Assessing Policy Quality in Multi-stage Stochastic Programming

Anukal Chiralaksanakul and David P. Morton
Graduate Program in Operations Research
The University of Texas at Austin
Austin, TX 78712

January 2003

Abstract

Solving a multi-stage stochastic program with a large number of scenarios and a moderate-to-large number of stages can be computationally challenging. We develop two Monte Carlo-based methods that exploit special structures to generate feasible policies. To establish the quality of a given policy, we employ a Monte Carlo-based lower bound (for minimization problems) and use it to construct a confidence interval on the policy's optimality gap. The confidence interval can be formed in a number of ways depending on how the expected solution value of the policy is estimated and combined with the lower-bound estimator. Computational results suggest that a confidence interval formed by a tree-based gap estimator may be an effective method for assessing policy quality. Variance reduction is achieved by using common random numbers in the gap estimator.

1 Introduction

Multi-stage stochastic programming with recourse is a natural and powerful extension of multi-period deterministic mathematical programming. This class of stochastic programs can be effectively used for modeling and analyzing systems in which decisions are made sequentially and uncertain parameters are modeled via a stochastic process. The timing of making a decision and observing a realization of the uncertain parameters is a key feature of these models. At each stage, a decision, subject to certain constraints, must be made with information available up to that stage, while the future evolution of the stochastic process is known only through a conditional probability distribution. The goal is to find a solution that optimizes the expected value of a specified performance measure over a finite number of decision stages. A solution to a multi-stage stochastic program is defined by a *policy*,

which specifies what decision to take at each stage, given the history of the stochastic process up to that stage.

Multi-stage stochastic programming with recourse originated with Dantzig [10], and has been applied in a variety of fields ranging from managing financial systems, including asset allocation and asset-liability management, to operating hydro-thermal systems in the electric power industry to sizing and managing production systems. See, for example, Birge and Louveaux [3], Dupačová et al. [19], Dupačová [20], Kall and Wallace [36], Prékopa [48], Wallace and Ziemba [56], and Ziemba and Mulvey [59].

When the underlying random parameters have a continuous distribution, or finite support with many realizations, it is usually impossible to evaluate the expected performance measure exactly, even for a fixed solution. This is true for one- and two-stage stochastic programs. Computational difficulties are further compounded in the multi-stage setting, in which the stochastic program is defined on a scenario tree, and problem size grows exponentially with the number of stages. As a result, there is considerable interest in developing approximation methods for such stochastic programs.

Approximation methods for multi-stage stochastic programs often utilize exact decomposition algorithms that are designed to handle multi-stage problems with a moderate number of scenarios. We call an optimization algorithm "exact" if it can solve a problem within a numerical tolerance. Exact decomposition algorithms can be broadly divided into two types: those that decompose by stage and those that decompose by scenario. The L-shaped method for multi-stage stochastic linear programs [2, 25] is a by-stage decomposition scheme. One of the approximation methods we develop in this paper is based on a multi-stage L-shaped method. By-scenario decomposition algorithms include Lagrangian-based methods [44, 49].

When a multi-stage stochastic program is too large, due to the number of scenarios, to be solved exactly one may approximate the scenario tree to achieve a problem of manageable size. Schemes to do so based on probability metrics and moment matching are described in [9, 18, 32, 47]. Bound-based approximations of scenario trees exploit convexity with respect to the random parameters; see [5, 21, 22, 23].

Another type of approximation is based on Monte Carlo sampling, and these methods can be further categorized by whether the sampling is performed "inside" or "outside" the solution algorithm. Internal sampling-based methods replace computationally difficult exact evaluations with Monte Carlo estimates during the execution of the algorithm. For multi-stage stochastic linear programs, several variants of internal sampling-based L-shaped methods have been proposed. Pereira and Pinto [46] estimate the expected performance measure by sampling in the forward pass of the L-shaped method. Their algorithm can be applied to stochastic linear programs with interstage independence that have many stages but a manageable number of descendant scenarios at each node in the scenario tree. Linear minorizing functions, or cuts, on the expected performance measure are computed exactly in the backward pass, and can be shared among subproblems in the same stage due to interstage

independence. Donohue's [16] "abridged" version of this algorithm reduces the computational effort associated with each iteration. Chen and Powell [6] and Hindsberger and Philpott [31] have developed related algorithms. Convergence properties for this class of algorithms are addressed in Linowsky and Philpott [41]. Dantzig and Infanger [12, 34] employ importance sampling in both forward and backward passes of a multi-stage L-shaped method for stochastic linear programs with interstage independence and obtain considerable variance reduction. Importance sampling has also been used by Dempster and Thompson [15]. Higle, Rayco and Sen [27] propose a sampling-based cutting-plane algorithm applied to a dual formulation of a multi-stage stochastic linear program.

In external sampling-based methods, the underlying stochastic process is approximated through a finite empirical scenario tree constructed by Monte Carlo sampling. By solving the multi-stage stochastic program on this empirical sample tree an estimate of the expected performance measure is obtained. Under appropriate assumptions, strong consistency of the estimated optimal value is ensured [13, 16, 37, 52], i.e., as the number of samples at each node grows large, the estimated optimal value converges to the true value with probability one.

Under mild conditions, the estimated optimal value from the empirical scenario tree provides a lower bound, in expectation, on the true optimal value, and we show how to use this lower bound to establish the quality of a candidate solution policy. As indicated above, we emphasize that the solution to a multi-stage stochastic program is a policy. Shapiro [52] discusses the fact that simply fixing the first-stage decision in a multi-stage problem does not lead to a statistical upper bound. So, we propose two policy-generation methods that do.

Our first method for generating a policy applies to multi-stage stochastic linear programs with relatively complete recourse whose stochastic parameters exhibit interstage independence. This approach may be viewed as an external sampling-based procedure that employs the multi-stage L-shaped algorithm to solve the approximating problem associated with an empirical scenario tree to obtain approximate cuts. These cuts are then used to form a policy. Due to interstage independence, the approximate cuts can be shared among the subproblems in the same stage. We also indicate how this method can be extended to handle a particular type of interstage dependency through cut-sharing formulae from [35]. The second policy-generation method we consider is computationally more expensive but applies to a more general class of multi-stage stochastic programs with recourse.

The value of using a lower bound to establish solution quality for a minimization problem is widely recognized in optimization. In the context of employing Monte Carlo sampling techniques in stochastic programming, exact lower bounds are not available; instead, lower bounds are statistical in nature. The type of lower bound we use in this paper has been analyzed and utilized before, mostly in one- or two-stage problems. Mak, Morton, and Wood [42] use a lower-bound estimator to construct a confidence interval on the optimality gap to assess the quality of a candidate solution for two-stage stochastic programs. Linderoth, Shapiro, and Wright [40] and Verweij et al. [55] report encouraging

computational results for this type of approach on different classes of two-stage stochastic programs. Norkin, Pflug, and Ruszczyński [45] develop a stochastic branch-and-bound procedure for discrete problems in which lower bound estimators are used in an internal fashion for pruning the search tree. Methods for assessing solution quality in the context of the stochastic decomposition method for two-stage stochastic linear programs, due to Higle and Sen [28], are discussed in [30] and a statistical bound based on duality is developed in [29].

The purpose of the current paper is to extend methods for testing solution quality to the multi-stage setting. Broadie and Glasserman [4] establish confidence intervals on the value of a Bermudan option, a multi-stage problem, using Monte Carlo bounds. Shapiro [52] examines lower bounding properties and consistency of sampling-based bounds for multi-stage stochastic linear programs. Another view of establishing solution quality lies in analyzing the sensitivity of the solution to changes in the probability distribution. There is a significant literature concerning stability results in stochastic programming and it is not our purpose to review it. We point only to the approach of Dupačová [17], which is applicable in the multi-stage setting and lends itself to computing bounds on the optimality gap when the original distribution is "contaminated" by another.

The remainder of the paper is organized as follows. Section 2 covers preliminaries: the class of multi-stage stochastic programs we consider along with the linear programming special case, sample scenario-tree generation, and a brief review of a multi-stage version of the L-shaped decomposition method. This decomposition method plays a central role in the policy generation method discussed in Section 3.1 for linear problems with interstage independence, or with a special type of interstage dependence. Section 3.2 details the second policy generation method, which applies to our more general class of problems. Estimating the expected cost of using a specific policy is discussed in Section 4. A statistical lower bound on the optimal objective function value is developed in Section 5. Procedures for constructing confidence intervals on the optimality gap of a given policy are described in Section 6, and associated computational results are reported in Section 7. Conclusions and extensions are given in Section 8.

2 Preliminaries

2.1 Problem Statement

We consider a T-stage stochastic program in which a sequence of decisions, $\{x_t\}_{t=1}^T$, is made with respect to a stochastic process, $\{\tilde{\xi}_t\}_{t=1}^T$, as follows: at stage t, the decision $x_t \in \mathbb{R}^{d_t}$ is made with only the knowledge of past decisions, $x_1, \ldots, x_{t-1}$, and of realized random vectors, $\xi_1, \ldots, \xi_t$, such that the conditional expected value of an objective function, $\phi_t(x_1, \ldots, x_t, \tilde{\xi}_1, \ldots, \tilde{\xi}_{t+1})$, given the history, $\xi_1, \ldots, \xi_t$, is minimized. Decision x_t is subject to constraints that may depend on $x_1, \ldots, x_{t-1}$ and $\xi_1, \ldots, \xi_t$. Throughout we refer to a realization of the random variable, $\tilde{\xi}_t$, as ξ_t. The requirement that

decision x_t not depend on future realizations of $\tilde{\xi}_{t+1}, \ldots, \tilde{\xi}_T$ is known in the stochastic programming literature as nonanticipativity, and is enforced by ensuring that x_t be measurable with respect to the stage t sigma-algebra generated by realizations of the stochastic process through stage t. In our notation, although ϕ_t depends on random vectors $\tilde{\xi}_1, \ldots, \tilde{\xi}_{t+1}$, the history of the process up to stage t is known and fixed through the conditional expectation.

