The Dissertation Committee for Anukal Chiralaksanakul
Certifies that this is the approved version of the following dissertation:

Monte Carlo Methods for Multi-stage Stochastic Programs

Committee:

David Morton, Supervisor

John Hasenbein

Paul Jensen

Douglas Morrice

Elmira Popova

Monte Carlo Methods for Multi-stage Stochastic Programs

by

Anukal Chiralaksanakul, B.S., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2003

Dedicated to my parents and grandparents.

Acknowledgments

I am, and will always be, grateful to my advisor, professor David Morton, for introducing me to stochastic programming, and for his support and guidance throughout my doctoral study at the University of Texas at Austin. I have realized that his approach to stochastic programming is practically intuitive. It is the least to say that working with him is truly an enlightening and inspiring experience. His meticulous proofreading greatly improves readability of this dissertation.

I sincerely thank professor John Hasenbein, Paul Jensen, Douglas Morrice, and Elimira Popova for serving on my doctoral committee, and for their help throughout my study at the University. I also thank several graduate student colleagues, Guzin Bayraksan, Stefcho Dokov, Praveen Korapaty, Feng Pan, Yong Min Wang, and Xinhui Zhang, for insightful discussions on operations research and stochastic programming, from which this dissertation benefits.

Occasionally, my study and research seem to be insurmountable. Many friends and Laurel House mates deserve a lot of thanks for getting my through those times as well as for sharing good times and making my stay in Austin a pleasant one.

As it has been for any of my endeavor, my spirit is kept high through constant love and encouragement from my parents and grandparents. I cannot write and say enough words to express my gratitude for everything they have provided me.

Monte Carlo Methods for Multi-stage Stochastic Programs

Publication No. _____

Anukal Chiralaksanakul, Ph.D.
The University of Texas at Austin, 2003

Supervisor: David Morton

Stochastic programming is a natural and powerful extension of deterministic mathematical programming, and it is effectively utilized for analyzing optimization problems when the problem's parameters are not known with certainty. These uncertain parameters are treated as a random vector with a known distribution in the stochastic programming framework. Typically, the size of stochastic programming models is large due to the number of dimensions and realizations of the random vector. With recent advances in optimization algorithms and computing technology, an increasing number of realistically-sized two- and multi-stage stochastic programming models are being successfully formulated and solved. Despite these successes, multi-stage stochastic programs in which the random vector has a large number of dimensions and/or realizations (or is even continuous), still remain a computational challenge primarily because of the exponential growth of the model's size with respect to the number of stages. In this dissertation, we exploit special structures in order to attack these computationally difficult problems.

Our research can be broadly divided into three parts. First, we propose two Monte Carlo sampling-based solution methods for multi-stage stochastic programs.

Both methods exploit special structures for a particular class of multi-stage problems, and result in feasible solution policies. These policies have desirable asymptotic properties, but, of course, in practice are generated using finite scenario trees. As a result, in the second part of the dissertation, we develop Monte Carlo techniques to determine the quality of an arbitrary feasible policy. In particular, we build a statistically-based point estimate for a lower bound of the optimal objective function value for a minimization problem, and use it to construct a confidence interval on the solution's quality. In the third part, we aim to develop procedures to reduce the bias associated with the lower-bound estimator, thereby improving our ability to construct a reasonably tight confidence interval on the solution's quality. Towards this goal, we vary the number of descendants in the sample tree to reduce the bias in the context of American-style option pricing and stochastic lot sizing. All proposed methodologies are numerically tested on problems from the literature.

Table of Contents

List of Tables

List of Figures

Chapter 1

Research Overview

1.1 Background and Motivation

Deterministic mathematical programming has been successfully used for modeling and analyzing a wide variety of systems requiring optimization. One of its assumptions is that the parameters in the models are known with certainty. In many important applications, this assumption is quite restrictive and usually violated because of the inherent randomness in the system. For instance, in planning the capacity of a telecommunications network, the future demand pattern under which the system will operate is typically not known with certainty. In financial portfolio management, one chooses to invest in financial assets whose future returns are not yet realized. In these settings, an optimal network design or investment policy obtained via a deterministic mathematical program is not satisfying because it either completely disregards or does not capture well the underlying random effect. In other words, the deterministic model does not take possible future scenarios into account when specifying an optimal solution. Consequently, stochastic programming models are developed in order to directly incorporate uncertainty into a mathematical program, hence enabling its optimal solution to better hedge against possible future scenarios.

Stochastic programming was first introduced in 1950's. Specifically, stochastic linear programming with recourse was proposed by Beale [6] and Dantzig [32] and chance-constrained programming by Charnes and Cooper [27]. Since then, these two

types of models have developed into the two most widely-studied classes of stochastic programming models. An extensive treatment of chance-constrained models is provided by Prékopa [107]. This dissertation focuses on stochastic programming with recourse.

In a two-stage stochastic program with recourse, the uncertainty in the system is modeled by a random vector with known probability distribution. The sequence of making decisions and observing a realization of the random vector occurs as follows. The first stage decision is made before the uncertainty is revealed. Then, a realization of the random vector becomes known, and the second stage decision is made correspondingly. The first and second stage decisions may be subject to constraints, and there are costs associated with these decisions. The objective is to find a first stage decision such that the expected value of a specified performance measure is optimized. The term recourse arises because we adaptively take corrective actions via the second stage decision. In this respect, the recourse model captures the dynamics of the decision-making processes under uncertainty. An example of a two-stage stochastic program with recourse is the capacity planning problem for the telecommunications network mentioned earlier. There, the first stage decision is how much capacity to install on each link of the network with only probabilistic knowledge of the network's demand pattern, and the total installation cost must satisfy a budget constraint. After the installation, the realized demand is routed optimally in the network through the second stage decision. One possible goal is to minimize the expected number of unsatisfied demands for a given budget.

The two-stage stochastic program with recourse generalizes naturally to a multi-stage program to capture the time-dynamics of a more complex sequential decision-making process under uncertainty. In the multi-stage case, the uncertainty is modeled by a sequence of random vectors that forms a stochastic process, and

a decision is made alternately with observing a realization of a random vector. In particular, the decision at the current stage is made before observing any of the future realizations, and must only depend on the known decisions and observations of the random vectors from previous stages. Then, a realization of the next stage's random vector is observed, and the next stage's decision is made and so on. The decision in each stage can be subject to constraints. Again, the goal is to find a sequence of decisions so that the expected value of a specified performance measure is optimized. This dissertation is primarily concerned with the development of solution methods, in particular approximation methods, for multi-stage stochastic programs.

An example of a multi-stage stochastic program with recourse is a discrete-time asset-allocation problem arising in financial portfolio management. In such a problem, the goal is to find an investment policy that maximizes the expected utility of wealth at the end of the horizon. A portfolio of assets is formed at the beginning of the first stage when only probabilistic knowledge of each asset's future return is available. Then, the portfolio is subsequently re-balanced at the beginning of every stage after each asset's return for that stage becomes known. At the beginning of the last stage, all assets in the portfolio are sold. Whenever the portfolio is re-balanced, asset balance and possibly some other trading constraints must be satisfied. In fact, managing financial systems, including asset-allocation and asset-liability management, is one of the most popular applications of multi-stage stochastic programming today. Specific applications work in finance includes [11, 17, 21, 22, 23, 30, 68, 69, 70, 96, 97]. For an overview of such systems and additional references see, e.g., [129].

Many other important real-world applications that involve sequential decision making under uncertainty, also fit very well in the modeling framework of a

multi-stage stochastic program. In the electric power industry, multi-stage stochastic programs arise in managing hydro-thermal systems [74, 101, 102] and in unit commitment problems [61, 104, 116, 117]. In addition, there is growing interest in combining financial and electric-power systems via stochastic programming, e.g., [53]. Katok et al. [80, 81] solve, via a heuristic, a multi-stage stochastic program for evaluating the benefits of flexibility in a production system. Other multi-stage models motivated by production systems include stochastic lot-sizing models [64, 89], related capacity planning models [1, 2, 49], and melt-control in the steel industry [42]. Other applications of multi-stage stochastic programming include work in vehicle allocation [29, 39, 54, 106], forestry [56], and military logistics [94]. Further applications can be found in Birge and Louveaux [16], Dupačová [45], Dupačová et al. [41], Kall and Wallace [78], King [83], and Prékopa [107].

The size of a stochastic programming model depends on the dimension, and the number of the realizations, of the model's random vector. When the underlying random vectors are of high dimension and have a continuous distribution, or finite support with many realizations, it is usually impossible to evaluate the expected total cost exactly, even for a fixed first stage decision. This is true for one- and two-stage stochastic programs. Computational difficulties are further compounded in the multi-stage setting, in which the problem size grows exponentially with the number of stages. Thus, solving a multi-stage stochastic program with a large number of scenarios and a moderate-to-large number of stages can be a computationally challenging task, despite the recent advances in optimization algorithms and computing technology. We develop practical approximate solution methods for this class of models and also test their computational viability in this dissertation.

Instead of solving a stochastic program, one could replace the random vector with a deterministic vector such as its mean and regard an optimal solution of the

resulting deterministic model as an approximate solution to the original problem. Birge and Louveaux [16, §1.2] give examples showing that this type of strategy can lead to a very poor solution. Wallace [121] argues that sensitivity analysis of mathematical programs and parametric programming are not appropriate paradigms for decision making under uncertainty because they do not explicitly recognize uncertainty as an element of the decision problem and completely ignore the timing of decisions, e.g., the first stage decision must be made prior to observing a realization of the random vector. Examples are provided in [121] to demonstrate that an optimal solution to a two-stage stochastic linear program with recourse cannot be constructed from optimal solutions of the deterministic models resulting from replacing the random vector with each of its realizations, and that a solution constructed in such a way can perform poorly. In addition, these examples show that there does not exist, in general, a deterministic vector such that when it replaces the random vector in the two-stage recourse model, an optimal solution to the original stochastic model can be recovered.

The subject of optimizing under uncertainty has been extensively studied for many years in areas other than stochastic programming, including statistical decision theory, stochastic optimal control theory, stochastic dynamic programming, Markov decision processes, and robust optimization. To a large extent, the theory and methods of these areas have developed independently of those of stochastic programming. While they are related, we do not discuss these other methods explicitly in this dissertation except that we apply stochastic dynamic programming to solve a special class of multi-stage stochastic optimization problems in Chapter 4. Birge and Louveaux [16, §2.8] discuss differences and similarities of these other approaches and stochastic programming while Dupačová and Sladký [43] provide a comparison between discrete-time stochastic dynamic programming and multi-stage stochastic

programming with recourse.

1.2 Research Objectives

The objectives of this research are three-fold: (i) to develop methods to solve multi-stage stochastic programs with a large number of scenarios and a moderate-to-large number of stages, (ii) to develop an effective method to determine the quality of an arbitrary feasible solution to such problems, and (iii) to demonstrate the computational viability of our methodologies.

A solution to a multi-stage stochastic program is a policy that specifies what decision to make at stage t, given the history of realizations of the random vectors up to that stage. So, achieving goal (i) involves systematic ways to construct good policies. We develop two Monte Carlo-based approaches for constructing such policies depending on the underlying problem structure. These two procedures generate feasible policies for multi-stage stochastic programs and hence can be viewed as solution methods for this class of problems.

When policy construction is rooted in Monte Carlo sampling, one way of justifying such an approach is to show that it is consistent, i.e., the policies are asymptotically optimal as the number of samples grows large. However, in practice, our polices are constructed without having the number of samples growing to infinity, and thus they are only feasible. In order to determine their quality, we develop Monte Carlo-based techniques to estimate the quality of an arbitrary policy, and this is what we pursue in goal (ii). Achieving goal (ii) involves building point estimates and confidence intervals for the objective function value (e.g., the cost) when using a feasible policy, and for z^* or a lower bound on z^* (assuming we are minimizing), where z^* denotes the optimal objective function value. These confidence intervals are combined to construct confidence intervals on the optimality gap of the feasible

policy. For the techniques in goal (ii) to be effective, we want the confidence interval's width to be as small as possible for a specified computational budget. As as a result, we apply a variance reduction technique known as common random numbers when the confidence interval is constructed, and also develop techniques to reduce the bias associated with the lower bound estimator of z^*. Towards goal (iii), a collection of problems from the literature are used to test our methodologies.

1.3 Dissertation Organization

Chapter 2 establishes notation and provides a review of stochastic programming. Two-stage stochastic programming with recourse is introduced in Section 2.2, and its extensions to multi-stage stochastic programming are detailed in Section 2.3. Solution methods for stochastic programs are reviewed in Section 2.4. Technical tools for establishing convergence of Monte Carlo approximation are presented in Section 2.5.

Chapter 3 develops policy generation methods and procedures to establish the quality of a policy for a multi-stage stochastic program. We begin by describing a procedure to construct a sample scenario tree in Section 3.2. Then, we develop two Monte Carlo-based methods for two classes of multi-stage stochastic program with recourse in Section 3.3. Procedures to estimate the cost of using a policy are discussed in Section 3.4. We then develop a lower-bound estimator in Section 3.5, and show how a confidence interval on the optimality gap can be constructed from the policy-cost and lower-bound estimators in order to establish the quality of a policy in Section 3.6. We present computational results of these procedures in Section 3.7.

Chapter 4 concerns reducing the bias of the lower-bound estimator developed in Chapter 3. We begin by developing in Section 4.2 a procedure for building

non-uniform sample trees, i.e., sample trees with varying number of descendants, in the context of pricing an American-style option. In Section 4.3, we extend the methodology to build non-uniform sample trees for the stochastic lot-sizing problem. We report computational results of the sample tree building procedures in Section 4.4.

Chapter 5 concludes the dissertation. We review our research objectives, summarize the sampling-based procedures we develop, and provides future research directions.

Chapter 2

Stochastic Programming Review

2.1 Introduction

This chapter reviews concepts and the literature of stochastic programming related to our proposed methodologies, and establishes notation for subsequent chapters. The review of basic concepts is not intended to be comprehensive. We refer the reader to the books by Birge and Louveaux [16], Dupačová et al. [41], Kall and Wallace [78], and Prékopa [107], and references therein for a more thorough review of the subject.

Let $(\Omega, \mathcal{F}, P)$ be the probability space on which the random vector $\tilde{\xi}$ is defined. A general formulation (see, e.g., Ermoliev and Wets [51]) for a stochastic program is

$$\begin{aligned} \min_{x} \quad & E\,\psi_0(x, \tilde{\xi}) \\ \text{s.t.} \quad & E\,\psi_i(x, \tilde{\xi}) \leq 0,\ i = 1, \ldots, m \\ & x \in X \subseteq \mathbb{R}^d, \end{aligned} \tag{2.1}$$

where E denotes the mathematical expectation operator, ψ_i denotes a mapping $\psi_i : X \times \Xi \to \mathbb{R}, i = 0, \ldots, m$, and Ξ is the support of $\tilde{\xi}$. The general formulation in (2.1) covers a wide range of stochastic programs, including the chance-constrained and the recourse models. We assume that the probability distribution P of the random vector is known and does not depend on x. Dupačová [44] surveys applications of stochastic programming under incomplete knowledge of the distribution P. In our

notation, we put a tilde over an entity to denote a random element, and omit it when refering to its realization. Throughout the dissertation, we use the terms cost and objective function interchangeably to refer to ψ_0.

To ensure that (2.1) is well-defined, we assume that (i) both the expectation in the objective function and constraints are finite for all $x \in X$, (ii) the feasible region of (2.1) is non-empty, and (iii) a minimizer of (2.1) is achieved on its feasible region. When we specialize the general formulation to a multi-stage stochastic program with recourse in Section 2.3, we will provide sufficient conditions to ensure these assumptions.

It will become clear that many terminologies and concepts of a two-stage recourse model can be extended in a straightforward fashion to a multi-stage recourse model. To ease the understanding of the concepts and notation, we therefore begin by considering a two-stage model in Section 2.2. Then, in Section 2.3, we extend the discussion to a multi-stage model, which is of main interest in this dissertation. We turn to solution methods in Section 2.4, and review technical tools needed for establishing convergence properties of Monte Carlo methods in Section 2.5.

2.2 Two-stage Recourse Models

A two-stage stochastic program with recourse can be formulated as

$$\begin{aligned} \min_{x_1} \ & E\phi_1(x_1, \tilde{\xi}_2) \\ \text{s.t.} \ & x_1 \in X_1, \end{aligned} \tag{2.2}$$

where

$$\begin{aligned} \phi_1(x_1, \xi_2) = \min_{x_2} \ & \phi_2(x_1, x_2, \xi_2) \\ \text{s.t.} \ & x_2 \in X_2(x_1, \xi_2). \end{aligned} \tag{2.3}$$

Model (2.2)-(2.3) capture the type of dynamics that arises in many real-world decision-making processes. In particular, the first stage decision x_1 must be made before the realization of the random vector $\tilde{\xi}_2$ is known. After a realization of the random vector $\tilde{\xi}_2$ is observed, an adaptive or recourse decision x_2 is then made. The associated cost of decisions x_1 and x_2 under realization ξ_2 is given by $\phi_2(x_1, x_2, \xi_2)$. The requirement that the decision x_1 be made with only distributional knowledge of the random vector $\tilde{\xi}_2$ is known in the stochastic programming literature as *nonanticipativity*. The two-stage stochastic program with recourse is a special case of (2.1) in which constraints involving $\psi_i, i = 1, \ldots, m$, are not present, $X = X_1$, and $\psi_0 = \phi_1$, defined by the second-stage program (2.3).

The first stage constraints of (2.2) are fixed and do not explicitly depend on the random vector $\tilde{\xi}_2$. However, the random vector $\tilde{\xi}_2$ can constrain x_1 through so-called *induced constraints* because not all decisions x_1 lead to a feasible second stage program under a certain realization of $\tilde{\xi}_2$. Let K be the set of induced constraints and adopt the following definition: $K = \{x_1 \in \mathbb{R}^{d_1} : \exists\, x_2 \text{ such that } x_2 \in X_2(x_1, \tilde{\xi}_2), \text{ wp1}\}$. Two alternative equivalent definitions of K are presented by Wets [123]. A two-stage stochastic program is said to be infeasible if $X_1 \cap K = \emptyset$. If $K = \mathbb{R}^{d_1}$, then the program is said to have *complete recourse*. This means that any $x_1 \in \mathbb{R}^{d_1}$ will yield a feasible second stage program. A weaker assumption, known as *relatively complete recourse*, is only concerned with decision $x_1 \in X_1$, and can be stated as $X_1 \cap K = X_1$. In other words, for each $x_1 \in X_1$, there exists a feasible solution x_2 to the second stage program (2.5), wp1. By using penalty-based formulations to capture infeasibility, most real-world problems can be formulated so that they have relatively complete recourse.

Relatively complete recourse is a desirable, and arguably necessary, property for the solution techniques based on Monte Carlo sampling we develop in Chapter 3.

To approximately solve (2.2), we can, for instance, sample n observations of $\tilde{\xi}_2$ and solve the resulting n-scenario version of (2.2). The associated solution may be viewed as an estimate of an optimal solution to (2.2). If the original model (2.2) does not have relatively complete recourse, the approximate optimal solution is not necessarily feasible to the original problem since the approximating problem contains only a subset of scenarios from the sample space. For similar reasons, we may have that $X_1 \cap K = \emptyset$ for the true problem while the n-scenario approximating problem may be feasible. Thus, without the assumption of relatively complete recourse, it may be difficult to justify this type of approximation scheme.

To have (2.2)-(2.3) well-defined (i.e., its optimal objective function value is finite), it is sufficient to assume that (2.2)-(2.3) have relatively complete recourse, and that ϕ_1, ϕ_2, X_1, X_2, and the distribution of $\tilde{\xi}_2$ satisfy certain properties. We defer the discussion of these properties to Section 2.3.1. There, we give sufficient conditions for multi-stage models that include (2.2)-(2.3) as a two-stage special case.

The realization of $\tilde{\xi}_2$ can be viewed as a scenario tree that contains only the root node in the first stage (defined by the first stage's deterministic parameters), and leaf nodes in the second stage, each of which corresponds to a realization of $\tilde{\xi}_2$. Since scenario trees play a lesser role in the two-stage than in the multi-stage setting, we will discuss in detail the notion of scenario tree in the context of multi-stage models in Section 2.3.2.

An important special case of (2.2)-(2.3) is a two-stage stochastic linear program with recourse, which can be stated as

$$\begin{aligned} \min_{x_1} \ & c_1 x_1 + Eh(x_1, \tilde{\xi}_2) \\ \text{s.t.} \ & A_1 x_1 = b_1 \\ & x_1 \geq 0, \end{aligned} \tag{2.4}$$

where

$$\begin{aligned} h(x_1, \xi_2) = \min_{x_2} \ & c_2 x_2 \\ \text{s.t.} \ & A_2 x_2 = b_2 - B_2 x_1 \\ & x_2 \geq 0, \end{aligned} \tag{2.5}$$

and $\tilde{\xi}_2$ is the vector of the random variables in $\tilde{A}_2, \tilde{B}_2, \tilde{b}_2$, and $\tilde{c}_2$. The dimensions of vectors and matrices are as follows: $A_1 \in \mathbb{R}^{m_1 \times d_1}, b_1 \in \mathbb{R}^{m_1}, c_1 \in \mathbb{R}^{1 \times d_1}, A_2 \in \mathbb{R}^{m_2 \times d_2}, B_2 \in \mathbb{R}^{m_2 \times d_1}, b_2 \in \mathbb{R}^{m_2}$, and $c_2 \in \mathbb{R}^{1 \times d_2}$. We define $h(x_1, \xi_2) = +\infty$ if the second stage program is infeasible for the given x_1 and ξ_2 combination. In this linear special case of (2.2), X_1 is $\{x \in \mathbb{R}^{d_1} : A_1 x_1 = b_1, x_1 \geq 0\}$, and ϕ_1 separates into the first stage cost, $c_1 x_1$, and the stochastic second stage cost, $h(x_1, \tilde{\xi}_2)$, defined by the second-stage linear program (2.5). Again, we defer the discussion of sufficient conditions under which (2.4)-(2.5) are well-defined to Section 2.3.1 where we discuss a multi-stage linear model.

In the stochastic programming literature, $Eh(x_1, \tilde{\xi}_2)$ is called the *recourse function*, and $\tilde{A}_2$ is called *recourse matrix*. A two-stage stochastic linear program is said to have *fixed recourse* if the matrix $\tilde{A}_2$ is non-stochastic, and is said to have *simple recourse* if the second stage decision vector x_2 is solely used to capture the magnitude of constraint violation in (2.5). Specifically, let $x_2 = (x_2^+, x_2^-), c_2 = (c_2^+, c_2^-)$, and $A_2 = (I, -I)$ where I is the identity matrix. By substituting these in (2.5), we obtain

$$\begin{aligned} h(x_1, \xi_2) = \min_{x_2^+, x_2^-} \ & c_2^+ x_2^+ + c_2^- x_2^- \\ \text{s.t.} \ & I x_2^+ - I x_2^- = b_2 - B_2 x_1 \\ & x_2^+, x_2^- \geq 0, \end{aligned} \tag{2.6}$$

where $\tilde{\xi}_2$ is the vector of random variables in $\tilde{B}_2$ and $\tilde{b}_2$. We assume $c_2^+ + c_2^- > 0$ so that (2.6) is bounded. The simple recourse model, defined by (2.4) and (2.6),

has complete and fixed recourse and is a static optimization model. The aircraft allocation under uncertain demand problem of Ferguson and Dantzig [52] is an early example of a two-stage stochastic linear program with simple recourse. More examples and analyses of simple recourse models are given by Ziemba [128].

Assuming that stochastic program (2.4)-(2.5) has relatively complete recourse, $\{x_1 \in \mathbb{R}^{d_1} : A_1 x_1 = b_1, x_1 \geq 0\}$ is not empty, and $\tilde{\xi}_2$ has finite support, we can express (2.4)-(2.5) as a large scale linear program known as the *deterministic equivalent program*:

$$\begin{aligned} \min_{x_1, x_2^i} \quad & c_1 x_1 + \sum_{i=1}^{n} p^i c_2^i x_2^i \\ \text{s.t.} \quad & A_1 x_1 = b_1 \\ & A_2^i x_2^i + B_2^i x_1 = b_2^i, i = 1, \ldots, n \\ & x_1 \geq 0 \\ & x_2^i \geq 0, i = 1, \ldots, n, \end{aligned} \tag{2.7}$$

where superscript i on an entity in (2.7) denotes its value under realization ξ_2^i, and p^i is the probability mass of ξ_2^i, i.e., $p^i = P(\tilde{\xi}_2 = \xi_2^i)$. The second stage decision x_2 adapts with the realizations of $\tilde{\xi}_2$. The nonanticipativity constraints on x_1 is implicit in (2.7), i.e., linear program (2.7) allows only one decision x_1 for every realization of $\tilde{\xi}_2$. We can express these nonanticipativity constraints explicitly for (2.7) by duplicating x_1 for each realization of the random vector, and requiring, for example, that $x_1^i = x_1^{i+1}$, $i = 1, \ldots, n-1$.

In the next section, we describe a generalization of (2.2)-(2.3) to the multi-stage setting in which decisions and observations of the random vectors are made alternately over time, and then discuss the notion of scenario tree and deterministic equivalent formulations of multi-stage recourse models under finite support assumption.

2.3 Multi-stage Recourse Models

2.3.1 Problem Statement

We consider a T-stage stochastic program in which a sequence of decisions, $\{x_t\}_{t=1}^T$, is made with respect to a stochastic process $\{\tilde{\xi}_t\}_{t=1}^T$ as follows: at stage t, the decision $x_t \in \mathbb{R}^{d_t}$ is made with only the knowledge of past decisions, $x_1, \ldots, x_{t-1}$, and of realized random vectors, $\xi_1, \ldots, \xi_t$, such that the conditional expected value of an objective function, $\phi_t(x_1, \ldots, x_t, \tilde{\xi}_1, \ldots, \tilde{\xi}_{t+1})$, given the history, $\xi_1, \ldots, \xi_t$, is minimized. Decision x_t is subject to constraints that may depend on $x_1, \ldots, x_{t-1}$, and $\xi_1, \ldots, \xi_t$. We assume that $\tilde{\xi}_1$ is a degenerate random vector that takes value ξ_1 with probability one, and that the probability law governing the evolution of $\{\tilde{\xi}_t\}_{t=1}^T$ is known and does not depend on $\{x_t\}_{t=1}^T$. Adopting the notation used by Frauendorfer [55] and others, we use a superscript t on an entity to denote its history through stage t; for example, $\xi^t = (\xi_1, \ldots, \xi_t)$ and $x^t = (x_1, \ldots, x_t)$. Let Ξ_t be the support of $\tilde{\xi}_t$ and Ξ^t be that of $\tilde{\xi}^t$ for $t = 1, \ldots, T$. The conditional distribution of $\tilde{\xi}_{t+1}$ given $\tilde{\xi}^t = \xi^t$ is denoted $F_{t+1}(\xi_{t+1}|\xi^t)$. A T-stage stochastic program can be stated as

$$\begin{aligned} \min_{x_1} \;& E[\phi_1(x_1, \tilde{\xi}^2)|\xi^1] \\ \text{s.t.} \;& x_1 \in X_1(\xi^1), \end{aligned} \tag{2.8}$$

where

$$\begin{aligned} \phi_{t-1}(x^{t-1}, \xi^t) = \min_{x_t} \;& E[\phi_t(x^{t-1}, x_t, \tilde{\xi}^{t+1})|\tilde{\xi}^t = \xi^t] \\ \text{s.t.} \;& x_t \in X_t(x^{t-1}, \xi^t), \end{aligned} \tag{2.9}$$

for $t = 2, \ldots, T-1$, and

$$\begin{aligned} \phi_{T-1}(x^{T-1}, \xi^T) = \min_{x_T} \;& \phi_T(x^{T-1}, x_T, \xi^T) \\ \text{s.t.} \;& x_T \in X_T(x^{T-1}, \xi^T). \end{aligned} \tag{2.10}$$

We take $\phi_t, t = 1, \ldots, T$, to be real-valued functions. The constraint set X_t depends on the history of both x^{t-1} and $\tilde{\xi}^t$. Note that (2.8)-(2.10) is a special case of the general formulation (2.1) where constraints involving $\psi_i, i = 1, \ldots, m$, are not present, and ψ_0 is defined as a nested optimization problem. When $T = 2$, (2.8)-(2.10) reduce to the general two-stage stochastic program with recourse defined by (2.2)-(2.3). Relatively complete recourse of a T-stage stochastic program means that for any feasible sequence of decisions from stage one to stage t, x^t, there exists a sequence of feasible decisions, $x_{t+1}, \ldots, x_T$, wp1.

An optimal solution of (2.8)-(2.10) is specified by an optimal *policy*. A policy may be viewed as a mapping, $x_t(\xi^t)$, with domain Ξ^t and range in $\mathbb{R}^{d_t}$, $t = 1, \ldots, T$. Restated, a policy is a rule which specifies what decision to take at each stage t of a multi-stage stochastic program for each possible realization of $\tilde{\xi}^t$ in $\Xi^t, t = 1, \ldots, T$. We only consider policies that satisfy the nonanticipativity requirement, i.e., stage t decision, x_t, can only depend on ξ^t and not on subsequent realizations of the random vectors. Therefore, a policy $\hat{x}^T(\tilde{\xi}^T) = (\hat{x}_1(\tilde{\xi}^1), \ldots, \hat{x}_T(\tilde{\xi}^T))$ is said to be feasible if it is nonanticipative, $\hat{x}_1(\tilde{\xi}^1) \in X_1(\tilde{\xi}^1)$, and $\hat{x}_t(\tilde{\xi}^t) \in X_t(\hat{x}^{t-1}(\tilde{\xi}^{t-1}), \tilde{\xi}^t)$, wp1, where $\tilde{\xi}^t = (\tilde{\xi}^{t-1}, \tilde{\xi}_t), t = 2, \ldots, T$.

In order to have (2.8)-(2.10) well-defined, we make the following assumptions:

(A1) (2.8)-(2.10) has relatively complete recourse, and $X_1(\xi^1)$ is non-empty.

(A2) $X_1(\xi^1)$ is compact, and for all feasible x^{t-1}, $X_t(x^{t-1}, \tilde{\xi}^t)$ is compact, wp1, $t = 2, \ldots, T$.

(A3) $\phi_T(x^T, \tilde{\xi}^T)$ is lower semi-continuous in x^T, wp1.

(A4) $E\phi_T^2(x^T, \tilde{\xi}^T) < \infty$ for all feasible x^T.

Recall that a real-valued function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be lower semi-continuous at $x \in \mathbb{R}^d$ if

$$f(x) \leq \liminf_{\nu \to \infty} f(x_\nu)$$

for every sequence $\{x_\nu\}$ converging to x. Feasibility of (2.8)-(2.10) is guaranteed by (A1). Compactness of $X_1(\xi^1) \neq \emptyset$, coupled with lower semi-continuity of $E[\phi_1(x_1, \tilde{\xi}^2)|\xi^1]$, ensures that its infimum is achieved on $X_1(\xi^1)$. These conditions on $X_1(\xi^1)$ are included in (A1) and (A2). Lower semi-continuity of $E[\phi_1(x_1, \tilde{\xi}^2)|\xi^1]$ results from:

(i) Compactness of $X_t(x^{t-1}, \tilde{\xi}^t)$ and lower semi-continuity of $E[\phi_t(x^{t-1}, x_t, \tilde{\xi}^{t+1})|\tilde{\xi}^t]$ in x^{t-1} and x_t, wp1, ensure lower semi-continuity of $\phi_{t-1}(x^{t-1}, \tilde{\xi}^t)$, wp1. (See Rockafellar and Wets [109, Theorem 1.17].)

