STATE-OF-THE-ART SURVEY

Stochastic Programming Computation and Applications:

JOHN R. BIRGE / Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Michigan 48109-2117, Email: jrbirge@umich.edu
(Received: August 1995; revised July 1996 and January 1997; accepted: February 1997)

Although decisions frequently have uncertain consequences, optimal-decision models often replace those uncertainties with averages or best estimates. Limited computational capability may have motivated this practice in the past. Recent computational advances have, however, greatly expanded the range of optimal-decision models with explicit consideration of uncertainties. This article describes the basic methodology for these stochastic programming models, recent developments in computation, and several practical applications.

Decision making inherently involves consideration of uncertain outcomes. Whereas computational advances in mathematical programming have aided decision making in many areas, their greatest impact may lie in enhancing decision making under uncertainty through stochastic programming, the optimization branch explicitly concerned with models containing random parameters.

In this article, we explore recent advances in computational capabilities for stochastic programs and the structure of problems that enables these procedures. We also provide some insight into the range of possible applications of the methods through a set of examples from actual practice.

The need for including uncertainty in complex decision models arose early in the history of mathematical programming. The first model forms, involving an action followed by observation and reaction (or recourse), appear in Beale and Dantzig. An alternative model (called chance or probabilistically constrained programming), was also developed quite early, principally by Charnes and Cooper. Both forms have their origin in statistical decision theory (see, e.g., Wald) but, in contrast to decision theory, stochastic programming has emphasized methods of solution and analytical solution properties over procedures for constructing objectives and updating probabilities.

The size of a stochastic program grows proportionally to the number of possible realizations of uncertain parameters. This number in turn generally increases exponentially in the number of time periods and parameters (as in the " curse of dimensionality"). To allow modeling of additional uncertainties, many of the algorithmic developments in the area have concentrated on methods to exploit stochastic program structure. For example, Dantzig and Madansky introduced Dantzig-Wolfe decomposition as a possible method, whereas Van Slyke and Wets developed a form of Benders decomposition for stochastic programs. In general, these methods focus on linear models which are also the focus in this article.

Beyond algorithms, much of stochastic programming research has considered various theoretical properties. Important convexity, continuity, and stability properties appear, for example, in Wets for recourse models. For chance-constrained models, Prekopa derives key convexity conditions. Wets provides a general overview of these results. In this article, we assume conditions for convexity where needed while emphasizing actual methods of solution.

The range of stochastic programming applications is as broad as any in optimization. Early applications include an airline fleet-assignment model by Ferguson and Dantzig. In that model, planes are assigned to routes to maximize expected revenue given a distribution for passenger demand on each route. This application is particularly noteworthy because the authors exploit the generalized network structure of their model with perhaps the first generalized network method. Their solution also yields a net gain of $142,000 or 9% over a solution that ignores uncertainty.

We give a brief overview of stochastic programs and the simplest formulations in Sections 1 and 2. We then describe various solution techniques and their computational implementations in Sections 3, 4, and 5. We consider a sample of applications in Section 6. Section 7 presents ideas for future directions in computation and modeling.

Our emphasis in this article is on computational methods with results in practically sized, large-scale problems. We review methods that achieve improved solutions for stochastic models over simplified deterministic models. To keep the scope of this article manageable, we emphasize computation for a given model over the other fundamental tasks of composing a model that accurately reflects reality and of verifying the model's result with practical compari-
sons. We also mainly discuss linear models over nonlinear models to reflect the majority of current applications.

1. Formulating Stochastic Programs

1.1 The News Vendor Problem

The simplest form of a stochastic program may be the news vendor (also known as the newsboy) problem. We use this familiar example to illustrate some of the fundamental characteristics of a stochastic program and to motivate broader uses.

In the news vendor model, the vendor must determine how many papers (x) to buy now at cost c without knowing the demand represented by a random variable, ξ, with distribution function, F, and a given selling price, p. We assume no salvage value, so that any papers bought in excess of demand are simply discarded with no return.

Assuming that the vendor observes the same distribution for demand every day, a reasonable objective is to maximize expected daily profit

$$\max_{x \geq 0} \phi(x) = -cx + \int_0^x p \xi \, dF(\xi) + \int_x^\infty px \, dF(\xi),$$

(1)

which has the basic form of the stochastic linear program with recourse: action x, followed by observation (ξ), and recourse (sales at price p that depend on the random variable ξ).

The formulation in (1) has the added advantage of an analytical solution. In this case, assuming F is continuous with density, f, we can differentiate to find x^* optimal in (1) such that

$$-c + px^* f(x^*) + \int_{x^*}^\infty p \, dF(\xi) - px^* f(x^*) = 0,$$

or, the well-known result,

$$F(x^*) = \frac{p - c}{p}.$$  

(2)

The formula in (2) illustrates the problem in simply substituting expected values for random quantities, as is often done in practice and widely thought to be of little consequence. This substitution in (1) would correspond to having a distribution F that places all demand at $\hat{\xi}$. The problem becomes:

$$\max_{x \geq 0} \phi(x) = -cx + p(x - \hat{\xi}) + p\hat{\xi}.$$  

(3)

The obvious solution of (3) is $x^{det} = \hat{\xi} = \int \xi \, dF(\xi)$, the mean demand. This value only matches $x^*$ if $F(x^{det}) = (p - c)/p$. For symmetric distributions with mean equal median, that result only occurs if $p = 2c$.

In the more likely case of low margins, the results are quite different. Suppose, for example, that $p = 0.25$ while $c = 0.20$. In this case, $F(x^*) = 0.20$. Suppose a uniform demand on [50, 150] with mean of 100. With these data, $x^* = 70$ while $x^{det} = 100$. The result is that $\phi(x^*) = -20(70) + 25(12) + 25(70)(4/5) = 300$ while $\phi(x^{det}) = -20(100) + 25(75)/2 + 25(100)(1/2) = 1.875$. Thus, the stochastic model yields an increase of $1.125$ or 60% in profits over the deterministic formulation. This difference is called the value of the stochastic solution (VSS) (see Birge1127).

1.2 Value of Information

The VSS is similar to the expected value of perfect information (EVPI) (see, e.g., Raiffa and Schlaifer1108), but the VSS measures a different aspect of uncertainty. For example, EVPI compares the stochastic program solution value with the value of a situation in which all uncertainty is revealed before a decision is made (i.e., with perfect information). In the news vendor case, we have a perfect information solution, $x^{opt} = \xi$ with expected value, $F(x^* = (p - c)\xi(\xi) \int \xi \, dF(\xi) = (p - c)\xi$. For the example given, we would have an expected value with perfect information (EVWPI) of $5.00$ and EVPI = EVWPI - $\phi(x^*) = 2.00$. The difference between EVPI and VSS is illustrated in Figure 1.

In this example, EVPI > VSS but this is not always true. In fact, either value can be zero while the other is positive (see Birge1127). The difference is that the value of resolving uncertainty can be quite different from the cost of ignoring uncertainty in making a decision. Our emphasis is on this latter form where uncertainty cannot be resolved further (in some economic way) but computational shortcuts, such as substituting mean values for random parameters, might still be used.

This simple example illustrates another difficulty for modelers attempting to use mean values alone. If the deterministic model is followed, then one might expect a profit of $(p - c)\xi$ or $5.00$ in this case. The vendor would, of course, be disappointed to find only $1.875$, or $3.125$ (62.5%) less than the value using means alone. In fact, this loss compared to the mean value problem always occurs when mean values for random constraint parameters are substituted in convex stochastic programs (see, e.g., Madansky189 or Birge and Webb288). This result may explain some losses of confidence in optimization models when results do not turn out as well as the model predicts.

The news vendor model considers an expected value of future costs without any modification for risk aversion. Although this approach is reasonable if the news vendor faces the same problem every day, it may not be the case for all situations. In the daily identically distributed demand case, after many days, the news vendor essentially earns the expected value with little variation (by the Laws of Large
Numbers. If, alternatively, the news vendor is actually a large fashion retailer facing a single season of demand that depends on overall market conditions, then the decision should consider the effect of risk.

In the fashion retailer case, the revenue from making the \( x^* \) decision found from (2) has the distribution given in Figure 2 (a uniform distribution over \([-1.5, 3.5]\) and mass of probability 0.8 at 3.5). The retailer may be averse to the risk in Figure 2. In this case, it may be necessary to add some nonlinear function of the outcomes in (1) to derive an optimal decision including risk aversion. In complete market cases, appropriate discounting is sufficient (with the use of option-pricing methodologies as in Birge\cite{b52}) to incorporate risk aversion, but more precise utility function forms are often necessary (see, for example, the discussion in Smith and Naut\cite{b43}).

For our purposes in this article, we concentrate on problems where risk aversion is possible through linearizations (which may approximate nonlinear utilities) but the general approaches apply to nonlinear functions as well. Kallberg and Ziemb\cite{b60} discuss the effects that different utility function forms can yield on decisions. They conclude that objective functions with approximately equal absolute risk-aversion indices (measured by the ratio of second-derivative to first-derivative of the objective function with respect to wealth) yield similar results.

2. Two-Stage Recourse Problems

To generalize the news vendor problem to other models, we preserve the characteristic of action, observation, and reaction. The news vendor problem is an example of a two-stage stochastic linear program with (fixed) recourse (2S-RLP), which is written

\[
\min_{x \in \mathbb{R}^n} c^T x + \mathbb{E}(x)
\]

s.t. \( Ax = b \) \hspace{1cm} (2S-RLP)
\[
x \geq 0,
\]

where \( \mathbb{E}(x) = \int_{\Omega} Q(x, \omega)p(d\omega) \), with \( Q(x, \omega) = \min_{y(\omega) \in \mathbb{R}^m} q(\omega)^T y(\omega) \)
\[
s.t. \quad W y(\omega) = h(\omega) - T(\omega) x, \quad (LRP)
\]
\[
y(\omega) \geq 0,
\]

and \( b \in \mathbb{R}^m, W \in \mathbb{R}^{m \times m}, \) and \( (q(\omega), h(\omega), T_1(\omega), \ldots, T_m(\omega)) \) is a random vector that depends on an underlying outcome \( \omega \in \Omega \) with a probability function \( P \). The recourse is called fixed because \( W \) is deterministic. Although \( W \) could also be random, it is usually assumed fixed to obtain clear definitions of feasible solutions (see, e.g., Kall\cite{b52}).

To fit the news vendor problem into the (2S-SLPR) framework, suppose that the random demand \( \xi \) in (1) depends on an outcome \( \omega = \xi(\omega) \). We can write (1) as

\[
Q^{\text{new}}(x, \xi(\omega)) = \{ \min -py_i | 0 \leq y_1 \leq x, y_1 \leq \xi(\omega) \}
\]

\[
= \{ \min -py_i | y_1 + y_2 = x, y_1 + y_2 = \xi(\omega), y_1, y_2, y_3 \geq 0 \},
\]

so that

\[
q = (-p, 0, 0)^T, \quad W = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}
\]

\[
T = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \quad \text{and} \quad h(\omega) = \begin{pmatrix} 0 \\ \xi(\omega) \end{pmatrix}.
\]

The news vendor model can also be written as a special form of (2S-SLPR) called simple recourse in which \( W = [I - I], \) forcing the \( y \) values by the sign of the components of \( h - Tx \). For this representation, the objective in (2S-SLPR) is changed from \( c \) to \( c - c \). The recourse function,

\[
Q^{\text{new-simple}}(x, \xi) = \{ \min -py_i | y_1 + y_2 = \xi - x, y_1, y_2 \geq 0 \},
\]

in (LRP). In this case, we have brought the potential revenues \( px \) into the first stage while making adjustments in the second stage. With some risk aversion and time value of revenue, this simple formulation would not, however, necessarily be possible.

