SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305-4022

**Algorithmic Advances In
Stochastic Programming**

by

David P. Morton

TECHNICAL REPORT SOL 93-6

July 1993

Research and reproduction of this report were partially supported by the Boeing Contract W297550; the Department of Energy Grant DE-FG03-92ER5116; NSF Grant ECS-8906260 and the Electric Power Research Institute Contract RP8010-09. 25116

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do NOT necessarily reflect the views of the above sponsors.

Also issued as Operations Research Department Technical Report 93-17. Reproduction in whole or in part is permitted for any purposes of the United States Government. This document has been approved for public release and sale; its distribution is unlimited.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Abstract

Practical planning problems with deterministic forecasts of inherently uncertain parameters often yield unsatisfactory solutions. Stochastic programming formulations allow uncertain parameters to be modeled as random variables with known distributions, but the size of the resulting mathematical programs can be formidable. Decomposition-based algorithms take advantage of special structure and provide an attractive approach to such problems. We consider two classes of decomposition-based stochastic programming algorithms. The first type of algorithm addresses problems with a “manageable” number of scenarios. The second class incorporates Monte Carlo sampling within a decomposition algorithm.

We develop and empirically study an enhanced Benders decomposition algorithm for solving multistage stochastic linear programs within a prespecified tolerance. The enhancements include warm start basis selection, preliminary cut generation, the multicut procedure, and decision tree traversing strategies. Computational results are presented for a collection of “real-world” multistage stochastic hydroelectric scheduling problems.

Recently, there has been an increased focus on decomposition-based algorithms that use sampling within the optimization framework. These approaches hold much promise for solving stochastic programs with many scenarios. A critical component of such algorithms is a stopping criterion to ensure the quality of the solution. With this as motivation, we develop a stopping rule theory for algorithms in which bounds on the optimal objective function value are estimated by sampling. Rules are provided for selecting sample sizes and terminating the algorithm under which asymptotic validity of confidence interval statements for the quality of the proposed solution can be verified. Issues associated with the application of this theory to two sampling-based algorithms are considered, and preliminary empirical coverage results are presented.

Acknowledgements

I would like to express my most sincere appreciation to my adviser, George B. Dantzig, for his guidance and encouragement throughout all of my work at Stanford. His approach to optimization and planning under uncertainty with continual emphasis on solving real-world problems has been very inspiring. This dissertation would not have been possible without his support, and its presentation is much improved due to his helpful suggestions.

Gerd Infanger played an instrumental role in introducing me to the field of stochastic programming. I wish to thank him for his constant support and enthusiasm which has been invaluable during my research at Stanford. I am also grateful to Peter W. Glynn for his guidance with respect to the stopping rules topic presented in Chapters 4 and 5.

My work at PG&E has been an integral part of my graduate education, and this is reflected in the research presented in Chapter 2. I am very grateful to the members of the MTHTO group, headed by Konstantin Staschus, including Jonathan M. Jacobs, Jan C. Grygier, Gary L. Schultz, Zhiming Wang, and Bin Zhang. Zhiming Wang deserves specific thanks for many valuable discussions on tree traversing strategies.

I have had many useful and enlightening discussions with fellow graduate students Richard Chatwin, Chuanpu Hu, P.V. Krishnarao, and Adrienne Liefer. I would also like to especially thank Alamura S. Krishna for providing a sounding board for many ideas and for his insight into many problems.

My parents have been extremely supportive and have encouraged me throughout my entire education; I would like to give them special thanks. I am deeply grateful to my wife, Erin, for her continual encouragement and support throughout my days at Stanford.

Contents

Abstract	ii
Acknowledgements	iii
1 Introduction to Stochastic Programming	1
1.1 Introduction	1
1.2 Problem Statement	2
1.3 Example Applications	4
1.3.1 Hydroelectric Scheduling	5
1.3.2 Portfolio Management	7
1.3.3 Forest Harvest Planning	8
1.4 Benders Decomposition	9
1.4.1 Two-Stage Deterministic Linear Programs	10
1.4.2 Two-Stage Stochastic Linear Programs	15
1.4.3 Multistage Stochastic Linear Programs	17
1.5 Value of the Stochastic Solution	19
2 An Enhanced Decomposition Algorithm for SLP-T	22
2.1 Introduction	22
2.2 Valid Cut Generation	23
2.3 Test Problems	27
2.4 Warm Start	29
2.5 Advanced Start	32
2.5.1 Expected Value Method	32
2.5.2 Prespecified Decisions Methods	33
2.6 Multicut Algorithms	37
2.7 Tree Traversing Strategies	39
2.7.1 The Strategies	40
2.7.2 Finite Termination Theorem	45
2.7.3 Computational Results	46
2.8 Direct LP Optimizers	47
2.9 Summary	49

3	Decomposition and Sampling-Based Algorithms	50
3.1	Introduction	50
3.2	Importance Sampling	51
3.3	Two Stage Algorithm	52
3.4	Multistage Algorithm	56
3.5	Interstage Dependence and Cut Sharing	60
3.6	Summary	64
4	Stopping Rules: Theory	66
4.1	Introduction	66
4.2	Stopping Rules: Problem Statement	66
4.3	Stopping Rules: Normal Differences	69
4.4	Stopping Rules: CLT Differences	74
4.5	Stopping Rules: History Dependent CLT Differences	81
4.6	Stopping Rules: Distribution-Free Differences	83
4.7	Stopping Rules: CLT Differences and Sample Variances	88
4.8	Summary	90
5	Stopping Rules: Applications	92
5.1	Introduction	92
5.2	The SLP-T Algorithm	93
5.3	The SLP-2 Algorithm	96
5.3.1	Algorithm	96
5.3.2	Upper Bound	96
5.3.3	Lower Bound	97
5.3.4	Stopping Rule Assumptions	102
5.4	Empirical Coverage Results	105
5.5	Summary and Areas for Future Research	110
	Bibliography	111

Chapter 1

Introduction to Stochastic Programming

1.1 Introduction

Many practical planning problems involve the allocation of scarce resources among competing activities. Linear programming has proven to be a very powerful optimization tool for such problems in a wide variety of applications. Linear programming models satisfy the following four assumptions: (i) additivity, (ii) proportionality, (iii) divisibility, and (iv) certainty. The optimization fields of nonlinear programming and mixed integer programming deal with mathematical models that fail to satisfy some combination of assumptions (i), (ii), and (iii). Sensitivity analysis is a well-known linear programming technique that attempts to ascertain the behavior of “optimal” solutions when some of the parameters of the underlying model are not known with certainty. In many practical planning problems, however, the impact of uncertainty is too great for local error analysis to yield useful results. Stochastic programming concerns mathematical programs in which we relax assumption (iv) and treat some of the parameters as random variables with known distribution. The field was introduced in the 1950’s by Dantzig [15], Beale [3], and Charnes and Cooper [13]. Dantzig’s paper, entitled “Linear Programming Under Uncertainty,” shows how to reduce a stochastic model to a certainty-equivalent linear program, which in practice is often enormous in size. Much of the research in stochastic programming has focused on techniques for solving relevant classes of typically very large-scale certainty-equivalent programs by exploiting their special structure.

This dissertation is organized as follows. In the remainder of this chapter we present a mathematical formulation of a multistage stochastic linear program with recourse and describe three example applications. We also review Benders decomposition algorithm as applied to two- and multistage

problems. In Chapter 2, several enhancements to the traditional nested decomposition algorithm are explored. Computational results are presented for a collection of multistage stochastic hydroelectric scheduling test problems. In Chapter 3, we review two recently proposed decomposition-based algorithms that incorporate Monte Carlo sampling. First we describe an algorithm introduced by Dantzig and Glynn [16], and further developed by Infanger [36], for the two-stage problem that uses a variance reduction technique called importance sampling. We then describe an algorithm due to Pereira and Pinto [51] for multistage problems in which there are only a few stochastic branches per stage but possibly many stages. Motivated by these two algorithms, we develop a stopping rule theory for a general class of algorithms with optimal objective function bounds estimated by sampling in Chapter 4. The stopping rules provide a sample size formula and termination criterion under which one can ensure asymptotic validity of desired confidence intervals. In Chapter 5, we apply the stopping rule theory to the algorithms presented in Chapter 3. Preliminary empirical coverage results are also presented.

1.2 Problem Statement

An important problem within the class of stochastic programming problems is the multistage stochastic linear program with recourse. A T -stage stochastic linear program with recourse (SLP-T) may be formulated as follows:

$$\begin{aligned}
 & \text{minimize} && \sum_{t=1}^T \sum_{\omega_t \in \Omega_t} p_t^{\omega_t} c_t^{\omega_t} x_t^{\omega_t} \\
 & \text{subject to} && -B_{t-1}^{\omega_t} x_{t-1}^{a(\omega_t)} + A_t^{\omega_t} x_t^{\omega_t} = b_t^{\omega_t}, \quad x_t^{\omega_t} \geq 0, \quad \omega_t \in \Omega_t \\
 & \text{for} && t = 1, \dots, T \\
 & \text{where} && B_0^{\omega_1} \equiv 0.
 \end{aligned}$$

SLP-T

The sample space for stage t is denoted Ω_t , and a sample point (scenario) in Ω_t is denoted ω_t . We assume a stage t (≥ 2) scenario, ω_t , has a unique stage $t-1$ ancestor denoted $a(\omega_t)$, and a stage t ($< T$) scenario has a set of descendent scenarios denoted $\Delta(\omega_t)$. More generally, the Δ notation may be applied to a set: $\Delta(\mathcal{S}) = \bigcup_{s \in \mathcal{S}} \Delta(s)$. A stage t realization, $\xi_t(\omega_t) = (c_t^{\omega_t}, b_t^{\omega_t}, B_{t-1}^{\omega_t}, A_t^{\omega_t})$, is to be read column-wise as a vector in $\mathbb{R}^{N_t}$, where $N_t = n_t + m_t + m_t \cdot n_{t-1} + m_t \cdot n_t$; $A_t^{\omega_t}$ is an $m_t \times n_t$ matrix and the remaining matrices and vectors are dimensioned to conform. We assume ξ_t

has finite support and a probability mass function given by $P\{\xi_t = \xi_t(\omega_t)\} = p_t^{\omega_t}$. For notational convenience, we assume the first stage sample space, Ω_1 is a singleton set and $\xi_1(\omega_1)$ represents the known state at the time decisions are made in the first stage; clearly, $p_1^{\omega_1}$ has value one. In general, a stage t realization depends on the outcomes of earlier stages. At the time decisions are made in stage t , the observation $\xi_t(\omega_t)$ and the previous stage's decision $x_{t-1}^{a(\omega_t)}$ are known to the decision maker. SLP-T has $\sum_{t=1}^T m_t |\Omega_t|$ structural constraints and $\sum_{t=1}^T n_t |\Omega_t|$ decision variables where $|\Omega_t|$ denotes the cardinality of Ω_t .

Figure 1.1: An example scenario tree

Figure 1.1 depicts a three stage scenario tree; there are four possible stage 3 realizations, $\Omega_3 = \{4, 5, 6, 7\}$ with corresponding probabilities $p_3^{(4)}$, $p_3^{(5)}$, $p_3^{(6)}$, and $p_3^{(7)}$. The ancestor and descendent notation for this example are illustrated below:

$$a(2) = a(3) = 1, \quad a(4) = a(5) = 2, \quad a(6) = a(7) = 3,$$

$$\Delta(1) = \{2, 3\}, \quad \Delta(2) = \{4, 5\}, \quad \text{and} \quad \Delta(3) = \{6, 7\}.$$

In the special case of interstage independence, the appropriate third stage realizations and conditional probabilities are equal:

$$\begin{aligned} \xi(4) &= \xi(6), \quad \xi(5) = \xi(7), \\ \frac{p_3^{(4)}}{p_2^{(2)}} &= \frac{p_3^{(6)}}{p_2^{(3)}}, \quad \text{and} \quad \frac{p_3^{(5)}}{p_2^{(2)}} = \frac{p_3^{(7)}}{p_2^{(3)}}. \end{aligned}$$

The SLP-T formulation allows for full interstage dependence of the stochastic parameters. In SLP-T, the decisions occur and the uncertainties unfold in the following manner. A first stage decision, x_1 , is made with knowledge of (c_1, b_1, A_1) and distributions, $p_t^{\omega_t}$, on future data; next, an observation, $\xi_2(\omega_2)$, is revealed and the second stage decision, $x_2^{\omega_2}$, is made knowing this data, the first stage decision, and the conditional probability distributions on future data ... The goal is to

find a first stage decision, x_1 , that minimizes the expected cost of operating the system modeled by SLP-T over T stages.

SLP-2 was first studied, independently, by Dantzig [15] and Beale [3]. Since then, SLP-T and several of its special cases have been widely studied in the literature. Two special cases are: (i) the two-stage program with *fixed* recourse in which the second stage structural matrix is nonstochastic, i.e., $A_2^{\omega_2} = A_2$ (e.g., Dantzig and Madansky [18], Kall [40], and Wets [65]) and (ii) the two-stage program with *simple* recourse in which $A_2^{\omega_2} = [I, -I]$ (e.g., Dantzig [14], Everitt and Ziemba [25], Wets [63], and Ziemba [68]).

The multistage stochastic linear program with recourse (SLP-T) is a special case of a more general class of problems called stochastic programming problems. An example of another class of stochastic programs is “chance-constrained” programs introduced by Charnes and Cooper [13]. In this type of stochastic program the goal is to minimize the expected cost while ensuring the probability that the solution is feasible satisfies a prespecified minimum tolerance. See, for example, Prékopa [55] [54] and Szantai [60]. The present development, however, will be restricted to stochastic programs satisfying the assumptions of the SLP-T framework.

1.3 Example Applications

Multistage stochastic linear programming models arise in many different practical settings. Indeed, many real-world problems currently solved as deterministic linear programs may be more realistically modeled as stochastic linear programs. An excellent reference to get a flavor for the variety of applications is Ermoliev and Wets [24]; chapters 22–28 of this book are devoted to stochastic programming applications. Examples of applications that may be modeled as multistage stochastic linear programs with recourse include hydroelectric scheduling, portfolio management, and forest harvest planning. In order to motivate the topic of stochastic programming, we will describe models for each of these applications in the following three sections.

Stochastic programs can be significantly larger and more costly to solve than the expected value problem in which random parameters are replaced by their means. On the other hand the quality of the expected value solution in the stochastic environment may be quite poor. See Birge [8] (and

references cited therein) for techniques to bound the value of the stochastic solution.

1.3.1 Hydroelectric Scheduling

Jacobs et al. [39] describe a stochastic hydro-thermal scheduling system with a planning horizon of about two years that is currently being implemented at the Pacific Gas & Electric Company. In this section, we describe a simplified version of the stochastic hydroelectric scheduling module of this system. In this model, energy generation is classified into hydro energy and all other energy collectively referred to as “thermal” energy. We assume hydro energy may be produced at no cost, but thermal energy generation incurs cost according to an increasing, nonlinear convex function. A simplistic justification for this form of the thermal cost function is that the thermal system consists of a number of generation sources; each source has finite generation capacity, and the cost of operating each thermal source is an increasing, linear function of its generation level. Thus, to schedule the thermal system *given* both the demand for energy and a hydro energy generation schedule, one simply loads thermal plants in decreasing order of efficiency to satisfy the remaining demand for energy. The goal is to operate the hydro-thermal system with minimum expected cost. Restated, we wish to operate the hydro system so as to maximize the expected savings from *avoided* thermal generation costs. The uncertain parameters in the stochastic hydro scheduling problem include natural inflow into reservoirs and the marginal values of energy. Consider two operating extremes for the first stage hydro decision: (i) generate a large amount of hydro energy (ii) generate a small amount of hydro energy. Strategy (i) may work well in a scenario in which there are large inflows into the reservoirs in the next stage, but under a scenario in which future inflows are small, a high cost is incurred because we will be forced to operate inefficient and costly thermal plants. On the other hand, if we follow strategy (ii) and save most of the water, then we will incur a high first stage cost and in a second stage scenario with large natural inflows we may be forced to spill water due to finite storage and generation capacities; these spills represent lost potential energy and lost future cost savings.

We now describe this model within the framework of SLP-T. The hydro system may be viewed as a network consisting of a number of reservoirs (nodes) that are interconnected by rivers, canals, and spillways (arcs). In this system, energy is generated as water flows through powerhouses. In

the SLP-T formulation, $A_t^{wt} = A$ is the node-arc incidence matrix for the physical network. This matrix contains two types of arcs: energy generation and other water transport arcs. Arcs of a third type are contained in the matrix $B_{t-1}^{wt} = B$; these “transition in time” arcs are used to equate the amount of water left in a reservoir at the end of one stage with the amount in the same reservoir at the beginning of the next stage. Generation, spill, and reservoir bounds appear as simple bounds on these three arc types. Initial reservoir volumes are contained in the vector b_1 ; subsequent b_t^{wt} vectors contain the uncertain future natural inflows into the reservoirs. Energy generation arcs have objective function coefficients that represent marginal values of energy in terms of dollars per thousand acre foot. These marginal values of energy are stochastic parameters, and are typically larger in drier natural inflow scenarios. Other water transport arcs and interstage arcs have objective function coefficients of zero. In general, the random cost and inflow parameters exhibit interstage dependence. For instance, relatively large inflows at lower elevations in winter months are often coupled with a growing snowpack at higher elevations which will, in turn, lead to large inflows when melting occurs in the spring.

The stochastic linear program described above is a subproblem of a larger multistage stochastic *nonlinear* program. In particular, hydro scheduling decisions must be made in a number of basins. Each basin contains interconnected reservoirs, but the different basins are not linked by hydrological constraints. The only constraints linking the different basins are the load constraints; these require that at each possible point in time, demand for energy be satisfied. The Dantzig-Wolfe decomposition principle may be applied to create a nonlinear master problem that contains proposed hydro solutions, the demand constraints, and the nonlinear thermal cost function. The Dantzig-Wolfe subproblem separates into a sum of independent sub-subproblems by hydro basin. Each basin sub-subproblem is then a SLP-T of the form described in the previous paragraph. See Jacobs et al. [39] for a more detailed discussion of this coordination model and for a more detailed description of the hydroelectric scheduling model. We refer the reader to Eiselt, Pederzoli, and Sandbloom [22] for a discussion of nonlinear Dantzig-Wolfe decomposition.

1.3.2 Portfolio Management

In this section we consider a portfolio management model; the formulation is primarily based on Mulvey and Vladimirov [49] [50]. The problem involves dynamically selecting portfolios over time from $N + 1$ assets with uncertain returns and with uncertain transaction costs. One of the assets is a risk-free asset, called cash. At the beginning of the first stage, the decision maker is endowed with an initial portfolio, and we assume this portfolio consists of some nonnegative quantity of each asset. In each stage, the decision maker may rearrange the portfolio subject to transaction costs. The transaction costs may be stochastic parameters; they are known with certainty in the current stage but not necessarily in future stages (e.g., foreign exchange rates). Figure 1.2 depicts the generalized network used to model the portfolio management problem for a four stage example. Generalized networks are networks with gains and losses; for example, the decision maker might exchange one unit of cash for two units of asset 1. In the network, this corresponds to sending a unit of flow along the appropriate vertical arc. The arc multipliers in these generalized networks are also stochastic parameters; they are known to the decision maker only in the current stage. The horizontal arcs in Figure 1.2 represent uncertain returns on the respective assets, and stochastic exogenous deposits and withdrawals may appear as supplies and demands on nodes within the network.

Figure 1.2: Portfolio management model

Within the framework of the SLP-T notation, A_t^{ac} is the node-arc incidence matrix for the single stage generalized network containing $N + 1$ nodes and $2N$ arcs; we assume that the only feasible transactions involve converting cash into an asset and an asset into cash. As the value of the assets

in the future, relative to cash, is unknown, this matrix contains stochastic elements for $t \geq 2$. For a particular scenario, the only arcs with nonzero objective function coefficients are transaction arcs and the final portfolio value arc. The transaction costs are incurred on flows in the vertical arcs shown in Figure 1.2. Transaction costs are uncertain for $t \geq 2$; thus, $c_t^{\omega_t}$ is stochastic. Uncertain returns are modeled as gains or losses incurred on the horizontal interstage arcs contained in $B_{t-1}^{\omega_t}$. The initial endowment vector is b_1 and subsequent $b_t^{\omega_t}$'s contain the stochastic exogenous deposits and withdrawals. At the end of the final stage, the value of the portfolio is calculated; this involves converting the assets to a common unit, say cash. Of course, this conversion need not actually occur in practice. In our simplified model, the goal is to maximize the expected value of the final portfolio less transaction costs incurred along the way. In general, it may be necessary to consider nonlinear objective functions and/or additional constraints to properly model a decision maker's risk preference.

1.3.3 Forest Harvest Planning

Gassmann [32] describes a stochastic programming model arising in forestry management that prescribes logging levels designed to maximize the expected volume of harvested wood over several 20-year stages. The stochastic element of the forest harvest planning problem involves potential destruction of standing timber by forest fires or other environmental hazards.

The forest is assumed to contain only one tree species belonging to K different age-classes. The forest is further assumed to be homogeneous with respect to logging cost. Transitions between age classes occur as follows. The trees in class $k < K$ that are not harvested or destroyed move to class $k + 1$ in the next time stage, and an area of trees that are destroyed or harvested moves to age class 1. Class K is an absorbing class for all trees exceeding a particular age ($20 \cdot K$ years), and class 1 contains trees less than 20 years old. We assume that cleared areas are reforested in the next stage with class 1 trees. At the beginning of the first stage, the forest is described by a K -vector, $s_1 \geq 0$, where the k^{th} component of s_1 denotes the area of the forest covered with class k trees. In the first stage, a decision is made to harvest some nonnegative area of each class; this decision vector is represented by x_1 ; clearly this decision must satisfy $x_1 \leq s_1$. Next a random event occurs; i.e., some proportion of the standing timber is destroyed. We denote the random proportion destroyed for age

class k by $q_k^{\omega_2}$. A particular second stage outcome, $(q_1^{\omega_2}, \dots, q_K^{\omega_2})$, occurs with known probability $p_2^{\omega_2}$. Given this event, the state of the forest at the beginning of the second stage is expressed:

$$s_2^{\omega_2} = Q^{\omega_2}(s_1 - x_1) + H x_1 \quad (1.1)$$

where

$$Q^{\omega_2} = \begin{bmatrix} q_1^{\omega_2} & q_2^{\omega_2} & \cdots & q_{K-1}^{\omega_2} & q_K^{\omega_2} \\ 1 - q_1^{\omega_2} & 0 & \cdots & 0 & 0 \\ 0 & 1 - q_2^{\omega_2} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 - q_{K-1}^{\omega_2} & 1 - q_K^{\omega_2} \end{bmatrix}$$

and

$$H = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Thus, within the SLP-T framework, stage t has two types of constraints. The first constraint type determines the inventory of standing timber; for example, (1.1) gives the stage 2 inventory equation. The second constraint type states that we cannot harvest more than the standing timber: $x_t^{\omega_2} - s_t^{\omega_2} \leq 0$. We assume that the per unit wood volume yield of harvesting a particular age class is known; thus, the objective function coefficients are deterministic. We further assume that the area of the forest we are modeling does not change, thus $b_t^{\omega_t} \equiv 0$ for $t \geq 2$. Therefore, in terms of the SLP-T notation, the stochastic parameters appear only in $B_{t-1}^{\omega_t}$; the coefficients in this matrix determine what fraction of the timber, standing after a harvest, survives to the next time stage.

1.4 Benders Decomposition

A number of algorithmic approaches have been proposed for the two-stage stochastic linear program with recourse (SLP-2). Van Slyke and Wets [61] described how to apply Benders decomposition algorithm to SLP-2 with the “L-shaped Algorithm.” Kall [40] and Strazicky [59] developed algorithms that take advantage of special basis structures of the “dual block angular” system of the two-stage program. Stochastic quasi-gradient methods, in which random search directions are sequentially selected, are another class of algorithms that have been studied; see, for example, Ermoliev [23]

and Gaivoronski [29]. A classical approximation scheme for the special case of SLP-2 in which random parameters appear only in the right-hand-side is to calculate lower and upper bounds via the inequalities of Jensen and Edmundson–Madansky, respectively. Kall [41], Frauendorfer [28], and Birge and Wets [11] [12] have developed refinements, extensions, and alternatives to this basic approach. Rockafellar and Wets [57] proposed a “Progressive Hedging” algorithm for SLP-T in which nonanticipativity constraints are enforced via Lagrangian penalty terms in the objective, and at each iteration, one multistage subproblem is solved for each scenario. Lustig et al. [48] have empirically studied general linear program solvers’ performance on different formulations of the two-stage problem. King and Wets [43] have applied the theory of epi-consistency to stochastic programming to verify consistency results for sequences of optimal solutions. Higle and Sen [34] have developed a “Stochastic Decomposition” algorithm rooted in Benders method which asymptotically creates an outer linearization of the second stage objective function and, like the stochastic quasi-gradient algorithms, only requires one random sample per iteration. Dantzig and Glynn [16], Infanger [36], and Pereira and Pinto [51] have developed modified Benders decomposition algorithms, coupled with Monte Carlo sampling techniques, for both the two-stage and multistage problems. In this development, we focus on variants of Benders decomposition as our main algorithmic tool.

Benders decomposition algorithm was originally developed for mixed integer programming [4]; in the continuous case, it is the dual of the primal decomposition approach developed by Dantzig and Wolfe [19]. Van Slyke and Wets [61] first described how to apply Benders decomposition to the two stage problem with the “L-shaped Algorithm.” Earlier, Dantzig and Madansky [18] showed that the dual of the two stage stochastic problem had a structure ideal for the application of primal (Dantzig–Wolfe) decomposition. Birge [7] [9] extended the L-shaped algorithm to the multistage case. We will begin by describing Benders method in the two-stage environment and then discuss the extension to the multistage case.

1.4.1 Two-Stage Deterministic Linear Programs

For simplicity, we begin with the special case of SLP-2 in which $|\Omega_2| = |\Omega| = 1$. Thus the problem of interest is

$$\begin{aligned}
& \text{minimize} && c_1 x_1 & + & c_2 x_2 \\
& \text{subject to} && A_1 x_1 & & = b_1 \\
& && -B_1 x_1 & + & A_2 x_2 = b_2 \\
& && x_1 \geq 0, & & x_2 \geq 0.
\end{aligned} \tag{1.2}$$

We may rewrite (1.2) as:

$$\begin{aligned}
& \text{minimize} && \left\{ c_1 x_1 + \begin{array}{l} \text{minimize} \\ A_2 x_2 = b_2 + B_1 x_1 \\ x_2 \geq 0 \end{array} [c_2 x_2] \right\}, \\
& \begin{array}{l} A_1 x_1 = b_1 \\ x_1 \geq 0 \\ x_1 \in \mathcal{K} \end{array}
\end{aligned} \tag{1.3}$$

where $\mathcal{K} = \{x_1 : \exists x_2 \text{ such that } A_2 x_2 = b_2 + B_1 x_1, x_2 \geq 0\}$. In some two-stage models, it is possible to make first stage decisions that can cause the second stage to be infeasible. The system, in this case, is said to be *without recourse*. If, however, we are able to recover from every contingency, regardless of the first stage decision, then the system is said to have *complete recourse*. More formally, the assumption is $\text{pos} A_2 \equiv \{y : y = A_2 x_2, x_2 \geq 0\} = \mathbb{R}^{m_2}$. (This assumption is stronger than that of *relatively complete recourse* [65] which only requires second stage feasibility for x_1 satisfying the first stage constraints.) The assumption of complete recourse is reasonable whenever second stage infeasibilities can be modeled via penalty costs. In order to simplify the discussion, we assume our model has this property; this is not a necessary assumption for Benders method (see Van Slyke and Wets [61]). In the present discussion, the assumption of complete recourse implies that $x_1 \in \mathcal{K}$ is no constraint at all.

We further assume that the second stage problem is bounded. Observe that if it were unbounded for some finite first stage decision then it would be unbounded for all finite first stage decisions as the dual feasible region is independent of x_1 . We enumerate the extreme points of $\Pi = \{\pi : \pi A_2 \leq c_2\}$ as $\pi^{(1)}, \dots, \pi^{(L)}$. We may now rewrite (1.3) as

$$\begin{aligned}
& \text{minimize} && \{ c_1 x_1 + h(x_1) \}, \\
& \begin{array}{l} A_1 x_1 = b_1 \\ x_1 \geq 0 \end{array}
\end{aligned} \tag{1.4}$$

where $h(x_1)$ may be expressed:

$$\begin{aligned}
h(x_1) = \min && c_2 x_2 && = \max && \pi(b_2 + B_1 x_1) && = \max && \pi^{(i)}(b_2 + B_1 x_1). \\
& \text{s.t.} && A_2 x_2 = b_2 + B_1 x_1 && \text{s.t.} && \pi A_2 \leq c_2 && 1 \leq i \leq L \\
& && x_2 \geq 0
\end{aligned} \tag{1.5}$$

The recourse function, $h(x_1)$, represents the cost that we will incur if we act optimally in the second stage, as a function of the first stage decision. The recourse function is a piecewise linear, convex function on $\mathcal{X}_1 = \{x_1 : A_1 x_1 = b_1, x_1 \geq 0\}$. Based on (1.4) and (1.5), we may express (1.2) in a form that suggests Benders algorithm:

$$\text{minimize} \quad c_1 x_1 + \theta \quad (1.6.1)$$

$$\text{subject to} \quad A_1 x_1 = b_1 \quad (1.6.2)$$

$$\text{cuts:} \quad -G^i x_1 + \theta \geq g^i \quad i = 1, \dots, L \quad (1.6.3)$$

$$x_1 \geq 0, \quad (1.6.4)$$

where $G^i = \pi^{(i)} B_1$ and $g^i = \pi^{(i)} b_2$.

Benders method decomposes (1.2) into a *master* and a *subproblem*. The master represents the first stage plus a number of necessary conditions, called *cuts*, derived from the second stage. The subproblem represents the second stage and is used to determine $h(x_1)$ for given values of x_1 . The master problem has the form of (1.6), except that at any particular iteration only a subset of the necessary cuts (1.6.3) have been collected so far. In an optimal solution of the master program, denoted $(\hat{x}_1, \hat{\theta})$, the scalar value $\hat{\theta}$ is an estimate of $h(\hat{x}_1)$; if the set of cuts is complete at $\hat{x}_1$ then $\hat{\theta} = h(\hat{x}_1)$ and if the set of cuts is incomplete $\hat{\theta} < h(\hat{x}_1)$. Three different formulations of the subproblem are shown in (1.5).

Benders algorithm is iterative, alternating between solving the master and subproblem. The master's feasible region and the subproblem's right-hand-side continually change as information is passed from one to the other. We may begin the algorithm with an initial cut $\theta \geq -M$ where $-M$ is a lower bound on $h(x_1) \forall x_1 \in \mathcal{X}_1$; cuts are then added one at a time to the master program as the algorithm proceeds. In one iteration of the algorithm, the master problem is solved and the decision $\hat{x}_1$ is passed to the subproblem. The subproblem then acts optimally, given the resources $b_2 + B_1 \hat{x}_1$, and passes marginal value of resource information back to the master in the form of a cut. In this setting, the cuts are supports of the convex piecewise linear recourse function; in particular, each cut is one of the L pieces of $h(x_1)$ (see (1.5)). As the set of cuts in the master program is augmented each iteration, "more informed" first stage decisions are made. The algorithm terminates when the

master has sufficient information regarding the recourse function in the neighborhood of an optimal first stage decision.

At any given iteration, an *upper bound* on the optimal objective function value is given by $c_1 \hat{x}_1 + c_2 \hat{x}_2$ as we have a feasible first stage decision $\hat{x}_1$ and a feasible second stage decision $\hat{x}_2$ (via the assumption of complete recourse). We can also obtain a *lower bound* on the optimal objective function value by evaluating $c_1 \hat{x}_1 + \hat{\theta}$ as the cuts collected so far provide an outer linearization of the recourse function; in particular, we have collected a number of necessary conditions, but the set of cuts may not be sufficient yet. Clearly, the lower bounds generated in successive iterations are increasing; at each iteration, the master program becomes more constrained. On the other hand, the upper bounds generated by the master program solutions need not be decreasing. Thus, in practice, we save the solution that corresponds to the least upper bound generated so far. If the upper and lower bounds on the optimal objective function at some iteration are equal then the algorithm terminates. If there is a gap in these bounds the subproblem must pass back a new cut to the master problem. The algorithm has this property; for otherwise, the master would have had sufficient information regarding $h(x_1)$ at $\hat{x}_1$. There are a finite number of possible cuts because the dual feasible region has a finite number of extreme points. Hence, if each linear program is solved in a finite number of iterations, Benders algorithm will terminate in a finite number of steps with an optimal solution. Benders algorithm for (1.2) is summarized in Figure 1.3.

