamic programming methods. Among these are nested decomposition (Birge [3]) and MSLiP (Gassmann [17]). Recall that the scenario tree is a critical construct in a stochastic programming model—and that each node in the scenario tree can be viewed as being associated with a linear program. In a stage decomposition of the problem, an approximation of the value function associated with the linear program is developed for each node in the scenario tree. For this reason, careful attention is paid to the manner in which nodes in the scenario tree are selected for improvement of the local approximation.

In many cases, it may be more advantageous to consider a scenario decomposition of the problem. Note that in (11)–(12), the nonanticipativity constraints are the only constraints that link scenarios together within the formulation. That is, in the absence of these nonanticipativity constraints, the multistage formulation naturally decomposes by scenario. As a result, a scenario decomposition can be undertaken via a Lagrangian relaxation of the nonanticipativity constraints as in the progressive hedging algorithm of Rockafellar and Wets [40] or the augmented Lagrangian decomposition method in Rosa and Ruszczyński [41], as well as the closely related diagonal quadratic approximation method in Mulvey and Ruszczyński [31, 32]. Alternatively, a scenario decomposition can be obtained via a dual representation of the multistage SLP, as discussed in Higle et al. [19]. It is interesting to note that Higle et al. [19] combine a scenario decomposition of the problem with statistical approximation methods.

6. Computational Illustration

To appreciate the scale of stochastic programming problems and the operating characteristics of some of the solution methodologies, a brief computational study is in order. A small collection of problem instances, which are well referenced in the stochastic programming literature and publicly available, are used. These problems are characterized as two-stage stochastic linear programs with general and complete recourse. The specific problems that we considered are:

PGP2 is a small power generation planning problem described in Higle and Sen [20]. The first stage involves decisions regarding generator capacities, while the second-stage models generator operations to meet demand for power, which is modeled with random variables.

20Term is a motor freight scheduling problem described in Mak et al. [30]. The first-stage models the position of fleet vehicles at the start of the day, while the second-stage models decisions regarding shipments of freight by these vehicles. The point-to-point demand for goods is modeled using random variables.

STORM is a large-scale air freight scheduling model arising in military logistics described in Mulvey and Ruszczyński [32]. Demand for freight movement is modeled using random variables.

SSN is a large-scale telecommunication network planning problem described in Sen et al. [43]. The first stage involves decisions regarding network link capacities, while the

Table 7. Test problems.

	First stage		Second stage			
Problem	Rows (m_1)	Columns (n_1)	Rows (m_2)	Columns (n_2)	No. of RVs	No. of scenarios (N_Ω)
PGP2	2	4	7	12	3	576
SSN	1	89	175	706	86	$> 5^{86}$
20Term	3	63	124	764	40	2^{40}
STORM	59	128	526	1259	118	5^{118}

second-stage models network operations (call routing) to meet demand, which is modeled using random variables.

To appreciate the size of the full representation of the SLP as in (3), the number of rows is $m_1 + N_\Omega m_2$ and the number of columns is $n_1 + N_\Omega n_2$. In this sense, it is clear that while PGP2 is a small problem, the remaining problems are quite large.

Prior to solving these problems, we begin with a quick check to determine whether or not it appears necessary to consider the stochastic elements of the problem. The results of these calculations appear in Table 8. The upper and lower bounds are calculated as described in (15). The relative error is calculated as the difference between the upper and lower bounds, relative to the lower bound. PGP2 involves a relatively small sample space, with $N_\Omega = 576$. Consequently, $E[h(\bar{x}, \tilde{\omega})]$ can be calculated precisely. Because the remaining problems involve sample spaces that are much too large to work with directly, the upper bounds are estimated statistically and are accurate to within 1% (with 95% confidence). Note that despite being the largest problem with regard to row, column, and scenario dimensions in Table 7, STORM should be considered to be the easiest problem among the group! The error associated with the mean value solution is estimated to be 0.2%. Given that the error in the upper bound estimate is 1%, this means that the estimated error is not significantly different from zero! It is clear that solving this particular problem as a stochastic program offers negligible value beyond the solution of the mean value problem. For this reason, we will omit it from further consideration and focus our attention on the remaining problems, for which uncertainty plays a more significant role.