We assume that $\tilde{\xi}_1$ is a degenerate random vector taking value ξ_1 with probability one, and that the distribution governing the evolution of $\{\tilde{\xi}_t\}_{t=1}^T$ is known and does not depend on $\{x_t\}_{t=1}^T$. A superscript t on an entity denotes its history through stage t, e.g., $\xi^t = (\xi_1, \ldots, \xi_t)$ and $x^t = (x_1, \ldots, x_t)$. Let Ξ_t be the support of $\tilde{\xi}_t$ and Ξ^t be that of $\tilde{\xi}^t, t = 1, \ldots, T$. The conditional distribution of $\tilde{\xi}_{t+1}$ given $\tilde{\xi}^t = \xi^t$ is denoted $F_{t+1}(\xi_{t+1}|\xi^t)$. A T-stage stochastic program can be expressed in the following form:

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

where

$$\begin{aligned} \phi_{t-1}(x^{t-1}, \tilde{\xi}^t) = \min_{x_t} \;\; & E[\phi_t(x^{t-1}, x_t, \tilde{\xi}^{t+1})|\tilde{\xi}^t] \\ \text{s.t.} \;\; & x_t \in X_t(x^{t-1}, \tilde{\xi}^t), \end{aligned} \tag{2}$$

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

$$\begin{aligned} \phi_{T-1}(x^{T-1}, \tilde{\xi}^T) = \min_{x_T} \;\; & \phi_T(x^{T-1}, x_T, \tilde{\xi}^T) \\ \text{s.t.} \;\; & x_T \in X_T(x^{T-1}, \tilde{\xi}^T). \end{aligned} \tag{3}$$

Stochastic program (1)-(3) is a relatively general class of multi-stage stochastic programs, and includes an important class of linear models that we describe later in this section.

A solution of (1)-(3) is specified by a policy, which may be viewed as a mapping, $x_t(\xi^t)$, with domain Ξ^t and range in $\mathbb{R}^{d_t}$, $t = 1, \ldots, T$. Restated, a policy is a rule which specifies what decision to take at each stage t of a multi-stage stochastic program for each possible realization of $\tilde{\xi}^t$ in Ξ^t, $t = 1, \ldots, T$. We only consider policies that satisfy the nonanticipativity requirement, i.e., x_t can only depend on ξ^t and not on subsequent realizations of the random parameters. A policy $\hat{x}^T(\xi^T) = (\hat{x}_1(\xi^1), \ldots, \hat{x}_T(\xi^T))$, is said to be feasible to (1)-(3) if it is nonanticipative, $\hat{x}_1(\tilde{\xi}^1) \in X_1(\tilde{\xi}^1)$, and $\hat{x}_t(\tilde{\xi}^t) \in X_t(\hat{x}^{t-1}(\tilde{\xi}^{t-1}), \tilde{\xi}^t)$, wp1, where $\tilde{\xi}^t = (\tilde{\xi}^{t-1}, \tilde{\xi}_t), t = 2, \ldots, T$. We make the following assumptions:

(A1) (1)-(3) has relatively complete recourse, and $X_1(\xi^1)$ is non-empty.

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

(A3) $E[\phi_t(x^t, \tilde{\xi}^{t+1})|\tilde{\xi}^t]$ is lower semi-continuous in x^t, wp1, $t = 1, \ldots, T-1$, and $\phi_T(x^T, \tilde{\xi}^T)$ is lower semi-continuous in x^T, wp1.

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

Feasibility of (1)-(3) is guaranteed by (A1). Attainment of the minimum (infimum) in each stage results from compactness of the feasible region in (A2) and lower semi-continuity of the objective function in (A3). The stronger assumption of continuity in place of (A3) is a natural assumption for multi-stage stochastic linear programs, but lower semi-continuity can arise when considering integer-constrained problems. The need for the finite second moment assumption in (A4) will arise when we use the central limit theorem in confidence interval construction.

As we now argue, a sufficient condition to ensure (A3) is that $\phi_T(x^T, \tilde{\xi}^T)$ is lower semi-continuous in x^T, wp1, and

(A3′) there exists $C_T(\cdot)$ with $\phi_T(x^T, \tilde{\xi}^T) \geq C_T(\tilde{\xi}^T)$ for all feasible x^T, wp1, where $E|C_T(\tilde{\xi}^T)| < \infty$.

Using (3) and (A3′) we have $\phi_{T-1}(x^{T-1}, \tilde{\xi}^T) \geq C_T(\tilde{\xi}^T)$, and then using (2) and $E|C_T(\tilde{\xi}^T)| < \infty$ we have, for $t = 1, \ldots, T-2$,

$$\phi_t(x^t, \tilde{\xi}^{t+1}) \geq \underbrace{E\left[C_T(\tilde{\xi}^T) \,|\, \tilde{\xi}^{t+1}\right]}_{\equiv C_t(\tilde{\xi}^{t+1})}, \text{where } E\left[\left|C_t(\tilde{\xi}^{t+1})\right| \,|\, \tilde{\xi}^t\right] < \infty, \text{ wp1.} \tag{4}$$

Then, lower semi-continuity of $E[\phi_t(x^t, \tilde{\xi}^{t+1})|\tilde{\xi}^t], t = 1, \ldots, T-1$, in (A3) is guaranteed via an induction argument which involves the following results:

(i) Lower semi-continuity of $E[\phi_{t+1}(x^{t+1}, \tilde{\xi}^{t+2})|\tilde{\xi}^{t+1}]$ in x^{t+1}, wp1, and compactness of $X_{t+1}(x^t, \tilde{\xi}^{t+1})$ ensure lower semi-continuity of $\phi_t(x^t, \tilde{\xi}^{t+1})$, wp1. (See Rockafellar and Wets [50, Theorem 1.17].)

(ii) Lower semi-continuity of $\phi_t(x^t, \tilde{\xi}^{t+1})$ and $E\left[\left|\phi_t(x^t, \tilde{\xi}^{t+1})\right||\tilde{\xi}^t\right] < \infty$, wp1, coupled with (4), ensure lower semi-continuity of $E[\phi_t(x^t, \tilde{\xi}^{t+1})|\tilde{\xi}^t]$, wp1. (See Wets [57, Proposition 2.2].)

(Note that the finite expectation hypothesis in (ii) follows from (A4).)

Lower semi-continuity is also preserved under the expectation operator in (ii) when $\phi_t(x^t, \tilde{\xi}^{t+1})$ is convex in x^t (again, see Wets [57, Proposition 2.2]). Therefore, an alternative to (A3′) for ensuring (A3) is to assume that $\phi_T(x^T, \tilde{\xi}^T)$ is lower semi-continuous in x^T, wp1, and

(A3″) $\phi_t(x^t, \tilde{\xi}^{t+1})$ is convex in x^t, wp1, $t = 1, \ldots, T-1$.

In sum, either (A3′) or (A3″), coupled with lower semi-continuity of $\phi_T(x^T, \tilde{\xi}^T)$ in x^T, is sufficient to ensure lower semi-continuity of $E[\phi_t(x^t, \tilde{\xi}^{t+1})|\tilde{\xi}^t]$ in x^t, wp1, $t = 1, \ldots, T-1$, in (A3).

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

$$f_t(x^t, \xi^{t+1}) = \begin{cases} \phi_t(x^t, \xi^{t+1}) & \text{if } x_t \in X_t(x^{t-1}, \xi^t) \\ \infty & \text{otherwise,} \end{cases} \tag{5}$$

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

$$f_T(x^T, \xi^T) = \begin{cases} \phi_T(x^T, \xi^T) & \text{if } x_T \in X_T(x^{T-1}, \xi^T) \\ \infty & \text{otherwise.} \end{cases} \tag{6}$$

(1)-(3) can now be re-stated as an unconstrained optimization problem:

$$z^* = \min_{x_1} \; E[f_1(x_1, \tilde{\xi}^2)|\tilde{\xi}^1], \tag{7}$$

where

$$f_{t-1}(x^{t-1}, \tilde{\xi}^t) = \min_{x_t} \; E[f_t(x^{t-1}, x_t, \tilde{\xi}^{t+1})|\tilde{\xi}^t], \tag{8}$$

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

$$f_{T-1}(x^{T-1}, \tilde{\xi}^T) = \min_{x_T} \; f_T(x^{T-1}, x_T, \tilde{\xi}^T). \tag{9}$$

An important special case of (1)-(3) is a multi-stage stochastic linear program with recourse in which the objective function has an additive contribution from each stage and the underlying optimization problems are linear programs. A T-stage stochastic linear program can be expressed in the following form:

$$\begin{aligned} \min_{x_1} \; & c_1 x_1 + E[h_1(x_1, \tilde{\xi}^2)|\tilde{\xi}^1] \\ \text{s.t.} \; & A_1 x_1 = b_1 \\ & x_1 \geq 0, \end{aligned} \tag{10}$$

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

$$\begin{aligned} h_{t-1}(x_{t-1}, \tilde{\xi}^t) = \min_{x_t} \; & \tilde{c}_t x_t + E[h_t(x_t, \tilde{\xi}^{t+1})|\tilde{\xi}^t] \\ \text{s.t.} \; & \tilde{A}_t x_t = \tilde{b}_t - \tilde{B}_t x_{t-1} \\ & x_t \geq 0, \end{aligned} \tag{11}$$

and $h_T = 0$. The random vector $\tilde{\xi}_t$ consists of the random elements from $(\tilde{A}_t, \tilde{B}_t, \tilde{b}_t, \tilde{c}_t)$. The dimensions of vectors and matrices are as follows: $c_t \in \mathbb{R}^{1 \times d_t}$, $A_t \in \mathbb{R}^{m_t \times d_t}$, $B_t \in \mathbb{R}^{m_t \times d_{t-1}}$, and $b_t \in \mathbb{R}^{m_t}, t = 1, \ldots, T$. We now return to assumptions (A1)-(A4) and describe sufficient conditions in a linear programming context to ensure (A1)-(A4). Relatively complete recourse carries over naturally to the constraints of (10) and (11) and is assumed to hold. We assume that the feasible

region of (10) is nonempty and bounded and that of (11) is bounded for all feasible x_{t-1}, wp1; hence, (A1) and (A2) hold. (A3) is ensured by convexity of $h_t(x_t, \tilde{\xi}^{t+1})$ in x_t, wp1. Finally, we assume that the distribution of $\tilde{\xi}^T$ is such that (A4) holds.

Realizations of $\{\tilde{\xi}_t\}_{t=1}^T$ form a scenario tree that represents all possible ways that $\{\tilde{\xi}_t\}_{t=1}^T$ can evolve, and organizes the realizations of the sequence $\{\tilde{\xi}_t\}_{t=1}^T$ with common sample paths up to stage t. From a computational perspective, we limit ourselves to finite scenario trees.