Although several methods are quite efficient for the general simple recourse problem (see, e.g., Wets\cite{b88} and Nazareth and Wets\cite{b90}), the simple nature of the recourse makes \( \mathbb{E}(x) \) straightforward to evaluate so that the simple recourse model can be solved as a convex (with discrete random variables, also piecewise linear) program with linear constraints. (Note that \( Q(x, \xi) \) is convex in \( x \), yielding \( \mathbb{E}(x) \) convex.)

2.1 Chance-Constrained Problems

An additional common characteristic of stochastic programs is to have a constraint that holds with a certain probability. For example, the news vendor may want to have papers for at least a fraction \( \alpha < 1 \) of the potential customers to maintain good public relations. The addition of these probabilistic constraints or chance constraints, generally of the form,

\[
P(T'x \geq h') \geq \alpha,
\]
creates a probabilistically or chance-constrained stochastic program. These constraints are often simply replaced by a deterministic equivalent form, \( T' x \succeq h' \), or nonlinear system, and added to the deterministic constraints.

A finite set of deterministic equivalent constraints to (6) is possible, for example, when \( T' x \succeq h' \) in (6) is a single constraint on \( x \) with \( T' \) corresponding to a single row. In that case, note that \( P(T' x \succeq h' (\omega)) = F(T' x) \) for \( F \) the distribution function of \( h' (\omega) \). Constraint (6) is then equivalent to \( T' x \succeq h' \) where \( h' = (F)^{-1} (a) \).

Many single chance-constraints can be added individually to a problem to form a set of constraints, \( F(T' x \succeq h' (\omega)) \succeq a_i \), for \( i = 1, \ldots, m_2 \), where \( P(T' x \succeq h' (\omega)) = F_i(T' x) \) for \( F_i \) the distribution function of \( h' (\omega) \). The deterministic equivalent is then

\[
T' x \geq (F)^{-1} (a_i), \quad i = 1, \ldots, m_2.
\]

In general, several constraints may hold jointly so that \( T' \) in (6) has several rows. These chance constraints require additional refinements of a linearization or a nonlinear form to obtain deterministic equivalents. In some cases, a convex equivalent region does not even exist. Prékopa\(^{108}\) discusses such procedures and potential complications extensively.

The choice of a recourse model, a chance-constrained model, or some combination of these models depends on the decision maker's attitude toward risk. As noted above, although (2S-SLPR) appears to imply risk neutrality with its linear objective, we can model many kinds of risk attitudes through appropriate discount factors and constraint modification with nonlinearities. In some cases, the decision makers may have multiple objectives or other utility forms that require nonlinear terms.

### 2.2 Multistage Problems

Many problems involve not just a single observation and reaction but a sequence of these decisions. In this case, with linear objectives, we have the multistage stochastic linear program with recourse (MS-SLPR)\

\[
\begin{align*}
\text{min} \quad & z = c_1 x_1 + E[ c_2(x_1) x_2(\omega_1) + \cdots + c_n(x_{n-1}) x_n(\omega_{n-1}) ] \\
\text{s.t.} \quad & W_i x_i = h_i, \\
& T_i(\omega_1) x_1 + W_2 x_2(\omega_1) = h_2(\omega_1), \\
& \vdots \\
& T_{n-1}(\omega_{n-1}) x_{n-1}(\omega_{n-1}) + W_n x_n(\omega_{n-1}) = h_n(\omega_{n-1}), \\
& x_1 \geq 0, \quad x(\omega') \geq 0, \quad \text{a.s.}; \quad t = 2, \ldots, n; \\
\end{align*}
\]

(\text{MS-SLPR})

where \( c_i \) is a known vector in \( \mathbb{R}^m \), \( h_i \) is a known vector in \( \mathbb{R}^m \), each \( W_i \) is a known \( m_i \times n_i \) matrix (for fixed recourse), and we represent the random observations at time \( t + 1 \) through a random outcome, \( \omega_{t+1} \), and a history through time \( t \) represented as \( \omega' = (\omega_1, \ldots, \omega_t) \). We use the subscript notation for a single period and the superscript notation to denote all previous periods (and an iteration counter in the algorithm discussion below). The complete data process, \( \{\omega = (\omega_1, \ldots, \omega_{T-1})\} \), is defined on \( \Omega \) with associated probability \( \mathbb{P} \). The random observations at \( t + 1 \) are \( N' \)-vectors, \( \xi_{t+1}(\omega') = (\xi_{t+1}(\omega_1), h_{t+1}(\omega'), T_{n-1}(\omega_{t+1}), \ldots, (T_{n-1}(\omega_{t+1}))) \).

This type of formulation becomes somewhat cumbersome because we have implicitly decided the decisions to depend only on past data. The more general formulation includes explicit constraints to maintain this property plus possibly nonlinear functions and integer variable restrictions. In this case, we can formulate a general multistage stochastic program with recourse as

\[
\begin{align*}
\inf_{\omega} \quad & E[ \sum_{t=1}^{n} f_{t+1}(\omega_t, x'(\omega_t), x_{t+1}(\omega_t)) ] \\
\text{s.t.} \quad & x_t(\omega) \in X_t(\omega) \text{ for all } \omega \in \Omega,
\end{align*}
\]

where \( E \) is expectation with respect to all events. We may relax the statement "for all \( \omega' \) to "almost all \( \omega' \)" which might exclude some outcomes with zero probability. The notation \( N' \) denotes all decisions that are nonanticipative, i.e., such that the decision process at time \( t \) must only depend on the data up to time \( t \). An alternative characterization of this nonanticipative property is that \( x'(\omega') = E(x'(|\omega'|)) \) for almost all \( \omega', t = 0, \ldots, \), where \( E \cdot |\omega'| \) is conditional expectation with respect to the history at time \( t \). If we let \( \Pi' x(\omega) = E(z(\omega)|\omega') \), \( t = 0, \ldots, \), then nonanticipativity is equivalent to

\[
(I - \Pi') x_t = 0, \quad t = 0, \ldots
\]

In our previous examples, the constraints in (2S-SLPR) were implicit by replacing \( x(\omega) \) with a single value, \( x \), wherever \( x(\omega) \) might appear. If we use the explicit representation, constraints (9) become

\[
x(\omega) - E(\omega) = 0, \quad \text{for all } \omega \not\in \Omega.
\]

The structure in (8) can often prove more useful than the representation in (MS-SLPR) because constraints (9) become the only constraints linking scenarios. As we shall see below, methods that relax these constraints and operate on the individual scenario problems independently can often achieve significant computation-time savings.

### 2.3 Approximations

The overwhelming difficulty in stochastic programming stems from the evaluation of random functions and their expectations, such as \( E(z) \) in (2S-SLPR) or the indicator function of \( T' x \succeq h' \) in the chance-constrained form (6), to reflect the expectation of all future objective terms given decision \( x_t \) at time \( t \) and a history \( \omega' \). Except in special cases, such as the simple recourse case mentioned above, practical solution requires some form of approximation. In some (perhaps most practical) situations, a distribution may not be known. In these cases, an approximating distribution may be chosen to yield results which can be far superior to a single-best estimate or mean-value substitution for all random parameters.

The majority of approximation methods involve some discretization of an underlying probability function. The discretization may be chosen to provide upper or lower
bounds on values with a known probability measure or known distribution with certain known characteris-
tics (such as mean and variance ranges, see, e.g., Birge and Wets [28] and Kall, Ruszczyński, and Fau-
rendorf [26]). Similar continuous bounding approximations replace the original high-dimensional integral with a combination of low-
dimensional integrals (see Birge and Qi [124], Birge and Wallace, [127], Birge and Wets [129], Wallace [130]). Each of these methods relies heavily on the convexity of the recourse problem.

The other common technique is to use an empirical dis-
bution based on past observations or some random gen-
eration scheme. For random methods, they can either be applied to a large sample approximation as in asymptotic statistical characteristics may apply (see, e.g., Dupačová and Wets [146], King and Rockafellar [171], Römisch and Schütz [111], Shapiro [112]) or implemented with several small sample batches in conjunction with an algorithm that converges asymptotically (see, e.g., Ermoliev [148], Gaiyovorski and Higle and Sen [151] and Pfaffenbichler [152]).

Another approach is to attempt to find decision function that improve an objective function over previous decisions for all members of a set of possible distributions or objective functions (see Martello [161]). This method is then similar to a pareto optimum-seeking method in multiobjective optimization that seeks an improving direction for all objectives. The approach explicitly contains the common circumstance where specific distributions are not available.

Lack of distribution information is a consistent problem in stochastic programming and other areas of operations research. Although the bounding approximation procedures can produce intervals for an optimal objective value, it is often impossible to identify a single "best" solution. Stochastic programs instead give some indication of the relative value of different solutions and provide some set of possible decisions that may exclude decisions considered "optimal" with deterministic modeling.

Stochastic programs often choose solutions to approxima-
tions that provide bounds (perhaps probabilistic) on pos-
sible solution values given the state of information about the uncertain parameters. It may be impossible to improve the information so that the bounds may converge. The solutions and the values of these errors are, nevertheless, relevant in providing decision makers with alternative actions. This article concentrates on methods to obtain those solutions and values.

Although choosing approximations and models is both difficult and important, finding solutions for models based on discrete approximations is complex because of problem size. Solutions to the approximations are also critical for identifying potential alternatives, rewards, and costs. Fortunately, the structure of stochastic programs allows significant efficiencies over straightforward procedures developed for standard deterministic problems. This article focuses on exploitation of that structure in current computational schemes.

3. Direct Methods using Stochastic Programming Structure

Our goal in this section and the next is to describe the fundamental methods and current experience with their computational implementation. The key factor we stress is how the various methods take advantage of problem structure. We do not consider broad general techniques except for comparisons.

We first consider solution methods that use standard optimization procedures without large-scale strategies such as decomposition and relaxation. We still, however, wish to concentrate on methods that use specific stochastic program structure. The computational results in this area are generally for linear models so we also focus on the basic extreme-point (simplex) method and interior-point methods.

3.1 Extreme-Point Methods

For extreme-point methods, we first consider the two-stage model, (2S-SLP), with a discrete distribution supported on \( \mathcal{E} = \{e_1, \ldots, e_n\} \), with associated probabilities, \( p_1, \ldots, p_n \). With these assumptions, \( 2S-SLP \) becomes

\[
\min_{x,y} c^T x + p_1 q_1 y_1 + \cdots + p_n q_n y_n
\]

\[\text{s.t. } A x = b,\]

\[T_i x + W y_i = h_i, \quad i = 1, \ldots, k,\]

\[x, y_1, \ldots, y_n \geq 0.\]

The structured direct methods take advantage of the non-
zero pattern of (11), called dual block angular. Extreme point methods consider a basis from the matrix in (11) and attempt a specific factorization (see Kall [163] and Strang [111]). This full factorization employs separate submatrix factorizations corresponding to diagonal blocks (with \( W \) components) and a single (at most \( n_1 \times n_2 \)) additional working basis.

Although some advantages over standard simplex meth-
ods were reported in early implementations (see Strang [111]), few implementations specifically exist for this structure because of the existing sparse factorization schemes in general-purpose linear programming codes and the relatively limited improvement potential for extreme-
point methods over commercial codes. It has also been shown (see Birge [169]) that a variant of the factorization method can trace the same iterates as decomposition meth-
ods and, hence, that decomposition may afford a broader range of possibilities.