(ii) Lower semi-continuity of $\phi_{t-1}(x^{t-1}, \tilde{\xi}^t)$ and $E\big[\big|\phi_{t-1}(x^{t-1}, \tilde{\xi}^t)\big||\tilde{\xi}^{t-1}\big] < \infty$, wp1, ensure lower semi-continuity of $E[\phi_{t-1}(x^{t-1}, \tilde{\xi}^t)|\tilde{\xi}^{t-1}]$, wp1. (See Wets [126, Proposition 2.2].)

The lower semi-continuity assumption of (A3) provides the base case in the induction argument from $t = T$ to show lower semi-continuity of $E[\phi_1(x_1, \tilde{\xi}^2)|\tilde{\xi}^1]$ via (i) and (ii). The preservation of lower semi-continuity under the expectation in (ii) uses the finite expectation hypothesis, i.e., $E\big[\big|\phi_t(x^t, \tilde{\xi}^{t+1})\big||\tilde{\xi}^t\big] < \infty$ for all feasible x^t, wp1, for $t = 1, \ldots, T-1$, which follows from (A4). The stronger assumption of continuity in place of (A3) is a natural assumption for multi-stage stochastic linear programs, but lower semi-continuity can arise when considering integer-constrained problems. As it will become apparent, the finite second moment assumption in (A4) also allows the use of the central limit theorem in the construction of a confidence interval.

For ease of exposition, we implicitly incorporate the constraint set in the objective function by using an extended-real-valued representation as follows.

$$f_t(x^t, \xi^{t+1}) = \begin{cases} \phi_t(x^t, \xi^{t+1}) & \text{if } x_t \in X_t(x^{t-1}, \xi^t) \\ \infty & \text{otherwise} \end{cases} \tag{2.11}$$

for $t = 1, \ldots, T-1$, and

$$f_T(x^T, \xi^T) = \begin{cases} \phi_T(x^T, \xi^T) & \text{if } x_T \in X_T(x^{T-1}, \xi^T) \\ \infty & \text{otherwise.} \end{cases} \tag{2.12}$$

The T-stage stochastic program defined by (2.8)-(2.10) can now be re-stated as an unconstrained optimization problem:

$$z^* = \min_{x_1} \; E[f_1(x_1, \tilde{\xi}^2)|\xi^1] \tag{2.13}$$

where

$$f_{t-1}(x^{t-1}, \xi^t) = \min_{x_t} \; E[f_t(x^{t-1}, x_t, \tilde{\xi}^{t+1})|\tilde{\xi}^t = \xi^t], \tag{2.14}$$

for $t = 2, \ldots, T-1$, and

$$f_{T-1}(x^{T-1}, \xi^T) = \min_{x_T} \; f_T(x^{T-1}, x_T, \xi^T). \tag{2.15}$$

This extended-real-valued formulation will be used in our development throughout the dissertation.

An important special case of (2.8)-(2.10) is a multi-stage stochastic program with recourse in which the objective function has an additive contribution from each stage and the underlying optimization problems are linear programs. A T-stage stochastic linear program with recourse can be expressed in the following form:

$$\begin{aligned} \min_{x_1} \; & c_1 x_1 + E[h_1(x_1, \tilde{\xi}^2)|\xi^1] \\ \text{s.t.} \; & A_1 x_1 = b_1 \\ & x_1 \geq 0, \end{aligned} \tag{2.16}$$

where, for $t = 2, \ldots, T$,

$$\begin{aligned} h_{t-1}(x_{t-1}, \xi^t) = \min_{x_t} \ & c_t x_t + E[h_t(x_t, \tilde{\xi}^{t+1}) | \tilde{\xi}^t = \xi^t] \\ \text{s.t.} \ & A_t x_t = b_t - B_t x_{t-1} \\ & x_t \geq 0, \end{aligned} \tag{2.17}$$

and $h_T = 0$. The components of the random vector $\tilde{\xi}_t$ consist of the random elements from $\tilde{A}_t, \tilde{B}_t, \tilde{b}_t$, and $\tilde{c}_t$. The dimension of vectors and matrices are as follows: $c_t \in \mathbb{R}^{1 \times d_t}, A_t \in \mathbb{R}^{m_t \times d_t}, B_t \in \mathbb{R}^{m_t \times d_{t-1}}$, and $b_t \in \mathbb{R}^{m_t}, t = 1, \ldots, T$. We now return to assumptions (A1)-(A4) and describe sufficient conditions in linear programming context to ensure (A1)-(A4). Relatively complete recourse carries over naturally to the constraints of (2.16) and (2.17) and is assumed to hold. We assume that the feasible region of (2.16) is nonempty and bounded and that of (2.17) is bounded for all feasible x_{t-1}, wp1; hence, (A1) and (A2) hold. Continuity of $h_t(x_t, \tilde{\xi}^t)$ in x_t, wp1, results from a basic linear programming property, i.e., the optimal objective function value is a piecewise linear convex function of the right-hand-side vector in the constraints (see, e.g., Bazaraa, Jarvis, and Sherali [5, §6.3]); hence, (A3) holds. Finally, we assume that the distribution of $\tilde{\xi}^T$ is such that (A4) holds. When $T = 2$, the formulation of the T-stage stochastic linear program above reduces to that of the two-stage stochastic linear program given by (2.4)-(2.5).

In a T-stage stochastic linear program, the stage t recourse function depends on the history ξ^t since it is defined in terms of a conditional expectation (except for $t = 1$ where $\tilde{\xi}^1$ is known with probability one). Solution methods of such a model must take this dependency into account. If the stochastic process $\{\tilde{\xi}\}_{t=1}^T$ is *interstage independent*, i.e., $\tilde{\xi}_t$ and $\tilde{\xi}_{t'}$ are independent for any $t, t' = 1, \ldots, T$ and $t \neq t'$, then all the conditional expectations in a T-stage stochastic linear program become unconditional ones. Thus, the stage t recourse function does not depend on

the history ξ^t under interstage independence of $\{\tilde{\xi}\}_{t=1}^T$. One of the solution methods we develop in Chapter 3 is designed to exploit this interstage independence structure.

2.3.2 Scenario Tree

Realizations of $\{\tilde{\xi}_t\}_{t=1}^T$ form a scenario tree that represents all the possible ways that $\{\tilde{\xi}_t\}_{t=1}^T$ can evolve, and organizes into a tree the realizations of the sequence $\{\tilde{\xi}_t\}_{t=1}^T$ that have common histories up to stage t. Scenario trees play a key role in multi-stage stochastic programs. So, we introduce their notation here. From a computational perspective, we limit ourselves to finite scenario trees, i.e., trees arising from $\{\tilde{\xi}_t\}_{t=1}^T$ whose supports Ξ_t are finite, $t = 1, \ldots, T$, so that realizations of $\tilde{\xi}^t$ can be enumerated as $\xi^{t,1}, \ldots, \xi^{t,n_t}$.

With this notation, a scenario tree has a total of n_T leaf nodes, one for each scenario $\xi^{T,i}, i = 1, \ldots, n_T$. Two scenarios $\xi^{T,i}$ and $\xi^{T,j}, i \neq j$, may be identical up to stage t. The number of distinct realizations of $\tilde{\xi}^T$ up to stage t is denoted by n_t so that the scenario tree has a total of n_t nodes at stage t, corresponding to each $\xi^{t,i}, i = 1, \ldots, n_t$. The unique node in the first stage is called the root node. For a given node, there is a unique scenario subtree, which is itself a tree rooted at that node and represents all possible evolutions of $\{\tilde{\xi}_{t'}\}_{t'=t}^T$ given the history ξ^t. We denote this subtree $\Gamma(\xi^t)$. Note that $\Gamma(\xi^1)$ is the entire scenario tree and the subtree of a leaf node is simply the leaf node itself, i.e., $\Gamma(\xi^T) = \xi^T$.

Consider a particular node i in stage $t < T$ with history $\xi^{t,i}$. Let $n(t,i)$ denote the number of stage $t+1$ descendant nodes of node i. These descendant nodes correspond to realizations $\xi^{t+1,j}$ where j is in the index set $D_t^i = \{k+1, \ldots, k+n(t,i)\}$, and

$$k = \sum_{r=1}^{i-1} n(t,r), \tag{2.18}$$

and $\sum_{r=1}^{0} \equiv 0$. The subvector of $\xi^{t+1,j}, j \in D_t^i$, that corresponds to the stage $t+1$ realization is $\xi_{t+1}^j, j \in D_t^i$. The ancestor of $\xi^{t,i}$ is denoted by $\xi^{t-1,a(i)}$. In this case, $a(i)$ is an integer between 1 and n_{t-1}. With our notation, $a(j) = i, \forall j \in D_t^i$. The ancestor operator $a(i) = a_t(i)$ has a stage dependency that we suppress for notational simplicity. The realization ξ^t can be expressed in terms of the original random vectors and matrices. Suppose $\xi^t = \text{vec}(A^t)$ in the stochastic linear program (2.16)-(2.17), then

$$\begin{aligned} \xi^{t,i} &= \text{vec}(A^{t,i}) \\ &= \text{vec}(A_1^{a^{t-1}(i)}, \ldots, A_t^i) \end{aligned}$$

where the notation $a^k(\cdot)$ means that the ancestor operator $a(\cdot)$ is applied k times. A_t^i is the realization of $\tilde{A}_t$ corresponding to node ξ_t^i in the scenario tree. The total number of nodes in each stage can be recursively computed from

$$n_t = \sum_{r=1}^{n_{t-1}} n(t-1, r), \quad \text{for } t = 2, 3, \ldots, T \tag{2.19}$$

where $n_1 \equiv 1$. Note that $D_t^i \bigcap D_t^{i'} = \emptyset$ for $i, i' \in \{1, \ldots, n_t\}$ and $i \neq i'$, and $\bigcup_{i=1}^{n_{t-1}} D_t^i = \{1, \ldots, n_t\}$ for $t = 1, \ldots, T-1$.

In Chapter 3, we will represent the conditional expectation given the history of $\{\tilde{\xi}_t\}_{t=1}^T$ at a *generic* stage t node in our statistical lower bound development. To facilitate this, we denote the number of immediate descendants of a generic stage t node, ξ^t, by $n(t) = |D_t|$, where D_t is the index set of the stage $t+1$ descendants of node ξ^t. In addition, $\xi_{t+1}^j, j \in D_t$ refers to the subvector of stage $t+1$ realizations of a generic stage t node ξ^t.

We illustrate our scenario tree notation by applying it to the four-stage scenario tree in Figure 2.1. The root node R corresponds to the unique stage 1

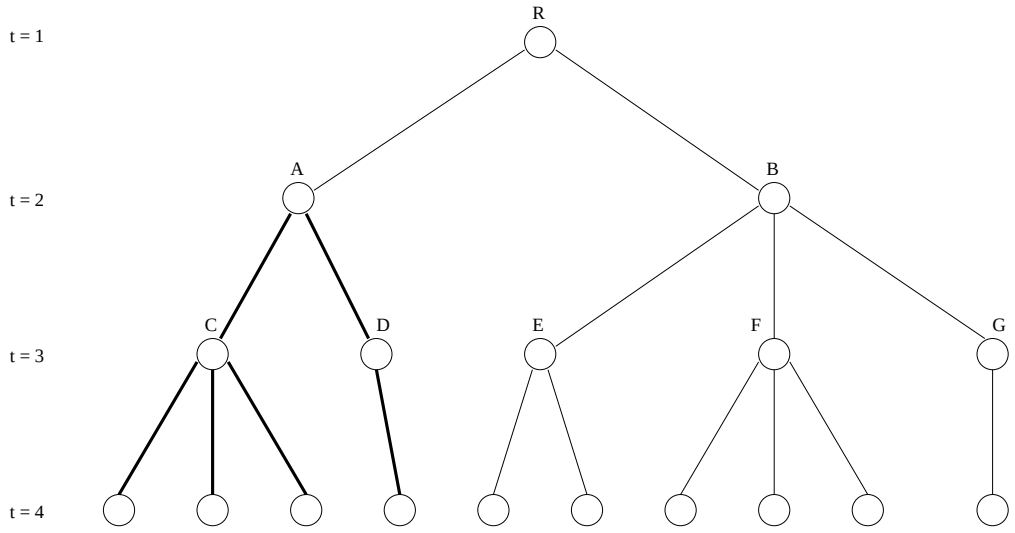

Figure 2.1: An example of a four-stage scenario tree.

realization ξ^1. Table 2.1 gives examples of the history notation and the number of immediate descendants for nodes $\mathrm{A}, \ldots, \mathrm{G}$. The subtree with its root at node A is represented by $\Gamma(\xi^{2,1})$. The branches of that subtree are darkened in Figure 2.1. The index set of the immediate descendants of node B is $D_2^2 = \{3, 4, 5\}$, and the corresponding stage 3 realizations are ξ_3^3, ξ_3^4, and ξ_3^5. Using (2.19), we obtain $n_2 = n(1,1) = 2$ and $n_3 = \sum_{r=1}^{2} n(2,r) = 2 + 3 = 5$. We refer to a generic node in the second stage, either A or B, by ξ^2, and a generic subtree rooted at ξ^2 by $\Gamma(\xi^2)$.

	A	B	C	D	E	F	G
$\xi^{t,i}$	$\xi^{2,1}$	$\xi^{2,2}$	$\xi^{3,1}$	$\xi^{3,2}$	$\xi^{3,3}$	$\xi^{3,4}$	$\xi^{3,5}$
$n(t,i)$	2	3	3	1	2	3	1

Table 2.1: A notation for the scenario tree in Figure 2.1

2.3.3 Deterministic Equivalent Formulations

Computational solution methods in stochastic programming, including the methods we develop in Chapter 3, often rely on the ability to solve a T-stage stochastic linear program defined on a finite scenario tree with a moderate number of scenarios. As for a two-stage stochastic program with recourse, we can formulate a T-stage stochastic linear program, (2.16)-(2.17), as a deterministic linear program in a couple of ways assuming that (A1)-(A4) hold, and that the supports of $\tilde{\xi}_t, t = 1, \ldots, T$, are finite. Different deterministic formulations are suitable to different solution methods that we discuss in Section 2.4.1.

For a finite scenario tree, the expectation operator in (2.16)-(2.17) can be expressed as a summation; therefore, (2.16)-(2.17) can be stated as

$$\begin{aligned} \min_{x_1} \ & c_1 x_1 + \sum_{i=1}^{n_2} p_2^{i|1} h_1(x_1, \xi^{2,i}) \\ \text{s.t.} \ & A_1 x_1 = b_1 \\ & x_1 \geq 0, \end{aligned} \tag{2.20}$$

where for all $j \in D_t^i,\ i = 1, \ldots, n_t,\ t = 2, \ldots, T$,

$$\begin{aligned} h_{t-1}(x_{t-1}, \xi^{t,j}) = \min_{x_t} \ & c_t^j x_t + \sum_{k \in D_t^j} p_{t+1}^{k|j} h_t(x_t, \xi^{t+1,k}) \\ \text{s.t.} \ & A_t^j x_t = b_t^j - B_t^j x_{t-1} \\ & x_t \geq 0, \end{aligned} \tag{2.21}$$

realization $\xi^{t+1,k} = (\xi^{t,j}, \xi_{t+1}^k)$, $k \in D_t^j$, and $h_T = 0$. The conditional probability mass function $p_t^{j|i}$ is defined as $p_t^{j|i} = P(\tilde{\xi}_t = \xi_t^j | \tilde{\xi}^{t-1} = \xi^{t-1,i}), j \in D_t^i, i = 1, \ldots, n_t, t = 2, \ldots, T$, and $p_{T+1}^{j|i} = 0,\ \forall i, j$.

Instead of expressing a T-stage stochastic linear program as recursions (2.20)-(2.21), we can enumerate all linear programming subproblems corresponding to each

scenario and express, as in the two-stage model, its deterministic equivalent program as follows:

$$\begin{aligned} \min \ & \sum_{t=1}^{T} \sum_{i=1}^{n_t} p_t^i c_t^i x_t^i \\ \text{s.t.} \ & B_t^i x_{t-1}^{a(i)} + A_t^i x_t^i = b_t^i, \ i = 1, \ldots, n_t, \ t = 1, \ldots, T \\ & x_t^i \geq 0, i = 1, \ldots, n_t, \ t = 1, \ldots, T \end{aligned} \tag{2.22}$$

where $p_t^i = P(\tilde{\xi}^t = \xi^{t,i})$, $B_1^1 x_0^{a(1)} \equiv 0$, and $x_{t-1}^{a(i)}$ is the decision vector at the stage $t-1$ ancestor node of $\xi^{t,i}$. The size of the deterministic equivalent program grows with the number of scenarios, which in turn grows exponentially with T. Nonanticipativity constraints are enforced implicitly through the ancestor operator in (2.22). This deterministic formulation (2.22) possesses a special structure in its constraint matrix amenable to methods that decompose (2.22) by time stage, e.g., the well-known L-shaped method [119].

To express the nonanticipativity constraints explicitly, we duplicate all decision variables for each scenario, and denote its scenario dependence by $x_t(s)$, where s corresponds to the scenario index of $\tilde{\xi}^{T,s}, s = 1, \ldots, n_T$. Thus, a T-stage stochastic linear program can be stated with explicit nonanticipativity constraints as

$$\min \ \sum_{s=1}^{n_T} \sum_{t=1}^{T} p(s) c_t(s) x_t(s) \tag{2.23a}$$

$$\text{s.t.} \ B_t(s) x_{t-1}(s) + A_t(s) x_t(s) = b_t(s), \ s = 1, \ldots, n_T, \ t = 1, \ldots, T \tag{2.23b}$$

$$x_t(s) \geq 0, s = 1, \ldots, n_T, \ t = 1, \ldots, T \tag{2.23c}$$

$$x_t(s) \in \mathcal{N}_t, \ s = 1, \ldots, n_T, \ t = 1, \ldots, T-1, \tag{2.23d}$$

where $(A_t(s), B_t(s), b_t(s), c_t(s))$ is given by $(A_t^{a^{T-t}(s)}, B_t^{a^{T-t}(s)}, b_t^{a^{T-t}(s)}, c_t^{a^{T-t}(s)})$ in our notation, $p(s)$ is the probability of scenario $\xi^{T,s}$, i.e., $p(s) = P(\tilde{\xi}^T = \xi^{T,s})$, and $\mathcal{N}_t$ is the nonanticipativity constraint set in stage t.

The nonanticipativity constraint sets, $\mathcal{N}_t, t = 1, \ldots, T-1$, can be algebraically expressed in a number of ways but all have the effect of requiring $x_t(s)$ and $x_t(s')$ to be identical if $\xi^{t,a^{T-t}(s)}$ and $\xi^{t,a^{T-t}(s')}$ are identical. Different expressions of $\mathcal{N}_t$ lead to different dual formulations and a particular expression may be desired for certain solution methods. More details can be found in [10, 25, 67], and references cited therein. Performing a Lagrangian relaxation with respect to the nonanticipativity constraints separates (2.23) by scenario and there are solution methods that exploit this structure, e.g., the diagonal quadratic approximation algorithm [98], and the progressive hedging algorithm [108].

Although larger than (2.22), formulation (2.23) is often easier to formulate a stochastic program, especially a multi-stage stochastic program with recourse, by explicitly expressing nonanticipativity constraints in modeling languages for mathematical programming such as GAMS and AMPL.

2.4 Solution Methods

To solve a T-stage stochastic program (2.8)-(2.10), at each stage t we must optimize an objective function that is a multi-dimensional integral. For the special case of a stochastic linear program (2.16)-(2.17), the stage t objective function is convex but, in general, non-smooth. While algorithms for optimizing such functions typically require an ability to evaluate (sub)gradients, it can be difficult or impossible to perform even a single exact function evaluation of $E[\phi_t(x^t, \tilde{\xi}^{t+1})|\tilde{\xi}^t]$. The degree of difficulty depends on both the nature of the stochastic process, $\{\tilde{\xi}_t\}_{t=1}^T$, and functions $\phi_t, t = 1, \ldots, T$. For instance, when each of the random vectors from $\{\tilde{\xi}_t\}_{t=1}^T$ is continuous and of high dimension, evaluating the recourse function requires a high dimensional numerical integration, which may be impossible to carry out exactly, even for a given policy $\hat{x}^T(\tilde{\xi}^T)$. On the other hand, if ϕ_t is separable

in each component of $\tilde{\xi}^{t+1}$, the conditional expectation becomes a one-dimensional integral and can be easy to evaluate numerically. Accordingly, solution methods for stochastic programs are tailored to the specific nature of $\{\tilde{\xi}_t\}_{t=1}^{T}$ and $\phi_t, t = 1, \ldots, T$, as we will indicate when reviewing each class of solution methods.

We classify a solution method as either an exact or approximation method. In our classification, exact methods include both analytical solution methods and those methods that algorithmically solve a stochastic program (without approximating its multi-dimensional integral objective function) to yield an objective function value that is arbitrarily close to the optimal objective function value. These exact methods are reviewed in Section 2.4.1.

We divide approximation methods into three types. The first type contains approximation methods arising from using Monte Carlo sampling to approximate a multi-dimensional integral objective function. The solution methods we develop in Chapter 3 for a multi-stage stochastic program involve both an "exact" decomposition algorithm from large-scale optimization, and Monte Carlo sampling. The second type are methods that form either distributional or functional approximations of the multi-dimensional integral objective function through deterministic bounds derived from certain inequalities, e.g., Jensen's inequality. These bounds can be sequentially improved to yield a deterministic statement regarding the optimal objective function value of the original problem. We only give a brief review for this type of methods since they do not play a central role in our subsequent development. The third type contains methods that attempt to approximate the underlying scenario tree. Here, the primary motivation is to reduce the number of scenarios in the scenario tree so that the resulting deterministic formulation of a stochastic program is of manageable size, and can be readily solved by existing solution methods developed for moderate-sized stochastic programs. All three types of approximation methods

are reviewed in Section 2.4.2.

2.4.1 Exact Methods

Exact solution methods for stochastic programs provide an exact optimal solution to a stochastic program, either analytically or algorithmically. Two-stage stochastic programs for which an optimal solution is analytically available are (i) the newsboy problem, which may be viewed as a stochastic linear program with simple recourse with a scalar decision variable (see, e.g., Birge and Louveaux [16, §1]) and (ii) stochastic programs with simple recourse (2.6) in which all parameters are deterministic except for $\tilde{b}_2$ whose components have marginal distributions of some special type, e.g., uniform [7], exponential [122]. Although the optimal value in some cases of (i) and (ii) can be analytically derived, we often still need to evaluate it either numerically or algorithmically. For example, an optimal solution of (i) with a normally-distributed random demand is expressed in terms of the normal cumulative distribution function, which must be numerically evaluated. An optimal value in case (ii) when $\tilde{b}_2$ is uniformly distributed and the constraints are linear can be obtained by solving a convex quadratic program.

When the support of the random vector is finite with a relatively small number of realizations, a stochastic program can be expressed as a deterministic equivalent program and solved "exactly" via an optimization algorithm (within a numerical tolerance). If it is a linear program, then it might be solved directly by the simplex method or by an interior point algorithm, and by a branch-and-bound procedure if it is a mixed-integer program.

For a problem with a larger number of realizations, a direct application of generic linear programming algorithm can fail due to the size of the deterministic equivalent program. However, in the linear programming setting, the special struc-

ture of the deterministic equivalent program allows application of large-scale optimization techniques, e.g., decomposition methods, which can still solve the deterministic equivalent program within a numerical tolerance. Decomposition methods are important tools for solving stochastic linear programs with a moderate number of scenarios.

In stochastic programming, we can categorize decomposition methods into two types. The first type decomposes a problem by stage and each stage t then has a collection of subproblems corresponding to each stage t node in the scenario tree. The L-shaped method [119] falls into this category. The second type decomposes a problem by scenario. Methods in this category are primarily based on Lagrangian relaxation of nonanticipativity constraints resulting in single-scenario deterministic mathematical programs corresponding to each scenario in the tree. Implementations of by-stage decomposition methods can incorporate multiple stages in a subproblem (instead of just one) and implementations of by-scenario decomposition methods can incorporate multiple scenarios in a subproblem.

Multi-stage stochastic optimization problems that can be modeled with only a few state variables (e.g., 2 or 3) and simple constraint sets can be solved by dynamic programming (see, e.g., Bellman [8] and Bertsekas [12]). Two multi-stage stochastic optimization problems we study in Chapter 4, an American-style option pricing problem [18] and a stochastic lot-sizing problem [63, 64], are solved via dynamic programming. The curse of dimensionality (due to a large number of state variables) limits the general applicability of dynamic programming.

Frequently, approximation methods use, in some manner, decomposition methods. For example, a stochastic program that has an "unmanageable" number of scenarios can be approximated with one that has a modest number, and then a decomposition method is applied to solve the modest-sized approximating problem.

Unmanageable problems arise when discrete probability distributions have a very large cardinality, or infinite, sample space. Thus, decomposition methods certainly play a key role in the solution of such problems. The approximation methods for multi-stage stochastic programs we develop in Chapter 3 also rely heavily on the L-shaped method, and so we provide a detailed review and establish associated notation before reviewing Lagrangian-based decomposition methods. Again, we begin by considering the L-shaped method for a two-stage stochastic linear program with recourse, and extend our discussion to the multi-stage L-shaped method for ease of exposition.

The L-shaped Method

The L-shaped method proposed by Van Slyke and Wets [119] is a decomposition technique that exploits special structure of the deterministic equivalent program of a two-stage stochastic linear program. It is closely related to Benders' decomposition method [9] for mixed-integer programming and Kelly's cutting-plane method [82] for non-linear programming. The L-shaped method is one of the most important exact solution methods for solving stochastic linear programs with discrete distributions [45, 114]. To develop the algorithm, consider a two-stage *deterministic* linear program of the form

$$\begin{aligned} \min_{x_1} \ & c_1 x_1 + h(x_1) \\ \text{s.t.} \ & A_1 x_1 = b_1 \\ & x_1 \geq 0, \end{aligned} \tag{2.24}$$

where

$$\begin{aligned} h(x_1) = \min_{x_2} \ & c_2 x_2 \\ \text{s.t.} \ & A_2 x_2 = b_2 - B_2 x_1 \quad : \pi \\ & x_2 \geq 0. \end{aligned} \tag{2.25}$$

Dual variables on constraint $A_2 x_2 = b_2 - B_2 x_1$ are the components of π. We assume that for every $x_1' \in \{x_1 \in \mathbb{R}^d : A_1 x_1 = b_1, x_1 \geq 0\}$, subproblem (2.25) is feasible and has a finite optimal solution. By linear programming duality, we can write (2.25) as

$$\begin{aligned} h(x_1) = \ & \max_{\pi} \ \pi(b_2 - B_2 x_1) \\ & \text{s.t.} \ \pi A_2 \leq c_2 \\ = \ & \max_{1 \leq i \leq L} \ \pi^{(i)}(b_2 - B_2 x_1) \end{aligned}$$

where $\pi^{(i)}, i = 1, \ldots, L$, are all the extreme points of $\Pi = \{\pi : \pi A_2 \leq c_2\}$. Thus, the two-stage deterministic linear program can be equivalently written as

$$\begin{aligned} \min_{x,\theta} \ & c_1 x_1 + \theta \\ \text{s.t.} \ & A_1 x_1 = b_1 \\ & \theta \geq \pi^{(i)}(b_2 - B_2 x_1), \ i = 1, \ldots, L \\ & x_1 \geq 0. \end{aligned} \tag{2.26}$$

The constraints on θ are called *cuts*; $-\pi^{(i)} B_2$ and $\pi^{(i)} b_2$ are cut-gradient vectors and cut-intercepts, respectively. Linear program (2.26) is known as the *full master* program, which has a total of L cuts corresponding to the L extreme points of Π. When the number of cuts is less than L, the full master program becomes a *relaxed* master program. The form of the full master program suggests an algorithm where only a subset of extreme points, $\{\pi^{(1)}, \ldots, \pi^{(l)}\}$, and hence cuts, are sequentially

generated from solving the second stage program at values of x specified by solving the relaxed master, which is iteratively updated with additional cuts.

Note that if $x_1 \in \mathbb{R}^{d_1}$, then only $d_1 + 1$ cuts are needed to identify an optimal solution, and typically $d_1 \ll L$. However, we do not know *a priori* which cuts are required and have no control, in general, of which ones are to be generated. Thus, in the worst case, we will need to generate all of them. In practice, the termination criterion is based on the gap between a lower and upper bounds on the optimal objective function value. In an iteration, the lower bound is given by the optimal objective function value of the relaxed master program, while the upper bound is obtained by evaluating (2.24)-(2.25) at an $(\hat{x}_1, \hat{x}_2)$ pair that comes from sequentially solving the relaxed master program for $\hat{x}_1$ and then the subproblem (2.25) for $\hat{x}_2$, given the master solution $\hat{x}_1$. The gap between lower and upper bounds identifies an $(\hat{x}_1, \hat{x}_2)$ pair that yields an objective function value within a prespecified tolerance of the optimal objective function value.

With appropriate definitions of A_2, B_2, b_2, c_2 and x_2 in (2.24)-(2.25), the L-shaped method can be directly applied to the deterministic equivalent of (2.4)-(2.5). Applying the decomposition scheme described earlier to (2.7) yields the following master problem and subproblems:

$$\begin{aligned} \min_{x_1,\theta} \quad & c_1 x_1 + \theta \\ \text{s.t.} \quad & A_1 x_1 = b_1 \\ & e\theta \geq \vec{G} x_1 + \vec{g} \\ & x_1 \geq 0, \end{aligned}$$

where e is the vector of all 1's, each row of the cut-gradient matrix $\vec{G}$ is

$$G = -\sum_{i=1}^{n} p^i \pi^i B_2^i, \tag{2.27}$$

each component of the cut-intercept vector $\vec{g}$ is

$$g = \sum_{i=1}^{n} p^i \pi^i b_2^i, \tag{2.28}$$

and π^i, $i = 1, \ldots, n$, are optimal dual solutions of sub(i):

$$\begin{aligned} \min_{x_2} \ & c_2^i x_2 \\ \text{s.t.} \ & A_2^i x_2 = b_2^i - B_2^i x_1 \quad : \pi \\ & x_2 \geq 0. \end{aligned}$$

In our development of the L-shaped method, we assume (2.24)-(2.25) has relatively complete recourse. Under such an assumption, it is not possible to generate a first-stage decision which results in an infeasible second stage subproblem. All of the test problems we use in this dissertation have relatively complete recourse. That said, it is possible to modify the L-shaped method so that it generates so-called feasibility cuts to handle problems not satisfying the assumption. Birge and Louveaux [16, §5.1] describe the L-shaped method for problems that do not have relatively complete recourse.

There are two important variations to the basic L-shaped method that we have just described: a variant that uses multi-cut and a variant that adds a proximal term in the master program. In our development in Chapter 3, we do not use these other variations directly. However, they can be easily incorporated in our solution methods with a straightforward modification of our methods. We briefly discuss the basic idea of these variations.