In Table 9, we compare solutions obtained via the L-shaped method and those obtained via the regularized SD (see van Slyke and Wets [47], and Higle and Sen [22], respectively). A few comments about the manner in which these computations were performed are in order. SD is a statistically based decomposition method, and uses randomly generated observations of $\tilde{\omega}$. For this reason, the values reported are the averages over 10 independent replications, with standard deviations reported parenthetically. Within SD, the sample is increased in each iteration, so that the sample size depends on the number of iterations executed prior to termination of the method. For these computations, SD was terminated based on indications that the solution's objective value was nearly optimal, using the specific rules discussed in Higle and Zhao [25]. As noted, PGP2 is a small problem, and complete enumeration of the set of scenarios does not impose a significant computational burden. For this reason, PGP2 was solved exactly using the L-shaped method. For the remaining problems, precise solution

Table 8. Bounds based on mean value solutions, $\bar{x}$.

Problem	Lower bound	Upper bound	Relative error
PGP2	428.5	505.3	0.179
SSN	0	73.4	(zero-divide)
20Term	239273	279334	0.167
STORM	15,524,803	15,560,749	0.002

Table 9. Illustrative computational results.

	L-shaped		Stochastic decomposition	
Problem	Time	Objective value	Time	Objective value
PGP2	2.4	447.32	4.9 (4.3)	448.7 (1.31)
SSN	36,893 (5,962)	10.02 (0.06)	2,913 (1,391)	10.26 (0.14)
20Term	11,757 (816.5)	254,744 (57)	250 (32)	254,581 (79)

using the L-shaped method is not possible, so a random sample was used to create a sample average approximation of the problem, as in (17). In these cases, the sample used was the same sample used by SD. In this fashion, it is possible to note the relative effort required by SD and L-shaped methods.

In Table 9, time is measured in seconds. Note that for the smaller problem, PGP2, the L-Shaped method performs quite well—there is no apparent need for a more sophisticated methodology for problems of this size. However, the time required to solve these problems using the L-shaped method on the sample average approximation of the problem is substantially larger than for the regularized SD, even though both methods use an identical set of observations! For SSN, the solution times for SD are less than 10% of the solution times for the L-shaped method, while for 20Term, the SD solution times are less than 2.5% of the solution times for the L-shaped method. Note also that the differences in the quality of the solutions produced are minimal. In short—methods designed specifically to exploit the structure of the problems can yield considerable computational advantage.

7. A Few Things That Were Omitted

In a tutorial chapter of this size, covering a field as dynamic as stochastic programming necessarily results in a variety of topics that must be omitted. For example, we have omitted discussion of any specific problem applications. For a variety of stochastic programming applications, the reader is referred to the book by Ziemba and Wallace [50]. In the remainder of this section, we include a few references to some of the topics that were omitted.

The recourse models that we focus on throughout this paper involve those for which optimality and feasibility are defined through expected values of relevant functions. An alternate set of models results when constraints must be satisfied with a given probability. These are referred to as models with probabilistic constraints. An excellent resource for the study of such models is the text by Prékopa [36].

Expected values are risk-neutral models, and do not always provide a satisfactory model for decisions under risk. Recent investigations of risk models that are amenable to large-scale computational efforts have lead to the development of *conditional value at risk*, a topic introduced by Rockafellar and Uryasev in [38] and [39]. An alternate measure of risk, which involves semideviations, was introduced in Ogryczak and Ruszczyński [33], and explored further in Ogryczak and Ruszczyński [34]. Modeling risk through dominance constraints is discussed in Dentcheva and Ruszczyński [13].

With regard to model development, we have focused on models involving discrete random variables. As a general rule, solution methods based on statistical approximations are applicable for both continuous and discrete distributions. The scenario tree is a critical element of the multistage model, and some level of care should be exercised when generating it. For an initial discussion of techniques that may be helpful in creating the scenario tree, see Hoyland and Wallace [26]. In cases where the scenario tree is extremely large, methods for aggregation and disaggregation of the tree can be extremely helpful, as discussed in Casey and Sen [8]. Finally, the models presented in this paper have been focused on continuous variables. Models and methods for stochastic integer programs form a very hot area of research. A stochastic integer programming bibliography can be found in van der Vlerk [46].