Extensions of basis-factorization techniques are also possible in multiple stages (see Birge [121]), but here the additional overhead for maintaining the basis structure appears to outweigh any computational savings. The results may, however, be quite different for interior-point methods. The key may be the degree to which each method can be performed in parallel.
3.2 Interior-Point Methods

Factorization schemes also offer substantial promise for interior-point methods, where there is much speculation that the solution effort grows linearly in the size of the problem. Discussion of computational results appears, for example, in Lustig, Marsten, and Shanno.\(^{(83,84)}\) In interior-point methods, problems of form (11) are again solved. At iteration \(\nu\) of these algorithms with a current iterate, \((x^\nu, y_1^\nu, \ldots, y_k^\nu),\) if we denote the constraint matrix in (11) by \(A,\) then the effort in most interior point methods is dominated by computations with a matrix, \(M = AD^2A^T,\) where \(D\) is a diagonal matrix. Letting \(D_0 = \text{diag}(x^\nu), D_k = \text{diag}(y_k^\nu), k = 1, \ldots, K,\) we would have

\[
M = \begin{pmatrix}
AD_0^2A^T & AD_0^2T_i^T & \cdots & AD_0^2T_k^T \\
T_iD_i^2A^T & T_iD_i^2T_i^T + WD_i^2W_i & \cdots & T_iD_i^2T_k^T \\
\vdots & \vdots & \ddots & \vdots \\
T_kD_k^2A^T & T_kD_k^2T_i^T & \cdots & T_kD_k^2T_k^T + WD_k^2W_k
\end{pmatrix}
\]

(12)

which is much denser than the original constraint matrix in (11). In this case, a straightforward implementation of an interior point method that solves systems with \(M\) is quite inefficient.

Note, however, that \(M\) in (12) has a great deal of structure that can be exploited in any solution scheme. This is the object of the factorization scheme given by Birge and Qi\(^{(83)}\) based on the following proposition.

Theorem 1. Let \(S_0 = I_2 \in \mathbb{R}^{m \times m_1}, S_i = W_iD_i^2W_i^T, i = 1, \ldots, K, S = \text{diag}(S_0, \ldots, S_K).\) Then \(S^{-1} = \text{diag}(S_0^{-1}, S_1^{-1}, \ldots, S_K^{-1}).\) Let \(I_1\) and \(I_2\) be identity matrices of dimensions \(n_1\) and \(m_1,\) respectively. Let

\[
G_1 = (D_0)^{-2} + A^TS_0^{-1}A + \sum_{i=1}^K T_i^TS^{-1}T_i, \quad G_2 = -AG_1^{-1}A^T,
\]

(13)

\[
U = \begin{pmatrix}
A & I_2 \\
T_1 & 0 \\
\vdots & \vdots \\
T_K & 0
\end{pmatrix}, \quad V = \begin{pmatrix}
A & -I_2 \\
T_1 & 0 \\
\vdots & \vdots \\
T_K & 0
\end{pmatrix}
\]

If \(A\) and \(W\) have full row rank, then \(G_2\) and \(M\) are invertible, and

\[
M^{-1} = S^{-1} - S^{-1}U \begin{pmatrix}
I_1 & G_1^{-1}A^T \\
0 & I_2
\end{pmatrix} \begin{pmatrix}
I_1 & -A \\
0 & I_2
\end{pmatrix} \begin{pmatrix}
G_1^{-1} & 0 \\
A & 0
\end{pmatrix} V^TS^{-1}.
\]

(14)

Proof. Follows Birge and Qi\(^{(83)}\) See also Birge and Holmes\(^{(21)}\) 

Following the assumptions and using the complexity result in Karmarkar\(^{(69)}\) the number of arithmetic operations in a single iteration using this factorization can be reduced from \(O(n_1 + Kn_2)^3\) using a general matrix technique, Birge and Qi, show that the effort is, in fact, dominated by \(O(K(n_1^2 + n_2^2) + n_2^2))\). For \(n = n_1 + Kn_2\) with \(L\) representing the size of the data and \(K \sim n_1 \sim n_2,\) the full arithmetic complexity is \(O(n^{2.5}L),\) as compared to Karmarkar's general result of \(O(n^{3.5}L).\) Thus, the factorization in (14) provides a significant order of magnitude improvement over a general solution scheme in the number of realizations \(K\) approaches the number of variables in the first and second stages. The same improvement applies to lower-order primal–dual interior-point methods as well.

Considering the relative increase in effort for these methods as a function of \(K,\) the number of realizations, we can obtain a method that is \(O(K^{1.5})\) with this technique. A more recent result by Ariyawansa\(^{(22)}\) gives a method that is actually \(O(K)\) which is consistent with computational results by Czyzyk, Fourer, and Mehrotra.\(^{(20)}\) With parallel processing, the elapsed computation time could in fact be relatively independent of \(K.\)

In practice, we would not compute \(M^{-1}\) explicitly. The work in (14) is dominated by the effort to solve systems of the form

\[
M\nu = u,
\]

(15)

using

\[
\nu = p - r,
\]

(16)

where

\[
Sp = u, \quad Gq = V^Tp, \quad Sr = Uq,
\]

(17)

where \(G\) represents the inverse of the matrix premultiplying \(V^T\) in (14). The systems in (17) require solving systems with \(S_p\) computation of \(G_1\) and \(G_2\) and solving systems with \(S_0\) and \(G_2.\) In practice, we find a Cholesky factorization of each \(S_p\) use them to find \(G_1\) and \(G_2\) and then find Cholesky factorizations of \(G_1\) and \(G_2.\)
3.3 Alternative Methods

Before we describe results using the factorization in (14), we consider other options for interior point methods. These possibilities are:

- Schur complement updates;
- Column splitting;
- Solution of the dual;
- Direct solution of the symmetric indefinite augmented system.

The Schur complement approach is used in many interior point method implementations (see, e.g., Choi, Monma, and Shanno\textsuperscript{(193)}. The basic idea is to write \( M \) as the sum of a matrix with sparse columns, \( A_dD_d^{-1}A_d^T \), and a matrix with dense columns, \( A_dD_d^{-1}A_d^T \). Using a Cholesky factorization of the sparse matrix, \( LL^T = A_dD_d^{-1}A_d^T \), the method involves solving \( Mu = v \) by

\[
\begin{pmatrix}
LL^T & -A_dD_d^{-1} \\
D_d^{-1} & I
\end{pmatrix}
\begin{pmatrix}
v \\
w
\end{pmatrix} =
\begin{pmatrix}
u \\
0
\end{pmatrix},
\]

(18)

which involves solving \( [I + D_d^{-1}A_d^T(LL^T)^{-1}A_dD_d]w = -(D_d^{-1}A_d^T(LL^T)^{-1}b) \) and \( LL^Tv = b + A_dD_d^Tw \), where \( I + D_d^{-1}A_d^T(LL^T)^{-1}A_dD_d \) is a Schur complement. The Schur complement is thus quite similar to the factorization method given in Section 3.3. If every column of \( x \) is considered a dense column, then the remaining matrix is quite sparse but rank-deficient. The factorization in (14) is a method for maintaining an invertible matrix when \( A_d^2A_d^T \) is singular. It can thus be viewed as an extension of the Schur complement to the stochastic linear program. Because of the possible rank deficiency and the size of the Schur complement, the straightforward Schur complement approach in (18) is quick but leads to numerical instabilities as Carpenter, Lustig, and Mulvey\textsuperscript{(193)} report.

3.4 Column Splitting

Carpenter, Lustig, and Mulvey\textsuperscript{(193)} also propose the column-splitting technique. The basic idea is to rewrite problem (11) with explicit constraints on nonanticipativity. The formulation then becomes

\[
\begin{aligned}
\min_{k} & \sum_{i=1}^{k} p_i(x_i + q_iy_i) \\
\text{s.t.} & \quad A_kx_k = b, \\
& \quad T_kx_k + Wy_k = h_k, \quad k = 1, \ldots, K; \\
& \quad x_k - x_{k+1} = 0, \quad k = 1, \ldots, K - 1; \\
& \quad x_k \geq 0, \quad y_k \geq 0, \quad k = 1, \ldots, K.
\end{aligned}
\]

(19) (20) (21) (22) (23)

The difference now is that the constraints in (20) and (21) separate into subproblems \( k = 1, \ldots, K \). Constraints (22) link the problems together. Alternating constraints, (20), (21), and (22) for each \( k \) in sequence, the full constraint matrix has the form

\[
\bar{A} = \begin{bmatrix}
A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
T_1 & W & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
I & 0 & -I & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & A & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & T_2 & W & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & I & 0 & -I & 0 & 0 & 0 & 0 \\
\vdots & \vdots & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\
0 & 0 & 0 & 0 & I & 0 & -I & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & A & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & T_k & W & 0
\end{bmatrix}
\]

(24)

If we form \( \bar{A}\bar{A}^T \), then we obtain
\[
\mathbf{\Lambda\Lambda}^T = \begin{pmatrix}
\mathbf{A}\mathbf{A}^T & \mathbf{A}\mathbf{T}_1^T & \mathbf{A} & 0 & 0 & 0 & 0 & 0 \\
\mathbf{T}_1\mathbf{A}^T & \mathbf{T}_1\mathbf{T}_1^T + \mathbf{W}\mathbf{W}^T & \mathbf{T}_1 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{A}^T & \mathbf{T}_1^T & 2\mathbf{I} & -\mathbf{A}^T & 0 & 0 & 0 & 0 \\
0 & 0 & -\mathbf{A} & \mathbf{A}\mathbf{A}^T & \mathbf{A}\mathbf{T}_2^T & \mathbf{A} & 0 & 0 \\
0 & 0 & \mathbf{T}_2\mathbf{A}^T & \mathbf{T}_2\mathbf{T}_2^T + \mathbf{W}\mathbf{W}^T & \mathbf{T}_2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{T}_2^T & 2\mathbf{I} & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \mathbf{A}^T & \mathbf{T}_{K-1}^T & 2\mathbf{I} & -\mathbf{A}^T & 0 \\
0 & 0 & 0 & 0 & 0 & -\mathbf{A} & \mathbf{A}\mathbf{A}^T & \mathbf{A}\mathbf{T}_K^T \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{T}_K\mathbf{A}^T & \mathbf{T}_K\mathbf{T}_K^T + \mathbf{W}\mathbf{W}^T
\end{pmatrix},
\]

which is clearly much sparser than the original matrix in (12). It is, however, larger than \(M\) in (12) leading to a tradeoff of increased size for reduced density.

The third additional approach is to form the dual of (11) and to solve that problem using a basic interior-point method. The dual problem is

\[
\text{max } \mathbf{b}^T \mathbf{\rho} + \sum_{k=1}^{K} p_k \pi_k^T \mathbf{h}_k
\]

s.t. \(\mathbf{A}\mathbf{\rho} + \sum_{k=1}^{K} p_k \mathbf{T}_k \pi_k \leq \mathbf{c}, \quad k = 1, \ldots, K,\)

\[
\mathbf{W}^T \pi_k \leq \mathbf{q}_k, \quad k = 1, \ldots, K,
\]

where the variables are not restricted in sign. For this problem, we can achieve a standard nonnegative variable, equality-constraint form by splitting the variables \(\pi_k\) and \(\mathbf{\rho}\) into differences of nonnegative variables and by adding slack variables to constraints (27) and (28). (Note that the dual problem no longer may have a bounded set of optima causing some theoretical difficulties for convergence results. In practice, bounds are placed on the variables to guarantee convergence.)