In a multi-cut variant of the L-shaped method [15], multiple cuts are appended in each iteration, i.e., every sub(i) adds its own cut to the master program instead of combining the dual solution of each sub(i) to generate a single cut. The

master program of the multi-cut variant is

$$\begin{aligned} \min_{x,\theta^i} \ & c_1 x_1 + \sum_{i=1}^{n} p^i \theta^i \\ \text{s.t.} \ & A_1 x_1 = b_1 \\ & e\theta^i \geq \vec{G}^i x_1 + \vec{g}^i, i = 1, \ldots, n \\ & x_1 \geq 0, \end{aligned} \tag{2.29}$$

where each row of the cut-gradient matrix is $G^i = -\pi^i B_2^i$, and each component of the cut-intercept vector is $g^i = \pi^i b_2^i$. Note that G^i and g^i are the cut gradient and intercept computed from a dual extreme point of sub(i).

The multi-cut master program has increased resolution of the recourse function. However, the number of cuts and decision variables in the multi-cut master is larger than that of the single-cut master program. For both single-cut and multi-cut variants, as the number of cuts grow, the master problem can be more difficult to solve. Thus, it is practically important to control the size of the master program. This can be done by limiting the number of cuts maintained in the relaxed master program. For example, cuts that have not been active for the largest number of iterations or the cut with the largest surplus can be dropped.

The second variant, proposed by Ruszczyński [111], is to add a proximal term in the objective function of the master program:

$$\begin{aligned} \min_{x,\theta} \ & c_1 x_1 + \theta + \rho \parallel x_1 - \hat{x}_1 \parallel_2^2 \\ \text{s.t.} \ & A_1 x_1 = b_1 \\ & e\theta \geq \vec{G} x_1 + \vec{g} \\ & x_1 \geq 0. \end{aligned}$$

Here, the proximal term penalty weight ρ is positive and can be dynamically modified during the algorithm, and $\hat{x}_1$ is the incumbent or best feasible solution known

so far in the course of the algorithm. The purpose of the proximal term is to discourage x_1 from moving too far from $\hat{x}_1$, and this can speed convergence of the L-shaped algorithm significantly, especially for problems in which the dimension of x_1 is high. However, the computational effort also increases because the master program is now a quadratic program. Other techniques to improve the efficiency of the L-shaped algorithm can be found in Birge and Louveaux [16, §5.4], Gassmann and Wallace [59], Ruszczyński and Świetanowski [113], Wets [124, 125].

The Multi-stage L-shaped Method

Birge [13] proposes the multi-stage L-shaped method for T-stage stochastic linear programs. Under the assumptions we provide for the multi-stage stochastic linear program with recourse, given by (2.20)-(2.21), in Section 2.3.1, we review the multi-stage L-shaped method for (2.20)-(2.21). By applying the (single-cut) L-shaped method to stage t subproblem under realization $\xi^{t,i}$, we can express this subproblem, denoted $\text{sub}(t,i)$, as

$$\min_{x_t,\theta_t} \; c_t^i x_t + \theta_t \tag{2.30a}$$

$$\text{s.t.} \; A_t^i x_t = b_t^i - B_t^i x_{t-1}^{a(i)} \tag{2.30b}$$

$$e\,\theta_t \geq \vec{G}_t^i x_t + \vec{g}_t^i \tag{2.30c}$$

$$x_t \geq 0, \tag{2.30d}$$

for $i = 1, \ldots, n_t$, $t = 1, \ldots, T-1$. Each row of the cut-gradient matrix, $\vec{G}_t^i$, and each component of the cut-intercept vector, $\vec{g}_t^i$, are computed from $\text{sub}(t+1, j)$, $j \in D_t^i$. Formulae for these computations are similar to those of a two-stage recourse model, given by (2.27) and (2.28), except that each component of the cut-intercept vector contains an additional term since $\text{sub}(t,i)$, $i = 1, \ldots, n_t, t = 1, \ldots, T-1$, contain cut constraints, and the conditional probability mass function is used instead. In

particular, let π_t and α_t be dual row vectors associated with constraints (2.30b) and (2.30c), respectively. We use (π_t^i, α_t^i) to denote an optimal dual solutions of $\text{sub}(t,i)$, i.e., (π_t^i, α_t^i) is an optimal solution of

$$\begin{aligned} \max_{\pi_t, \alpha_t} \ & \pi_t(b_t^i - B_t^i x_{t-1}^{a(i)}) + \alpha_t \vec{g}_t^i \\ \text{s.t.} \ & \pi_t A_t^i - \alpha_t \vec{G}_t^i \leq c_t^i \\ & \alpha_t \, e = 1 \\ & \alpha_t \geq 0, \end{aligned} \tag{2.31}$$

for $i = 1, \ldots, n_t, t = 1, \ldots, T-1$. Then, we form each row of the cut-gradient matrix and each component of the cut-intercept vector of $\text{sub}(t,i)$ by

$$G_t^i = -\sum_{j \in D_t^i} p_{t+1}^{j|i} \pi_{t+1}^j B_{t+1}^j, \tag{2.32}$$

and

$$g_t^i = \sum_{j \in D_t^i} p_{t+1}^{j|i} \pi_{t+1}^j b_{t+1}^j + \sum_{j \in D_t^i} p_{t+1}^{j|i} \alpha_{t+1}^j \vec{g}_{t+1}^j. \tag{2.33}$$

Since $\xi^{T,i}, i = 1, \ldots, n_T$, are leaf nodes, $\text{sub}(T,i)$, $i = 1, \ldots, n_T$, do not contain cut constraints and the variable θ_T. Accordingly, the components of the cut-intercept vectors, $g_{T-1}^i, i = 1, \ldots, n_{T-1}$, do not contain the last term in (2.33).

A statement of the multi-stage L-shaped method for a T-stage stochastic linear program is given in Figure 2.2. Multi-cut and proximal-term variants of the multi-stage L-shaped method are available. We refer the reader to Gassmann [57] and Ruszczyński [112] for further details since we do not use these variants in our implementation.

The scenario tree can be traversed in a different ways when we use the multi-stage L-shaped method. The algorithmic statement in Figure 2.2 illustrates the *fastpass* tree traversal [127] in which an optimal solution from each stage is

Step 0 Define $toler \geq 0$ and let $\overline{z} = \infty$ (upper bound).
Initialize the set of cuts for sub(t, i) with $\theta \geq -M$,
$i = 1, \ldots, n_t$, for $t = 1, \ldots, T-1$.

Step 1 Solve sub$(1, 1)$ and let (x_1, θ_1) be its solution.
Let $\underline{z} = c_1 x_1 + \theta_1$ (lower bound).

Step 2 Do $t = 2$ to T.
  Do $i = 1$ to n_t.
    Form the right-hand side of sub(t, i): $b_t^i - B_t^i x_{t-1}^{a(i)}$.
    Solve sub(t, i). Let x_t^i be its solution.
    If $t = T$, also let π_T^i be the optimal dual vector.
  Let $\hat{z} = c_1 x_1 + \sum_{t=2}^{T} \sum_{i=1}^{n_t} p_t^i c_t^i x_t^i$.

Step 3 If $\hat{z} < \overline{z}$ then let $\overline{z} = \hat{z}$ and $x_t^{i,*} = x_t^i, \forall i, t$.
If $\overline{z} - \underline{z} \leq \min(|\overline{z}|, |\underline{z}|) \cdot toler$ then stop: $x_t^{i,*} \forall i, t$ is a policy with an objective function value within $100 \cdot toler\%$ of the optimal value.

Step 4 Do $t = T - 1$ downto 2.
  Do $i = 1$ to n_t.
    Augment sub(t, i)'s set of cuts with $\theta_t - G_t^i x_t \geq g_t^i$.
    Form the right-hand side of sub(t, i): $b_t^i - B_t^i x_{t-1}^{a(i)}$.
    Solve sub(t, i). Let (π_t^i, α_t^i) be the optimal dual vector.
  Augment sub$(1, 1)$'s set of cuts with $\theta_1 - G_1^1 x_1 \geq g_1^1$.
  Goto Step 1.

Figure 2.2: The multi-stage L-shaped algorithm using the fastpass tree traversal strategy for a T-stage stochastic linear program.

passed down to all its corresponding descendants until the last stage is reached, and then the cuts formed by the descendants at each stage are passed back up to the corresponding ancestor subproblems until the first stage is reached. This process repeats until the algorithm terminates. Another alternative is that at stage t the cuts are not passed back to the stage $t-1$ ancestor until an optimal solution is obtained to the problem defined by the subtree rooted at the stage t nodes. This type of tree traversal strategy is called *shuffle* [93]. A third strategy is to not pass a stage t solution down to the stage $t+1$ descendants until the cut that would be

passed back to the stage $t-1$ ancestor is redundant. This type of tree traversal strategy is known as *cautious* [127]. Experiments have suggested that the fastpass strategy is the most efficient strategy [57, 93, 127]. So, we only implement the fastpass tree traversal of the multi-stage L-shaped method in this dissertation.

Lagrangian-based Decomposition Methods

We only give a brief overview of Lagrangian-based decomposition methods in stochastic programming since the solution methods we develop in Chapter 3 do not directly involve this type of decomposition. For further details of Lagrangian-based decomposition methods, we refer the reader to [24, 62] and references cited therein.

The motivation of applying Lagrangian relaxation in stochastic programming is the same as that in deterministic mathematical programming. Specifically, the relaxation of complicating constraints decomposes the original problem into subproblems that can be solved efficiently. For example, nonanticipativity constraints, (2.23d), in a deterministic formulation of a T-stage stochastic linear program can be viewed as complicating constraints, and are typically dualized in a Lagrangian-based decomposition method. If (2.23) possesses a network structure, the resulting single-scenario subproblems of such relaxation are deterministic multi-stage network programs for which an efficient algorithm may be readily available. Gröwe-Kuska et al. [62] apply a Lagrangian-based decomposition method in stochastic programming to relax power and demand constraints instead to achieve subproblems for which specialized algorithms are developed.

One computational disadvantage of Lagrangian relaxation methods is due to slow convergence of the subgradient search algorithm when the associated Lagrangian dual problem (a non-smooth convex optimization problem) is optimized.

One remedy is to use instead an augmented Lagrangian, which contains a quadratic proximal term, to help speed up the convergence. In stochastic programming, Lagrangian-based decomposition methods that use augmented Lagrangian includes progressive hedging algorithm [108], diagonal quadratic approximation method [98], and methods proposed by Rosa and Ruszczyński [110]. However, these methods need to approximate the quadratic proximal term such that the scenario decomposition is still achieved.

For non-convex problems, such as stochastic mixed-integer programs, Lagrangian-based decomposition methods can yield a lower bound on the optimal objective function value, or can be combined with other search techniques or heuristic to find an optimal, or near-optimal, solution. Carøe and Schultz [24] integrate a Lagrangian-based decomposition method with the branch-and-bound procedure for two-stage stochastic mixed-integer program and report encouraging results. Gröwe-Kuska et al. [62] develop a Lagrangian-based heuristic to obtain a near-optimal solution for a multi-stage stochastic mixed-integer program. Haugen, Løkketangen and Woodruff [64, 88] apply the progressive hedging algorithm and the tabu search heuristic to multi-stage stochastic mixed-integer programs.

In summary, exact methods solve stochastic programs either analytically or algorithmically (within a numerical tolerance). Analytical solutions are available for only a few classes of stochastic programs. When the cardinality of the sample space is small, we can apply deterministic optimization algorithms to solve the deterministic equivalent program directly. Dynamic programming can be applied to multi-stage stochastic optimization problems with simple constraint sets when the number of state variables is small. Decomposition methods such as the L-shaped and Lagrangian-based algorithms can extend the size of problems that we can address, but the total number of scenarios still must be relatively modest. Decomposition

methods also play an important role in the solution methods for stochastic programs with a large number of scenarios. In such situations, decomposition methods are used to solve approximating problems with a modest number of scenarios.

2.4.2 Approximation Methods

Monte Carlo Sampling-based Methods

In numerical integration, Monte Carlo method is regarded as the method of choice for numerically estimating difficult high-dimensional integrals or summations. Therefore, solution methods for stochastic programs, whose objective function is a high-dimensional integral or summation, naturally incorporate Monte Carlo sampling. Such stochastic programs can arise, for example, when each random vector of $\{\tilde{\xi}_t\}_{t=1}^{T}$ is of high dimension and continuous or discrete with a large number of realizations. A recent survey on the use of Monte Carlo sampling in stochastic optimization is given by Morton and Popova [92].

In stochastic programming, Monte Carlo sampling is performed either "inside" or "outside" of the optimization algorithm. Internal sampling-based methods replace computationally expensive or difficult exact computations with Monte Carlo estimates during the execution of the algorithm. In the two-stage stochastic program (2.2), internal sampling procedures replace exact evaluations of $E\phi_1(x_1, \tilde{\xi}_2)$ and/or its (sub)gradient with the standard sample mean estimator. For example, stochastic quasi-gradient algorithms use a sampling-based estimate of a (sub)gradient of $E\phi_1(x_1, \tilde{\xi}_2)$ and may be regarded as a sampling-based steepest-descent algorithm. Ermoliev describes stochastic quasi-gradient methods in [50]. A sampling-based L-shaped method developed by Infanger [72] and the stochastic decomposition method developed by Higle and Sen [66] employ sampling to estimate cuts of $E\phi_1(x_1, \tilde{\xi}_2)$ in the L-shaped method for two-stage stochastic linear programs.

Several variants of internal sampling-based L-shaped methods are developed for multi-stage stochastic linear programs. Their differences lie in the use of Monte Carlo estimation in different steps of the L-shaped algorithm depending on the characteristics of the underlying stochastic process. Pereira and Pinto [102] estimate the objective function by sampling in the forward pass of the L-shaped method for interstage independent linear problems that have many stages but a manageable number of scenarios per stage. Objective function cuts are computed exactly in the backward pass, and can be shared among subproblems in the same stage due to interstage independence. Donohue [39] enhances this algorithm with a better scheme for selecting feasible solutions in the forward pass. Dantzig, Glynn, and Infanger [33, 34] employ importance sampling in both forward and backward passes for interstage independent linear problems with possibly larger number of scenarios per stage and report considerable variance reduction. Chen and Powell [28] employ sampling within the L-shaped method to construct deterministically valid cuts for the objective function in each stage of the problem whose random parameters appear only in the right hand side of the constraints.

In external sampling-based methods, the underlying stochastic process is approximated by a finite empirical scenario tree constructed through Monte Carlo sampling. An approximate optimal objective function value is obtained by solving a stochastic program defined on an empirical scenario tree. This type of method is also known as sample average approximation [115] in the stochastic programming literature. Under appropriate assumptions, strong consistency of the approximate optimal objective function value for the multi-stage problems is ensured by the results in [35, 39, 79, 115], i.e., as the number of samples at each node of the sample scenario tree grows large, the approximate optimal objective function value converges to the true optimal objective function value with probability one. For finite

sample trees, we will show in Chapter 3 that the approximate optimal value provides a lower bound, in expectation, to the true optimal value. This lower bound has been independently developed and applied in various settings [18, 91, 99, 115] for establishing solution quality in stochastic programming. The value of using a lower bound to establish a solution quality for a minimization problem is widely recognized in other areas of optimization. In the context of employing Monte Carlo techniques in stochastic programming, exact lower bounds are not available; instead the lower bound are statistical in nature. A lower bound estimator is used to construct a confidence interval on the optimality gap to determine the quality of a feasible solution by Mak, Morton, and Wood [91] for two-stage stochastic programs, and by Broadie and Glasserman [18] for the American-style option pricing problem. Recently, Linderoth, Shapiro, and Wright [87] and Verweij et al. [120] report encouraging computational results of this approach for different classes of two-stage stochastic programs. Norkin, Pflug, and Ruszczyński [99] develop a stochastic branch-and-bound procedure in which a lower bound estimator is used in internal fashion for pruning the search tree.

As is frequently the case with Monte Carlo estimation, the method is more effective if the variance of estimators is small. Many standard variance reduction techniques have been applied in conjunction with Monte Carlo sampling-based methods for stochastic programs. Dantzig, Glynn, and Infanger [33, 72] employ importance sampling in sampling-based cutting-plane algorithms. Mak, Morton, Wood [91] employ the common random numbers technique to reduce variance for the external sampling procedure they develop. Linderoth, Shapiro, and Wright [87] use Latin hypercube sampling in their empirical study of a external sampling-based procedure, while Bailey, Jensen, and Morton [4] use Latin hypercube sampling to reduce variance in estimating the recourse function. Higle [65] studies the effectiveness

of various well-known variance reduction techniques for two-stage stochastic linear programs.

Bounding Methods

This type of approximation method attempts to deterministically bound the optimal objective function value of a stochastic program by either approximating the distribution of the random vector or the objective function so that the multi-dimensional integral objective function is easier to compute. The resulting bounds can be sequentially refined to sufficient accuracy provided that the dimension of the random vector is moderate. In such situations, bounding methods are attractive options when the random vector is continuous or the cardinality of its sample space is too large for exact computation.

The validity of the bounds typically requires specific properties of the objective function, notably convexity with respect to the random vector. For example, bounding methods developed for a two-stage stochastic linear program with recourse whose objective function is convex with respect to the random vector, i.e., (2.4)-(2.5) with random elements appearing only in $\tilde{B}_2$ and $\tilde{b}_2$, employ the classical lower and upper bounds for the expectation of convex functions, which based on the inequalities of Jensen [75] and Edmundson-Madansky [48, 90]. We refer the reader to Dokov and Morton [38], and references therein for a literature review, and to Kall [77] for a tutorial on the bounding methods. Much of the work on this class of methods has been in two-stage stochastic programming. Bounding methods for multi-stage stochastic program with recourse include those methods developed by Edirisinghe [46], Edirisinghe and Ziemba [47], and Frauendorfer [55]. For a special class of multi-stage stochastic programs known as dynamic vehicle allocation problems, Cheung, Frantzeskakis, and Powell [29, 54] develop bounding methods based

on functional approximation.

Scenario Tree Approximation-based Methods

This type of approximation method attempts to approximate a scenario tree that has a large number of scenarios with a modest-sized tree so that the resulting approximating problem can be solved by a decomposition method. Dupačová, Gröwe-Kuska and Römisch [40] and Pflug [103] construct modest-sized approximating scenario trees such that its approximation error, quantified via probability metrics, is minimized. Consigli, Dupačová and Wallace [31] and Høyland, Kaut and Wallace [71] apply statistical techniques such as cluster analysis, importance sampling, and moment matching, in scenario tree approximation. Pennanen and Koivu [100] use a low-discrepancy sequence in sample tree construction to reduce the approximation error. The methods of [31, 40, 71, 100, 103] attempt to approximate the stochastic program by focusing primarily on properties of the stochastic process $\{\tilde{\xi}_t\}_{t=1}^T$.

Dempster and Thompson [37] construct non-uniform sample trees whose number of descendants is determined dynamically by an estimate of the expected value of perfect information (EVPI). Korapaty [85] constructs sample trees with varying number of descendants at each node to reduce bias of an upper bound estimator associated with an American-style option pricing problem. For multistage stochastic linear programs with random right-hand-side, Casey and Sen [26] attempt to approximate both the underlying stochastic process and problem structure by applying the sensitivity analysis of linear programming to guide scenario tree construction. The methods of [26, 37, 85] take both the properties of the stochastic process $\{\tilde{\xi}_t\}_{t=1}^T$ and the underlying problem's structure into account to approximate the scenario tree. The goal of all the scenario tree approximation-based methods

is to construct an approximate scenario tree such that it yields the "best" approximating problem while keeping its size within a given computational budget.

2.5 Epi-convergence

Approximation methods of stochastic programming often generate and solve a sequence of approximating problems. It is important to know whether an approximation method can asymptotically yield an optimal solution to the original problem. Epi-convergence is rooted in convergence of epi-graphs of functions, and is a powerful approach to establish convergence for approximation methods of stochastic programming. Rockafellar and Wets [109] provide an extensive development of epi-convergence. An excellent elementary review is also provided by Kall [76].

A sequence of real-valued function $\{f_\nu\}$ where $f_\nu : \mathbb{R}^d \to \mathbb{R}$ is said to epi-converge to an epi-limit f if for each $x \in \mathbb{R}^d$

$$\exists\, \{\hat{x}_\nu\} \text{ such that } \hat{x}_\nu \to x \text{ with } \limsup_{\nu\to\infty} f_\nu(\hat{x}_\nu) \le f(x),$$
$$\text{and } \forall \{x_\nu\} \text{ such that } x_\nu \to x,\ f(x) \le \liminf_{\nu\to\infty} f_\nu(x_\nu).$$

Epi-convergence does not imply, in general, pointwise convergence, and vice versa. An important implication of epi-convergence is that if a sequence approximating objective functions epi-converges to that of the original problem, then a limit point, if it exists, of the associated sequence of approximating optimal solutions solves the original problem. This important result is summarized in the following theorem.

Theorem 1. *Suppose that $\{f_\nu\}$ epi-converges to f. Let $\hat{x}_\nu$ be an optimal solution of*

$$\inf_x f_\nu(x),\ \nu = 1, 2, \ldots.$$

If there is a subsequence $\{\hat{x}_{\nu_k}\}$ of $\{\hat{x}_\nu\}$ converging to a limit point $\hat{x}$, then

$$f(\hat{x}) = \inf_x f(x) = \lim_{k\to\infty} \big(\inf_x f_{\nu_k}(x)\big).$$

Consequently, to establish convergence of an approximation method via epi-convergence, one must show that its sequence of approximating objective functions epi-converges to the original problem. King and Wets [84, Theorem 3.1] provide sufficient conditions under which convergence of Monte Carlo sampling-based methods can be achieved, via epi-convergence, for a general class of convex optimization problems that include two-stage stochastic linear programs with recourse as a special case. Donohue [39] extends the results of King and Wets [84] to multi-stage stochastic linear programs under the assumption that $\{\tilde{\xi}_t\}_{t=1}^{T}$ is interstage independent with finite support.

The following special case of results by Attouch and Wets [3, Theorem 6.2] is useful for establishing convergence of Monte Carlo sampling-based methods for a certain class of two-stage stochastic programs through epi-convergence.

Theorem 2. *If (i) $f(x,\tilde{\xi})$ is lower semi-continuous for each $x \in X \subset \mathbb{R}^d$, and bounded below by a constant, wp1, and (ii) $Ef(\hat{x},\tilde{\xi}) < \infty$ for some $\hat{x} \in X$, then $\nu^{-1}\sum_{i=1}^{\nu} f(x,\tilde{\xi}^i)$, where $\tilde{\xi}^i, i = 1,\ldots,\nu$, are iid samples of $\tilde{\xi}$, epi-converges to $Ef(x,\tilde{\xi})$, wp1, on X.*

Convergence of Monte Carlo sampling-based methods are also obtained under the notion of *uniform* strong law of large numbers for multi-stage stochastic programs. See Kaňková [79] and Shapiro [115]. Dempster [35] develops an approximation method that uses importance sampling and provides its convergence for multi-stage stochastic linear programs.

Chapter 3

Policy Generation and Testing Policy Quality

3.1 Introduction

We develop Monte Carlo sampling-based methods that exploit special structures to solve two classes of computationally difficult multi-stage stochastic programs. As described in Chapter 2, a solution to a multi-stage stochastic program is a policy that specifies what decision to make at each stage under each realization of the stochastic process. The first method we consider applies to multi-stage stochastic linear programs with relatively complete recourse whose stochastic parameters exhibit interstage independence. It is an external sampling-based procedure that uses the multi-stage L-shaped algorithm to solve an approximating problem associated with an empirical scenario tree (with interstage independence) to obtain an approximation of cuts, which are used to generate a policy. These (approximate) cuts can be shared among the subproblems in the same stage similar to the methods developed in [33, 34, 39, 102]. We also indicate how the first method can be extended to handle a particular type of interstage dependency through cut-sharing formulae from [73]. The second method we consider is more computationally expensive but applies to a more general class of multi-stage stochastic programs with recourse. A caveat for this procedure is that it requires the ability to solve that general class of problems with a moderate number of scenarios, i.e., the approximating problems based on an empirical scenario (sub)tree. As we will indicate, asymptotic optimality of both methods can be inferred from strong consistency results in [39, 79, 115]. However, in practice we can only solve stochastic programs with finite empirical

scenario trees. As a result, we develop techniques to determine the quality of an arbitrary feasible policy. In particular, we develop a lower bound estimator and use it to construct a confidence interval on the optimality gap to establish the policy quality in a general multi-stage stochastic programming context in the same spirit as that of [18, 91].

The remainder of this chapter is organized as follows. Section 3.2 details sample scenario tree construction, which plays an important role in our methods. The first policy-generation method is discussed in Section 3.3.1 for multi-stage stochastic linear programs with interstage independence or with a special type of interstage dependence. Section 3.3.2 discusses the second policy generation method, which applies to a more general class of multi-stage problems. Estimating the expected cost of using a specific policy is discussed in Section 3.4. A statistical lower bound on the optimal objective function value is developed in Section 3.5. Procedures for constructing confidence intervals on the optimality gap of a given policy are described in Section 3.6, and associated computational results are reported in Section 3.7.

3.2 Sample Scenario Tree Construction

To construct a sample scenario tree, we perform the sampling in the following conditional fashion: we begin by drawing $n(1) = n_2$ observations of $\tilde{\xi}_2$ from $F_2(\xi_2|\xi^1)$ where ξ^1 is the known first stage realization. Then, we form the descendants of each observation $\xi^{2,i}, i = 1, \ldots, n_2$, by drawing $n(2,i)$ observations of $\tilde{\xi}_3$ from $F_3(\xi_3|\xi^{2,i})$. This process continues until we have sampled $n(T-1,i)$ observations of $\tilde{\xi}_T$ from $F_T(\xi_T|\xi^{T-1,i}),\ i = 1, \ldots, n_{T-1}$. The notation developed in Section 2.3 for a generic finite scenario tree applies to a sample scenario tree. The number of descendants of a node $\xi^{t,i}$ is now determined by the sample size $n(t,i)$. The total number of

nodes in stage $t+1$ is $n_{t+1} = \sum_{r=1}^{n_t} n(t,r)$, and $n(t) = |D_t|$ is the number of immediate descendants of a generic stage t node, ξ^t. The subtree associated with each descendant of node $\xi^{t,i}$ is $\Gamma(\xi^{t+1,j}), j \in D_t^i$.

In addition to the above structure for constructing a sample scenario tree, we require for the purposes of the estimators developed in Section 3.5 that the samples of $\tilde{\xi}_{t+1}$ be drawn from $F_{t+1}(\xi_{t+1}|\xi^t)$ such that they satisfy the following unbiasedness condition

$$E[f_t(x^t, \tilde{\xi}^{t+1})|\tilde{\xi}^t] = E\Big[\frac{1}{n(t)} \sum_{i \in D_t} f_t(x^t, \tilde{\xi}^{t+1,i})|\tilde{\xi}^t\Big], \tag{3.1}$$

wp1, $t = 1, \ldots, T-1$. The simplest method for generating $\tilde{\xi}_{t+1}^i, i \in D_t$, to satisfy (3.1) is to require that they be (conditionally) independent and identically distributed (iid).

Within this framework there are different types of sample scenario trees that can be generated. Consider the case when $\{\tilde{\xi}_t\}_{t=1}^T$ is interstage independent. One possibility is to generate a single set of iid observations of $\tilde{\xi}_{t+1}$ and use this same set of descendants for all stage t nodes $\xi^{t,i}, i = 1, \ldots, n_t$. Another possibility is to generate mutually independent sets of $t+1$ descendant nodes for all stage t nodes. We say the former method uses "common samples" and the latter "independent samples." Both methods of generating a scenario tree satisfy (3.1). The independent-samples method introduces interstage dependency in the sample tree, which was not present in the original tree while the common-samples method preserves interstage independence. Another advantage of the common-samples approach (relative to an independent-samples tree) is that the associated stochastic program lends itself to the solution procedures of Pereira and Pinto [102] and Donohue [39]. On the other hand, because of increased diversity in the sample, one might expect solutions under the independent-samples tree to have lower variability. We return to this issue in Section 3.7 on computational results.

Sampling schemes that generate non-iid descendants can also be used in our framework. In particular, a number of schemes designed to reduce variance satisfy (3.1) and have been successfully used in sampling-based solution methods for stochastic programming. We give a literature review of variance reduction techniques in stochastic programming in Section 2.4.2.

When using the common-samples approach the number of descendant nodes within each stage must be identical but the cardinality of D_t could vary with stage. In the independent-samples approach, we have freedom to select different sample sizes at each node in the scenario tree. Dempster and Thompson [37] use the expected value of perfect information to guide sample tree construction. Korapaty [85] and the procedures we develop in Chapter 4 select the cardinality of descendant sets to reduce bias.

Provided that sampling is done in the conditional manner described above, with (3.1) satisfied, the methods we develop here can be applied to trees with non-constant sizes of descendant sets. That said, in our computation (Section 3.7) we restrict attention to balanced, uniform trees, i.e., $n(t,i) = |D_t^i|$ is constant for all i and t.

Given a sample scenario tree, an approximating problem for multi-stage stochastic program (2.13)-(2.15) can be stated as

$$\begin{aligned} \hat{z}^* &= \min_{x_1} \frac{1}{n(1)} \sum_{i \in D_1} \hat{f}_1(x_1, \tilde{\xi}^1, \Gamma(\tilde{\xi}^{2,i})) \\ &= \min_{x_1} \frac{1}{n_2} \sum_{i=1}^{n_2} \hat{f}_1(x_1, \tilde{\xi}^1, \Gamma(\tilde{\xi}^{2,i})), \end{aligned} \tag{3.2}$$

where

$$\hat{f}_{t-1}(x^{t-1}, \tilde{\xi}^{t-1}, \Gamma(\tilde{\xi}^{t,j})) = \min_{x_t} \frac{1}{n(t,j)} \sum_{i \in D_t^j} \hat{f}_t(x^{t-1}, x_t, \tilde{\xi}^{t,j}, \Gamma(\tilde{\xi}^{t+1,i})), \tag{3.3}$$

$\tilde{\xi}^{t,j} = (\tilde{\xi}^{t-1}, \tilde{\xi}_t^j)$, $j \in D_{t-1}, t = 2, \ldots, T-1$, and

$$\begin{aligned} \hat{f}_{T-1}(x^{T-1}, \tilde{\xi}^{T-1}, \Gamma(\tilde{\xi}^{T,j})) &= f_{T-1}(x^{T-1}, \tilde{\xi}^{T,j}) \\ &= \min_{x_T} f_T(x^{T-1}, x_T, \tilde{\xi}^{T,j}), \end{aligned} \tag{3.4}$$

$\tilde{\xi}^{T,j} = (\tilde{\xi}^{T-1}, \tilde{\xi}_T^j)$, $j \in D_{T-1}$. The value function at a stage t node ξ^t depends on the stochastic history (known at time t), $\tilde{\xi}^t = \xi^t$, the associated decision history, x^t, and the sample subtree $\Gamma(\xi^t)$. In going from (2.13)-(2.15) to (3.2)-(3.4), we are approximating the original population scenario tree by a sample scenario tree.

One of the policy-generation methods we develop is for multi-stage stochastic linear programs and so we explicitly state the associated approximating problem of multi-stage stochastic linear program (2.16)-(2.17):

$$\begin{aligned} \min_{x_1} \; & c_1 x_1 + \frac{1}{n_2} \sum_{i=1}^{n_2} \hat{h}_1(x_1, \tilde{\xi}^1, \Gamma(\xi^{2,i})) \\ \text{s.t.} \; & A_1 x_1 = b_1 \\ & x_1 \geq 0 \end{aligned} \tag{3.5}$$

where for all $j \in D_t^i$, $i = 1, \ldots, n_t$, $t = 2, \ldots, T$,

$$\begin{aligned} \hat{h}_{t-1}(x_{t-1}, \tilde{\xi}^{t-1}, \Gamma(\xi^{t,j})) = \min_{x_t} \; & c_t^j x_t + \frac{1}{n(t,j)} \sum_{k \in D_t^j} \hat{h}_t(x_t, \tilde{\xi}^{t,j}, \Gamma(\xi^{t+1,k})) \\ \text{s.t.} \; & A_t^j x_t = b_t^j - B_t^j x_{t-1} \\ & x_t \geq 0, \end{aligned} \tag{3.6}$$

$\tilde{\xi}^{t,j} = (\tilde{\xi}^{t-1}, \tilde{\xi}_t^j)$ and $\hat{h}_T \equiv 0$.