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

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

$$k = \sum_{r=1}^{i-1} n(t,r), \tag{12}$$

and $\sum_{r=1}^{0} \equiv 0$. The subvector of $\xi^{t+1,j}, j \in D_t^i$, that corresponds to the stage $t+1$ realization is $\xi_{t+1}^j, j \in D_t^i$. The ancestor of $\xi^{t,i}$ is denoted $\xi^{t-1,a(i)}$. In this case, $a(i)$ is an integer between 1 and n_{t-1}. With our notation, $a(j) = i, \forall j \in D_t^i$. The total number of nodes in each stage can be recursively computed from

$$n_t = \sum_{r=1}^{n_{t-1}} n(t-1,r), \quad \text{for } t = 2, \ldots, T, \tag{13}$$

where $n_1 \equiv 1$. Note that $D_t^i \bigcap D_t^{i'} = \emptyset$ for $i, i' \in \{1, \ldots, n_t\}$ and $i \neq i'$, and $\bigcup_{i=1}^{n_{t-1}} D_{t-1}^i = \{1, \ldots, n_t\}$ for $t = 2, \ldots, T$.

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

We illustrate our notation by applying it to the four-stage scenario tree in Figure 1. The root node R corresponds to the unique stage 1 realization ξ^1. Table 1 gives examples of the history notation and the number of immediate descendants for nodes A, ..., G. The subtree with its root at node A is represented by $\Gamma(\xi^{2,1})$ and its branches are darkened in Figure 1. The index set of the immediate

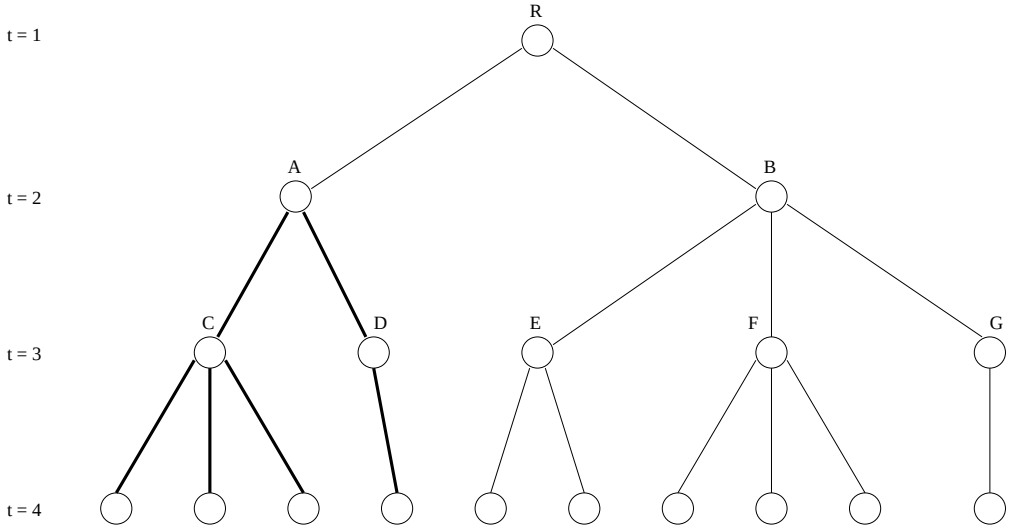

Figure 1: An example of a four-stage scenario tree.

descendants of node B is $D_2^2 = \{3, 4, 5\}$, and the corresponding stage 3 realizations are ξ_3^3, ξ_3^4, and ξ_3^5. We have $n_2 = n(1, 1) = 2$ and $n_3 = \sum_{r=1}^{2} n(2, r) = 2 + 3 = 5$. We refer to a generic node in the second stage, either A or B, by ξ^2, and a generic subtree rooted at ξ^2 by $\Gamma(\xi^2)$.

Table 1: Notation for the scenario tree in Figure 1

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

By using the notation introduced, we can write (10)-(11), when $\{\tilde{\xi}_t\}_{t=1}^T$ has finite support, as follows:

$$\begin{aligned} \min_{x_1} \ & c_1 x_1 + \sum_{k \in D_1^1} p_2^{k|1} h_1(x_1, \xi^{2,k}) \\ \text{s.t.} \ & A_1 x_1 = b_1 \\ & x_1 \geq 0, \end{aligned} \tag{14}$$

where for all $j = 1, \ldots, n_t,\ t = 2, \ldots, T$,

$$\begin{aligned} h_{t-1}(x_{t-1}, \xi^{t,j}) = \min_{x_t} \ & c_t^j x_t + \sum_{k \in D_t^j} p_{t+1}^{k|j} h_t(x_t, \xi^{t+1,k}) \\ \text{s.t.} \ & A_t^j x_t = b_t^j - B_t^j x_{t-1} \\ & x_t \geq 0, \end{aligned} \tag{15}$$

where $\xi^{t+1,k} = (\xi^{t,j}, \xi_{t+1}^k)$, $k \in D_t^j$, and $h_T = 0$. The conditional mass function is defined as

$$p_{t+1}^{k|j} = P(\tilde{\xi}_{t+1} = \xi_{t+1}^k | \tilde{\xi}^t = \xi^{t,j}), \ \ k \in D_t^j,$$

and the stage t marginal mass function is $p_t^i = P(\tilde{\xi}^t = \xi^{t,i})$, $i = 1, \ldots, n_t$. Note that $p_{T+1}^{j|i} = 0,\ \forall i, j$. We will use this formulation when we review the multi-stage L-shaped method in Section 2.3.

2.2 Sample Scenario Tree Construction

To construct a sample scenario tree, we perform the sampling in the following conditional fashion: we begin by drawing $n(1,1) = n_2$ observations of $\tilde{\xi}_2$ from $F_2(\xi_2|\xi^1)$ where ξ^1 is the known first stage realization. Then, we form the descendants of each observation $\xi^{2,i}, i = 1, \ldots, n_2$, by drawing $n(2,i)$ observations of $\tilde{\xi}_3$ from $F_3(\xi_3|\xi^{2,i})$. This process continues until we have sampled $n(T-1,i)$ observations of $\tilde{\xi}_T$ from $F_T(\xi_T|\xi^{T-1,i}),\ i = 1, \ldots, n_{T-1}$. The notation developed in Section 2.1 for a general finite scenario tree applies to a sample scenario tree. The number of descendants of a node $\xi^{t,i}$ is now determined by the sample size $n(t,i)$. The total number of nodes in stage $t+1$ is $n_{t+1} = \sum_{r=1}^{n_t} n(t,r)$, and $n(t) = |D_t|$ is the number of immediate descendants of a generic stage t node, ξ^t. The subtree associated with each descendant of node $\xi^{t,i}$ is $\Gamma(\xi^{t+1,j}), j \in D_t^i$.

In addition to the above structure for constructing a sample scenario tree, we require for the purposes of the estimators developed in Section 4 that the samples of $\tilde{\xi}_{t+1}$ be drawn from $F_{t+1}(\xi_{t+1}|\xi^t)$ so that they satisfy the following unbiasedness condition

$$E[f_t(x^t, \tilde{\xi}^{t+1})|\tilde{\xi}^t] = E[\frac{1}{n(t)} \sum_{i \in D_t} f_t(x^t, \tilde{\xi}^{t+1,i})|\tilde{\xi}^t], \tag{16}$$

wp1, $t = 1, \ldots, T-1$. The simplest method for generating $\tilde{\xi}_{t+1}^i, i \in D_t$, to satisfy (16) is to require that they be (conditionally) independent and identically distributed (iid), but other methods, including some variance reduction schemes that have been used in stochastic programming (see, e.g., [1, 11, 26, 33, 40]), also satisfy (16).

Within the conditionally iid framework there are different types of sample scenario trees that can be generated. Consider the case when $\{\tilde{\xi}_t\}_{t=1}^T$ is interstage independent. One possibility is to generate a single set of iid observations of $\tilde{\xi}_{t+1}$ and use this same set of descendants for all stage t nodes $\xi^{t,i}, i = 1, \ldots, n_t$. Another possibility is to generate mutually independent sets of stage $t+1$

descendant nodes for all stage t nodes. We say the former method uses "common samples" and the latter "independent samples." Both methods of generating a scenario tree satisfy (16). The independent-samples method introduces interstage dependency in the sample tree, which was not present in the original tree while the common-samples method preserves interstage independence. Another advantage of the common-samples approach (relative to an independent-samples tree) is that the associated stochastic program lends itself to the solution procedures of [6, 16, 31, 46]. On the other hand, because of increased diversity in the sample, one might expect solutions under the independent-samples tree to have lower variability.

When using the common-samples approach the number of descendant nodes within each stage must be identical but the cardinality of D_t could vary with stage. In the independent-samples approach, we have freedom to select different sample sizes at each node in the scenario tree. Dempster and Thompson [15] use the expected value of perfect information to guide sample tree construction. Korapaty [38] and Chiralaksanakul [8] select the cardinality of descendant sets to reduce bias. Provided that sampling is done in the conditional manner described above, with (16) satisfied, the methods we develop here can be applied to trees with non-constant sizes of descendant sets. That said, in our computation (Section 7) we restrict attention to uniform sample trees, i.e., $n(t,i) = |D_t^i|$ is constant for all i and t.

Given an empirical, i.e., sampled, scenario tree an approximating problem for (7)-(9) can be stated as

$$\hat{z}^* \quad = \quad \min_{x_1} \ \frac{1}{n(1,1)} \sum_{i \in D_1^1} \hat{f}_1(x_1, \tilde{\xi}^1, \Gamma(\tilde{\xi}^{2,i})) \tag{17}$$

where

$$\hat{f}_{t-1}(x^{t-1}, \tilde{\xi}^{t-1}, \Gamma(\tilde{\xi}^{t,j})) = \min_{x_t} \ \frac{1}{n(t,j)} \sum_{i \in D_t^j} \hat{f}_t(x^{t-1}, x_t, \tilde{\xi}^{t,j}, \Gamma(\tilde{\xi}^{t+1,i})), \tag{18}$$

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

$$\begin{aligned} \hat{f}_{T-1}(x^{T-1}, \tilde{\xi}^{T-1}, \Gamma(\tilde{\xi}^{T,j})) \quad &= \quad f_{T-1}(x^{T-1}, \tilde{\xi}^{T,j}) \\ &= \quad \min_{x_T} \ f_T(x^{T-1}, x_T, \tilde{\xi}^{T,j}), \end{aligned} \tag{19}$$

$\tilde{\xi}^{T,j} = (\tilde{\xi}^{T-1}, \tilde{\xi}_T^j)$, $j \in D_{T-1}$. The value function at a stage t node ξ^t depends on the stochastic history (known at time t), $\tilde{\xi}^t = \xi^t$, the associated decision history, x^t, and the sample subtree $\Gamma(\xi^t)$. In going from (7)-(9) to (17)-(19), we are approximating the original population scenario tree by a sample scenario tree.

