Confidence Interval Software for Multi-stage Stochastic Programs

Xiaotie Chen
Applied Mathematics, UC Davis

Sylvain Cazaux
École Polytechnique

Brian C. Knight
Applied Mathematics, UC Davis

David L. Woodruff
Graduate School of Management, UC Davis
DLWoodruff@UCDavis.edu

September 28, 2021

Abstract

When the uncertainty is explicitly modeled in an optimization problem, it is often necessary to use samples to compute a solution, which gives rise to a need to compute confidence intervals around the objective function value that is obtained. In this paper we describe software that implements well-known methods for two stage problems and we provide extensions to multiple stages. The software is available as part of the mpi-sppy package Knueven et al. (2021), which is available for download from `https://github.com/Pyomo/mpi-sppy`.

1 Introduction

Optimization technology provides an opportunity to enhance decision making based on mathematical models; however, the input data are almost always uncertain. When the uncertainty is explicitly modeled it is often necessary to use samples to compute a solution, which gives rise to a need to compute confidence intervals around the objective function value that is obtained. The intervals are useful for subsequent analysis as well as for making decisions about how many samples to draw, which implies allocation of computing resources.

Our interest is in situations where decisions are made in stages as uncertainty is resolved. The solution methods we support are intended for use when the decisions made here-and-now are the important decisions and policies concerning future decisions are included in the model only to enhance the quality of immediate decisions. Hence, much of the research to date has focused on two stages. The methods are appropriate for large numbers of decision variables (thousand or millions) but at most a handful of stages in the model.

The notation we use follows more-or-less from (Knueven et al. 2021, Chiralaksanakul and Morton 2004). We are considering a T-stage stochastic program. We use $t \in \{1, \ldots, T\}$ to index stages, and use $\{\boldsymbol{\xi}_t\}_{t=2}^{T}$ to denote the associated stochastic process. The bold letter $\boldsymbol{\xi}_t$ is used to denote random variables, which can be vector valued, and ξ_t is used to denote their realizations. The values of ξ_t for $t = 2, \ldots, T$ are revealed at the end of stage $t-1$, therefore we refer to $\boldsymbol{\xi}_t$ and ξ_t only for $t = 2, \ldots, T$. We use the symbol $\vec{\xi}^t$ to refer to the realized values of all random variables up to and including stage t, i.e.,

$$\vec{\xi}^t = (\xi_2, \xi_3, \ldots, \xi_t).$$

When $t = T$, $\vec{\xi}^T$ is known as a *scenario.*

We use x_t to represent the decision that is made at stage t, and use the notation $\vec{x}^t$ for $1 \leq t \leq T$ to represent the decisions for all stages up to, and including, stage t, i.e.

$$\vec{x}^t = (x_1, x_2, \ldots, x_t)$$

For a T-stage stochastic program, the decision x_t for $t = 2, \ldots, T$ is made such that given the realized $\vec{\xi}^t$, a problem-specific objection function is minimized. To form a recursive expression, we rely on a functional ϕ to capture the objectives for the problem at hand given scenario data. Note that at each stage t, the decision x_t is made with only the knowledge of $\vec{x}^{t-1}$ and $\vec{\xi}^t$ and is independent of the future decisions and realizations of $\boldsymbol{\xi}_{t+1}, \ldots, \boldsymbol{\xi}_T$. Such independence is known as non-anticipativity. The first stage solution x_1 is made so that it minimizes the expected value of $\phi_2(x_1; \boldsymbol{\xi}_2)$.