3.3 Two Policy Generation Methods

3.3.1 Linear Problems with Interstage Independence

In this section, we develop a procedure to generate feasible policies for the multi-stage stochastic linear program (2.20)-(2.21) when $\{\tilde{\xi}_t\}_{t=1}^T$ is interstage inde-

pendent. Our method works as follows: First, we construct a sample scenario tree, denoted Γ_c, using the common-samples method described in Section 3.2. Then, the instance of (3.5)-(3.6) associated with Γ_c is solved with the multi-stage L-shaped algorithm of Figure 2.2. When the algorithm stops, we obtain a policy whose expected cost is within $100 \cdot toler\%$ of optimal objective value for (3.5)-(3.6). We now describe how we use this solution to obtain a policy for the "true" problem (2.16)-(2.17).

When the algorithm of Figure 2.2 terminates, each $\text{sub}(t, i)$ contains the set of cut constraints generated during the solution procedure. Since Γ_c is constructed with the common-samples scheme, the sample subtrees rooted at the stage t nodes are all identical, i.e., the sample scenario tree Γ_c exhibits interstage independence. Thus, the cuts generated for a stage t node are valid for all other nodes in stage t. We will use the collection of cuts at each stage to construct a policy to problem (2.16)-(2.17).

Let $\vec{G}^i_{t,c}$ and $\vec{g}^i_{t,c}$ denote the cut-gradient matrix and cut-intercept vector for $\text{sub}(t, i)$ when the multi-stage L-shaped method terminates. Then, we define a stage t optimization problem used to generate the policy for (2.16)-(2.17) as follows:

$$\begin{aligned} \min_{x_t} \ & c_t x_t + \theta_t \\ \text{s.t.} \ & A_t x_t = b_t - B_t x_{t-1} \\ & -\vec{G}^i_{t,c} x_t + e\,\theta_t \geq \vec{g}^i_{t,c}, \ i = 1, \ldots, n_t \\ & x_t \geq 0, \end{aligned} \tag{3.7}$$

for $t = 2, \ldots, T$. For $t = 1$, (3.7) does not contain the term $B_1 x_0$ in the first set of constraint. A policy must specify what decision, $\hat{x}_t(\xi^t)$, to take at each stage t for a given ξ^t. Our policy computes $\hat{x}_t(\xi^t)$ by solving (3.7) with (A_t, B_t, b_t, c_t) specified by ξ^t, and with x_{t-1} determined by having already solved (3.7) under $\xi^{t'}$, $t' = 1, \ldots, t-1$. Such a policy is nonanticipative because when solving (3.7) the

Step 1	Construct a sample scenario tree Γ_c with the common-samples procedure (Section 3.2).
Step 2	Solve (3.5)-(3.6) based on Γ_c with the multi-stage L-shaped algorithm (Figure 2.2).
Step 3	When the algorithm stops (Step 3 of Figure 2.2), store the cut-gradient matrix, $\vec{G}^i_{t,c}$, and the cut-intercept vector, $\vec{g}^i_{t,c}$, associated with each $\text{sub}(t,i)$, $\forall t, i$.
Step 4	Do $t = 1$ to T Solve optimization problem (3.7) under ξ_t with x_{t-1} equal to $\hat{x}_{t-1}(\xi^{t-1})$, and denote its optimal solution $\hat{x}_t(\xi^t)$, where $\xi^t = (\xi^{t-1}, \xi_t)$.

Figure 3.1: A procedure to generate a feasible policy for a T-stage stochastic linear program with relatively complete recourse when $\{\tilde{\xi}_t\}_{t=1}^T$ is interstage independent.

process $\{\tilde{\xi}_t\}_{t=1}^T$ is known only through stage t. Relatively complete recourse ensures that $\hat{x}_t(\xi^t)$ will lead to a feasible decision in stages $t+1, \ldots, T$. The superscript on the cut-gradient matrix and the cut-intercept vector in (3.7) denotes the index of stage t node in Γ_c from which we obtain the cuts, and n_t is the total number of stage t nodes in Γ_c. So, if $\text{sub}(t,i)$ in Γ_c has K^i_t cuts then the total number of cuts in (3.7) is $\sum_{i=1}^{n_t} K^i_t$. We refer to this procedure as $\mathcal{P}_1$ and summarize it in Figure 3.1.

The solution procedure, as we have described it above, is a naive version of the multi-stage L-shaped method because it stores a separate set of cuts at each $\text{sub}(t,i)$ when solving (3.5)-(3.6) under Γ_c. Because Γ_c is interstage independent, we instead store a single set of cuts at each stage. This speeds the solution process and aids in eliminating redundant cuts when forming (3.7).

We have described the method for generating cuts at each stage by solving (3.5)-(3.6) under Γ_c exactly (or within $100 \cdot toler\%$) using the algorithm of Figure 2.2. However, this may be computationally expensive to carry out if Γ_c is large. If T is large but the number of descendants at each stage t node is "manageable" then

we could instead employ the algorithm of Pereira and Pinto [102], or better, its computationally-enhanced version due to Donohue [39]. See Section 2.4.2 for a description of these algorithms. We note that we do not pursue this possibility in our computational results.

Procedure $\mathcal{P}_1$ exploits convexity and interstage independence to generate feasible policies. Interstage independence plays a key role since the set of cuts generated as an approximation to $E[h_t(x_t, \tilde{\xi}^{t+1})|\tilde{\xi}^t = \xi^t]$ can also be used for $E[h_t(x_t, \tilde{\xi}_{t+1})|\tilde{\xi}^t = \xi'^t]$ when $\xi^t \neq \xi'^t$ because these two functions are identical. Generalizing $\mathcal{P}_1$ to handle problems with interstage dependency requires specifying how to adapt, or modify, cuts generated for $E[h_t(x_t, \tilde{\xi}_{t+1})|\tilde{\xi}^t = \xi^t]$ to another cost-to-go function conditioned on $\tilde{\xi}^t = \xi'^t$. For general types of dependency structures, this may be difficult (and so we develop a different approach in the next section). However, such adaptations of cuts is possible in the special case where $\{\tilde{\xi}_t\}_{t=1}^T$ consists of $\{(\tilde{c}_t, \tilde{A}_t, \tilde{B}_t, \tilde{\eta}_t)\}_{t=1}^T$, which is interstage independent and $\{\tilde{b}_t\}_{t=1}^T$ has the following dependency structure:

$$\tilde{b}_t = \sum_{j=1}^{t-1} (R_j^t \tilde{b}_j + S_j^t \tilde{\eta}_j) + \tilde{\eta}_t, \ t = 2, \ldots, T. \tag{3.8}$$

Here, R_j^t and S_j^t are given deterministic matrices with appropriate dimensions. (3.8) is a generalization of vector ARMA (autoregressive moving average) models; see, e.g., Tiao and Box [118]. With this probabilistic structure, Infanger and Morton [73] derive cut sharing formulae to be used in the L-shaped method. These results can be applied to modify Step 3 and 4 of $\mathcal{P}_1$. In Step 3, we store scenario-independent cut information, i.e., cut gradients, independent cut intercepts, and so-called cumulative expected dual vectors (see [73]) obtained from the multi-stage L-shaped algorithm in Step 2. Then, in Step 4, for a given ξ^t, scenario-dependent cuts in (3.7) can be computed using the analytical formulae of [73, Theorem 3].

3.3.2 Problems with Interstage Dependence

The method of Section 3.3.1 can handle linear programs with interstage independence, or a special type of dependence. In this section, we propose a different approach, which is computationally more demanding but allows for nonconvex problems with relatively complete recourse and general interstage dependency structures. In particular we consider the general T-stage stochastic program defined by (2.13)-(2.15) under assumptions (A1)-(A4) given in Section 2.3.1.

Our feasible policy construction for (2.13)-(2.15) works as follows: For a given ξ^t, we obtain $\hat{x}_t(\xi^t)$ by solving an approximating problem (from stage t to T) based on an independently-generated sample subtree, denoted by $\Gamma_r(\xi^t)$. Specifically, for a given ξ^t and x^{t-1}, $\Gamma_r(\xi^t)$ is constructed by the conditional sampling procedure described in Section 3.2 (either the common-samples or independent-samples method can be used). Then, $\hat{x}_t(\xi^t)$ is defined as an optimal solution of

$$\min_{x_t} \ \frac{1}{n(t)} \sum_{i \in D_t} \hat{f}_t(x^{t-1}, x_t, \Gamma_r(\tilde{\xi}^{t+1,i})), \tag{3.9}$$

where

$$\hat{f}_{\tau-1}(x^{\tau-1}, \tilde{\xi}^{\tau-1}, \Gamma_r(\tilde{\xi}^{\tau,j})) = \min_{x_\tau} \ \frac{1}{n(\tau,j)} \sum_{i \in D_\tau^j} \hat{f}_\tau(x^{\tau-1}, x_\tau, \tilde{\xi}^{\tau,j}, \Gamma_r(\tilde{\xi}^{\tau+1,i})),$$

$\tilde{\xi}^{\tau,j} = (\tilde{\xi}^{\tau-1}, \tilde{\xi}_\tau^j)$, $j \in D_{\tau-1}, \tau = t+1, \ldots, T-1$, and

$$\hat{f}_{T-1}(x^{T-1}, \tilde{\xi}^{T-1}, \Gamma_r(\tilde{\xi}^{T,j})) = \min_{x_T} \ f_T(x^{T-1}, x_T, \tilde{\xi}^{T,j}),$$

$\tilde{\xi}^{T,j} = (\tilde{\xi}^{T-1}, \tilde{\xi}_T^j)$, $j \in D_{T-1}$.

Our policy, which computes $\hat{x}_t(\xi^t)$ by solving (3.9), is nonanticipative. None of the decisions made at descendant nodes in stages $t+1, \ldots, T$ are part of the policy. Decisions in these subsequent stages (e.g., $t+1$) are found by solving another

Do $t = 1$ to T
Independently construct a sample subtree $\Gamma_r(\xi^t)$.
Solve approximating problem (3.9) with x^{t-1} equal to $\hat{x}^{t-1}(\xi^{t-1})$, and denote its optimal solution $\hat{x}_t(\xi^t)$, where $\xi^t = (\xi^{t-1}, \xi_t)$

Figure 3.2: A procedure to generate a feasible policy for a T-stage stochastic program with relatively complete recourse.

approximating problem (e.g., from stage $t+1$ to T) with an independently-generated sample tree. Similarly, the decisions at previous stages needed to find x^{t-1} are also computed using independently-generated sample trees. Relatively complete recourse ensures that $\hat{x}^t(\xi^t)$ will lead to feasible solutions in stages $t+1, \ldots, T$. We denote this policy-generation procedure by $\mathcal{P}_2$ and summarize it in Figure 3.2. Although $\mathcal{P}_2$ is applicable to a more general class of stochastic programs than $\mathcal{P}_1$, we still need a viable solution procedure to solve (3.9). In a non-convex variation of (3.9), finding its optimal solution can be computationally difficult.

3.4 Policy Cost Estimation

Under scenario $\tilde{\xi}^T$, the cost of using a given feasible policy, $\hat{x}^T(\tilde{\xi}^T)$, in (2.13)-(2.15) is $f_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T)$. Because this is a feasible, but not necessarily optimal policy, $Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T) \geq z^*$. In general, it is impossible to compute this expectation exactly. In this section, we describe a scenario-based method and a tree-based method to estimate $Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T)$. These estimation procedures can be carried out for any feasible policy but, when appropriate, we discuss specific implementation issues for policies generated by $\mathcal{P}_1$ and $\mathcal{P}_2$.

3.4.1 Scenario-based Estimator

Using $\mathcal{P}_1$ or $\mathcal{P}_2$ for scenario ξ^T, we can obtain a sequence of feasible solutions, $\hat{x}_1(\xi^1), \ldots, \hat{x}_T(\xi^T)$ (see Figures 3.1 and 3.2). The cost under scenario ξ^T is then given by $f_T(\hat{x}^T(\xi^T), \xi^T)$. In case of a T-stage stochastic linear program, this cost is

$$f_T(\hat{x}^T(\xi^T), \xi^T) = \sum_{t=1}^{T} c_t(\xi^t)\hat{x}_t(\xi^t). \tag{3.10}$$

Again, we emphasize that with both $\mathcal{P}_1$ and $\mathcal{P}_2$, $\hat{x}^T(\xi^T)$ is nonanticipative because when we carry out the procedures of Figures 3.1 and 3.2 to find $\hat{x}_t(\xi^t)$ the remaining portion of the scenario $(\xi_{t+1}, \ldots, \xi_T)$ is not used (in fact, in implementation it need not even be generated yet).

In order to form a point estimate of $Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T)$ whose error can be quantified, we generate ν iid observations of $\tilde{\xi}^T$, $\tilde{\xi}^{T,i}, i = 1, \ldots, \nu$. To form each $\tilde{\xi}^{T,i}$, observations of $\tilde{\xi}_t$ are sequentially drawn from the conditional distribution $F_t(\xi_t|\xi^{t-1,i})$ for $t = 2, \ldots, T$. Then, the sample mean estimator is

$$\bar{U}_\nu = \frac{1}{\nu}\sum_{i=1}^{\nu} f_T(\hat{x}^T(\tilde{\xi}^{T,i}), \tilde{\xi}^{T,i}). \tag{3.11}$$

Let S_u^2 be the standard sample variance estimator of $\operatorname{var} f_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T)$. Since $Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T) = E\bar{U}_\nu$,

$$P\left\{Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T) \leq \bar{U}_\nu + t_{\nu-1,\alpha}\frac{S_u}{\sqrt{\nu}}\right\} = P\left\{\frac{\sqrt{\nu}(\bar{U}_\nu - E\bar{U}_\nu)}{S_u} \geq -t_{\nu-1,\alpha}\right\},$$

where $t_{\nu-1,\alpha}$ denotes the $(1-\alpha)$-level quantile of a Student's t random variable with $\nu - 1$ degrees of freedom. By the central limit theorem for iid random variables,

$$\lim_{\nu\to\infty} P\left\{\frac{\sqrt{\nu}(\bar{U}_\nu - E\bar{U}_\nu)}{S_u} \geq -t_{\nu-1,\alpha}\right\} = 1 - \alpha.$$

Hence, for sufficiently large ν, we infer an approximate one-sided $100 \cdot (1 - \alpha)\%$ confidence interval for $Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T)$ of the form $(-\infty, \bar{U}_\nu + t_{\nu-1,\alpha}\frac{S_u}{\sqrt{\nu}}]$.

3.4.2 Tree-based Estimator

The scenario-based estimation procedure of the previous section generated ν iid observations of $\tilde{\xi}^T$. The estimation procedure in this section is instead based on generating ν iid sample scenario trees. Later, in Section 3.5, we turn to estimating a lower bound on z^*. That lower bound is based on sample scenario trees and can be combined with either the scenario- or tree-based estimators to establish the quality of a solution policy. As it will become apparent, the tree-based estimator in this section can be coupled with the lower-bound estimator in a manner not possible for the scenario-based estimator.

Let Γ be a sample scenario tree generated according to the conditional sampling framework of Section 3.2, and let n_T be the number of leaf nodes. Then, Γ may be viewed as a collection of scenarios, $\tilde{\xi}^{T,j}$, $j = 1, \ldots, n_T$, which are identically distributed but are not independent. An unbiased point estimate of $Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T)$ is given by

$$W = \frac{1}{n_T} \sum_{j=1}^{n_T} f_T(\hat{x}^T(\xi^{T,j}), \xi^{T,j}). \tag{3.12}$$

The numerical evaluation of $f_T(\hat{x}^T(\xi^{T,j}), \xi^{T,j})$, $j = 1, \ldots, n_T$, under policies $\mathcal{P}_1$ and $\mathcal{P}_2$ occurs in an identical manner to that described in Section 3.4.1.

To quantify the error associated with the point estimate in (3.12), we generate ν iid sample trees, $\Gamma^i, i = 1, \ldots, \nu$. Each of these iid trees is constructed according to the procedure described in Section 3.2 (again, under either the common-samples or independent-samples procedure). The number of scenarios in each Γ^i is again n_T, and the scenarios of Γ^i are $\xi^{T,ij}, j = 1, \ldots, n_T$. The point estimate under Γ^i is

$$W^i = \frac{1}{n_T} \sum_{j=1}^{n_T} f_T(\hat{x}^T(\tilde{\xi}^{T,ij}), \tilde{\xi}^{T,ij}), \tag{3.13}$$

$i = 1, \ldots, \nu$, and $W^i, i = 1, \ldots, \nu$, are iid by construction. So, the tree-based point estimate of $Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T)$ is

$$\bar{W}_\nu = \frac{1}{\nu} \sum_{i=1}^{\nu} W^i.$$

Let S_w^2 be the standard sample variance estimator of $\operatorname{var} W$. Because $E\bar{W}_\nu = EW = Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T)$, a confidence interval under the tree-based approach is constructed in a similar manner as in the scenario-based case, i.e., $(-\infty, \bar{W}_\nu + t_{\nu-1,\alpha} \frac{S_w}{\sqrt{\nu}}]$ is an approximate one-sided $100 \cdot (1-\alpha)\%$ confidence interval for $Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T)$.

3.5 Lower Bound Estimation

In this section, we develop a statistical lower bound for z^*, the optimal value of (2.13)-(2.15), and describe how to use this estimator to construct a confidence interval. Again, the motivation for forming such a confidence interval is to establish the quality of a feasible policy, including those generated by $\mathcal{P}_1$ or $\mathcal{P}_2$. The optimality gap of a policy with expected cost $Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T)$ is defined to be $Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T) - z^*$.

The lower bound estimator requires little structure on the underlying problem; therefore, we derive the lower bound using the notation of Section 2.3. First, we state the lower bound result for (2.13) when $T = 2$ in Lemma 3. In this case, (2.13) becomes a two-stage stochastic program with recourse, and the approximating problem, (3.2)-(3.4), reduces to

$$\hat{z}^* = \min_{x_1} \ \frac{1}{n_2} \sum_{i=1}^{n_2} f_1(x_1, \tilde{\xi}_2^i), \tag{3.14}$$

where

$$f_1(x^1, \xi_2^i) = \min_{x_2} \ f_2(x^1, x_2, \xi_2^i),$$

for $i = 1, \ldots, n_2$.

Lemma 3. *(Mak, Morton and Wood [91]) Let z^* be defined as in program (2.13) when $T = 2$ and $\hat{z}^*$ be defined as in program (3.14). If $\tilde{\xi}_2^1, \ldots, \tilde{\xi}_2^{n_2}$ satisfy*

$$E[f_1(x^1, \tilde{\xi}^2)|\xi^1] = E\Big[\frac{1}{n_2}\sum_{i=1}^{n_2} f_1(x^1, \tilde{\xi}^{2,i})|\xi^1\Big],$$

i.e., condition (3.1) with $t = 1$, then

$$z^* \geq E\hat{z}^*.$$

Proof. The lower bound is obtained by exchanging the order of expectation and optimization:

$$\begin{aligned} z^* &= \min_{x_1} E[f_1(x_1, \tilde{\xi}^2)|\xi^1] \\ &= \min_{x_1} E\Big[\frac{1}{n_2}\sum_{i=1}^{n_2} f_1(x_1, \tilde{\xi}^{2,i})|\xi^1\Big] \\ &\geq E\Big[\min_{x_1} \frac{1}{n_2}\sum_{i=1}^{n_2} f_1(x_1, \tilde{\xi}_{2,i})|\xi^1\Big] \\ &= E\hat{z}^*. \end{aligned}$$

□

Theorem 4. *Assume that (A1)-(A4) hold and let z^* and $\hat{z}^*$ be defined as in (2.13) and (3.2), respectively. If the sample tree $\Gamma(\xi^1)$ is constructed so that the observations of each descendant satisfy the unbiasedness condition (3.1) for $t = 1, \ldots, T-1$, then*

$$z^* \geq E\hat{z}^*,$$

i.e., the estimator $\hat{z}^*$ of z^* has a negative bias.

Proof. It suffices to show, for a given $\tilde{\xi}^\tau = \xi^\tau$, that

$$f_{\tau-1}(x^{\tau-1}, \xi^\tau) \geq E[\min_{x_\tau} \frac{1}{n(\tau)}\sum_{i\in D_\tau} \hat{f}_\tau(x^{\tau-1}, x_\tau, \tilde{\xi}^\tau, \Gamma(\tilde{\xi}^{\tau+1,i}))|\xi^\tau], \tag{3.15}$$

for $\tau = 1, ..., T-1$. Recursion (2.14) with $t = 0$ is $f_0(x^0, \xi^1) \equiv z^*$; hence, (3.15) is equivalent to the statement $z^* \geq E\hat{z}^*$ when $\tau = 1$. We proceed by induction, beginning with the base case, $\tau = T-1$. For a given $\tilde{\xi}^{T-1} = \xi^{T-1}$, $f_{T-2}(x^{T-2}, \xi^{T-1})$ is the optimal objective value of a two-stage stochastic program with recourse; therefore, by Lemma 3 and (3.4), the following relationship holds.

$$\begin{aligned} f_{T-2}(x^{T-2}, \xi^{T-1}) &\geq E[\min_{x_{T-1}} \frac{1}{n(T-1)} \sum_{i \in D_{T-1}} f_{T-1}(x^{T-2}, x_{T-1}, \tilde{\xi}^{T-1}, \tilde{\xi}^i_T) | \xi^{T-1}] \\ &= E[\min_{x_{T-1}} \frac{1}{n(T-1)} \sum_{i \in D_{T-1}} \hat{f}_{T-1}(x^{T-1}, \tilde{\xi}^{T-1}, \Gamma(\tilde{\xi}^{T,i})) | \xi^{T-1}], \end{aligned}$$

where $\tilde{\xi}^{T,i} = (\xi^{T-1}, \tilde{\xi}^i_T)$. For the inductive part, we show that if (3.15) holds for $\tau = t$ then (3.15) holds for $\tau = t-1$. For $\tau = t-1$, we express the left-hand side of (3.15) by using (2.14) for a particular descendant, say $\xi^{t-1,k} = (\xi^{t-2}, \xi^k_{t-1}), k \in D_{t-2}$, of node ξ^{t-2} as

$$\begin{aligned} & f_{t-2}(x^{t-2}, \xi^{t-1,k}) \\ &= \min_{x_{t-1}} E[f_{t-1}(x^{t-2}, x_{t-1}, \tilde{\xi}^t) | \xi^{t-1,k}] \\ &= \min_{x_{t-1}} E[\frac{1}{n(t-1,k)} \sum_{i \in D^k_{t-1}} f_{t-1}(x^{t-2}, x_{t-1}, \tilde{\xi}^{t,i}) | \xi^{t-1,k}] \\ &\geq \min_{x_{t-1}} E[\frac{1}{n(t-1,k)} \sum_{i \in D^k_{t-1}} E[\min_{x_t} \frac{1}{n(t,i)} \sum_{j \in D^i_t} \hat{f}_t(x^t, \tilde{\xi}^{t,i}, \Gamma(\tilde{\xi}^{t+1,j})) | \xi^{t,i}] | \xi^{t-1,k}]. \end{aligned} \tag{3.16}$$

We use the unbiasedness condition (3.1) and the fact that $\xi^{t,i} = (\xi^{t-1,k}, \xi^i_t)$ to obtain the second equality, and the inductive hypothesis that (3.15) holds for $\tau = t$ to obtain the last inequality.

The outer conditional expectation in the last line of (3.16) is taken with respect to all immediate descendant nodes $\tilde{\xi}^{t,i}, i \in D^k_{t-1}$ of a given node $\tilde{\xi}^{t-1,k} = \xi^{t-1,k}$, while the inner expectation is with respect to all the subtrees $\Gamma(\tilde{\xi}^{t+1,j}), j \in$

D_t^i, which are rooted at each of the descendants of a given node $\tilde{\xi}^{t,i} = \xi^{t,i}$. By combining these expectations and using recursion (3.3), we can write (3.16) as

$$\begin{aligned} & f_{t-2}(x^{t-2}, \xi^{t-1,k}) \\ & \geq \min_{x_{t-1}} E[\frac{1}{n(t-1,k)} \sum_{i \in D_{t-1}^k} \hat{f}_{t-1}(x^{t-2}, x_{t-1}, \tilde{\xi}^{t-1,k}, \Gamma(\tilde{\xi}^{t,i}))|\xi^{t-1,k}] \\ & \geq E[\min_{x_{t-1}} \frac{1}{n(t-1,k)} \sum_{i \in D_{t-1}^k} \hat{f}_{t-1}(x^{t-2}, x_{t-1}, \tilde{\xi}^{t-1,k}, \Gamma(\tilde{\xi}^{t,i}))|\xi^{t-1,k}], \end{aligned}$$

where the conditional expectation is with respect to all the subtrees $\Gamma(\tilde{\xi}^{t,i}), i \in D_{t-1}^k$, each of which roots at the immediate descendant of a given node $\tilde{\xi}^{t-1,k}$. Since the descendant node $\tilde{\xi}^{t-1,k}$ of node $\tilde{\xi}^{t-2}$ is arbitrarily chosen, the inequality

$$f_{t-2}(x^{t-2}, \xi^{t-1}) \geq E[\min_{x_{t-1}} \frac{1}{n(t-1)} \sum_{i \in D_{t-1}} \hat{f}_{t-1}(x^{t-2}, x_{t-1}, \tilde{\xi}^{t-1}, \Gamma(\tilde{\xi}^{t,i}))|\xi^{t-1}]$$

holds for any node in stage $t-1$. □

In summary, $\hat{z}^*$ is the optimal value of the approximating problem (3.2)-(3.4), and z^* is the optimal value of the original problem (2.13)-(2.15). Theorem 4 states that if the sample scenario tree associated with (3.2)-(3.4) is constructed so that its observations satisfy the unbiasedness condition (3.1) then $\hat{z}^*$ is an estimator of z^* with negative bias, i.e., $E\hat{z}^* \leq z^*$. In Section 3.6, we show how to use this result in conjunction with a given feasible policy to construct a confidence interval on its optimality gap.

To assess the error associated with estimator z^* we generate multiple replications of $\hat{z}^*$. In particular, we construct iid sample trees, $\Gamma^1, \ldots, \Gamma^\nu$, according to the procedure explained in Section 3.2, and then form the standard sample mean estimator as

$$\bar{L}_\nu = \frac{1}{\nu} \sum_{i=1}^{\nu} \hat{z}^{*,i},$$

where, for $i = 1, \ldots, \nu$,

$$\hat{z}^{*,i} = \min_{x_1} \frac{1}{n_2} \sum_{j=1}^{n_2} \hat{f}_1(x_1, \Gamma^i(\tilde{\xi}^{2,j})).$$

Let S_l^2 be the standard sample variance estimator of var $\hat{z}^*$. Since $z^* \geq E\hat{z}^* = E\bar{L}_\nu$,

$$\begin{aligned} P\left\{z^* \geq \bar{L}_\nu - t_{\nu-1,\alpha}\frac{S_l}{\sqrt{\nu}}\right\} &\geq P\left\{E\bar{L}_\nu \geq \bar{L}_\nu - t_{\nu-1,\alpha}\frac{S_l}{\sqrt{\nu}}\right\} \\ &= P\left\{\frac{\sqrt{\nu}(\bar{L}_\nu - E\bar{L}_\nu)}{S_l} \leq t_{\nu-1,\alpha}\right\}. \end{aligned}$$

By the central limit theorem for iid random variables, we infer, for sufficiently large ν, that $[\bar{L}(\nu) - t_{\nu-1,\alpha}\frac{S_l}{\sqrt{\nu}}, \infty)$ is an approximate one-sided $100 \cdot (1-\alpha)\%$ confidence interval for z^*.

3.6 Confidence Interval Construction

Confidence intervals on the optimality gap can be constructed in a number of ways depending on how the policy cost estimators developed in Section 3.4.1 and 3.4.2 and the lower bound estimator developed in Section 3.5 are used together. We suggest two approaches: separate and gap estimators.

3.6.1 Separate Estimator

Our first approach to form the confidence interval is by using a policy cost estimator and a lower bound estimator that are separately estimated. We have a choice of combining either the scenario-based or the tree-based estimator with the lower bound estimator. We begin with the case of the tree-based estimator, and denote the sampling errors associated with the tree-based estimator and the lower bound estimator by $\tilde{\epsilon}_w = t_{\nu-1,\alpha}\frac{S_w}{\sqrt{\nu}}$ and $\tilde{\epsilon}_l = t_{\nu-1,\alpha}\frac{S_l}{\sqrt{\nu}}$, respectively. From their confidence intervals, the probability of the events $\{\bar{L}_\nu - \tilde{\epsilon}_l \leq E\hat{z}^*\}$ and $\{Ef_T(\hat{x}^T, \tilde{\xi}^T) \leq \bar{W}_\nu + \tilde{\epsilon}_w\}$

are (individually) approximately $1-\alpha$. So, if the two events are independent then

$$P\left\{\bar{L}_\nu - \tilde{\epsilon}_l \leq E\hat{z}^*, Ef_T(\hat{x}^T, \tilde{\xi}^T) \leq \bar{W}_\nu + \tilde{\epsilon}_w\right\} \approx (1-\alpha)^2, \tag{3.17}$$

and if they are dependent, then

$$P\left\{\bar{L}_\nu - \tilde{\epsilon}_l \leq E\hat{z}^*, Ef_T(\hat{x}^T, \tilde{\xi}^T) \leq \bar{W}_\nu + \tilde{\epsilon}_w\right\} \approx (1-2\alpha). \tag{3.18}$$

The dependent case follows from the Boole-Bonferroni inequality (see, e.g., Law and Kelton [86]), and can arise, for instance, if policy $\hat{x}^T$ is constructed from the solutions of (3.2)-(3.4), which are used in forming the lower bound estimator. $\mathcal{P}_1$ and $\mathcal{P}_2$ generate policies without using such techniques, and so we focus on the former case. We know that $E\hat{z}^* \leq z^* \leq Ef_T(\hat{x}^T, \tilde{\xi}^T)$, and so

$$\begin{aligned} & P\left\{\bar{L}_\nu - \tilde{\epsilon}_l \leq E\hat{z}^*, Ef_T(\hat{x}^T, \tilde{\xi}^T) \leq \bar{W}_\nu + \tilde{\epsilon}_w\right\} \\ \leq \quad & P\left\{(\bar{W}_\nu - \bar{L}_\nu)^+ + \tilde{\epsilon}_l + \tilde{\epsilon}_w \geq Ef_T(\hat{x}^T, \tilde{\xi}^T) - z^*\right\}, \end{aligned}$$

where $(\,\cdot\,)^+ = \max\{\cdot, 0\}$. From this, we can infer that $[0, (\bar{W}_\nu - \bar{L}_\nu)^+ + \tilde{\epsilon}_l + \tilde{\epsilon}_w]$ is an approximate confidence interval at level $(1-\alpha)^2$ level for the independent case. The scenario-based upper bound estimator can also be combined with the lower bound estimator to form a confidence interval on the optimality gap in a similar manner. To do so, we just replace $\bar{W}_\nu$ and $\tilde{\epsilon}_w$ with $\bar{U}_\nu$ and $\tilde{\epsilon}_u = t_{\nu-1,\alpha}\frac{S_u}{\sqrt{\nu}}$, respectively, in the development above.