One of the policy-generation methods we develop is for multi-stage stochastic linear programs and

so we explicitly state the associated approximating problem of (10)-(11):

$$\begin{aligned} \min_{x_1} \;& c_1 x_1 + \frac{1}{n(1,1)} \sum_{k \in D_1^1} \hat{h}_1(x_1, \tilde{\xi}^1, \Gamma(\tilde{\xi}^{2,k})) \\ \text{s.t.} \;& A_1 x_1 = b_1 \\ & x_1 \geq 0, \end{aligned} \tag{20}$$

where for all $j = 1, \ldots, n_t,\ t = 2, \ldots, T$,

$$\begin{aligned} \hat{h}_{t-1}(x_{t-1}, \tilde{\xi}^{t-1}, \Gamma(\tilde{\xi}^{t,j})) = \min_{x_t} \;& c_t^j x_t + \frac{1}{n(t,j)} \sum_{k \in D_t^j} \hat{h}_t(x_t, \tilde{\xi}^{t,j}, \Gamma(\tilde{\xi}^{t+1,k})) \\ \text{s.t.} \;& A_t^j x_t = b_t^j - B_t^j x_{t-1} \\ & x_t \geq 0, \end{aligned} \tag{21}$$

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

2.3 The Multi-stage L-shaped Method

In this section we briefly review the multi-stage version of the L-shaped method. The method was originally developed by Van Slyke and Wets [54] for two-stage stochastic linear programs, and was later extended to multi-stage programs by Birge [2]. It is an effective solution method for such problems [20, 51] and plays a central role in the policy generation procedure we discuss in Section 3.1. The multi-stage L-shaped method decomposes (14)-(15) by stage and then separates stage-wise problems by scenario to achieve a subproblem at each node $\xi^{t,i}$, denoted $\text{sub}(t,i)$, $i = 1, \ldots, n_t$, $t = 1, \ldots, T-1$, of the following form:

$$\begin{aligned} \min_{x_t, \theta_t} \;& c_t^i x_t + \quad \theta_t & & \\ \text{s.t.} \;& A_t^i x_t & = \; b_t^i - B_t^i x_{t-1}^{a(i)} & : \pi_t \\ & -\vec{G}_t^i x_t + e\,\theta_t & \geq \; \vec{g}_t^{\,i} & : \alpha_t \\ & x_t & \geq \; 0. & \end{aligned} \tag{22}$$

The rows of the matrix $\vec{G}_t^i$ contain cut gradients; the elements of the vector $\vec{g}_t^{\,i}$ are cut intercepts; and, e is the vector of all 1's. π_t and α_t are dual row vectors associated with each set of constraints. For $t = T$, the subproblems are similar to (22) except that there are no cut constraints and no variable θ_T. To compute the cut gradient and intercept in $\text{sub}(t,i)$, all the descendants of $\text{sub}(t,i)$ are solved at a given stage t decision, x_t, to obtain $(\pi_{t+1}^j, \alpha_{t+1}^j), j \in D_t^i$. Then, the cut gradient is

$$G_t^i = -\sum_{j \in D_t^i} p_{t+1}^{j|i} \pi_{t+1}^j B_{t+1}^j, \tag{23}$$

and the cut intercept is

$$g_t^i = \sum_{j \in D_t^i} p_{t+1}^{j|i} \pi_{t+1}^j b_{t+1}^j + \sum_{j \in D_t^i} p_{t+1}^{j|i} \alpha_{t+1}^j \vec{g}_{t+1}^j, \tag{24}$$

where the second term on the right-hand side of (24) is absent if $t = T - 1$. For sub(t, i), the rows of the matrix $\vec{G}_t^i$ are composed of the cut gradient row vectors, G_t^i, and the components of the vector $\vec{g}_t^i$ are composed of the cut intercepts, g_t^i. An algorithmic statement of the multi-stage L-shaped method using the so-called fastpass tree traversal strategy is given in Figure 2. In the fastpass strategy, an optimal solution from each subproblem is passed to its descendants until the last stage is reached, and then the cuts formed by the descendants at each stage are passed back up to the corresponding ancestor subproblems. Other tree-traversal strategies are also possible but empirical evidence appears to support the use of the fastpass strategy [25, 43, 58].

Step 0	Define $toler \geq 0$ and let $\overline{z} = \infty$.
	Initialize the set of cuts for sub(t, i) with $\theta_t \geq -M$, $i = 1, \ldots, n_t$, for $t = 1, \ldots, T-1$. (M sufficiently large.)
Step 1	Solve sub$(1, 1)$ and let (x_1, θ_1) be its solution.
	Let $\underline{z} = c_1 x_1 + \theta_1$.
Step 2	Do $t = 2$ to T
	Do $i = 1, \ldots, n_t$
	Form the right-hand side of sub(t, i): $b_t^i - B_t^i x_{t-1}^{a(i)}$.
	Solve sub(t, i). Let x_t^i be its solution.
	If $t = T$, let π_T^i be the optimal dual vector.
	Let $\hat{z} = c_1 x_1 + \sum_{t=2}^{T} \sum_{i=1}^{n_t} p_t^i c_t^i x_t^i$.
Step 3	If $\hat{z} < \overline{z}$ then let $\overline{z} = \hat{z}$ and $x_t^{i,*} = x_t^i$, $\forall i, t$.
	If $\overline{z} - \underline{z} \leq \min(\lvert\overline{z}\rvert, \lvert\underline{z}\rvert) \cdot toler$ then stop: $x_t^{i,*}, \forall i, t$ is a policy with objective function value within $100 \cdot toler\%$ of optimal.
Step 4	Do $t = T - 1$ downto 2
	Do $i = 1, \ldots, n_t$
	Form (G_t^i, g_t^i).
	Augment sub(t, i)'s set of cuts with $-G_t^i x_t + \theta_t \geq g_t^i$.
	Form the right-hand side of sub(t, i): $b_t^i - B_t^i x_{t-1}^{a(i)}$.
	Solve sub(t, i). Let (π_t^i, α_t^i) be the optimal dual vector.
	Form (G_1^1, g_1^1).
	Augment sub$(1, 1)$'s set of cuts with $-G_1^1 x_1 + \theta_1 \geq g_1^1$.
	Goto Step 1.

Figure 2: The multi-stage L-shaped algorithm using the fastpass tree traversal strategy for a T-stage stochastic linear program.

3 Two Policy Generation Methods

3.1 Linear Problems with Interstage Independence

In this section, we develop a procedure to generate a feasible policy for the multi-stage stochastic linear program (14)-(15) when $\{\tilde{\xi}_t\}_{t=1}^T$ is interstage independent. Our method works as follows: First, we construct a sample scenario tree, denoted Γ_c, using the common-samples method described in Section 2.2. Then, the instance of (20)-(21) associated with Γ_c is solved with the multi-stage L-shaped algorithm of Figure 2 (the "c" subscript on Γ stands for "cuts"). When the algorithm stops, we obtain a policy whose expected cost is within $100 \cdot toler\%$ of optimal for (20)-(21). We now describe how we use this solution to obtain a policy for the "true" problem (10)-(11).

When the algorithm of Figure 2 terminates, each $\text{sub}(t,i)$ contains the set of cut constraints generated during the solution procedure. Since Γ_c is constructed with the common-samples scheme, the sample subtrees rooted at the stage t nodes are all identical, i.e., the sample scenario tree Γ_c exhibits interstage independence. Thus, the cuts generated for a stage t node are valid for all other nodes in stage t. We will use the collection of cuts at each stage to construct a policy to problem (10)-(11).

Let $\vec{G}_{t,c}^i$ and $\vec{g}_{t,c}^i$ denote the cut-gradient matrix and cut-intercept vector for $\text{sub}(t,i)$ when the multi-stage L-shaped method terminates. Then, we define a stage t optimization problem used to generate the policy for (10)-(11) as follows:

$$\begin{aligned} \min_{x_t} \ & c_t x_t + \theta_t \\ \text{s.t.} \ & A_t x_t = b_t - B_t x_{t-1} \\ & -\vec{G}_{t,c}^i x_t + e\,\theta_t \geq \vec{g}_{t,c}^i, \ i = 1, \ldots, n_t \\ & x_t \geq 0, \end{aligned} \tag{25}$$

for $t = 2, \ldots, T$. For $t = 1$, (25) does not contain the term $B_1 x_0$ in the first set of constraints, and for $t = T$ the cut constraints are absent. A policy must specify what decision, $\hat{x}_t(\xi^t)$, to take at each stage t for a given ξ^t. Our policy computes $\hat{x}_t(\xi^t)$ by solving (25) with (A_t, B_t, b_t, c_t) specified by ξ^t, and with x_{t-1} determined by having already solved (25) under subvectors of ξ^t corresponding to the preceding stages. Such a policy is nonanticipative because when solving (25) the process $\{\tilde{\xi}_t\}_{t=1}^T$ is known only through stage t. Relatively complete recourse ensures that $\hat{x}_t(\xi^t)$ will lead to a feasible decision in stages $t+1, \ldots, T$. The superscript on the cut-gradient matrix and the cut-intercept vector in (25) denotes the index of the stage t node in Γ_c from which we obtain the cuts, and n_t is the total number of stage t nodes in Γ_c. So, if $\text{sub}(t,i)$ in Γ_c has K_t^i cuts then the total number of cuts in (25) is $\sum_{i=1}^{n_t} K_t^i$. We refer to this procedure as $\mathcal{P}_1$ and summarize it in Figure 3.