- step 0** define $toler \geq 0$;
 initialize the set of cuts with $\theta \geq -M$;
 let $\bar{z} = +\infty$
- step 1** solve the master problem and obtain $(\hat{x}_1, \hat{\theta})$;
 let $\underline{z} = c_1 \hat{x}_1 + \hat{\theta}$
- step 2** form the right-hand-side of the subproblem: $B_1 \hat{x}_1 + b_2$;
 solve the subproblem and obtain $(\hat{x}_2, \hat{\pi})$;
 let $\hat{z} = c_1 \hat{x}_1 + c_2 \hat{x}_2$;
 if $\hat{z} < \bar{z}$ then let $\bar{z} = \hat{z}$ and $(x_1^*, x_2^*) = (\hat{x}_1, \hat{x}_2)$
- step 3** if $\bar{z} - \underline{z} \leq \min(|\bar{z}|, |\underline{z}|) \cdot toler$ then stop: (x_1^*, x_2^*) is a solution
 with objective function value within $(100 \cdot toler)\%$ of the
 optimal objective function value.
- step 4** augment the set of cuts with $\theta - \hat{\pi} B_1 x_1 \geq \hat{\pi} b_2$;
 goto step 1

Figure 1.3: Benders Decomposition Algorithm for SLP-2 with $|\Omega| = 1$

1.4.2 Two-Stage Stochastic Linear Programs

SLP-2 may be expressed:

$$\begin{aligned}
 & \text{minimize} && c_1 x_1 + p^1 c_2^1 x_2^1 + \dots + p^K c_2^K x_2^K \\
 & \text{subject to} && A_1 x_1 = b_1 \\
 & && -B_1^1 x_1 + A_2^1 x_2^1 = b_2^1 \\
 & && \vdots \\
 & && -B_1^K x_1 + A_2^K x_2^K = b_2^K \\
 & && x_1 \geq 0, \quad x_2^1 \geq 0, \quad \dots \quad x_2^K \geq 0
 \end{aligned} \tag{1.7}$$

where $\Omega = \{1, 2, \dots, K\}$. A particular second stage realization is denoted $\xi(k) = (c_2^k, b_2^k, B_1^k, A_2^k)$, and $P\{\xi = \xi(k)\} = p^k$ is its associated probability. From a formal viewpoint, (1.7) may be considered a special case of (1.2) in which:

$$\begin{aligned}
 c_2 &= [p^1 c_2^1, \dots, p^K c_2^K], \\
 b_2 &= \begin{bmatrix} b_2^1 \\ \vdots \\ b_2^K \end{bmatrix}, \quad x_2 = \begin{bmatrix} x_2^1 \\ \vdots \\ x_2^K \end{bmatrix}, \\
 A_2 &= \begin{bmatrix} A_2^1 & 0 \\ & \ddots \\ 0 & A_2^K \end{bmatrix}, \quad \text{and} \quad B_1 = \begin{bmatrix} B_1^1 \\ \vdots \\ B_1^K \end{bmatrix}.
 \end{aligned}$$

As indicated in the Introduction, in practice K is often enormous in size; therefore from a computational standpoint, a *key observation* is that the second stage subproblem is separable into one sub-subproblem for each outcome $\omega \in \Omega$. Thus, we may solve a very large subproblem by solving K smaller subproblems. We develop the notation for SLP-2 to reflect this computational approach. In particular we write (1.7) as:

$$\begin{aligned}
 & \text{minimize} && \{ c_1 x_1 + E_\xi h(x_1, \xi(\omega)) \}, \\
 & A_1 x_1 = b_1 \\
 & x_1 \geq 0
 \end{aligned} \tag{1.8}$$

where E_ξ denotes the expected value operator with respect to ξ , and $E_\xi h(x_1, \xi(\omega))$ denotes the recourse function. We will typically abbreviate E_ξ and $h(x_1, \xi(\omega))$ by E_ω and $h(x_1, \omega)$, respectively.

The second stage cost we will incur, if we observe outcome ω and act optimally given first stage decision x_1 , is $h(x_1, \omega)$ and may be expressed:

$$\begin{aligned} h(x_1, \omega) = & \text{minimize} && c_2^\omega x_2^\omega \\ & \text{subject to} && A_2^\omega x_2^\omega = b_2^\omega + B_1^\omega x_1 \quad : \pi^\omega \\ & && x_2^\omega \geq 0 \end{aligned} \quad (1.9)$$

where π^ω denotes the corresponding dual multipliers.

Following the development of Section 1.4.1, each $h(x_1, \omega)$ is a piecewise linear, convex function on $\mathcal{X}_1$. The recourse function, $E_\omega h(x_1, \omega)$, is a convex combination of K functions, $h(x_1, \omega)$, $\omega \in \Omega$, and hence, it is also a piecewise linear, convex function on $\mathcal{X}_1$. The cut gradient and intercept formulas are given by:

$$G^i = E_\omega \pi^\omega B_1^\omega \text{ and } g^i = E_\omega \pi^\omega b_2^\omega$$

where π^ω denotes an optimal dual vector obtained from solving subproblem(ω), shown in (1.9), with a specific choice of the first stage decision $x_1 = \hat{x}_1$; from an algorithmic standpoint, it is appropriate to consider $\pi^\omega = \pi^\omega(\hat{x}_1)$.

Assumptions regarding the model, and properties of the algorithm, discussed in Section 1.4.1 carry over to the present model and implementation of Benders algorithm. In particular, the notion of complete recourse, the fact that cuts are supports of the recourse function, computations for upper and lower bounds on the optimal objective function value, and the finite termination argument all carry over directly to Benders algorithm applied to SLP-2. We will, however, make a distinction between the two models with respect to what we call a “solution.” In the stochastic model, we are primarily interested in the decision that should be made in the first stage; thus, a solution to SLP-2 will consist of an optimal first stage decision, x_1^* , and its associated expected cost. The algorithm we present here also obtains optimal second stage decisions for all possible outcomes, so this distinction may seem unnecessary, but the reason for this definition of a solution will become clear when we later discuss modified Benders algorithms that use Monte Carlo sampling techniques. Further justification for this definition of a solution is derived from the fact that in practice, many time-staged applications involve “rolling horizons,” and a new model may be formulated and solved

once we have arrived in the second stage. Benders decomposition algorithm for (1.7) is summarized in Figure 1.4.

```

step 0  define  $toler \geq 0$ ;
         initialize the set of cuts with  $\theta \geq -M$ ;
         let  $\bar{z} = +\infty$ 

step 1  solve the master problem and obtain  $(\hat{x}_1, \hat{\theta})$ ;
         let  $\underline{z} = c_1 \hat{x}_1 + \hat{\theta}$ 

step 2  do  $\omega = 1$  to  $K$ 
         form the right-hand-side of subproblem( $\omega$ ):  $B_1^\omega \hat{x}_1 + b_2^\omega$ ;
         solve the subproblem and obtain  $(\hat{x}_2^\omega, \hat{\pi}^\omega)$ ;
         enddo

         let  $\hat{z} = c_1 \hat{x}_1 + \sum_{\omega=1}^K p^\omega c_2^\omega \hat{x}_2^\omega$ ;
         if  $\hat{z} < \bar{z}$  then let  $\bar{z} = \hat{z}$  and  $x_1^* = \hat{x}_1$ 

step 3  if  $\bar{z} - \underline{z} \leq \min(|\bar{z}|, |\underline{z}|) \cdot toler$  then stop:  $x_1^*$  is a solution
         with objective function value within  $(100 \cdot toler)\%$  of the
         optimal objective function value.

step 4  augment the set of cuts with  $\theta - \sum_{\omega=1}^K p^\omega \hat{\pi}^\omega B_1^\omega x_1 \geq \sum_{\omega=1}^K p^\omega \hat{\pi}^\omega b_2^\omega$ ;
         goto step 1

```

Figure 1.4: Benders Decomposition Algorithm for SLP-2

1.4.3 Multistage Stochastic Linear Programs

The extension of Benders decomposition algorithm from SLP-2 to SLP-T can be viewed as follows. SLP-2 is a linear program that we solve by decomposing it into two stages; thus, we first decompose SLP-T into two stages: stage 1 and stage 2, $\dots$, T . After “master 1” makes decision x_1 and the corresponding resources, $B_1^{\omega_2} x_1 + b_2^{\omega_2}$, are placed on the right-hand sides of the stage 2 subproblems, we have $|\Omega_2|$ linear programs to solve. We solve each of these linear programs, in turn, by decomposing them into two stages: stage 2 and stage 3, $\dots$, T . There are $|\Omega_2|$ second stage master problems, and each makes a decision $x_2^{\omega_2}$ and passes the corresponding resources, $B_2^{\omega_3} x_2^{\omega_2} + b_3^{\omega_3}$, $\omega_3 \in \Delta(\omega_2)$, to

its third stage descendants ... This recursive method is the traditional nested Benders decomposition approach to multistage stochastic linear programs (see Birge [7]). We provide a more formal description of this algorithm in Chapter 2.

We now take up an implementation issue with respect to a special case of SLP-T in which the stochastic parameters exhibit interstage independence. In addition to examining an important special case, this discussion will enable us to further illustrate the scenario indexing notation for the multistage problem introduced in Section 1.2. The stage t recourse function for scenario subproblem(ω_t) may be expressed:

$$E_{\omega_{t+1}|\omega_t} h_t(x_t, \omega_{t+1}) = \sum_{\omega_{t+1} \in \Delta(\omega_t)} p_{t+1}^{\omega_{t+1}|\omega_t} h_t(x_t, \omega_{t+1}). \quad (1.10)$$

The cost we will incur in stage $t + 1$ if we act optimally under scenario ω_{t+1} , given decision x_t , is

$$\begin{aligned} h_t(x_t, \omega_{t+1}) = & \text{minimize} \quad c_{t+1}^{\omega_{t+1}} x_{t+1}^{\omega_{t+1}} + E_{\omega_{t+2}|\omega_{t+1}} h_{t+1}(x_{t+1}^{\omega_{t+1}}, \omega_{t+2}) \\ & \text{subject to} \quad A_{t+1}^{\omega_{t+1}} x_{t+1}^{\omega_{t+1}} = b_{t+1}^{\omega_{t+1}} + B_t^{\omega_{t+1}} x_t \\ & \quad \quad \quad x_{t+1}^{\omega_{t+1}} \geq 0. \end{aligned} \quad (1.11)$$

The conditional probability mass function, $p_{t+1}^{\omega_{t+1}|\omega_t}$, is defined as follows:

$$p_{t+1}^{\omega_{t+1}|\omega_t} = \frac{p_{t+1}^{\omega_{t+1}}}{\sum_{\omega_{t+1} \in \Delta(\omega_t)} p_{t+1}^{\omega_{t+1}}}.$$

As we are assuming interstage independence of the stochastic parameters the stage $t + 1$ probabilities and realizations do not depend on ω_t (see example in Section 1.2). Thus, it is clear that the recourse functions for the next-to-last stage $T - 1$ scenario subproblems are identical in functional form as $h_T \equiv 0$. Based on (1.11), inductive application of this observation shows that the stage t recourse functions for each of the scenario subproblems are identical in functional form. Note, however, that the decision variables, in general, do not exhibit interstage independence. Restated, the set

$$\mathcal{X}_t^{\omega_t} = \left\{ x_t^{\omega_t} : A_t^{\omega_t} x_t^{\omega_t} = b_t^{\omega_t} + B_{t-1}^{\omega_t} x_{t-1}^{a(\omega_t)}, x_t^{\omega_t} \geq 0, x_{t-1}^{a(\omega_t)} \in \mathcal{X}_{t-1}^{a(\omega_t)} \right\}$$

depends on ω_t , but

$$E_{\omega_{t+1}|\omega_t} h_t(x_t, \omega_{t+1}|\omega_t) = E_{\omega_{t+1}|\omega_t'} h_t(x_t, \omega_{t+1}|\omega_t') \quad \forall \omega_t, \omega_t' \in \Omega_t. \quad (1.12)$$

Recall that by the assumption of complete recourse, the stage $t + 1$ subproblems are feasible for any stage t decision x_t . Thus, while in general $\mathcal{X}_t^{\omega_t} \neq \mathcal{X}_t^{\omega'_t}$, the two functions in (1.12) are well defined at all $x_t \in \mathbb{R}^{n_t}$. This is an important observation from an implementation viewpoint. Instead of storing a separate set of cuts for each stage t scenario we can simply store all the stage t cuts in one location. When we solve any stage t problem, we use all the cut constraints collected so far on stage t . We are assured that any cut produced for a particular stage t scenario is valid for all stage t scenarios. This is important not only because it yields a potentially dramatic savings in the memory requirement, but also because it may accelerate convergence of the algorithm. As the algorithm proceeds, a particular scenario not only gains information regarding the recourse function from its descendants, but also from the descendants of its neighbors. The ability to “share” cuts in this, or some other, fashion is a critical component of Benders decomposition-based algorithms that incorporate Monte Carlo sampling techniques for multistage problems. We return to the topic of sharing cuts in Section 3.5 in which we consider certain types of interstage dependence of the stochastic parameters.

1.5 Value of the Stochastic Solution

In this section we illustrate the value of the stochastic solution through a very simple example. We consider a two-stage hydroelectric scheduling model with one reservoir and a planning horizon of two months from the present time. The natural inflow in the first month is assumed to be known (and for simplicity taken to be zero), while the natural inflow in the second month, denoted ξ , is modeled as a random variable distributed uniformly on the interval $(0, b)$. The model has the following form:

$$\begin{aligned} & \text{maximize} && c_1 x_1 + Eh(x_1, \xi) \\ & \text{subject to} && 0 \leq x_1 \leq b \\ & \text{where} && h(x_1, \xi) = \begin{aligned} & \text{maximize} && c_2 x_2 - \rho y \\ & \text{subject to} && x_2 - y = \xi - x_1 \\ &&& x_2, y \geq 0 \end{aligned} \end{aligned}$$

and where $\xi \sim U(0, b)$.

The scalar decision variables x_1 and x_2 denote the volume of water we pass through the reservoir’s powerhouse to generate energy in month 1 and month 2, respectively. The objective function

coefficients c_1 of x_1 and c_2 of x_2 are positive and represent the marginal value of energy (\$/MWh) in each month. In general, c_1 and c_2 are different due to seasonal variations in the demand for energy. We will assume one unit of water generates one unit of energy and speak of the two interchangeably. The total amount of water used for energy generation during the two month planning period should not exceed the total natural inflow represented by ξ . Generation levels may exceed natural inflow by drawing on the reservoir's reserve water supply, but, in this case, the system is penalized at rate $\rho > \max\{c_1, c_2\}$. Thus, we must make a first-stage energy generation decision based solely upon knowledge of the probability distribution of the total volume of water available for the entire planning period. After generating x_1 and observing a natural inflow realization, ξ , there is $(\xi - x_1)^+ \equiv \max\{\xi - x_1, 0\}$ available for generation in the second month. This stochastic program with simple recourse may be expressed:

$$\begin{aligned} \text{maximize} \quad & f(x_1) = c_1 x_1 + b^{-1} \int_0^b [c_2(\xi - x_1)^+ - \rho(x_1 - \xi)^+] d\xi \\ \text{subject to} \quad & 0 \leq x_1 \leq b \end{aligned}$$

Evaluating the integral in the objective function yields

$$f(x_1) = c_1 x_1 + c_2 (b - x_1)^2 / (2b) - \rho x_1^2 / (2b)$$

The optimal solution to this one-variable quadratic program is

$$x_1^* = b \frac{(c_1 - c_2)^+}{\rho - c_2}. \quad (1.13)$$

If c_2 , the marginal value of energy in the second month, is larger than c_1 , then the "obvious" solution is to wait and generate all of the natural inflow in month 2. The "interesting" case is $c_1 > c_2$; i.e., it is financially more attractive to generate energy in month 1. In this case we generate a positive quantity given by (1.13) and take the risk that we will not observe sufficient inflow in the second month to cover the first month's level of generation. From (1.13) we see that the fraction of total possible inflow that we are willing to risk in the first month increases as the marginal gain in the first stage grows, and decreases as the penalty for over-generation, ρ , grows. Now consider the *expected value problem* in which we replace the random second month natural inflow with the mean $b/2$. When $c_2 < c_1$ the expected value solution is identical to the stochastic solution (i.e., $x_1 = 0$), and the value of the stochastic solution is zero. However, in the interesting case when $c_1 > c_2$, the

expected value solution is $x_1 = b/2$ because we "know" that we will receive a second stage natural inflow of $b/2$ so we generate all of it in the financially more attractive first month. Substitution of this decision into the stochastic objective function yields

$$f(b/2) = b[c_1/2 + c_2/8 - \rho/8].$$

If x_1^* denotes the optimal solution to the stochastic problem then it is clear the value of the stochastic solution (i.e., $f(x^*) - f(b/2)$) grows without bound as $\rho \rightarrow \infty$ since $f(x^*) \geq f(0) = 0$. To consider a specific example, let $c_1 = 2c_2$ and $\rho = 5c_2$. Then $x_1^* = b/4$ and $f(b/4) = c_2b(5/8)$ while $f(b/2) = c_2b(1/2)$. In this case, the stochastic formulation yields a solution, $x_1^* = b/4$, whose objective function value is a 25% *improvement* over that of the solution, $x_1 = b/2$, obtained by solving the deterministic problem in which the uncertain natural inflow is replaced by its mean.

Chapter 2

An Enhanced Decomposition Algorithm for SLP-T

2.1 Introduction

As indicated in Chapter 1, Benders decomposition is a tool which may be used to solve multistage stochastic linear programs with recourse (SLP-T). Many practical models state simple upper bounds on decision variables explicitly; SLP-T for such models has the following form:

$$\begin{aligned}
 & \text{minimize} && \sum_{t=1}^T \sum_{\omega_t \in \Omega_t} p_t^{\omega_t} c_t^{\omega_t} x_t^{\omega_t} \\
 & \text{subject to} && -B_{t-1}^{\omega_t} x_{t-1}^{a(\omega_t)} + A_t^{\omega_t} x_t^{\omega_t} = b_t^{\omega_t}, \quad 0 \leq x_t^{\omega_t} \leq u_t^{\omega_t}, \quad \omega_t \in \Omega_t \\
 & \text{for} && t = 1, \dots, T \\
 & \text{where} && B_0^{\omega_1} \equiv 0.
 \end{aligned}
 \tag{SLP-T}$$

Recall, $|\Omega_1| = 1$; notational convenience is the only reason for having a first stage sample space. The definition of the random vectors ξ_t , $t = 1, \dots, T$, can be naturally extended to incorporate simple upper bounds. A particular stage t realization, $\xi_t(\omega_t) = (c_t^{\omega_t}, u_t^{\omega_t}, b_t^{\omega_t}, B_{t-1}^{\omega_t}, A_t^{\omega_t})$, is to be read column-wise as a vector in $\Re^{N_t}$, where $N_t = 2 \cdot n_t + m_t + m_t \cdot n_{t-1} + m_t \cdot n_t$. Recall, $A_t^{\omega_t}$ is an $m_t \times n_t$ matrix, and the remaining matrices and vectors are dimensioned to conform. In an attempt to avoid cumbersome notation, we will only deal with simple upper bounds explicitly when necessary. When simple upper bounds are not displayed explicitly the reader may assume they have been included in the structural constraints.

In this chapter, several modifications to the traditional nested Benders algorithm are presented. We have found by means of empirical tests, that certain of these modifications can be combined to

produce a computationally attractive algorithm. A collection of large-scale hydroelectric scheduling problems was used for test problems. Since four different enhancements will be presented, a brief overview is in order:

- (i) *Warm start* techniques: these are used to obtain “good” initial basic feasible solutions for subproblems based on optimal basis information from previous subproblem solutions.
- (ii) *Advanced start* procedures: these are used to generate preliminary cuts prior to initiating a formal Benders algorithm.
- (iii) The *multicut Benders* algorithm: this is used to return one cut for each descendant of a particular subproblem instead of a single aggregate cut.
- (iv) *Tree traversing strategies*: these are used to prescribe the order in which subproblems of the decision tree are solved.

The scope of this chapter is limited to variants of Benders algorithm which solve SLP-T “directly” within a prespecified tolerance. Algorithms that incorporate Monte Carlo sampling techniques will be discussed in subsequent chapters.

2.2 Valid Cut Generation

Prior to presenting the enhancements to the traditional nested Benders algorithm, we prove some lemmas concerning valid cut generation in this section. A *valid* cut is defined to be a cut which lies below the recourse function. In Lemmas 2.1 and 2.2, we show that a valid cut is an inequality generated by linear combinations of the primal constraints using as weights a feasible dual solution. We first formalize this notion in Lemma 2.1 for the two-stage case where the proof is particularly transparent.

Lemma 2.1 *Consider SLP-2 with $|\Omega| = K$. If $(\pi^1, \dots, \pi^K)$ is a collection of dual feasible vectors for the second stage subproblems then these dual vectors can be used to generate a valid cut for the first stage master program.*

Proof

SLP-2 may be written:

$$\begin{array}{ll} \text{minimize} & \left\{ c_1 x_1 + \sum_{\omega \in \Omega} p^\omega h(x_1, \omega) \right\}, \\ \text{subject to} & A_1 x_1 = b_1 \\ & x_1 \geq 0 \end{array}$$

where

$$h(x_1, \omega) = \begin{array}{ll} \text{maximize} & \pi^\omega (b_2^\omega + B_1^\omega x_1) \\ \text{subject to} & \pi^\omega A_2^\omega \leq c_2^\omega. \end{array}$$

Let $\Pi^\omega = \{\pi : \pi A_2^\omega \leq c_2^\omega\}$ be the set of feasible dual solutions. By hypothesis, $\hat{\pi}^\omega \in \Pi^\omega \forall \omega \in \Omega$.

Let $\beta(x_1, \omega) = \hat{\pi}^\omega (b_2^\omega + B_1^\omega x_1)$. Then $\beta(x_1, \omega) \leq h(x_1, \omega) \forall x_1$ since

$$\hat{\pi}^\omega (b_2^\omega + B_1^\omega x_1) \leq \max_{\pi^\omega \in \Pi^\omega} \pi^\omega (b_2^\omega + B_1^\omega x_1) \quad \forall x_1.$$

Thus

$$E_\omega \beta(x_1, \omega) = \left\{ \sum_{\omega \in \Omega} p^\omega \hat{\pi}^\omega b_2^\omega \right\} + \left\{ \sum_{\omega \in \Omega} p^\omega \hat{\pi}^\omega B_1^\omega \right\} x_1 \leq E_\omega h(x_1, \omega) \quad \forall x_1;$$

i.e., $\theta \geq Gx_1 + g$ is a valid cut where $G = \{\sum_{\omega \in \Omega} p^\omega \hat{\pi}^\omega B_1^\omega\}$ and $g = \{\sum_{\omega \in \Omega} p^\omega \hat{\pi}^\omega b_2^\omega\}$. ■

We now wish to generalize the idea that “feasible dual variables generate valid cuts” to the multistage setting. With the exception of the final stage, the primary difference is that subproblems generating cuts for their ancestors contain their own cuts. We may apply Lemma 2.1 to verify dual feasible vectors to the stage T subproblems will generate valid cuts for their stage $T - 1$ ancestors, but a new result is needed to cover the general case. Lemma 2.2 states: If the stage t ($2 \leq t < T$) subproblems contain valid cuts then they will, given dual feasible vectors, generate valid cuts for their stage $t - 1$ ancestors.

The stage t ($1 \leq t < T$) subproblem under scenario ω_t , denoted $\text{sub}(\omega_t)$, has the following form:

$$\begin{array}{ll} \text{minimize} & c_t^{\omega_t} x_t^{\omega_t} + \theta_t^{\omega_t} \\ \text{subject to} & A_t^{\omega_t} x_t^{\omega_t} = b_t^{\omega_t} + B_{t-1}^{\omega_t} x_{t-1}^{a(\omega_t)} \\ & -G_t^{\omega_t} x_t^{\omega_t} + e \theta_t^{\omega_t} \geq g_t^{\omega_t} \\ & x_t^{\omega_t} \geq 0 \end{array}$$

The rows of the matrix $G_t^{\omega_t}$ contain cut gradients; the elements of the vector $g_t^{\omega_t}$ are cut intercepts; and e denotes the vector of all 1's. As the algorithm proceeds, the row dimensions of these quantities will grow.

Lemma 2.2 Consider SLP- T and in particular, $\text{sub}(\omega_t)$ for some t satisfying $1 \leq t \leq T-2$. Suppose $|\Delta(\omega_t)| = K$ and the descendants of $\text{sub}(\omega_t)$ contain valid cuts. If $[(\hat{\pi}_{t+1}^1, \hat{\alpha}_{t+1}^1), \dots, (\hat{\pi}_{t+1}^K, \hat{\alpha}_{t+1}^K)]$ denotes a collection of dual feasible vectors for the descendants of $\text{sub}(\omega_t)$ then these dual vectors can be used to generate a valid cut for $\text{sub}(\omega_t)$.

Proof

Let

$$\begin{aligned} f_t(x_t, \omega_{t+1}) = \quad & \text{minimize} \quad c_{t+1}^{\omega_{t+1}} x_{t+1}^{\omega_{t+1}} + \theta_{t+1}^{\omega_{t+1}} \\ & \text{subject to} \quad A_{t+1}^{\omega_{t+1}} x_{t+1}^{\omega_{t+1}} = b_{t+1}^{\omega_{t+1}} + B_t^{\omega_{t+1}} x_t \\ & \quad -G_{t+1}^{\omega_{t+1}} x_{t+1}^{\omega_{t+1}} + e \theta_{t+1}^{\omega_{t+1}} \geq g_{t+1}^{\omega_{t+1}} \\ & \quad x_{t+1}^{\omega_{t+1}} \geq 0, \end{aligned} \quad (2.1)$$

where the cuts in (2.1) denote a set of valid cuts. For each $\omega_{t+1} \in \Delta(\omega_t)$ there exists a finite set of cuts, denoted by gradient matrix $\tilde{G}_{t+1}^{\omega_{t+1}}$ and intercept vector $\tilde{g}_{t+1}^{\omega_{t+1}}$, such that

$$\begin{aligned} h_t(x_t, \omega_{t+1}) = \quad & \text{minimize} \quad c_{t+1}^{\omega_{t+1}} x_{t+1}^{\omega_{t+1}} + \theta_{t+1}^{\omega_{t+1}} \\ & \text{subject to} \quad A_{t+1}^{\omega_{t+1}} x_{t+1}^{\omega_{t+1}} = b_{t+1}^{\omega_{t+1}} + B_t^{\omega_{t+1}} x_t \\ & \quad -\tilde{G}_{t+1}^{\omega_{t+1}} x_{t+1}^{\omega_{t+1}} + e \theta_{t+1}^{\omega_{t+1}} \geq \tilde{g}_{t+1}^{\omega_{t+1}} \\ & \quad x_{t+1}^{\omega_{t+1}} \geq 0. \end{aligned} \quad (2.2)$$

Let $(\hat{\pi}_{t+1}^{\omega_{t+1}}, \hat{\alpha}_{t+1}^{\omega_{t+1}})$ be dual feasible for (2.1). Then

$$\hat{\pi}_{t+1}^{\omega_{t+1}} (b_{t+1}^{\omega_{t+1}} + B_t^{\omega_{t+1}} x_t) + \hat{\alpha}_{t+1}^{\omega_{t+1}} g_{t+1}^{\omega_{t+1}} \leq f_t(x_t, \omega_{t+1}) \leq h_t(x_t, \omega_{t+1}) \quad \forall x_t. \quad (2.3)$$

The left hand inequality is immediate by writing (2.1) in its dual form. The right hand inequality follows as the constraints representing valid cuts in (2.1) are dominated by the cuts of (2.2) which define the recourse function. The desired result is obtained by multiplying (2.3) by $p_{t+1}^{\omega_{t+1}|\omega_t}$ and summing over all $\omega_{t+1} \in \Delta(\omega_t)$. ■

We now describe a method for generating valid cuts for a subproblem without solving its descendants. Denote the dual variables associated with structural constraints, cuts, and simple upper bounds by $\pi_t^{\omega_t}$, $\alpha_t^{\omega_t}$, and $-\mu_t^{\omega_t}$, respectively. With this notation, the dual of $\text{sub}(\omega_t)$ ($1 \leq t < T$) may be written

$$\begin{aligned}
& \text{maximize} && \pi_t^{\omega_t} \left(b_t^{\omega_t} + B_{t-1}^{\omega_t} x_{t-1}^{a(\omega_t)} \right) + \alpha_t^{\omega_t} g_t^{\omega_t} - \mu_t^{\omega_t} u_t^{\omega_t} \\
& \text{subject to} && \pi_t^{\omega_t} A_t^{\omega_t} - \alpha_t^{\omega_t} G_t^{\omega_t} - \mu_t^{\omega_t} \leq c_t^{\omega_t} \\
& && e^T \alpha_t^{\omega_t} = 1 \\
& && \alpha_t^{\omega_t} \geq 0 \quad \mu_t^{\omega_t} \geq 0.
\end{aligned}$$

In describing the method, super and subscripts are suppressed for clarity. Suppose we have solved a stage t subproblem with cost vector $\tilde{c}$, gradient matrix $\tilde{G}$, and structural matrix $\tilde{A}$ and we have obtained optimal dual prices $(\tilde{\pi}, \tilde{\alpha}, \tilde{\mu})$. Given another stage t subproblem with cost vector c , gradient matrix G , and structural matrix A , feasible dual prices for this problem can be generated directly from $(\tilde{\pi}, \tilde{\alpha}, \tilde{\mu})$ via

$$(\pi, \alpha, \mu) = \left(\tilde{\pi}, \tilde{\alpha}, [\tilde{\pi}A - \tilde{\alpha}G - c]^+ \right). \quad (2.4)$$

Dual feasibility of (π, α, μ) is easily verified by substitution. The $[v]^+$ notation means take the positive part of the vector v , component wise. We will refer to (2.4) as the *dual sharing formula*. Note the dual sharing formula is also valid for stage T subproblems when the (vacuous) cut gradient matrix and associated dual variables are dropped.

Suppose we have solved the descendants of the stage t subproblem $\text{sub}(\omega_t)$. Using the optimal dual variables from these descendants, the dual sharing formula may be applied to compute a valid cut for $\text{sub}(\omega'_t)$. To compute this cut, we match elements of $\Delta(\omega_t)$ and $\Delta(\omega'_t)$. Other approaches, however, are clearly possible. For example, we could select the dual vectors which produce the strongest cut at a particular stage t decision. This could be accomplished by evaluating the dual objective function of each descendant problem at all dual vectors in a finite candidate set and selecting the vector which yields the largest value.

A cut generated by naively applying the dual sharing formula can be arbitrarily weak. For example, suppose a component of a primal decision vector has no simple upper bound (i.e., the bound is $+\infty$) in both the donor and recipient subproblems. The corresponding donor $\tilde{\mu}_i$ will have value zero; however, if the two subproblems have different costs for this decision then it is possible

the shared dual satisfy $\mu_i > 0$. The generated cut would then have an intercept of $-\infty$. Fortunately, this situation does not arise for the stochastic hydroelectric scheduling test problems described in Section 2.3; the only arcs which can have nonzero shared μ 's have natural finite upper bounds.

2.3 Test Problems

In the subsequent sections of this chapter, the order of presentation will be to describe a modification to the traditional nested Benders algorithm and then to observe its performance empirically. We begin by briefly describing a collection of test problems and an implementation of a Benders decomposition code.

The test problems are preliminary versions of multistage stochastic hydroelectric scheduling problems that will serve as subproblems in a larger stochastic midterm hydro-thermal optimization model for Pacific Gas and Electric Company. The research project at PG&E (see Jacobs et al. [39]) is ongoing at the time of this writing (July 1993). The test problems are models of multireservoir hydrological basins in central and northern California. The natural inflows into reservoirs, marginal values of energy, and simple bounds on arc flows are the stochastic parameters. The test problems are models with different time horizons, stage definitions, and discretizations of natural inflow distributions. The models are based on two different hydrological basins: Mokelumne (Moke) and Yuba-Bear-South Feather (Ybsf). The assumption of complete recourse is modeled by creating an auxiliary source with ample supply. Each reservoir can receive from and send water to the supply source, but there is a large penalty cost associated with such transactions. The reader is referred to Chapter 1 and Jacobs et al. [39] for more detailed descriptions of the stochastic hydroelectric scheduling model.

In Table 2.1, "Name" indicates the hydrological basin, the number of stages and number of scenarios; for example, Moke3.9 is a three stage model of the Mokelumne basin with $|\Omega_3| = 9$. "Size" gives the row and column dimensions of the A_t matrices for each stage $t = 1, \dots, T$. The dimension of the domain of the recourse function is the number of large reservoirs in a basin, and this value is denoted "Dim." "Deterministic Equivalent" gives the row and column dimensions of the linear program given by the SLP-T formulation. Simple upper bounds are not included, but the complete

recourse nodes and arcs are included in “Size” and “Deterministic Equivalent” dimensions.