The above notation allows us to write the multi-stage stochastic problem (MSSP) recursively as

$$\begin{aligned} z^* = \min_{x_1} \quad & \mathbb{E}_{\boldsymbol{\xi}_2} \phi_2 (x_1; \boldsymbol{\xi}_2), \\ s.t. \quad & x_1 \in X_1 \end{aligned} \tag{1}$$

where for each $t = 2, \ldots, T-1$, we have

$$\begin{aligned} \phi_t(\vec{x}^{t-1}; \vec{\xi}^t) \quad &= \quad \min_{x_t} \mathbb{E}_{\vec{\boldsymbol{\xi}}^{t+1}} [\phi_{t+1}((\vec{x}^{t-1}, x_t); \vec{\boldsymbol{\xi}}^{t+1}) | \vec{\boldsymbol{\xi}}^t = \vec{\xi}^t] \\ s.t. \quad & \quad x_t \in X_t \left(\vec{x}^{t-1}, \vec{\xi}^t\right), \end{aligned} \tag{2}$$

and

$$\begin{array}{rcl} \phi_T(\vec{x}^{T-1};\vec{\xi}^T) & = & \min_{x_T} \phi'_T(\vec{x}^T;\vec{\xi}^T) \\ s.t. & & x_T \in X_T\left(\vec{x}^{T-1},\vec{\xi}^T\right), \end{array}$$

where $\phi'_T(\vec{x}^T;\vec{\xi}^T)$ is the problem-specific objective function for the final stage given the full scenario data and solutions to the previous stage(s). At each stage we impose a constraint for the feasibility of x_t, such that x_t belongs to a feasible region $x_t \in X_t\left(\vec{x}^{t-1},\vec{\xi}^t\right)$. We incorporate feasible region constraints into the objective function by introducing the extended representation

$$f_{t+1}(\vec{x}^t;\vec{\xi}^{t+1}) = \begin{cases} \phi_{t+1}(\vec{x}^t;\vec{\xi}^{t+1}), & x_t \in X_t(\vec{x}^{t-1},\vec{\xi}^t) \\ \infty, & \text{otherwise.} \end{cases}$$

Then, the MSSP problem (1) can be simplified to

$$z^* = \min_{x_1} \mathbb{E}_{\boldsymbol{\xi}_2} f_2\left(x_1;\boldsymbol{\xi}_2\right), \tag{3}$$

where for each $t = 2,\ldots,T-1$, we have

$$f_t(\vec{x}^{t-1};\vec{\xi}^t) = \min_{x_t} \mathbb{E}_{\vec{\boldsymbol{\xi}}^{t+1}}[f_{t+1}((\vec{x}^{t-1},x_t);\vec{\boldsymbol{\xi}}^{t+1})|\vec{\boldsymbol{\xi}}^t = \vec{\xi}^t], \tag{4}$$

and

$$f_T(\vec{x}^{T-1};\vec{\xi}^T) = \min_{x_T} f'_T(\vec{x}^T;\vec{\xi}^T)$$

where

$$f'_T(\vec{x}^T;\vec{\xi}^T) = \begin{cases} \phi'_T(\vec{x}^T;\vec{\xi}^T), & x_T \in X_T(\vec{x}^{T-1},\vec{\xi}^T) \\ \infty, & \text{otherwise.} \end{cases}$$

For theoretical purposes, we will assume that we are dealing with stochastic programs with relatively complete recourse, which means that for solution values that are feasible in earlier stages there exist feasible solution values in subsequent stages.

A special case is the T-stage stochastic linear program, which is defined as

$$\min_{x_1\in X_1} c_1x_1 + \mathbb{E}\left[\min_{x_2\in X_2(x_1,\xi_2)} c_2x_2 + \mathbb{E}\left[\cdots + \mathbb{E}[\min_{x_T\in X_T(x_{T-1},\xi_T)} c_Tx_T]\right]\right] \tag{5}$$

where ξ_t represents the random data at stage t, $\xi_t = \{c_t, A_t, B_t, b_t\}$, and feasible region $X_1 = \{x_1 : A_1x_1 = b_1, x_1 \geq 0\}$, and for $t = 2,\ldots,T$, we have

$$X_t(x_{t-1};\xi_t) = \{x_t : B_tx_{t-1} + A_tx_t = b_t, x_t \geq 0\}$$

To draw a connection between the multi-stage linear program and the general MSSP problem defined above, notice that by taking