With the changes above, the constraint matrix for (27) and (28) becomes

\[
\mathbf{A}' = \begin{pmatrix}
\mathbf{A}^T & -\mathbf{A}^T & \mathbf{T}_1^T & -\mathbf{T}_1^T & 0 & \mathbf{T}_1^T & -\mathbf{T}_2^T & \cdots & \mathbf{T}_1^T & -\mathbf{T}_K^T & 0 & \mathbf{I} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{W}^T & -\mathbf{W}^T & \mathbf{I} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mathbf{W}^T & \mathbf{W}^T & \mathbf{I}
\end{pmatrix}
\]

The matrix in (29) may again be much larger than the original, but the gain comes in considering \(\mathbf{A}'\mathbf{A}'^T\) which is now

\[
\begin{pmatrix}
2(\mathbf{A}'\mathbf{A} + \sum_{k=1}^{K} \mathbf{T}_k\mathbf{T}_k^T) + \mathbf{I} & 2\mathbf{T}_1\mathbf{W} & 2\mathbf{T}_2\mathbf{W} & \cdots & 2\mathbf{T}_K\mathbf{W} \\
2\mathbf{W}'\mathbf{T}_1 & 2\mathbf{W}'\mathbf{W} + \mathbf{I} & 0 & 0 & 0 \\
2\mathbf{W}'\mathbf{T}_2 & 0 & 2\mathbf{W}'\mathbf{W} + \mathbf{I} & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
2\mathbf{W}'\mathbf{T}_K & 0 & 0 & 0 & 2\mathbf{W}'\mathbf{W}
\end{pmatrix},
\]

with inherent sparsity. In fact, it is not necessary to take the dual in order to use this alternative factorization form. As shown in Biege, Freund, and Vanderbei, the Sherman–Morrison–Woodbury formula can be applied to the original problem in (11) so that computations with the structure of \(\mathbf{\Lambda}\mathbf{\Lambda}^T\) are replaced by computations with \(\mathbf{\Lambda}^T\mathbf{\Lambda}\).

### 3.5 Results

In Carpenter, Lustig, and Mulvey, increasing numbers of scenarios were included into a network formulation of a financial decision problem. They used the variable-splitting option with an additional observation that many of the \(\mathbf{T}_k\) columns were zero and that the corresponding variables need not be split. By splitting only those variables with nonzero \(\mathbf{T}_k\) entries, they developed a partial splitting model that proved most effective. Their results with this partial splitting model show an approximately linear increase in speed-up for using split variables compared to the original primal form with the interior point code OB1 (see Lustig, Marsten, and Shanno). For 18 scenarios, the speed-up was 1.48 whereas for 72 scenarios of the same problem, the partial splitting speed-up was 11.82.

Carpenter, Lustig, and Mulvey also used the Schur
complement approach and achieved additional speeds-up over partial splitting, although the additional speed-up was about the same (essentially a factor of two) for all numbers of scenarios. They did, however, report an order-of-magnitude increase in a measure of primal infeasibility because of numerical instability with the basic Schur complement approach.

Birge and Holmes\cite{Holmes9} implemented the dual approach and the primal factorization of Birge and Qi on a test set of stochastic programs based on the multistage problems compiled by Ho and Loutet.\cite{Loutet} These problems represent practical problem structures in economic development, agriculture, energy modeling, structural design, traffic assignment, and production scheduling. Solutions with up to 64 scenarios were included.

The implementation in \cite{BirgeHolmes} used SPARSPAK\cite{Karypis} to solve the systems of equations in each interior-point form. The results indicated that the dual form appeared to be the most efficient technique with significant speed-up (generally between 10 and 100) over the primal form. However, in some cases, this matrix became too large for the operating system. In these cases, the factorization approach appeared relatively more efficient. The largest advantage of the factorization may be, however, that the operations in (17) for each $S_j$ can be performed in parallel.

Yang and Zenios\cite{YangZenios} have implemented the factorization in (14) in parallel on a Connection Machine CM-5 with up to 64 processors. Because of memory efficiencies in the parallel version, they actually achieve superlinear speeds-up in parallel compared to serial times. They compared their method with Vanderbei’s LOQO interior-point solver (see Vanderbei\cite{Vanderbei}) which enables alternative factorizations of a symmetric indefinite system (see Vanderbei and Carpenter\cite{VanderbeiCarpenter}). Yang and Zenios report that serial times using the factorization in (14) do not generally overtake LOQO except for very large problems, but that their parallel processing achieves order-of-magnitude speeds-up over LOQO on large versions of the test problems from \cite{BirgeHolmes}. They also report solutions of problems with up to eighteen million variables and almost three million constraints.

### 3.8 Symmetric Indefinite System

The symmetric indefinite system considered in LOQO corresponds to the system in primal-dual interior-point methods that find primal step directions, \( \delta \bar{x} = (\delta x, \delta y) \), and dual step directions \( \delta \bar{\pi} = (\delta \pi, \delta \rho) \), to satisfy

\[
\begin{pmatrix}
-I & DA \\
AD & 0
\end{pmatrix}
\begin{pmatrix}
\delta \bar{x} \\
\delta \bar{\pi}
\end{pmatrix} =
\begin{pmatrix}
v \\
w
\end{pmatrix},
\]

where \( v \) and \( w \) depend on the iterate. The primal method can be viewed as eliminating for \( \delta \bar{x} \) first in (31) whereas the dual method effectively solves for \( \delta \bar{\pi} \) first. Other methods can, however, take direct advantage of the structure in (31) without resorting to either of these extreme cases.

Czyzyk, Fourer, and Mehrotra\cite{Czyzyk9} report on solutions using the symmetric-indefinite, augmented system in (31) and a pivot ordering that yields a factorization similar to (14). Using large versions of the test problems in Birge and Holmes\cite{Holmes9} they report improvement over split-variable formulations on all problems except one and speeds-up ranging up to two orders of magnitude over split-variable formulations. They also indicate serial computation times that increase linearly in $K$.

For multistage problems, Berger et al.\cite{Berger} provide a similar factorization form, called \textit{tree dissection}, for the augmented system and a possibly quadratic objective. The general idea is to force pivots along links joining scenarios at different points in time. Their implementation of this strategy achieves several orders-of-magnitude improvements over LOQO's default setting, and as much as an order-of-magnitude improvement over LOQO when LOQO is directed to use a "dual first" strategy (and the subsequent $A^T A$ form).

They were also able to solve significantly larger problems than other methods with as many as one million constraints and almost two million variables.

From these developments, it appears that special-purpose factorizations offer particular advantages for interior-point methods for stochastic programs. The availability of parallel processing and capabilities for performing necessary routines in parallel may determine which form that factorization should take. Virtually all of the results report improvements for large problems over direct simplex-based techniques. Decomposition methods, however, have often achieved even greater efficiencies. We discuss these methods in the next section.

### 4. Decomposition Approaches

The general idea behind decomposition methods for (25-SLPR) and (MS-SLPR) is that a smaller problem in $x$ or $x_i$ alone can be solved with some approximation of the recourse function, $\beta(x)$. This idea started with a method called \textit{L-Shaped} by Van Slyke and Wets\cite{VANSLYKE} that is a form of Benders\cite{Benders} decomposition. From that, various extensions have been suggested. We discuss these in the next section.

#### 4.1 The L-Shaped Method

Consider the general formulation in (25-SLPR). The basic idea of the L-shaped method is to approximate the nonlinear term in the objective function of this problem. A general principle behind this approach is that, because the nonlinear objective function term (the recourse function) involves a solution of all second-stage recourse linear programs, we would like to avoid numerous function evaluations for it. We therefore use that term to build a master problem in $x$ but we only evaluate the recourse function exactly as a subproblem.

To make this approach possible, we assume again that the random vector $\xi$ has finite support. Consider the formulation in (11). The dual-block-angular structure leads to the name, L-shaped Method, for the decomposition method we describe here. The method is essentially a Dantzig–Wolfe decomposition (inner linearization)\cite{DantzigWolfe} of the dual or a Benders decomposition (outer linearization)\cite{Benders} of the primal. This method, due to Van Slyke and Wets\cite{VANSLYKE} for stochastic programming, also considers feasibility questions of particular relevance for stochastic programs. In the following, we sup-
pose the first-stage decision, x, is bounded and that (11) has a finite optimal value. Procedures to remove such assumptions are straightforward.

**L-shaped Method**

**Step 0.** Set s = t = ν = 0.

**Step 1.** Set ν = ν + 1. Solve the linear program (32)-(34).

\[
\begin{align*}
\min z &= c^T x + \theta \\
\text{s.t.} \quad Ax &= b, \\
D_j x &\geq d_j, \quad \ell = 1, \ldots, r, \\
E_j x + \theta &\geq e_j, \quad \ell = 1, \ldots, s; \\
x &\geq 0, \quad \theta \in \mathbb{R}.
\end{align*}
\]

Let \((x^*, \theta^*)\) be an optimal solution. If no constraint (34) is present, \(\theta^*\) is set equal to \(-\infty\) and not considered in the computation of \(x^*\).

**Step 2.** For \(k = 1, \ldots, K\) solve the linear program

\[
\begin{align*}
\min w &= e^T v^+ + e^T v^- \\
\text{s.t.} \quad W y + Iv^+ - Iv^- &= h_k - T_k x^*, \\
y &\geq 0, \quad v^+ \geq 0, \quad v^- \geq 0,
\end{align*}
\]

where \(e^T = (1, 1, \ldots, 1)\), until, for some \(k\), the optimal value \(w^* \geq 0\). In this case, let \(\sigma^*\) be the associated dual multipliers and define

\[D_{+1} = \sigma^* T_k\]

and

\[d_{+1} = \sigma^* h_k\]

to generate a constraint (called a feasibility cut) of type (33). Set \(r = r + 1\), add to the constraint set (33), and return to Step 1. If, for all \(k, w^* > 0\), go to Step 3.

**Step 3.** For \(k = 1, \ldots, K\) solve the linear program

\[
\begin{align*}
\min w &= q_k^T y \\
\text{s.t.} \quad W y &= h_k - T_k x^*, \\
y &\geq 0.
\end{align*}
\]

Let \(q_k^T\) be the simplex multipliers associated with the optimal solution of Problem k of type (39). Define

\[E_{+1} = \sum_{i=1}^{K} p_i q_i^T T_k\]

and

\[e_{+1} = \sum_{i=1}^{K} p_i q_i^T h_k.\]

Let \(w^* = e_{+1} - E_{+1} x^*\). If \(\theta^* \geq w^*\), stop. The solution \(x^*\) is optimal. Otherwise, set \(s = s + 1\), add to the constraint set (34), and return to Step 1.

The method solves an approximation of (2S-SLPR) using an outer linearization of \(\Theta\). Two types of constraints are sequentially added: (i) feasibility cuts (33) determining \(x^2(\lambda) < +\infty\) and (ii) optimality cuts (34) which are linear supports of \(\lambda\) on its domain of finiteness.

This method has also been generalized to multiple stages (see Birge\(^{13}\)) as a nested decomposition method. The basic idea of the nested L-Shaped or Bender decomposition method is to place cuts on the period \(t\) value function, \(z_{t+1}(x_t)\), and to add other cuts to achieve an \(x_t\) that has a feasible completion in the future. The cuts represent successive linear approximations of \(z_{t+1}\). Because of the polyhedral structure of \(z_{t+1}\), this process converges to an optimal solution in a finite number of steps.