Name	Size	Dim	Deterministic Equivalent
Moke3.9	$169 \times 820, 337 \times 1713, 673 \times 3298$	7	7237×28473
Ybsf3.9	$319 \times 1559, 637 \times 3119, 1273 \times 6239$	12	13687×53489
Moke4.45	$57 \times 271, 113 \times 548, 337 \times 1713, 673 \times 3298$	7	35736×140656
Ybsf4.45	$107 \times 519, 213 \times 1039, 637 \times 3119, 1273 \times 6239$	12	68012×265836

Table 2.1: Test Problems

The algorithm is implemented in FORTRAN 77, and the computational results are from a Hewlett Packard 9000/750 workstation. The code uses NETSIDE developed by Kennington and Farhangian as the subproblem solver. NETSIDE is a special purpose code which solves the minimum cost flow network problem *with side constraints*. The algorithm is a specialization of the primal simplex method (see Kennington and Helgason [42] and Barr et al. [2]). In our setting the set of side constraints includes Benders cuts.

The performance of a particular algorithmic enhancement will be analyzed with respect to a base case strategy. The base case strategy is a “good” or recommended strategy, and thus we evaluate the marginal effect of each enhancement. We also occasionally indicate how an enhancement’s impact would differ under alternative strategies. The base case strategy is given in Table 2.2 for reference, and the details are given in the respective sections that follow.

<i>Warm Start</i>	Candidate list of length 20 until relative error $\leq 5\%$
<i>Advanced Start</i>	Prespecified Decisions Method with shared cuts
<i>Multicut</i>	Yes
<i>Tree Traversing Strategy</i>	Fastpass

Table 2.2: Base Case Strategy

All problems are solved to within an objective function tolerance of 0.01%. Table 2.3 summarizes the performance of the base case strategy on the four test problems. Reported CPU times exclude input and output operations. The # subproblems column gives the number of subproblems solved in each stage during the course of the algorithm.

Name	# subproblems	# iterations	CPU (sec.)
Moke3.9	10-59-99	10	55.4
Ybsf3.9	7-47-72	7	119.8
Moke4.45	10-64-274-459	10	221.4
Ybsf4.45	8-83-205-369	8	404.0
Average	N/A	8.8	200.2

Table 2.3: Base Case Strategy Performance

2.4 Warm Start

At any given iteration of a decomposition algorithm for SLP-T, the subproblems associated with the various branches of the decision tree for a stage may be similar. A particular subproblem may also be similar from one iteration to the next; indeed, we might expect this similarity to increase as the converging algorithm proceeds. Techniques designed to take advantage of optimal basis information from previous subproblem solutions are critical for developing efficient algorithms.

Wets [64] [62] and Garstka and Rutenberg [30] have proposed bunching and sifting techniques, respectively for solving a large number of similar linear programs. The sifting method, however, requires component-wise independence of the right-hand-side and deterministic objective function coefficients; our test problems violate these requirements. Moreover, in our experiments there were a low number of “repeat” optimal bases which seemed to indicate bunching might not occur. While the primary computational challenge in the two stage problem lies in the solution of a large number of similar second stage problems *each iteration*, the greatest potential for computational savings in a multistage problem, with only a few stochastic branches each stage, rests in selecting good initial bases for each subproblem.

We propose a warm start technique in which initial bases are selected from a candidate list until the relative error is sufficiently small. In subsequent iterations subproblems are initialized with the basis from their previous optimal solution. For the stochastic hydro subproblems under consideration, most rows of the constraint matrix are part of a network flow problem and relatively few rows correspond to side constraints. The columns of a basic solution can be partitioned into a network component and a side constraint component (see Kennington and Helgason [42]). The

heuristic used to select the best basis from the candidate list for a particular subproblem proceeds as follows:

- (i) Use the *network* component of the bases from the candidate list and determine the corresponding network flows.
- (ii) Calculate solutions for the entire subproblem based on each network flow solution.
- (iii) Determine the objective function value of each solution.
- (iv) Select the basis from the candidate list that has the minimum corresponding objective function value.

Step (i) can be performed quickly due to the tree structure of the network basis. In step (ii) we simply substitute the solution from (i) into the side constraints and generate a feasible solution from slack, surplus, artificial, or future cost (i.e., θ) variables. We then evaluate the objective function at each solution and select the corresponding “best” basis from the candidate list. The objective function value calculated in (iii) is only an estimate of the actual objective function value the basis will yield since the side constraints’ contribution to the basis is not considered. The method for selecting a warm start basis ignores the side constraints due to the computational effort required for refactorization relative to the time required to find a basic feasible flow in the network and the fact the network constitutes most of the subproblem. Minimal storage is required for each basis: arc indices within a pointer structure, a list of upper bound arc indices for the network, and a list of column indices for the side constraints. The actual warm start procedure within the NETSIDE optimizer utilizes the basic columns of the side constraints; see Jacobs [38] for the details of the basis insertion. The parameters the user must select are the maximum size of the candidate lists and the relative error at which the method switches from candidate lists to simply reusing the previous optimal basis for each subproblem. Reasonable values for these parameters are suggested below in the computational results discussion.

Computational Results

Table 2.4 details the performance of the algorithm when each subproblem solution begins with an all artificial basis. The “ $\times$ increase” column is defined as T/T_{bc} , where T denotes the running time of the current strategy (i.e., base case with no warm start) and T_{bc} denotes the running time of the

base case strategy. Thus, if the "x increase" column has value 20 then it is to be read: "the current strategy's running time is slower than the base case by a factor of 20." The "x increase" column for the Average row is the average of the "x increase" column, not the factor increase of the average running time; in this way, we give equal weight to all test problems. Table 2.4 reveals the dramatic impact of warm starts.

Name	# iterations	CPU (sec.)	x increase
Moke3.9	13	909.1	16.4
Ybsf3.9	8	1973.3	16.5
Moke4.45	11	3793.4	17.1
Ybsf4.45	11	14267.8	35.3
Average	10.75	5253.9	21.3

Table 2.4: No Warm Start

Table 2.5 summarizes the performance of the warm start procedure in which we simply recall the previous optimal basis for each subproblem. If such a basis does not exist (because a particular subproblem has not yet been solved) then the optimal basis of another subproblem from the same stage is used. The first subproblem solved on each stage is initialized with the all-artificial basis. The % increase column is defined as $(T - T_{bc})/T_{bc} \cdot 100$. Thus, if the % increase column is 20 then it is to be read: "the current strategy's running time is 20% longer than the running time of the base case strategy." The % increase column for the Average row is the average of the % increase column.

Name	# iterations	CPU (sec.)	% increase
Moke3.9	12	75.0	35.4
Ybsf3.9	8	163.4	36.4
Moke4.45	11	263.4	19.0
Ybsf4.45	9	526.1	30.2
Average	10	257.0	30.3

Table 2.5: Warm Start Recall Previous Basis

Table 2.5 indicates substantial computational savings can result from using the heuristic to select good bases early in the algorithm. The values of 5% for the switch over tolerance and 20 for the candidate list length (see Table 2.2) were determined by varying these parameters on a number of test

problems using the base case strategy and the base case strategy with no advanced start. Moke4.45 is a typical example: If we decrease the switch over tolerance from 5% to 1% the candidate list heuristic applies during the advanced start and the first four iterations as opposed to the advanced start and first two iterations. In this case, better initial bases resulted in a decrease of CPU time spent within the subproblem optimizer, NETSIDE, from 197.2 sec. to 194.1 sec. However, the time the heuristic spent searching for good bases increased from 15.8 sec. to 31.0 sec. The result was a net increase in total CPU time. The trade off between NETSIDE optimization time and heuristic search time is similar when the candidate list length is modified.

2.5 Advanced Start

If a Benders decomposition algorithm is initiated with no preliminary cuts then the first iteration decisions will be myopic, and in the early iterations, the decisions are likely to be "extreme." The idea behind an advanced start method is to calculate some valid preliminary cuts which will help guide the early iterations of the algorithm. In Sections 2.5.1 and 2.5.2 we describe two basic approaches for computing advanced start cuts.

2.5.1 Expected Value Method

The *expected value problem* is the multistage deterministic linear program in which the stochastic parameters of SLP-T are replaced by their population means. Clearly, we may use Benders decomposition to solve such a problem; the expected value problem is a special case of SLP-T with just one scenario. The expected value method for computing advanced start cuts is to solve the expected value problem within a prespecified tolerance and use the cuts produced in the process to initiate the decomposition algorithm for SLP-T. The question then naturally arises: Under what conditions will the cuts produced during the solution of the expected value problem be valid cuts for SLP-T?

We begin by examining this question for SLP-2. Consider the dual feasible regions of the second stage subproblems in the special case in which $A_2^\omega = A_2$ and $c_2^\omega = c_2 \forall \omega \in \Omega$. In this case, a dual feasible vector, π , generates a cut in the expected value problem of the following form: $\theta \geq \pi \bar{B}_1 x_1 + \pi \bar{b}_2$, where $\bar{b}_2 = \sum_{\omega \in \Omega} p^\omega b_2^\omega$ and $\bar{B}_1 = \sum_{\omega \in \Omega} p^\omega B_1^\omega$. In SLP-2, the second stage dual feasible regions are nonstochastic, and thus, the set of vectors $(\pi^1, \dots, \pi^K) = (\pi, \dots, \pi)$ are feasible

for the second stage subproblems. By Lemma 2.1 these vectors generate a valid cut of the form: $\theta \geq \sum_{\omega \in \Omega} p^\omega \hat{\pi} B_1^\omega + \sum_{\omega \in \Omega} p^\omega \hat{\pi} b_2^\omega = \hat{\pi} \bar{B}_1 x_1 + \hat{\pi} \bar{b}_2$. Therefore, we may conclude, in the special case when SLP-2 has nonstochastic second stage dual feasible regions, the cuts produced during a Benders decomposition solution of the expected value problem are valid cuts for SLP-2. Infanger has utilized this method [36].

Consider the special case of SLP-T where $A_t^{\omega_t} = A_t$, $c_t^{\omega_t} = c_t$, and the stochastic parameters exhibit interstage independence. In this case, a similar analysis shows the cuts produced during the solution of the expected value problem are valid for SLP-T. In particular, we may apply Lemma 2.1 and the argument of the previous paragraph to conclude the expected value cuts are valid for the stage $T-1$ subproblems. Using this as the base case, inductive application of Lemma 2.2 shows the stage t expected value cuts are valid for SLP-T since: (i) the stage $t+1$ expected value cuts are valid by inductive hypothesis and (ii) the dual variables for the expected value problem are feasible for the *nonstochastic* dual feasible regions of SLP-T. Note we require interstage independence so that

$$\sum_{\omega_{t+1} \in \Delta(\omega_t)} p_{t+1}^{\omega_{t+1}|\omega_t} B_t^{\omega_{t+1}} = \bar{B}_t \quad \text{and}$$

$$\sum_{\omega_{t+1} \in \Delta(\omega_t)} p_{t+1}^{\omega_{t+1}|\omega_t} b_{t+1}^{\omega_{t+1}} = \bar{b}_{t+1}.$$

In the stochastic hydroelectric scheduling problems described in Section 2.3, the objective function coefficients correspond to marginal values of energy and are subject to uncertainty via hydro (natural inflow) conditions, and the natural inflow vectors are temporally correlated. Thus, the expected value method is not applicable and we see an alternate approach.

2.5.2 Prespecified Decisions Methods

We describe two “prespecified decisions” methods for computing preliminary cuts. The first method computes stronger cuts, but the second requires less computational work. These methods require the “user” to designate, for each stage t ($1 \leq t < T$), a collection of decision vectors, denoted $\{x_t^{(i)} : i = 1, \dots, N_t\}$; a typical value of N_t is three. The basic idea behind the prespecified decision methods is to generate preliminary cuts by solving subproblems with right-hand sides of the form: $B_{t-1}^{\omega_{t-1}} x_{t-1}^{(i)} + b_t^{\omega_t}$. The first method is outlined in Figure 2.1, and while the method represents

considerable work, the hope is if the prespecified decisions are selected in an intelligent fashion then we will gain a computational advantage over naive application of Benders algorithm.

```

do  $t = T$  downto 2
  do  $i = 1$  to  $N_{t-1}$ 
    do  $\omega_{t-1} \in \Omega_{t-1}$ 
      do  $\omega_t \in \Delta(\omega_{t-1})$ 
        form RHS of sub( $\omega_t$ ):  $B_{t-1}x_{t-1}^{(i)} + b_t^{\omega_t}$ 
        solve sub( $\omega_t$ )
      enddo
      pass cut to sub( $\omega_{t-1}$ )
    enddo
  enddo
enddo

```

Figure 2.1: Prespecified Decisions Method - No Sharing

The second prespecified decisions method is similar, but it significantly reduces the computational work required. In particular, it reduces the work of the “do $\omega_{t-1} \in \Omega_{t-1}$ ” loop by instead selecting a single scenario $\omega_{t-1} = \bar{\omega}_{t-1}$. “Reasonable” choices of scenario ω_{t-1} include an appropriately defined “median” scenario or the mean scenario; we do not require $\bar{\omega}_{t-1} \in \Omega_{t-1}$. The descendants of scenario ω_{t-1} are solved and then a cut is passed back not only to sub(ω_{t-1}) but also to all its neighbors via the dual sharing formula (2.4). In utilizing the dual sharing formula to compute valid cuts for sub(ω_{t-1}), (for $\omega_{t-1} \neq \bar{\omega}_{t-1}$) we match elements of $\Delta(\omega_{t-1})$ and $\Delta(\bar{\omega}_{t-1})$. Figure 2.2 summarizes the method. This streamlined method can significantly reduce the computational work required when compared with the method of Figure 2.1. For example, in a symmetric four stage, 45 scenario problem with 1, 3, 15, and 45 scenarios on each stage the number of subproblems solved in the two methods is 189 and 33, respectively when $N_t = 3$.

The only relevant components of the prespecified decision vectors are ones which contribute to the product $B_{t-1}^{\omega_t} x_{t-1}^{(i)}$. In the stochastic hydro scheduling problem this corresponds to end of stage reservoir storage volumes. Reservoirs have natural lower and upper storage bounds, and in the absence of additional information (e.g., historical storage levels or values from prior optimizations), the prespecified decisions are defined as percentiles between the upper and lower bounds. In particular, we use three prespecified vectors at 20%, 50%, and 80%, and we define $\bar{\omega}_t = \lceil K_t/2 \rceil$ where

```

define  $\bar{\omega}_t, t = 1, \dots, T-1$ 
do  $t = T$  downto 2
  do  $i = 1$  to  $N_{t-1}$ 
    do  $\omega_t \in \Delta(\bar{\omega}_{t-1})$ 
      form RHS of sub( $\omega_t$ ):  $B_{t-1}x_{t-1}^{(1)} + b_t^{\omega_t}$ 
      solve sub( $\omega_t$ )
    enddo
    pass cut to sub( $\omega_{t-1}$ )  $\forall \omega_{t-1} \in \Omega_{t-1}$ 
  enddo
enddo

```

Figure 2.2: Prespecified Decisions Method – Sharing Duals

$\Omega_t = \{1, \dots, K_t\}$ and where $\lceil \cdot \rceil$ is a function that returns the smallest integer greater than its argument. The order of cut computation in these two advanced start methods is relevant. For example, in a three stage problem all preliminary cuts are passed to the second stage prior to passing any cuts to the first stage. In this way, the maximum possible amount of information is subsequently passed to the first stage.

Computational Results

Table 2.6 details the performance of the strategy in which no advanced start is used, and Table 2.7 shows the performance of the strategy in which we use the advanced start without the dual sharing formula.

Name	# iterations	CPU (sec.)	% increase
Moke3.9	15	68.4	23.5
Ybsf3.9	9	118.5	-1.1
Moke4.45	13	318.9	44.0
Ybsf4.45	13	614.7	52.2
Average	12.5	280.1	29.7

Table 2.6: No Advanced Start

In choosing an advanced start procedure, one must balance the computational benefit that the preliminary cuts yield against the cost of generating the cuts. Tables 2.6 and 2.7 show the base case strategy of solving only a subset of subproblems on each stage and utilizing the dual sharing formula provides an attractive advanced start. The average relative error after the first iteration for the base case strategy and advanced start strategy with no sharing is 6.8% and 6.1%, respectively.

Name	# iterations	CPU (sec.)	% increase
Moke3.9	10	60.0	8.3
Ybsf3.9	7	129.1	7.8
Moke4.45	9	255.3	15.3
Ybsf4.45	9	553.1	36.9
Average	8.8	249.4	17.1

Table 2.7: Advanced Start: No Dual Sharing

However, the corresponding average CPU times for the first iteration are 67.6 sec. and 117.8 sec., respectively. While the method of Figure 2.1 produces slightly stronger preliminary cuts than the method of Figure 2.2, the empirical results indicate the computational expense is too high. The tables reveal, however, the slower advanced start procedure is preferable to none at all. Both tables also indicate the advanced start procedures provide greater computational savings in the four stage problems than in the three stage problems.

Figure 2.3: Relative error vs. CPU time - Moke4.45

The three advanced start strategies' relative error as a function of CPU time is plotted in Figure 2.3 for test problem Moke4.45. The ordinate has a $\log_{10}$ scale; thus -1 corresponds to 10%

relative error and -2 corresponds to 1%, etc. One may observe the computational effort required for the advanced start (if any) through the first iteration, and the corresponding relative errors, for each strategy. Also, note the effect of the warm start procedure described in Section 2.4; as the algorithm proceeds, the time required per iteration decreases markedly.

2.6 Multicut Algorithms

Birge and Louveaux [10] introduced the multicut L-shaped algorithm for SLP-2. This algorithm extends naturally to the multistage setting; thus, we will describe the details for the two-stage problem. Recall SLP-2 may be written:

$$\begin{array}{ll} \text{minimize} & \left\{ c_1 x_1 + \sum_{\omega \in \Omega} p^\omega h(x_1, \omega) \right\}, \\ \text{subject to} & A_1 x_1 = b_1 \\ & x_1 \geq 0 \end{array}$$

where

$$h(x_1, \omega) = \begin{array}{ll} \text{maximize} & \pi (b_2^\omega + B_1^\omega x_1) \\ \text{subject to} & \pi A_2^\omega \leq c_2^\omega. \end{array}$$

The traditional aggregate cut Benders decomposition algorithm creates a master program by replacing $\sum_{\omega \in \Omega} p^\omega h(x_1, \omega)$ by θ and sequentially adding cuts of the form $\theta \geq (\sum_{\omega \in \Omega} p^\omega \pi^\omega B_1^\omega) x_1 + \sum_{\omega \in \Omega} p^\omega \pi^\omega b_2^\omega$ each iteration. The multicut version instead replaces $h(x_1, \omega)$ by θ^ω for all $\omega \in \Omega$ and each iteration appends $|\Omega|$ cuts of the form $\theta^\omega \geq (\pi^\omega B_1^\omega) x_1 + \pi^\omega b_2^\omega$. When compared with the aggregate cut algorithm, the multicut algorithm has the disadvantage of requiring more decision variables; similarly, after a given number of iterations, it also maintains a larger number of cut constraints in the master program. This, however, is countered by the advantage of increased resolution of the recourse function. The example in Figure 2.4 is designed to illustrate this concept; referring to Figure 2.4, suppose in the first two iterations of the multicut algorithm we make (scalar) master program decisions such that one decision is less than 2.0 and the other is greater than 3.0. In this case, we will have information regarding the recourse function that the aggregate cut algorithm would require three iterations to acquire. In practice we expect multicut algorithms will typically take fewer iterations than their aggregate cut counterparts. (Birge and Louveaux [10] however, present a counter example showing this is not always the case.)

Figure 2.4: Recourse function resolution

The multicut algorithm extends to the multistage setting in a straightforward fashion. Each descendant scenario passes back a cut to its ancestor and the ancestor objective function has the form:

$$c_t^{*T} x_t^{*} + \sum_{\omega_{t+1} \in \Delta(\omega_t)} p_t^{\omega_{t+1}|\omega_t} \theta_t^{\omega_{t+1}|\omega_t}.$$

Other generalizations of the multicut algorithm are possible; the descendants can be partitioned into disjoint sets and a “ θ ” defined for each set. In the multicut algorithm described above, each set of the partition is a singleton, and the aggregate cut algorithm is the special case where the only set of the partition is $\Delta(\omega_t)$ itself. A coarse partition version of the multicut algorithm should be particularly attractive when the number of scenarios is large. The other enhancements discussed in this chapter can be programmed to run in either the aggregate cut or multicut mode.

Computational Results

Table 2.8 details the performance of the strategy in which the multicut method is replaced by the single cut procedure in the base case strategy.

The average number of iterations in the multicut procedure is about half that of the single cut method. However, the quality of the warm start procedure is affected by how much the subproblems’ right hand sides differ from the previous iteration, and this, in turn, is closely tied to the relative error. Thus, it is difficult to measure the detrimental effect of the additional cuts and decision variables in the multicut method at any particular iteration. In any case, Table 2.8 shows the multicut method yields a significant computational advantage. Figure 2.5 depicts the effect of the

Name	# iterations	CPU (sec.)	% increase
Moke3.9	22	107.3	93.7
Ybsf3.9	13	146.3	22.1
Moke4.45	18	318.1	43.7
Ybsf4.45	17	662.0	63.9
Average	17.5	308.4	55.9

Table 2.8: Single Cut

multicut procedure's increased resolution on the convergence of the algorithm by comparing the Relative error vs. CPU time plots of the single and multicut methods for test problem Ybsf4.45.

Figure 2.5: Relative error vs. CPU time Ybsf4.45

2.7 Tree Traversing Strategies

In Chapter 1 we indicated that the extension of Benders algorithm from SLP-2 to SLP-T can be viewed as follows. SLP-2 is a linear program which is solved by decomposition into two stages; thus, SLP-T is first decomposed into two stages: stage 1 and stage 2, ..., T . After "master 1" makes decision x_1 and the corresponding resources are placed on the right-hand-sides of the stage 2 subproblems, there are $|\Omega_2|$ linear programs to solve. Each of these linear programs is solved

via decomposition into two stages: stage 2 and stage 3, ..., T etc. This recursive method is the traditional nested Benders decomposition approach to multistage stochastic linear programs (see Birge [7]). We refer to this decision tree traversing strategy as *shuffle* and describe the algorithm more formally below. Abrahamson [1] and Wittrock [66] developed tree traversing strategies other than *shuffle* for deterministic multistage linear programs; we consider two of these strategies: *fastpass* and *cautious*. Gassmann [31] tested *shuffle*, *fastpass*, and *cautious* in the stochastic setting. Abrahamson, Wittrock, and Gassmann found *fastpass* to be an effective strategy. In addition to these three strategies two new classes of tree traversing strategies are presented.

We now state a theorem which permits consideration of tree traversing strategies other than *shuffle*. In *shuffle*, a subproblem only receives a cut from its descendants after each descendant subproblem is solved with respect to its descendants. In this context *solved* means the upper and lower bounds on the optimal objective function value coincide. The tree traversing theorem states that a cut passed back to a subproblem “prematurely” (i.e., while the gaps in descendant objective function bounds are still positive) is a valid cut.

Theorem 2.3 *A cut passed back to $\text{sub}(\omega_t)$ when the subproblems, $\text{sub}(\omega_{t+1})$, $\omega_{t+1} \in \Delta(\omega_t)$, are not solved with respect to their descendants (due to insufficient cuts) is a valid cut.*

Proof

This result follows directly from Lemma 2.1 and Lemma 2.2: The subproblems contain valid cuts, and the dual variables associated with optimal solutions to the subproblems are dual feasible. ■

As a consequence of Theorem 2.3, one has essentially a free choice in designing algorithms with respect to the order in which the subproblems of the decision tree are solved. Our goal is to devise a strategy which will allow us to solve large-scale multistage stochastic linear programs as quickly as possible.

2.7.1 The Strategies

Absolute error and *discrepancy* will be two useful concepts in formally defining the tree traversing strategies. The definition of the absolute error, $\mathcal{A}_t(x_t^{\omega_t}, \omega_t)$, for the stage t subproblem given outcome ω_t , i.e., $\text{sub}(\omega_t)$, is

$$\mathcal{A}_t(x_t^{\omega_t}, \omega_t) = \left\{ \sum_{i=t+1}^T \sum_{\omega_i \in \Delta^{T-i}(\omega_t)} p_i^{\omega_i | \omega_t} c_i^{\omega_i} x_i^{\omega_i} \right\} - \theta_t^{\omega_t}.$$

This expression is simply the difference between upper and lower bounds on the optimal objective function value for $\text{sub}(\omega_t)$; note this optimal objective function value depends on $\text{sub}(\omega_t)$'s right hand-side and therefore its ancestor's decisions. The absolute error for any last-stage T subproblem is zero because these linear programs are solved directly. The expected absolute error for stage t is $E_{\omega_t} \mathcal{A}_t(x_t^{\omega_t}, \omega_t)$. Recall, the Δ notation applied to a set, $\mathcal{S}$, is defined: $\Delta(\mathcal{S}) = \bigcup_{s \in \mathcal{S}} \Delta(s)$ and that Δ^n means apply Δ n times, e.g., $\Delta^2(\omega_t) = \Delta(\Delta(\omega_t))$.

Scott [58] defined *discrepancy*, $\mathcal{D}_t$, in the deterministic case. We extend the notion of discrepancy to the stochastic setting. The t^{th} stage expected discrepancy is defined to be $E_{\omega_t} \mathcal{D}_t(x_t^{\omega_t}, \omega_t)$, where

$$\mathcal{D}_t(x_t^{\omega_t}, \omega_t) = \left\{ \sum_{\omega_{t+1} \in \Delta(\omega_t)} p_{t+1}^{\omega_{t+1} | \omega_t} [c_{t+1}^{\omega_{t+1}} x_{t+1}^{\omega_{t+1}} + \theta_{t+1}^{\omega_{t+1}}] \right\} - \theta_t^{\omega_t}$$

is the discrepancy for $\text{sub}(\omega_t)$. The second term in the discrepancy, $\theta_t^{\omega_t}$, represents $\text{sub}(\omega_t)$'s current estimate of the expected cost to be incurred by its descendants in stages $t+1, \dots, T$. The first term represents the expected cost realized in stage $t+1$ plus the stage $t+1$ estimate of the expected cost in stages $t+2, \dots, T$ (conditioned on ω_t). The discrepancy for any stage T subproblem is zero.

We now define two subroutines: *forward*(u, v) and *backward*(v, u). The former sweeps forward from stage u to stage v . It first forms the right hand sides of the appropriate stage u subproblems (if $u = 1$ this is trivial) and solves them, and then it forms the right hand sides of the appropriate stage $u+1$ subproblems and solves them $\dots$ until it has solved all the appropriate stage v subproblems. On the other hand, *backward*(v, u) first passes cuts back to the appropriate stage v subproblems and solves them, and then passes cuts back to the appropriate stage $v-1$ subproblems and solves them $\dots$ until it has passed cuts back to the appropriate stage u subproblems (it does not solve them). In the execution of *forward*(u, v) and *backward*(v, u), it is not always necessary to solve every subproblem that we might address. For example, in subroutine *forward*(u, v), some of the stage u subproblems (and therefore its descendants) might have a sufficiently small absolute error, and in subroutine *backward*(v, u), some of the stage v subproblems (and possibly its ancestors) might have a sufficiently small discrepancy. For declaring specific subproblems temporarily "solved" in this

fashion, we use discrepancies in the *cautious* strategies and absolute errors in the *shuffle* strategies. In *fastpass*, we will use absolute errors (note that discrepancy could be used instead).

When iterating between $\text{sub}(\omega_t)$ and its descendants in stage $t + 1$, one can show $\mathcal{D}_t(x_t^{\omega_t}, \omega_t) \geq 0$. Furthermore, $\mathcal{D}_t(x_t^{\omega_t}, \omega_t) = 0$ if and only if the cut that would be passed back to $\text{sub}(\omega_t)$ is a redundant cut. Two more useful facts relating absolute error and discrepancy are:

$$E_{\omega_t} \mathcal{D}_t(x_t^{\omega_t}, \omega_t) = E_{\omega_t} \mathcal{A}_t(x_t^{\omega_t}, \omega_t) - E_{\omega_{t+1}} \mathcal{A}_{t+1}(x_{t+1}^{\omega_{t+1}}, \omega_{t+1}) \quad (2.5)$$

$$\mathcal{A}_1(x_1^{\omega_1}, \omega_1) = \sum_{t=1}^{T-1} E_{\omega_t} \mathcal{D}_t(x_t^{\omega_t}, \omega_t). \quad (2.6)$$

When solving SLP-T, the user must specify a tolerance, say *toler*. When

$$\mathcal{A}_1(x_1, \omega_1) \leq \text{toler} \cdot \max\{\min(|U|, |L|), \epsilon'\},$$

SLP-T is declared to be solved. U and L denote upper and lower bounds, respectively, on $\text{sub}(\omega_1)$ and ϵ' is an additional predefined tolerance. In what follows, we will abbreviate this stopping rule by $\mathcal{A}_1(x_1, \omega_1) \leq \text{toler} \cdot \min(|U|, |L|)$.

The definitions of sufficiently small expected absolute error and sufficiently small expected discrepancy for $\text{sub}(\omega_t)$ are motivated by (2.5) and (2.6). These equations indicate we should expect, all else being equal, each stage's expected discrepancy to have similar values. On the other hand, we should expect the expected absolute error will increase (additively) as the stage decreases. These observations lead to the following guidelines regarding the definition of "sufficiently small":

$$E_{\omega_t} \mathcal{A}_t(x_t^{\omega_t}, \omega_t) = \sum_{\omega_i \in \Omega_t} p_i^{\omega_t} \mathcal{A}_t(x_t^{\omega_i}, \omega_i) \leq \frac{(T-t)\text{toler}}{T-1} \cdot \min(|U|, |L|) \quad (2.7)$$

$$E_{\omega_t} \mathcal{D}_t(x_t^{\omega_t}, \omega_t) = \sum_{\omega_i \in \Omega_t} p_i^{\omega_t} \mathcal{D}_t(x_t^{\omega_i}, \omega_i) \leq \frac{\text{toler}}{T-1} \cdot \min(|U|, |L|). \quad (2.8)$$

$\mathcal{A}_t(x_t^{\omega_t}, \omega_t)$ may be defined at the conclusion of a forward sweep. $\mathcal{D}_t(x_t^{\omega_t}, \omega_t)$ may be defined after the descendant stage $t + 1$ subproblems have been solved. In the five strategies outlined below, (2.7) and (2.8) are utilized.

(1) *fastpass*

- step 0 define *toler*; set *iter* = 1
- step 1 *forward*(1, *T*)
- step 2 if $\mathcal{A}_1(x_1, \omega_1) \leq \text{toler} \cdot \min(|U|, |L|)$ then stop: optimal solution at hand
- step 3 *backward*(*T* - 1, 1); *iter* = *iter* + 1
- step 4 go to step 1

(2) *shuffle*

- step 0 define *toler*; set *iter* = 1, *t* = 1
- step 1 *forward*(*t*, *T*)
- step 2 if $\mathcal{A}_1(x_1, \omega_1) \leq \text{toler} \cdot \min(|U|, |L|)$ then stop: optimal solution at hand
- step 3 $t = \max \left\{ i : E_{\omega_1} \mathcal{A}_1(x_i^{**}, \omega_i) > \frac{\text{toler}}{T-1} \cdot \min(|U|, |L|) \right\}$
- step 4 *backward*(*t*, *t*)
- step 5 if *t* = 1 then *iter* = *iter* + 1
- step 6 go to step 1

(3) *cautious*

- step 0 define *toler*; set *iter* = 1, *t*₁ = 1, *t*₂ = 2
- step 1 apply one iteration of the *fastpass* algorithm¹; *iter* = *iter* + 1
- step 2 *forward*(*t*₁, *t*₂)
- step 3 if *t*₂ = *T* then
 - iter* = *iter* + 1;
 - if $\mathcal{A}_1(x_1, \omega_1) \leq \text{toler} \cdot \min(|U|, |L|)$ then stop: optimal solution at hand
- step 4 if $E_{\omega_{t_2-1}} D_{t_2-1}(x_{t_2-1}^{**}, \omega_{t_2-1}) \leq \frac{\text{toler}}{T-1} \cdot \min(|U|, |L|)$ then
 - t*₁ = *t*₂ + 1;
 - t*₂ = *t*₁
- else
 - backward*(*t*₂ - 1, 1);
 - t*₁ = 1,
 - t*₂ = 2
- step 5 go to step 2

¹ *cautious* and *r-cautious* begin with one iteration of *fastpass* so initial upper and lower bounds on $\text{sub}(\omega_1)$ may be defined.