The solution procedure, as we have described it above, is a naive version of the multi-stage L-shaped method because it stores a separate set of cuts at each $\text{sub}(t,i)$ when solving (20)-(21) under

Step 1	Construct a sample scenario tree Γ_c with the common-samples procedure (Section 2.2).
Step 2	Solve (20)-(21) based on Γ_c with the multi-stage L-shaped algorithm (Figure 2).
Step 3	When the algorithm stops (Step 3 of Figure 2), store the cut-gradient matrix, $\vec{G}^i_{t,c}$, and the cut-intercept vector, $\vec{g}^i_{t,c}$, associated with each sub(t,i), $\forall t, i$.
Step 4	Given sample path ξ^T, Do $t = 1$ to T Solve optimization problem (25) under ξ_t with x_{t-1} equal to $\hat{x}_{t-1}(\xi^{t-1})$, and denote its optimal solution $\hat{x}_t(\xi^t)$, where $\xi^t = (\xi^{t-1}, \xi_t)$.

Figure 3: Procedure $\mathcal{P}_1$ to generate a feasible policy for a T-stage stochastic linear program with relatively complete recourse when $\{\tilde{\xi}_t\}_{t=1}^T$ is interstage independent.

Γ_c. Because Γ_c is interstage independent, we instead store a single set of cuts at each stage. This speeds the solution process and aids in eliminating redundant cuts when forming (25).

We have described the method for generating cuts at each stage by solving (20)-(21) under Γ_c exactly (or within $100 \cdot toler\%$) using the algorithm of Figure 2. However, this may be computationally expensive to carry out if Γ_c is large. If T is large but the number of descendants at each stage t node is "manageable" then we could instead employ one of the sampling-based algorithms designed for such problems [6, 16, 31, 46].

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

$$\tilde{b}_t = \sum_{j=1}^{t-1} (R_j^t \tilde{b}_j + S_j^t \tilde{\eta}_j) + \tilde{\eta}_t, \ t = 2, \ldots, T. \tag{26}$$

Here, R_j^t and S_j^t are given deterministic matrices with appropriate dimensions. Series (26) is a generalization of a vector ARMA (autoregressive moving average) model; see, e.g., Tiao and Box

[53]. With this probabilistic structure, Infanger and Morton [35] derive cut sharing formulae to be used in the L-shaped method. These results can be applied to modify Step 3 and 4 of $\mathcal{P}_1$. In Step 3, we store scenario-independent cut information, i.e., cut gradients, independent cut intercepts, and so-called cumulative expected dual vectors (see [35]) obtained from the multi-stage L-shaped algorithm in Step 2. Then, in Step 4, for a given ξ^t, scenario-dependent cuts in (25) can be computed using the analytical formulae of [35, Theorem 3].

3.2 Problems with Interstage Dependence

The method of Section 3.1 handles stochastic linear programs with interstage independence, or a special type of dependence. In this section, we propose a different approach, which is computationally more demanding but allows for nonconvex problems with relatively complete recourse and general interstage dependency structures. In particular we consider the general T-stage stochastic program defined by (7)-(9) under assumptions (A1)-(A4) given in Section 2.1.

Our feasible policy construction for (7)-(9) works as follows: For a given ξ^t, we obtain $\hat{x}_t(\xi^t)$ by solving an approximating problem (from stage t to T) based on an independently-generated sample subtree, denoted $\Gamma_r(\xi^t)$ (the "r" subscript stands for "rolling"). Specifically, for a given ξ^t and x^{t-1}, $\Gamma_r(\xi^t)$ is constructed by the conditional sampling procedure described in Section 2.2 (either the common-samples or independent-samples method can be used). Then, $\hat{x}_t(\xi^t)$ is defined as an optimal solution of

$$\min_{x_t} \ \frac{1}{n(t)} \sum_{i \in D_t} \hat{f}_t(x^{t-1}, x_t, \Gamma_r(\tilde{\xi}^{t+1,i})), \tag{27}$$

where

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

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

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

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

Our policy, which computes $\hat{x}_t(\xi^t)$ by solving (27), is nonanticipative. None of the decisions made at descendant nodes in stages $t+1, \ldots, T$, are part of the policy. Decisions in these subsequent stages (e.g., $t+1$) are found by solving another approximating problem (e.g., from stage $t+1$ to T) with an independently-generated sample tree. Similarly, the decisions at previous stages needed to find x^{t-1} are also computed using independently-generated sample trees. Relatively complete recourse ensures that $\hat{x}^t(\xi^t)$ will lead to feasible solutions in stages $t+1, \ldots, T$. We denote this policy-generation procedure by $\mathcal{P}_2$ and summarize it in Figure 4. Although $\mathcal{P}_2$ is applicable to a more general class of stochastic programs than $\mathcal{P}_1$, we still need a viable solution procedure to solve (27). In a non-convex instance of (27), finding an optimal solution can be computationally difficult.

Given sample path ξ^T,
Do $t = 1$ to T
Independently construct a sample subtree $\Gamma_r(\xi^t)$.
Solve approximating problem (27) with x^{t-1} equal to $\hat{x}^{t-1}(\xi^{t-1})$, and denote its optimal solution $\hat{x}_t(\xi^t)$, where $\xi^t = (\xi^{t-1}, \xi_t)$.

Figure 4: Procedure $\mathcal{P}_2$ to generate a feasible policy for a T-stage stochastic program with relatively complete recourse.

4 Policy Cost Estimation

Under scenario $\tilde{\xi}^T$, the cost of using a given feasible policy, $\hat{x}^T(\tilde{\xi}^T)$, in (7)-(9) is $f_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T)$, and $Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T) \geq z^*$ because this is a feasible, but not necessarily optimal, policy. In general, it is impossible to compute this expectation exactly. In this section, we describe a scenario-based method and a tree-based method to estimate $Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T)$. These estimation procedures can be carried out for any feasible policy but, when appropriate, we discuss specific issues for policies $\mathcal{P}_1$ and $\mathcal{P}_2$.

4.1 Scenario-based Estimator

When employing a policy under scenario ξ^T, we obtain a sequence of feasible solutions, $\hat{x}_1(\xi^1)$, ..., $\hat{x}_T(\xi^T)$ (see Figures 3 and 4 for policies $\mathcal{P}_1$ and $\mathcal{P}_2$). The cost under scenario ξ^T is then given by $f_T(\hat{x}^T(\xi^T), \xi^T)$. In the case of a T-stage stochastic linear program, this cost is

$$f_T(\hat{x}^T(\xi^T), \xi^T) = \sum_{t=1}^{T} c_t(\xi^t)\hat{x}_t(\xi^t). \tag{28}$$

Again, we emphasize that with both $\mathcal{P}_1$ and $\mathcal{P}_2$, $\hat{x}^T(\xi^T)$ is nonanticipative because when we carry out the procedures of Figures 3 and 4 to find $\hat{x}_t(\xi^t)$ the subsequent realizations, $\xi_{t+1}, \ldots, \xi_T$, are not used (in fact, they need not even be generated yet).

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

$$\bar{U}_\nu = \frac{1}{\nu} \sum_{i=1}^{\nu} f_T(\hat{x}^T(\tilde{\xi}^{T,i}), \tilde{\xi}^{T,i}). \tag{29}$$

Let S_u^2 be the standard sample variance estimator of var $f_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T)$. Then,

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

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

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

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

4.2 Tree-based Estimator

The scenario-based estimation procedure of the previous section generates ν iid observations of $\tilde{\xi}^T$. The estimation procedure in this section is instead based on generating ν iid sample scenario trees. Later, in Section 5, we turn to estimating a lower bound on z^*. That lower bound is based on sample scenario trees and can be combined with either the scenario- or tree-based estimators to establish the quality of a solution policy. As will become apparent, the tree-based estimator in this section can be coupled with the lower-bound estimator in a manner not possible for the scenario-based estimator.

Let Γ be a sample scenario tree generated according to the conditional sampling framework of Section 2.2, and let n_T be the number of leaf nodes. Then, Γ may be viewed as a collection of scenarios, $\tilde{\xi}^{T,j}$, $j=1,\ldots,n_T$, which are identically distributed but are not independent. An unbiased point estimate of $Ef_T(\hat{x}^T(\tilde{\xi}^T),\tilde{\xi}^T)$ is given by

$$W = \frac{1}{n_T}\sum_{j=1}^{n_T} f_T(\hat{x}^T(\xi^{T,j}),\xi^{T,j}). \tag{30}$$

The numerical evaluation of $f_T(\hat{x}^T(\xi^{T,j}),\xi^{T,j})$, $j=1,\ldots,n_T$, under a specific policy occurs in the manner described in Section 4.1.

To quantify the error associated with the point estimate in (30), we generate ν iid sample trees, $\Gamma^i, i=1,\ldots,\nu$. Each of these trees is constructed according to the procedure described in Section 2.2 (again, under either the common-samples or independent-samples procedure). The number of scenarios in each Γ^i is again n_T, and the scenarios of Γ^i are $\xi^{T,ij}, j=1,\ldots,n_T$. The point estimate under Γ^i is

$$W^i = \frac{1}{n_T}\sum_{j=1}^{n_T} f_T(\hat{x}^T(\tilde{\xi}^{T,ij}),\tilde{\xi}^{T,ij}). \tag{31}$$

By construction, $W^i, i=1,\ldots,\nu$, are iid. So,

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

is the tree-based point estimate of $Ef_T(\hat{x}^T(\tilde{\xi}^T),\tilde{\xi}^T)$. Let S_w^2 be the standard sample variance estimator of $\operatorname{var} W$. Because $E\bar{W}_\nu = EW = Ef_T(\hat{x}^T(\tilde{\xi}^T),\tilde{\xi}^T)$, a confidence interval under the tree-based ap-

proach is constructed in a similar manner as in the scenario-based case, i.e., $\left(-\infty, \bar{W}_\nu + t_{\nu-1,\alpha} S_w/\sqrt{\nu}\right]$ is an approximate one-sided $100 \cdot (1-\alpha)\%$ confidence interval for $Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T)$.

5 Lower Bound Estimation

In this section, we develop a statistical lower bound for z^*, the optimal value of (7)-(9), and describe how to use this estimator to construct a one-sided confidence interval on z^*. Again, the motivation for forming such a confidence interval is to couple it with one of the confidence intervals from the previous section in order to establish the quality of a feasible policy, including those generated by $\mathcal{P}_1$ and $\mathcal{P}_2$. Here, quality is measured via the optimality gap of a policy defined as $Ef_T(\hat{x}^T(\tilde{\xi}^T), \tilde{\xi}^T) - z^*$.