In general, for every stage \(t = 1, \ldots, H - 1\), we have \(k = 1, \ldots, K\), possible outcomes or scenarios that are joined to an ancestor scenario, \(a(k)\) at \(t = 1\), and a set of descendant scenarios at \(t + 1\). We then have the following master problem that generates cuts to stage \(t\) and proposals for stage \(t + 1\):

\[
\begin{align*}
\min c_{i,t} x_{i,t} + \theta_{i,t} \\
\text{s.t.} \quad W_{i,t} x_{i,t} &= h_{i,t} - T_{i-1} x_{i,t-1} - d_{i,t}, \\
D_{i,t} x_{i,t} &\geq d_{i,t}, \quad j = 1, \ldots, r_{i,t}, \\
E_{i,t} x_{i,t} + \theta_{i,t} &\geq e_{i,t}, \quad j = 1, \ldots, s_{i,t}, \\
x_{i,t} &\geq 0,
\end{align*}
\]

where \(x_{i-1,t} a_0\) is the current solution from the ancestor scenario, \(a(k)\), and where, for \(t = 1\), we interpret \(b = h_{1,t} - T_{0,1} x_{0,1}\) as initial conditions of the problem.

The nested decomposition method iterates among problems of the form (42-46) from stage \(t\) to \(t + 1\) passing forward proposals, \(x^t\), and from \(t + 1\) to \(t\) passing backward prices in the form of cuts. With suitable choices of these movements, the method terminates finitely.

Many alternative strategies are possible in this algorithm for determining the next subproblem (42-46) to solve. For feasible solutions, one approach explores all scenarios at \(t\) before deciding to move to \(t - 1\) or \(t + 1\). For feasible iterations, the algorithm often proceeds in a consistent direction either forward (to \(t + 1\)) or backward (to \(t - 1\)) as long as movement in that direction is possible. This approach is the "fast forward--fast back" procedure proposed by Wittrock\(^{137}\) for deterministic problems and implemented by Gassmann\(^{138}\) for stochastic problems. One may alternatively enforce a move from \(t\) to \(t - 1\) ("fast back") or from \(t\) to \(t + 1\) ("fast forward") whenever it is possible. From experiments conducted by Gassmann,\(^{133}\) Birge et al.,\(^{138}\) and Morton\(^{139}\) (who develops several alternative protocols), the fast forward--fast back sequencing protocol seems generally to work better than either of these alternatives.

This decomposition method also allows significant parallel computation. Ariyawansa and Hudson\(^{141}\) show that the two-stage model scales relatively well on parallel processors when the serial methods do not take particular advantage of similarities among scenarios. Sequential solutions may, however, take advantage of a solution for a single subproblem (39) to solve others. The methods, called trickling (Gassmann\(^{138}\) or bunching (see Wets\(^{139}\)), effectively generate a
tree of optimal bases for subproblems (39). In some cases, these approaches achieve significant efficiencies.

Other efficiencies are often possible by replacing θ in (32, 34) with ∑xₖ pₖθ_k and then placing cuts on each of the θₖ variables using only the θₖth subproblem prices, πₖ. For a given number of main problem solutions, this multicut approach (from Birge and Louveaux[22]) produces a more complete characterization of the polyhedral structure of the graph of 2(c), resulting in generally fewer major iterations. The expense of an expanded main problem, however, sometimes overcomes this efficiency (in particular, for small problems).

Comparisons on these approaches appear in Birge et al.[147] Figure 3 (from [18]) presents comparisons for solution times of four problems with varying structures (see references in [18]). Note that the scales represent the logarithms of times in CPU seconds and the number of deterministic equivalent variables. The deterministic equivalent problems were solved on an IBM RS/6000 320H using IBM's OSL primal simplex algorithm (IBM[69]), Gassmann's MSLIP (Gassmann[33]) solved the problems with both single cuts and no trickling except at the last stage (MSLIP-SC) and multicuts and trickling throughout the subproblem tree (MSLIP-MC). ND-UM refers to a nested decomposition code developed at the University of Michigan, with times reflecting single cuts and no bunching or trickling. ND-Bunch times give some indication of bunching advantages on the problem labeled STOCHFIR in Figure 3 (which is taken from Gassmann[33]).

Figure 3 also presents the results from a parallel implementation under Best Parallel. This version implemented the nested decomposition method in parallel over a network of RS/6000s using the PVM (Geist et al.[159]) protocol. Efficiencies (fractions of available CPU times used on all processors) exceeded 50% in all cases with an average of greater than 75% efficiency for the large problems.

In all cases, the deterministic equivalent problems became too large to solve directly on the available workstations. Of the examples here, the direct simplex method appears only superior (by a small amount) for the smaller STORM problems (taken from Ruszczyński[114]). Other studies (see Ruszczyński[114]), however, indicate that direct methods lose efficiency significantly on this problem as the number of scenarios increases. In all problems except SCFM, the decomposition method times are approximately linear in the number of variables. For SCFM, the growth is much less than linear because of the decomposition's ability to find a solution quickly that also solves most of the scenarios.

The results also show that trickling or bunching and multicuts generally offer greater efficiencies (except on SCFM where a single scenario dominates the others). Other efficiencies are also possible in some problems by adding a regularizing term, ½∥x − x∗∥², to the objective in (32). This approach from Ruszczyński[112, 113, 114] appears to offer advantages when initial cuts force the iterates, x*, to oscillate widely after initialization. Ruszczyński[114] also includes a compact representation to eliminate consideration of slack constraints and gain additional savings. The method can also be extended directly to multiple stages and again offers many possibilities for parallel computation (see Ruszczyński[112] and the implementation by Korycki[79]).

The regularized method also introduces possibilities for nonlinear objectives in the decomposition framework. Louveaux[203] provides one approach for multistage problems with quadratic objectives. Following the deterministic approach for convex objectives in O'Neill,197 Pereira and
Pinto\cite{98, 99} developed a method with nonlinear objectives. Noël and Smeers\cite{60} extended the basic method to treat nonlinear constraints within a time period. Further developments appear in Birge and Rosa\cite{23} with solutions of examples with over 4000 variables involving nonlinearities and over 1400 nonlinear constraints.

Extensions of these cutting-plane methods are also possible in problems with integer variables. If all integer variables are in the first stage, then the decomposition procedures apply directly (see Wollmer\cite{138}). When integer variables appear in future stages, however, methods become more difficult because the recourse function, \( \tilde{z}(x) \), loses convexity. One remedy is to force all integer variables (repeated for different scenarios) into a first stage, as done by Bienstock and Shapiro.\cite{125} Alternatively, for two-stage problems, La porte and Louveaux\cite{78} provide valid cutting procedures that adapt the L-shaped method to problems with integer variables. Specific procedures for constructing the convex hull of the recourse function are also available.\cite{173, 74, 75, 82}

General references on stochastic integer programs include Stougie\cite{120} and Van der Vlerk.\cite{125}

Whereas these methods all apply to simplex solutions of (32–34) and (42–46) (or at least to a solution with optimal values to check convergence), interior point methods can also be adapted to take advantage of the problem structure without direct solution of the full deterministic equivalent problem. One approach, the Analytic Center Cutting Plane Method, appears in Bahn et al.\cite{23} Bahn et al. generate cuts to reduce a main problem feasible region and then move toward the center of that region with successive subproblem solutions and improved objective function values. Their results on randomly generated problems of moderate size (up to 10,000 variables) give speed-ups from one to two orders of magnitude over both direct-interior (dual formulation) and extreme-point methods. The relative advantage also increases with problem size. These results confirm the potential for improvements over direct solvers and offer some possibilities for improvements over simplex-based decomposition methods in certain cases.

### 4.2 Decomposition Methods with Sampling

As mentioned earlier, stochastic programming formulations often include some approximation of an underlying probability distribution. When that distribution is known, it is common to use a sample approximation. A potential disadvantage of sampling, however, is that some effort might be wasted on an inaccurate approximation. An approach to avoid this problem is to use sampling within another algorithm without complete optimization. A natural candidate is to embed sampling into the L-shaped method described in the previous section. We consider two such approaches, one based on large samples to derive cuts (see Dantzig and Glynn\cite{40}) Dantzig and Infanger\cite{41} and the other based on a sample that grows as the algorithm progresses (see Higle and Sen\cite{56}).

The first approach by Dantzig, Glynn, and Infanger is to sample \( Q(x, \xi) \) in the L-shaped method instead of actually computing \( \tilde{z}(x) \). Given an iterate \( x^k \), the result is an estimate,

\[
2^{-\nu k^2} = \left( (1/\nu) \Sigma_{i=1}^{\nu} Q(x^k, \xi^i) \right),
\]

and an estimate of \( \tilde{z}(x^k) \) as \( \tilde{z}_k = \left( 1/\nu \right) \Sigma_{i=1}^{\nu} \tilde{z}_i \) where \( \tilde{z}_i \in \delta Q(x^k, \xi^i) \). Now, for \( Q \) convex in \( x \), one obtains

\[
Q(x, \xi) \geq Q(x^k, \xi) + \tilde{z}(x - x^k), \quad (47)
\]

for all \( x \). Assuming that we also have a finite value in \( Q(x^k, \xi) \) for any \( \xi \) to prevent problems with infeasibility,

\[
2 \alpha(x) = \left( \frac{1}{\nu} \right) \left( \sum_{i=1}^{\nu} Q(x, \xi^i) \right) \geq 2 \alpha(x^k) + \tilde{z}(x - x^k) = LB(x), \quad (48)
\]

where \( \nu \) times the right-hand side in (48) is asymptotically normally distributed (see, e.g., Shapiro\cite{117}).

Infanger\cite{41} makes several assumptions to form confidence intervals on optimal values with this method. Experiments with large problems appear in Dantzig and Infanger\cite{41}. Their results improve significantly with importance-sampling variance-reduction techniques.

A central alternative, called stochastic decomposition, by Higle and Sen\cite{56} generates many cuts with increasingly large samples built on previous samples and adjusts these cuts to drop as the algorithm continues processing. Higle and Sen assume complete recourse and a known (probability 1) lower bound on \( Q(x, \xi) \) (e.g., 0). They also assume that the set of dual solutions to the recourse problem (39) is bounded and that the set \( \{ x | Ax = b, x \geq 0 \} \) and \( \Xi \) are also compact.

With these assumptions, the basic Stochastic Decomposition Method generates iterates, \( x^k \), and observations, \( \xi^k \), in the following way.

#### Stochastic Decomposition Method

**Step 1.** Set \( k = 0, \xi^0 = \xi^f \) and let \( x^1 \) solve

\[
\min_{x \in \Xi} \left\{ c^T x + Q(x, \xi^0) \right\}. \quad (49)
\]

**Step 2.** Let \( k = k + 1 \) and \( \xi^k \) be an independent sample generated from \( \xi \). Find \( 2 \alpha(x^k) = (1/k) \Sigma_{i=1}^{k} Q(x^k, \xi^i), \xi^k = (1/k) \Sigma_{i=1}^{k} \xi_i \). Let \( \tilde{z}_k = (1/k) \Sigma_{i=1}^{k} \tilde{z}_i \) and \( \xi^k = (1/k) \Sigma_{i=1}^{k} \xi_i^k \).

**Step 3.** Update all previous cuts by \( E_i \leftarrow [(k - 1)/k] E_i \) and \( e_i \leftarrow [(k - 1)/k] e_i \) for \( i = 1, \ldots, k - 1 \).