When problems are solved using partial information, as is the case when statistical approximations are used, questions arise regarding the validity of the solutions obtained. In some cases, these questions can be addressed directly via optimality tests incorporated within an algorithmic technique (as is the case with SD; Higle and Sen [22, 23]). In other cases, there is a need to investigate solution quality more directly. Methods designed to address such questions appear in Mak et al. [30] and Bayraksan and Morton [1]. Additionally, investigations of possible uses of variance reduction techniques with statistically motivated solutions methods are discussed in Infanger [27], Koivu [29], and Higle [18].

Some problem instances suitable for testing algorithms have been developed, but the choices available are very limited, especially within the realm of large-scale instances. To facilitate the sharing of problem data, the SMPS format has been created (Birge et al. [7]). For an excellent discussion of the format, which is related to the MPS format for mathematical programming problems, the reader is referred to Gus Gassmann's helpful website [16].

8. Resources on the Web

The stochastic programming community maintains a website with general information on the discipline [10]. A vast bibliography of stochastic programming references is maintained by Maarten van der Vlerk [46] (University of Groningen, The Netherlands). Maarten is always happy to accept information regarding updates to this website. A separate website is maintained by Werner Römisch (Humboldt University, Germany) for dissemination of recent (i.e., prepublication) results in stochastic programming [45].

Acknowledgments

The author acknowledges the contributions of Stein Wallace for discussions leading to the example in §2.1, and Lei Zhao for his assistance with the computational work reported in this paper. This work was funded by Grant DMS-0400085 from the National Science Foundation.

References

[1] G. Bayraksan and D. P. Morton. Assessing solution quality stochastic programs. `http://www.speps.info`, 2005.

[2] J. F. Benders. Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik* 4:238–252, 1961.

[3] J. R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research* 33(5):989–1007, 1985.

[4] J. R. Birge. The value of the stochastic solution in stochastic linear programs with fixed recourse. *Mathematical Programming* 31(1):25–41, 1982.

[5] J. R. Birge and F. V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research* 34:384–392, 1988.

[6] J. R. Birge and R. J.-B. Wets. Designing approximations schemes for stochastic optimization problems, in particular, for stochastic programs with recourse. *Mathematical Programming Study* 27:54–102, 1986.

[7] J. R. Birge, M. A. H. Dempster, H. I. Gassmann, E. A. Gunn, A. J. King, and S. W. Wallace. A standard input format for multiperiod stochastic linear programs. *COAL Newsletter* 17:1–19, 1987.

[8] M. Casey and S. Sen. The scenario generation algorithm for multi-stage stochastic linear programming. *Mathematics of Operations Research*, 2005. Forthcoming.

[9] A. Charnes and W. W. Cooper. Chance-constrained programming. *Management Science*, 6(1):73–79, 1959.

[10] Stochastic programming community website. `http://www.stoprog.org`.

[11] G. B. Dantzig. Linear programming under uncertainty. *Management Science* 1(3–4):197–206, 1955. Republished in the 50th anniversary issue of *Management Science* 50(12):1764–1769, 2004.

[12] M. A. H. Dempster. On stochastic programming. II. dynamic problems under risk. *Stochastics* 25(1):15–42, 1988.

[13] D. Dentcheva and A. Ruszczyński. Optimization with stochastic dominance constraints. *SIAM Journal on Optimization* 14(2):548–566, 2003.

[14] C. Edirisinghe and W. T. Ziemba. Bounds for two-stage stochastic programs with fixed recourse. *Mathematics of Operations Research* 19:292–313, 1994.

[15] K. Frauendorfer. Solving SLP recourse problems with aribtrary multivariate distributions. *Mathematics of Operations Research* 13:377–394, 1988.

[16] H. I. Gassmann. The SMPS format for stochastic linear programs. `http://www.mgmt.dal.ca/sba/profs/hgassmann/SMPS2.htm#RefBirge_etal` (last updated Feb. 2005).

[17] H. I. Gassmann. MSLiP: A computer code for the multistage stochastic linear programming problem. *Mathematical Programming* 47:407–423, 1990.

[18] J. L. Higle. Variance reduction and objective function evaluation in stochastic linear programs. *INFORMS Journal on Computing* 10(2):236–247, 1998.

[19] J. L. Higle, B. Rayco, and S. Sen. Stochastic scenario decomposition for multi-stage stochastic programs. `http://www.sie.arizona.edu/faculty/higle`, 2004.