Our lower-bound estimator requires little structure on the underlying problem, and we derive it using the notation of Section 2.1. First, we state the lower bound result for (7) when $T = 2$ in Lemma 1 (see also [42, 45]). In this case, (7) becomes a two-stage stochastic program with recourse, and the approximating problem, (17)-(19), reduces to

$$\hat{z}^* = \min_{x_1} \ \frac{1}{n_2} \sum_{i=1}^{n_2} f_1(x_1, \tilde{\xi}^{2,i}), \tag{32}$$

where

$$f_1(x_1, \tilde{\xi}^{2,i}) = \min_{x_2} \ f_2(x_1, x_2, \tilde{\xi}^{2,i}),$$

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

Lemma 1. *Assume $X_1(\xi^1) \neq \emptyset$ and is compact, $f_2(x_1, \cdot, \tilde{\xi}^2)$ is lower semi-continuous, wp1, for all $x_1 \in X_1(\xi^1)$, and $E\left|\inf_{x_2} f_2(x_1, x_2, \tilde{\xi}^2)\right| < \infty$ for all $x_1 \in X_1(\xi^1)$. Let z^* be defined as in program (7) with $T = 2$ and $\hat{z}^*$ be defined as in program (32). If $\tilde{\xi}^{2,1}, \ldots, \tilde{\xi}^{2,n_2}$ satisfy*

$$E\left[f_1(x_1, \tilde{\xi}^2)\right] = E\left[\frac{1}{n_2} \sum_{i=1}^{n_2} f_1(x_1, \tilde{\xi}^{2,i})\right],$$

i.e., condition (16) with $t = 1$, then

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

Proof. The lower semi-continuous and finite expectation assumptions on f_2 ensure that the objective functions of (7) and (32) are lower semi-continuous, and hence both have finite optimal solutions achieved on $X_1(\xi^1)$. The lower bound is then obtained by exchanging the order of expectation and

optimization:

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

□

Theorem 2. *Assume (A1)-(A4) and let z^* and $\hat{z}^*$ be defined as in (7) and (17), respectively. If the sample tree $\Gamma(\xi^1)$ is constructed so that the observations of each descendant satisfy the unbiasedness condition (16) for $t = 1, \ldots, T-1$, then*

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

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

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

$$f_{\tau-1}(x^{\tau-1}, \tilde{\xi}^\tau) \geq E\left[\min_{x_\tau} \frac{1}{n(\tau)} \sum_{i \in D_\tau} \hat{f}_\tau(x^{\tau-1}, x_\tau, \tilde{\xi}^\tau, \Gamma(\tilde{\xi}^{\tau+1,i}))\middle| \tilde{\xi}^\tau\right], \tag{33}$$

for $\tau = 1, \ldots, T-1$. Recursion (8) with $t = 1$ is $f_0(x^0, \tilde{\xi}^1) = z^*$; hence, (33) is equivalent to $z^* \geq E\hat{z}^*$ when $\tau = 1$. We proceed by induction, beginning with the base case, $\tau = T-1$. For a given $\tilde{\xi}^{T-1}$, $f_{T-2}(x^{T-2}, \tilde{\xi}^{T-1})$ is the optimal value of a two-stage stochastic program with recourse; therefore, by Lemma 1 and (19), the following relationship holds

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

where $\tilde{\xi}^{T,i} = (\tilde{\xi}^{T-1}, \tilde{\xi}_T^i)$. For the inductive part, we show that if (33) holds for $\tau = t$ then (33) holds for $\tau = t-1$. For $\tau = t-1$, we express the left-hand side of (33) by using (8) for a particular

descendant, say $\tilde{\xi}^{t-1,k} = (\tilde{\xi}^{t-2}, \tilde{\xi}^k_{t-1}), k \in D_{t-2}$, of node $\tilde{\xi}^{t-2}$ as

$$
\begin{aligned}
& f_{t-2}(x^{t-2}, \tilde{\xi}^{t-1,k}) \\
& \quad = \min_{x_{t-1}} \; E\left[f_{t-1}(x^{t-2}, x_{t-1}, \tilde{\xi}^t) \middle| \tilde{\xi}^{t-1,k} \right] \\
& \quad = \min_{x_{t-1}} \; E\left[\frac{1}{n(t-1,k)} \sum_{i \in D^k_{t-1}} f_{t-1}(x^{t-2}, x_{t-1}, \tilde{\xi}^{t,i}) \middle| \tilde{\xi}^{t-1,k} \right] \\
& \quad \geq \min_{x_{t-1}} E\left[\frac{1}{n(t-1,k)} \sum_{i \in D^k_{t-1}} E\left[\min_{x_t} \frac{1}{n(t,i)} \sum_{j \in D^i_t} \hat{f}_t(x^t, \tilde{\xi}^{t,i}, \Gamma(\tilde{\xi}^{t+1,j})) \middle| \tilde{\xi}^{t,i} \right] \middle| \tilde{\xi}^{t-1,k} \right]. \qquad (34)
\end{aligned}
$$

We use the unbiasedness condition (16) and the fact that $\tilde{\xi}^{t,i} = (\tilde{\xi}^{t-1,k}, \tilde{\xi}^i_t)$ to obtain the second equality, and the inductive hypothesis that (33) holds for $\tau = t$ to obtain the last inequality.

The outer conditional expectation in (34) is taken with respect to all immediate descendant nodes $\tilde{\xi}^{t,i}, i \in D^k_{t-1}$ of a given node $\tilde{\xi}^{t-1,k} = \xi^{t-1,k}$, while the inner expectation is with respect to all the subtrees $\Gamma(\tilde{\xi}^{t+1,j}), j \in D^i_t$, which are rooted at each of the descendants of a given node $\tilde{\xi}^{t,i} = \xi^{t,i}$. By combining these expectations and using recursion (18), we can write (34) as

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

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

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

holds for any node in stage $t-1$. □

In summary, $\hat{z}^*$ is the optimal value of the approximating problem (17)-(19), and z^* is the optimal value of the original problem (7)-(9). Theorem 2 states that if the sample scenario tree associated with (17)-(19) is constructed so that its observations satisfy the unbiasedness condition (16) then $\hat{z}^*$ is an estimator of z^* with negative bias, i.e., $E\hat{z}^* \leq z^*$. In Section 6, we show how to use this result in conjunction with a given feasible policy to construct a confidence interval on its optimality gap.

To assess the error associated with this estimator, we generate multiple replications of $\hat{z}^*$. In particular, we construct iid sample trees, $\Gamma^1, \ldots, \Gamma^\nu$, according to the procedure explained in Section 2.2, and then form the standard sample mean estimator as

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

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

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

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

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

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

6 Confidence Interval Construction

Confidence intervals on the optimality gap can be constructed in a number of ways depending on how the policy cost estimators developed in Sections 4.1 and 4.2 and the lower-bound estimator developed in Section 5 are combined. We explore two approaches: separate and gap estimators.

6.1 Separate Estimators

Our first approach to form a confidence interval for the optimality gap uses a policy cost estimator and a lower-bound estimator that are formed separately. In this setting, we can either combine the scenario-based estimator or the tree-based estimator with the lower-bound estimator. We begin with the tree-based case, and denote the sampling errors associated with the tree-based estimator and the lower-bound estimator by $\tilde{\epsilon}_w = t_{\nu-1,\alpha} S_w/\sqrt{\nu}$ and $\tilde{\epsilon}_l = t_{\nu-1,\alpha} S_l/\sqrt{\nu}$, respectively. From their confidence intervals, the probability of the events $\{\bar{L}_\nu - \tilde{\epsilon}_l \leq E\hat{z}^*\}$ and $\{Ef_T(\hat{x}^T, \tilde{\xi}^T) \leq \bar{W}_\nu + \tilde{\epsilon}_w\}$ are (individually) approximately $1-\alpha$.

So, if the two events are independent then

$$P\left\{ \bar{L}_\nu - \tilde{\epsilon}_l \leq E\hat{z}^*, Ef_T(\hat{x}^T, \tilde{\xi}^T) \leq \bar{W}_\nu + \tilde{\epsilon}_w \right\} \approx (1-\alpha)^2, \tag{35}$$

and if they are not independent then

$$P\left\{ \bar{L}_\nu - \tilde{\epsilon}_l \leq E\hat{z}^*, Ef_T(\hat{x}^T, \tilde{\xi}^T) \leq \bar{W}_\nu + \tilde{\epsilon}_w \right\} \approx (1-2\alpha). \tag{36}$$

We know that $E\hat{z}^* \leq z^* \leq Ef_T(\hat{x}^T, \tilde{\xi}^T)$, and so

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

where $(\,\cdot\,)^+ = \max\{\cdot\,, 0\}$. From this, we can infer that $[0, (\bar{W}_\nu - \bar{L}_\nu)^+ + \tilde{\epsilon}_l + \tilde{\epsilon}_w]$ is an approximate confidence interval at level $(1-\alpha)^2$ in the independent case and at level $(1-2\alpha)$ in the dependent case.

The scenario-based upper bound estimator can also be combined with the lower bound estimator to form a confidence interval on the optimality gap in a similar manner. To do so, we simply replace $\bar{W}_\nu$ and $\tilde{\epsilon}_w$ with $\bar{U}_\nu$ and $\tilde{\epsilon}_u = t_{\nu-1,\alpha} S_u/\sqrt{\nu}$, respectively, in the development above.

Dependency, and hence the need for the $(1-2\alpha)$-level confidence interval, can arise when the policy, $\hat{x}^T$, is constructed from the same instances of (17)-(19) that are used in forming the lower-bound estimator. For example, consider a T-stage stochastic linear program with interstage independence. In order to form $\bar{L}_\nu$ and $\tilde{\epsilon}_l$, we solve ν instances of (20)-(21), with respect to iid trees $\Gamma^i, i = 1, \ldots, \nu$. Assume that these trees are constructed using the common-samples approach described in Section 2.2. In solving an instance of (20)-(21) on sample tree Γ^i we can use the multi-stage L-shaped method, and cuts collected at each stage are valid for all nodes of that stage in Γ^i (because the sample tree exhibits interstage independence). So, after solving the approximating problems associated with each $\Gamma^i, i = 1, \ldots, \nu$, we have ν potential cut-based policies of the form $\mathcal{P}_1$. We choose one of these policies through some means, e.g., a separate screening procedure. (Other approaches are also possible.) Then, we follow the tree-based estimation procedure of Section 4.2 associated with event $\{Ef_T(\hat{x}^T, \tilde{\xi}^T) \leq \bar{W}_\nu + \tilde{\epsilon}_w\}$ and in solving the ν lower bounding problems we have the estimators associated with event $\{\bar{L}_\nu - \tilde{\epsilon}_l \leq E\hat{z}^*\}$. These events are not independent, in general, because the policy, $\hat{x}^T$, in the former event depends on samples used in forming $\bar{L}_\nu$ and $\tilde{\epsilon}_l$. We have discussed this issue in detail because these types of dependency issues can be easily overlooked. To avoid such dependency, we simply need to construct a policy that does not depend on samples used in forming the lower-bound estimator, e.g., using an independently generated tree, say, Γ^0. We pursue such a course in our computation and hence will use the $(1-\alpha)^2$ level when employing separate estimators.