**Step 4.** Solve the L-shaped master problem as in (32–34) to obtain \( x^{k+1} \). Go to 1.

This method differs slightly from the basic method in Higle and Sen\cite{56} in our assumption \( \Sigma_{i=1}^{k} \xi_i \) to be optimal dual solutions in each iteration. Higle and Sen allow a restricted set of dual optima that may decrease the solution effort (with perhaps fewer effective cuts). The main convergence result is contained in the following theorem.

**Theorem 2.** Assuming complete recourse, \( Q(x, \xi) \geq 0 \), bounded dual solutions to (39), \( \{ x | Ax = b, x \geq 0 \} \) and \( \Xi \) compact, there exists a subsequence, \( \{ x^* \} \), of the iterates of the Stochastic Decomposition Method such that every limit point of \( \{ x^* \} \) solves the recourse problem (25-5LPR) with probability one.

**Proof.** See Higle and Sen.\cite{59}}
One difficulty in this basic method is that convergence to an optimum may only occur on a subsequence. To remedy this, Higle and Sen suggest retaining an incumbent solution which changes at low objective function values. The incumbent is updated each time a sufficient decrease in the kth iteration objective value is obtained. They also show that the sequence of incumbents contains a subsequence with optimal limit points and how this subsequence can be identified. Various approaches may be used for practical stopping conditions. Their results show good convergence behavior on a variety of problems, especially those arising in network design (see Sen, Doverspike, and Cosares[113]).

5. Lagrangian-Based Approaches

The methods in Section 4 are all motivated by the linear formulations, although extensions include nonlinear problems. Lagrangian-based methods, however, have their motivation directly in nonlinear problems and may provide more flexibility than the decomposition approaches. We describe the main approaches next.

Consider the general nonlinear stochastic program given in (8). We can consider a variety of approaches to solving (8) based on available nonlinear programming methods. For example, we may consider gradient projection, reduced-gradient methods, and straightforward penalty-type procedures. These methods generally assume, however, that gradients of functions, $E_{\sigma}(f_i, x'_i, x_i, \sigma)$ are available and relatively inexpensive to acquire. This is not often the case in stochastic programs because each evaluation may involve solving several problems. Lagrangian approaches have been proposed to avoid this problem by disaggregating the objective function terms in (8).

5.1 Lagrangian Dual Ascent

The basic motivation behind the Lagrangian approaches is that only the nonanticipativity constraints (9) link the scenarios. If these constraints are relaxed, then the solution reduces to separate problems for each realization of $\xi$. To see how the procedure develops, assume a multiplier, $\pi$, (defined appropriately as in, for example, Rockafellar and Wets[116]) to obtain a dual problem to (8)

$$\max w = \theta(\pi),$$

where

$$\theta(\pi) = \inf_{x,k} E \left( \sum_{i=1}^{k} f_i(x_i, x'_i, x_{i+1}(\sigma)) \right)$$

where

$$X$$ represents all constraints in $X_\xi(\sigma)$ and $\pi'$ corresponds to the first period components of $\pi$.

As an example, we consider the two-stage case with finite support of $\xi$. In this case, if we suppose the $x_0$ are data in $f_0$, $x_1$ becomes $x$, and $x_2$ becomes $y$, then (8) becomes

$$\min z = \sum_{i=1}^{k} p_i(f_i(x_i) + f_2(x_2, y_2))$$

$$+ \sum_{i=1}^{k} p_i x_i = 0, k = 1, \ldots, K$$

The dual program is then

$$\max w = \theta(\pi)$$

$$= \inf_{x,y} \sum_{i=1}^{k} p_i \left( f_i(x_i) + f_2(x_2, y_2) \right)$$

$$+ \pi(y - \sum_{i=1}^{k} p_i x_i)$$

We state the standard duality result (for a proof, see, e.g., Bazaraa, Sherali, and Shetty[14]) Theorem 6.2.4) for the finite distribution case in the following theorem.

Theorem 3. Suppose the stochastic nonlinear program (52) with all functions convex has a finite optimal value, and suppose $\Omega = \{1, \ldots, K\}$ with $P[w = k] = p_k$ for $k = 1, \ldots, K$, then $z = w$ for every feasible $x_1, y_1, \ldots, y_K$ in (52) and $x_1, \ldots, x_K$ feasible in (53), and their optimal values coincide, $z^* = w^*$.

We can now describe a basic gradient method for the dual problem. For our purposes, we assume that (53) always has a unique solution.

Lagrangian Dual Ascent Method

Step 0. Set $\pi^0, \nu = 0$ and go to Step 1.

Step 1. Given $\pi = \pi^\nu$ in (53), let a solution be $x_1^\nu, \ldots, x_K^\nu, y_1^\nu, \ldots, y_K^\nu$.

Step 2. If $x_1 - x_2^{\nu-1} p_k x_2 = 0, k = 1, \ldots, K$, stop (optimal).

Otherwise, let $\pi_k = x_k - \sum_{k=1}^{K} p_k x_k$ and go to Step 3.

Step 3. Let $\lambda^\nu$ minimize $\sum_{k=1}^{K} (\pi_k^2 + \lambda^2)$ over $\sigma^2 + \lambda^2$. Stop if it is feasible in (53).

Assuming the unique solution property, this algorithm always produces an ascent direction in $\sigma$. The algorithm either converges finitely to an optimal solution or, assuming a bounded set of optima, produces an infinite sequence with all limit points optimal. With multiple optima in (52), some nondifferentiable procedure must be used. In this case, one could consider finding the maximum norm subgradient to be assured of ascent, or one could use various bundle type methods (see, e.g., Lemaréchal[99] and Kiwiel[172]).

The basic hope for computational efficiency in the dual ascent procedure is that the number of dual iterations will be small compared to the number of function evaluations that might be required by directly attacking (8). Substantial time may be spent solving (53) but that should be somewhat easier than solving (8) because the linking constraints appear as objective terms instead of hard constraints. Overall, however, this type of procedure generally is slow due especially
to a single-point linearization of \( \theta \). This observation has led to other types of Lagrangian approaches to (8) that use more global or second-order information.

Rockafellar and Wets\(^{109}\) suggest one such procedure for a special case of (52) where \( f_k \) is equivalent to a convex quadratic function over a convex region, and where \( f_0 \) is equivalent to a quadratic function subject to linear constraints. Their approach is, however, to restrict the search not just to a single search direction but to allow optimization over a low-dimensional set in the dual space. An implementation of their method (Lagrangian finite generation method for linear-quadratic stochastic linear programs) is described in King.\(^{109}\) The method was used successfully to solve practical water-management problems concerning Lake Balaton in Hungary (Somlyódy and Wets\(^{119}\)).

5.2 Augmented Lagrangian Methods

Another method for speeding convergence is to ensure strictly convex objective terms. Rockafellar and Wets\(^{109}\) discuss methods for adding quadratic terms to ensure positive definite Hessians. In this way, the finite generation method becomes a form of augmented Lagrangian procedure.

In a general augmented Lagrangian approach (see, e.g., Bertsekas\(^{88}\)), one adds a penalty \( r/2||x_k - \sum_{k=1}^{K} p_k z_k||^2 \) to each term \( k \) of \( \theta(x) \) and performs the iterations with a fixed step size such that \( \pi_k^{k+1} = \pi_k^k + r p_k (x_k - \sum_{k=1}^{K} p_k z_k) \). The advantage (see, e.g., the discussion in Dempster\(^{46}\)) is that Newton-type steps can be applied because the method maintains a nonsingular Hessian. The result generally should be that convergence becomes superlinear in terms of the dual objective without a significantly greater computational burden over pure dual ascent.

The computational experience reported by Dempster suggests that this procedure may be quite efficient with a new variable, \( x_0 \), substituting for \( \sum_{k=1}^{K} p_k z_k \) to solve for

\[
\hat{\theta}(x) = \min_{x, y} f_1(x_0) + \sum_{k=1}^{K} p_k l_k f_2(x_k, y_k) + \pi_k(x_k - x_0) + (r/2)||x_k - x_0||^2 .
\]

(54)

Iterations alternate between searches in \( x_k \) and then separable solutions in each \( x_k, y_k \). In this way, the augmented Lagrangian approach of solving (54) to find a dual ascent Newton-type direction achieves superlinear convergence in dual iterations. The only problem may be the time to construct the search directions through solutions of (54).

This method is similar to the Progressive Hedging Algorithm (PHA) of Rockafellar and Wets\(^{110}\) which actually achieves a full separation of the separate scenario problems for each iteration, resulting in considerably less work per iteration (but possibly an increased number of iterations). PHA appears to offer many computational advantages at least for structured problems as reported by Mulvey and Vladimiroiu.\(^{91}\) A related approach is the extension of the row-action algorithm of Censor and Lent\(^{63}\) by Nielsen and Zienois\(^{94, 95}\) which has particular efficiencies for network constraints. The key to the success of those methods is that individual subproblem structure is maintained throughout the algorithm. We describe the basic method for two-stage problems. Extensions to multistage use slight modifications of the expectation operator in the algorithm below.

The basic progressive hedging method begins with a nonanticipative solution \( \bar{x}^\nu \) and a multiplier \( \pi^\nu \). The nonanticipative (but not necessarily feasible) solution is used in place of \( x_0 \) in (54). The first period constraints are also split into each \( x_k \). In this way, we obtain a subproblem

\[
\inf_{x} z = \sum_{k=1}^{K} p_k l_k f_1(x_k) + f_2(x_k, y_k) + \pi_k(x_k - x_0) + (r/2)||x_k - x_0||^2 ,
\]

(55)

where all constraints are assumed represented within the objective functions. Now, (55) splits directly into subproblems for each \( k \) so these can be treated separately. Supposing that \( (x_k^{r+1}, y_k^{r+1}) \), \( k = 1, \ldots, K \), solves (55), we obtain a new nonanticipative decision by taking the expected value of \( x_k^{r+1} \) as \( \bar{x}_k^{r+1} \) and step in \( x_k \) by \( \pi_k^{r+1} = \pi_k^r + (x_k^{r+1} - \bar{x}_k^{r+1}) \).

The steps then are simply stated as below.

**Progressive Hedging Algorithm**

**Step 0.** Suppose some nonanticipative \( x_0 \), some initial multiplier \( \pi^0 \), and \( r > 0 \). Let \( \nu = 0 \). Go to Step 1.

**Step 1.** Let \( (x_k^{r+1}, y_k^{r+1}) \) for \( k = 1, \ldots, K \) and solve (55). Let \( \bar{x}_k^{r+1} = \sum_{k=1}^{K} p_k l_k f_2(x_k^{r+1}, y_k^{r+1}) \) for all \( k = 1, \ldots, K \).

**Step 2.** Let \( \pi_k^{r+1} = \pi_k^r + (x_k^{r+1} - \bar{x}_k^{r+1}) \). If \( \pi_k^{r+1} = \pi_k^r \) and \( x_k^{r+1} = \bar{x}_k^{r+1} \), then, stop; \( \bar{x}^r \) and \( \pi^r \) are optimal. Otherwise, let \( \nu = \nu + 1 \) and go to Step 1.

The convergence of this method is based on Rockafellar's proximal point method.\(^{107}\) The basis for this approach is not dual descent but the contraction of the pair, \( (x_k^{r+1}, \pi_k^{r+1}) \), about a saddle point.