3.6.2 Gap Estimator

A natural way to form the confidence interval on the optimality gap is to combine the tree-based and lower bound estimator that are formed through the same set of sample trees. In simulation, this approach is referred to as using the common random numbers and is used to reduce variance (see, e.g., Law and Kelton [86]).

Problems	Interstage indep.	T	n_T	Average size of stage t LP		
				Rows	Columns	Non-zeros
STFOR	yes	7	512	18	17	35
DVA	yes	4	1.4×10^{331}	104	446	875
WATSON	no	10	512	34	60	113

Table 3.1: The characteristics of three test problems used in the computational experiment.

We define a gap estimator for a given sample tree, Γ, by

$$G = W - \hat{z}^*.$$

Note that $G \geq 0$, wp1, and

$$EG = Ef_T(\hat{x}^T, \tilde{\xi}^T) - E\hat{z}^* \geq Ef_T(\hat{x}^T, \tilde{\xi}^T) - z^* \geq 0.$$

Thus, we can form the confidence interval on the optimality gap by generating sample trees, $\Gamma^i, i = 1, \ldots, \nu$, and estimate EG by the standard sample mean estimator given by

$$\bar{G}_\nu = \frac{1}{\nu} \sum_{i=1}^{\nu} G^i,$$

where $G^i = W^i - \hat{z}^{*,i}$ is formed by using Γ^i. Let S_g^2 be the standard sample variance estimator of $\operatorname{var} G$. Again, for sufficiently large ν, we can infer from the central limit theorem for iid random variables that $[0, \bar{G}_\nu + \tilde{\epsilon}_g]$ is an approximate $(1-\alpha) \cdot 100$ % confidence interval for $Ef_T(\hat{x}^T, \tilde{\xi}^T) - z^*$, where $\tilde{\epsilon}_g = t_{\nu-1,\alpha} \frac{S_g}{\sqrt{\nu}}$.

3.7 Computational Results

We present computational results of the proposed procedures on three test problems from the literature; all three are multi-stage stochastic linear programs. The characteristics of the test problems are given in Table 3.1. The first prob-

lem, STFOR, is a stochastic forest harvesting model from Gassmann [56]. The second problem, DVA, is a variation of the dynamic vehicle allocation model of Chueng and Powell [29], and is built on a network flow model with side constraints. Donohue [39] adapts the original model to allow the demand to be backlogged and the cost to be stochastic, and provides a modified model-generation procedure in which the problem's parameters and the distribution of the random vector are randomly generated. We use Donohue's procedures to create an instance of DVA. For ease of solution, we relax the integrality requirement. The third problem, WATSON, is an instance of the asset-liability management model developed by Dempster [36]. Unlike STFOR and DVA, WATSON does not have interstage independence. A collection of WATSON test problems in SMPS format [14, 58] is available at `www-cfr.jims.cam.ac.uk/research/stprog.html`. Two scenario files (SMPS stoch file) are provided for each instance of WATSON; one is constructed from an independent scenario generation method, and the other from the conditional scenario generation method. Both methods generate a scenario tree that is interstage dependent. Description of these methods are provided at the web address above. We use an independent-scenario version of WATSON because the earlier computational results (also available at the web address above) show that its expected value of perfect information is larger than that of the conditional-scenario version.

Although the 512-scenario STFOR and WATSON problems can be solved to optimality with the multi-stage L-shaped method, we apply $\mathcal{P}_1$ and $\mathcal{P}_2$ to generate feasible policies in order to verify our procedures. In addition, we choose to use STFOR and WATSON with 512 scenarios in our computational experiments so that we can asymptotically test our procedures, $\mathcal{P}_1$ and $\mathcal{P}_2$. So, for STFOR and WATSON, we construct sample trees with the same 512 scenarios as those in the underlying scenario tree, but assign probability weight of each node to an empirical

one obtained from sampling. In this way, we can increase the sample size as large as we would like without increasing the computational effort needed to solve the approximating problem.

Balanced and uniform sample trees are used through out our testing, i.e., trees which have the same number of descendants at every node. We adopt the notation in Consigli et al. [31] to denote the tree structure, e.g., 4^6 denotes a 7-stage scenario tree in which every node has four descendants. In addition, the letter "c" or "i" is appended to refer to the common- and independent-samples methods (Section 3.2), respectively. All computational results in this chapter are performed on a Dell Precision 530 (two 1.8 GHz processors) with 1GB of memory running SuSE Linux 7.3 operating system. The multi-stage L-shaped method is implemented with C programming language and CPLEX 7.5 callable library; all are compiled by GCC 2.95.3. Throughout our experiment, we only use one processor.

Table 3.2 reports computational results for STFOR under policy generation method $\mathcal{P}_1$, using separate estimators (Section 3.6.1) for the tree-based policy-cost estimator and the lower bound estimator. Column (1) indicates the size of the scenario tree Γ_c, either 4^6c or 10^6c, used in forming the policy. The size and type of the sample trees $\Gamma^i, i = 1, \ldots, 30$, used in evaluating policy quality are given in column (2). The confidence intervals on the optimality gap are specified in column (6). There three sources of errors contributing to the width of the confidence interval: (i) the bias of the lower bound estimator, (ii) the suboptimality of the policy, and (iii) the sampling error. Estimates of each error are given in columns (3), (4), and (5), respectively. We can compute estimates for (i) and (ii) because we can solve STFOR exactly for z^*. Note that the estimate of (ii) in the first three rows of Table 3.2, when the size of Γ_c is 10^6c, is negative, i.e., $\bar{W}_{30}$ is smaller than z^*. This is because the small-sized evaluation trees Γ^i ($2^6c, 2^6i$ and 4^6c) have a large sampling

errors relative to $E\bar{W}_{30} - z^*$, which is small because the large policy-generation tree ($10^6 c$) yields a near-optimal policy. Column (7) gives the confidence interval width as a percentage of z^*. Column (8) gives the CPU time in seconds for the computation of each confidence interval, including time required to generate the policy.

In Table 3.2, three independent random number streams are used for: (i) Γ_c, (ii) $\Gamma^i, i = 1, \ldots, 30$, used to compute $\bar{W}_{30}$, and (iii) $\Gamma^i, i = 1, \ldots, 30$, used to compute $\bar{L}_{30}$. The same random number stream is used to compute $\bar{L}_{30}$ when the size of Γ_c is $4^6 c$ and $10^6 c$; therefore, the bias results in the top and bottom half of Table 3.2 are identical. Similarly, the same set of trees $\Gamma^i, i = 1, \ldots, 30$, are used to compute $\bar{W}_{30}$ when the size of Γ_c is $4^6 c$ and $10^6 c$. So, the smaller values in the bottom half of the table are primarily due to increased quality of the policy due to the larger policy-generation tree.

Table 3.3 lists similar numerical results to Table 3.2, but the confidence intervals are constructed via gap estimators (Section 3.6.2), i.e., the same set of trees, $\Gamma^i, i = 1, \ldots, 30$, are used for both $\bar{L}_{30}$ and $\bar{W}_{30}$. The random number streams to generate Γ_c and $\Gamma^i, i = 1, \ldots, 30$ for Table 3.3 are the same as the ones used for Γ_c and $\bar{W}_{30}$ in Table 3.2. The gap point estimate $\bar{G}_{30}$ may be obtained by adding columns (3) and (4) (see Section 3.6.2). Column (9) in Table 3.3, denoted "VR", is computed by $(\frac{\tilde{\epsilon}_w + \tilde{\epsilon}_l}{\tilde{\epsilon}_g})^2$, where $\tilde{\epsilon}_w + \tilde{\epsilon}_l$ is from Table 3.2 and $\tilde{\epsilon}_g$ from Table 3.3. This ratio indicates the approximate number of times that ν must increase in Table 3.2 to have the sampling errors in Table 3.2 and 3.3 be in the same order of magnitude. The CPU times in Table 3.3 are consistently smaller than those in Table 3.2.

From Table 3.2 and 3.3, we observe that the quality of feasible policies generated by $\mathcal{P}_1$ improves as the sample size of Γ_c increases. The bias in column (3) decreases as the size of Γ^i increases. Independent-samples method usually gives lower

Γ_c	Γ^i	$z^* - \bar{L}_{30}$	$\bar{W}_{30} - z^*$	$\tilde{\epsilon}_w + \tilde{\epsilon}_l$	95% CI	% z^*	min.
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
	2^6c	781.42	650.92	2107.08	[0, 3539.42]	8.07	9
	2^6i	672.53	417.63	1114.77	[0, 2204.93]	5.03	9
	4^6c	719.96	262.67	985.38	[0, 1968.01]	4.49	10
4^6c	4^6i	384.01	784.77	580.22	[0, 1749.00]	3.99	10
	10^6c	143.90	684.68	714.64	[0, 1543.22]	3.52	11
	10^6i	233.42	496.40	260.52	[0, 990.34]	2.26	11
	50^6c	116.53	564.09	266.40	[0, 947.02]	2.16	12
	50^6i	41.22	540.17	155.08	[0, 736.47]	1.68	13
	2^6c	781.42	-91.41	797.03	[0, 1487.04]	3.60	13
	2^6i	672.53	-41.72	681.84	[0, 1249.65]	2.94	14
	4^6c	719.96	-22.02	602.93	[0, 1300.87]	3.02	14
10^6c	4^6i	384.01	109.84	327.75	[0, 821.60]	1.87	15
	10^6c	143.90	128.52	424.16	[0, 696.58]	1.59	15
	10^6i	233.42	50.23	161.95	[0, 445.60]	1.02	16
	50^6c	116.53	80.45	164.56	[0, 361.54]	0.82	16
	50^6i	41.22	74.27	82.60	[0,198.09]	0.45	17

Table 3.2: Computational results for STFOR ($z^* = -43868.93$): 95% confidence interval on the optimality gap of feasible policies constructed by using $\mathcal{P}_1$. Each confidence interval is formed by separate estimators of the policy cost and the lower bound with $\nu = 30$.

sampling error than the common-samples one (the only exceptions are row $10^6c/2^6$ and $10^6c/50^6$). The sampling error is further reduced when the gap estimator is used due to the variance reduction effect of the common random numbers technique.

In further experiments, we use the scenario-based estimator for the expected policy cost in place of tree-based estimators reported in Table 3.2. We do not report these results in detail, but only indicate that obtaining a similar value of $\tilde{\epsilon}_w$ can be accomplished with a slightly smaller ν. However, the associated CPU time (again, to achieve similar value of $\tilde{\epsilon}_w$) is substantially larger than that associated with the tree-based estimator. This is due to an increased number of subproblems needed to be solved to form the scenario-based policy-cost estimator.

Γ_c	Γ^i	$z^* - \bar{L}_{30}$	$\bar{W}_{30} - z^*$	$\tilde{\epsilon}_g$	95% CI	% z^*	min.	VR
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
	2^6c	923.87	650.92	1430.17	[0, 3004.96]	6.85	8	2
	2^6i	900.99	417.63	621.07	[0, 1939.69]	4.42	8	3
	4^6c	640.90	262.67	406.36	[0, 1309.93]	2.99	9	6
4^6c	4^6i	308.05	784.77	287.56	[0, 1380.38]	3.15	9	4
	10^6c	193.79	684.68	299.41	[0, 1177.88]	2.68	10	6
	10^6i	136.83	496.40	90.94	[0, 724.17]	1.65	10	8
	50^6c	6.17	564.09	100.08	[0, 670.34]	1.53	11	7
	50^6i	12.81	540.17	60.23	[0, 613.21]	1.40	12	7
	2^6c	923.87	-91.41	188.92	[0, 1021.38]	2.33	13	18
	2^6i	900.99	-41.72	205.63	[0, 1064.90]	2.43	13	9
	4^6c	640.90	-22.02	123.65	[0, 742.53]	1.69	14	24
10^6c	4^6i	308.05	109.84	60.73	[0, 478.62]	1.09	14	29
	10^6c	193.79	128.52	82.48	[0, 404.79]	0.92	14	26
	10^6i	136.83	50.23	26.87	[0, 213.93]	0.49	15	36
	50^6c	6.17	80.45	17.66	[0, 104.28]	0.24	16	87
	50^6i	12.81	74.27	19.36	[0, 106.44]	0.24	16	18

Table 3.3: Computational results for STFOR ($z^* = -43868.93$): 95% confidence intervals on the optimality gap of feasible policies constructed by using procedure $\mathcal{P}_1$. Each confidence interval is formed by the gap estimator with $\nu = 30$ ($\bar{L}_{30}$ and $\bar{W}_{30}$ are based on the same 30 sample trees).

Based on the computational results for STFOR, we only use the gap estimator in DVA and WATSON to construct 95% confidence intervals, and $\Gamma^i, i = 1, \ldots, \nu$, are constructed by the independent-samples method. Table 3.4 reports computational results for DVA. Policy generation is again done by procedure $\mathcal{P}_1$ with the size of Γ_c indicated in column (1) and that of $\Gamma^i, i = 1, \ldots, 30$, in column (2). Columns (3)-(5) contain the gap point estimate, sampling error, and resulting 95% confidence interval. In DVA, we cannot separate the bias and quality of the policy in the gap estimate since we cannot solve DVA exactly for z^*. The width of the confidence interval as a percentage of the magnitude of the policy cost estimator is given in

column (6). The CPU time is given in minutes in column (7). Again, this CPU time includes time required to generate policy. Due to the independent-samples method of tree construction and the common random numbers effect in the gap estimator, the sampling error in DVA is, again, relatively small compared to the magnitude of the gap estimator. Overall, the computational results for DVA are qualitatively similar to those of STFOR discussed previously.

Γ_c	Γ^i	$\bar{G}_{30}$	$\tilde{\epsilon}_g$	95% CI	% $\lvert\bar{W}_{30}\rvert$	min.
(1)	(2)	(3)	(4)	(5)	(6)	(7)
	2^3i	8183.86	428.09	[0, 8611.95]	3.74	11
7^3c	4^3i	5821.40	270.06	[0, 6091.46]	2.66	39
	7^3i	4668.25	121.69	[0, 4790.14]	2.08	178
	10^3i	4202.30	107.88	[0, 4310.18]	1.88	542
	2^3i	6826.28	545.25	[0, 7371.53]	3.18	30
15^3c	4^3i	4335.62	217.13	[0, 4552.75]	1.98	61
	7^3i	3147.82	106.08	[0, 3253.90]	1.41	205
	10^3i	2664.00	103.50	[0, 2767.50]	1.20	541

Table 3.4: Computational results for DVA: 95% confidence intervals on the optimality gap of feasible policies constructed by using procedure $\mathcal{P}_1$. Each confidence interval is formed by the tree-based gap estimator with $\nu = 30$.

Table 3.5 gives computational results for the WATSON test problem, which does not have interstage independence. Hence, we can only use $\mathcal{P}_2$ to form a policy. One key issue in forming the policy is the size of the subtree $\Gamma_r(\xi^t)$ constructed at each node in each stage to generate $\hat{x}(\xi^t)$ to form a policy (see Section 3.3.2). In Table 3.5, we construct $\Gamma_r(\xi^t)$ such that it has the same number of leaf nodes for all $\xi^t, t = 1, \ldots, T-1$, except for $\Gamma_r(\xi^T)$. Two sizes of $\Gamma_r(\xi^t)$ are used and these are denoted "Method (A)" and "Method (B)" in Table 3.5 and Table 3.6. The structure of subtree $\Gamma_r(\xi^t)$ is given in Table 3.6. To read Table 3.6, for example, a subtree $\Gamma_r(\xi^{\,8})$ rooted at a node in stage 8 has a constant number of descendants of 4^5 and 4^4 for a stage 8 and a stage 9 node, respectively, under method (A). The bias, sampling

error, and quality of the policy associated with WATSON computational results in Table 3.5 are also qualitatively similar to those of STFOR and DVA.

Subtree size	Γ^i	$z^* - \bar{L}_{30}$	$\bar{W}_{30} - z^*$	$\tilde{\epsilon}_g$	95% CI	% z^*	min.
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
	$2^9 i$	309.86	70.66	53.45	[0, 433.97]	22.15	98
method	$4^9 i$	241.60	-40.94	29.50	[0, 230.16]	11.75	142
(A)	$10^9 i$	39.29	55.19	8.76	[0, 103.24]	5.27	183
	$50^9 i$	5.64	31.46	2.53	[0, 39.63]	2.02	185
	$2^9 i$	309.86	37.68	50.37	[0, 397.91]	20.31	120
method	$4^9 i$	241.60	-63.11	23.41	[0, 201.90]	10.30	165
(B)	$10^9 i$	39.29	49.69	9.00	[0, 97.98]	5.00	207
	$50^9 i$	5.64	25.03	2.13	[0, 32.80]	1.67	208

Table 3.5: Computational results for WATSON ($z^* = -1959.64$): 95% confidence intervals on the optimality gap of feasible policies constructed by using procedure $\mathcal{P}_2$. Each confidence interval is formed by the gap estimator with $\nu = 30$, i.e., $\bar{L}_{30}$ and $\bar{W}_{30}$ are based on the same 30 sample trees. The sample size for subtrees in $\mathcal{P}_2$ is determined by (A) and (B), whose details are given in Table 3.6.

Stage of the subtree root node	Number of samples in each stage method (A)	method (B)
1	$4, \ldots, 4$	$10, \ldots, 10$
2	$4^2, 4, \ldots, 4$	$10^2, 10, \ldots, 10$
3	$4^3, 4, \ldots, 4$	$10^3, 10, \ldots, 10$
4	$4^3, 4^2, 4, \ldots, 4$	$10^3, 10^2, 10, \ldots, 10$
5	$4^3, 4^3, 4, \ldots, 4$	$10^3, 10^3, 10, \ldots, 10$
6	$4^3, 4^3, 4^2, 4$	$10^3, 10^3, 10^2, 10$
7	$4^3, 4^3, 4^3$	$10^3, 10^3, 10^3$
8	$4^5, 4^4$	$10^5, 10^4$
9	4^6	10^6

Table 3.6: Sample size used for generating subtrees in procedure $\mathcal{P}_2$ for WATSON test problem.

Chapter 4

Bias Reduction Techniques

4.1 Introduction

In Chapter 3, we propose two procedures to generate feasible policies for two classes of multi-stage stochastic programs. We use a lower bound estimator (of a minimization problem) to establish the quality of these policies by constructing approximate confidence intervals on their optimality gaps. In our procedures, the width of the confidence interval on the optimality gap is determined by three factors: (i) the quality of the policy, (ii) the bias of the lower bound estimator, and (iii) the sampling error. It is desirable that the contribution from (ii) and (iii) to the confidence interval's width be as small as possible so that when a given feasible policy is tested, we can effectively determine its quality by the width of the confidence interval. From our computational experiments on three test problems from the literature in Chapter 3, we observe that the contribution of the sampling error to the width is relatively small, especially when we construct the confidence interval using the variance reduction technique known as common random numbers. In addition, there are other well-known variance reduction techniques that have been successfully applied in Monte Carlo sampling-based methods for stochastic programs to control the sampling error. We give a literature review of such variance reduction techniques in Section 2.4.2.

The contribution from the bias, on the other hand, is relatively large in our three test problems in Chapter 3, and can be, in general, large for multi-stage

stochastic programs. This is true in the two-stage setting as shown by the numerical results reported in [87, 91, 95, 120]. Although the bias, under appropriate assumptions, tends to zero as the number of scenarios in the sample tree grows, the computational effort required to solve a multi-stage approximating problem increases exponentially with tree size. As a result, we are motivated to try to reduce the bias associated with the lower bound estimator for a given computational resource. The motivation is analogous to that of developing an effective variance reduction procedure for a limited computational resource. Effective variance and bias reduction will result in a confidence interval whose width more closely reflects the quality of the policy being tested. Throughout this chapter, we will refer to the lower-bound estimator (of a minimization problem) developed in Chapter 3 as the optimistic-bound estimator to avoid confusion when we develop our bias reduction procedure for an American-style option pricing problem, which is generally formulated as a maximization problem. In such a situation, the bias is instead associated with an upper-bound estimator.

In this chapter, we attempt to reduce bias in multi-stage stochastic optimization problems by the way we construct sample scenario trees. In Chapter 3, we use uniform sample trees in our computation, i.e., sample trees that have the same number of descendants at every non-leaf node. The procedure we propose generates sample trees with varying number of descendants, depending on an estimate of each node's contribution to the total bias. The resulting non-uniform tree is designed to reduce the bias of the optimistic-bound estimator associated with that tree. Our method for estimating bias at a node and allocating samples to reduce this bias extends earlier work of Korapaty [85]. We give a literature review on scenario tree approximation methods in Section 2.4.2.

We begin by characterizing the bias of the optimistic-bound estimator for a

relatively simple multi-stage stochastic optimization problem known as an American-style option pricing problem. In this setting, the optimisic bound estimator can be computed by applying dynamic programming (DP) to solve an approximating problem defined on a sample tree as proposed by Broadie and Glasserman [18]. The main drawback of their method is that the computational effort grows exponentially with the number of exercise opportunities; thus, they propose a stochastic mesh method [19, 20] in which the computational effort is linear both in the number of exercise opportunities and DP state variables. Nonetheless, we adopt the DP-based approach of Broadie and Glasserman [18] for our research since it gives us insights on bias characterization that can be useful in handling more complex multi-stage stochastic programs.

Instead of using a sample tree in the DP-based procedure, we use a "state-based" sample tree, which based on the notion of DP state variables. This is appropriate in our development because we derive a bias estimate at all (discretized) values of the DP state variables. We develop this procedure in detail in Section 4.2. In Section 4.3, we extend our results to a stochastic lot-sizing problem in which the optimistic-bound estimator is also computed using DP and state-based sample trees. Computational results on American-style option pricing problem, and the stochastic lot-sizing problem are reported in Section 4.4.

4.2 Reducing Bias in Pricing American-style Options

4.2.1 Problem Statement

A call option on an asset gives its holder the right to buy the asset at a set price K called the strike price. The asset price is a stochastic process denoted $\{\tilde{S}_t\}_{t=0}^{T}$. If the asset price is higher than the strike price at the time t of exercise, the holder makes a profit of $S_t - K$. If the asset price is lower than the strike price,

the holder does not need to exercise the option, and the option value is zero. The payoff of the call option is then $(S_t - K)^+$ where $x^+ \equiv \max\{0, x\}$. A European-style option allows the holder to exercise only at the time of maturity T; thus, its payoff is just $(S_T - K)^+$, while an American-style option allows exercise any time before maturity. The price of an American call option is given by

$$C = \max_{\tau \leq T} E[e^{-r\tau}(\tilde{S}_\tau - K)^+], \tag{4.1}$$

where r is the riskless rate of interest and E is the expectation with respect to the so-called risk-neutral measure. The optimization is carried out over all stopping times τ that are less than T. This is a continuous-time optimization problem, but we will instead consider the case when there are a finite number of exercise opportunities, i.e., $t \in \{t_0 = 0, t_1, \ldots, t_d = T\}$, where $t_0 < t_1 < \cdots < t_d$. The discrete-time problem can be viewed either as an approximation of the continuous-time pricing problem or as its own problem known as a Bermudan option. In addition, under certain assumptions regarding dividend payments, the continuous-time problem simplifies to a discrete-time problem. See Broadie and Glasserman [18, §4.2] for more details and for other types of option pricing problems that have a finite number of exercise opportunities. For notational simplicity, we sometimes index with i, instead of t_i, a variable associated with stage t_i. For example, the asset price at stage t_i can be written as either S_i or S_{t_i}.

By assuming that the asset price process $\{\tilde{S}_t\}_{t=0}^T$ is Markovian, i.e., the future evolution only depends on the current asset price S_t but not on the past values, $\{S_{t'}\}_{t'=0}^{t-1}$, we can express the discrete-time problem using DP recursions with the asset price as a state variable as follows. Given that the asset price $\tilde{S}_t = S_t$ at stage t, the decision to be made is either to exercise or to continue to hold the option in order to maximize the conditional expected option value at stage t. Let

$f_t(S_t)$ denote the option value at time t given that the asset price is S_t, i.e.,

$$f_t(S_t) = \max\{h_t(S_t), g_t(S_t)\}, \tag{4.2}$$

where $g_t(S_t)$ is the expected continuation value of the option given by

$$g_t(S_t) = E[e^{-r_{t+1}} f_{t+1}(\tilde{S}_{t+1})|\tilde{S}_t = S_t], \tag{4.3}$$

and $h_t(S_t)$ is the payoff from immediate exercise given by

$$h_t(S_t) = (S_t - K)^+. \tag{4.4}$$

The terminal value is $f_T(S_T) = h_T(S_T)$. Under risk-free interest rate r, the discount factor between stage t_i and t_{i+1} is $e^{-r_{t+1}} \equiv e^{-r(t_{i+1}-t_i)}$. For a given initial (deterministic) asset price S_0, the pricing problem is to find $f_0(S_0)$.

The expectation in (4.3) is taken with respect to the price process $\{\tilde{S}_t\}_{t=1}^T$, which is governed by a risk-neutral measure derived by applying no-arbitrage arguments to a geometric Brownian motion price process. For $\tilde{S}_{t_i} = S_{t_i}$, the resulting risk-neutral price process is given by

$$\tilde{S}_{t_i} = S_{t_{i-1}} \exp((r - \delta - \sigma^2/2)(t_i - t_{i-1}) + \sigma\sqrt{t_i - t_{i-1}}\ \tilde{Z}_{t_i}), \tag{4.5}$$

for $i = 1, \ldots, d$, where $\tilde{Z}_i$ is a standard normal random variable, δ is the dividend rate, and $\sigma > 0$ is the volatility. The price process in (4.5) is Markovian.

The problem as described is a well-solved problem. Geske and Jonhnson [60] give an analytical solution to this discrete-time pricing problem for a single underlying asset. Our goal in using this problem is to gain insight on how to construct sample scenario trees in order to reduce bias, in the hope that this insight can be generalized to more complex problems. That the true value is known for pricing Bermudan options helps in assessing the ability of our procedures to accomplish our goal in a simple setting.

Broadie and Glasserman [18] propose a sampling-based methodology to construct a confidence interval on the optimality gap of an exercise policy for this discrete-time pricing problem. In their approach, a lower bound on $f_0(S_0)$ is obtained via the suboptimality of a particular exercise policy. After repeatedly simulating this exercise policy, a lower bound estimator is formed by the standard sample mean of the simulated option price. The optimistic-bound estimator is formed by the standard sample mean of the option price computed by applying DP on scenario trees generated via Monte Carlo sampling. Then, a confidence interval on the optimality gap is constructed from the lower and optimistic-bound estimators. Uniform sample trees are used in [18] to compute the optimistic-bound estimator in much the same way as in our procedure developed in Chapter 3.

In Section 4.2.2, we describe the underlying DP solution procedure applied to a state-based sample scenario tree (instead of the standard sample scenario trees used in Chapter 3, and in [18]). In an attempt to reduce the optimistic-bound estimator's bias, we derive in Section 4.2.3 an analytical approximation for the bias that arises when using sample scenario trees, and use the result to allocate sample sizes to build a non-uniform sample scenario tree in Section 4.2.4.

4.2.2 Dynamic Programming Solution Procedure

Broadie and Glasserman [18] construct sample trees in the same manner that we describe in Section 3.2. In particular, let $F_t(S_t|S_{t-1})$ be the conditional distribution governing $\tilde{S}_t$. Using the scenario tree notation of Section 2.3.2, we generate $n(0)$ iid observations of $\tilde{S}_1$ from $F_1(S_1|S_0)$ at stage t_0. Then, we form descendants of each stage t_1 observation $S_1^i, i = 1, \ldots, n_1 = n(0)$, by drawing $n(1,i)$ conditionally iid observations of $\tilde{S}_2$ from $F_2(S_2|S_1^i), i = 1, \ldots, n_1$. The total number of stage 2 nodes is $n_2 = \sum_{i=1}^{n_1} n(1,i)$. The process continues until all stage T

observations are generated. The computational effort of solving a DP on this type of sample tree grows exponentially with the number of exercise opportunities, but not with the number of assets or state variables.

Instead of using the above approach, we generate what we call a *state-based sample tree*, which is built on a discretized grid of DP state variables. In the American-style option pricing problem, we use the asset price as the state variable. The support of $\tilde{S}_{t_i}$ in (4.5) is discretized as follows: we replace $\tilde{Z}_{t_i}$ with a truncated standard normal random variable with support $[-\Phi^{-1}(c), \Phi^{-1}(c)]$, where Φ is the cumulative distribution function of the standard normal (in our computation, c takes a value of 5). This induces a bounded support for $\tilde{S}_{t_i}$. Let m_t be the number of stage t cells. We partition the (bounded) support of $\tilde{S}_{t_i}$ into cells $[G^j_{t_i}, G^j_{t_i} + \Delta S], j = 1, \ldots, m_{t_i}$.

Given this discretization, a state-based tree is constructed as follows. For stage t_0, generate $n(0)$ iid observations of $\tilde{S}_1$ from $F_1(S_1|S_0)$. For stage t_1 cell i, generate $n(1,i)$ conditionally iid observations of $\tilde{S}_2$ from $F_2(S_2|\tilde{S}_1 = G^i_1 + \Delta S), i = 1, \ldots, m_1$. Then, continue until the stage T observation is generated. Typically, one would use the midpoint of a cell to condition on in generating observations for the next stage. We instead condition on $G^i_1 + \Delta S$ from cell $[G^i_1, G^i_1 + \Delta S]$ because this ensures that the resulting estimator maintains its optimistic (high) bias. (As in Chapter 3, this is key to our constructing confidence intervals on solution quality.) For the same reason, we use $G^i_1 + \Delta S$ as the "representation point" from cell $[G^i_1, G^i_1 + \Delta S]$ when evaluating g_t and h_t (see (4.3) and (4.4)) in the DP solution of the state-based tree. Clearly, the magnitude of the error induced depends on grid size ΔS and we investigate this issue in our computational results of Section 4.4.1. The computational effort of solving a DP on such a state-based tree grows exponentially with the number of states, but linearly with the number of

exercise opportunities.

Similar to the construction of a sample scenario tree in Section 3.2, we need m_{t-1} sets of iid standard normal observations of $\tilde{S}_t$ from $F_t(S_t|\tilde{S}_{t-1} = G_{t-1} + \Delta S), t = 2, \ldots, T$, to form a tree. As described in Section 3.2, we can use two schemes: independent-samples and common-samples. In the independent-samples scheme, each set of $n(t-1,i)$ iid standard normal observations are generated independently for $i = 1, \ldots, m_{t-1}$, $t = 2, \ldots, T$. In the common-samples scheme, we generate one set of $\bar{n}(t-1)$ iid standard normal observations where $\bar{n}(t-1) = \max\{n(t-1,1), \ldots, n(t-1,m_{t-1})\}$. Then, for each cell $i = 1, \ldots, m_{t-1}$, we use the first $n(t-1,i)$ of $\bar{n}(t-1)$ standard normal observations to compute $S_t^j, j = 1, \ldots, n(t-1,i)$, $t = 2, \ldots, T$. We investigate the effect of these two sampling schemes on the bias and variance of the optimistic-bound estimator in Section 4.4.1.

Although we describe our tree building procedure in terms of sample state-based trees, this procedure can be directly applied to both ordinary and state-based sample trees. Therefore, in what follows we use the state variable $\tilde{S}_t$ to refer to the node and cell interchangeably. In particular, the node at which $\tilde{S}_t = S_t$ on the ordinary sample tree corresponds to the cell in the state-based sample tree such that $\tilde{S}_t \in [G_t, G_t + \Delta S]$.