(4) *c-shuffle*

- step 0 define *toler*, *c*; set *iter* = 1, *t* = 1
- step 1 forward(*t*, *T*)
- step 2 if $\mathcal{A}_1(x_1, \omega_1) \leq \text{toler} \cdot \min(|U|, |L|)$ then stop: optimal solution at hand
- step 3 $t = \max \left\{ \max \left\{ i : E_{\omega_i} \mathcal{A}_i(x_i^{*i}, \omega_i) > \frac{t-1}{T-1} \cdot c \cdot \min(|U|, |L|) \right\}, 1 \right\}$
- step 4 backward(*T* - 1, *t*)
- step 5 if *t* = 1 then *iter* = *iter* + 1
- step 6 go to step 1

(5) *c-cautious*

- step 0 define *toler*, *c*; set *iter* = 1, *t*₁ = 1, *t*₂ = 2
- step 1 apply one iteration of the *fastpass* algorithm¹; *iter* = *iter* + 1
- step 2 forward(*t*₁, *t*₂)
- step 3 if *t*₂ = *T* then
 - iter* = *iter* + 1;
 - if $\mathcal{A}_1(x_1, \omega_1) \leq \text{toler} \cdot \min(|U|, |L|)$ then stop: optimal solution at hand
- step 4 if $E_{\omega_{t_2-1}} D_{t_2-1}(x_{t_2-1}^{*t_2-1}, \omega_{t_2-1}) \leq \frac{t_1}{T-1} \cdot \min(|U|, |L|)$ and *t*₁ ≤ *T* - 1 then
 - t*₁ = *t*₂ + 1;
 - t*₂ = *t*₁
- else
 - backward(*t*₂ - 1, 1);
 - t*₁ = 1;
 - t*₂ = 2
- step 5 go to step 2

The two extreme strategies are *shuffle* and *cautious*. Only if *shuffle* cannot go forward does it go backward up the tree. In particular, it solves all the stage *t* + 1, ..., *T* subproblems explicitly (within a tolerance) prior to passing cuts back to stage *t*. On the other hand, *cautious* never goes forward down the tree unless it cannot go backward, it will not go forward to stage *t* + 1 unless all cuts that would be passed back to stage *t* - 1 are redundant (more precisely, the expected stage *t* - 1 discrepancy must be sufficiently small). *fastpass* is a strategy "half way" between *shuffle* and *cautious*. *c-shuffle* is a strategy that is in between *shuffle* and *fastpass*, it is less hesitant to go backward up the tree than the former but more hesitant than the latter. *c-cautious* is similarly in between *cautious* and *fastpass*. As *c* increases both *c*-strategies become more like *fastpass*². As *c* approaches the tolerance, *toler*, from above, the two *c*-strategies more closely mimic their (non *c*) counterparts³.

²If *c* = ∞ then *c-shuffle* and *fastpass* are equivalent. Due to the use of discrepancies in *c-cautious* and absolute errors in *fastpass* for declaring individual subproblems "solved" these two strategies are slightly different.

³If *c* = *toler* then *cautious* and *c-cautious* are equivalent. Due to a technical difference in step 4 of *shuffle* and

The primary concern with *shuffle* is it may spend too long solving, for example, the stage $2, \dots, T$ subproblems with respect to a very poor first stage decision. The quality of the information passed up the tree is high, but too much computational effort may be spent computing the cuts. Similarly, *cautious* may spend too much effort reducing the stage $t - 1$ discrepancies when the stage t cuts do not yet give a good approximation of the expected costs to be incurred in stages $t + 1, \dots, T$; in this case, the cuts passed to stage $t - 1$ are of low quality. The more closely a cut resembles a support of the recourse function, the higher its quality. The “best” tree traversing strategy will properly balance the quality of the information passed back up the tree with the computational effort required to generate the cuts, and in general, we might expect this will depend on the problem being solved. The search for this balance motivates considering strategies which range between *shuffle* and *cautious*. *fastpass* is one such strategy; it provides faster transfer of information to all parts of the decision tree. The ϵ -strategies enable us to more fully investigate intermediate strategies. The ϵ -strategies, however, do not fully span the range of all possible tree traversing strategies between *shuffle* and *cautious*. One could, for example, consider a *dynamic ϵ -strategy*: one in which ϵ could change as the algorithm proceeds.

2.7.2 Finite Termination Theorem

We now sketch a finite convergence proof of the tree traversing strategies described in Section 2.7.1. When the stage T subproblems are solved and cuts are passed to the stage $T - 1$ subproblems the cuts are supports, in particular pieces, of the stage T piecewise linear convex recourse functions:

$$\sum_{\omega_T \in \Delta(\omega_{T-1})} p_T^{\omega_T} h_{T-1}(x_{T-1}, \omega_T).$$

Consider the following two cases.

- (i) There is at least one new cut passed to stage $T - 1$.
- (ii) All cuts (that would be) passed to stage $T - 1$ are redundant.

The first case can occur only a finite number of times as the stage T linear programs contain a finite number of bases. If the second case occurs, the stage $T - 1$ subproblems are then solved with respect to their descendants, and the cuts passed back to stage $T - 2$ will be supports of the stage $T - 1$ recourse functions. Inductive application of this argument gives us the following theorem

ca shuffle the two algorithms are identical only if $\epsilon = \text{toler} = 0$.

Theorem 2.4 *Suppose SLP-T has a finite optimal solution, all linear programs are solved finitely, and $\text{toler} = 0$. Then each of the five tree traversing strategies described in Section 2.7.1 has finite convergence to an optimal solution of SLP-T.*

Theorem 2.4 and its proof is much in the spirit of Wittrock [66]. Theorem 2.4 holds for any tree traversing strategy which has “sufficient access” to all parts of the tree.

2.7.3 Computational Results

Computational results for the four test problems are summarized in Tables 2.9–2.11. Table 2.9 simply restates the base case (*fastpass*) strategy’s results for convenient reference.

Name	<i>fastpass</i>		
	CPU (sec)	iter	# Subs
Moke3.9	55.4	10	10-59-99
Ybsf3.9	119.8	7	7-47-72
Moke4.45	221.4	10	10-64-274-459
Ybsf4.45	404.0	8	8-83-205-369

Table 2.9: Base Case Strategy *fastpass*

Tables 2.10 and 2.11 use “ $\times$ increase” with respect to *fastpass* as the performance measure. For example, in Table 2.10, “ $\times$ increase” is the ratio of the CPU time for a strategy (either *cautious* or *shuffle*) to the CPU time for *fastpass* on a particular test problem; if this ratio is less than 1.0, the strategy outperformed *fastpass* on that particular problem. By examining the # Subs columns in Tables 2.9 and 2.10 one can contrast the distribution of computing effort per stage for *fastpass*, *cautious*, and *shuffle*.

Name	<i>cautious</i>			<i>shuffle</i>		
	$\times$ increase	iter	# Subs	$\times$ increase	iter	# Subs
Moke3.9	0.92	7	19-82-72	1.08	7	7-51-135
Ybsf3.9	1.03	6	21-86-63	1.21	6	6-45-117
Moke4.45	1.06	8	27-140-449-369	2.10	4	4-40-296-852
Ybsf4.45	1.00	6	29-202-274-279	2.18	8	8-71-263-771

Table 2.10: *cautious* and *shuffle*

In Table 2.11 the “ $\times$ increase” column is the average of the CPU ratios for the four test problems. This value is displayed for some specific values of ϵ in the ϵ -*cautious* and ϵ -*shuffle* strategies.

ϵ -cautious		ϵ -shuffle	
ϵ	$\times$ increase	ϵ	$\times$ increase
0.0001	1.00	0.0001	1.64
0.0004	0.98	0.0004	1.52
0.0016	0.97	0.0016	1.32
0.0064	0.95	0.0064	1.16
0.0256	1.06	0.0256	1.08
0.1024	1.01	0.1024	0.96
0.4096	1.01	0.4096	1.00

Table 2.11: ϵ -cautious and ϵ -shuffle

The computational results indicate *fastpass*, *cautious*, and ϵ -*cautious* strategies perform comparably and outperform *shuffle* and ϵ -*shuffle*. It is only as ϵ becomes large (and hence ϵ -*shuffle* approaches *fastpass*) that ϵ -*shuffle* performs comparably. It is interesting to note how little the ϵ -*cautious* running times change as ϵ varies. Intuitively, an ϵ -*cautious* strategy is appealing in a multistage problem (especially with discounting) where there is a desire to avoid the computationally expensive later stages. This effect (if it exists) may require problems with more stages; Table 2.10 indicates there was not a significant difference between the three and four stage problems for the *cautious* (also ϵ -*cautious*) strategies. On the other hand, the *shuffle* (also ϵ -*shuffle*) strategy's performance is significantly worse on the four stage problems. As one might expect, the "extreme" strategies performance deteriorates when the advanced start procedure is removed. The average CPU ratio of *cautious* and *shuffle* to *fastpass* (also without advanced start) is 1.13 and 2.07, respectively.

2.8 Direct LP Optimizers

The multistage stochastic hydro scheduling problems used as test problems in this chapter are "simply" large-scale linear programs. Thus, general LP optimizers based on the simplex and/or interior point methods provide an alternate algorithmic approach. Table 2.12 summarizes the performance of the enhanced decomposition algorithm described in this chapter (i.e., the base case) and general LP optimizers on an enlarged set of test problems. These eight problems are based on the Mokelumne and Yuba-Bear-South Feather river basin models with 1, 9, 27, and 45 scenarios. We restate the base case running times for convenient reference and use the " $\times$ increase" measure with respect to the base case for three other LP solution strategies. The first such strategy is the primal dual

predictor-corrector interior point algorithm as implemented in IBM's Optimization Subroutine Library (Release 2); the second strategy uses the same interior point algorithm followed by the simplex method to generate an (optimal) extreme point solution. The final column in Table 2.12 shows the run-time results of the primal simplex method as implemented in CPLEX 2.0. The results show that on single scenario problems, the decomposition algorithm is outperformed by general LP optimizers. However, as the number of scenarios grows, the decomposition algorithm outperforms the general optimizers, and the larger the number of scenarios the better the relative performance of the enhanced decomposition algorithm.

Name	Base Case	Int. Pt.	Int. Pt. → Simplex	Simplex
	(sec.)	× increase		
Moke4.1	16.8	0.32	0.46	0.40
Moke4.9	85.0	1.00	2.41	6.31
Moke4.27	132.9	2.26	6.94	27.54
Moke4.45	221.4	3.68	11.37	47.18
Ybsf4.1	48.1	0.23	0.42	0.59
Ybsf4.9	177.3	1.12	2.94	16.04
Ybsf4.27	252.9	2.15	14.84	61.53
Ybsf4.45	404.0	N/A	N/A	100.20

Table 2.12: Comparison with Direct LP Optimizers

The base case decomposition strategy terminates when the relative error between upper and lower bounds is less than 10^{-4} . The same duality gap tolerance was used in both interior point solution strategies, and all tests were performed on a Hewlett Packard 9000/750 workstation. Due to memory limitations⁴ we were unable to solve the largest (Ybsf4.45) problem via the interior point strategies. Recall, in Section 1.3.1 we stated that the individual stochastic hydro scheduling problems are subproblems in a large nonlinear Dantzig-Wolfe decomposition algorithm. The Dantzig-Wolfe scheme requires extreme point solutions to the subproblems. Thus a solution for a hydro scheduling "subproblem" from a pure interior point algorithm is not sufficient, and this is why the corresponding time to generate an optimal extreme point solution (via the simplex method) from the interior point

⁴OSL's "dspace" array was allocated 64 Mb

solution is shown in Table 2.12. It is not difficult to verify that an optimal solution generated by the Benders decomposition scheme is extreme.

2.9 Summary

We have described a number of enhancements to the nested Benders decomposition algorithm for multistage stochastic linear programming. The performance of these enhancements was examined empirically on a collection of multistage stochastic hydroelectric scheduling problems. These “real world” problems are part of an ongoing research and development project at Pacific Gas and Electric Company; see Jacobs et al. [39] for a more detailed description of the project. The computational results indicated the single most important enhancement is a warm start technique which utilizes optimal basis information from previous subproblem solutions. We also described an advanced start procedure that generates preliminary cuts to help guide the early iterations of the decomposition algorithm; this enhancement provided additional speedup over the traditional algorithm. We found the multicut method due to Birge and Louveaux [10] also yielded computational savings over its single cut counterpart. Confirming earlier findings of Abrahamson [1], Wittrock [66] and Gassmann [31], we found that the *fastpass* tree traversing strategy performed well. However, a new class of ϵ -cautious tree traversing strategies produced comparable results to *fastpass*; further investigation of this class of strategies may be warranted for problems with many stages. Finally, we showed that taking advantage of the problem’s special structure through the enhanced decomposition algorithm provides a computationally attractive alternative to direct LP optimizers on medium to large size problems.

Chapter 3

Decomposition and Sampling-Based Algorithms

3.1 Introduction

The fundamental difficulty in implementing numerical solution techniques for stochastic programming lies in evaluating the recourse function and its (sub)gradient. In Chapter 2 we described an algorithm for models in which the number of scenarios is sufficiently small so the problems can be solved exactly. In many practical problems, however, the number of scenarios may be so large that exact solution methods are impractical. Recall in SLP-2 the recourse function has form $Eh(x_1, \xi) = \sum_{\omega \in \Omega} h(x_1, \xi(\omega))P\{\xi = \xi(\omega)\}$ where

$$\begin{aligned} h(x_1, \xi(\omega)) = & \text{minimize} && c_2^* x_2^* \\ & \text{subject to} && A_2^* x_2^* = b_2^* + B_1^* x_1 \\ & && x_2^* \geq 0 \end{aligned}$$

and $\xi(\omega) = (c_2^*, b_2^*, B_1^*, A_2^*)$. It has long been recognized that incorporating Monte Carlo sampling techniques within a decomposition algorithm might provide an attractive approach. The paper in which Dantzig and Madansky [18] first showed the dual form of SLP-2 had a structure ideal for the application of Dantzig-Wolfe decomposition, closed by remarking that the effect of sampling at each iteration of the algorithm would be “an interesting area for future consideration.” Dantzig and Glynn (1990) [16], Infanger (1992) [36], Pereira and Pinto (1991) [51], and Hight and Sen (1991) [34] have put forward decomposition-based algorithms that incorporate sampling at each iteration. In this chapter, we discuss the algorithms due to (i) Dantzig, Glynn, and Infanger and (ii) Pereira and Pinto. To this end, we begin by reviewing a variance reduction technique called importance sampling.

3.2 Importance Sampling

Let V denote a random d -vector with probability density function $f(v)$, and suppose $C : \mathbb{R}^d \rightarrow \mathbb{R}_+$. Our goal is to estimate $\alpha = EC(V) = \int C(v)f(v)dv$ by Monte Carlo sampling. Let $\bar{C}_n$ denote a sample mean of n independent and identically distributed (i.i.d.) observations of the random variable $C(V)$. It is well-known that $\text{var}[\bar{C}_n] = \text{var}[C(V)]/n$; thus, an attractive feature of Monte Carlo sampling is that the rate at which the error converges to zero is independent of the dimension of the integral. This is to be contrasted with “partitioning” methods for numerical integration; see Davis and Rabinowitz [20]. Nevertheless, the error’s proportionality constant, $\sqrt{\text{var}[C(V)]}$, can be large, and compensating for a large standard error by increasing the sample size n can be an expensive proposition. For example, to gain one additional significant figure in accuracy of the estimator, the sample size must be increased by a factor of 100. Importance sampling is a technique that attempts to reduce the proportionality constant by employing a change of probability measure. Let $\Gamma(v) \geq 0$ be a function which we will choose later in such a way to reduce the variance. Observe

$$\alpha = \int C(v)f(v)dv = \bar{\Gamma} \int \frac{C(v)}{\Gamma(v)} [f(v)\Gamma(v)/\bar{\Gamma}] dv, \quad \text{where } \bar{\Gamma} = \int \Gamma(v)f(v)dv.$$

Therefore, with $q(w)dw = [f(v)\Gamma(v)/\bar{\Gamma}] dv$, $q(w)$ may be regarded as the density function of a random vector W and

$$\alpha = \bar{\Gamma} E\rho(W) = \bar{\Gamma} \int \rho(w)q(w)dw, \quad \text{where } \rho(w) = \frac{C(w)}{\Gamma(w)}.$$

The key to a successful importance sampling procedure is the selection of the function Γ . If we redefine the point estimate $\bar{C}_n$ to be a sample mean of n i.i.d. observations of the random variable $\bar{\Gamma}\rho(W)$, then the variance term of interest is

$$\text{var}[\bar{\Gamma}\rho(W)] = \bar{\Gamma}^2 \int (\rho(w) - \alpha/\bar{\Gamma})^2 q(w)dw. \quad (3.1)$$

From (3.1) it is clear the “optimal” choice of Γ is $\Gamma(w) = C(w)\bar{\Gamma}/\alpha$ because it yields $\text{var}[\bar{C}_n] = 0$. Unfortunately, this choice of Γ requires the value of the very parameter we wish to estimate, namely α . Nevertheless, this result indicates Γ should be chosen so that it *approximates* C . Additionally, it is desirable that $\bar{\Gamma} = \int \Gamma(v)f(v)dv$ be an integral that can be calculated analytically (in an inexpensive fashion) and that random samples from the density function q can be generated easily. Intuitively,

a good approximation function would be one that alters the probability distribution in such a way that high cost (i.e., large $C(v)$) events are sampled more frequently.

3.3 Two-Stage Algorithm

In this section, we present an overview of an algorithm proposed by Dantzig and Glynn [16] and further developed by Infanger [36] for solving SLP-2. The reader is referred to these references for more detailed discussions of the algorithm. In Figure 3.1 we provide a “naive” Monte Carlo sampling version of the algorithm which is a natural extension of the L-shaped algorithm proposed by Van Slyke and Wets [61]. In the L-shaped algorithm all expectations are calculated exactly, and in the algorithm outlined in Figure 3.1 these expectations are replaced with sample means. The reader is referred to Chapter 1 of this paper for a description of the “universe” analog of the algorithm of Figure 3.1.

```

step 0  let  $k = 0$ ; initialize the set of cuts with  $\theta \geq -M$ ;

step 1  solve the master program and obtain  $(x_1^k, \theta^k)$ ;
         let  $\underline{z}^k = c_1 x_1^k + \theta^k$ ;

step 2  select a set of random samples  $\mathcal{S}^k$  according to  $p(\omega)$ ;
         do  $\omega \in \mathcal{S}^k$ 
             form RHS of sub( $\omega$ ):  $B_1(\omega)x_1^k + b_2(\omega)$ ;
             solve and obtain  $(x_2^k(\omega), \pi_2^k(\omega))$ ;
         enddo
         let  $\bar{z}^k = c_1 x_1^k + \frac{1}{|\mathcal{S}^k|} \sum_{\omega \in \mathcal{S}^k} c_2(\omega) x_2^k(\omega)$ ;

step 3  if  $\bar{z}^k - \underline{z}^k \leq 0$  then stop:  $x_1^k$  is the proposed solution;

step 4  augment the set of cuts with  $\theta - \tilde{G}_{k+1} x_1 \geq \tilde{g}_{k+1}$  where
          $\tilde{G}_{k+1} = \frac{1}{|\mathcal{S}^k|} \sum_{\omega \in \mathcal{S}^k} \pi_2^k(\omega) B_1(\omega)$  and  $\tilde{g}_{k+1} = \frac{1}{|\mathcal{S}^k|} \sum_{\omega \in \mathcal{S}^k} \pi_2^k(\omega) b_2(\omega)$ ;
         let  $k = k + 1$ ; goto step 1;

```

Figure 3.1: Benders Decomposition Algorithm for SLP-2

We use a subscript index on the cut gradient and intercept to allow for the possibility that the j^{th} cut is updated (perhaps via resampling) at the k^{th} iteration; we return to this possibility in

Chapter 5. We now describe an importance sampling variant of the algorithm in Figure 3.1. In many problems, only some of the components of $\xi = (c_2, b_2, B_1, A_2)$ are random. We denote the subvector of ξ which contains random elements by the random d -vector V . We assume the components of V are independent random variables with discrete (finite) distributions. Component-wise independence allows us to express $\Omega = \prod_{i=1}^d \Omega_i$ and $p(v^\omega) = \prod_{i=1}^d p_i(v_i^{\omega_i})$. The independence assumption is easily generalized to the case in which V may be expressed as a linear combination of a vector of component-wise independent random variables. Three quantities in the algorithm of Figure 3.1 are estimated by sample means: the second stage cost, the cut gradient, and the cut intercept. We begin by describing an importance sampling procedure for calculating the second stage cost given a particular first stage decision, x_1 . The second stage cost is simply $h(x_1, v^\omega)$. A *base case* scenario, τ , which satisfies $h(x_1, v^\tau) \leq h(x_1, v^\omega)$ for all $\omega \in \Omega$ is assumed to be known; it need not be a scenario which occurs with positive probability. In many problems such a scenario is easily determined: for example, in the stochastic hydroelectric scheduling problem, larger inflows yield lower costs provided water can be spilled without penalty. In order to estimate $Eh(x_1, V)$, it suffices to estimate $EC(V)$, where $C(V) = h(x_1, V) - h(x_1, v^\tau)$. For simplicity of notation, we suppress the dependence of $C(V)$ on x_1 , and discuss the estimation procedure at a generic k^{th} iteration (thus $x_1 = x_1^k$ is fixed).

In order to apply an importance sampling procedure as outline in Section 3.2, an approximation function is required. Dantzig et al. [17] proposed an *additive* approximation function which has produced excellent variance reduction for many problems; for example, the variance was reduced by a factor of 20,000 over “naive” sampling in a power expansion planning model [17]. Krishna [44] has developed a piecewise linear approximation function that yields superior results on some problems in which the additive function does not provide a good approximation. In this section, however, we restrict our attention to the additive approximation which has the following form:

$$\Gamma(V) = \sum_{i=1}^d \Gamma_i(V_i), \quad \text{where } \Gamma_i(V_i) = C(v_1^T, \dots, v_{i-1}^T, V_i, v_{i+1}^T, \dots, v_d^T).$$

Note Γ is defined so that it is identical to C on the margins; i.e., the functions are equal when all random elements are set to the base case except one. Interior points are approximated by the sum of the corresponding marginal values. The additive property of the approximation function permits

its expected value to be calculated as a sum of d one-dimensional integrals. In particular,

$$\bar{\Gamma} = \sum_{\omega \in \Omega} \sum_{i=1}^d \Gamma_i(v_i^{\omega_i}) p_1(v_1^{\omega_1}) \cdots p_d(v_d^{\omega_d}) = \sum_{i=1}^d \bar{\Gamma}_i$$

where $\bar{\Gamma}_i = \sum_{\omega_i \in \Omega_i} \Gamma_i(v_i^{\omega_i}) p_i(v_i^{\omega_i})$. It is assumed that $|\Omega_i|$ is sufficiently small so that $\bar{\Gamma}_i$ may be calculated exactly. To generate the approximation function and its expected value (at most) $\sum_{i=1}^d |\Omega_i| + 1$ linear programs must be solved each iteration.

Now

$$EC(V) = \sum_{\omega \in \Omega} C(v^\omega) p(v^\omega) = \sum_{i=1}^d \sum_{\omega \in \Omega} \rho(v^\omega) p(v^\omega) \Gamma_i(v_i^{\omega_i}), \quad (3.2)$$

where $\rho(v^\omega) = C(v^\omega)/\Gamma(v^\omega)$. In (3.2), $EC(V)$ is written as the sum of d expectations and we will employ importance sampling to estimate each expectation separately. The i^{th} term of (3.2) may be expressed:

$$\bar{\Gamma}_i \sum_{\omega \in \Omega} \rho(v^\omega) p_1(v_1^{\omega_1}) \cdots p_{i-1}(v_{i-1}^{\omega_{i-1}}) \left[\Gamma_i(v_i^{\omega_i}) / \bar{\Gamma}_i \right] p_i(v_i^{\omega_i}) p_{i+1}(v_{i+1}^{\omega_{i+1}}) \cdots p_d(v_d^{\omega_d}).$$

The estimate of $EC(V)$ is denoted $\bar{C}_n$ where

$$\bar{C}_n = \sum_{i=1}^d \frac{\bar{\Gamma}_i}{n_i} \sum_{\omega \in \mathcal{S}_i} \rho(v^\omega) \quad (3.3)$$

and $n_i = |\mathcal{S}_i|$. $\mathcal{S}_i$ may be regarded as a subset of the random samples, $\mathcal{S}^k$, selected in step 2 of the algorithm in Figure 3.1; the superscript k is suppressed for notational simplicity. In estimating the i^{th} term of (3.2), the samples for set $\mathcal{S}_i$ are selected as follows: V_j , $j \neq i$ is sampled according to its original distribution $p_j(v_j^{\omega_j})$, and V_i is sampled according to the importance mass function $[\Gamma_i(v_i^{\omega_i}) / \bar{\Gamma}_i] p_i(v_i^{\omega_i})$. The variance of the estimate (3.3) is

$$\text{var}(\bar{C}_n) = \sum_{i=1}^d \frac{\bar{\Gamma}_i^2 \sigma_i^2}{n_i}$$

where $\sigma_i^2 = \text{var}_i[\rho(V)]$ and var_i indicates that the variance is calculated with respect to the i^{th} importance mass function. At the k^{th} iteration of the algorithm a sample size “budget” of size $n = |\mathcal{S}^k|$ is allocated for the estimate $\bar{C}_n$. This budget must be partitioned into d components of size $n_i \geq 1$ where $\sum_{i=1}^d n_i = n$; ideally, this partitioning should be done so that $\text{var}(\bar{C}_n)$ is minimized. The solution to this optimization problem (ignoring integrality constraints) is to select

n_i proportional to $\bar{\Gamma}_i \sigma_i$. The quantity σ_i is, of course, unknown and thus in the absence of other information we select

$$n_i \approx \frac{\bar{\Gamma}_i}{\sum_{i=1}^d \bar{\Gamma}_i} \cdot n.$$

This budget allocation is reasonable, provided that the variance terms, σ_i^2 , $i = 1, \dots, d$, are of comparable magnitude. If it is believed that this is not the case, a preprocessing or dynamic updating scheme could be used to yield additional variance reduction.

As stated above, three quantities in the algorithm of Figure 3.1 are estimated by sample means: the second stage cost, the cut gradient, and the cut intercept. We have described the manner in which the second stage cost estimate is formed. In estimating the cut gradient and cut intercept, the same set of samples generated via the importance distribution for the second stage cost is used. In other words, the above calculations are simply repeated with new definitions of $C(V)$. In particular, for the cut gradient estimate

$$C(v^\omega) = \pi_2^\omega B_1^\omega - \pi_2^\tau B_1^\tau,$$

and for the cut intercept

$$C(v^\omega) = \pi_2^\omega b_2^\omega - \pi_2^\tau b_2^\tau.$$

Thus, while the importance sampling scheme is driven by the second stage cost function, the same distribution is used to calculate cut gradient and intercept. An alternate approach would be to generate a different importance sampling distribution for cut estimation. Such an approach could yield further variance reduction with respect to the cut estimators, but additional computation would be required as more subproblems must be solved each iteration. This computational trade-off and the type of approximation functions that would work well for cut estimation have yet to be explored. Infanger [36] has implemented the algorithm described in this section and has reported impressive computational results on several very large two-stage stochastic programs. Discussion of the following topics is deferred to Chapters 4 and 5: (i) the algorithm's stopping criterion of step 3, (ii) rules for selecting the samples sizes $|S^k|$, and (iii) confidence interval generation for the objective function value of the proposed solution.

3.4 Multistage Algorithm

Pereira and Pinto [51] have proposed a sampling and decomposition-based algorithm for multistage stochastic linear programming with recourse. The class of multistage problems that the algorithm is designed to address have interstage independent stochastic parameters and a “manageable” number of descendants per node in the scenario tree. Pereira and Pinto [51] have applied the algorithm to a 10-stage stochastic hydroelectric scheduling model with an inflow sequence that has two realizations per stage and thus a total of $2^9 = 512$ inflow scenarios. The sequence of inflows in their test problem is not temporally correlated, but in an earlier paper Pereira [52] states that autoregressive inflow sequences can be incorporated. In Section 3.5, we provide the required modifications for generating valid cuts when the stochastic right-hand side vectors are linearly dependent.

The class of SLP-T model under consideration have a small number of stochastic branches per scenario tree node; however, if the number of stages T is even moderately large a full forward pass through the tree may be computationally impractical. The basic idea behind the algorithm is to compute upper bound estimates by “sampling paths” through the scenario tree on a forward pass, and to compute exact cuts on a backward pass. Figures 3.2 and 3.3 are designed to illustrate this concept.

Figure 3.2: Forward pass - from left to right

A sample path, $\omega = (\omega_1, \omega_2, \dots, \omega_T)$, specifies exactly one node per stage in the scenario tree and has the property that the nodes identify a path from the (unique) stage 1 node to a stage T leaf node. Nonanticipativity constraints are satisfied on a forward pass along a sample path: The first stage problem with only previous cut information regarding the future. The first stage decision

Figure 3.3: Backward pass – from right to left

is then to passed to the right-hand-side of a randomly selected second stage subproblem, $\text{sub}(\omega_2)$, and the process is then repeated from stage 2 to 3. A forward pass along a sample path simulates operation of the system given the cuts collected in previous iterations. A new cut is then appended to the set of cuts at each stage on a backward pass along each sample path. Infanger [35] has proposed a multistage algorithm that may be viewed as an extension of this approach that includes the ideas of the two-stage algorithm described in Section 3.3. In particular, cuts are estimated by sampling and importance sampling is utilized for both cut and sample path generation. Here, we will restrict the discussion to the simpler case described above in which cuts are computed exactly and upper bounds are estimated by (naive) sample paths.

Recall, the dual of a stage t subproblem under scenario ω_t may be written:

$$\begin{aligned}
 & \underset{\pi_t^{\omega_t}, \alpha_t^{\omega_t}}{\text{maximize}} && \pi_t^{\omega_t} \left(b_t^{\omega_t} + B_{t-1}^{\omega_t} x_{t-1}^{a(\omega_t)} \right) + \alpha_t^{\omega_t} \tilde{g}_t^{\omega_t} \\
 & \text{subject to} && \pi_t^{\omega_t} A_t^{\omega_t} - \alpha_t^{\omega_t} \vec{G}_t^{\omega_t} \leq c_t^{\omega_t} \\
 & && e^T \alpha_t^{\omega_t} = 1 \\
 & && \alpha_t^{\omega_t} \geq 0.
 \end{aligned} \tag{3.4}$$

If $t = T$, the cut gradient matrix and cut intercept vector are absent. In this and the subsequent section, we require use of both the matrix of cut gradients, denoted $\vec{G}_t^{\omega_t}$, and a particular cut gradient, denoted $G_t^{\omega_t}$. We similarly distinguish between the vector of cut intercepts, $\tilde{g}_t^{\omega_t}$ and a scalar cut intercept $g_t^{\omega_t}$. The cut gradient and intercept are found via

$$G_t^{\omega_t} = \sum_{\omega_{t+1} \in \Delta(\omega_t)} p_{t+1}^{\omega_{t+1}|\omega_t} \pi_{t+1}^{\omega_{t+1}} B_t^{\omega_{t+1}} \tag{3.5}$$

and

$$g_t^{\omega_t} = \sum_{\omega_{t+1} \in \Delta(\omega_t)} p_{t+1}^{\omega_{t+1}|\omega_t} \pi_{t+1}^{\omega_{t+1}} b_{t+1}^{\omega_{t+1}} + \sum_{\omega_{t+1} \in \Delta(\omega_t)} p_{t+1}^{\omega_{t+1}|\omega_t} \alpha_{t+1}^{\omega_{t+1}} g_{t+1}^{\omega_{t+1}} \quad (3.6)$$

Under the assumption that the stochastic parameters are interstage independent, the cuts computed for any stage t subproblem are valid for all other subproblems on that stage; thus, it is not necessary to index the cut gradient and intercept by ω_t . Recall, we discussed the validity of cut sharing and some of its practical implications for the case of interstage independent parameters in Section 1.4.3. The algorithm for SLP-T is summarized in Figure 3.4. As in the previous section we defer discussion of stopping rules and confidence interval generation to Chapters 4 and 5.

step 0 let $k = 0$;
 initialize the set of stage t cuts with $\theta_t \geq -M_t, t = 1, \dots, T-1$;

step 1 solve the stage 1 master program and obtain (x_1^k, θ_1^k) ;
 let $\underline{z}^k = c_1 x_1^k + \theta_1^k$;

step 2 select a set of random sample paths $\mathcal{S}^k$ according to $p_T(\omega_T)$;
 do $\omega \in \mathcal{S}^k$
 do $t = 2$ to T
 form RHS of sub(ω_t): $B_{t-1}(\omega_t)x_{t-1}^k(\omega_{t-1}) + b_t(\omega_t)$;
 solve and obtain $x_t^k(\omega_t)$;
 enddo
 enddo
 let $\bar{z}^k = c_1 x_1^k + \frac{1}{|\mathcal{S}^k|} \sum_{\omega \in \mathcal{S}^k} \sum_{t=2}^T c_t(\omega_t) x_t^k(\omega_t)$;

step 3 if $\bar{z}^k - \underline{z}^k \leq 0$ then stop: x_1^k is the proposed solution;

step 4 do $t = T-1$ downto 1
 do $\omega \in \mathcal{S}^k$
 do $\omega_{t+1} \in \Delta(\omega_t)$
 form RHS of sub(ω_{t+1}): $B_t(\omega_{t+1})x_t^k(\omega_t) + b_{t+1}(\omega_{t+1})$;
 solve and obtain $\pi_{t+1}(\omega_{t+1})$ and $\alpha_{t+1}(\omega_{t+1})$;
 enddo
 augment the set of stage t cuts with $\theta_t - G_t x_t \geq g_t$
 enddo
 enddo

step 5 let $k = k + 1$; goto step 1;

Figure 3.4: Benders Decomposition Algorithm for SLP-T

3.5 Interstage Dependence and Cut Sharing

In Section 1.4.3, “cut sharing” in the case of interstage independence of the stochastic parameters was described. The term cut sharing was introduced to mean that a cut generated for a particular scenario subproblem is valid for the other scenario subproblems on the same stage. The ability to share cuts is critical in the multistage setting when paths are sampled as described in the previous section particularly when we do not have the memory capacity to store cuts separately for each scenario, and even if we had this capacity at hand, the frequency with which any particular node in the scenario tree is revisited will be quite low (e.g., zero) in a sampling oriented algorithm. In this section, we show that the ability to share cuts can be extended to certain interstage dependency structures of the stochastic right-hand side vectors.