6.2 Gap Estimator

Another way to form a confidence interval on the optimality gap of a given policy $\hat{x}^T$ is to combine the tree-based estimator and lower-bound estimator using the same set of sample trees. In simulation, this approach is referred to as using common random numbers and is often used to reduce variance (see, e.g., Law and Kelton [39]). We define a gap random variable for a given sample tree, Γ, by

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

Here, $\hat{z}^*$ is formed by solving (17)-(19) on sample tree Γ, and W is formed by the tree-based method of Section 4.2 on the *same* tree Γ. Note that this implies $G \geq 0$, wp1, and

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

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

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

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

7 Computational Results

We present computational results of our procedures on the three multi-stage stochastic linear programming test problems characterized in Table 2. STFOR is a stochastic forest harvest model from Gassmann [24]. DVA is a variant of the dynamic vehicle allocation model of Cheung and Powell [7], which is due to Donohue [16]. Donohue adapts the original model to allow backlogging and stochastic costs, and he provides a model-generation procedure that we used to generate our instance. WATSON is an asset-liability management model of Dempster [14]. Unlike STFOR and DVA, WATSON does not have interstage independence. A collection of WATSON test problems is available at `www-cfr.jims.cam.ac.uk/research/stprog.html` and our instance is labeled there as WATSON.10.512.I.

Table 2: Characteristics of three problems used in carrying our procedures for generating feasible policies and testing their quality.

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

Although the 512-scenario STFOR and WATSON problems can be solved to optimality, we still apply $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, to generate feasible policies for these problems, and we use our

quality-testing procedures to generate confidence intervals on the resulting optimality gap. As we will explain, the fact that these problems can be solved exactly allows us to examine the relative contributions of policy-suboptimality and bias to the confidence interval width. We use uniform sample trees, i.e., trees with the same number of descendants at every non-leaf node. Notation b^{T-1} denotes a T-stage sample scenario tree in which every node, up to stage $T-1$, has b descendants, and $b^{T-1}c$ or $b^{T-1}i$ indicates whether the common- or independent-samples method (Section 2.2) is used.

When solving instances of STFOR and WATSON on sampling-based scenario trees, we construct sample trees with the same 512 scenarios as those in the underlying scenario tree, but assign probability weights as the empirical weights obtained from sampling. In this way, as the sample size grows large the computational effort to solve the approximating problem does not grow beyond that of a 512-scenario tree. In the common-samples method, the (at most) 512-scenario tree resulting from assigning the empirical weights can be constructed without generating the actual sample tree with b^{T-1} scenarios because the future is identical at all nodes in the same stage. Hence, assigning the appropriate empirical weights in the tree with 512 scenarios yields a scenario tree equivalent to the one with b^{T-1} scenarios. This type of reduction is not possible when using the independent-samples method, and so we instead approximate the empirical weight of the descendants of each node in the 512-scenario tree by using an independent set of b observations.

All computational results are performed on a Dell Precision 530 (two 1.8 GHz processors) with 1GB of memory running the SuSE Linux 7.3 operating system. The multi-stage L-shaped method is implemented in C using the CPLEX 7.5 callable library and is compiled by GCC 2.95.3. Throughout our experiments, we only use one processor.

Table 3 reports computational results for STFOR under policy generation method $\mathcal{P}_1$, using separate estimators (Section 6.1) for the tree-based policy-cost estimator and the lower-bound estimator. Column (1) indicates the size of the scenario tree Γ_c, either 4^6c or 10^6c, used in forming the policy. The size and type of the sample trees $\Gamma^i, i = 1, \ldots, 30$, used in evaluating policy quality are given in column (2). The confidence intervals on the optimality gap are specified in column (6). Three sources of error contribute to confidence interval width: (i) the bias of the lower-bound estimator, (ii) the suboptimality of the policy, and (iii) the sampling error. Estimates of these contributions are given in columns (3), (4), and (5), respectively. We can compute estimates for (i) and (ii) because we know z^* for STFOR. Note that the estimates of (ii) in the first three rows of Table 3, when the size of Γ_c is 10^6c, are negative, i.e., $\bar{W}_{30} < z^*$. This is because the small-sized evaluation trees Γ^i ($2^6c, 2^6i$ and 4^6c) have large sampling errors relative to $E\bar{W}_{30} - z^*$, which is small because the large policy-generation tree (10^6c) yields a near-optimal policy. Column (7) gives the confidence interval width as a percentage of z^*. Column (8) gives the CPU time in seconds for the computation of each confidence interval, including the time required to generate the policy.

In Table 3, three independent random number streams are used to generate: (i) Γ_c, for forming the

cut-based policy, (ii) $\Gamma^i, i = 1, \ldots, 30$, for computing $\bar{W}_{30}$, and (iii) $\Gamma^i, i = 1, \ldots, 30$, for computing $\bar{L}_{30}$. The same random number stream is used to compute $\bar{L}_{30}$ when the size of Γ_c is 4^6c and 10^6c; therefore, the bias results in the top and bottom half of Table 3 are identical. Similarly, the same set of trees $\Gamma^i, i = 1, \ldots, 30$, are used to compute $\bar{W}_{30}$ when the size of Γ_c is 4^6c and 10^6c. So, the narrower confidence intervals in the bottom half of the table are primarily due to increased quality of the policy due to the larger policy-generation tree, Γ_c.

Table 4 lists similar numerical results to Table 3, but the confidence intervals are constructed via gap estimators (Section 6.2), i.e., the same set of trees, $\Gamma^i, i = 1, \ldots, 30$, are used for both $\bar{L}_{30}$ and $\bar{W}_{30}$. The random number streams to generate Γ_c and $\Gamma^i, i = 1, \ldots, 30$, for Table 4 are the same as the ones used for Γ_c and $\bar{W}_{30}$ in Table 3. The gap point estimate $\bar{G}_{30}$ may be obtained by adding columns (3) and (4). Column (9) in Table 4 gives the squared ratio of $\tilde{\epsilon}_w + \tilde{\epsilon}_l$ (from Table 3) and $\tilde{\epsilon}_g$ (from Table 4). These ratios indicate the approximate factor by which ν must increase in Table 3 in order for the sampling errors in Tables 3 and 4 to be similar. The CPU times in Table 4 are consistently slightly smaller than those in Table 3 because (i) only one set of ν sample trees is generated, and (ii) evaluating the upper and lower bounds on the same sample tree benefits from an advanced initial solution.

From Tables 3 and 4, we observe that the quality of feasible policies generated by $\mathcal{P}_1$ improves as the sample size of Γ_c increases. The bias in column (3) decreases as the size of Γ^i used to compute $\bar{L}_{30}$ increases. The independent-samples method usually gives lower sampling error than the common-samples method (the only exceptions are rows $10^6c/2^6$ and $10^6c/50^6$ in Table 4). The sampling error is more dramatically reduced by using the gap estimator due to the variance reduction effect of using common random numbers.

In further experiments, the scenario-based estimator of expected policy cost replaced the tree-based estimator reported in Table 3. We do not detail these results, but only indicate that with the same value of ν, the sampling error is slightly smaller than that of the tree-based estimator. However, the associated CPU time is substantially larger than that of the tree-based estimator. This is due to the increased number of subproblems that must be solved to form the scenario-based policy-cost estimator.

Based on the STFOR results, we only use the tree-based gap estimator for DVA and WATSON, and $\Gamma^i, i = 1, \ldots, \nu$, are constructed by the independent-samples method. Table 5 contains results for DVA. Policy generation is again via $\mathcal{P}_1$ with the size of Γ_c in column (1) and that of $\Gamma^i, i = 1, \ldots, 30$, in column (2). Columns (3)-(5) contain the gap point estimate, sampling error, and resulting 95% confidence interval. The optimal value, z^*, for DVA is not known and so we cannot separate contributions to the gap estimate due to bias and policy suboptimality. Confidence interval width as a percentage of the policy cost estimator is given in column (6). CPU time in minutes is shown in column (7), and again, includes the time required to generate the policy. For DVA the sampling error

is small relative to policy cost and the gap estimate. This is, in part, due to the variance reduction from using common scenario trees. Overall, the computational results for DVA are qualitatively similar to those of STFOR.

The WATSON test problem does not have interstage independence, and hence we use $\mathcal{P}_2$ to form a policy. Table 6 shows the associated computational results. An issue in forming the policy is the size of the subtree $\Gamma_r(\xi^t)$ constructed at each node in each stage when generating $\hat{x}_t(\xi^t)$ (see Section 3.2). We construct $\Gamma_r(\xi^t)$ so that it has the same number of leaf nodes for all $\xi^t, t = 1, \ldots, T-1$. Two different sizes of $\Gamma_r(\xi^t)$ are used for doing so, and these are denoted "Method (A)" and "Method (B)" in Table 6. The details of the structure of subtree $\Gamma_r(\xi^t)$ are given in Table 7, where subtree $\Gamma_r(\xi^8)$, for example, is rooted at a node in stage 8 and has a constant number of descendants of 4^5 for the stage 8 node and 4^4 for each stage 9 node under Method (A). The bias, sampling error, and quality of the policy associated with the computational results for WATSON in Table 6 are qualitatively similar to those of STFOR and DVA.

8 Conclusions

In this paper, we have developed two Monte Carlo-based methods to generate feasible policies for multi-stage stochastic programs. The first method can be applied to multi-stage stochastic linear programs that exhibit interstage independence, or a special type of interstage dependence. To form a policy, this approach uses cuts obtained from solving an approximating problem via the multi-stage L-shaped method. The second method requires neither convexity of the problem nor interstage independence of the stochastic process. Although it can be applied to a larger class of stochastic programs, it is more computationally intensive.