Mulvey and Vladimiroiu\(^{91}\) report on comparisons for generalized networks with linear and nonlinear objectives. Their results indicate that PHA solution times were generally better than direct solution with nonlinear codes, such as MINOS (Murtagh and Saunders\(^{92}\)), and, in most cases, interior point methods. The results were not overwhelmingly positive but they mention the advantage that PHA may have for situations with limited memory and potential for parallel processing.

Nielsen and Zienois\(^{94, 95}\) report significant improvements with their parallel implementation over serial methods. Their results show single order-of-magnitude speeds-up with parallel implementation (which is comparable to serial decomposition results), but they also obtain solutions to larger problems than they could solve serially.

The general structure of the augmented Lagrangian methods and PHA allows for a variety of conditions on the objective functions (and implicitly defined constraints). In fact, when the number of scenarios increases in augmented Lagrangian formulations, the duality gap for the integer problem actually goes to zero (Birge and Dempster\(^{127}\). Al-
though the augmented Lagrangian can be extended easily in this way to integer variable problems, PHA may not converge in certain integer cases (see Takriti, Birge, and Long1229). In structured power-generation scheduling problems, PHA was, however, able to obtain solutions quite efficiently for integer problems despite the lack of guaranteed convergence.

Mulvey and Ruszczynski1003 have developed a variant of the augmented Lagrangian method called diagonal quadratic approximation. Their approach is to write the nonanticipativity constraints with a permutation order, \( \sigma_j, k = 1, \ldots, K \), as \( x_k - x_{\sigma(k)} \), \( k = 1, \ldots, K \), where the permutation is used to link all constraints but to introduce only a single additional constraint for each subproblem. Then, they approximate the \( \| x_k - x_{\sigma(k)} \| \) terms in the objective by using a current iterate, \( \bar{x}_k \). The result is that the original augmented Lagrangian problem then decomposes entirely by \( k \) into subproblems

\[
\inf z_k = f_k(x_k) + f(x_{\sigma(k)} - x_k) + \frac{\alpha}{2} \| x_k - x_{\sigma(k)} \|^2 + \| x_k - \bar{x}_{\sigma(k)} \|^2,
\]

(56)

where \( \sigma^{-1}(k) \) refers to the scenario \( j \) such that \( \sigma(j) = k \). This method also allows for significant parallelization. Mulvey and Ruszczynski report results that outperform direct procedures for linear problems but do not appear as efficient as decomposition methods. The diagonal quadratic approximation method is, however, completely generalizable to nonlinear problems as in standard augmented Lagrangian approaches. Its primary advantage may, therefore, be for loosely coupled computing environments to solve complex nonlinear problems that decompose relatively easily.

6. Recent Applications

Stochastic programming has seen great gains since its original applications in airline fleet assignment by Ferguson and Dantzig.149 Our aim in this section is to show the advantages obtained in solving stochastic problems in place of deterministic models. These examples arise in all sectors of the economy.

6.1 Finance

Financial problems inherently involve uncertainty. These problems are especially attractive for stochastic formulations because deterministic approximations are quite obviously flawed. For example, using a deterministic expected return results in investments only in the highest risk (and also highest expected return) asset, but few people are willing to follow such a strategy. The essence of a stochastic program is to develop a hedging strategy that mixes investments in a way that a strictly deterministic approximation cannot reproduce.

As an example, we consider the Russell-Yasuda Kasai Model (see Caritho et al.1231), which won the second prize in the 1993 Franz Edelman Award Competition for Management Science Achievement. In that model, decisions are made for a Japanese insurance company on how best to invest in assets to meet a random liability stream over time. The investment returns are also random. Additional complications include legal restrictions about the use of income to meet liabilities.

The overall Russell-Yasuda Kasai model has the structure in (MS-SLP) with a piecewise linear convex objective. The following description is the abbreviated model in [31]. The problem involves \( H = 6 \) time periods with increasing period lengths and number of branches to represent greater detail in near-term conditions that most affect the current decision. The model includes eight branches in the first period representing the next quarter, four branches in a second period for the following three quarters, four branches again in the fourth period for the following year, two branches in the fifth period for years three to five, and one branch in period six to represent terminal conditions (see Figure 4). The model involves a total of 256 scenarios.

The main decisions are the market value held in a variety of assets, \( v_j(t), j = 1, \ldots, J \). Other variables are necessary to calculate objective function values and to balance accounts. The wealth value \( W_t \) represents total fund market value at \( t \), \( u_t \) represents income shortfall, and \( \eta_t \) represent income surplus. The shortfall enters the objective function through a piecewise linear function \( c(u_t) \).

The model also involves a number of parameters that determine investment effects and reflect regulatory requirements. The key random parameters include

- \( \epsilon_{j+1} \) — price return of \( j \)th asset in period \( t + 1 \);
- \( \epsilon_{j+1} \) — income return of \( j \)th asset in period \( t + 1 \);
- \( F_{j+1} \) — deposit inflow in period \( t + 1 \);
- \( P_{j+1} \) — principal payout in period \( t + 1 \);
- \( I_{j+1} \) — income payout in period \( t + 1 \);

![Figure 4. Six-period scenarios for the Russell-Yasuda Kasai model.](image)
• $g_{t+1}$—rate of interest paid on policies in period $t + 1$;
• $l_{t+1}$—liability valuation at the end of period $t + 1$.

The overall model is then to find:

$$\max \ E_0 \left[ W_H - \sum_{t=1}^{n} c_t(u_t) \right]$$

subject to:

$$\sum_{j=1}^{J} w_{ij} - W_i = 0,$$

$$W_{t+1} = \sum_{j=1}^{J} (1 + rp_{j,t+1} + ri_{j,t+1}) w_{ij} = F_{t+1} - P_{t+1} - L_{t+1},$$

$$\sum_{j=1}^{J} ri_{j,t+1} w_{ij} + u_{t+1} - v_{t+1} = g_{t+1} L_{t+1},$$

$$w_{ij}, v_{t+1}, u_{t+1} \geq 0,$$

where $j = 1, \ldots, J$, $t = 0, \ldots, H - 1$,

and

$$L_{t+1} = (1 + g_{t+1}) L_t + F_{t+1} - P_{t+1} - L_{t+1},$$

to be consistent. Other constraints are actually present in the practical model but the factors above give the main problem structure.

The entire model involves 256 scenarios with each having a significant number of variables. Carifio et al. use a nested Benders decomposition method to solve these models on workstations with available commercial software (OSL). Their approach was generally capable of solving the full model from a cold start in less than three hours and from a hot start in under twenty minutes. They report (311) that ongoing improvements have led to further computational reductions and problem-size increases.

For such financial models, deterministic solutions clearly provide inadequate solutions because they ignore risk. Static portfolio models, based on Markowitz's model, have, however, been used in many cases. The difficulty with this approach is that it ignores the transaction costs required to rebalance the portfolio and meet liabilities. The result can be significant transaction costs and unnecessary liquidation of assets.

The solution of Russell-Yasuda Kasai model improved significantly over the static portfolio procedures that had been used previously. The authors state that the Russell-Yasuda Kasai model yielded additional income of $379 million in its first two years of use. The qualitative advantage of the model is that it allows flexibility across accounts and the ability to react to future events with a dynamic strategy that would not be possible in any static model.

The Russell-Yasuda Kasai model experience confirms results from others, such as Kusy and Ziembba's study of bank asset and liability management. They also found significant advantages to the stochastic model even with limited abilities to change holdings within each period. A recent article in Business Week describes other successes and lists stochastic programming as a key new tool for all chief financial officers.

### 6.2 Manufacturing

The complexity and inherent uncertainty in both demand and supply aspects of manufacturing make this area also particularly fruitful for stochastic programming models. Eppen, Martin, and Schrage provide an interesting example for a capacity-planning model at General Motors. The goal was to determine capacity for various products at a number of plants to maximize an expected profit objective with a downside risk constraint. The model fits the form (25-SPLR) with the addition of a risk constraint imposing an additional link across scenarios.

The basic aim of the model was to decide what configuration $h(t)$ would hold at each plant site $i$ in each time period $t$. A configuration is represented through binary variables, $a_{ip}$. The operation of retooling involves another binary decision, $w_{i(t)}$. These decisions are assumed independent of the future outcomes, in particular, of demand. This property, called block separable recourse, allows solution of multistage problems in two stages corresponding to capacity decisions in stage 1 and production decisions in stage 2.

Assuming $y_i$ for the other decision variables, the capacity planning model has the form:

$$\min_{a_{ip}, \ u \in [0, 1]} \ \sum_{i=1}^{n} \ \sum_{p=1}^{P} \ a_{ip} x_{ip} + \sum_{i=1}^{n} \ \sum_{p=1}^{P} \ y_i,$$

subject to:

$$A_i u + A_i w = b_i,$$

$$Q(u, w) = \min q(w) y(w)$$

where $\theta(y, w) = E(Q(u, w, \omega))$ with

$$Q(u, w, \omega) = \min q(w) y(w)$$

and

$$\max \ \sum_{i=1}^{n} \ \sum_{p=1}^{P} \ a_{ip} x_{ip} + \sum_{i=1}^{n} \ \sum_{p=1}^{P} \ y_i,$$

subject to:

$$W y(w) = h(\omega) - T_i u - T_i w, \ y \geq 0,$$

where the uncertainty involves demands and prices. Eppen, Martin, and Schrage's model includes three scenarios in each of five periods (with the last period representing all future events) and a total of 243 total scenarios.

Beyond the purely linear, mixed integer model in (58), Eppen, Martin, and Schrage imposed an additional constraint to reflect an increased risk attitude for the decision makers at General Motors. Assuming a profit $\pi(u, w, \omega)$ as the negative of $\pi^2 + \pi^4 + Q(u, w, \omega)$ from (58) and a target $\bar{\pi}$, they defined downside risk as:

$$f(\pi(u, w, \omega)) = (\pi - \pi(u, w, \omega))^+$$

They then used the expected downside risk, $E_{\omega}[f(\pi(u, w, \omega))]$, in an additional constraint in (58).

To solve the stochastic model, Eppen, Martin, and Schrage describe a solution with the LINDO (Linear and Discrete Optimizer) procedures for mixed integer linear programs. They find a solution of the full problem with almost
1000 rows and over 2000 variables (including 160 binary variables) in 1.3 CPU hours on a VAX 8650. They use a spreadsheet environment to enter the data and then various matrix generation routines to prepare the data in a suitable form for the mainframe version of Lindo.

The result of their stochastic model is an increase of over 6% in expected profit over two potential deterministic scenarios and complete avoidance of downside risk that occurred in another potential scenario. In this case, no individual deterministic scenario could provide the optimal stochastic programming (hedging) solution. Epper, Martin, and Schrage also report that the model had the additional advantage of highlighting products that should not be in an optimal product mix. These model outcomes caused serious evaluation of the products at General Motors and may have led to substantial savings beyond the specific recommendations of the capacity analysis.

6.3 Telecommunications
Telecommunication systems planning and operations naturally involve many uncertainties in terms of system traffic, performance, and reliability. Network design and configuration decisions require consideration of numerous point-to-point demands with frequent high-variance forecasts. Sen, Doverspike, and Cosares, for example, describe a private-line service problem with 86 demand pairs, 89 links, and 706 potential routes.

In Sen, Doverspike, and Cosares's model, \( n_t \) represents the number of links considered for additional capacity whereas the decision variables, \( x_{j} \), \( j = 1, \ldots, n_t \), give the additional capacity in each link \( j \). The total capacity that can be installed is \( b \).