In order to describe this extension, additional structure on the SLP-T formulation given in Section 1.2 is required, in particular, we assume that we may express

$$\Omega_t = \Sigma_t \times \cdots \times \Sigma_t \text{ for } t = 2, \dots, T$$

where Σ_t denotes the set of “history independent” scenarios for stage t . A stage t scenario, $\omega_t \in \Omega_t$, may be written $\omega_t = (\sigma_2, \dots, \sigma_t)$ where $\sigma_i \in \Sigma_i$.

We begin by assuming a simple linear interstage dependence structure of the right-hand side vectors. More specifically, we assume an autoregressive lag-one model

$$b_t = R_{t-1}b_{t-1} + \eta_t, \text{ for } t = 2, \dots, T, \quad (3.7)$$

where η_t , $t = 2, \dots, T$ are independent random vectors and the matrix R_{t-1} is deterministic. The other stochastic parameters, namely c_t , A_t , and B_{t-1} , are assumed to exhibit interstage independence. In order to illustrate the underlying idea (and define additional notation), we begin with the simple case of $T = 3$. We focus on cut calculations for second stage problems as this is the only nontrivial case for the three-stage problem. The second stage cut gradient and intercept are given by

$$G_2^{\omega_2} = \sum_{\omega_3 \in \Delta(\omega_2)} p_3^{\omega_3 | \omega_2} \pi_3^{\omega_3} B_3^{\omega_3} \quad (3.8)$$

and

$$g_2^{\omega_2} = \sum_{\omega_3 \in \Delta(\omega_2)} p_3^{\omega_3 | \omega_2} \pi_3^{\omega_3} b_3^{\omega_3} \quad (3.9)$$

As c_3 and A_3 are independent of the second stage parameters, the collection of dual variables $\{\pi_3^{\omega_3} : \omega_3 \in \Delta(\omega_2)\}$ are feasible for the descendants of any second stage scenario. Thus, by Lemma 2.2 these dual variables will generate a valid cut for any second stage subproblem, and we now show that these valid cuts may be generated and recalled in a compact fashion. Due to the dual feasibility structure, we will label the dual variables with a σ_3 index. In this framework, the conditional probability mass function $p_3^{\omega_3 | \omega_2} = p_3^{\sigma_3}$ does not depend on ω_2 . Furthermore, as B_3 is independent of the second stage random parameters the cut gradient formula (3.8) is valid for all $\omega_2 \in \Omega_2$, and the ω_2 index on G_2 may be dropped. The cut intercept formula, however, involves b_3 which contains interstage dependence according to (3.7). Upon substitution of the lag one model into (3.9) it is clear that

$$g_2^{\omega_2} = g_2^{\text{ind}} + g_2^{\text{dep}}(\omega_2) \quad (3.10)$$

where

$$g_2^{\text{ind}} = \sum_{\sigma_3 \in \Sigma_3} p_3^{\sigma_3} \pi_3^{\sigma_3} \eta_3^{\sigma_3} \quad (3.11)$$

and

$$g_2^{\text{dep}}(\omega_2) = \underbrace{\left[\sum_{\sigma_3 \in \Sigma_3} p_3^{\sigma_3} \pi_3^{\sigma_3} \right]}_{\pi_3} R_2 b_2^{\omega_2}. \quad (3.12)$$

In (3.10), the second stage cut intercept has been expressed as the sum of an ω_2 -independent (ind) term and an ω_2 -dependent (dep) term. The scenario dependent term given by (3.12) has a simple structure. Construction of $g_2^{\text{dep}}(\omega_2)$ for a particular second stage scenario requires knowledge of the second stage right-hand-side realization, $b_2^{\omega_2}$, and the expected value of the third stage dual variables, π_3 . Thus, in a three-stage model the following information is stored for each cut: (i) cut gradient, G_2 , (ii) scenario independent cut intercept term, g_2^{ind} , and (iii) expected dual vector, π_3 . Given a second stage scenario, ω_2 , valid cuts may then be formed for $\text{sub}(\omega_2)$ from this information by computing the scenario dependent cut intercept (3.12) and then computing the cut intercept via (3.10).

The three stage model does not reveal all of the complexities associated with the lag one model for a general multistage problem. In a T -stage model, the analysis for the three stage case presented above is valid for cuts computed for stage $T - 1$. However, cut intercepts for a general stage t are given by (3.6), and this formula indicates that the stage t cut intercepts require the stage $t + 1$ cut intercepts which in turn require the stage $t + 2$ cut intercepts, etc. Computationally, it is clearly preferable to avoid the recursive calculation of these respective intercept terms through the exponentially growing scenario tree. In particular, we seek a "compact" representation of the cut intercept in the multistage problem with the lag one autoregressive dependence model of (3.7).

Observe that the dual feasible regions of a stage t subproblem (3.4) are unaffected by the interstage dependence of the right-hand side vectors. This is an important observation because dual feasibility ensures that valid cuts can be generated; again, see Lemma 2.2. This is to be contrasted with the case in which A_t , c_t , or B_t contain interstage dependencies. We now turn to the problem of finding a compact representation for the cut intercepts; to do this, some additional notation is required. The stage t subproblem is assumed to have m_t rows (excluding cuts) and L_t cuts. We define $\mathcal{P}_t$ to be an $L_{t-1} \times m_t$ matrix whose rows contain the expected value of the structural constraint dual variables, $\bar{\pi}_t = E_{\sigma_t} \pi^{\sigma_t}$. Similarly, $\mathcal{A}_t$ is defined as the $L_{t-1} \times L_t$ matrix whose rows contain the expected value of the cut dual variables, $\bar{\alpha}_t = E_{\sigma_t} \alpha^{\sigma_t}$. As can be seen from (3.6), these dual variables are used in the cut computation for stage $t - 1$. The i^{th} row of $\mathcal{P}_t$ and $\mathcal{A}_t$ contain the expected value of the dual variables used to generate the i^{th} stage $t - 1$ cut. In Theorem 3.1 we show

$$g_t^{\omega_t} = g_t^{ind} + g_t^{dep}(\omega_t) \quad (3.13)$$

where

$$g_t^{ind} = E_{\sigma_{t+1}} [\pi_{t+1}^{\sigma_{t+1}} + \alpha_{t+1}^{\sigma_{t+1}} D_{t+1} R_{t+1}] \eta_{t+1}^{\sigma_{t+1}} + \bar{\alpha}_{t+1} \bar{g}_{t+1}^{ind} \quad (3.14)$$

and

$$g_t^{dep}(\omega_t) = [\bar{\pi}_{t+1} + \bar{\alpha}_{t+1} D_{t+1}] R_t b_t^{\omega_t}, \quad (3.15)$$

where the matrix D_t is defined recursively

$$D_t = [\mathcal{P}_{t+1} + \mathcal{A}_{t+1} D_{t+1}] R_t, \quad D_T = 0. \quad (3.16)$$

Theorem 3.1 Assume the lag-one model (3.7) with the remaining stochastic parameters exhibiting interstage independence. The cut intercepts for stage t , $t = 2, \dots, T-1$, are given by (3.13), (3.14), and (3.15) where D_t is defined by (3.16) with $D_T = 0$ and $\bar{g}_T^{ind} = 0$.

Proof

We proceed by induction. The base case is $t = T - 1$ and the expression for g_{T-1} is found by the method described for the three-stage case above in which $D_T = 0$. The inductive hypothesis is then (3.13), (3.14) and (3.15) and we will verify the same expressions with t decremented by one. It will be convenient to adopt the vector analog of (3.15):

$$\bar{g}_t^{dep}(\omega_t) = [\mathcal{P}_{t+1} + \mathcal{A}_{t+1} D_{t+1}] R_t b_t^{\omega_t}. \quad (3.17)$$

A stage $t - 1$ intercept for scenario ω_{t-1} is defined as

$$g_{t-1}^{\omega_{t-1}} = \sum_{\omega_t \in \Delta(\omega_{t-1})} p_t^{\omega_t | \omega_{t-1}} \pi_t^{\omega_t} b_t^{\omega_t} + \sum_{\omega_t \in \Delta(\omega_{t-1})} p_t^{\omega_t | \omega_{t-1}} \alpha_t^{\omega_t} \bar{g}_t^{\omega_t}. \quad (3.18)$$

Substitution of the lag-one model (3.7) into the first term on the right hand side of (3.18) yields

$$\pi_t R_{t-1} b_{t-1}^{\omega_{t-1}} + E_{\sigma_t} \pi_t^{\sigma_t} \eta_t^{\sigma_t}. \quad (3.19)$$

Substitution of the inductive hypothesis (3.13), (3.15) and the lag-one model (3.7) into the second term on the right hand side of (3.18) yields

$$\bar{\alpha}_t \bar{g}_t^{ind} + \bar{\alpha}_t [\mathcal{P}_{t+1} + \mathcal{A}_{t+1} D_{t+1}] R_t R_{t-1} b_{t-1}^{\omega_{t-1}} + E_{\sigma_t} \alpha_t^{\sigma_t} [\mathcal{P}_{t+1} + \mathcal{A}_{t+1} D_{t+1}] R_t \eta_t^{\sigma_t} \quad (3.20)$$

Using the definition of D_t from (3.16) and partitioning (3.19) and (3.20) into the scenario dependent and scenario independent parts yields the desired result. ■

We can generalize the result of Theorem 3.1 to more complex interstage dependency structures. For example, suppose

$$b_t = \sum_{i=1}^{t-1} R_i^t b_i + \eta_t, \text{ for } t = 2, \dots, T, \quad (3.21)$$

where η_t , $t = 2, \dots, T$ are independent random vectors and R_i^t are deterministic matrices. In this case, a simple change to the dependent cut intercept term is all that is required.

Corollary 3.2 *Assume the general additive lag model (3.21) with the remaining stochastic parameters exhibiting interstage independence. The cut intercepts for stage t , $t = 2, \dots, T-1$, are given by (3.13), (3.14), and*

$$g_t^{dep}(\omega_t) = [\bar{\pi}_{t+1} + \bar{\alpha}_{t+1} D_{t+1}] \sum_{i=1}^t R_i^{t+1} b_i^{\omega_t}, \quad (3.22)$$

where D_t is defined by (3.16) with $D_T = 0$ and $\bar{g}_T^{ind} = 0$.

Proof

The proof is virtually identical to that of Theorem 3.1. ■

Equations (3.13), (3.14), and (3.22) yield a compact representation of the stage t cut intercepts. In SLP-T with a general additive lag model (3.21) the following information is stored for each stage t cut: (i) cut gradient, G_t , (ii) the scenario independent cut intercept term, g_t^{ind} , and (iii) the “cumulative” expected dual vector $\bar{\pi}_{t+1} + \bar{\alpha}_{t+1} D_{t+1}$. Given a particular stage t scenario ω_t , valid cuts may be formed for $\text{sub}(\omega_t)$ from this information by computing the scenario dependent cut intercept term (3.22) and then computing the cut intercept via (3.13). This generalization of cut sharing from the interstage independent case to a special class of interstage dependence permits the algorithm of Section 3.4 to address a richer class of multistage stochastic linear programs.

3.6 Summary

In many stochastic programming models, the number of scenarios is so large that direct application of Benders decomposition algorithm is not computationally practical. It has long been recognized (Dantzig and Madansky (1961) [18]) that incorporating Monte Carlo sampling within a decomposition scheme might provide an attractive approach for problems with many scenarios. Dantzig and Glynn (1990) [16] and Infanger (1992) [36] have utilized a variance reduction technique called importance sampling within a Benders decomposition algorithm for SLP-2. Pereira and Pinto (1991) [51] have developed a Benders decomposition algorithm that uses Monte Carlo sampling for SLP-T in which each node of the scenario tree has a “manageable” number of descendents. In this chapter, we have given an overview of importance sampling and outlined two algorithms. A method by which the multistage algorithm can be extended to certain types of interstage dependence of the right-hand-side vectors was also presented. Encouraging computational results on practical large-scale

planning problems have been reported for these algorithms, but stopping rules designed to ensure the quality of the proposed solution have yet to be developed. The primary purpose of the material reviewed in this chapter is to provide motivation for the stopping rule theory which is presented in Chapter 4.

Chapter 4

Stopping Rules: Theory

4.1 Introduction

In Chapter 3, two stochastic programming algorithms which utilize Monte Carlo sampling techniques *within* an optimization framework were presented. These algorithms have the attractive feature of vastly reducing the need to weaken a model by preselecting a small set of “important” scenarios. A critical component of these algorithms is a stopping criterion that ensures the quality of the recommended solution.

A host of questions arise when one replaces the exact upper and lower objective function bounds generated by Benders algorithm with estimates formed from sample means. How should an “optimal” solution be characterized and how should the sampling procedure proceed so as to ensure an appropriately defined notion of convergence? The purpose of the development detailed in this chapter is to provide a framework in which these issues may be addressed. The stopping rules are comprised of two components: (i) a criterion for terminating the algorithm and (ii) a rule for selecting the sample sizes. The main results of this chapter provide stopping rules that guarantee asymptotic validity of confidence interval statements as the interval width shrinks and the sample sizes grow. In Chapter 5, the application of this theory to the two algorithms reviewed in Chapter 3 is discussed and preliminary empirical coverage results are presented.

4.2 Stopping Rules: Problem Statement

Consider the following general optimization problem:

$$\begin{array}{ll} \text{minimize } z(x) \\ \text{subject to } x \in \mathcal{X}. \end{array} \tag{4.1}$$

Suppose we have at hand an algorithm that at each iteration k selects a *sample size* n_k , produces a feasible decision x_k that generates estimates for upper and lower bounds on the optimal objective function value, z^* . We denote the upper and lower bound estimates $U_k(n_k)$ and $L_k(n_k)$, respectively, and assume at each iteration k the *difference random variable*, $D_k(n_k) = U_k(n_k) - L_k(n_k)$, satisfies the central limit theorem (CLT):

$$\sqrt{n_k}(D_k(n_k) - \mu_k) \Rightarrow N(0, \sigma_k^2) \text{ as } n_k \rightarrow \infty, \quad \text{where } \sigma_k > 0, \quad (4.2)$$

$$D_1(n_1), D_2(n_2), \dots \text{ are independent} \quad (4.3)$$

and where $\Rightarrow$ denotes convergence in distribution. Throughout this chapter, $N(\mu, \sigma^2)$ will denote a normal random variable with mean μ and variance σ^2 . We will typically suppress the sample size parameter when referring to the upper, lower, and difference random variables. The true upper and lower bounds at iteration k are denoted u_k and l_k and satisfy $U_k \Rightarrow u_k$ and $L_k \Rightarrow l_k$ as $n_k \rightarrow \infty$, where $l_k \leq z^* \leq u_k$, and (necessarily) $\mu_k = u_k - l_k$. Under these assumptions, U_k and L_k do not necessarily satisfy respective central limit theorems; they could, for example, have a common (perfectly correlated) random variable which cancels in the difference. It may often be the case, however, that U_k and L_k are independent and do satisfy respective CLT's; in this case, the CLT for D_k follows as a consequence.

Little structure is placed on the convergence properties of the algorithm; in particular, we assume only:

$$\text{there exists a subsequence } \{\mu_{k_j}\}_{j=1}^{\infty} \text{ such that } \mu_{k_j} \rightarrow 0. \quad (4.4)$$

Let x_T denote the feasible decision which the algorithm “recommends” at the random *stopping iteration* T ; we assume this decision corresponds to the upper bound, i.e., $u_T = z(x_T)$. In addition, we assume at the stopping iteration, the algorithm permits re-evaluation of the difference random variable through independent resampling. In this setting two questions naturally arise: When should we stop the algorithm and what can we say about the quality of the proposed solution, x_T , when we stop?

We say the algorithm *stops correctly* if $z(x_T) \leq z^* + \epsilon$ where ϵ is a positive *confidence interval width*. Note the algorithm is ensured to have stopped correctly if $\mu_T \leq \epsilon$. Using this observation

and the CLT (4.2) we develop “stopping rules” which prescribe rules for selecting the sample sizes, n_k , and terminating the algorithm such that we can make a statement regarding the probability the algorithm stops correctly. At the heart of the procedures we develop is the idea that the sample sizes must increase as the algorithm proceeds. While our immediate motivation lies in developing stopping rules for two stochastic programming algorithms which incorporate Monte Carlo sampling, the theory may also be applicable to other optimization algorithms that use simulation techniques.

In Section 4.3 we assume the difference random variables at each iteration are independent and normally distributed with respective variance terms proportional to n_k^{-1} . Under this assumption, we show it suffices to increase the sample size at a rate proportional to $\log k$ to guarantee the probability of stopping correctly satisfies a prescribed confidence level. This result holds for any positive confidence interval width, but the assumption of normal differences is quite restrictive. The purpose of the development in Section 4.3 is to illustrate the underlying ideas in a setting where the derivations are quite simple. In Section 4.4, we assume the difference random variables satisfy the CLT (4.2). In this case, we provide an $\mathcal{O}(\log^2 k)$ sample-size formula and state conditions under which one can recover the results of Section 4.3 in the form of an asymptotic validity result; i.e., we show the desired coverage probability holds as the confidence interval width shrinks and the corresponding sample sizes grow. Moreover, we show with respect to required computational effort, the $\mathcal{O}(\log^2 k)$ sample-size formula is preferable to the $\mathcal{O}(\log k)$ formula.

As the development unfolds, we also indicate the main results hold under more general assumptions regarding the difference random variables. For example, the same results may be obtained when the parameters μ_k and σ_k depend on the random *history* of the algorithm in previous iterations. We first verify this in the simplistic setting of Section 4.3 and subsequently in Section 4.5 prove the asymptotic validity result under an appropriately defined history dependent CLT hypothesis. In Section 4.6, a more conservative (but still $\mathcal{O}(\log^2 k)$) sample-size formula leads to a *distribution-free* confidence interval which is valid for all positive confidence interval widths and not just asymptotically as the sample sizes grow large. Finally, in Section 4.7 we describe a method to incorporate sample variance estimators in the confidence interval statements.

4.3 Stopping Rules: Normal Differences

Example 4.1 This example shows that a fixed sample size can lead to unattractive results. Consider the following stopping rule: At each iteration k , let the sample size be $n_k = n$ and terminate at the first iteration in which $D_k \leq 0$. With this stopping rule we construct an example satisfying $\mu_k \rightarrow 0$ such that the probability of stopping correctly is arbitrarily close to 0. In particular, suppose $u_k = \mu_k = \mu > \epsilon$ for $k = 1, \dots, K$, and $\mu_k = 0$ for $k \geq K + 1$, and $\sigma_k = \sigma > 0$ for all k . The algorithm will not stop correctly if it stops at any iteration $T \leq K$. By choosing K sufficiently large we can make $P\{D_1 > 0, \dots, D_K > 0\} = [P\{D_1 > 0\}]^K$ arbitrarily small, and hence, the probability we stop correctly is arbitrarily close to 0. ■

The stopping rules we propose will overcome the difficulty which we have illustrated in Example 4.1. The framework for the development in this section is as follows: At the k^{th} iteration of the algorithm, we choose a sample size n_k and then observe the random variable:

$$D_k \sim N(\mu_k, \sigma_k^2/n_k), \quad \text{where } \sigma_k > 0 \quad (4.5)$$

and where $D_1, D_2, \dots$ are independent. The underlying algorithm is assumed to satisfy the convergence criterion (4.4). We select a *stopping tolerance* ϵ' satisfying $0 < \epsilon' < \epsilon$, where ϵ is the confidence interval width and terminate the algorithm on the first iteration in which the difference random variable drops below the stopping tolerance; i.e.,

$$T = \inf_{k \geq 1} \{k : D_k \leq \epsilon'\}. \quad (4.6)$$

Under these assumptions Theorem 4.2 provides a sample-size formula under which the algorithm stops with probability one and the probability the algorithm stops correctly satisfies a minimum prescribed confidence level of $1 - \alpha$; e.g., 0.95. We first require the following lemma regarding bounds on the tail of a normal distribution.

Lemma 4.1 *If $z > 0$ then*

$$P\{N(0, 1) \geq z\} \leq \frac{1}{\sqrt{2\pi}} \frac{1}{z} e^{-z^2/2}.$$

Proof

See Feller Chapter VII §1 [27]. ■

Theorem 4.2 Assume (4.5), (4.6), and define

$$n_k = \left\lceil \left(\frac{\sigma_k}{\epsilon - \epsilon'} \right)^2 (\beta + 2p \ln k) \right\rceil, \quad (4.7)$$

where $0 \leq \epsilon' < \epsilon$, $\beta = \max \{2 \ln [\zeta(p)/(\sqrt{2\pi} \alpha)], 1\}$, $0 < \alpha < 1$, and $\zeta(p) = \sum_{k=1}^{\infty} k^{-p}$, $p > 1$.

Then

$$P\{\mu_T \leq \epsilon\} \geq 1 - \alpha. \quad (\text{coverage result})$$

If, in addition, (4.3), (4.4), and $\epsilon' > 0$ hold then

$$P\{T < \infty\} = 1. \quad (\text{finite stopping time})$$

Proof

Let $\mathcal{K}_m = \{k : k \leq m, \mu_k \leq \epsilon'\}$, and observe

$$P\{T < \infty\} \geq 1 - \prod_{k=1}^m P\{D_k > \epsilon'\} \geq 1 - (1/2)^{|\mathcal{K}_m|}.$$

The finite stopping time result follows as $\epsilon' > 0$ and (4.4) imply $|\mathcal{K}_m| \rightarrow \infty$ as $m \rightarrow \infty$.

Consider

$$\begin{aligned} P\{\mu_T \geq D_T - \epsilon' + \epsilon\} &= \sum_{k=1}^{\infty} P\{D_1 > \epsilon', \dots, D_{k-1} > \epsilon', D_k \leq \epsilon', D_k \leq \mu_k - (\epsilon - \epsilon')\} \\ &\leq \sum_{k=1}^{\infty} P\{D_k \leq \mu_k - (\epsilon - \epsilon')\}. \end{aligned}$$

Now $\sum_{k=1}^{\infty} [\alpha/\zeta(p)] k^{-p} = \alpha$; thus, it suffices to show

$$P\{D_k \leq \mu_k - (\epsilon - \epsilon')\} \leq \frac{\alpha}{\zeta(p)} k^{-p}.$$

To this end consider

$$\begin{aligned} P\{D_k \leq \mu_k - (\epsilon - \epsilon')\} &= P\left\{\frac{D_k - \mu_k}{\sigma_k/\sqrt{n_k}} \leq -(\epsilon - \epsilon')\sqrt{n_k}/\sigma_k\right\} \\ &= P\{N(0, 1) \geq (\epsilon - \epsilon')\sqrt{n_k}/\sigma_k\} \\ &\leq P\{N(0, 1) \geq (\beta + 2p \ln k)^{1/2}\} \end{aligned}$$

Based on the bound of the tail of normal distribution from Lemma 4.1:

$$P\{D_k \leq \mu_k - (\epsilon - \epsilon')\} \leq \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(\beta + 2p \ln k)\right],$$

since $\beta \geq 1$. Hence it suffices to show

$$\frac{1}{\sqrt{2\pi}} \exp \left[-\frac{1}{2} (\beta + 2p \ln k) \right] \leq \frac{\alpha}{\zeta(p)} k^{-p}.$$

This inequality is immediate from the definition of β . We have established

$$P\{\mu_T \geq D_T - \epsilon' + \epsilon\} \leq \alpha,$$

and the coverage result follows as $D_T - \epsilon' \leq 0$. ■

Remark 4.1 The coverage result of Theorem 4.2 states that $[0, \epsilon]$ is a $(1 - \alpha) \cdot 100\%$ confidence interval for μ_T . In terms of the original optimization problem (4.1) this implies:

$$P\{z(x_T) \leq z^* + \epsilon\} \geq 1 - \alpha.$$

In other words, if we were to solve 100 problems we would expect, on average, at least $100(1 - \alpha)$ of the proposed solutions, x_T , to have an objective function value within ϵ of the optimal objective function value.

Remark 4.2 Tables of values of the Riemann-Zeta function, $\zeta(p)$, may be found in Dwight [21]; $\zeta(p)$ is also an intrinsic function in some mathematical software packages such as *Mathematica* [67]. We will return to the topics of choosing values for the parameter p and the total number of samples required in Section 4.4; an abbreviated table of $\zeta(p)$ values is given there.

Remark 4.3 The term “ k^{th} iteration” should be interpreted liberally; a better phrase is “ k^{th} stopping cycle.” A stopping cycle could consist of a number of algorithm iterations in which a fixed sample size is used *plus* one resampling iteration in which the sample-size formula (4.7) is applied. The fixed sample-size phase of a stopping cycle is terminated when a “pretest” is passed; e.g., a negative difference is observed. The sample-size formula and stopping criterion described above are then applied to a realization of the difference random variable generated from an independent sample. Implementation techniques such as these are important to help minimize computational burden.

Remark 4.4 The primary emphasis of our analysis is on controlling, in an *a priori* sense, the quality of the proposed solution x_T . Any number of heuristic stopping rules are possible which

can generate a solution $x_{T'}$, and through resampling of the associated difference random variable we can generate a $(1 - \delta) \cdot 100\%$ confidence interval of the form $[z^*, z^* + D_{T'} + w_\delta \sigma_{T'}/\sqrt{n}]$ for $z(x_{T'})$ where n denotes the resampling sample size and w_δ satisfies $P\{N(0,1) \leq w_\delta\} = 1 - \delta$. Since we know $\mu_{T'} \geq 0$ we can form a more conservative confidence interval by replacing $D_{T'}$ with $D_{T'}^+ \equiv \max\{D_{T'}, 0\}$. In any case, the disadvantage of results based on heuristic stopping rules is that we cannot control the interval width, $D_{T'}^+ + w_\delta \sigma_{T'}/\sqrt{n}$. The primary purpose of Theorem 4.2 (and the subsequent generalizations we present in this chapter) is to provide such *a priori* control on the confidence interval width.

Remark 4.5 In the same vein as the previous remark, the specific value of z^* or $z(x_{T'})$ is of secondary interest relative to controlling the quality of the proposed solution. However, if U_T and L_T are each normally distributed (or more generally satisfy respective CLTs) then we can also develop confidence intervals for these quantities through the resampling procedure.

Remark 4.6 The proof of the coverage result ignores the algorithm's convergence structure. The sample-size formula is designed so that if the algorithm does not converge (in the sense of (4.4)) then the stopping criterion (4.6) will not be satisfied with probability $1 - \alpha$. This is why the very weak convergence structure (4.4) is possible. Since many algorithms will have a stronger underlying convergence structure, the coverage result of Theorem 4.2 may be regarded as conservative.

We can obtain the results as Theorem 4.2 in a more general setting with respect to the assumptions on the interaction of the algorithm with the upper and lower bound estimates. In the development above, we assume $\{\mu_k, \sigma_k\}_{k=1}^\infty$ is simply a sequence of constants. In many algorithmic settings, however, the mean and variance at the k^{th} iteration are random variables; they depend on the history of the algorithm through iteration $k - 1$ which we denote $\mathcal{H}_{k-1}$. For example, in a Benders decomposition algorithm for two stage stochastic programs in which cuts are obtained by sampling, the sequence of master program decisions depend on which scenarios were (randomly) selected to compute cuts in previous iterations. In this example, it is clear μ_k , σ_k , and D_k are random variables that are “sample-path” dependent. A realization of $\mathcal{H}_k$ may be thought of as the information necessary to reconstruct exactly the steps of the algorithm through iteration k . In this more general setting we will assume that conditioned on the history random variable, the mean and

variance are constants and the difference random variables satisfy

$$P \left\{ \frac{D_k - \mu_k}{\sigma_k / \sqrt{n_k}} \leq y \mid \mathcal{H}_{k-1} \right\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^y e^{-u^2/2} du. \quad (4.8)$$

In addition, for the finite stopping result we need to generalize the notion of independence of the difference random variables to the history setting. Under termination criterion (4.6) it is clear $P\{T > m\} = P\{D_1 > \epsilon', D_2 > \epsilon', \dots, D_m > \epsilon'\}$. We assume the following natural generalization of hypothesis (4.3):

$$P\{D_1 > \epsilon', D_2 > \epsilon', \dots, D_m > \epsilon'\} = \int \prod_{k=1}^m P\{D_k > \epsilon' \mid \mathcal{H}_{k-1}\} dP_{\mathcal{H}_{m-1}}. \quad (4.9)$$

With regard to the convergence structure of the algorithm we assume there exists a subsequence of $\{\mu_k\}_{k=1}^\infty$ which converges to zero with probability one; i.e.,

$$P\{\omega : \mathcal{H}_\infty(\omega) \text{ implies } \exists \{\mu_{k_j}(\omega)\}_{j=1}^\infty \text{ such that } \mu_{k_j}(\omega) \rightarrow 0\} = 1. \quad (4.10)$$

Theorem 4.3 Assume (4.6), (4.8), and define n_k and the corresponding parameters as in Theorem 4.2. Then

$$P\{\mu_T \leq \epsilon\} \geq 1 - \alpha. \quad (\text{coverage result})$$

If in addition, (4.9), (4.10), and $\epsilon' > 0$ hold then

$$P\{T < \infty\} = 1. \quad (\text{finite stopping time})$$

Proof

By hypothesis (4.9),

$$P\{T < \infty\} \geq 1 - \int \prod_{k=1}^m P\{D_k > \epsilon' \mid \mathcal{H}_{k-1}\} dP_{\mathcal{H}_{m-1}}.$$

The argument of Theorem 4.2 coupled with hypothesis (4.10) implies $\prod_{k=1}^m P\{D_k > \epsilon' \mid \mathcal{H}_{k-1}\} \rightarrow 0$ w.p.1 as $m \rightarrow \infty$. The finite stopping time result then follows from the bounded convergence theorem.