4.2.3 Bias Characterization

Consider a node $\tilde{S}_t = S_t$, and replace $g_t(S_t)$ in (4.2) and (4.3) with a sample mean estimator. Then, we can form

$$f'_t(S_t) = \max\Big\{h_t(S_t), \frac{1}{n(t)} \sum_{i \in D_t} e^{-r_{t+1}} f_{t+1}(\tilde{S}^i_{t+1})\Big\}, \tag{4.6}$$

where D_t is the index set of the descendants, $\tilde{S}^i_{t+1}, i = 1, \ldots, n(t)$, of S_t (see the description of scenario tree notation in Section 2.3.2). These descendants are iid

observations of $\tilde{S}_{t+1}$ drawn from $F_{t+1}(S_{t+1}|S_t)$. Of course, it is not possible to evaluate $f_{t+1}(\tilde{S}_{t+1})$ exactly for $\tilde{S}_{t+1} = S_{t+1}$; instead, it must be estimated. Doing so, we are lead to recursively define, for $t = 0, \ldots, T-1$,

$$\hat{f}_t(S_t) = \max\Big\{h_t(S_t), \frac{1}{n(t)} \sum_{i \in D_t} e^{-r_{t+1}} \hat{f}_{t+1}(\tilde{S}^i_{t+1})\Big\}, \tag{4.7}$$

and $\hat{f}_T(S^i_T) = f_T(S^i_T), \forall i$. This allows us to recursively compute $\hat{f}_0(S_0)$ which is the optimistic-bound estimator of $f_0(S_0)$. Broadie and Glasserman [18] show that $\hat{f}_0(S_0)$ is biased high and is consistent under the assumption that, for some $q > 1$, $E|h_t(S_t)|^q < \infty$ for all t, and all nodes in the sample tree have the same number of branches, i.e., these results do not require that $\{\tilde{S}_t\}_{t=1}^T$ is governed by (4.5).

The bias at node $\tilde{S}_t = S_t$ is given by

$$\beta_t(S_t) = E\hat{f}_t(S_t) - f_t(S_t), \tag{4.8}$$

and depends on the sample subtree on which $\hat{f}_t(S_t)$ is defined. The subtree dependency of β_t is identical to that of the optimistic-bound estimator in the case of a multi-stage stochastic program with recourse described in Chapter 3. Ideally, we want to obtain an analytical expression for $\beta_t(S_t)$, but it may not be possible due to the nested expectation and optimization involved. Therefore, instead of dealing directly with $\beta_t(S_t)$, we approximate $\beta_t(S_t)$ with

$$b_t(S_t, n(t)) = Ef'_t(S_t) - f_t(S_t) \geq 0. \tag{4.9}$$

This approximation ignores bias introduced in future periods by replacing $\hat{f}_{t+1}(\tilde{S}^i_{t+1})$ in (4.7) with $f_{t+1}(\tilde{S}^i_{t+1})$, but as we will see $b_t(S_t, n(t))$ still yields a useful approximation of $\beta_t(S_t)$. Nonnegativity of b_t follows directly from Lemma 3 in Chapter 3. The expectations in (4.8) and (4.9) are conditional expectations on $\tilde{S}_t = S_t$, and we do not explicitly express their conditional parts for notational simplicity.

To evaluate $b_t(S_t, n(t))$, we must solve an optimization problem that defines $f_t(S_t)$ in (4.6), which requires the knowledge of the correct decision at the node $\tilde{S}_t = S_t$. Below, we show in Proposition 5 that $b_t(S_t, n(t))$ can be expressed analytically without an explicit solution of that optimization problem, i.e., we do not need to know the correct decision at the node. This turns out to be a key step in formulating an optimization problem for sample allocation described in Section 4.2.4.

Proposition 5. *Let f_t, g_t and h_t be defined in (4.2)-(4.4). Define $f'_t(S_t)$ as in (4.6), where $\tilde{S}^i_{t+1}, i = 1, \ldots, n(t)$, are iid observations of $\tilde{S}_{t+1}$ generated from $F_{t+1}(S_{t+1}|S_t)$. Let*

$$W_t = \frac{\bar{X}_t - \mu_t(S_t)}{\sigma_t(S_t)/\sqrt{n(t)}},$$

where

$$\bar{X}_t = \frac{1}{n(t)} \sum_{i \in D_t} e^{-r_{t+1}} f_{t+1}(\tilde{S}^i_{t+1}),$$

$\mu_t(S_t) = E[e^{-r_{t+1}} f_{t+1}(\tilde{S}^1_{t+1})|\tilde{S}_t = S_t]$, *and* $\sigma^2_t(S_t) = \mathrm{var}\,[e^{-r_{t+1}} f_{t+1}(\tilde{S}^1_{t+1})|\tilde{S}_t = S_t]$. *If the distribution function of W_t is symmetric, then bias approximation $b_t(S_t, n(t))$, given in (4.9), can be expressed as*

$$b_t(S_t, n(t)) = \frac{\sigma_t(S_t)}{\sqrt{n(t)}} E \max \left\{ -\frac{|h_t(S_t) - \mu_t(S_t)|}{\sigma_t(S_t)/\sqrt{n(t)}}, W_t \right\}. \tag{4.10}$$

In addition, $b_t(S_t, \cdot)$ is convex on $\mathbb{R}_+$. If we further assume that W_t is a standard normal with cumulative distribution function Φ, then

$$\begin{aligned} b_t(S_t, n(t)) &= -\big|h_t(S_t) - \mu_t(S_t)\big| \, \Phi\Big(- \frac{|h_t(S_t) - \mu_t(S_t)|}{\sigma_t(S_t)} \sqrt{n(t)} \Big) \\ &\quad + \frac{\sigma_t(S_t)}{\sqrt{2\pi n(t)}} \exp\Big(- \frac{(h_t(S_t) - \mu_t(S_t))^2}{2\sigma^2_t(S_t)} n(t) \Big). \end{aligned} \tag{4.11}$$

Proof. To simplify the notation, we suppress the dependence on S_t in the proof

(except for $b_t(S_t, n(t))$). Using the definitions given in the hypotheses, we can write

$$\begin{aligned} Ef_t' &= E\max\left\{h_t, \bar{X}_t\right\} \\ &= \mu_t + E\max\left\{h_t - \mu_t, \bar{X}_t - \mu_t\right\} \\ &= \mu_t + \frac{\sigma_t}{\sqrt{n(t)}} E\max\left\{\frac{h_t - \mu_t}{\sigma_t/\sqrt{n(t)}}, W_t\right\}. \end{aligned} \tag{4.12}$$

The "max" in (4.12) represents the decision to hold or exercise and so we consider these two cases. First, if $h_t \geq \mu_t$, then the correct decision is to exercise. Thus, $f_t = h_t$, and

$$\begin{aligned} b_t(S_t, n(t)) &= Ef_t' - h_t \\ &= \mu_t - h_t + \frac{\sigma_t}{\sqrt{n(t)}} E\max\left\{\frac{h_t - \mu_t}{\sigma_t/\sqrt{n(t)}}, W_t\right\}. \end{aligned} \tag{4.13}$$

The second case is that $h_t < \mu_t$. In this case, $f_t = \mu_t$, and

$$\begin{aligned} b_t(S_t, n(t)) &= Ef_t' - \mu_t \\ &= \frac{\sigma_t}{\sqrt{n(t)}} E\max\left\{\frac{h_t - \mu_t}{\sigma_t/\sqrt{n(t)}}, W_t\right\} \\ &= \frac{\sigma_t}{\sqrt{n(t)}} E\max\left\{-\frac{|h_t - \mu_t|}{\sigma_t/\sqrt{n(t)}}, W_t\right\}. \end{aligned} \tag{4.14}$$

Consider the case when $h_t \geq \mu_t$, and let $a = \sqrt{n(t)}(h_t - \mu_t)/\sigma_t \geq 0$. From

(4.13),

$$
\begin{aligned}
b_t(S_t, n(t)) &= -|\mu_t - h_t| + \frac{\sigma_t}{\sqrt{b_t}} E \max\{a, W_t\} \\
&= \int_{-\infty}^{\infty} -|\mu_t - h_t| \phi(u) du + \int_{-\infty}^{a} (h_t - \mu_t) \phi(u) du \\
&\quad + \frac{\sigma_t}{\sqrt{n(t)}} \int_{a}^{\infty} u \phi(u) du \\
&= \int_{a}^{\infty} -|h_t - \mu_t| \phi(u) du + \frac{\sigma_t}{\sqrt{n(t)}} \int_{a}^{\infty} u \phi(u) du \\
&= \int_{-\infty}^{-a} -|h_t - \mu_t| \phi(u) du + \frac{\sigma_t}{\sqrt{n(t)}} \int_{-a}^{\infty} u \phi(u) du \quad \text{(by symmetry)} \\
&= \frac{\sigma_t}{\sqrt{n(t)}} E \max\left\{ -\frac{|h_t - \mu_t|}{\sigma_t / \sqrt{n(t)}}, W_t \right\}. \qquad (4.15)
\end{aligned}
$$

To show convexity of $b_t(S_t, \cdot\,)$, rewrite (4.15) as

$$
b_t(S_t, n(t)) = E \max\Big\{ -\big|h_t - \mu_t\big|, \bar{X}_t - \mu_t \Big\}.
$$

Observe that $\bar{X}_t$ is convex in $n(t)$ and is the only element that depends on $n(t)$. Since the "max" function of a finite collection of convex functions is convex, and the expectation of a convex function is also convex, we have that $b_t(S_t, \cdot\,)$ is convex on $\mathbb{R}_+$.

If W_t is a standard normal random variable, we obtain (4.11) by carrying out the integration in the last line of (4.15) using the standard normal density function. □

The symmetry of the distribution function of W_t is asymptotically justified since when $n(t)$ grows large, by the central limit theorem we know that the

distribution function of W_t converges to that of the standard normal. Note that $b_t(S_t, n(t))$ is maximized when $h_t = \mu_t$; namely, the bias is high when the decision either to hold or to exercise is not obvious. Under the normality assumption of W_t, $b_t(S_t, n(t))$ is an increasing function of σ_t since its first derivative with respect to σ_t is non-negative.

In general, the true value of $\mu_t(S_t)$ and $\sigma_t(S_t)$ in (4.10) and (4.11) are not known and must be estimated. We denote $\hat{b}_t(S_t, n(t))$ the estimated bias approximation resulting from using estimates $\hat{\mu}_t(S_t)$ and $\hat{\sigma}_t(S_t)$ in (4.10) and (4.11). The impact of replacing μ_t and σ_t with estimators on the effectiveness of the tree building procedure (described in Section 4.2.4) is numerically tested in Section 4.4.

4.2.4 Sample Tree Construction

Using the bias approximation in (4.11) for a given node, we can form an optimization problem to determine the number of branches emanating from each grid point in the DP solution method so that the total expected bias is minimized. We begin by considering allocation within a single stage t by assuming that a total number of n_{t+1} stage $t+1$ observations of $\tilde{S}_{t+1}$ may be drawn, and that each grid point i must have at least $\underline{n}(t,i)$ stage $t+1$ samples. Let $n(t,i)$ be the number of stage $t+1$ samples allocated to cell $i = 1, \ldots, m_t$, and $\hat{p}_t^i$ be an estimate of p_t^i, the probability that $\tilde{S}_t$ is in cell i at stage t. We formulate the stage t optimization problem as

$$\begin{aligned} \min_{n(t,1),\ldots,n(t,m_t)} \quad & \sum_{i=1}^{m_t} \hat{p}_t^i \, \hat{b}_t(S_t^i, n(t,i)) \\ \text{s.t.} \quad & \sum_{i=1}^{m_t} n(t,i) = n_{t+1} \\ & n(t,i) \geq \underline{n}(t,i)\,, i = 1, \ldots, m_t. \end{aligned} \tag{4.16}$$

Since $b_t(S_t, \cdot)$ is convex on $\mathbb{R}_+$, (4.16) is a convex program. Although we can compute p_t^i exactly from the marginal distribution of the asset price process given in (4.5), we use its empirical estimate, $\hat{p}_t^k$, from the sample tree being constructed to form the objective function of (4.16). Our motivation is that only the cells in which observations of stage t asset price fall, should be included in the objective function of (4.16). Note that for ordinary sample trees, p_t^i is equal to the probability that $\tilde{S}_t = S_t^i, i = 1, \ldots, m_t = n_t$, the total number of stage t nodes.

Solving (4.16) provides an allocation decision for a specific stage t. To build a T-stage tree, we sequentially solve (4.16) from t_1 to t_{d-1} since this scheme is not useful for choosing samples out of stage t_0 and not needed for stage t_d. An algorithmic statement of the non-uniform state-based sample tree building procedure is summarized in Figure 4.1.

Step 1	For a given c and ΔS, construct a DP grid for stage $t_0, \ldots, t_d$.
Step 2	At stage t_0, draw $n(0) = n_1$ iid samples of $\tilde{S}_1$ from $F_1(S_1\|S_0)$.
	Compute estimates $\hat{p}_1^i,\ i = 1, \ldots, m_1$.
Step 3	Do $t = t_1, \ldots, t_{d-1}$
	For a given n_{t+1}, solve (4.16) for $n^*(t,i)$, $i = 1, \ldots, m_t$.
	Draw $n^*(t,i)$ (conditionally) iid samples of $\tilde{S}_{t+1}$ from
	$F_{t+1}(S_{t+1}\|S_t^i = G_t^i + \Delta S)$ for each cell $i = 1, \ldots, m_t$.
	Compute estimates $\hat{p}_{t+1}^j,\ j = 1, \ldots, m_{t+1}$.

Figure 4.1: A procedure to generate a state-based non-uniform sample tree for the American-style option pricing problem in order to reduce the bias associated with the optimistic-bound estimator.

To quantify error associated with the optimistic-bound estimator, we independently replicate the procedure in Figure 4.1 ν times and form the estimator of $f_0(S_0)$ with the standard sample mean, i.e.,

$$\bar{f}_0(S_0) = \frac{1}{\nu} \sum_{i=1}^{\nu} \hat{f}_0^i(S_0),$$

where $\hat{f}_0^i(S_0)$ is the option value computed from sample tree i. This allows confidence interval construction as described in Section 3.5.

Instead of specifying computational budget for each stage, one can specify the total budget for building a sample tree, and then try to compute the optimal allocation of the budget both between stages and between cells in the same stage. There are two issues. First, the value of $\hat{p}_t^i$ is not known until the sampling of $\tilde{S}_t$ is done. Second, we do not know how the bias from stage $t+1$ propagates back to stage t. To circumvent these two problems, we use the true probability weight p_t^i computed from the marginal of $\tilde{S}_t$ in (4.16), and assume that the bias propagation is additive. As our numerical results indicate, this assumption is (numerically) justified.

Specifically, for a given total budget N, we solve the following optimization problem to allocate the total budget among stages.

$$\begin{aligned} \min_{n(t,i)} \quad & \sum_{t=0}^{T-1} \sum_{i=1}^{m_t} p_t^i \, \hat{b}_t(S_t^i, n(t,i)) \\ \text{s.t.} \quad & \sum_{t=0}^{T-1} \sum_{i=1}^{m_t} n(t,i) = N \\ & n(t,i) \geq \underline{n}(t,i), i = 1, \ldots, m_t, t = 1, \ldots, T-1. \end{aligned} \tag{4.17}$$

Instead of using the optimal solution of (4.17) directly to build a sample tree, we compute the budget for stage $t = 1, \ldots, T$, with

$$n_t^* = \sum_{i=1}^{m_t} n^*(t-1, i),$$

where $n^*(t,i)$ is an optimal solutions of (4.17). Then, we apply the procedure in Figure 4.1 to build a sample tree with the stage-wise budget $(n_1^*, \ldots, n_T^*)$. This has the benefit of adapting the sample size, $n(t,i)$, to different sample trees when replicating the procedure in Figure 4.1, instead of using a fixed $n^*(t,i)$ directly for every replication. We relax the integer requirement when we solve (4.16) and (4.17),

and round off their solutions such that their computational budget constraints are satisfied. We present computational results of these procedures in Section 4.4.

4.3 Reducing Bias in Stochastic Lot Sizing

In this section, we extend the results of Section 4.2.3 to a stochastic lot-sizing problem. Deterministic lot sizing is a well-studied problem, and the stochastic variant we will describe is a well-solved problem via DP. Again, our intention is to gain insights to bias characterization and the sample tree construction to be used for more complex multi-stage stochastic optimization problems.

Since there is a large volume of literature on lot-sizing problems, we review only those involving with the stochastic programming literature. In the context of multi-stage stochastic mixed-integer programming, Haugen, Løkketangen and Woodruff [64] develop a heuristic that combines the progressive hedging algorithm with the tabu search to obtain a near-optimal solution. Lulli and Sen [89] propose a branch-and-price decomposition method for a stochastic lot-sizing model. Ahmed, King, and Parija [1] develop a dynamic capacity expansion model that contains a stochastic lot-sizing substructure, and propose a lower and upper bounding heuristic to be used with a branch-and-bound search to obtain a global optimal solution.

We adapt the deterministic formulation of Pochet and Wolsey [105] to the stochastic setting. At stage t, let x_t denote the binary decision of whether to produce, y_t the number of items to be produced, s_t the number of items in the inventory at the end of stage t, and r_t the level of backlogged demand at the end of stage t.

We can express the stochastic lot-sizing with the following recursion:

$$\begin{aligned} z^* = \min_{x_1,y_1,r_1,s_1} \quad & a_1x_1 + c_1y_1 + g_1r_1 + q_1s_1 + Eh_1(s_1 - r_1, \tilde{d}_2) \\ \text{s.t.} \quad & y_1 + r_1 - s_1 = \tilde{d}_1 + r_0 - s_0 \\ & y_1 \le Kx_1 \\ & x_1 \in \{0,1\} \\ & y_1, r_1, s_1 \in \mathbb{Z}_+, \end{aligned} \tag{4.18}$$

where, for $t = 2, \ldots, T$,

$$\begin{aligned} h_{t-1}(s_{t-1} - r_{t-1}, \tilde{d}_t) \; = \; \min_{x_t,y_t,r_t,s_t} \quad & a_tx_t + c_ty_t + g_tr_t + q_ts_t + Eh_t(s_t - r_t, \tilde{d}_{t+1}) \\ \text{s.t.} \quad & y_t + r_t - s_t = \tilde{d}_t + r_{t-1} - s_{t-1} \\ & y_t \le Kx_t \\ & x_t \in \{0,1\} \\ & y_t, r_t, s_t \in \mathbb{Z}_+, \end{aligned} \tag{4.19}$$

and $h_T \equiv 0$.

Data parameters a_t, c_t, g_t, and q_t denote the deterministic set-up, per unit production, backlogging and holding costs, respectively. We assume that the demand process, $\{\tilde{d}_t\}_{t=1}^T$, is interstage independent; thus, we do not use conditional expectation in (4.18) and (4.19). An initial demand $\tilde{d}_1$ is a degenerate random variable taking value d_1 with probability one. We denote the probability density function of the random demand by $F_t(d_t)$, and assume that the support of each $\tilde{d}_t$ is a finite set of nonnegative integers. If the capacity K is large enough, e.g., larger than the sum of the largest demand realization in each stage, then the problem is said to be uncapacitated. This is the case that we use in our computational experiments. The stochastic future cost at stage t is $h_t(s_t - r_t, \tilde{d}_{t+1})$. We use an inventory

position at the beginning of stage t as the state variable for DP. The inventory position at stage t is denoted I_t, and is defined as

$$I_t = s_{t-1} - r_{t-1} - d_t, \tag{4.20}$$

for $\tilde{d}_t = d_t$. The inventory position forms a random process, $\{\tilde{I}_t\}_{t=1}^T$, and I_1 is deterministic.

Only one binary decision is made at each stage t, i.e., to hold or to exercise the option, in the American-style option pricing problem, while two decisions are made at each stage t in the stochastic lot-sizing problem, i.e., we need to decide whether to produce, and if so, how many items to produce. The stage t objective function for a given inventory position $\tilde{I}_t = I_t$ is denoted

$$f_t(I_t, x_t, y_t) = a_t x_t + c_t y_t + g_t r_t + q_t s_t + E h_t(s_t - r_t, \tilde{d}_{t+1}).$$

From (4.19) and (4.20), $s_t - r_t = y_t + I_t$. If $y_t + I_t \geq 0$, then $s_t = y_t + I_t$ and $r_t = 0$. If $y_t + I_t < 0$, then $r_t = -(y_t + I_t)$ and $s_t = 0$. So, values of s_t and r_t are determined by the value of I_t, x_t, and y_t; hence, f_t needs not be expressed as a function of s_t and r_t. In the derivation of the bias approximation formulae later in this section, it is also convenient to use only I_t, x_t, and y_t as arguments to f_t.

As in the American-style option pricing problem, we form an optimistic-bound estimator by solving a DP on a state-based sample tree. The procedure to construct the state-based sample tree described in Section 4.2.2 needs to be modified slightly for the lot-sizing problem, and we return to this issue later in this section. The stage t objective function based on the state-based sample tree, for $\tilde{I}_t = I_t$, is

$$\hat{f}_t(I_t, x_t, y_t) = a_t x_t + c_t y_t + g_t r_t + q_t s_t + \frac{1}{n(t)} \sum_{i \in D_t} \hat{h}_t(s_t - r_t, \tilde{d}_{t+1}^i), \tag{4.21}$$

where $\tilde{d}_{t+1}^i, i = 1, \ldots, n(t)$, are iid observations of $\tilde{d}_{t+1}$ drawn from $F_{t+1}(d_{t+1})$, $n(t)$ is the number of branches emanating from a generic node I_t, and the approximating

future cost function at stage t is recursively defined as

$$\begin{aligned}
&\hat{h}_{\tau-1}(s_{\tau-1} - r_{\tau-1}, \tilde{d}_\tau) \;\; = \\
&\min_{x_\tau, y_\tau, r_\tau, s_\tau} \;\; a_t x_\tau + c_\tau y_\tau + g_\tau r_\tau + q_\tau s_\tau + \frac{1}{n(\tau)} \sum_{i \in D_\tau} \hat{h}_\tau(s_\tau - r_\tau, \tilde{d}^i_{\tau+1}) \\
&\qquad \text{s.t.} \;\; y_\tau + r_\tau - s_\tau = \tilde{d}_\tau + r_{\tau-1} - s_{\tau-1} \\
&\qquad\qquad y_\tau \le K x_\tau \\
&\qquad\qquad x_\tau \in \{0, 1\} \\
&\qquad\qquad y_\tau, r_\tau, s_\tau \in \mathbb{Z}_+,
\end{aligned} \tag{4.22}$$

for $\tau = t+1, \ldots, T$, and $\hat{h}_T \equiv 0$. To obtain (4.22), the expectation in (4.19) is replaced with the standard sample mean estimator associated with the sample tree. We define the bias of a stage t optimistic-bound estimator for the stochastic lot-sizing problem by

$$\beta_t(I_t) = \min_{x_t, y_t} f_t(I_t, x_t, y_t) - E \min_{x_t, y_t} \hat{f}_t(I_t, x_t, y_t) \ge 0. \tag{4.23}$$

Again, we approximate $\beta_t(I_t)$ with

$$b_t(I_t, n(t)) = \min_{x_t, y_t} f_t(I_t, x_t, y_t) - E \min_{x_t, y_t} f'_t(I_t, x_t, y_t) \ge 0, \tag{4.24}$$

where

$$f'(I_t, x_t, y_t) = a_t x_t + c_t y_t + g_t r_t + q_t s_t + \frac{1}{n(t)} \sum_{i \in D_t} h_t(s_t - r_t, \tilde{d}^i_{t+1}), \tag{4.25}$$

and $\tilde{d}^i_{t+1}, i = 1, \ldots, n(t)$, are iid observations of $\tilde{d}_{t+1}$. In the definition of f'_t, we ignore the bias in stages $t+1, \ldots, T-1$, by replacing $\hat{h}_t(s_t - r_t, \tilde{d}^i_{t+1})$ in (4.21) with $h_t(s_t - r_r, \tilde{d}^i_{t+1})$. Nonnegativity of β_t and b_t follows from Theorem 4.

Denote the optimistic-bound estimator by

$$\hat{z}^* = \min_{x_1, y_1} \hat{f}_1(I_1, x_1, y_1).$$

To obtain an analytical expression for b_t, which will aid in allocating computational resource in sample tree construction in order to reduce the bias associated with $\hat{z}^*$, we re-write (4.24) as

$$\begin{aligned} b_t(I_t, n(t)) &= \min_{x_t, y_t} f_t(I_t, x_t, y_t) - E \min_{x_t, y_t} f'_t(I_t, x_t, y_t) \\ &= \min\{f_t(I_t, 0, 0), f_t(I_t, 1, 1), \ldots, f_t(I_t, 1, K)\} - \\ &\quad E \min\left\{f'_t(I_t, 0, 0,), f'_t(I_t, 1, 1), \ldots, f'_t(I_t, 1, K,)\right\}. \end{aligned} \tag{4.26}$$

Let y_t^* and $\hat{y}_t^*$ satisfy

$$f_t(I_t, 1, y_t^*) \leq f_t(I_t, 1, y_t), \quad y_t = 1, \ldots, K, \tag{4.27}$$

and

$$f'_t(I_t, 1, \hat{y}_t^*) \leq f'_t(I_t, 1, y_t), \quad y_t = 1, \ldots, K. \tag{4.28}$$

At node $\tilde{I}_t = I_t$, if $f_t(I_t, 0, 0) < f_t(I_t, 1, y_t^*)$, then the correct decision is not to produce, and (4.26) reduces to

$$\begin{aligned} b_t(I_t, n(t)) &= f_t(I_t, 0, 0) - E \min\left\{f'_t(I_t, 0, 0), f'_t(I_t, 1, \hat{y}_t^*)\right\} \\ &= f_t(I_t, 0, 0) - E f'_t(I_t, 0, 0) - E \min\left\{0, f'_t(I_t, 1, \hat{y}_t^*) - f'_t(I_t, 0, 0)\right\} \\ &= -E \min\left\{0, f'_t(I_t, 1, \hat{y}_t^*) - f'_t(I_t, 0, 0)\right\}. \end{aligned} \tag{4.29}$$

The last equality in (4.29) follows from the definition of f'_t in (4.25), which implies $E f'_t(I_t, 0, 0) = f_t(I_t, 0, 0)$. On the other hand, if $f_t(I_t, 0, 0) \geq f_t(I_t, 1, y_t^*)$, then the correct decision is to produce y_t^*, and (4.26) reduces to

$$\begin{aligned} b_t(I_t, n(t)) &= f_t(I_t, 1, y_t^*) - E \min\left\{f'_t(I_t, 0, 0), f'_t(I_t, 1, \hat{y}_t^*)\right\} \\ &= f_t(I_t, 1, y_t^*) - E f'_t(I_t, 1, \hat{y}_t^*) \\ &\quad -E \min\left\{f'_t(I_t, 0, 0) - f'_t(I_t, 1, \hat{y}_t^*), 0\right\}. \end{aligned} \tag{4.30}$$

We show in Proposition 6 that the bias approximation in (4.29) and (4.30) can be expressed as a function of $n(t)$ that differs by a constant (independent of $n(t)$) under assumptions that the decision $\hat{y}_t^* = y_t^*$, wp1, and the distribution function of a random variable associated with $f_t'(I_t, 1, \hat{y}_t^*) - f_t'(I_t, 0, 0)$ is symmetric. By assuming $\hat{y}_t^* = y_t^*$, wp1, we ignore the bias associated with the decision variable y_t since $Ef_t'(I_t, 1, y_t^*) = f_t(I_t, 1, y_t^*)$.

Proposition 6. *Let y_t^* and $\hat{y}_t^*$ be defined by (4.27) and (4.28), and $\hat{y}_t^* = y_t^*$, wp1. Define $f'(I_t, x_t, y_t)$ as in (4.25) where $\tilde{d}_{t+1}^i, i = 1, \ldots, n(t)$, are iid observations of $\tilde{d}_{t+1}$. Let*

$$W_t = \frac{\bar{X}_t - \mu_t(I_t)}{\sigma_t(I_t)/\sqrt{n(t)}},$$

where

$$\bar{X}_t = f_t'(I_t, 1, y_t^*) - f_t'(I_t, 0, 0),$$

$\mu_t(I_t) = E\bar{X}_t$, and $\sigma_t(I_t) = \operatorname{var} \bar{X}_t$. If the distribution function of W_t is symmetric, then

$$b_t(I_t, n(t)) = \begin{cases} -\mu_t(I_t) + b_t'(I_t, n(t)) \text{ if } f_t(I_t, 0, 0) < f_t(I_t, 1, y_t^*) \\ b_t'(I_t, n(t)) \text{ if } f_t(I_t, 0, 0) \geq f_t(I_t, 1, y_t^*), \end{cases} \tag{4.31}$$

where

$$b_t'(I_t, n(t)) = -\frac{\sigma_t(I_t)}{\sqrt{n(t)}} E \min\Big\{ -\frac{\mu_t(I_t)}{\sigma_t(I_t)/\sqrt{n(t)}}, W_t \Big\}. \tag{4.32}$$

In addition, $b_t'(I_t, \cdot)$ is convex on $\mathbb{R}_+$. If we further assume that W_t is a standard normal with cumulative distribution function Φ, then

$$b_t'(I_t, n(t)) = -\mu_t(I_t)\, \Phi\Big(-\frac{\mu_t(I_t)}{\sigma_t(I_t)} \sqrt{n(t)} \Big) + \frac{\sigma_t(I_t)}{\sqrt{2\pi n(t)}} \exp\Big(-\frac{\mu_t^2(I_t)}{2\sigma_t^2(I_t)} n(t) \Big). \tag{4.33}$$

Proof. We suppress the dependence on state I_t (except for $b_t(I_t, n(t))$) for notational

simplicity. Consider the case when $f_t(I_t, 0, 0) < f_t(I_t, 1, y_t^*)$, and re-write (4.29) as

$$\begin{aligned} b_t(I_t, n(t)) &= -E \min\{0, \bar{X}_t\} \\ &= -\mu_t - \frac{\sigma_t}{\sqrt{n(t)}} E \min\Big\{ - \frac{\mu_t}{\sigma_t/\sqrt{n(t)}}, \frac{\bar{X}_t - \mu_t}{\sigma_t/\sqrt{n(t)}} \Big\} \\ &= -\mu_t - \frac{\sigma_t}{\sqrt{n(t)}} E \min\Big\{ - \frac{\mu_t}{\sigma_t/\sqrt{n(t)}}, W_t \Big\} \end{aligned}$$

Under the hypothesis that $\hat{y}_t^* = y_t^*$, wp1,

$$E f_t'(I_t, 1, y_t^*) = f_t(I_t, 1, y_t^*).$$

This follows from the definition of f_t and f', and that fact that $\tilde{d}_{t+1}^i, i = 1, \ldots, n(t)$, are iid observations of $\tilde{d}_{t+1}$. So, we can re-write (4.30) as

$$\begin{aligned} b_t(I_t, n(t)) &= -E \min\{ - \bar{X}_t, 0\} \\ &= \mu_t - \frac{\sigma_t}{\sqrt{n(t)}} E \min\Big\{ \frac{-\bar{X}_t + \mu_t}{\sigma_t/\sqrt{n(t)}}, \frac{\mu_t}{\sigma_t/\sqrt{n(t)}} \Big\} \\ &= \mu_t - \frac{\sigma_t}{\sqrt{n(t)}} E \min\Big\{ - W_t, \frac{\mu_t}{\sigma_t/\sqrt{n(t)}} \Big\}. \end{aligned} \tag{4.34}$$

Let $a = \mu_t \sqrt{n(t)}/\sigma_t$. We write (4.34) as

$$\begin{aligned} b_t(I_t, n(t)) &= \int_{-\infty}^{\infty} \mu_t \phi(u) du \; - \int_{-\infty}^{-a} \mu_t \phi(u) du \; + \frac{\sigma_t}{\sqrt{n(t)}} \int_{-a}^{\infty} u \phi(u) du \\ &= \int_{-a}^{\infty} \mu_t \phi(u) du \; - \frac{\sigma_t}{\sqrt{n(t)}} \int_{-\infty}^{-a} u \phi(u) du \quad \text{(by symmetry)} \\ &= -\frac{\sigma_t}{\sqrt{n(t)}} E \min\Big\{ - \frac{\mu_t}{\sigma_t/\sqrt{n(t)}}, W_t \Big\}. \end{aligned}$$

So, we have shown that $b_t(I_t, n(t))$ is given by (4.31).