Given the capacity decisions subject to the capacity budget constraint, the objective is to minimize the expected number of unserved link requests. For a single demand realization, \( \xi \), for \( i = 1, \ldots, m_2 \) point-to-point pairs, the recourse function is:

\[
Q(x, \xi) = \min_{y} \sum_{i=1}^{m_2} s_i
\]

\[
\text{s.t.} \sum_{i=1}^{m_2} W_{i} y_{i} = h^{0} + x,
\]

\[
\sum_{i \in R(i)} y_{i} + s_{i} = \xi,
\]

where \( y_{i} \) is the number of lines for pair \( i \) directed along route \( r \), \( s_{i} \) gives the number of unserved requests, \( W_{i} \) is an accident matrix for the route \( r \) of pair \( i \) with the links \( j \), and \( h^{0} \) is the \( n_1 \) vector of existing link capacities.

The overall objective is then to find:

\[
\min_{x \in \mathbb{R}^{n}} \mathcal{A}(x)
\]

\[
\text{s.t.} \quad e^{T} x \leq b,
\]

where \( \mathcal{A}(x) = \int_{x} Q(x, \xi) \mathcal{P}(d\xi) \).

Sen, Doverspike, and Cosares solve (60) using the stochastic decomposition method. For an initial solution based on optimization for the mean, they estimate over 24 unserved line requests. After 1100 iterations, they obtain a solution with just over five unserved requests in expectation. The result in this case is that the value of the stochastic solution is almost four times the optimal recourse objective value.

The model assumes a static network while in fact routing decisions are made dynamically. Sen, Doverspike, and Cosares conducted simulations of the dynamic situation, however, and discovered consistent results on overall blocking and the effect of different plans generated in the stochastic decomposition procedure.

6.4 Transportation
Transportation models remain a prominent area for application of stochastic programming models. From Ferguson and Dantzig’s models, several other areas have developed. An area of particular focus has been in dynamic vehicle allocation (see, e.g., Frantzeskakis and Powell, Powell, Powell and Chung).

The basic problem in dynamic vehicle allocation is to route a set of vehicles (notably trucks or freight cars) to meet demands along routes and to position them for anticipated future loads. The overall goal is to maximize returns over given horizons. The model again fits the form of (MS-SLPR) but the wide distribution on load demands makes approximation schemes critical.

Powell and his colleagues have developed a range of approximations that effectively approximate the value function, \( \mathcal{A}(x) \), at each time period and yield a form of dynamic programming approximation. Their results with realistic vehicle fleet sizes of up to 1000 vehicles indicate approximation errors that are often below 10%. Other experiments (Powell) show that their methods, coupled with a dispatcher’s decision making, yields 10% to 15% gains over purely manual dispatches.

6.5 Other Application Areas
Power system models have also been a common area of application and source of developments in stochastic programming methods. Pereira and Pinto use decomposition procedures for models of the Brazilian power system which includes a number of reservoirs that must be controlled with uncertain inputs from precipitation as well as uncertain loads and other sources. A similar model, Socrates (Jacobs et al.), is used at Pacific Gas and Electric for coordination of their wide range of hydrological power sources.

As an example of the potential of these models, Takriti, Birge, and Long consider a model of the Michigan power system. This model is designed for daily scheduling and, therefore, explicitly includes binary variables for an individual unit’s state of operation (on or off). Their procedure uses the Progressive Hedging Algorithm to achieve convergence to near optimal solutions quickly. The results for a sample
Table I. Selected Applications of Stochastic Programming, Solution Methods, and Values of the Stochastic Solution

<table>
<thead>
<tr>
<th>Reference</th>
<th>Area</th>
<th>Model Type</th>
<th>Solution Method</th>
<th>VSS (cf. optimal value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31]</td>
<td>Finance</td>
<td>Linear, multistage</td>
<td>Nested decomposition</td>
<td>4%*</td>
</tr>
<tr>
<td>[77]</td>
<td>Finance</td>
<td>Linear, multistage</td>
<td>Simple recourse [134]</td>
<td>10%</td>
</tr>
<tr>
<td>[47]</td>
<td>Manufacturing capacity</td>
<td>Linear, integer</td>
<td>Mixed integer [115]</td>
<td>6%</td>
</tr>
<tr>
<td>[116]</td>
<td>Telecommunications</td>
<td>Linear</td>
<td>Stochastic decomposition [56]</td>
<td>362%</td>
</tr>
<tr>
<td>[49]</td>
<td>Fleet assignment</td>
<td>Linear, two-stage</td>
<td>Generalized network</td>
<td>9%</td>
</tr>
<tr>
<td>[102]</td>
<td>Vehicle allocation</td>
<td>Linear multistage</td>
<td>Dynamic network, approximation</td>
<td>10%</td>
</tr>
<tr>
<td>[124]</td>
<td>Power generation</td>
<td>Linear, multistage, integer</td>
<td>Lagrangian, progressive hedging</td>
<td>1%</td>
</tr>
<tr>
<td>[26]</td>
<td>Energy planning</td>
<td>Nonlinear, multistage</td>
<td>Nonlinear nested decomposition</td>
<td>2%</td>
</tr>
</tbody>
</table>

* Compared to Mean-Variance model (VSS value equals $79$ million)

week show potential savings over a deterministic procedure of almost $150,000 in generation costs for that week alone.

Stochastic programming models have often been used in economic policy considerations as well. Manne considers the United States' decision on whether to begin or delay a breeder reactor program. The result shows a low cost of waiting for more information that may have influenced the decision to delay.

Birge and Resende consider a model of the U.S. economy with random returns on investment in new technologies with reduced carbon emissions. They consider various methods of limiting carbon emissions in the future because of environmental concerns. The stochastic nature of investment in this situation again can have a great effect on the preferred policy. Deterministic models yield solutions which concentrate investment on a given winner technology. The stochastic program solution hedges and produces acceptable results in a variety of circumstances. The overall value of the stochastic solution is as high as 2% of the total economic activity in the United States.

Table I presents a summary of some of these applications, their solution methods, and values of the stochastic solutions. Many other applications have been extensively explored (see, for example, King for other areas). These applications provide an example of the range of possible applications and the value of the stochastic programming solution compared to the deterministic model with means replacing random variables.

7. Future Directions

This article has provided a brief overview of developments in stochastic programming computation. Because of the size and complexity of these problems and the vast range of potential applications, many more developments can be expected in the future. We give some indication of the possibilities here.

7.1 Model Management

One particularly important development for practical implementations is the ability to represent a model in different forms and to attempt various solution methodologies, because no single approach appears to be best in all circumstances. Birge et al. provide a standard input format (called Stochastic Mathematical Programming System (SMPS)) to enable model sharing across platforms and solution methods. Recently, Gassmann and Schweitzer have proposed extensions to SMPS to enable a broader range of models consistent with the large-scale practical problems consistently solved today.

Standard formats have been of great use to researchers in the area, but practical implementations often require different views of models, rapid data changes, and structural changes consistent in different operations. Dempster and Ireland developed one such system, MIDAS, for financial decision making. Kall and Mayer present a general system that allows different representations of random variables, interfaces to several solvers, and provides a modeling language representation.

Today's model-management tools still depend on the modelers' ability to recognize and construct the basic building blocks of a stochastic program. For general practical use, the tools will need to provide greater assistance in the process of building the model. Model management tools should be a seamless part of other decision-support tools and eventually capable of extracting the necessary data for model development from interconnected enterprise-wide information and knowledge systems. This development is critical for stochastic programming to emerge as a common practical decision-making tool.

7.2 Uses of Specific Model Structure

Given the ability to form models, the computational methods described in this article show that structure can yield significant efficiencies. The process of identifying particularly useful structures should be another important area for research in the future. For example, the identification and use of model structure for efficient approximation should remain an important consideration.

In practice, significant savings in modeling time are pos-
sible with the ability to distinguish, through problem structure, those uncertain quantities that cannot be ignored from those that might easily be replaced with mean values. The ability to perform such preprocessing steps can be of particular benefit for stochastic programs (see Wallace and Wets (1967, 1969)). Other aspects of problem structure might be used to determine how sensitive solutions are to changes in the distribution, and whether given solutions might be robust (or close to optimal) over a range of distributions.

Optimization routines may benefit particularly from procedures tailored to particular problem structures. For example, discovering where a model might have particularly weak interactions between different parts might allow efficient decomposition across those links. This might also allow for more distributed forms of solution that would not require frequent synchronization.

7.3 Incorporation of Discrete Decision Variables

Practical problems, such as capacity planning in Section 6.2, often involve discrete decisions on, for example, whether to invest in a given technology or use a particular type of material. Research to date has indicated some potential for solving these problems when conditions, such as block-separable recourse (1971) or loose linkages (1984) warrant. In the most general circumstances, we have some results on extensions of cutting-plane methods as discussed in Section 4.1, but practical applications are still quite limited.

For broad applicability, stochastic programming methods must include discrete variables in a more comprehensive manner. Significant gains may be possible through additional uses of problem structure. An indication of the potential for this area appears, for example, in Bertsimas (1989) where a variety of stochastic optimization problems with discrete decisions are reduced to deterministic-equivalent or near-equivalent problems. Extensions of these results with higher-order approximation models may enable significant increases in the variety of practical problems that can be solved.

7.4 New Applications

The traditional applications, such as those mentioned in Section 6, generally involve planning-level decisions with focus on production and logistics. Greater impact from stochastic programming may, however, be possible at both the on-line operational level and the strategic, long-term design level.

In real-time systems, conditions change rapidly, but computational intensive solutions are not possible because of the need to maintain system control. Rapid or even instantaneous solution techniques are necessary in these environments. Some initial investigations of stochastic programming methods for on-line use (see Birge and Wets (1988) and Takriti (1991)) indicate that prior solution identification and on-line pattern-matching may have potential for application. True real-time operations, however, would require quicker, more reliable, and efficient methods.

The area of design also has potential (see, e.g., Birge (1988)) because of large potential losses and uncertainties in product uses and manufacture. Little progress on truly practical size problems has, however, been made. These problems generally involve significant nonlinearities that would preclude many of the linear-programming-based techniques. Special structure might again, however, lead to efficiencies in particular problem instances.

Conclusions

Progress continues rapidly in all areas of stochastic programming computation and modeling. While the state-of-the-art advances, recent textbooks by Kall and Wallace (1996) and Prékopa (1986) contain the basic methods and analysis that are especially appropriate for students and researchers entering the field. Interested readers can consult these references for further background on the approaches given in this article.

To keep a record of results to date and to provide access to standard problems and examples for testing, a World Wide Web site (http://www-personal.umich.edu/~jheirg/doe/hsolms/stoproc.html) has been established with instructions for obtaining test problems and for reporting computational results. The results from various methods on the test set problems are updated on the page to provide current information about the various techniques. We encourage everyone with computational experience in stochastic programming to participate by accessing the information for suggestions and by submitting problems and results for the benefit of others.

Other test problems (including interactive problems) and a brief introduction to stochastic programming appear on the Network-Enabled Optimization System (http://www.mcs.anl.gov/home/otc/Guide/) of the Optimization Technology Center jointly operated by Argonne National Laboratory and Northwestern University.

Acknowledgments

This work is supported by the National Science Foundation under Awards Number DDM-9215921 and DMI-9523275. The author thanks the referees for numerous suggestions that have greatly improved the quality of this article.

References

Variables Programming Problems, Numerische Mathematik 4, 238–252.
for Linear Programming, Linear Algebra and its Application 152, 191–222.
for the Unit Commitment Problem, IEEE Transactions on Power Systems 11, 1497–1508.