To verify the coverage result, we begin by mimicking the proof of Theorem 4.2 and see it suffices to show

$$P\{D_k \leq \mu_k - (\epsilon - \epsilon')\} \leq \frac{\alpha}{\zeta(p)} k^{-p}.$$

To this end we observe

$$\begin{aligned}
 P\{D_k \leq \mu_k - (\epsilon - \epsilon')\} &= P\left\{\frac{D_k - \mu_k}{\sigma_k/\sqrt{n_k}} \leq -(\epsilon - \epsilon')\sqrt{n_k}/\sigma_k\right\} \\
 &= \int P\left\{\frac{D_k - \mu_k}{\sigma_k/\sqrt{n_k}} \leq -(\epsilon - \epsilon')\sqrt{n_k}/\sigma_k \mid \mathcal{H}_{k-1}\right\} dP_{\mathcal{H}_{k-1}} \\
 &\leq \int P\{N(0, 1) \geq (\beta + 2p \ln k)^{1/2}\} dP_{\mathcal{H}_{k-1}} \\
 &= P\{N(0, 1) \geq (\beta + 2p \ln k)^{1/2}\}.
 \end{aligned}$$

The rest of the proof is identical to that of Theorem 4.2. ■

4.4 Stopping Rules: CLT Differences

In Section 4.3 we verified that if the difference random variables are normally distributed and the sample size at each iteration is increased according to (4.7) then the probability the algorithm stops correctly satisfies a minimum prescribed confidence level. This confidence interval statement holds for any positive interval width. In this section we relax the “normally distributed” assumption and assume instead the difference random variables satisfy the central limit theorem; i.e., assumption (4.2). Under this weaker hypothesis, we provide conditions on the difference random variables under which one may recover the confidence interval result of the previous section in the limit as the interval width shrinks and the sample sizes grow. In particular, we show

$$\lim_{\epsilon \downarrow \epsilon'} P\{z(x_{T(\epsilon)}) \leq z^* + \epsilon\} \geq 1 - \alpha, \quad (4.11)$$

where the stopping time, $T = T(\epsilon)$, is again the first iteration in which the observed difference drops below the stopping tolerance ϵ' ; i.e., stopping criterion (4.6). We also recover the finite termination result of Theorem 4.2 for all positive confidence interval widths under the assumption of independence (4.3). Note in the asymptotic validity statement (4.11), the stopping tolerance ϵ' is fixed while the confidence interval with ϵ shrinks to width ϵ' ; thus we parameterize the stopping iteration T by the variable interval width ϵ .

We began Section 4.3 with an example designed to demonstrate the need to increase the sample size as the algorithm proceeds. However, Example 4.1 (with K fixed) has the property that for a sufficiently large sample size (i.e., sufficiently small ϵ) the probability of stopping incorrectly is

arbitrarily small. Thus Example 4.1 was only appropriate for the previous section because the confidence interval statement was made for all positive interval widths, ϵ . In Example 4.2 we again use an identical sample size for all iterations, and we construct a problem in which the probability of stopping correctly is zero in the limit as interval width shrinks to zero.

Example 4.2 Let $D_k \sim N(k^{-1}, n_k^{-1})$ be independent and define $m = \lceil \epsilon^{-1} \rceil - 1$. If the algorithm terminates prior to iteration $m + 1$ it has stopped incorrectly. Thus, with the stopping tolerance, ϵ' set to zero we have

$$\begin{aligned} P\{\mu_{T(\epsilon)} \leq \epsilon\} &\leq P\{D_1 > 0, \dots, D_m > 0\} \\ &= \prod_{k=1}^m P\{N(0, 1) < k^{-1}\sqrt{n_k}\} \end{aligned}$$

Let $\epsilon = Cn_k^{-1/2} > 0$ so the sample size has the same value for all iterations and observe:

$$\begin{aligned} P\{\mu_{T(\epsilon)} \leq \epsilon\} &\leq \prod_{k=1}^m P\{N(0, 1) < k^{-1}(m+1)C'\} \\ &\leq \prod_{k=\lceil m/2 \rceil}^m P\{N(0, 1) < (m+1)C/[\lceil m/2 \rceil]\} \\ &\leq [P\{N(0, 1) < 3C\}]^{\lceil m/2 \rceil} \end{aligned}$$

Thus it is clear $P\{\mu_{T(\epsilon)} \leq \epsilon\} \rightarrow 0$ as $\epsilon \downarrow 0$. ■

Example 4.2 demonstrates that a fixed sample size, no matter how large, may lead to "confidence" intervals with quite unsatisfactory coverage properties. The key idea, once again, is that we must increase the sample size as the algorithm proceeds; Theorem 4.4 provides stopping rules under which the confidence intervals are asymptotically valid.

Theorem 4.4 Assume (4.2), (4.6), and

$$\sup_{k \geq 1} E \exp \left[\gamma \left(\frac{D_k - \mu_k}{\sigma_k / \sqrt{n_k}} \right) \right] \text{ is bounded for } |\gamma| \leq \gamma_0. \quad (4.12)$$

Let $0 \leq \epsilon' < \epsilon$, $\beta' = \max \{2 \ln [\phi(p)/(\sqrt{2\pi} \alpha)], 1\}$, where $\phi(p) = \sum_{k=1}^{\infty} k^{-p \ln k}$, $p > (2\gamma_0^2)^{-1}$ and $0 < \alpha < 1$.

If

$$n_k = \left\lceil \left(\frac{\sigma_k}{\epsilon - \epsilon'} \right)^2 (\beta' + 2p \ln^2 k) \right\rceil \quad (4.13)$$

then

$$\lim_{\epsilon \downarrow \epsilon'} P \{ \mu_{T(\epsilon)} \leq \epsilon \} \geq 1 - \alpha. \quad (\text{asymptotic validity})$$

If, in addition, (4.3), (4.4), and $\epsilon' > 0$ hold then

$$P \{ T(\epsilon) < \infty \} = 1. \quad (\text{finite stopping time})$$

Proof

Let $Z_k = \sigma_k^{-1} \sqrt{n_k} (D_k - \mu_k)$. For any $\epsilon' > 0$, condition (4.12) and $n_k/\sigma_k^2 \rightarrow \infty$ as $k \rightarrow \infty$ ensures there exists K such that $k \geq K$ implies $E|D_k - \mu_k| < \epsilon'/4$. Recall Markov's inequality (see, e.g., Loève §9 [47]):

$$\text{If } Y \geq 0 \text{ and } y > 0 \text{ then } P \{ Y \geq y \} \leq EY/y.$$

Applying this inequality we find

$$P \{ |D_k - \mu_k| \geq \epsilon'/2 \} \leq E|D_k - \mu_k|/(\epsilon'/2).$$

Define $\mathcal{K}_m = \{k : k \leq m, k \geq K, \mu_k \leq \epsilon'/2\}$, and observe

$$\prod_{k=1}^m P \{ D_k > \epsilon' \} \leq \prod_{k \in \mathcal{K}_m} P \{ D_k - \mu_k > \epsilon' - \mu_k \} \leq \prod_{k \in \mathcal{K}_m} P \{ |D_k - \mu_k| \geq \epsilon'/2 \} \leq (1/2)^{|\mathcal{K}_m|}.$$

The finite stopping result follows as (4.4) implies $|\mathcal{K}_m| \rightarrow \infty$ as $m \rightarrow \infty$.

To prove the asymptotic validity result we begin as in Theorem 4.2 and see it suffices to show

$$\lim_{\epsilon \downarrow \epsilon'} \sum_{k=1}^{\infty} P \left\{ Z_k \leq -(\beta' + 2p \ln^2 k)^{1/2} \right\} \leq \alpha. \quad (4.14)$$

Next we show the order of the limit and sum on the left hand side of (4.14) may be exchanged with equality by employing the dominated convergence theorem. From Markov's inequality with $Y = e^{-\gamma Z_k}$, $\gamma > 0$ we have

$$P \left\{ Z_k \leq -(\beta' + 2p \ln^2 k)^{1/2} \right\} \leq \exp \left[-\gamma(\beta' + 2p \ln^2 k)^{1/2} \right] E e^{-\gamma Z_k}. \quad (4.15)$$

With $\gamma = \gamma_0$, the right hand side of (4.15) is bounded above by $\{\sup_{k \geq 1} E e^{-\gamma_0 Z_k}\} k^{-\gamma_0 \sqrt{2p}}$. As $\gamma_0 \sqrt{2p} > 1$, the order of the limit and sum may be exchanged with equality and it remains to show

$$\sum_{k=1}^{\infty} \lim_{\epsilon \downarrow \epsilon'} P \left\{ Z_k \leq -(\beta' + 2p \ln^2 k)^{1/2} \right\} \leq \alpha.$$

We apply the central limit theorem (4.2) and then complete the proof in a fashion analogous to that of Theorem 4.2. ■

In a more general distributional setting on the difference random variables we have recovered the coverage result of Theorem 4.2 in an asymptotic sense. We may choose the positive stopping tolerance ϵ' arbitrarily small, and thus the asymptotic validity result of Theorem 4.4 may be interpreted as follows: For sufficiently small choices of ϵ , $[z^*, z^* + \epsilon]$ is an *approximate* $(1 - \alpha) \cdot 100\%$ confidence interval for $z(x_T)$.

We now address two issues with respect to the hypotheses of Theorem 4.4. First we provide conditions under which the bounded expectation assumption (4.12) holds and then we examine the function $\phi(p)$. Verification of (4.12) is straightforward in the setting in which the difference random variable may be expressed as a sample mean of independent and identically distributed random variables (i.i.d.r.v.'s), due to well-known properties of moment generating functions. In particular, it suffices to verify the underlying random variables from which sample means are formed have moment generating functions for $|\gamma| \leq \gamma_0$ where γ_0 does not depend on k . This claim is established in Proposition 4.6; first we state the following Lemma 4.5 from elementary calculus.

Lemma 4.5 *If $a_n \rightarrow a$ then $\lim_{n \rightarrow \infty} \left(1 + \frac{a_n}{n}\right)^n = e^a$.*

In the next Proposition we will use the following notation: For each k , $D_{k1}, D_{k2}, \dots$ denote i.i.d.r.v.'s, $D_k = \frac{1}{n_k} \sum_{i=1}^{n_k} D_{ki}$, $ED_{k1} = \mu_k$, and $E(D_{k1} - \mu_k)^2 = \sigma_k^2$ where $0 < \sigma_k^2 < \infty$.

Proposition 4.6 *If $\sup_{k \geq 1} E \exp \left[\gamma \left(\frac{D_{k1} - \mu_k}{\sigma_k} \right) \right] < \infty$ then $\sup_{k \geq 1} E \exp \left[\gamma \left(\frac{D_k - \mu_k}{\sigma_k / \sqrt{n_k}} \right) \right]$ is bounded.*

Proof

Let $Y_{ki} = \sigma_k^{-1} (D_{ki} - \mu_k)$. Then

$$\begin{aligned} E \exp \left[\gamma \left(\frac{D_k - \mu_k}{\sigma_k / \sqrt{n_k}} \right) \right] &= E \exp \left[\frac{\gamma}{\sqrt{n_k}} \sum_{i=1}^{n_k} Y_{ki} \right] = \left(E \exp \left[\frac{\gamma}{\sqrt{n_k}} Y_{k1} \right] \right)^{n_k} \\ &= \left(1 + \frac{1}{n_k} \left[\frac{\gamma^2}{2} + \frac{1}{\sqrt{n_k}} \sum_{m=3}^{\infty} \frac{\gamma^m}{m!} E Y_{k1}^m n_k^{-(m-3)/2} \right] \right)^{n_k} \\ &\leq \left(1 + \frac{1}{n_k} \left[\frac{\gamma^2}{2} + \frac{1}{\sqrt{n_k}} \sup_{k \geq 1} E e^{\gamma Y_{k1}} \right] \right)^{n_k}. \end{aligned}$$

The result follows by Lemma 4.5 and the assumption $\sup_{k \geq 1} E e^{\gamma Y_{k1}} < \infty$. ■

We now seek to characterize the function $\phi(p) = \sum_{k=1}^{\infty} k^{-p \ln k}$. By comparing terms of the $\phi(p)$ series with those of $\zeta(2) = \sum_{k=1}^{\infty} k^{-2}$ for sufficiently large k we may observe $\phi(p) < \infty$ if $p > 0$. Moreover, $\phi(p)$ is decreasing in p , and $\phi(p) > 1$. Proposition 4.7 provides bounds on $\phi(p)$ that facilitate numerical function evaluations and further characterizes both $\phi(p)$ and $\zeta(p)$.

Proposition 4.7

(i) Let $v_N = \sqrt{2p} [\ln N - (2p)^{-1}]$ and $K_p = \sqrt{\pi/p} \exp [(4p)^{-1}]$, $p > 0$. Then

$$K_p P\{N(0, 1) \geq v_{N+1}\} \leq \phi(p) - \sum_{k=1}^N k^{-p \ln k} \leq K_p P\{N(0, 1) \geq v_N\}$$

and

$$K_p (P\{N(0, 1) \geq v_N\} - P\{N(0, 1) \geq v_{N+1}\}) \leq N^{-p \ln N}.$$

(ii) $\ln \zeta(p)$ and $\ln \phi(p)$ are convex functions on $(1, \infty)$ and $(0, \infty)$, respectively.

Proof

Based on the inequalities

$$(k+1)^{-p \ln(k+1)} \leq \int_k^{k+1} u^{-p \ln u} du \leq k^{-p \ln k}$$

we find

$$\int_{N+1}^{\infty} u^{-p \ln u} du \leq \phi(p) - \sum_{k=1}^N k^{-p \ln k} \leq \int_N^{\infty} u^{-p \ln u} du.$$

A change of variables yields

$$\int_y^{\infty} u^{-p \ln u} du = (2p)^{-1/2} \exp [(4p)^{-1}] \int_{v_y}^{\infty} e^{-u^2/2} du = K_p P\{N(0, 1) \geq v_y\},$$

where $v_y = \sqrt{2p} [\ln y - (2p)^{-1}]$. This establishes the first part of (i), and the error bound follows from

$$\int_N^{\infty} u^{-p \ln u} du - \int_{N+1}^{\infty} u^{-p \ln u} du \leq N^{-p \ln N}.$$

For result (ii) we shall require the infinite series version of Hölder's inequality:

$$\sum_{k=1}^{\infty} |a_k b_k| \leq \left[\sum_{k=1}^{\infty} |a_k|^r \right]^{1/r} \left[\sum_{k=1}^{\infty} |b_k|^s \right]^{1/s}$$

where $r, s > 1$ and $1/r + 1/s = 1$. For the Riemann-Zeta function we must show

$$\ln \zeta(\lambda p_1 + (1 - \lambda)p_2) \leq \lambda \ln \zeta(p_1) + (1 - \lambda) \ln \zeta(p_2).$$

Using the definition of $\zeta(p)$ we see it suffices to show:

$$\sum_{k=1}^{\infty} (k^{-p_1})^{\lambda} (k^{-p_2})^{1-\lambda} \leq \left[\sum_{k=1}^{\infty} k^{-p_1} \right]^{\lambda} \left[\sum_{k=1}^{\infty} k^{-p_2} \right]^{1-\lambda}.$$

This inequality follows from Hölder's inequality. The proof for $\ln \phi(p)$ is virtually identical. ■

Selected values of the function $\phi(p)$ (generated using part (i) of Proposition 4.7) and the Riemann-Zeta function $\zeta(p)$ (generated in *Mathematica* [67]) are given in Table 4.1.

p	$\zeta(p)$	p	$\zeta(p)$	p	$\phi(p)$	p	$\phi(p)$
1.0	∞	1.6	2.285 766	0.0	∞	0.6	3.348 318
1.005	200.577 58	1.7	2.054 289	0.02	3 363 108.7	0.7	2.930 462
1.01	100.577 94	1.8	1.882 230	0.04	4 590.335 3	0.8	2.633 438
1.05	20.580 844	1.9	1.749 746	0.06	466.315 53	0.9	2.411 111
1.08	13.083 010	2.0	1.644 934	0.08	142.241 35	1.0	2.238 181
1.1	10.584 448	2.5	1.341 487	0.1	67.918 029	1.5	1.741 034
1.2	5.591 582	3.0	1.202 057	0.2	13.547 263	2.0	1.501 477
1.3	3.931 949	4.0	1.082 323	0.3	7.215 758	4.0	1.154 839
1.4	3.105 547	5.0	1.036 928	0.4	5.048 588	5.0	1.092 976
1.5	2.612 375	15.0	1.000 031	0.5	3.980 611	20.0	1.000 067

Table 4.1: $\zeta(p)$ and $\phi(p)$ values

Under the assumption of normal differences, Theorem 4.2 provides an $\mathcal{O}(\log k)$ sample-size formula (4.7) for which the desired coverage result for stopping correctly can be ensured. Under a central limit theorem assumption, Theorem 4.4 provides a moment generating function condition and an $\mathcal{O}(\log^2 k)$ sample-size formula (4.7) under which the desired coverage result is valid in an asymptotic sense. At first it may appear disappointing we did not verify the asymptotic coverage result using sample-size formula (4.7). In fact, we can recover the coverage result under sample-size formula (4.7) provided we make a stronger assumption than (4.12) on the difference random variables; in particular it suffices to assume

$$\sup_{k \geq 1} E \exp \left[\gamma \left(\frac{D_k - \mu_k}{\sigma_k / \sqrt{n_k}} \right)^2 \right] \text{ is bounded for } |\gamma| \leq \gamma_0.$$

This assumption is more difficult to verify than the moment generating function assumption; i.e., we do not have analogs of Proposition 4.6. More importantly, we will show with respect to required

computational effort, the $\mathcal{O}(\log^2 k)$ sample-size formula is preferable to the $\mathcal{O}(\log k)$ formula for practical problems. Quantifying this statement is closely tied to the issue of selecting good values for the parameter p . Given, *a priori*, the number of required iterations, N , an optimal choice of p , denoted p^* , would minimize $\sum_{k=1}^N n_k$. However, N is clearly unknown; in fact, in our setting it is a random variable. Rough estimates for N (or EN) may be available given previous computational experience for certain classes of problems. For this discussion we will assume $\sigma_k^2/(\epsilon - \epsilon')^2$ is the same for each iteration; this would be the case if the confidence width and tolerance level are selected proportional to σ_k (see subsequent Remark 4.8). In this case, minimizing the total number of samples or “work” over N iterations corresponds to minimizing

$$W_\zeta(p) = N \max \left\{ 2 \ln \left[\zeta(p) / (\sqrt{2\pi} \alpha) \right], 1 \right\} + 2p \left(\sum_{k=1}^N \ln k \right)$$

for sample-size formula (4.7) or

$$W_\phi(p) = N \max \left\{ 2 \ln \left[\phi(p) / (\sqrt{2\pi} \alpha) \right], 1 \right\} + 2p \left(\sum_{k=1}^N \ln^2 k \right)$$

for sample-size formula (4.13). From Proposition 4.7, part (ii) it is clear $W_\zeta(p)$ and $W_\phi(p)$ are convex functions on $(1, \infty)$ and $(0, \infty)$, respectively. Table 4.2 displays the results of minimizing the respective work functions for various choices of N ; the confidence level parameter α takes on the value 0.05 in all calculations.

N	formula (4.7) $\mathcal{O}(\log k)$			formula (4.13) $\mathcal{O}(\log^2 k)$		
	p^*	$\zeta(p^*)$	$W_\zeta(p^*)$	p^*	$\phi(p^*)$	$W_\phi(p^*)$
10	1.5	2.612 375	106	0.4	5.048 588	96
25	1.35	3.459 237	323	0.25	9.379 868	291
50	1.29	4.046 193	731	0.19	14.865 183	660
10^2	1.24	4.761 075	1 630	0.155	22.270 678	1 473
10^3	1.15	7.254 695	21 715	0.09	94.647 997	19 720
10^4	1.11	9.676 075	269 211	0.065	325.046 04	246 153
10^5	1.09	11.694 841	3 199 021	0.050	1 175.999 4	2 944 556

Table 4.2: Optimal Choices of p

For each value of N , Table 4.2 contains the value, p^* , which minimizes $W_\zeta(p)$ and $W_\phi(p)$, respectively (to within 0.1%); the corresponding values of $\zeta(p^*)$ and $\phi(p^*)$ are also given for reference. The surprising observation to be made from Table 4.2 is that the $\mathcal{O}(\log^2 k)$ sample-size formula

consistently outperforms the $\mathcal{O}(\log k)$ formula. The reason for this result becomes more clear when the range of permissible values of p for each formula is considered. Sample-size formula (4.7) requires $p > 1$, and this leads to a multiplier greater than 2 on the $\sum_{k=1}^N \ln k$ term of the work function. The corresponding term, $\sum_{k=1}^N \ln^2 k$, of the work function for sample-size formula (4.13) can have any positive coefficient. The smaller coefficients for the $\mathcal{O}(\log^2 k)$ formula yields superior results even for values of N dramatically larger than of practical interest. Another measure of interest is to study ranges of N for which $W_\phi(p) < W_\zeta(q)$ where p and q are *fixed* and set to “reasonable” values. As an example we select the values of p and q from the $N = 100$ row of Table 4.2: $p = 0.155$ and $q = 1.24$. In this case, $W_\phi(0.155) < W_\zeta(1.24)$ for $12 \leq N \leq 1405$, and for $1 \leq N \leq 11$ $W_\phi(0.155) - W_\zeta(1.24) < 6.0$. For the $N = 1000$ row $W_\phi(0.09)$ is smaller for $52 \leq N \leq 46507$, and for $1 \leq N \leq 51$ $W_\phi(0.09) - W_\zeta(1.15) < 24.0$.

When applying Theorem 4.4 one may not be able to use the above recommended values of p^* due to the $p > (2\gamma_o^2)^{-1}$ condition. In many practical problems, however, the underlying random variables may be bounded, and under the i.i.d. hypotheses of Proposition 4.6 one can then choose any $p > 0$.

4.5 Stopping Rules: History Dependent CLT Differences

The results of Theorem 4.4 can be recovered in a history dependent setting analogous to that of Section 4.3. We assume that conditioned on the history random variable the mean and variance are constants and the difference random variables satisfy

$$P \left\{ \frac{D_k - \mu_k}{\sigma_k / \sqrt{n_k}} \leq y \mid \mathcal{H}_{k-1} \right\} \rightarrow \frac{1}{\sqrt{2\pi}} \int_{-\infty}^y e^{-u^2/2} du. \quad (4.16)$$

As in Section 4.3, we assume the difference random variables satisfy the generalized notion of inter-iteration independence defined by (4.9). With regard to the convergence structure of the algorithm we again assume there exists a subsequence of $\{\mu_k\}_{k=1}^\infty$ which converges to zero with probability one; i.e., hypothesis (4.10). We will require a conditional version of Markov’s inequality stated in Lemma 4.8.

Lemma 4.8 *Let $Y \geq 0$ and $y > 0$ then*

$$P\{Y \geq y | X\} \leq E(Y | X)/y.$$

Proof

$$E(Y | X) \geq \int_{\{Y \geq y\}} Y dP_{Y|X} \geq y \int_{\{Y \geq y\}} dP_{Y|X} = yP\{Y \geq y | X\}. \blacksquare$$

Theorem 4.9 *Assume (4.6), (4.12), (4.16), and define n_k and the corresponding parameters as in Theorem 4.4.*

Then

$$\lim_{\epsilon \downarrow \epsilon'} P\{\mu_{T(\epsilon)} \leq \epsilon\} \geq 1 - \alpha. \quad (\text{asymptotic validity})$$

If, in addition, (4.9), (4.10), $\epsilon' > 0$, and

$$E \exp \left[\gamma \left(\frac{D_k - \mu_k}{\sigma_k / \sqrt{n_k}} \right) \middle| \mathcal{H}_{k-1} \right] < M \text{ for } |\gamma| \leq \gamma_o, \quad (4.17)$$

hold where M and γ_o do not depend on the iteration k , the history, $\mathcal{H}_k$, or the confidence interval width, $\epsilon > \epsilon'$ (which determines n_k) then

$$P\{T(\epsilon) < \infty\} = 1. \quad (\text{finite stopping time})$$

Proof

By hypothesis (4.9)

$$P\{T < \infty\} \geq 1 - \int \prod_{k=1}^m P\{D_k > \epsilon' | \mathcal{H}_{k-1}\} dP_{\mathcal{H}_{m-1}}.$$

We know from the bounded convergence theorem it suffices to show

$$\lim_{m \rightarrow \infty} \prod_{k=1}^m P\{D_k > \epsilon' | \mathcal{H}_{k-1}\} = 0 \text{ w.p.1.}$$

This result follows from a direct generalization of the proof of Theorem 4.4 using the conditional version of Markov's inequality (Lemma 4.8) and the “converges w.p.1” hypothesis (4.10).

Let $Z_k = \sigma_k^{-1} \sqrt{n_k} (D_k - \mu_k)$. To prove the asymptotic validity result we begin as in Theorem 4.2 and see it suffices to show

$$\lim_{\epsilon \downarrow \epsilon'} \sum_{k=1}^{\infty} P\{Z_k \leq -(\beta' + 2p \ln^2 k)^{1/2}\} \leq \alpha.$$

From the proof of Theorem 4.4 we know hypothesis (4.12) permits us to exchange the order of limit and summation with equality. Now observe

$$\begin{aligned}
 & \sum_{k=1}^{\infty} \lim_{\epsilon \downarrow \epsilon'} \int P \{ Z_k \leq -(\beta' + 2p \ln^2 k)^{1/2} \mid \mathcal{H}_{k-1} \} dP_{\mathcal{H}_{k-1}} \\
 &= \sum_{k=1}^{\infty} \int \lim_{\epsilon \downarrow \epsilon'} P \{ Z_k \leq -(\beta' + 2p \ln^2 k)^{1/2} \mid \mathcal{H}_{k-1} \} dP_{\mathcal{H}_{k-1}} \\
 &= \sum_{k=1}^{\infty} \int P \{ N(0, 1) \leq -(\beta' + 2p \ln^2 k)^{1/2} \} dP_{\mathcal{H}_{k-1}} \\
 &= \sum_{k=1}^{\infty} P \{ N(0, 1) \leq -(\beta' + 2p \ln^2 k)^{1/2} \}.
 \end{aligned}$$

The remainder of the proof is analogous to that of Theorem 4.2. ■

Remark 4.7 The proof of the finite stopping time result requires the moment generating function condition (4.17). The asymptotic validity result holds under the weaker moment generating function hypothesis (4.12).

Remark 4.8 The asymptotic validity results of Theorems 4.4 and 4.9 both still hold when we select ϵ and ϵ' proportional to σ_k ; i.e., a so-called “relative precision” confidence interval. In particular with $\epsilon = \hat{\epsilon} \sigma_k$ and $\epsilon' = \hat{\epsilon}' \sigma_k$ the only required modification is to replace $\lim_{\epsilon \downarrow \epsilon'}$ with $\lim_{\hat{\epsilon} \downarrow \hat{\epsilon}'}$. In this setting, the finite stopping time result may also be recovered without modification; however, it is more natural to redefine the stopping time T in formula (4.6) by replacing ϵ' with $\hat{\epsilon}'$. Under this new definition we must additionally assume the variance terms, σ_k^2 , are bounded in order to recover the finite stopping time result. As we will see in Chapter 5, this is not an unreasonable assumption for the applications we will consider. When referring to relative precision confidence intervals in the remainder of the development we will use the less cumbersome ϵ and ϵ' notation (instead of $\hat{\epsilon}$ and $\hat{\epsilon}'$) whenever possible. For example, the sample-size formula (4.13) in this setting is $n_k = \lceil (\epsilon - \epsilon')^{-2} (\beta' + 2p \ln^2 k) \rceil$.

4.6 Stopping Rules: Distribution-Free Differences

When estimating a parameter associated with a performance measure of a simulation, say $\mu = EX$, one typically utilizes limit theorems from which *approximate* confidence intervals are inferred. The

asymptotic coverage results of Theorems 4.4 and 4.9 are intended to be applied very much in this spirit. In many situations it is also possible to derive *exact* confidence intervals. The well-known Chebyshev inequality provides a distribution-free or exact confidence interval. The reason such confidence intervals are rarely used in practice is the bounds they provide are not usually sharp. Restated, the sample size required to achieve a given confidence interval width may be significantly larger than the sample size needed for an approximate confidence interval. Nevertheless, based on the framework we have developed thus far a distribution-free result is virtually at hand; the result is stated in Theorem 4.10.

Theorem 4.10 Assume (4.6), and

$$E \exp \left[\gamma \left(\frac{D_k - \mu_k}{\sigma_k / \sqrt{n_k}} \right) \right] < M_k \text{ for } |\gamma| \leq \gamma_o.$$

Let $0 \leq \epsilon' < \epsilon$, $\beta_k = (\gamma_o^{-1} \ln [M_k \zeta(p\gamma_o)/\alpha])^+$, $p > \gamma_o^{-1}$, and $0 < \alpha < 1$.

If

$$n_k = \left\lceil \left(\frac{\sigma_k}{\epsilon - \epsilon'} \right)^2 (\beta_k + p \ln k)^2 \right\rceil \quad (4.18)$$

then

$$P\{\mu_T \leq \epsilon\} \geq 1 - \alpha. \quad (\text{coverage result})$$

If, in addition, (4.4), $\epsilon' \geq 0$, and $\sup_{k \geq 1} M_k < \infty$ hold then

$$P\{T(\epsilon) < \infty\} = 1. \quad (\text{finite stopping time})$$

Proof

The proof of the finite stopping time result is identical to that of Theorem 4.4.

Let $Z_k = \sigma_k^{-1} \sqrt{n_k} (D_k - \mu_k)$ and observe

$$\begin{aligned} P\{\mu_T \geq \epsilon\} &\leq \sum_{k=1}^{\infty} P\{Z_k \leq -(\beta_k + p \ln k)\} \\ &= \sum_{k=1}^{\infty} P\{e^{-\gamma Z_k} \geq \exp[\gamma(\beta_k + p \ln k)]\} \\ &\leq \sum_{k=1}^{\infty} E e^{-\gamma Z_k} e^{-\gamma \beta_k} k^{-p\gamma} \end{aligned}$$

follows from Markov's inequality. With $\gamma = \gamma_0$ the desired result is immediate from the definition of β_k . ■

While sample-size formula (4.18) is $\mathcal{O}(\log^2 k)$, it contains a "cross term" proportional to $\beta_k p \ln k$ where β_k depends on M_k . On the basis of our numerical experience in calculating the total number of samples (see Section 4.4) it is clear that unless M_k is small, the "ln k term" will dominate until k is very large. Thus, as expected, the exact confidence intervals provided by this approach are obtained at the expense of large sample sizes.

Moreover, the result is of little use unless we can determine the bounds M_k . We know from Proposition 4.6 that if the difference random variables can be expressed as a sample mean of i.i.d.r.v.'s and the underlying random variables have moment generating functions then $M_k < \infty$. This result, however, does not provide a method for calculating M_k . In Lemma 4.12, we provide a bound on the moments of the normalized sample mean which can be useful in calculating M_k . An application of Lemma 4.12 is given in Proposition 4.13 for the special case in which the underlying random variables are bounded. The proof of Lemma 4.12 requires the following result.

Lemma 4.11 *There are no more than $[\sqrt{sn}]^s$ ways to place s distinguishable balls in n distinguishable bins such that no bin contains exactly one ball.*

Proof

Let γ_n^s denote the quantity in question; we must show $\gamma_n^s \leq [\sqrt{sn}]^s$. The desired result clearly holds when $s \geq n$ and is also immediate for $s \leq 3$. Thus, we restrict our attention to the case $s \geq 4$ and $n \geq s$ and proceed by induction. By conditioning on whether the first two balls are placed in the same or different bins we obtain the following recursion relation:

$$\gamma_n^s = n \sum_{j=0}^{s-2} \binom{s-2}{j} \gamma_{n-1}^{s-2-j} + n(n-1) \sum_{j=2}^{s-2} \binom{s-2}{j} (2^j - 2) \gamma_{n-2}^{s-2-j}$$

It is clear, after reindexing the sum in the second term of the right hand side of the recursion relation, that to verify the inductive step it suffices to show:

$$\begin{aligned} s^{s/2} n^{s/2} &\geq n \sum_{j=0}^{s-2} \binom{s-2}{j} (s-2-j)^{(s-2-j)/2} (n-1)^{(s-2-j)/2} \\ &\quad + n(n-1) \sum_{j=0}^{s-4} \binom{s-2}{j+2} (2^{j+2} - 2) (s-4-j)^{(s-4-j)/2} (n-2)^{(s-4-j)/2} \end{aligned}$$

Observe for the range of s and n under consideration

$$\binom{s-2}{j+2} (2^{j+2} - 2) (s-4-j)^{(s-4-j)/2} \leq \binom{s-4}{j} 2^j (s-2)(s-3)(s-4)^{(s-4-j)/2}.$$

Applying the binomial theorem we see it suffices to show:

$$s^{s/2} n^{s/2} \geq n \left[1 + \sqrt{(s-2)(n-1)} \right]^{s-2} + n(n-1)(s-2)(s-3) \left[2 + \sqrt{(s-4)(n-2)} \right]^{s-4}. \quad (4.19)$$

Further observe for our range of s and n

$$1 + \sqrt{(s-2)(n-1)} \leq \sqrt{sn}, \quad (4.20)$$

and

$$2 + \sqrt{(s-4)(n-2)} \leq \sqrt{sn}. \quad (4.21)$$

The inductive step may be verified upon substitution of inequalities (4.20) and (4.21) into (4.19). ■

In the following lemma we will use the notation: $X_1, \dots, X_n$ denote i.i.d.r.v.'s. $\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$, $EX_1 = \mu$, and $E(X_1 - \mu)^2 = \sigma^2$ where $0 < \sigma^2 < \infty$.