To show convexity of $b_t'(I_t, \cdot)$, we re-write (4.32) as

$$b_t'(I_t, n(t)) = -E \min\{ - \mu_t, \bar{X}_t - \mu_t\}.$$

Observe that $\bar{X}_t$ is a convex function of $n(t)$, and is the only element in the above equation that depends on $n(t)$. Using the fact that the negative of a "min" function of a finite collection of convex functions is convex, and that the expectation of a convex function is also convex, we conclude that $b'_t(I_t, \cdot)$ is convex on $\mathbb{R}_+$.

If W_t is a standard normal, we can evaluate the expectation in (4.32) using the normal density function, and obtain (4.33). □

The values of $\mu_t(I_t)$ and $\sigma_t(I_t)$ must be estimated. We denote these estimates by $\hat{\mu}_t(I_t)$ and $\hat{\sigma}_t(I_t)$, and the resulting bias expression based on these estimates by $\hat{b}_t$ and $\hat{b}'_t$. As before, let n_{t+1} denote the total number of samples of $\tilde{d}_{t+1}$ (of all stage t nodes), and

$$p_t^i = P_t(\tilde{I}_t = I_t^i),\ i = 1, \ldots, m_t,$$

where P_t is the marginal probability distribution of $\tilde{I}_t$, and m_t is the number of DP grid points in stage t. The marginal P_t is induced by F_t, and we need to know an optimal solution of (4.18)-(4.19) at all the DP grid points in order to compute it. So, we use instead its estimate, which we describe below how to compute. Based on the estimates of p_t^i and $b_t(I_t, n(t))$, we can state the stage t optimization problem for sample size allocation as

$$\begin{aligned} \min_{n(t,1),\ldots,n(t,m_t)} \quad & \sum_{i=1}^{m_t} \hat{p}_t^i\, \hat{b}_t(I_t^i, n(t,i)) \\ \text{s.t.} \quad & \sum_{i=1}^{m_t} n(t,i) = n_{t+1} \\ & n(t,i) \geq \underline{n}(t,i),\ i = 1, \ldots, m_t. \end{aligned} \tag{4.35}$$

Since only the terms that depend on $n(t)$ are relevant in the sample allocation decision of (4.35), in our implementation we replace $\hat{b}_t(I_t^i, n(t,i))$ with $\hat{b}'_t(I_t^i, n(t,i))$. Since $\hat{b}_t(I_t^i, \cdot)$ is convex on $\mathbb{R}_+$, (4.35) is a convex program. Due to the finite integer

nature of the demand process and optimization models, the state variable I_t is a finite integer, and we do not need to discretize it or make other approximations when we construct the DP grid. As before, we can use (4.35) to allocate sample size in a sequential fashion. However, the fact that the state variable of the stochastic lot-sizing problem depends explicitly on decision variables in the previous stage prevent us from directly employing the procedure described in Figure 4.1 for the state-based sample tree construction of the American-style option pricing problem. In particular, we cannot compute the estimate of p_t^i since we need to know an optimal solution of (4.18)-(4.19) at all stage $t-1$ inventory positions in order to obtain the realization of $\tilde{I}_t$ through (4.20).

We therefore modify the procedure in Figure 4.1 by constructing an initial uniform state-based sample tree to obtain an estimate of an optimal solution at each cell. Then, the initial sample tree is refined by drawing additional samples for each cell based on the solution of (4.35) for a given additional computational budget. The statement of the modified procedure is shown in Figure 4.2 for a given number of tree refinement iterations, $i_{\max} \geq 1$. We denote n_{add} an additional computational budget for each cell.

The estimates of $\mu_t(I_t)$ and $\sigma_t(I_t)$ can be obtained in a similar manner as in the case of the American-style option pricing problem. Specifically, an independently-generated uniform sample tree is constructed and solved to obtain $\hat{\mu}_t(I_t)$ and $\hat{\sigma}_t(I_t)$ at each DP grid point. However, we alternatively obtain the estimates of $\mu_t(I_t)$ and $\sigma_t(I_t)$ as follows. Initial estimates are obtained after solving the lot-sizing problem based on the uniform state-based sample tree constructed in Step 1 of the procedure in Figure 4.2. The initial estimates, $\hat{\mu}_t(I_t)$ and $\hat{\sigma}_t(I_t)$, are then updated after the sample tree is refined in Step 5 and the approximating problem is re-solved in Step 6. We refer to this alternative procedure as being dynamic. The procedure

Step 0	Set $iter = 0$.
Step 1	Build an initial uniform state-based sample tree in which every node has n_{init} branches. Set the number of samples $n(t,i)$ to n_{init} for $i = 1, \ldots, m_t,\ t = 1, \ldots, T-1$.
Step 2	Solve an approximating problem based on the sample tree constructed in Step 1. Store the resulting optimal solution at each node as an incumbent solution.
Step 3	For the first stage, draw additional n_{add} iid samples of $\tilde{d}_2$ from $F_2(d_2)$. Compute estimates, $\hat{p}_2^i,\ i = 1, \ldots, m_2$, based on the incumbent solution stored at each node. Let η_2 be the number of nodes that have non-zero $\hat{p}_2^k$.
Step 4	Update the number of samples by $n(1,1) \leftarrow n(1,1) + n_{add}$.
Step 5	Do $t = 2, \ldots, T-1$ Let $n_{t+1} = \eta_t \cdot n_{\text{add}} + \sum_{i=1}^{m_t} n(t,i)$. Set the lower bound $\underline{n}(t,i) = n(t,i), i = 1, \ldots, m_t$. Solve (4.35) for $n^*(t,i), i = 1, \ldots, m_t$. If $n^*(t,i) - n(t,i) > 0$, then draw $n^*(t,i) - n(t,i)$ iid samples of $\tilde{d}_{t+1}$ from $F_{t+1}(d_{t+1})$ for $i = 1, \ldots, m_t$. Update the value of $n(t,i)$ to $n^*(t,i), i = 1, \ldots, m_t$. Compute $\hat{p}_{t+1}^j$, $j = 1, \ldots, m_{t+1}$, using the incumbent solution.
Step 6	Solve the approximating problem based on the refined sample tree. Store the resulting optimal solution as a new incumbent.
Step 7	$iter \leftarrow iter + 1$. If $iter < i_{\max}$, goto Step 3. Otherwise, stop.

Figure 4.2: A procedure to generate a state-based non-uniform sample tree for the stochastic lot-sizing problem in order to reduce the bias associated with the optimistic-bound estimator.

described in Figure 4.2 can be modified in a number of ways; for example, instead of using incumbent solutions to compute $\hat{p}_{t+1}^{j}$ in Step 5 (during the Do loop), the approximating problem can be re-solved and the resulting optimal solution can be used instead to compute $\hat{p}_{t+1}^{j}$. Also, n_{add} can be changed from iteration to iteration. These modifications are numerically tested in Section 4.4.2.

To quantify the error associated with $\hat{z}^*$, we generate iid replications of $\hat{z}^*$, and construct its sample mean estimator as

$$\bar{z}_{\nu}^{*} = \frac{1}{\nu} \sum_{i=1}^{\nu} \hat{z}^{*,i},$$

where $\hat{z}^{*,i}$ is computed from sample tree $i = 1, \ldots, \nu$. This allows confidence interval constructions as described in Section 3.5. We relax the integer requirement when we solve (4.35), and round off its solution such that the computational budget constraint is satisfied. We present computational results of the sample tree building procedure we propose for the stochastic lot-sizing problem in Section 4.4.2.

4.4 Computational Results

We present numerical results on the methods developed in the previous sections. Sampling-based DP solution procedures for pricing American-style options and lot-sizing problems are implemented in the C programming language. We use MINOS 5.4 as a subroutine to solve the sample size allocation problem (4.16), (4.17), and (4.35). All computations for the American-style option pricing problem in Sections 4.4.1 and for the stochastic lot-sizing problem in Section 4.4.2 are performed on an IBM PC (1.2 GHz) with 512 MB of memory running Windows ME.

4.4.1 American-style Option Pricing Problem

We consider a 4-stage American-style option pricing problem in which T is the length of one year, and $t_i, i = 0, 1, 2, 3$, correspond to each quarter of the year. The problem's parameters are taken from [18]: $S_0 = 110, K = 100, r = 0.05, \delta = 0.1$. We choose $S_0 = 110$ since it gives the highest percentage of the bias when fixing other parameters. The analytical value of this option is 11.341 and is obtained by using the analytical result of Geske and Johnson [60]. We set the truncation parameter c to be 5.0 for all stages (see Section 4.2.2). We first investigate the bias resulting from the discretization error, and the use of common- and independent-sample schemes (see Section 4.2.2). We use $\Delta S = (\Delta S_1, \Delta S_2, \Delta S_3)$ to denote the cell's width for t_1, t_2, and t_3, and is constant for all cells within a stage. Table 4.1 shows the value of the estimate of $f_0(S_0)$ based on a uniform tree in which we set $n(t, i)$ to a relatively large value of $20,000$, for $i = 1, \ldots, m_t$, $t = 0, \ldots, 2$. As the grid becomes finer, the error decreases, but the computational effort increases. Although the true value of the option is $f_0(S_0) = 11.341$, we use the estimate in Table 4.1 under the large uniform sample size of $20,000$ as our reference when we compute the bias because our goal is to focus on the bias due to sampling, not discretization of state variables. In particular, for the remainder of this section, we use the grid size of (0.7, 0.7, 0.7); thus, the reference value (from independent-samples column) is 11.729. "DP minutes" in Table 4.1 is the CPU time to compute $\bar{f}_0(S_0)$, which includes the time to generate and solve DPs.

Next, we construct uniform sample trees with $n(t, i) = 10, \ldots, 50, \forall i, t$, using the common- and independent-samples scheme in order to compare the bias associated with $\hat{f}(S_0)$. Using $\nu = 10,000$ sample trees, we report computational results for common- and independent-samples method in Table 4.2 and 4.3, respectively. The "95% HW" columns denote the half width of the 95% confidence interval on $\bar{f}(S_0)$.

ΔS	$\bar{f}(S_0)$		95 % HW		DP minutes	
	common	indep.	common	indep.	common	indep.
(1,2,2)	12.195	12.193	0.024	0.016	28.6	33.8
(.7,.7,.7)	11.737	11.729	0.023	0.019	186.3	184.7
(.5,.5,.5)	11.624	11.616	0.023	0.016	836.6	856.2
(.3,.3,.3)	11.512	11.503	0.023	0.017	3647.6	3653.8

Table 4.1: Optimistic-bound estimators for the value of an American call option on a single asset based on 50 uniform trees with $20,000$ branches. Each row corresponds to the estimate obtained for the specified grid size. "common" and "indep." refer to whether the common- or independent-samples method is used. The analytical value of the option price is 11.341.

The last column in Table 4.3 gives the percentage of bias reduction achieved by using the independent-samples method in comparison to the results of the common-samples method reported in Table 4.2.

branch	$\bar{f}(S_0)$	bias	95% HW	DP minutes
10	13.150	1.421	0.064	2.5
20	12.489	0.760	0.045	7.1
30	12.255	0.526	0.037	13.2
40	12.108	0.379	0.032	18.6
50	12.039	0.310	0.029	25.2

Table 4.2: Estimates of the value of an American call option on a single asset based on $\nu = 10,000$ uniforms tree generated by the common-samples method.

From Table 4.2 and Table 4.3, we observe that the independent-samples method outperforms the common-samples method. The value of $\bar{f}(S_0)$ computed from sample trees with independent samples has both lower variance and bias. We do not have a rigorous proof for bias reduction under the independent-samples method. Our intuitive explanation is that the independent-samples method generates sample trees with greater variety of scenarios; therefore, the optimization is harder since an optimal decision needs to hedge against more variety of scenarios. As shown

branch	$\bar{f}(S_0)$	bias	95% HW	DP minutes	%reduced
10	12.863	1.134	0.051	2.5	20.19
20	12.345	0.616	0.037	9.2	18.95
30	12.150	0.421	0.031	13.6	19.96
40	12.036	0.317	0.027	20.7	16.36
50	11.969	0.240	0.024	28.7	22.58

Table 4.3: Estimates of the value of an American call option on a single asset based on $\nu = 10,000$ uniforms tree generated by the independent-samples method.

in Table 4.3, the percentage of bias reduction tends to decrease as the number of branches grow, except the last row with the number of branches being 50.

To investigate the quality of our approximation, $b_t(S_t, n(t))$, of the true bias function, $\beta_t(S_t)$, from Proposition 5, we plot an estimate of $\beta_t(S_t)$ against an estimate of $b_t(S_t, n(t))$ for $t = t_1$ in Figures 4.3 and 4.4. Estimate $\hat{b}_1$ is computed from (4.11) with μ_1 and σ_1 estimated from a uniform sample tree with 20,000 branches. Estimate $\hat{\beta}_1$ is an average of the bias over 10,000 uniform sample trees. To compute the bias estimate, $\hat{\beta}_1$, we estimate the optimal value at each grid point using a uniform sample tree with 20,000 branches, and it is the same uniform sample tree that we use to estimate μ_1 and σ_1.

Figure 4.3 and Figure 4.4 suggest that the shape of the approximation function of bias, $b_t(S_t, n(t))$, matches very well with the true bias, $\beta_t(S_t)$. The difference between the two functions plotted in Figure 4.3 results from the fact that $b_t(S_t, n(t))$ ignores bias introduced in stages beyond t. When the number of branches increases to 30, both plots are almost identical. The plots for other stages are similar to those for stage $t = t_1$. The fact that the plot of the bias approximation and the true bias matches reasonably well indicates that the use of our approximation function in the sample size allocation optimization problem, (4.16) or (4.17), may be effective.

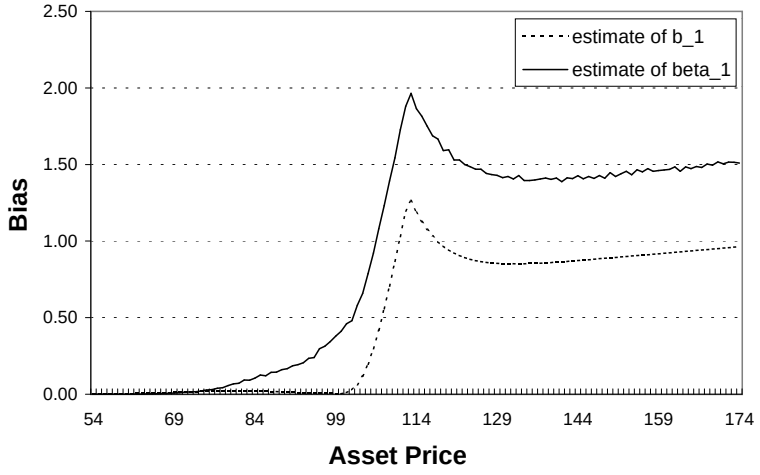

Figure 4.3: Comparison between an estimate of bias approximation, $\hat{b}_1$, and an estimate of bias, $\hat{\beta}_1$ for uniform sample tree with 10 branches.

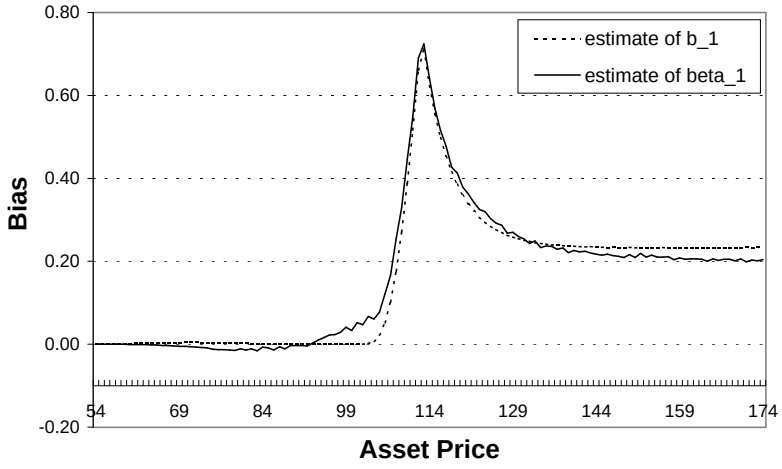

Figure 4.4: Comparison between an estimate of bias approximation, $\hat{b}_1$, and an estimate of bias, $\hat{\beta}_1$ for uniform sample tree with 30 branches.

Next, we present computational results for the procedure in Figure 4.1 from Section 4.2.3. Table 4.4 gives computational results when we estimate the value of μ_t and σ_t by using an independently-generated uniform sample trees with $20,000$ branches, while Table 4.5 gives results when using a uniform tree with 10 branches. The percentage of bias reduction in Tables 4.4 and 4.5 shows how the accuracy of the parameters affects the procedure. The percentage of bias reduction in the last two rows of Table 4.5 is negative because the bias estimates are larger than those in the corresponding rows of Table 4.3. CPU times are separated into the time to solve DPs, and the time to solve nonlinear convex programs for sample size allocation. These CPU times are denoted in Tables 4.4 and 4.5 with "DP" and "NLP", respectively. To obtain estimates of μ_t and σ_t for each cell in each stage t, we solve an approximating problem associated with an independently-generated uniform sample tree for the values of $\hat{\mu}_t$ and $\hat{\sigma}_t$ needed in the procedure of Figure 4.1. In general, this strategy may be impractical because we need to initially solve a relatively large uniform sample tree to obtain accurate estimates of μ_t and σ_t. We describe an alternative approach to dynamically obtain these estimates in Section 4.3.

In order to be able to compare the computational results in Table 4.4 and Table 4.5 with those in Table 4.3, we compute the average number of stage t cells, denote K_t, into which the observations of $\tilde{S}_t$ fall, when we conduct our experiments for Table 4.3. We then set stage t budget, n_t, in (4.16) to $K_t \times n_{\text{eff}}$ when constructing sample trees associated with the computation of each row in Table 4.4 and Table 4.5. So, the bias in each row of Table 4.4 and Table 4.5 is comparable to that in the corresponding row of Table 4.3, i.e., the row with n_{eff} being equal to "branch." Note that the n_{eff} columns in Table 4.4 and Table 4.5 do not give the actual number of branches for each cell.

Finally, Table 4.6 gives computational results of the procedure described in

n_{eff}	$\bar{f}(S_0)$	bias	95% HW	CPU minutes DP	CPU minutes NLP	%reduced
10	12.802	1.073	0.052	2.7	212.0	5.38
20	12.249	0.520	0.037	9.6	519.6	15.58
30	12.058	0.329	0.030	14.5	734.0	21.85
40	11.955	0.226	0.027	20.9	901.3	28.71
50	11.870	0.141	0.024	29.7	1043.2	41.25

Table 4.4: Estimates of the value of an American call option on a single asset based on $\nu = 10,000$ sample trees. Sample trees are constructed by the procedure in Figure 4.1 using the independent-samples method. The estimates of μ_t and σ_t are obtained separately from a independent-samples uniform trees with $20,000$ branches.

n_{eff}	$\bar{f}(S_0)$	bias	95% HW	CPU minutes DP	CPU minutes NLP	%reduced
10	12.779	1.050	0.052	2.5	206.4	7.41
20	12.313	0.584	0.037	9.7	480.5	5.19
30	12.147	0.418	0.031	15.9	681.7	0.71
40	12.047	0.318	0.027	20.7	852.2	-0.32
50	11.973	0.244	0.024	27.2	1014.3	-1.67

Table 4.5: Estimates of the value of an American call option on a Sample trees are constructed by the procedure in Figure 4.1 using the independent-samples method. The estimates of μ_t and σ_t are obtained separately from independent-samples uniform trees with 10 branches.

n_{eff}	$\bar{f}(S_0)$	bias	95% HW	CPU minutes DP	CPU minutes NLP	%reduced
10	12.637	0.908	0.043	2.9	283.7	19.93
20	12.107	0.378	0.037	7.9	607.8	38.64
30	11.966	0.237	0.022	12.7	841.9	43.71
40	11.877	0.148	0.019	19.2	1042.9	53.31
50	11.832	0.103	0.018	234.0	1213.1	57.08

Table 4.6: Estimates of the value of an American call option on a single asset based on $\nu = 10,000$ sample trees. Sample trees are constructed by the procedure in Figure 4.1 using the independent-samples method. The estimates of μ_t and σ_t are obtained separately from an independent-sample uniform trees with $20,000$ branches.

Figure 4.1 when the budget for each stage is obtained from solving (4.17). Similarly, we compute an average of the total number of branches (for all stages) when we conduct experiments for Table 4.3. Then, we set N in (4.17) to that value so that the bias in Table 4.6 is comparable to that in Table 4.3, Table 4.4, and Table 4.5. We observe larger percentage of bias reduction in Table 4.6 due to better sample size allocation between stages.

In Tables 4.4, 4.5, and 4.6, the time to solve DPs is approximately the same as that of Tables 4.3. However, the time to solve nonlinear programs for sample size allocation listed in Table 4.4, 4.5, and 4.6 is much larger than the time to solve DPs. We anticipate such difference because solution time for a nonlinear program generally dominates that of a DP (with one state variable). Our motivation is to develop a sample size allocation procedure for multi-stage stochastic programming. In such setting, we instead expect solution time of a multi-stage stochastic program will dominate that of a nonlinear program (with one constraint).

4.4.2 Stochastic Lot-sizing Problem

We generate a 4-stage instance of the stochastic lot-sizing model described by (4.18)-(4.19) using the data parameters in Table 4.7. Backlogging is not allowed in stage 4, i.e., $r_4 = 0$.

t	a_t	c_t	g_t	q_t
1	300	1.80	7.50	1.50
2	250	2.10	18.00	3.63
3	350	2.20	15.00	3.13
4	200	2.40	18.00	3.46

Table 4.7: Data parameters of the stochastic lot-sizing model used in the computational experiment.

We set $r_0 = s_0 = 0$, and $d_1 = 1$. The random demand for stage $t = 2, \ldots, T$, is assumed to be independent and distributed as a truncated Poisson random variable with mean equal to 12. We truncate any realization that has probability mass less than 10^{-4} (and adjust the probability mass so that it adds up to 1.0). The resulting support of the demand random variable is $\{2, \ldots, 26\}$. As indicated in Section 4.3, the DP grid for the stochastic lot-sizing problem is exact since the state variable I_t is integer. This problem can be solved exactly, and by doing so we obtain the optimal objective function value to be 548.174. Optimistic-bound estimators using uniform sample trees with different numbers of branches are given in Table 4.8.

Table 4.9 reports the values of optimistic-bound estimators based on the sample trees constructed by the procedure described in Figure 4.2. We compute the estimates of μ_t and σ_t needed in the procedure from an independently-generated uniform state-based sample tree with 25 branches at each grid point. In the procedure, we begin with an initial uniform sample tree that has $n_{\text{init}} = 10$, and use $n_{\text{add}} = 5$

branch	$\bar{z}^*_{1000}$	95% HW	bias	DP minutes
10	540.566	0.471	7.607	1.8
15	543.508	0.386	4.666	2.6
20	544.943	0.314	3.231	3.4
25	545.635	0.278	2.539	4.2

Table 4.8: Optimistic-bound estimators of the optimal objection function value of the stochastic lot-sizing test problem based on 1,000 uniform state-based sample trees (with independent samples).

to increase the tree size at each iteration. In order to compare the bias column in Table 4.8 and that in Table 4.9, we choose n_{init} and n_{add} in such a way that the total number of branches of a uniform sample tree used for Table 4.8 and that of a non-uniform sample tree used for Table 4.8 is approximately the same. So, the bias in each row of Table 4.8 is comparable to the bias in each row of Table 4.9 if the value of "branch" is equal to n_{eff}. The percentage of bias reduction is accordingly computed from the corresponding rows. We also experiment with other variations of the basic procedure (as explained in Section 4.3). When the estimate of μ_t and σ_t are obtained dynamically, the resulting bias is approximately the same as that of the uniform sample tree. Changing values of n_{add} from iteration to iteration or additionally re-solving the sample tree in Step 5 of Figure 4.2 does not really alter the bias reduction percentage given in Table 4.9.

We separate the time to solve DPs from the time to solve nonlinear programs to compute sample size in Table 4.9. The time to solve DPs in Table 4.9 is approximately the same as that in Table 4.8. The time to compute sample size allocation is, however, larger than the time to solve DPs (with one state variable) because it involves solving nonlinear programs. Again, we expect that if the sample size allocation procedure is to be used for multi-stage stochastic programming, the solution time of a multi-stage stochastic program will dominate that of a nonlinear

n_{eff}	$i_{\max}$	$\bar{z}^*_{1000}$	95% HW	bias	%reduced	CPU minutes	
						DP	NLP
15	1	544.902	0.343	3.272	29.89	2.1	3.9
20	2	546.223	0.313	1.951	39.62	3.8	6.4
25	3	546.414	0.280	1.761	30.66	4.0	9.6

Table 4.9: Optimistic-bound estimators of the optimal objective function value of the stochastic lot-sizing problem based on 1,000 non-uniform state-based sample trees (with independent-samples) constructed by the procedure of Figure 4.2 with $n_{\text{init}} = 10$ and $n_{\text{add}} = 5$.

program.

Chapter 5

Conclusions

We conclude the dissertation and provide future research directions in this chapter. Throughout the dissertation, we focus on multi-stage stochastic programs that have a large number of scenarios and a moderate-to-large number of stages. As discussed in Section 1.1, this class of models can be applied to many important applications, but is often computationally difficult to solve in spite of recent advances in optimization algorithms and computing technology. The first research objective of developing methods to solve such multi-stage stochastic programs is accomplished in Chapter 3. There, we develop two Monte Carlo sampling-based methods to generate a feasible policy for two classes of multi-stage stochastic programs. The first policy-generation method is applicable to multi-stage stochastic linear programs that exhibit interstage independence. To form a policy, it uses cuts generated by solving an approximating problem with the multi-stage L-shaped method. With a minor modification, certain types of dependency can be handled. The second policy-generation method is applicable to problems with general interstage dependency and does not require convexity of the underlying problem. However, we must be able to solve that class of problems with a modest number of scenarios.

In this dissertation, we adopt a view that a solution of a multi-stage stochastic program is a policy that generates a feasible decision at every stage under each scenario. In a closely-related area of stochastic control, this is a widely-accepted view for a solution of an optimization problem involving sequential decision making

under uncertainty. Alternatively, one may adopt a view that a solution of a multi-stage stochastic program is merely specified by a first stage decision with a rationale that a decision maker needs to know only what action to take now, and that future decisions are irrelevant at the current stage. Many approximation methods, such as those reviewed in Section 2.4.2, for multi-stage stochastic programs often take this view; as a result, they are designed to produce only a first stage decision. Nevertheless, we can still construct a policy solution from those approximation methods by applying them in a rolling forward fashion.

The second research objective of developing effective methods to determine the solution's quality is also accomplished in Chapter 3. In particular, to estimate the expected cost of an arbitrary policy, we propose two policy-cost estimators: scenario-based and tree-based estimators. Combining these policy-cost estimators with the optimistic-bound estimator developed in Section 3.5, we construct a confidence interval on the optimality of an arbitrary policy to establish the policy's quality. We propose two ways to combine these estimators to form a confidence interval: separate and gap estimators. The procedures to determine the policy's quality for multi-stage stochastic programs is a new contribution to the field. In our approach, the quality is measured by the objective function value; therefore, two different policies that yield the same expected cost are of the same quality. To determine the solution's quality when only a first stage decision is specified (and is not applied in a rolling forward fashion) is an open problem.

To accomplish the third research objective of demonstrating computational viability of our solution methods, we conduct computational experiments of the procedures we develop. Our preliminary computational results suggest that both policy-generation procedures are computationally viable, and that the procedure to determine the policy's quality via the gap estimator may be more effective due to

the variance reduction achieved by the common random numbers. In addition, the independent-samples method for sample tree construction produces an estimator with lower variance.

Practically speaking, the Monte Carlo sampling-based procedures we develop in Chapter 3 address three important issues that a decision make faces when using multi-stage stochastic programming for decision support. These three issues are (i) how to obtain a policy to operate a system, (ii) what the expected cost is under a given policy, and (iii) how well a given policy performs. We summarize our results using flowcharts as shown in Figures 5.1, 5.2, and 5.3 along the line of these issues. For issue (i), the flowchart in Figure 5.1 suggests when it is appropriate to apply procedure $\mathcal{P}_1$ or $\mathcal{P}_2$ (Section 3.3) to construct a policy for a given multi-stage stochastic program. For issue (ii), although we describe two estimators in Section 3.4 to estimate the expected cost of using a policy, as shown in the flowchart in Figure 5.2, we recommend using the scenario-based estimator if the policy's quality is not of interest. For issue (iii), we describe two estimators in Section 3.5 to estimate the optimality gap of a given policy, but recommend using the gap estimator to determine the policy's quality. In the flowchart in Figure 5.3, we only take a policy generated by either $\mathcal{P}_1$ or $\mathcal{P}_2$ as an input. It is, however, straightforward to modify the procedure for an arbitrary policy. Either sample trees with common or independent samples can be used in $\mathcal{P}_1$, and an instance of each type is illustrated in Figure 5.4. As an example, to determine the quality of a policy constructed under $\mathcal{P}_1$ as shown on the second line of Table 3.4 in Section 3.7, we generate one uniform sample tree with 7 descendants using the common-samples method in order to construct a policy, and then generate 30 uniform sample trees with 4 descendants using the independent-samples method in order to form the gap estimator.