$$\phi'_T(\vec{x}^T;\vec{\xi}^T) = c_1x_1 + \ldots + c_Tx_T,$$

then by recursive form (2), we have

$$\phi_T(\vec{x}^{T-1};\vec{\xi}^T) = c_1x_1 + \ldots + c_{T-1}x_{T-1} + \min_{x_T} c_T x_T,$$

$$\begin{aligned}\phi_{T-1}(\vec{x}^{T-2};\vec{\xi}^{T-1}) &= \min_{x_{T-1}} \mathbb{E}_{\vec{\boldsymbol{\xi}}^T}[\phi_T(\vec{x}^{T-1};\vec{\boldsymbol{\xi}}^T)|\vec{\boldsymbol{\xi}}^{T-1} = \vec{\xi}^{T-1}] \\ &= \min_{x_{T-1}} \mathbb{E}_{\vec{\boldsymbol{\xi}}^T}[c_1x_1 + \ldots + \min_{x_T} c_T x_T|\vec{\boldsymbol{\xi}}^{T-1} = \vec{\xi}^{T-1}] \\ &= c_1x_1 + \ldots + \min_{x_{T-1}}\left[c_{T-1}x_{T-1} + \mathbb{E}_{\boldsymbol{\xi}_T} \min_{x_T} c_T x_T\right]\end{aligned}$$

and similarly we can recursively define the object function for $t = T - 2, \ldots, 2$, and in the end we would have

$$\begin{aligned}\phi_2(x_1;\xi_2) &= \min_{x_2} \mathbb{E}_{\vec{\boldsymbol{\xi}}^3}[\phi_3(\vec{x}^2;\vec{\boldsymbol{\xi}}^3)|\boldsymbol{\xi}_2 = \xi_2] \\ &= \min_{x_2} \mathbb{E}_{\vec{\boldsymbol{\xi}}^3}[c_1x_1 + c_2x_2 + \min_{x_3}[c_3x_3 + \mathbb{E}[\ldots + \mathbb{E}\min_{x_T} c_T x_T]]|\boldsymbol{\xi}_2 = \xi_2] \\ &= c_1x_1 + \min_{x_2}[c_2x_2 + \mathbb{E}[\min_{x_3} c_3x_3 + \mathbb{E}[\ldots + \mathbb{E}\min_{x_T} c_T x_T]]],\end{aligned}$$

and finally $\min_{x_1} \mathbb{E}_{\boldsymbol{\xi}_2} \phi_2(x_1, \boldsymbol{\xi}_2)$ would be in the form of (5).

For multi-stage problems it is useful to consider a scenario tree with the property that scenarios with the same realization up to stage t share a node at that stage. Consequently, $\vec{\xi}^t$ can be regarded as a node in the scenario tree at stage t. We use $\vec{\xi}^1$ to represent the dummy root node of the scenario tree, whose descendant nodes are realizations of $\boldsymbol{\xi}_2$. We emphasize here that $\vec{\xi}^1$ is not a random variable. We use $\Gamma(\vec{\xi}^t)$ to denote a general subtree with root $\vec{\xi}^t$. Depends on the context, $\Gamma(\vec{\xi}^t)$ can be generated by either taking the entire subtree in from the original scenario tree, or by sampling from the original tree. We will talk about generating $\Gamma(\vec{\xi}^t)$ by sampling in later sections. For a specific node $\vec{\xi}^t$, we use $D(\vec{\xi}^t)$ to represent the set of its descendant nodes. The cardinality of set $D(\vec{\xi}^t)$ is denoted as $|D(\vec{\xi}^t)|$. We use $F(\xi_t|\vec{\xi}^{t-1})$ to denote the conditional distribution of $\boldsymbol{\xi}_t$ given the history up to stage $t-1$.

In this paper we describe software that implements well-known methods for two stage problems and we provide extensions to multiple stages along with supporting theory and software. The software is available as part of the mpi-sppy package Knueven et al. (2021), which is available for download from `https://github.com/Pyomo/mpi-sppy`. The source code is in the `confidence_intervals` directory. The rest of the paper proceeds as follows. Literature is reviewed in Section 2 with particular attention to the two-stage methods that we implemented. These methods are extended to more than two stages in Section 3. Computational experiments are described in Section 4 and the paper closes with conclusions and directions for further research in Section 5.