Lemma 4.12 *If $E|X_1 - \mu|^s < \infty$ for some positive integer s then*

$$E \left(\frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \right)^s \leq E|X_1 - \mu|^s s^{s/2} / \sigma^s.$$

Proof

Let $Y_i = X_i - \mu$. Then

$$E \left(\frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \right)^s = n^{-s/2} \sigma^{-s} E \left(\sum_{i=1}^n Y_i \right)^s.$$

Thus, it suffices to verify

$$E \left(\sum_{i=1}^n Y_i \right)^s \leq E|Y_1|^s (\sqrt{sn})^s.$$

We will denote $k = (k_1, \dots, k_n)$, $\mathcal{S}_n = \{k : \sum_{i=1}^n k_i = s, k_i \geq 0\}$, and $\mathcal{S}_n^- = \{k : k \in \mathcal{S}_n, k_i \neq 1\}$.

Now

$$E \left(\sum_{i=1}^n Y_i \right)^s = \sum_{k \in \mathcal{S}_n} \frac{s!}{k_1! \dots k_n!} \prod_{i=1}^n EY_1^{k_i} \leq E|Y_1|^s \sum_{k \in \mathcal{S}_n^-} \frac{s!}{k_1! \dots k_n!}$$

since

$$\max_{k \in \mathcal{S}_n} \prod_{i=1}^n EY_1^{k_i} \leq E|Y_1|^s. \quad (4.22)$$

Inequality (4.22) follows easily from the result: $[E|Y_1|^r]^{1/r}$ is nondecreasing on $(0, s]$ (see Feller §V.8 [26]). It remains to show

$$\sum_{k \in S_n^-} \frac{s!}{k_1! \cdots k_n!} \leq (\sqrt{sn})^s, \quad (4.23)$$

and (4.23) follows immediately from Lemma 4.11. ■

Proposition 4.13 *If $|X_1 - \mu| \leq C$ then $E \exp \left[\gamma \left(\frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \right) \right] \leq M$ where*

$$M \leq \sum_{k=0}^{N-1} \frac{(\gamma C \sqrt{k}/\sigma)^k}{k!} + \frac{1}{\sqrt{2\pi}} \left(\frac{\gamma e C}{\sigma \sqrt{N}} \right)^N / \left(1 - \frac{\gamma e C}{\sigma \sqrt{N}} \right)$$

and where $N > (\gamma e C / \sigma)^2$.

Proof

Let $Z_n = \sigma^{-1} \sqrt{n} (\bar{X} - \mu)$ and observe

$$E e^{\gamma Z_n} = \sum_{k=0}^{\infty} \frac{\gamma^k}{k!} E Z_n^k \leq \sum_{k=0}^{\infty} \frac{\gamma^k}{k!} E |X_1 - \mu|^k k^{k/2} / \sigma^k$$

where the inequality follows from Lemma 4.12. Based on the Stirling inequality (see Feller §II.10 [27])

$$k! > \sqrt{2\pi} k^{k+\frac{1}{2}} e^{-k} e^{(12k+1)^{-1}}$$

it follows

$$\frac{k^{k/2}}{k!} < \frac{e^k}{\sqrt{2\pi} k^{k/2}}.$$

Thus

$$\begin{aligned} E e^{\gamma Z_n} &\leq \sum_{k=0}^{N-1} \frac{(\gamma C \sqrt{k}/\sigma)^k}{k!} + \frac{1}{\sqrt{2\pi}} \sum_{k=N}^{\infty} (\gamma e C / \sigma)^k (N)^{-k/2} \\ &= \sum_{k=0}^{N-1} (\gamma e C / \sigma)^k k^{-k/2} + \frac{1}{\sqrt{2\pi}} \left(\frac{\gamma e C}{\sigma \sqrt{N}} \right)^N / \left(1 - \frac{\gamma e C}{\sigma \sqrt{N}} \right). \quad \blacksquare \end{aligned}$$

Proposition 4.13 is intended to provide a method for calculating bounds to be used in Theorem 4.10. This calculation requires knowledge of C and σ which may not be at hand. However, in some special cases one may be able to calculate bounds on C (upper bound) and σ (lower bound) which facilitate numerical calculation of M . In some applications one may have scenarios which

exhibit a (partial) ordering with respect to their cost. In the stochastic hydroelectric scheduling example of Chapter 1, larger natural inflows yield smaller costs provided one can always spill without penalty. In portfolio optimization applications, larger returns are usually preferred. This information may be used to calculate a “worst” and “best” scenario needed to calculate C . Similarly, aggregation of scenarios with respect to the partial ordering can be used to calculate bounds on moments.

In summary, an exact confidence interval is provided via exponential inequalities derived from Markov’s inequality. However, one may expect to pay a high price in terms of the number of samples required to generate a confidence interval of given width. Moreover, except in some special cases, it may be difficult to obtain sharp bounds for M_k , which exacerbates the sample size issue. For these reasons, in the remainder of the development we will restrict our attention to approximate confidence intervals generated through limit theorems.

4.7 Stopping Rules: CLT Differences and Sample Variances

Theorems 4.4 and 4.9 yield desirable asymptotic coverage results for the objective function value of the proposed solution to optimization problem (4.1) when the sample sizes are selected according to formula (4.13). If the confidence interval width, ϵ , and the stopping tolerance, ϵ' are preselected, fixed values, then we interpret the theorems as providing an approximate *absolute precision* confidence interval. However, in practice the population variance terms, σ_k^2 , are typically unknown. Hence the required sample sizes provided by (4.13) are also unknown, and implementation of the procedure is not possible. Alternatively, as we indicated in Remark 4.8, ϵ and ϵ' may be selected proportional to σ_k , and we obtain a *relative precision* confidence interval. In this case, the sample sizes do not depend on the variances and are known. The resulting confidence interval, however, has length $\epsilon\sigma_T$, where T is the stopping time; this width is unknown. There are “standard” approaches to this difficulty based on well-known results from parameter estimation in statistics. The techniques differ somewhat for the relative and absolute cases. We begin by stating two results which we will use several times in the remainder of this and the subsequent chapter.

Theorem 4.14 [Continuous Mapping] Suppose $h : \mathbb{R}^d \rightarrow \mathbb{R}$, $X_n \Rightarrow X$, and $P\{X \in \mathcal{D}_h\} = 0$ where $\mathcal{D}_h$ is the set of discontinuities of h . Then

$$h(X_n) \Rightarrow h(X).$$

Proof

See Corollary 1 to Theorem 5.1 of Billingsley [6]. ■

Theorem 4.15 [Converging-Together] Suppose $h : \mathbb{R}^d \rightarrow \mathbb{R}$, $X_n \Rightarrow X$, and $Y_n \Rightarrow a$ where a is deterministic. If $P\{(X, a) \in \mathcal{D}_h\} = 0$ where $\mathcal{D}_h$ is the set of discontinuities of h then

$$h(X_n, Y_n) \Rightarrow h(X, a).$$

Proof

Theorem 4.4 of Billingsley [6] states $X_n \Rightarrow X$ and $Y_n \Rightarrow a$ implies $(X_n, Y_n) \Rightarrow (X, a)$. The desired result then follows from the Continuous Mapping result. ■

For simplicity, we will describe the “standard” approach to handle the unknown population variance in the setting of Theorem 4.4 for the relative precision case. In addition to the CLT hypothesis (4.2) we require a sample variance estimator, s_k^2 , that satisfies $s_k^2 \Rightarrow \sigma_k^2 > 0$; by the Converging-Together result

$$\frac{D_k - \mu_k}{s_k / \sqrt{n_k}} \Rightarrow N(0, 1). \quad (4.24)$$

We modify the moment generating function condition (4.12) and the sample-size formula (4.13) by replacing the population variance with the sample variance. With these straightforward modifications we can verify a “sample variance” analog of Theorem 4.4 for the relative precision case. The reason why such a result is not particularly useful is the sample variance equivalent of the moment generating function hypothesis (4.12) may be difficult to verify. In the population variance case, Proposition 4.6 provided “reasonable” conditions under which the hypothesis is true. However, in the sample variance case, if D_k is a sample mean of i.i.d.r.v.’s which are normally distributed then $s_k^{-1} \sqrt{n_k} (D_k - \mu_k)$ is a Student’s T random variable which does not have a moment generating function. Since the moment generating function hypothesis fails even in this simple instance we will

seek an alternative approach to dealing with the unknown confidence interval width. (The situation for an absolute precision confidence interval is no better. The sample size in this case is a random variable, and the prospect of verifying condition (4.12) in this setting is at best unimproved.)

In light of Remark 4.4, a relatively simple solution to this difficulty is at hand through a resampling procedure. After iteration T we have a proposed solution $\mathbf{x}_T$ and we can resample the corresponding difference random variable. We denote this random variable $\tilde{D}_T$ and assume it satisfies the CLT (4.2) with mean μ_T and variance σ_T^2 ; at the time of resampling these parameters are deterministic. Hypothesis (4.2) for $\tilde{D}_T$ coupled with a weakly consistent sample variance (i.e., $\hat{s}_T^2 \Rightarrow \sigma_T^2$) ensures the sample variance version of the CLT result (4.24) holds for $\tilde{D}_T$. From this we infer

$$\left[z^*, z^* + \tilde{D}_T^* + \frac{w_\delta \hat{s}_T}{\sqrt{n}} \right] \quad (4.25)$$

is an approximate $100(1 - \delta)\%$ confidence interval for $z(\mathbf{x}_T)$ where n denotes the resampling sample size and w_δ satisfies $P\{N(0, 1) \leq w_\delta\} = 1 - \delta$. As Theorems 4.4 and 4.9 ensure μ_T is not too large, we have *a priori* control on the confidence interval width. Restated: For sufficiently small ϵ and for sufficiently large $n = n(\eta)$ we have

$$P\left\{ \tilde{D}_T^* + \frac{w_\delta \hat{s}_T}{\sqrt{n}} \geq \epsilon \sigma_T + \eta \right\} \approx \alpha \quad \text{for any } \eta > 0. \quad (4.26)$$

Indeed, if μ_T is small then the main source of control of the confidence interval width resides in the term $w_\delta \hat{s}_T / \sqrt{n}$. Based on sample variance information from iteration T and previous iterations we can make a reasonable choice of the resampling sample size n in this regard. Thus (4.26) verifies that we have control of the width of the confidence interval (4.25) in a relative precision sense.

4.8 Summary

In this chapter we developed a stopping rule theory for a class of optimization algorithms that estimate upper and lower bounds on the optimal objective function value via Monte Carlo sampling. In the “main result” we assume that the difference random variables satisfy history dependent central limit theorems. Under this hypothesis, we provide appropriate conditions and a sample size formula under which the desired confidence interval for the objective function value of the proposed

solution is asymptotically valid. We regard the recommended procedure as conservative because: (1) underlying convergence properties of the optimization algorithm are ignored in developing the methodology; and (2) the normal tail bound used to derive the sample-size formula is not sharp, particularly in the early iterations. We also derived an even more conservative distribution-free result which holds for all positive confidence interval widths, and we discussed resampling procedures for incorporating sample variance estimators in confidence interval construction.

Chapter 5

Stopping Rules: Applications

5.1 Introduction

In Sections 5.2 and 5.3 we apply the stopping rule theory developed in Chapter 4 to the stochastic programming algorithms reviewed in Chapter 3. The history dependent result of Theorem 4.9 will prove particularly appropriate for the applications we consider. We will primarily be concerned with developing confidence intervals of the form $[z^*, z^* + \epsilon]$ for the objective function value of the proposed solution; i.e., $z(x_T)$. These confidence intervals are derived from the asymptotic validity results of the previous chapter. The issue of finite stopping for these two applications will not be addressed. We view the question of valid confidence intervals as more important than finite stopping. A finite stopping result hardly ensures a practical algorithm. For example, suppose the feasible region, $\mathcal{X}$, of optimization problem (4.1) is compact, $z(x)$ is smooth, and at each iteration we randomly select a feasible decision $x^k \in \mathcal{X}$. Under appropriate conditions on the upper and lower bound estimates, finite stopping may be verified for such an algorithm. A random search, however, is not likely to provide an efficient algorithm. Moreover, heuristic stopping criterion can always be implemented. With a CLT assumption on the difference random variables, valid confidence intervals, albeit without *a priori* bounds on the width of the interval, may be developed in this setting (see Remark 4.4 and Section 4.7). Provided the heuristic stopping times yield good solutions, such an approach may be attractive in practice.

In the algorithm we consider in Section 5.2 the lower bounds contain no error due to sampling, and in the history dependent sense, the upper bounds are sample means of i.i.d.r.v.'s. Applying Theorem 4.9 in this setting is straightforward. The algorithm we consider in Section 5.3 is more

complex to analyze. The upper bounds, once again, are sample means of i.i.d.r.v.'s. The lower bounds, however, are nonlinear functions of i.i.d. random vectors. As a result, the analysis in this setting is more difficult.

5.2 The SLP-T Algorithm

In Chapter 3 we described an algorithm for SLP-T where the number of stages may be large, but each node of the scenario tree has a “manageable” number of descendants. Upper bounds are estimated by “sampling paths” of the scenario tree. Recall a sample path, $\omega = (\omega_1, \omega_2, \dots, \omega_T)$, specifies exactly one node per stage in the scenario tree and has the property that the nodes identify a path from the (unique) stage 1 node to a stage T leaf node. Lower bounds, on the other hand, are computed exactly; i.e., there is no estimation error due to sampling in the lower bounds. This algorithm was introduced by Pereira and Pinto [51]. The algorithm for SLP-T is summarized for convenient reference in Figure 5.1.

step 0 let $k = 0$;
 initialize the set of stage t cuts with $\theta_t \geq -M_t, t = 1, \dots, T-1$;

step 1 solve the stage 1 master program and obtain (x_1^k, θ_1^k) ;
 let $\underline{z}^k = c_1 x_1^k + \theta_1^k$;

step 2 select a set of random sample paths $\mathcal{S}^k$ according to $p_T(\omega_T)$;
 do $\omega \in \mathcal{S}^k$
 do $t = 2$ to T
 form RHS of sub(ω_t): $B_{t-1}(\omega_t)x_{t-1}^k(\omega_{t-1}) + b_t(\omega_t)$;
 solve and obtain $x_t^k(\omega_t)$;
 enddo
 enddo
 let $\bar{z}^k = c_1 x_1^k + \frac{1}{|\mathcal{S}^k|} \sum_{\omega \in \mathcal{S}^k} \sum_{t=2}^T c_t(\omega_t) x_t^k(\omega_t)$;

step 3 if $\bar{z}^k - \underline{z}^k \leq 0$ then stop: x_1^k is the proposed solution;

step 4 do $t = T-1$ downto 1
 do $\omega \in \mathcal{S}^k$
 do $\omega_{t+1} \in \Delta(\omega_t)$
 form RHS of sub(ω_{t+1}): $B_t(\omega_{t+1})x_t^k(\omega_t) + b_{t+1}(\omega_{t+1})$;
 solve and obtain $\pi_{t+1}(\omega_{t+1})$;
 enddo
 augment the set of stage t cuts with $\theta_t - G_t x_t \geq g_t$
 enddo
 enddo

step 5 let $k = k + 1$; goto step 1;

Figure 5.1: Benders Decomposition Algorithm for SLP-T

Observe the first stage master program decision at the k^{th} iteration, x_1^k , is a random variable. The decision depends on the set of first stage cuts, and these cuts, in turn, depend on the randomly selected sample paths of previous iterations. However, given the history of the algorithm through the first $k - 1$ iterations, $\mathcal{H}_{k-1}$, the decision x_1^k is known, and the upper bound estimate $\bar{z}^k$ is the sum of a deterministic constant, $c_1 x_1^k$, and a sample mean of i.i.d.r.v.'s with realizations of the form $z_2^k(\omega) = \sum_{t=2}^T c_t(\omega_t) x_t^k(\omega_t)$. Suppose the subproblems at each stage have bounded primal feasible regions. The random variables $\xi_t = (c_t, b_t, B_{t-1}, A_t)$, $t = 2, \dots, T$ are bounded as there are only a finite number of realizations, and hence the random variable z_2^k is bounded. Thus, the upper bound estimate satisfies a conditional CLT (see (4.16)) with mean $u_k = c_1 x_1^k + E z_2^k$ and variance $\sigma_{u_k} = E (z_2^k - E z_2^k)^2$. The lower bound, $\underline{z}^k = l_k$, is deterministic when conditioned on $\mathcal{H}_{k-1}$. It follows that the difference random variable $D_k = \bar{z}^k - \underline{z}^k$ satisfies the conditional CLT hypothesis (4.16) of Theorem 4.9 where $\mu_k = u_k - l_k$ and $\sigma_k^2 = \sigma_{u_k}^2$. Moreover, as z_2^k is bounded the conditional moment generating function hypothesis (4.17) is satisfied (see Proposition 4.6), and the weaker condition (4.12) necessary for asymptotic validity then follows.

We now describe the recommended algorithm for SLP-T based on the stopping rule theory of Chapter 4. The modified algorithm utilizes the idea of stopping cycles (see Remark 4.3) which help to minimize computational burden. The existing steps of the algorithm in Figure 5.1 are modified as follows. To step 0 we append “define $\epsilon > \epsilon' \geq 0$; let $\nu = 0$, $n_\nu > 0$ (e.g., 30), $p > 0$, $0 < \alpha < 1$ (e.g., 0.05), and let $\beta' = \max \{2 \ln [\phi(p)/(\sqrt{2\pi} \alpha)], 1\}$.” See Section 4.4 for the definition of $\phi(p)$ and for recommended values of p . In step 2 we select the random sample $\mathcal{S}^k$ to be of size n_ν . In step 3, if the hypothesis is satisfied then a stopping cycle is complete and we go to step 6. The additional steps are detailed in Figure 5.2.

We are employing a relative precision confidence interval as discussed in Section 4.7. The sample size formula (4.13) of Theorem 4.9 is satisfied by the modified algorithm, and hence the asymptotic validity result holds. Note $P\{\mu_T \leq \epsilon \sigma_T\} \geq 1 - \alpha$ implies $P\{l_T \leq z(x_T) \leq l_T + \epsilon \sigma_T\} \geq 1 - \alpha$, and in this setting, l_T is a known lower bound when the algorithm terminates. The variance, σ_T , is typically unknown, and thus it may be necessary to employ an additional resampling step as described in Section 4.7 to obtain a confidence interval of the form (4.25). We can also strengthen

step 6 let $\nu = \nu + 1$; and $n_\nu = \lceil (\epsilon - \epsilon')^{-2} (\beta' + 2p \ln^2 \nu) \rceil$

step 7 independently select a set of new sample paths $\mathcal{S}^k$ according to $p_T(\omega_T)$ of size n_ν ;
do $\omega \in \mathcal{S}^k$
do $t = 2$ to T
form RHS of sub(ω_t): $B_{t-1}(\omega_t)x_{t-1}^k(\omega_{t-1}) + b_t(\omega_t)$;
solve and obtain $x_t^k(\omega_t)$;
enddo
enddo
let $\bar{z}^k = c_1 x_1^k + \frac{1}{|\mathcal{S}^k|} \sum_{\omega \in \mathcal{S}^k} \sum_{t=2}^T c_t(\omega_t) x_t^k(\omega_t)$;

step 8 if $\bar{z}^k - \underline{z}^k \leq \epsilon'$ then stop: x_1^k is the proposed solution; otherwise goto step 4

Figure 5.2: Additional “Stopping Rule” Steps to the Algorithm for SLP-T

this result by replacing z^* by the known lower bound l_T .

5.3 The SLP-2 Algorithm

5.3.1 Algorithm

In Chapter 3, we described an algorithm developed by Dantzig and Glynn [16] and Infanger [36] for solving two stage stochastic linear programs with recourse (SLP-2). The algorithm uses a variance reduction technique called importance sampling within a Benders decomposition framework, and both upper and lower bounds on the optimal objective function value are estimated via sampling. For simplicity of exposition, we will assume in this section the Monte Carlo sampling is performed naively. The algorithm for SLP-2 is summarized in Figure 5.3.

5.3.2 Upper Bound

In this section on the upper bound, and Section 5.3.3 on the lower bound, we will assume any inter-iteration dependencies of the algorithm due to sampling are negligible. We will rectify this situation by subsequently generalizing the discussion to the history dependent setting in Section 5.3.4. The upper bound estimate, $\bar{z}^k$, is the sum of a deterministic constant, $c_1 x_1^k$, and a sample mean of i.i.d.r.v.'s with realizations of the form $z_2^k(\omega) = c_2(\omega) x_2^k(\omega)$. Under mild conditions on the random

step 0 let $k = 0$; initialize the set of cuts with $\theta \geq -M$;

step 1 solve the master program and obtain (x_1^k, θ^k) ;
let $\underline{z}^k = c_1 x_1^k + \theta^k$;

step 2 select a set of random samples $\mathcal{S}^k$ according to $p(\omega)$;
do $\omega \in \mathcal{S}^k$
 form RHS of sub(ω): $B_1(\omega)x_1^k + b_2(\omega)$;
 solve and obtain $(x_2^k(\omega), \pi_2^k(\omega))$;
enddo
let $\bar{z}^k = c_1 x_1^k + \frac{1}{|\mathcal{S}^k|} \sum_{\omega \in \mathcal{S}^k} c_2(\omega) x_2^k(\omega)$;

step 3 if $\bar{z}^k - \underline{z}^k \leq 0$ then stop: x_1^k is the proposed solution;

step 4 augment the set of cuts with $\theta - \tilde{G}_{k+1} x_1 \geq \tilde{g}_{k+1}$ where
 $\tilde{G}_{k+1} = \frac{1}{|\mathcal{S}^k|} \sum_{\omega \in \mathcal{S}^k} \pi_2^k(\omega) B_1(\omega)$ and $\tilde{g}_{k+1} = \frac{1}{|\mathcal{S}^k|} \sum_{\omega \in \mathcal{S}^k} \pi_2^k(\omega) b_2(\omega)$;
let $k = k + 1$; goto step 1;

Figure 5.3: Benders Decomposition Algorithm for SLP-2

variable z_2^k , the upper bound estimate satisfies a central limit theorem with mean $u_k = c_1 x_1^k + E z_2^k$ and variance $\sigma_{u_k}^2 = E (z_2^k - E z_2^k)^2$.

5.3.3 Lower Bound

Verifying that the lower bound estimate, $\underline{z}^k$, is asymptotically normal is more difficult than the upper bound. First, we require a result regarding differentiability of a linear program when viewed as a function of its data. The result we verify is contained in Proposition 5.2, and to this end we begin with the following lemma.

Lemma 5.1 *Suppose $A \in \mathbb{R}^{n \times n}$ is nonsingular. If $E \in \mathbb{R}^{n \times n}$ and $\|E\| \|A^{-1}\| < 1$ then $A + E$ is nonsingular.*

Proof

$A + E = A(I + A^{-1}E)$; thus it suffices to show $I + A^{-1}E$ is nonsingular. If y and z are vectors and B a matrix of appropriate dimensions, recall the following properties of any vector and consistent

matrix norm: $\|y - z\| \geq \left| \|y\| - \|z\| \right|$ and $\|By\| \leq \|B\| \|y\|$. With these properties it follows that for any x

$$\|(I + A^{-1}E)x\| = \|x + A^{-1}Ex\| \geq \|x\| - \|A^{-1}Ex\| \geq (1 - \|A^{-1}\| \|E\|) \|x\|.$$

By hypothesis, $1 - \|A^{-1}\| \|E\| > 0$; hence, $\|(I + A^{-1}E)x\| = 0$ only if $x = 0$. The proof is complete as only the trivial linear combination of the columns of $I + A^{-1}E$ yields zero. ■

Consider the following perturbed linear program (in standard form) and its dual:

$$\begin{array}{ll} \min & cx \\ (P_\epsilon) \quad \text{s.t.} & (A + \epsilon H)x = b + \epsilon h \\ & x \geq 0 \end{array} \quad \begin{array}{ll} \max & \pi(b + \epsilon h) \\ (D_\epsilon) \quad \text{s.t.} & \pi(A + \epsilon H) \leq c \end{array}$$

where $A \in \mathbb{R}^{m \times n}$, $\text{rank}(A) = m$, and $\epsilon \geq 0$ is a scalar. Let $f(A, b) = \min \{cx : Ax = b, x \geq 0\}$. We denote optimal solutions to (P_ϵ) and (D_ϵ) by $x^*(\epsilon)$ and $\pi^*(\epsilon)$, respectively. When referring to the unperturbed (i.e., $\epsilon = 0$) problems and their solutions we suppress ϵ ; i.e., (P) , (D) , x^* , and π^* .

Proposition 5.2 *If (P) has a unique, nondegenerate solution then $f(A, b)$ is continuously differentiable in a neighborhood of (A, b) .*

Proof

Let $\mathcal{B}$ denote an index set which corresponds to the m basic columns of a basic solution; the remaining $n - m$ column indices are elements of the nonbasic set $\mathcal{N} = \mathcal{B}^c$. A vector or matrix with subscript $\mathcal{B}$ (or $\mathcal{N}$) denotes the subvector or submatrix indexed by $\mathcal{B}$ (or $\mathcal{N}$). Note there is a unique basis index set $\mathcal{B}$ that corresponds to the optimal solution x^* . To see this, suppose $\mathcal{B}$ and $\mathcal{B}'$ are distinct and both correspond to x^* . Let $j \in \mathcal{B} \cap \mathcal{N}' \neq \emptyset$. Then $x_j^* > 0$ as the solution is nondegenerate and $j \in \mathcal{B}$, but by the definition of a nonbasic column, $x_j^* = 0$ as $j \in \mathcal{N}'$. From the unique, nondegenerate hypothesis, it follows from complementary slackness $\pi^* = c_{\mathcal{B}} A_{\mathcal{B}}^{-1}$ is the unique optimal solution of (D) . By the definition of nondegeneracy we have $x_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b > 0$, and since there is a unique optimal basis we have $c_{\mathcal{N}} - \pi A_{\mathcal{N}} > 0$.

Observe $x(\epsilon) = (x_{\mathcal{B}}(\epsilon), 0)$ and $\pi(\epsilon)$ defined by

$$(A_{\mathcal{B}} + \epsilon H_{\mathcal{B}})x_{\mathcal{B}}(\epsilon) = b + \epsilon h \quad \text{and} \quad \pi(\epsilon)(A_{\mathcal{B}} + \epsilon H_{\mathcal{B}}) = c_{\mathcal{B}} \quad (5.1)$$

are continuously differentiable in a neighborhood of $\epsilon = 0$. This follows by applying Cramer's rule and observing the resulting ratio of polynomials is continuously differentiable provided $(A_B + \epsilon H_B)$ is nonsingular. The nonsingularity condition is guaranteed by Lemma 5.1 provided ϵ satisfies $\epsilon \|H_B\| \|A_B^{-1}\| < 1$.

We now verify if ϵ is sufficiently small then $x(\epsilon)$ and $\pi(\epsilon)$, defined by (5.1), are the unique optimal solutions to (P_ϵ) and (D_ϵ) . By definition, $x(\epsilon)$ satisfies the structural primal constraints. Continuity of $x_B(\epsilon)$ ensures $x_B(\epsilon) > 0$ for sufficiently small ϵ since $x_B^* > 0$. The definition of $\pi(\epsilon)$ and $x_N(\epsilon)$ ensure complementary slackness, and it remains to verify dual feasibility for columns of N . We may write

$$c_N - \pi(\epsilon)(A_N + \epsilon H_N) = c_N - \pi^* A_N - [(\pi(\epsilon) - \pi^*)(A_N + \epsilon H_N) + \epsilon \pi^* H_N].$$

The right hand bracketed term has value zero at $\epsilon = 0$ and is continuous in ϵ ; hence, $c_N - \pi(\epsilon)(A_N + \epsilon H_N) > 0$ for sufficiently small ϵ . Uniqueness follows as we have primal nondegeneracy and strictly positive reduced costs for the nonbasic columns. $f(A + \epsilon H, b + \epsilon h) = c_B x_B^*(\epsilon)$, and in a neighborhood of $\epsilon = 0$ we have shown $x_B^*(\epsilon) = x_B(\epsilon)$. Thus $f(A, b)$ is continuously differentiable in a neighborhood of (A, b) . ■

The lower bound estimate in the two stage algorithm is generated from the optimal objective function value of the master program. A "generic" master program has the form

$$\begin{array}{llll} \text{master} & \text{minimize} & c_1 x_1 & + \quad \theta \\ & \text{subject to} & A_1 x_1 & = b_1 \quad : \pi_1 \\ & \text{cuts:} & -G x_1 & + \quad e \theta \geq g \quad : \lambda \\ & & x_1 & \in \mathcal{R}_+^{n_1} \end{array}$$

where

$$e = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad G = \begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ G_K \end{bmatrix}, \quad g = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_K \end{bmatrix}, \quad \text{and} \quad \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_K \end{bmatrix}.$$

G_k and g_k denote the gradient and intercept of the k^{th} cut, respectively, and π_1 and λ denote dual variables. We distinguish between two master programs: the *true master* and the *pseudo master*. The true master is the linear program with *exact* cuts; the k^{th} exact cut has gradient $G_k = E_\omega \pi_2^k(\omega) B_1(\omega)$

and intercept $g_k = E_\omega \pi_2^k(\omega) b_2(\omega)$. The true master's solutions are denoted: primal (x_1, θ) , dual (π_1, λ) , and objective l_K . The pseudo master is the linear program with cuts generated by sampling; the k^{th} sampled (or pseudo) cut has gradient $\tilde{G}_k$ and intercept $\tilde{g}_k$. The pseudo master's solutions are denoted: primal $(\tilde{x}_1, \tilde{\theta})$, dual $(\tilde{\pi}_1, \tilde{\lambda})$, and objective $\underline{z}^K$. These solutions all correspond to the master programs at iteration K and may also be indexed by K . For example, λ_j^K denotes the dual variable associated with the j^{th} cut when the true master is solved at the K^{th} iteration. With the exception of the objective function values, we will suppress this additional index whenever possible. We define the true master so that its cuts are evaluated at the same sequence of master program decisions as the pseudo master; more specifically, we ensure $E\tilde{G}_k = G_k$ and $E\tilde{g}_k = g_k$. Let $\bar{V}_n^k = [\tilde{g}_k, -\tilde{G}_k]$, and $V_n = [V_n^1, \dots, V_n^K]$. V_n^k is a sample mean of n i.i.d. random vectors with realizations of the form $V^k(\omega) = [\pi_2^k(\omega) b_2(\omega), -\pi_2^k(\omega) B_1(\omega)]$. We may view $\underline{z}^K$ as a nonlinear function, f , of the sample mean V_n where $f : \mathbb{R}^{(n_1+1)K} \rightarrow \mathbb{R}$ is the optimal value of the master program given cut gradient and intercept values. We will require the following result regarding nonlinear functions of sample means.

Theorem 5.3 *Let $V_1, V_2, \dots$ be a sequence of random vectors. Suppose there exists $v \in \mathbb{R}^d$ such that*

$$\sqrt{n}(V_n - v) \Rightarrow N(0, \Sigma) \quad \text{as } n \rightarrow \infty.$$

If $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is continuously differentiable in a neighborhood of v then

$$\sqrt{n}(f(V_n) - f(v)) \Rightarrow N(0, \sigma^2) \quad \text{as } n \rightarrow \infty,$$

where $\sigma^2 = \nabla^T f(v) \Sigma \nabla f(v)$.

Proof

See Glynn and Whitt for a functional central limit theorem version of this result [33]. ■

Under mild conditions on the underlying random vectors V^k , each V_n^k satisfies a multivariate CLT. Moreover, as each cut is generated from an independently selected sample (recall we are currently neglecting inter iteration interactions) the composite vector V_n satisfies the CLT hypothesis

of Theorem 5.3 where the covariance matrix Σ has the following block diagonal structure

$$\Sigma = \begin{bmatrix} \Sigma^1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \Sigma^K \end{bmatrix},$$

and where $\Sigma^k = \text{cov}(V^k, V^k)$; i.e., $[\Sigma^k]_{ij} = E(V_i^k - EV_i^k)(V_j^k - EV_j^k)$. In our application, v is the vector of exact cut intercepts and gradients of the true master; i.e., $v = [g_1, -G_1, \dots, g_K, -G_K]$. $f(v)$ is the objective function value of the true master program which we have denoted l_K , and $f(\bar{V}_n)$ is the objective function value of the pseudo master denoted $\bar{z}^K$. If the true master (when expressed in standard form) has a unique nondegenerate solution then by Proposition 5.2, the function f satisfies the “continuously differentiable in the neighborhood of v ” hypothesis of Theorem 5.3 and we have

$$\sqrt{n}(\bar{z}^K - l_K) \Rightarrow N(0, \sigma_{l_K}^2) \quad \text{as } n \rightarrow \infty \quad (5.2)$$

where

$$\sigma_{l_K}^2 = \nabla^T f(v) \Sigma \nabla f(v) = \sum_{k=1}^K \lambda_k^2 \begin{bmatrix} 1, & x_1^T \end{bmatrix} \Sigma^k \begin{bmatrix} 1 \\ x_1 \end{bmatrix}. \quad (5.3)$$

The gradient calculation of f is a straightforward exercise in linear programming sensitivity analysis. In particular,

$$\frac{\partial f}{\partial g_k} = \lambda_k \quad \text{and} \quad \frac{\partial f}{\partial (G_k)_i} = (x_1)_i \lambda_k,$$

where we recall x_1 and λ are optimal solutions to the true master program. Thus $\nabla^T f(\mu) = [\lambda^1(1, x_1^T), \dots, \lambda^K(1, x_1^T)]$ and we obtain the variance expression (5.3).