The methodology we propose may be important in solving multi-stage stochas-

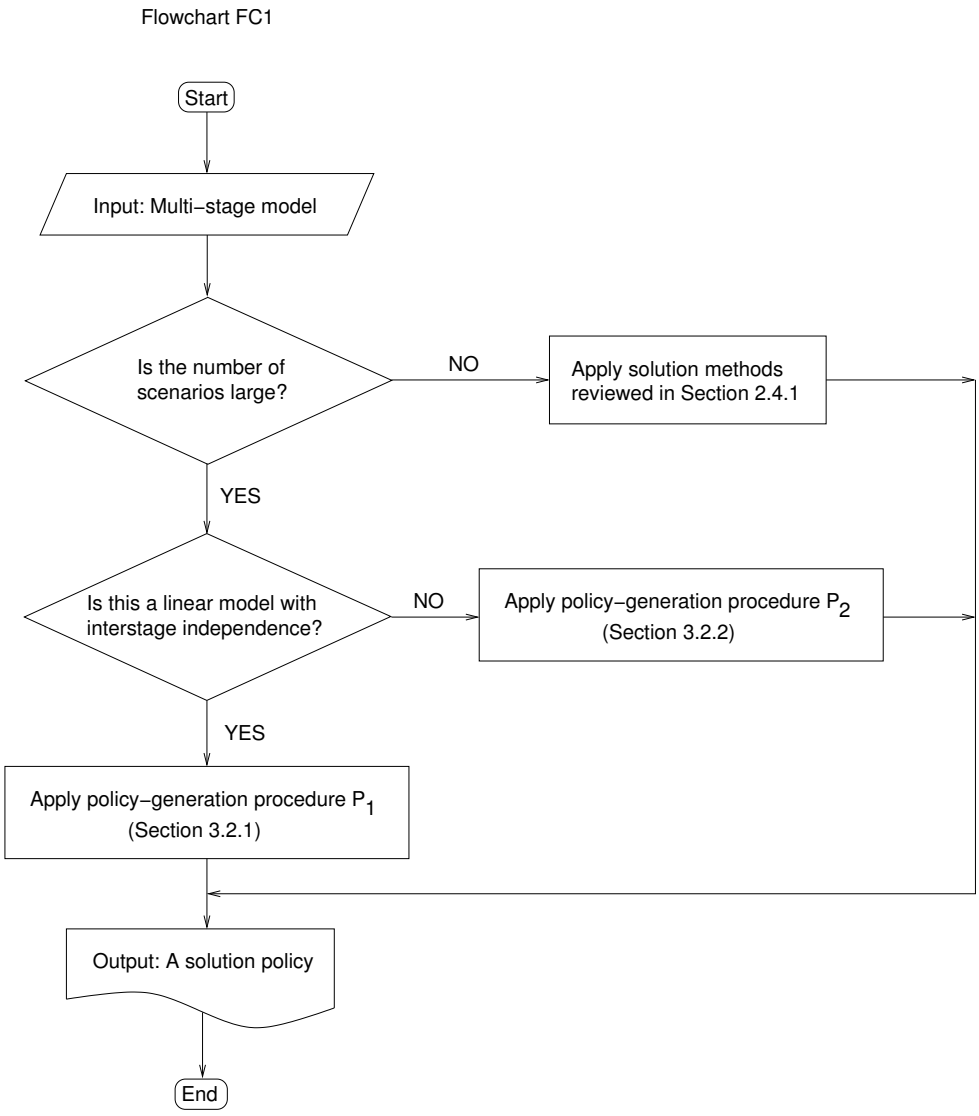

Figure 5.1: Policy generation for multi-stage stochastic programs.

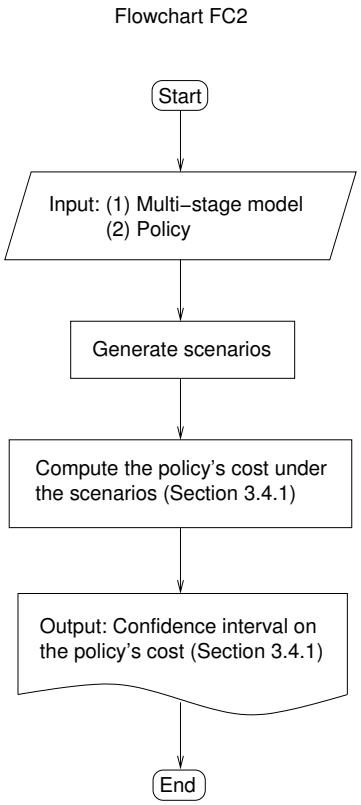

Figure 5.2: Scenario-based estimation of the policy's cost.

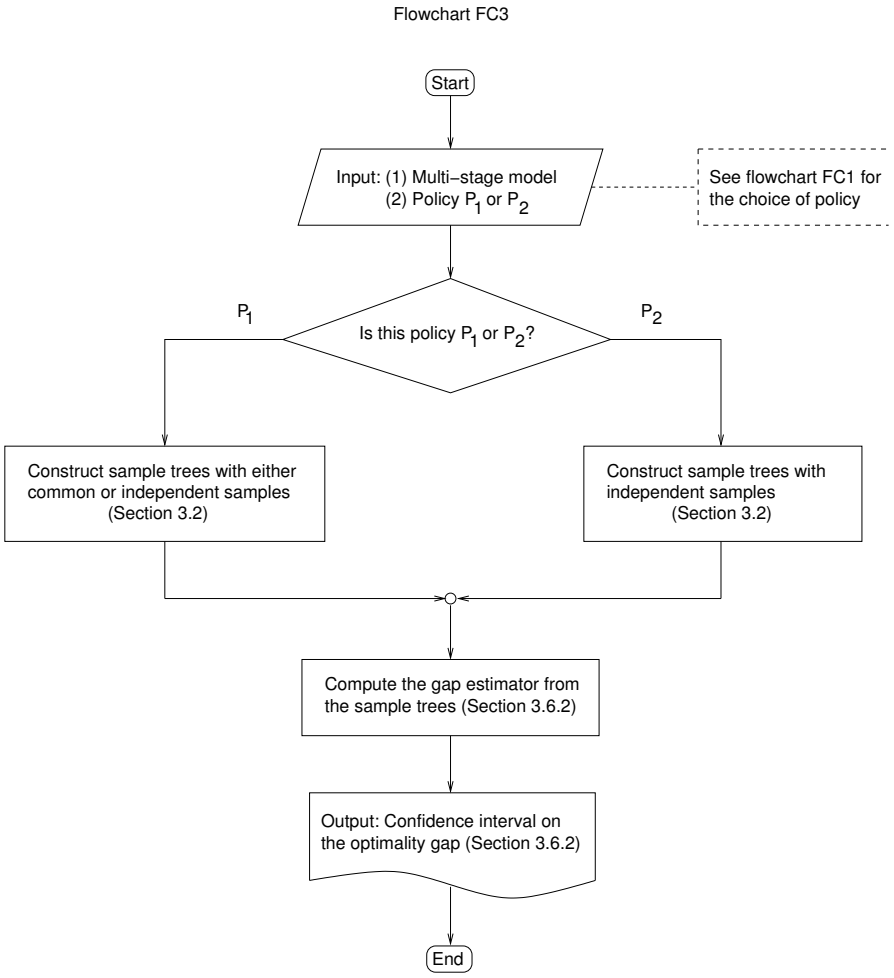

Figure 5.3: Establishing the policy's quality with the gap estimator.

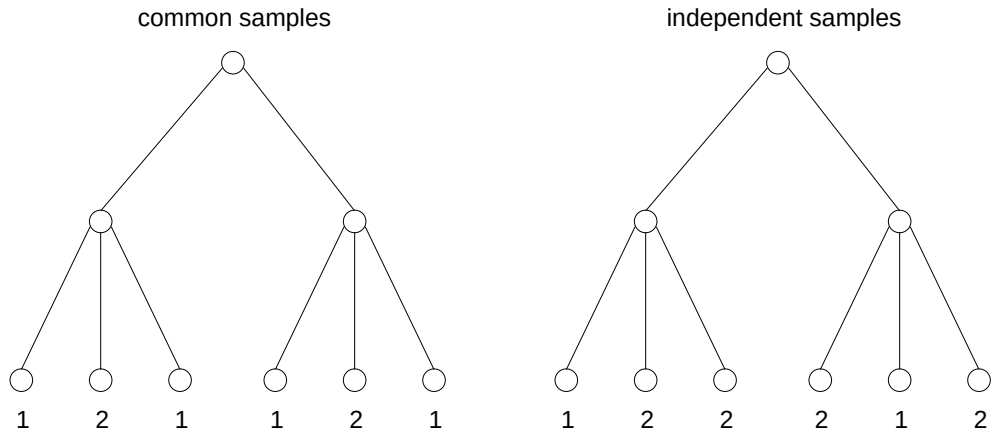

Figure 5.4: Instances of a sample tree with common and independent samples.

tic programs with a moderate-to-large number of stages and a large number of scenarios. Extensions that make this methodology more effective are therefore valuable. To obtain a confidence interval of the gap estimator, multiple replications are used otherwise asymptotic normality of the gap estimator is not guaranteed under practical assumptions. This can be inefficient because each replication involves solving a multi-stage stochastic program defined on a sample tree, and can be computationally demanding. One improvement of our methodology is then to apply a recent result (of two-stage stochastic programs) showing that a confidence interval on the optimality gap can be constructed with a single replication procedure. Under certain circumstances, this single replication may be more efficient.

In our policy generation methods, we use "naive" sample trees in generating a feasible solution at each node. A natural extension is to construct a "smart" sample tree so that an approximating problem yields a high quality policy with approximately the same computational effort. Other important improvements of the methodology include developing techniques to reduce the variance and bias of the gap estimator so that the width of the confidence interval on the optimality gap

is as small as possible for a fixed computational resource. These improvements will enable us to distinguish a policy with high quality in a more effective manner.

With this motivation, we investigate in Chapter 4 how to reduce the bias of the optimistic-bound estimator for multi-stage stochastic programs. We begin by studying bias reduction in a relatively simple multi-stage stochastic program known as American-option pricing problem. Under mild assumptions, explicit formulae for the bias approximation are derived. Using these formulae, we set up a convex optimization problem to allocate sample size to construct a sample tree with varying number of descendants to reduce the bias of the optimistic-bound estimator. Then, we extend the methodology to stochastic lot-sizing problem. From our computational results, the percentage of bias reduction of the optimistic-bound estimator in the American-style option pricing and the stochastic lot-sizing problems ranges from 5% to 57%.

An extension of these results to construct sample trees with varying number of descendants for a general multi-stage stochastic program with recourse is valuable since, as explained above, reducing the bias of the optimistic-bound estimator will improve the effectiveness of the policy's quality testing procedure, which may serve as a viable tool for a decision maker to solve real-world multi-stage stochastic programs with a large number of scenarios. At this point, we are, however, not optimistic in being able to directly extend the bias approximation formulae for multi-stage stochastic programs that involve more than one binary decision variable. Instead, we seek a heuristic procedure to allocate sample size using the insights obtained from American-style option pricing and stochastic lot-sizing problems.

Bibliography

[1] S. Ahmed, A. J. King, and G. Parija. A multi-stage stochastic integer programming approach for capacity expansion under uncertainty. *Journal of Global Optimization*. To appear.

[2] S. Ahmed and N.V. Sahinidis. An approximation scheme for stochastic integer programs arising in capacity expansion. *Operations Research*. To appear.

[3] H. Attouch and R.J.-B. Wets. Epigraphical processes: Laws of large numbers for random lsc functions. In *Séminaire d'Analyse Convexe*, volume 20, pages 13.1–13.29, Department Des Sciences Mathematiques, Universite Montpellier II, 1990.

[4] T.G. Bailey, P. Jensen, and D.P. Morton. Response surface analysis of two-stage stochastic linear programming with recourse. *Naval Research Logistics*, 46:753–778, 1999.

[5] M.S. Bazaraa, J.J. Jarvis, and H.D. Sherali. *Linear Programming and Network Flows*. John Wiley and Sons, second edition, 1990.

[6] E.M.L. Beale. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society*, 17B:173–184, 1955.

[7] E.M.L. Beale. The use of quadratic programming in stochastic linear programming. RAND P-2404, RAND Corporation, 1961.

[8] R.E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[9] J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[10] A.J. Berger, J. Mulvey, and A. Ruszczyński. An extension of the DQA algorithm to convex stochastic programs. *SIAM Journal of Optimization*, 4(4):735–753, 1994.

[11] M. Bertocchi, J. Dupačová, and V. Moriggia. Sensitivity of bond portfolio's behavior with respect to random movements in yield curve: A simulation study. *Annals of Operations Research*, 99:267–286, 2000.

[12] D.P. Bertsekas. *Dynamic Programming and Optimal Control, Volume 1 and 2*. Athena Scientific, Belmont, Massachusetts, 1995.

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

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

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

[16] J.R. Birge and F.V. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, New York, 1997.

[17] G.C.E. Boender, P. van Aalst, and F. Heemskerk. Modelling and management of assets and liabilities of pension plans in the Netherlands. In W.T. Ziemba and J.M. Mulvey, editors, *Worldwide Asset and Liability Modeling*, pages 561–580. Cambridge University Press, Cambridge, United Kingdom, 1998.

[18] M. Broadie and P. Glasserman. Pricing American-style securities using simulation. *Journal of Economic Dynamics and Control*, 21:1323–1352, 1997.

[19] M. Broadie and P. Glasserman. A stochastic mesh method for pricing high-dimensional American options. *Working Paper, Columbia University*, 1997.

[20] M. Broadie, P. Glasserman, and Z. Ha. Pricing American options by simulation using a stochastic mesh with optimized weights. In S.P. Uryasev, editor, *Probablistic Constrained Optimization*. Kluwer Academic, 2000.

[21] D.R. Cariño, T. Kent, D.H. Meyers, C. Stacy, M. Sylvanus, A.L. Turner, K. Watanabe, and W.T. Ziemba. The Russell-Yasuda Kasai model: An asset/liability model for a Japanese insurance company using multistage stochastic programming. *Interfaces*, 24:29–49, 1994.

[22] D.R. Cariño, D.H. Meyers, and W.T. Ziemba. Concepts, technical issues, and uses of the Russell-Yasuda Kasai financial planning model. *Operations Research*, 46:450–462, 1998.

[23] D.R. Cariño and W.T. Ziemba. Formulation of the Russell-Yasuda Kasai financial planning model. *Operations Research*, 46:433–449, 1998.

[24] C. Carøe and R. Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24:37–54, 1999.

[25] T.J. Carpenter, I.J. Lustig, and J.M. Mulvey. Formulating two-stage stochastic programs for interior point methods. *Operations Research*, 39:757–770, 1991.

[26] M. Casey and S. Sen. The scenario generation algorithm for multistage stochastic linear programming. *Working paper, University of Puget Sound*, 2002. http://www.math.ups.edu/∼mcasey.

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

[28] Z.L. Chen and W.B. Powell. Convergent cutting-plane and partial-sampling algorithm for multistage stochastic linear programs with recourse. *Journal of Optimization Theory and Applications*, 102(3):497–524, 1999.

[29] R.K. Cheung and W.B. Powell. An algorithm for multistage dynamic networks with random arcs capacities, with an application to dynamic fleet management. *Operations Research*, 44(6):951–963, 1996.

[30] G. Consigli and M.A.H. Dempster. Dynamic stochastic programming for asset-liability management. *Annals of Operations Research*, 81:131–161, 1998.

[31] G. Consigli, J. Dupačová, and S. Wallace. Generating scenarios for multistage stochastic programs. *Annal of Operations Research*, 100:25–53, 2000.

[32] G.B. Dantzig. Linear programming under uncertainty. *Management Science*, 1:197–206, 1955.

[33] G.B. Dantzig and P.W. Glynn. Parallel processors for planning under uncertainty. *Annals of Operations Research*, 22:1–21, 1990.

[34] G.B. Dantzig and G. Infanger. Multi-stage stochastic linear programs for portfolio optimization. *Annals of Operations Research*, 45:59–76, 1993.

[35] M.A.H. Dempster. Sequential importance sampling algorithms for dynamic stochastic programming. *Judge Institute of Management Working Paper 32/98*, 1998.

[36] M.A.H. Dempster and G. Consigli. The CALM stochastic programming model for dynamic asset-liability management. In J.M. Mulvey and W.T.

Ziemba, editors, *World Wide Asset and Liability Modelling*, pages 464–500. Cambridge University Press, 1998.

[37] M.A.H. Dempster and R.T. Thompson. EVPI-based importance sampling solution procedures for multistage stochastic linear programmes on parallel MIMD architectures. *Annal of Operations Research*, 90:161–184, 1999.

[38] S.P. Dokov and D.P. Morton. Second-order lower bounds on the expectation of a convex function. *Stochastic Programming E-Print Series*, 2001. www.speps.info.

[39] C. Donohue. *Stochastic Network Programming and the Dynamic Vehicle Allocation Problem*. Ph.D. dissertation, The University of Michigan, Ann Arbor, 1996.

[40] J. Dupačová, N. Gröwe-Kuska, and W. Römisch. Scenario reduction in stochastic programming: An approach using probability metrics. *Mathematical Programming*, To appear.

[41] J. Dupačová, J. Hurt, and J. Štěpán. *Stochastic Modeling in Economics and Finance*. Kluwer Academic Publishers, 2002.

[42] J. Dupačová and P. Popela. Melt control: charge optimization via stochastic programming. In S.W. Wallace and W.T. Ziemba, editors, *Applications of Stochastic Programming*. MPS-SIAM Series in Optimization. forthcoming.

[43] J. Dupačová and K. Sladký. Comparison of multistage stochastic programs with recourse and stochastic dynamic programs with discrete time. *ZAMM·Z. Angew. Math. Mech.*, 82:753–765, 2002.

[44] J. Dupačová. Application of stochastic programming under incomplete information. *Journal of Computational and Applied Mathematics*, 56:113–125, 1994.

[45] J. Dupačová. Mutistage stochastic programs: The state-of-the-art and selected bibliography. *Kybernetika*, 31(2):151–174, 1995.

[46] N.C.P. Edirisinghe. Bound-based approximations in multistage stochastic programming: Nonanticipativity aggregation. *Annals of Operations Research*, 85:103–127, 1999.

[47] N.C.P. Edirisinghe and W.T. Ziemba. Tight bounds for stochastic convex programs. *Operations Research*, 40:660–677, 1992.

[48] H.P. Edmundson. Bounds on the expectation of a convex function of a random variable. Technical report, The Rand Corporation Paper 982, Santa Monica, California, 1956.

[49] G.D. Eppen, R.K. Martin, and L. Schrage. A scenario approach to capacity planning. *Operations Research*, 37:517–527, 1989.

[50] Y. Ermoliev. Stochastic quasigradient methods. In Y. Ermoliev and R.J.-B. Wets, editors, *Numerical Techniques for Stochastic Optimization*, pages 141–185. Springer-Verlag, Berlin, 1988.

[51] Y. Ermoliev and R.J.-B. Wets. Stochastic programming, an introduction. In Y. Ermoliev and R.J.-B. Wets, editors, *Numerical Techniques for Stochastic Optimization*, pages 1–32. Springer-Verlag, Berlin, 1988.

[52] A. Ferguson and G.B. Dantzig. The allocation of aircraft to routes: an example of linear programming under uncertain demands. *Management Science*, 3:45–73, 1956.

[53] S.-E. Fleten, S.W. Wallace, and W.T. Ziemba. Hedging electricity portfolios via stochastic programming. In C. Greengard and A. Ruszczynski, editors, *Decision Making Under Uncertainty: Energy and Power, IMA Volumes on Mathematics and Its Applications*, pages 71–94. Springer-Verlag, 2002.

[54] L.F. Frantzeskakis and W.P. Powell. A successive linear approximation procedure for stochastic, dynamic vehicle allocation problems. *Transportation Science*, 24:40–57, 1990.

[55] K. Frauendorfer. Barycentric scenario trees in convex multistage stochastic programming. *Mathematical Programming*, 75:277–293, 1996.

[56] H.I. Gassmann. Optimal harvest of a forest in the presence of uncertainty. *Canadian Journal of Forest Research*, 19:1267–1274, 1989.

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

[58] H.I. Gassmann and E. Schweitzer. Proposed extensions to the SMPS input format for stochastic linear programs. *Dalhousie School of Business Administration Working Paper, WP-96-1*, 1996.

[59] H.I. Gassmann and S.W. Wallace. Solving linear programs with multiple right-hand sides: Pricing and ordering schemes. *Annals of Operations Research*, 64:237–259, 1996.

[60] R. Geske and H.E. Johnson. The American put options valued analytically. *Journal of Finance*, 39:1511–1524, 1984.

[61] N. Gröwe, W. Römisch, and R. Schultz. A simple recourse model for power dispatch under uncertain demand. *Annals of Operations Research*, 59:135–164, 1995.

[62] N. Gröwe-Kuska, K. Kiwiel, M. Nowak, W. Römisch, and I. Wegner. Power management in a hydro-thermal system under uncertainty by Lagrangian relaxation. *IMA Volumes of Mathematics and its Applications*, 128, 2002.

[63] G. Hadley. *Nonlinear and dynamic programming*. Addison-Wiley, 1964.

[64] K. Haugen, A. Løkketangen, and D. Woodruff. Progressive hedging as a meta-heuristic applied to stochastic lot-sizing. *European Journal of Operations Research*, 132:103–109, 2001.

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

[66] J.L. Higle and S. Sen. Stochastic decomposition: an algorithm for two-stage linear programs with recourse. *Mathematics of Operations Research*, 16:650–669, 1991.

[67] J.L. Higle and S. Sen. Multistage stochastic convex programs: Duality and its implications. *The Stochastic Programming E-Print Series*, 2002. www.speps.info.

[68] M.R. Holmer. The asset-liability management system at Fannie Mae. *Interfaces*, 24:3–21, 1994.

[69] M.R. Holmer. Integrated asset-liability management: An implementation case study. In W.T. Ziemba and J.M. Mulvey, editors, *Worldwide Asset and Liability Modeling*, pages 581–605. Cambridge University Press, Cambridge, United Kingdom, 1998.

[70] M.R. Holmer and S.A. Zenios. The productivity of financial intermediation and the technology of financial product management. *Operations Research*, 43:970–982, 1995.

[71] K. Høyland, M. Kaut, and S. Wallace. A heuristic for moment-matching scenario generation. *Computational Optimization and Applications*, To appear.

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

[73] G. Infanger and D.P. Morton. Cut sharing for multistage stochastic linear programs with interstage dependency. *Mathematical Programming*, 75:241–256, 1996.

[74] J. Jacobs, G. Freeman, J. Grygier, D. Morton, G. Schultz, K. Staschus, and J. Stedinger. Socrates: a system for scheduling hydroelectric generation under uncertainty. *Annals of Operations Research*, 59:99–133, 1995.

[75] J.L. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta. Mathematica*, 30:175–193, 1906.

[76] P. Kall. Approximation to optimization problems: An elementary review. *Mathematics of Operations Research*, 11:9–18, 1986.

[77] P. Kall. Bounds for and approximations to stochastic linear programs with recourse. In K. Marti and P. Kall, editors, *Stochastic Programming Methods and Technical Applications, Lecture Notes in Economics and Mathematical Systems*, volume 458. Springer, 1998.

[78] P. Kall and S.W. Wallace. *Stochastic Programming*. John Wiley & Sons, Chichester, 1994.

[79] V. Kaňková. A note on multistage stochastic programs: Markov dependence. Reseach Report 2020, Academy of Sciences of the Czech Republic, Institute of Information Theory and Automation, Prague, Czech Republic, 2001.

[80] E. Katok, W. Tarantino, and T.P. Harrison. Investment in production resource flexibility: An empirical investigation of methods for planning under uncertainty. *Naval Research Logistics*. To appear.

[81] E. Katok, W. Tarantino, and R. Tiedeman. Flexibility planning and technology management at Jeppesen Sanderson Inc. *Interfaces*, 31:7–29, 2001.

[82] J.E. Kelley. The cutting plane method for solving convex programs. *SIAM Journal of Industrial and Applied Mathematics*, 8:703–712, 1960.

[83] A. King. Stochastic programming problems: Examples from the literature. In Y. Ermoliev and R.J.-B. Wets, editors, *Numerical Techniques for Stochastic Optimization*, pages 543–567. Springer-Verlag, Berlin, 1988.

[84] A.J. King and R.J.-B. Wets. Epi-consistency of convex stochastic programs. *Stochastics and Stochastics Reports*, 34:83–92, 1991.

[85] P. Korapaty. *Constructing scenario trees for pricing American-style options*. The University of Texas at Austin, 2001. M.S. Report.

[86] A.M. Law and W.D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, Boston, third edition, 2000.

[87] J. Linderoth, A. Shapiro, and S. Wright. The empirical behavior of sampling methods for stochastic programming. Technical report, Computer Sciences Department, University of Wisconsin-Madison, February 2001.

[88] A. Løkketangen and D. Woodruff. Progressive hedging and Tabu search applied to mixed integer (0,1) multistage stochastic programming. *Journal of Heuristics*, 2:111–128, 1996.

[89] G. Lulli and S. Sen. Stochastic batch-sizing. In D.L. Woodruff, editor, *Network Interdiction and Stochastic Integer Programming*. Kluwer Academic Publishers, Boston, 2003.

[90] A. Madansky. Bounds on the expectation of a convex function of a multivariate random variable. *Annals of Mathematical Statistics*, 30:743–746, 1959.

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

[92] D. Morton and E. Popova. Monte Carlo simulations for stochastic optimization. In C.A. Floudas and P.M. Pardalos, editors, *Encyclopedia of Optimization*. Kluwer Academic Publishers, 2001.

[93] D.P. Morton. An enhanced decomposition algorithm for multistage stochastic hydroelectric scheduling. *Annals of Operations Research*, 64:211–235, 1996.

[94] D.P. Morton, J. Salmerón, and R.K. Wood. A stochastic program for optimizing military sealift subject to attack. *Stochastic Programming E-Print Series*, 2003. www.speps.info.

[95] D.P. Morton and R.K. Wood. On a stochastic knapsack problem and generalizations. In D.L. Woodruff, editor, *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search: Interfaces in Computer Science and Operations Research*, pages 149–168. Kluwer Academic Publishers, Boston, 1998.

[96] J.M. Mulvey. Generating scenarios for the Towers Perrin investment system. *Interfaces*, 26:1–15, 1996.

[97] J.M. Mulvey, C. Gould, and C. Morgan. An asset and liability management system for Towers Perrin-Tillinghast. *Interfaces*, 30:96–114, 2000.

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

[99] V.I. Norkin, G.Ch. Pflug, and A. Ruszczyński. A branch and bound method for stochastic global optimization. *Mathematical Programming*, 83:425–450, 1998.

[100] T. Pennanen and M. Koivu. Integration quadratures in discretization of stochastic programs. *Stochastic Programming E-Print Series*, 2002. www.speps.info.

[101] M.V.F. Pereira and L.M.V.G. Pinto. Stochastic optimization of a multireservoir hydroelectric system—a decomposition approach. *Water Resources Research*, 21:779–792, 1985.

[102] M.V.F. Pereira and L.M.V.G. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52:359–375, 1991.

[103] G.Ch. Pflug. Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical Programming*, 89:251–271, 2001.

[104] A.B. Philpott, M. Craddock, and H. Waterer. Hydro-electric unit commitment subject to uncertain demand. *European Journal of Operational Research*, 125:410–424, 2000.

[105] Y. Pochet and L. Wolsey. Lot-size models with backlogging: strong reformulations and cutting planes. *Mathematical Programming*, 40:317–335, 1988.

[106] W. Powell. A comparative review of alternative algorithms for the dynamic vehicle allocation problem. In B.L. Golden and A.A. Assad, editors, *Vehicle Routing: Methods and Studies*. North-Holland, New York, 1988.

[107] A. Prékopa. *Stochastic Programming*. Kluwer Academic Publishers, Dordrecht, 1995.

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

[109] R.T. Rockafellar and R.J.-B. Wets. *Variational Analysis*. Springer-Verlag, Berlin, 1998.

[110] C.H. Rosa and A. Ruszczyński. On augmented Lagrangian decomposition methods for multistage stochastic programs. *Annals of Operations Research*, 64:289–309, 1996.

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

[112] A. Ruszczyński. Parallel decomposition of multistage stochastic programming problems. *Mathematical Programming*, 58:201–228, 1993.

[113] A. Ruszczyński and A. Świetanowski. Accelerating the regularized decomposition method for two stage stochastic linear problems. *European Journal of Operational Research*, 101:328–342, 1997.

[114] S. Sen. Stochastic programming: Computatinal issues and challenges. In S. Gass and C. Harris, editors, *Encyclopedia of OR/MS*. Kluwer Academic Publishers, 2001.

[115] A. Shapiro. Statistical inference of multistage stochastic programming problems. *Optimization Online*, 2002. http://www.optimization-online.org.

[116] S. Takriti, J.R. Birge, and E. Long. A stochastic model for the unit commitment problem. *IEEE Transactions on Power Systems*, 11:1497–1508, 1996.

[117] S. Takriti, B. Krasenbrink, and L.S.-Y. Wu. Incorporating fuel constraints and electricity spot prices into the stochastic unit commitment problem. *Operations Research*, 48:281–293, 2000.

[118] G.C. Tiao and G.E.P. Box. Modeling multiple time series with applications. *Journal of the American Statistical Association*, 76:802–816, 1981.

[119] R.M. Van Slyke and R.J.-B. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17:638–663, 1969.

[120] B. Verweij, S. Ahmed, A. Kleywegt, G. Nemhauser, and A. Shapiro. The sample average approximation method applied to stochastic vehicle routing problems: a computational study. *Computational and Applied Optimization*, 24:289–333, 2003.

[121] S.W. Wallace. Decision making under uncertainty: Is sensitivity analysis of any use? *Operations Research*, 48(1):20–25, 2000.

[122] R.J.-B. Wets. Programming under uncertainty: The complete problem. *Zeitschrift für Warscheinlichkeitstheorie und Verw. Gebiete*, 4:316–339, 1966.

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

[124] R.J.-B. Wets. Stochastic programming: Solution techniques and approximation schemes. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming: State-of-the-art 1982*, pages 560–603. Springer-Verlag, 1983.

[125] R.J.-B. Wets. Large-scale linear programming techniques in stochastic programming. In Y. Ermoliev and R.J.-B. Wets, editors, *Numerical Techniques for Stochastic Optimization*. Springer-Verlag, Berlin, 1988.

[126] R.J.-B. Wets. Stochastic programming. In G.L. Nemhauser, A.H.G. Rinnoy Kan, and M.J. Todd, editors, *Handbook on Operations Research and Management Science*, volume 1, pages 573–629. North-Holland, 1989.

[127] R.J. Wittrock. Advances in a nested decomposition algorithm for solving staircase linear programs. Technical Report SOL 83-2, Systems Optimization Laboratory, Stanford University, Stanford, California, 1983.

[128] W.T. Ziemba. Stochastic programs with simple recourse. In P.L. Hammer and G. Zoutendijk, editors, *Mathematical Programming in Theory and Practice*, pages 213–273. North-Holland, Amsterdam, 1974.

[129] W.T. Ziemba and J.M. Mulvey, editors. *Worldwide Asset and Liability Modeling*. Cambridge University Press, Cambridge, United Kingdom, 1998.

Vita

Born on June 23, 1971, in Bangkok, Thailand, Anukal Chiralaksanakul is a son of Lertnarong Chiralaksanakul and Notha Chiralaksanakul. After completing his study at Triamudomsuksa High School, Bangkok, in 1988, he entered Chulalongkorn University, Bangkok, where he earned a B.S. in Mechanical Engineering with a second-class honor in 1992. He had worked as a mechanical engineer at Sedco-Forex (Schlumberger) in Thailand and Pakistan until he began his graduate study at the University of Texas at Austin in 1994.

After receiving a M.S. in Mechanical Engineering in 1997, Anukal Chiralaksanakul continued his graduate study in Operations Research. Currently, he is a doctoral candidate in the Graduate Program in Operations Research at the University of Texas at Austin. His dissertation research draws upon techniques from large-scale optimization and Monte Carlo sampling to develop computational methods for a class of computationally difficult models known as multi-stage stochastic programs with recourse. He has worked as a summer intern at Southwestern Bell, Schlumberger, and i2 Technologies on various projects related to resource planning and scheduling, and as a teaching assistant to courses in optimization, probability, and statistics.

Permanent address: 2230/3 Narathiwat 18
Bangkok 10120, Thailand.

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.