[20] J. L. Higle and S. Sen. Finite master programs in regularized stochastic decomposition. *Mathematical Programming* 67:143–168, 1994.

[21] J. L. Higle and S. Sen. Stochastic decomposition: An algorithm for two stage linear programs with recourse. *Mathematics of Operations Research* 16(3):650–669, 1991.

[22] J. L. Higle and S. Sen. *Stochastic Decomposition: A Statistical Method for Large Scale Stochastic Linear Programming.* Kluwer Academic Publishers, Boston, MA, 1996.

[23] J. L. Higle and S. Sen. Statistical approximations for stochastic linear programs. *Annals of Operations Research* 85:173–192, 1999.

[24] J. L. Higle and S. W. Wallace. Sensitivity analysis and uncertainty in linear programming. *Interfaces* 33(4):53–60, 2003.

[25] J. L. Higle and L. Zhao. Adaptive and nonadaptive samples in solving stochastic linear programs: A computational approach. 2004. `http://www.sie.arizona.edu/faculty/higle.`

[26] K. Hoyland and S. W. Wallace. Generating scenario trees for multistage decision problems. *Management Science* 47(2):295–307, 2001.

[27] G. Infanger. Monte Carlo (importance) sampling within a benders decomposition algorithm for stochastic linear programs. *Annals of Operations Research* 39:69–95, 1992.

[28] A. J. Kleywegt, A. Shapiro, and T. Homem-de-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization* 12(2):479–502, 2001.

[29] M. Koivu. Variance reduction in sample approximation of stochastic programs. *Mathematical Programming* 103(3):463–485, 2005.

[30] W. K. Mak, D. P. Morton, and R. K. Wood. Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters* 24:47–56, 1999.

[31] J. M. Mulvey and A. Ruszczyński. A diagonal quadratic approximation method for large-scale linear programs. *Operations Research Letters* 12:205–215.

[32] J. M. Mulvey and A. Ruszczyński. A new scenario decomposition method for large-scale stochastic optimization. *Operations Research* 43(3):477–490, 1995.

[33] W. Ogryczak and A. Ruszczyński. From stochastic dominance to mean-risk models: Semideviations as risk measures. *European Journal of Operational Research* 116, 1999.

[34] W. Ogryczak and A. Ruszczyński. On consistency of stochastic dominance and mean-semideviation models. *Mathematical Programming* 89, 2001.

[35] B.-R. Fu, S. M. Robinson, E. L. Plambeck, and R. Suri. Sample-path optimization of convex stochastic performance functions. *Math. Programming B* 75:137–176, 1996.

[36] A. Prékopa. *Stochastic Programming.* Kluwer Academic Publishers Group, Boston, MA, 1995.

[37] S. M. Robinson. Analysis of sample-path optimization. *Mathematics of Operations Research* 21:513–528, 1996.

[38] R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk* 2(21), 2000.

[39] R. T. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, 26(7):1443–1471, 2002.

[40] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research* 16(1):119–147, 1991.

[41] C. H. Rosa and A. Ruszczyński. On augmented Lagrangian decomposition methods for multistage stochastic programs. *Annals of Operations Research* 64:289–309.

[42] A. Ruszczyński. A regularized decomposition method for minimizing a sum of polyhedral functions. *Mathematical Programming* 35:309–333, 1986.

[43] S. Sen, R. D. Doverspike, and S. Cosares. Network planning with random demand. *Telecommunications Systems* 3:11–30, 1994.

[44] A. Shapiro. Stochastic programming by Monte Carlo methods. `http://www.isye.gatech.edu/ashapiro/publications.html`, 2000.

[45] Stochastic programming e-print series. `http://www.speps.info`.

[46] M. van der Vlerk. Stochastic programming bibliography. `http://mally.eco.rug.nl/index.html?spbib.html`.

[47] R. van Slyke and R. J.-B. Wets. L-shaped programs with applications to control and stochastic programming. *SIAM J. on Applied Mathematics* 17:638–663, 1969.

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

[49] W. L. Winston and M. Venkataramanan. *Introduction to Mathematical Programming: Applications and Algorithms*. Duxbury Press, Belmont, CA, 2002.

[50] W. T. Ziemba and S. W. Wallace, eds. *Application of Stochastic Programming*. SIAM Series on Optimization. SIAM, Philadelphia, PA, 2005.