The exact cuts required to form the true master are unknown, and to generate confidence intervals of known width we require a (weakly consistent) point estimate of the population variance. Observe if we take S_n^k to be the usual sample covariance matrix corresponding to Σ^k and S_n to be the composite block diagonal sample covariance matrix then $S_n \Rightarrow \Sigma$ and, of course, $\bar{V}_n \Rightarrow v$. By the Converging-Together result, $(S_n, \bar{V}_n) \Rightarrow (\Sigma, v)$, and under the “unique and nondegenerate” assumption on the true master we can apply the Continuous Mapping Theorem and conclude

$$s_{l_K}^2 = \sum_{k=1}^K \bar{\lambda}_k^2 \begin{bmatrix} 1, & \bar{x}_1^T \end{bmatrix} S_n^k \begin{bmatrix} 1 \\ \bar{x}_1 \end{bmatrix} \quad (5.4)$$

is a weakly consistent estimator of $\sigma_{l_k}^2$. Thus by again applying the Converging Together result we achieve

$$\frac{\bar{z}^K - l_K}{s_{l_K}/\sqrt{n}} \Rightarrow N(0, 1). \quad (5.5)$$

Given a random sample $\mathcal{S}^k$ the (i, j) entry of the sample covariance matrix S_n^k is

$$[S_n^k]_{ij} = \frac{1}{|\mathcal{S}^k| - 1} \sum_{\omega \in \mathcal{S}^k} ([V^k(\omega)]_i - [\bar{V}^k]_i) ([V^k(\omega)]_j - [\bar{V}^k]_j)$$

where of course

$$[\bar{V}^k]_i = \frac{1}{|\mathcal{S}^k|} \sum_{\omega \in \mathcal{S}^k} [V^k(\omega)]_i.$$

5.3.4 Stopping Rule Assumptions

We now require some additional structure on SLP-2. Suppose a finite optimal solution to SLP-2 exists, the master program has a bounded primal feasible region, and the random vector $\xi_2 = (c_2, b_2, B_1, A_2)$ is bounded.

The upper and lower bounds at iteration k are generated from independent samples, and as we are currently neglecting inter-iteration dependencies carried by x_1^k (or otherwise), the upper and lower bounds are independent. Thus, if each cut and the upper bound are generated from samples of size n then based on the previous two sections, the difference random variable $D_k = \bar{z}^k - \underline{z}^k$ satisfies the central limit theorem $\sqrt{n}(D_k - \mu_k) \Rightarrow N(0, \sigma_k^2)$, where $\mu_k = u_k - l_k$ and $\sigma_k^2 = \sigma_{u_k}^2 + \sigma_{l_k}^2$. However, we know from Section 4.2 it is necessary to increase the sample size as the algorithm proceeds. In order to satisfy the sample size hypothesis of Theorem 4.4 (and 4.9) we must have $n = n_k \geq [(\epsilon - \epsilon')^{-2} (\beta' + 2p \ln^2 k)]$. We are employing a relative precision confidence interval as discussed in Section 4.7. Thus, we must ensure all *binding* cuts at the k^{th} iteration have been calculated using at least n_k samples; to achieve this, we may need to independently resample cuts from previous iterations. To resample cut j we must recall the corresponding master program decision x_1^j . Resampling cuts is clearly an operation we do not wish to perform every iteration of the algorithm in Figure 5.3, and we again emphasize the importance of good heuristics for defining “stopping cycles” (see Remark 4.3). An example of such an implementation is detailed in the recommended algorithm later in this section.

Recall in Section 4.3 we defined the history of the algorithm through iteration k , $\mathcal{H}_k$, as the information necessary to reconstruct exactly the steps of the algorithm through iteration k . The upper bound is a deterministic constant plus a sample mean of i.i.d.r.v.'s only when conditioned on $\mathcal{H}_{k-1}$. In two separate runs of the algorithm the value of u_k (which satisfies $\bar{z}_k \Rightarrow u_k$) will differ because different samples will produce two distinct sequences of master program decisions. Since we select the sample $\mathcal{S}^k$ independently at iteration k , $\bar{z}_k$ satisfies a conditional CLT (see (4.16)). On the other hand, cuts generated from samples selected in previous iterations have deterministic gradient and intercept when conditioned on the history. However, if we independently resample all binding cuts at the k^{th} iteration (stopping cycle) then the lower bound also satisfies a conditional CLT. The difference random variable may be written $D_k = \frac{1}{|\mathcal{S}^k|} \sum_{\omega \in \mathcal{S}^k} c_2(\omega) x_2^k(\omega) - \theta^k$ and satisfies the conditional CLT hypothesis (4.16) of Theorem 4.9.

Two unverified hypotheses remain: (i) the moment generating function hypothesis (4.12) and (ii) the unique and nondegenerate solution to the true master program assumption. The upper bound is a sample mean of i.i.d.r.v.'s, and satisfies (i) under the "bounded" assumptions on SLP-2 discussed at the beginning of this section (see Proposition 4.6 and the discussion of Section 5.2). We have shown the lower bound may be viewed as a nonlinear function of sample means. Verifying (i) in this case is considerably more difficult, and we will not further examine this issue here. However, under hypothesis (ii) we have verified a sample variance CLT result for the lower bound (5.5), and due to the simple structure of the upper bound an analogous result holds. Thus the difference random variable satisfies a sample variance CLT, and we can generate asymptotically valid confidence intervals for $z(x_T)$ under the assumption the true master at the stopping iteration T has a unique, nondegenerate solution. See Section 4.7 for further discussion of these confidence intervals. Strictly speaking we cannot apply Theorem 4.9, and hence, we are not assured probabilistic control over the resulting confidence interval width in the sense of (4.26). Nevertheless, the underlying idea of increasing the sample size at each stopping cycle according to formula (4.13) has attractive asymptotic coverage results for the sample mean case and hence remains the recommended procedure in this case as well. One of the primary sources of violation of (ii) may be two (or more) binding cuts in the pseudo master program which are identical in expected value and thus cause degeneracy in the true master program. This source of violation could be eliminated if the recourse function were strictly

convex. While we will not further examine this here, conditions under which strict convexity holds are therefore of interest. In large part, the algorithm will be attractive if, in practice, the confidence intervals generated are of satisfactory width and have good empirical coverage results, see (4.25); encouraging preliminary results in this regard have been reported; see, Infanger [36].

We now summarize the recommended algorithm. The existing steps in the algorithm of Figure 5.3 are modified as follows. To step 0 we append “define $\epsilon > \epsilon' \geq 0$; let $\nu = 0$, $n_\nu > 0$ (e.g., 30), $p > 0$, $0 < \alpha < 1$ (e.g., 0.05), and let $\beta' = \max \{2 \ln [\phi(p)/(\sqrt{2\pi} \alpha)], 1\}$.” See Section 4.4 for the definition of $\phi(p)$ and for recommended values of p . In step 2 we select the random sample $\mathcal{S}^k$ to be of size n_ν . In step 3, if the hypothesis is satisfied then a stopping cycle is complete and we go to step 5. The remaining steps are detailed in Figure 5.4 where we suppress additional subscripts to avoid confusion.

- step 5** let $\nu = \nu + 1$; and $n_\nu = \lceil (\epsilon - \epsilon')^{-2} (\beta' + 2p \ln^2 \nu) \rceil$
- step 6** for each tight cut j : independently select $\mathcal{S}$ of size n_ν according to $p(\omega)$;
do $\omega \in \mathcal{S}$
form RHS of sub(ω): $B_1(\omega)x_1^j + b_2(\omega)$;
solve and obtain $\pi_2(\omega)$;
enddo
compute the resulting gradient and intercept; replace cut j with the new cut
- step 7** solve the master problem and obtain (x_1, θ) ;
let $\underline{z} = c_1 x_1 + \theta$;
- step 8** independently select $\mathcal{S}$ of size n_ν according to $p(\omega)$;
do $\omega \in \mathcal{S}$
form RHS of sub(ω): $B_1(\omega)x_1 + b_2(\omega)$;
solve and obtain $x_2(\omega)$;
enddo
let $\bar{z} = c_1 x_1 + \frac{1}{n_\nu} \sum_{\omega \in \mathcal{S}} c_2(\omega) x_2(\omega)$;
- step 9** if $\bar{z} - \underline{z} \leq \epsilon'$ then stop: x_1 is the proposed solution; otherwise goto step 1

Figure 5.4: Additional “Stopping Rule” Steps to the Algorithm for SLP-2

5.4 Empirical Coverage Results

In this section we present preliminary empirical coverage results. These results are designed to illustrate dangers associated with naive stopping rules and to show empirical performance of the recommended stopping rules developed in Chapter 4. We consider two sets of test problems. In the first set of problems we use a (linear congruential) pseudo-random number generator directly to form the difference random variables. The second set of test problems are two-stage stochastic linear programs solved by a sampling and decomposition-based algorithm.

Our first set of coverage results is based on “test problems” motivated by Example 4.1. In the first group of these problems, the true gap (i.e., difference between upper and lower bounds) is $\mu_k = 2/3$ for $k = 1, \dots, K$ and $\mu_k = 0$ for $k \geq K + 1$. The prespecified confidence interval width is $\epsilon = 1/3$; thus if the algorithm stops prior to iteration $K + 1$ it will have stopped incorrectly. We use a stopping tolerance $\epsilon' = 0$; i.e., we stop on the first iteration in which the difference random variable drops below zero. The difference random variables are sample means of i.i.d.r.v.’s of the form $U(-\sqrt{3}, \sqrt{3}) + 2/3$, where $U(a, b)$ is a uniform random variable on the interval (a, b) . The choice of $(a, b) = (-\sqrt{3}, \sqrt{3})$ yields $\sigma_k = 1$. Table 5.1 depicts the empirical coverage results for two separate stopping rules on four test problems with different values of K . The first stopping rule ignores the sequential nature of the problem and uses the fixed sample size that would generate an asymptotically valid 95% one-sided confidence interval of width $1/3$ for the true gap, if the algorithm consisted of only one iteration. In particular, for sufficiently large values of n

$$P \left\{ D_k(n) \geq \mu_k - \frac{z_\alpha \sigma_k}{\sqrt{n}} \right\} \approx 1 - \alpha$$

where $P\{N(0, 1) \geq z_\alpha\} = \alpha$. Thus, if we observe $D_k(n) \leq 0$ we infer $\mu_k \leq z_\alpha \sigma_k / \sqrt{n} = \epsilon$ with (approximate) probability $1 - \alpha$. We choose $\alpha = 0.05$ which yields $z_\alpha = 1.645$ and thus $n = \lceil (z_\alpha \sigma_k / \epsilon)^2 \rceil = 25$. In the second stopping rule we increase the sample size at each iteration according to the $\mathcal{O}(\log^2 k)$ sample size formula (4.13). The parameter p is selected to be the “optimal” value from Table 4.2 for each value of K . Table 5.1 is based on 1000 replications and shows that ignoring the sequential nature of the problem does not lead to undercoverage results in this setting until K grows large. The sample size formula (4.13) leads to 100% coverage in each case.

K	$n = 25$	$n_k = \mathcal{O}(\log^2 k)$
	coverage	
10	0.997	1.000
100	0.963	1.000
1000	0.692	1.000
10000	0.029	1.000

Table 5.1: Coverage Results: $\epsilon = 1/3$ and $\mu = 2/3$

We now consider the case in which $\mu_k = \epsilon = 1/3$ for $k = 1, \dots, K$. Thus the only modification to the previous group of test problems is the difference random variables are sample means of i.i.d.r.v.'s of the form $U(-\sqrt{3}, \sqrt{3}) + 1/3$. This is a more demanding test of the stopping rule theory because the true gap and the confidence width are identical. The coverage results are summarized in Table 5.2. As one might expect the naive stopping rule has very poor coverage results in this setting. The coverage results obtained via sample size formula (4.13) significantly exceed the 95% confidence level in each case.

K	$n = 25$	$n_k = \mathcal{O}(\log^2 k)$
	coverage	
10	0.613	0.994
100	0.007	0.995
1000	0.000	0.993
10000	0.000	0.989

Table 5.2: Coverage Results: $\epsilon = 1/3$ and $\mu = 1/3$

The second set of coverage results is in the setting of two-stage stochastic linear programming, and we consider two test problems: APL1P and HYDRO1. APL1P is a small power expansion planning model (see Infanger [36]) with 5 stochastic parameters, 1280 scenarios, and two capacity expansion decision variables that connect the first and second stages. HYDRO1 is a two-stage hydroelectric scheduling model with 6 stochastic parameters, 64 scenarios, and 30 reservoir storage level decisions connecting the two stages. For these experiments, we modified a FORTRAN code developed by Infanger [36]. The modified algorithm for these experiments is essentially a variance-reduction (via importance sampling) version of the algorithm depicted in Figure 5.3 (see also Chapter 3), *except* that the cuts are computed exactly. Thus at iteration k we have an exact (deterministic)

lower bound, l_T , and a sample mean upper bound on the optimal objective function value. When deterministic cuts in the modified algorithm are computed, a deterministic upper bound is also at hand, but this upper bound is ignored and an upper bound is estimated by sampling second stage subproblems. The sole purpose of the modified algorithm is to study coverage results in a setting in which the difference random variable is a sample mean; it is not designed to be an intelligent solution strategy. Aside from the importance sampling procedure, the modified algorithm may also be viewed as the Pereira-Pinto algorithm applied to the two-stage problem.

In the modified algorithm we perform a preprocessing step which involves executing the standard modified algorithm where the sample mean upper bound is calculated with a constant sample size of 30 until iteration $T' = \inf\{k : D_k \leq 0\}$. After the preprocessing iterations we apply a constant sample size of $n = 30$ for stopping rule # 1 and use sample size formula (4.13) with $\alpha = 0.05$, $p = 0.155$, and $\epsilon = 1.699 s_k / \sqrt{30}$ for stopping rule # 2. On the first iteration $T' > T'$ in which the upper bound estimate drops below the deterministic lower bound the algorithm is terminated, and the proposed first stage decision is x_T . We then compute $z(x_T)$ by solving all the second stage subproblems and check whether

$$z(x_T) \in \left[l_T, l_T + \frac{1.699 s_T}{\sqrt{30}} \right].$$

We use the more conservative 1.699 multiplier from the Student's T distribution. The empirical results are summarized in Tables 5.3 and 5.4 for APL1P and HYDRO1 based on 100 replications. The optimal objective function value for APL1P is $z^* = 24642.3$ and for HYDRO1 is $z^* = 400807.3$. $\bar{z}(x_T)$ denotes the average (exact) objective function value of the recommended first stage decisions. $\overline{CI}/z^*$ shows the average relative confidence interval width, and $\overline{\text{iter}}$ denotes the average number of iterations *after* the preprocessing step. Figures 5.5 and 5.6 depict the true gap at each iteration with error bars. Stopping at any particular iteration is correct only if the error bars in the graph cover zero. It is clear from these figures that there are many more opportunities for incorrect stopping in HYDRO1 than in APL1P. This is consistent with the empirical coverage results in which the algorithm with the naive stopping rule stops correctly for test problem APL1P 100% of the time, but HYDRO1 stopped correctly only 58 out of 100 times with the naive rule.

In these experiments we choose the confidence interval width, ϵ , proportional to the sample

standard deviation, s_k . Based on the discussion of Section 4.7 we know that the stopping rule theory does not directly apply to this case because the sample variance analog of the moment generation function condition (4.12) has not been verified. Thus, we do not view these results as empirical validation of the stopping rule theory, but as another indication of the dangers associated with naive stopping rules in this sequential setting.

Stopping Rule	Coverage $z(x_T)$	$\bar{z}(x_T)$	$\overline{CI/z^*}$	$\overline{\text{iter}}$
1	100%	24670	1.4%	2.7
2	100%	24658	1.4%	3.2

Table 5.3: Coverage Results: APL1P

Stopping Rule	Coverage $z(x_T)$	$\bar{z}(x_T)$	$\overline{CI/z^*}$	$\overline{\text{iter}}$
1	58%	400970	0.08%	16.7
2	99%	400875	0.09%	38.2

Table 5.4: Coverage Results: HYDRO1

Figure 5.5: True gap vs. iteration: APLIP

Figure 5.6: True gap vs. iteration: HYDRO1

5.5 Summary and Areas for Future Research

One of the primary difficulties associated with sampling and decomposition-based algorithms is obtaining good lower bound estimates for the optimal objective function value. (Good upper bound estimates can typically be obtained through sample means by simulating the performance of a particular decision.) The algorithm of Pereira and Pinto (see Chapter 3) avoids the lower bound difficulty by considering a class of problems for which deterministic lower bounds can be calculated. In this case, the difference random variable at each iteration is a sample mean of i.i.d.r.v.'s (in the history dependent sense) and the stopping rule theory of Chapter 4 is applicable in straight-forward fashion. We showed that the lower bound in the two-stage algorithm of Dantzig, Glynn, and Infanger may be viewed as a nonlinear function of a (vector-valued) sample mean. Under a nondegeneracy condition on the "true" master program the lower bound is asymptotically normal and we calculated the corresponding variance term. While we indicated the special case in which the recourse function is strictly convex may be of particular interest, it is desirable to (more completely) characterize the class of stochastic programs which ensure that the nondegeneracy hypothesis is satisfied.

More generally, several interesting questions arise when applying the stopping rule theory to a particular algorithm. In addition to verifying the (history dependent) central limit theorem hypothesis, these include addressing the issue of finite stopping and verifying the moment generating function condition required for *a priori* relative width control of the asymptotic confidence intervals. While the preliminary empirical coverage results of Chapter 5 supported the claim that the stopping rule procedures we have developed are conservative, empirical results on a wide variety of test problems will be necessary so that the quality of the approximate confidence intervals can be verified. Due to the simple structure of the difference random variables (i.e., they are sample means), an implementation of the multistage algorithm described in Chapter 3 to include variance reduction techniques and interstage dependent right-hand-side vectors should provide an excellent platform for studying the stopping rules on an interesting class of multistage problems.

Bibliography

- [1] Abrahamson, P.G. (1983): A Nested Decomposition Approach for Solving Staircase Linear Programs, Ph.D. Dissertation, Department of Operations Research, Stanford University.
- [2] Barr, R., K. Farhangian, J.L. Kennington (1986): Networks with Side Constraints: An LU Factorization Update, *The Annals of the Society of Logistics Engineers* 1, 66-85.
- [3] Beale, E.M.L. (1955): On Minimizing a Convex Function Subject to Linear Inequalities, *Journal of the Royal Statistical Society* 17B, 173-184.
- [4] Benders, J.F. (1962): Partitioning Procedures for Solving Mixed-Variable Programming Problems, *Numerische Mathematic* 4, 238-252.
- [5] Bhattacharya, R.N. and R. Ranga Rao (1976): *Normal Approximation and Asymptotic Expansions*, John Wiley & Sons Inc.
- [6] Billingsley, P. (1968): *Convergence of Probability Measures*, John Wiley & Sons Inc.
- [7] Birge, J.R. (1985): Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs, *Operations Research* 33, 989-1007.
- [8] Birge, J.R. (1982): The Value of the Stochastic Solution in Stochastic Linear Programs with Fixed Recourse, *Mathematical Programming* 24, 314-325.
- [9] Birge, J.R. (1980): Solution Methods for Stochastic Dynamic Linear Programs, Technical Report SOL 80-29, Systems Optimization Laboratory, Department of Operations Research, Stanford University.
- [10] Birge, J.R. and F.V. Louveaux (1988): A Multicut Algorithm for Two-stage Stochastic Linear Programs, *European Journal of Operational Research* 34, 384-392.
- [11] Birge, J.R. and R. J.-B. Wets (1989): Sublinear Upper Bounds for Stochastic Programs with Recourse, *Mathematical Programming* 43, 131-149.
- [12] Birge, J.R. and R. J.-B. Wets (1986): Designing Approximation Schemes for Stochastic Optimization Problems, in particular, for Stochastic Programs with Recourse, *Mathematical Programming Study* 27, 54-102.
- [13] Charnes, A. and W.W. Cooper (1959): Chance-Constrained Programming, *Management Science* 6, 73-79.

- [14] Dantzig, G.B. (1963): *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey.
- [15] Dantzig, G.B. (1955): Linear Programming Under Uncertainty, *Management Science* 1, 197-206.
- [16] Dantzig, G.B., P.W. Glynn (1990): Parallel Processors for Planning Under Uncertainty, *Annals of Operations Research* 22, 1-21.
- [17] Dantzig, G.B., P.W. Glynn, M. Avriel, J.C. Stone, R. Entriken, and M. Nakayama (1989): Decomposition Techniques for Multi-Area Generation and Transmission Planning Under Uncertainty, EPRI Report 2940-1.
- [18] Dantzig, G.B. and M. Madansky (1961): On the Solution of Two-Staged Linear Programs under Uncertainty, *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability I*, ed. J. Neyman, 165-176.
- [19] Dantzig, G.B. and P. Wolfe (1960): The Decomposition Principle for Linear Programs, *Operations Research* 8, 101-111.
- [20] Davis, P.J. and P. Rabinowitz (1975): *Methods of Numerical Integration*, New York, Academic Press.
- [21] Dwight, H.B. (1958): *Mathematical Tables of Elementary and Some Higher Mathematical Functions*, 2nd edition Dover Publications Inc., New York, New York.
- [22] Eiselt, H.A., G. Pederzoli, C.-L. Sandbloom (1987): *Continuous Optimization Models*, Berlin; New York: W. de Gruyter.
- [23] Ermoliev, Y. (1983): Stochastic Quasigradient Methods and their Applications to Systems Optimization, *Stochastics* 9, 1-36.
- [24] Ermoliev, Y. and R.J-B Wets (eds.) (1988): *Numerical Techniques for Stochastic Optimization*, Springer Verlag.
- [25] Everitt, R. and W.T. Ziemba (1979): Two-Period Stochastic Programs with Simple Recourse, *Operations Research* 27, 485-502.
- [26] Feller, W. (1966): *An Introduction to Probability Theory and Its Applications, Volume II*, John Wiley & Sons Inc., second edition.
- [27] Feller, W. (1957): *An Introduction to Probability Theory and Its Applications, Volume I*, John Wiley & Sons Inc., third edition.
- [28] Frauendorfer, K. (1988): Solving SLP Recourse Problems with Arbitrary Multivariate Distributions - The Dependent Case, *Mathematics of Operations Research* 13, 377-394.
- [29] Gaivoronski, A. (1988): Stochastic Quasigradient Methods and their Implementation, in Y. Ermoliev and R.J-B Wets (eds.) *Numerical Techniques for Stochastic Optimization*, Springer Verlag.

- [30] S. Garstka and D. Rutenberg (1973): Computation in Discrete Stochastic Programs with Recourse, *Operations Research* 21, 112-122.
- [31] Gassmann, H.I. (1990): MSLiP: A Computer Code for the Multistage Stochastic Linear Programming Problem, *Mathematical Programming* 47, 407-423.
- [32] Gassmann, H.I. (1989): Optimal Harvest of a Forest in the Presence of Uncertainty, *Canadian Journal of Forest Research* 19, 1267-1274.
- [33] Glynn, P.W. and W. Whitt (1992): The Asymptotic Efficiency of Simulation Estimators, *Operations Research* 40, 505-520.
- [34] Higle, J.L. and S. Sen (1991): Stochastic Decomposition: An Algorithm for Two-Stage Linear Programs with Recourse, *Mathematics of Operations Research* 16, 650-669.
- [35] Infanger, G. (1992): Planning Under Uncertainty – Solving Large-Scale Stochastic Linear Programs, Technical Report SOL 92-8, Systems Optimization Laboratory, Department of Operations Research, Stanford University.
- [36] Infanger, G. (1992): Monte Carlo (Importance) Sampling within a Benders Decomposition Algorithm for Stochastic Linear Programs, *Annals of Operations Research* 39, 69-95.
- [37] Infanger, G. (1990): Monte Carlo (Importance) Sampling within a Benders Decomposition Algorithm for Stochastic Linear Programs, Technical Report SOL 89-13R, Systems Optimization Laboratory, Department of Operations Research, Stanford University.
- [38] Jacobs, J.M. (1992): Advanced Basis Insertion for Repeated Optimization of Networks with Side Constraints, Pacific Gas & Electric Company.
- [39] Jacobs, J., G. Freeman, J. Grygier, D. Morton, G. Schultz, K. Staschus, J. Stedinger, and B. Zhang (1993): Stochastic Optimal Coordination of River-basin and Thermal Electric Systems (SOCRATES): A System for Scheduling Hydroelectric Generation Under Uncertainty, submitted to *Annals of Operations Research*.
- [40] Kall, P. (1979): Computational Methods for Solving Two-Stage Stochastic Linear Programming Problems, *Journal of Applied Mathematics and Physics* 30, 261-271.
- [41] Kall, P. (1974): Approximations to Stochastic Programs with Complete Fixed Recourse, *Numerische Mathematic* 22, 333-339.
- [42] Kennington, J.L., R.V. Helgason (1980): *Algorithms for Network Programming*, John Wiley and Sons, New York, NY.
- [43] King, A.J., R. J-B. Wets (1989): Epi-Consistency of Convex Stochastic Programs, submitted to *Stochastics*.
- [44] Krishna, A.S. (1993): Techniques for Stochastic Programming, Dissertation in progress, Department of Operations Research, Stanford University.

- [45] Kusy, M.I., W.T. Ziemba (1986): A Bank Asset and Liability Management Model, *Operations Research*, 34, 356-376.
- [46] Kusy, M.I., W.T. Ziemba (1986): A Bank Asset and Liability Management Model, *Operations Research* 34, 356-376.
- [47] Loève, M. (1963): *Probability Theory*, D. Van Nostrand Company, Inc., Princeton, New Jersey, third edition.
- [48] Lustig, I.J, J.M. Mulvey, and T.J. Carpenter (1991): Formulating Two-Stage Stochastic Programs for Interior Point Methods, *Operations Research* 39, 757-770.
- [49] Mulvey, J.M., H. Vladimirov (1990): Applying the Progressive Hedging Algorithm to Stochastic Generalized Networks, Statistics and Operations Research Series Report SOR-89-9, Princeton University.
- [50] Mulvey, J.M., H. Vladimirov (1989): Stochastic Network Optimization Models for Investment Planning, *Annals of Operations Research: Network Optimization and Applications* 20, 187-217.
- [51] Pereira, M.V.F. and L.M.V.G. Pinto (1991): Multi-stage Stochastic Optimization Applied to Energy Planning, *Mathematical Programming* 52, 359-375.
- [52] Pereira, M.V.F. (1988): Optimal Stochastic Operations Scheduling of Large Hydroelectric Systems, Technical Report, CEPEL - Centro de Pesquisas de Energia Elétrica, Rio de Janeiro, Brazil.
- [53] Pereira, M.V.F., L.M.V.G. Pinto (1985): Stochastic Optimization of a Multireservoir Hydroelectric System: A Decomposition Approach, *Water Resources Research* 21, 779-792.
- [54] Prékopa, A. (1988): Numerical Solution of Probabilistic Constrained Programming Problems, in Y. Ermoliev and R.J-B Wets (eds.) *Numerical Techniques for Stochastic Optimization*, Springer Verlag.
- [55] Prékopa, A. (1973): Contributions to the Theory of Stochastic Programming, *Mathematical Programming* 4, 202-221.
- [56] Robbins, H.(1955): A Remark on Stirling's Formula, *American Mathematical Monthly* 62, 26-29.
- [57] Rockafellar, R.T. and R.J.-B. Wets (1991): Scenarios and Policy Aggregation in Optimization under Uncertainty, *Mathematics of Operations Research* 16, 119-147.
- [58] Scott, D.M. (1985): A Dynamic Programming Approach to Time-Staged Convex Programs, Technical Report SOL 85-3, Systems Optimization Laboratory, Department of Operations Research, Stanford University.
- [59] Strazicky, B. (1980): Some Results Concerning an Algorithm for the Discrete Recourse Problem, in M. Dempster (ed.) *Stochastic Programming*, Academic Press, London.
- [60] Szantai, T. (1988): A Computer Code for Solution of Probabilistic Constrained Stochastic Programming Problems, in Y. Ermoliev and R.J-B Wets (eds.) *Numerical Techniques for Stochastic Optimization*, Springer Verlag.

- [61] Van Slyke, R.M., R.J-B Wets (1969): L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming, *SIAM Journal of Applied Mathematics* 17, 638-663.
- [62] Wets, R. J-B (1988): Large Scale Linear Programming Techniques, in Y. Ermoliev and R.J-B Wets (eds.) *Numerical Techniques for Stochastic Optimization*, Springer Verlag.
- [63] Wets, R. J-B (1983): Solving Stochastic Programs with Simple Recourse, *Stochastics* 10, 219-242.
- [64] Wets, R. J-B (1983): Stochastic Programming: Solution Techniques and Approximation Schemes in A. Bachem, M. Grötschel and B. Korte (eds.) *Mathematical Programming: The State of the Art*, Springer Verlag (1983), 566-603.
- [65] Wets, R. J-B (1974): Stochastic Programs with Fixed Recourse: the Equivalent Deterministic Program, *SIAM Review* 16, 309-339.
- [66] Wittrock, R.J. (1983): Advances in a Nested Decomposition Algorithm for Solving Staircase Linear Programs, Technical Report SOL 83-2, Systems Optimization Laboratory, Department of Operations Research, Stanford University.
- [67] Wolfram, S. (1988): *Mathematica: A System for Doing Mathematics by Computer*, Addison-Wesley Publishing Company, Redwood City, California.
- [68] Ziemba, W.T. (1974): Stochastic Programs with Simple Recourse, in P. Hammer and G. Zoutendijk (eds.) *Mathematical Programming in Theory and Practice*, North-Holland, Amsterdam.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</small>				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE July 1993	3. REPORT TYPE AND DATES COVERED Technical Report	
4. TITLE AND SUBTITLE Algorithmic Advances In Stochastic Programming			5. FUNDING NUMBERS DE-FG03-92ER5116	
6. AUTHOR(S) David P. Morton				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Department of Operations Research - SOL Stanford University Stanford, CA 94305-4022			8. PERFORMING ORGANIZATION REPORT NUMBER 1111MA	
9. SPONSORING MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Energy Research U.S. Department of Energy Washington, DC 20585			10. SPONSORING / MONITORING AGENCY REPORT NUMBER SOL 93-6	
12a. DISTRIBUTION AVAILABILITY STATEMENT Unlimited			12b. DISTRIBUTION CODE UL	
13. ABSTRACT (Maximum 200 words) (Abstract on back page)				
14. SUBJECT TERMS Stochastic programming, Decomposition, Monte Carlo Sampling, Stopping rules, Large-Scale			15. NUMBER OF PAGES 115	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT	

ALGORITHMIC ADVANCES IN STOCHASTIC PROGRAMMING

Practical planning problems with deterministic forecasts of inherently uncertain parameters often yield unsatisfactory solutions. Stochastic programming formulations allow uncertain parameters to be modeled as random variables with known distributions, but the size of the resulting mathematical programs can be formidable. Decomposition-based algorithms take advantage of special structure and provide an attractive approach to such problems. We consider two classes of decomposition based stochastic programming algorithms. The first type of algorithm addresses problems with a "manageable" number of scenarios. The second class incorporates Monte Carlo sampling within a decomposition algorithm.

We develop and empirically study an enhanced Benders decomposition algorithm for solving multistage stochastic linear programs within a prespecified tolerance. The enhancements include warm start basis selection, preliminary cut generation, the multicut procedure, and decision tree traversing strategies. Computational results are presented for a collection of "real world" multistage stochastic hydroelectric scheduling problems.

Recently, there has been an increased focus on decomposition based algorithms that use sampling within the optimization framework. These approaches hold much promise for solving stochastic programs with many scenarios. A critical component of such algorithms is a stopping criterion to ensure the quality of the solution. With this as motivation, we develop a stopping rule theory for algorithms in which bounds on the optimal objective function value are estimated by sampling. Rules are provided for selecting sample sizes and terminating the algorithm under which asymptotic validity of confidence interval statements for the quality of the proposed solution can be verified. Issues associated with the application of this theory to two sampling based algorithms are considered, and preliminary empirical coverage results are presented.

END

DATE
FILMED

12/03/93