Given a feasible policy, one should estimate its expected cost and its quality. We described two policy-cost estimators—a scenario-based estimator and a tree-based estimator. To assess the quality of a feasible policy, we developed a statistical lower bound, and showed how to construct a confidence interval on the policy's optimality gap (or, more precisely, a confidence interval on a bound of the optimality gap) by combining a policy-cost estimator with a lower-bound estimator. Several combinations are possible, but our computational results suggest that the confidence interval formed by using the gap estimator, constructed by combining the tree-based policy-cost estimator and the lower-bound estimator both of which are computed from the same set of sample trees, may be effective.

One valuable enhancement of our methodology would be to couple it with one of the family of sampling-based decomposition algorithms from [6, 16, 31, 46]. Specifically, consider a T-stage stochastic linear program with continuous random parameters having interstage independence (or the type of dependency described in [35]). We can build a sample scenario tree with b^{T-1} leaf nodes

using the common-samples approach (Section 2.2). This problem, under the sampled scenario tree, can be solved via one of these methods provided b is of manageable size, even when b^{T-1} is very large, and these approaches can be used for both policy generation and for lower-bound estimation.

In our policy generation methods, we use naive Monte Carlo sample trees in generating a feasible solution at each node. Another enhancement would be to construct a tree designed to produce a higher-quality policy with comparable computational effort. Scenario tree approximation methods reviewed in Section 1 may be of use in this regard. Other important improvements include developing techniques that reduce the variance and bias of the gap estimator so that the width of the confidence interval on the optimality gap is as small as possible for a fixed computational resource. This will enable us to distinguish a high quality policy in a more effective manner.

9 Acknowledgement

This research was supported by the National Science Foundation under Grants DMI-9702217 and DMI-0217927 and the State of Texas Advanced Research Program through grant #003658-0405.

References

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

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

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

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

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

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

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

[8] Anukal Chiralaksanakul. *Monte Carlo Methods for Multi-stage Stochastic Programs*. PhD thesis, The University of Texas at Austin, 2003.

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

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

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

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

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

[14] M.A.H. Dempster and G. Consigli. The CALM stochastic programming model for dynamic asset-liability management. In J.M. Mulvey and W.T. Ziemba, editors, *World Wide Asset and Liability Modelling*, pages 464–500. Cambridge University Press, 1998.

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

[16] C. Donohue. *Stochastic Network Programming and the Dynamic Vehicle Allocation Problem*. PhD thesis, The University of Michigan, Ann Arbor, 1996.

[17] J. Dupačová. Postoptimality for multistage stochastic linear programs. *Annals of Operations Research*, 56:65–78, 1995.

[18] J. Dupačová, N. Gröwe-Kuska, and W. Römisch. Scenario reduction in stochastic programming: An approach using probability metrics. *Mathematical Programming*, 95:493–511, 2003.

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

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

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

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

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

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

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

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

[27] J.L. Higle, B. Rayco, and S. Sen. Stochastic scenario decomposition for multi-stage stochastic programs. Technical report, Systems and Industrial Engineering Department, University of Arizona, 2001.

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

[29] J.L. Higle and S. Sen. Duality and statistical tests of optimality for two stage stochastic programs. *Mathematical Programming*, 75:257–275, 1996.

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

[31] M. Hindsberger and A.B. Philpott. ReSa: A method for solving multistage stochastic linear programs. Technical report, University of Auckland, New Zealand, 2003.

[32] K. Høyland, M. Kaut, and S. Wallace. A heuristic for moment-matching scenario generation. *Computational Optimization and Applications*, 24:169–185, 2003.

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

[34] G. Infanger. *Planning Under Uncertainty: Solving Large-Scale Stochastic Linear Programs*. The Scientific Press Series, Boyd & Fraser, Danvers, Mass., 1994.

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

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

[37] V. Kaňková. A remark on empirical estimates in multistage stochastic programming. *Bulletin of the Czech Econometric Society*, 9:31-50, 2002.

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

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

[40] J. Linderoth, A. Shapiro, and S. Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*. To appear.

[41] K. Linowsky and A.B. Philpott. On the convergence of sampling based decomposition algorithms for multistage stochastic programs. Technical report, University of Auckland, New Zealand, 2003.

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

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

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

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

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

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

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

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

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

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

[52] A. Shapiro. Inference of statistical bounds for multistage stochastic programming problems. *Mathematical Methods of Operations Research*, 58:57–68, 2003.

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

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

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

[56] S.W. Wallace and W.T. Ziemba, editors. *Applications of Stochastic Programming*. MPS-SIAM Series in Optimization, 2003. To appear.

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

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

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

Table 3: Computational results for STFOR: 95% confidence intervals on the optimality gap of feasible policies constructed by using cut-based procedure $\mathcal{P}_1$ with scenario trees of size given in column (1). Each confidence interval is formed by separate estimators of the policy cost and the lower bound with $\nu = 30$. The confidence interval, and its width as a percentage of z^* are given in columns (6) and (7). Contributions to the confidence interval due to bias, suboptimality and sampling error are estimated in columns (3)-(5). CPU time is shown in column (8) in seconds.

Γ_c	Γ^i	$z^* - \bar{L}_{30}$	$\bar{W}_{30} - z^*$	$\tilde{\epsilon}_w + \tilde{\epsilon}_l$	95% CI	% z^*	sec.
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
	2^6c	781.42	650.92	2107.08	[0, 3539.42]	8.07	543
	2^6i	672.53	417.63	1114.77	[0, 2204.93]	5.03	541
	4^6c	719.96	262.67	985.38	[0, 1968.01]	4.49	589
4^6c	4^6i	384.01	784.77	580.22	[0, 1749.00]	3.99	612
	10^6c	143.90	684.68	714.64	[0, 1543.22]	3.52	658
	10^6i	233.42	496.40	260.52	[0, 990.34]	2.26	665
	50^6c	116.53	564.09	266.40	[0, 947.02]	2.16	719
	50^6i	41.22	540.17	155.08	[0, 736.47]	1.68	768
	2^6c	781.42	-91.41	797.03	[0, 1487.04]	3.60	806
	2^6i	672.53	-41.72	681.84	[0, 1249.65]	2.94	821
	4^6c	719.96	-22.02	602.93	[0, 1300.87]	3.02	843
10^6c	4^6i	384.01	109.84	327.75	[0, 821.60]	1.87	890
	10^6c	143.90	128.52	424.16	[0, 696.58]	1.59	913
	10^6i	233.42	50.23	161.95	[0, 445.60]	1.02	944
	50^6c	116.53	80.45	164.56	[0, 361.54]	0.82	977
	50^6i	41.22	74.27	82.60	[0,198.09]	0.45	1046

Table 4: Computational results for STFOR. 95% confidence intervals on the optimality gap of feasible policies constructed by using cut-based procedure $\mathcal{P}_1$. Each confidence interval is formed by the gap estimator with $\nu = 30$, in which $\bar{L}_{30}$ and $\bar{W}_{30}$ are based on the same 30 sampled scenario trees. The table has the same format as Table 3 except that column (9) estimates variance reduction from using common scenario trees.

Γ_c	Γ^i	$z^* - \bar{L}_{30}$	$\bar{W}_{30} - z^*$	$\tilde{\epsilon}_g$	95% CI	% z^*	sec.	$(\frac{\tilde{\epsilon}_w + \tilde{\epsilon}_l}{\tilde{\epsilon}_g})^2$
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
	2^6c	923.87	650.92	1430.17	[0, 3004.96]	6.85	478	2.17
	2^6i	900.99	417.63	621.07	[0, 1939.69]	4.42	471	3.22
	4^6c	640.90	262.67	406.36	[0, 1309.93]	2.99	548	5.88
4^6c	4^6i	308.05	784.77	287.56	[0, 1380.38]	3.15	552	4.07
	10^6c	193.79	684.68	299.41	[0, 1177.88]	2.68	591	5.70
	10^6i	136.83	496.40	90.94	[0, 724.17]	1.65	607	8.21
	50^6c	6.17	564.09	100.08	[0, 670.34]	1.53	670	7.09
	50^6i	12.81	540.17	60.23	[0, 613.21]	1.40	709	6.63
	2^6c	923.87	-91.41	188.92	[0, 1021.38]	2.33	763	17.80
	2^6i	900.99	-41.72	205.63	[0, 1064.90]	2.43	779	9.06
	4^6c	640.90	-22.02	123.65	[0, 742.53]	1.69	829	23.78
10^6c	4^6i	308.05	109.84	60.73	[0, 478.62]	1.09	856	29.13
	10^6c	193.79	128.52	82.48	[0, 404.79]	0.92	868	26.45
	10^6i	136.83	50.23	26.87	[0, 213.93]	0.49	895	36.33
	50^6c	6.17	80.45	17.66	[0, 104.28]	0.24	943	86.83
	50^6i	12.81	74.27	19.36	[0, 106.44]	0.24	967	18.20

Table 5: Computational results for DVA: 95% confidence intervals on the optimality gap of feasible policies constructed by using procedure $\mathcal{P}_1$. Each confidence interval is formed by the tree-based gap estimator with $\nu = 30$.

Γ_c	Γ^i	$\bar{G}_{30}$	$\tilde{\epsilon}_g$	95% CI	% $\lvert\bar{W}_{30}\rvert$	min.
(1)	(2)	(3)	(4)	(5)	(6)	(7)
	2^3i	8183.86	428.09	[0, 8611.95]	3.74	11.02
7^3c	4^3i	5821.40	270.06	[0, 6091.46]	2.66	38.75
	7^3i	4668.25	121.69	[0, 4790.14]	2.08	178.23
	10^3i	4202.30	107.88	[0, 4310.18]	1.88	542.07
	2^3i	6826.28	545.25	[0, 7371.53]	3.18	30.45
15^3c	4^3i	4335.62	217.13	[0, 4552.75]	1.98	60.87
	7^3i	3147.82	106.08	[0, 3253.90]	1.41	204.67
	10^3i	2664.00	103.50	[0, 2767.50]	1.20	540.67

Table 6: Computational results for WATSON: 95% confidence intervals on the optimality gap of feasible policies constructed by using procedure $\mathcal{P}_2$. Each confidence interval is formed by the gap estimator with $\nu = 30$ ($\bar{L}_{30}$ and $\bar{W}_{30}$ are based on the same 30 sample trees). The sample size for subtrees in $\mathcal{P}_2$ is determined by Methods (A) and (B), whose details are given in Table 7.

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

Table 7: Sample size for generating subtrees in procedure $\mathcal{P}_2$ for WATSON test problem.

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