2 Literature

Multistage stochastic optimization problems provide a framework for optimized decision-making under uncertainty. Originally proposed by Dantzig

(1955), it has been applied to a broad spectrum of areas, including energy planning Pereira and Pinto (1991), hydrothermal scheduling De Matos et al. (2017), vehicle routing Kenyon and Morton (2003), financial modeling Ziemba and Vickson (2014), supply chain management Keyvanshokooh et al. (2016), and many others.

In practice, it is often necessary to approximate a multistage stochastic optimization problem, because the problem parameters may be a random variables with continuous distribution for which it is impossible to evaluate the expected cost. Even when the distribution of the parameter has a finite support, one may still have computational difficulties due to the large size of the problem.

Therefore, there has been a growing literature in studying Monte Carlo sampling methods for approximating stochastic optimization problems. For example, Homem-de Mello and Bayraksan (2014) and Norkin et al. (1998) samples within the branch-and-bound algorithm and uses the stochastic upper and lower bound estimators to prune the search tree. A stochastic cutting plane method in Higle and Sen (2013) uses sampling-based cutting planes within the L-shape method for stochastic programming. Higle and Sen (1999) discuss methods for assessing solution quality and develop stopping rules in the stochastic cutting plane context. Similar ideas were later used to develop stopping rules for other algorithms as discussed below.

Other works uses Monte Carlo methods for external sampling, where the sampling procedure is done before the solution is being calculated. For two-stage stochastic programming, this coincides with the well-studied sample-and-average method and there has been a rich literature on the asymptotic convergence results (Dupacová and Wets 1988, King and Rockafellar 1993).

Mak et al. (1999) developed sampling-based upper and lower bound estimators to construct a confidence interval on the optimality gap for two-stage stochastic programming based on the asymptotic normality. Additional computational results can be found in, for example, Linderoth et al. (2006).

Shapiro (2003) discussed the difficulty in extending such sampling methodology to multistage linear programming problems and proposed a conditional sampling procedure for constructing consistent lower bound. Chiralaksanakul and Morton (2004) further proposed two Monte Carlo sampling method to construct upper bounds for the optimality gap in the case of stagewise independence and stagewise dependence. Together with the lower bound estimator based on conditional sampling, they are able to provide a confidence interval for the optimality gap in multistage stochastic programming.

We developed software referred to as *MMW* based on Mak et al. (1999), Shapiro (2003), Chiralaksanakul and Morton (2004) that assess the quality of solutions for multistage programmings. We also developed software we refer to as *sequential sampling* based on Bayraksan and Morton (2011, 2009), Bayraksan and Pierre-Louis (2012), which develop stopping rules for sequential sampling procedure that estimate the optimality gap of a sequence of candidates solutions under an increased sample size.

We note here that a few other papers discussed how to assess the solution quality for multi-stage programmings with inter-stage independence. These algorithms are mostly adapted from the SDDP algorithms Pereira and Pinto (1991). Homem-de Mello et al. (2011) presented an alternative stopping criteria for SDDP that is more statistically robust. Kozmík and Morton (2015) shows that importance sampling can be used to improve the upper bound estimators in the risk-averse setting. Their work improves the results of Shapiro (2011). De Matos et al. (2017) proposed an algorithm that can construct a confidence interval of the optimality gap using a single scenario tree.

3 Extension to Multi-stage

3.1 Prerequisites before estimating confidence interval

Equation (1) suggests that if we can compute the function value of $\phi_2(x_1; \boldsymbol{\xi}_2)$ exactly, we can then regard a multi-stage problem as a two-stage programming problem, and apply MMW directly to find a CI estimation of the optimality gap. However, in practice, one may not be able to evaluate $\phi_2(x_1; \boldsymbol{\xi}_2)$ as it is a stochastic programming problem in the multi-stage case. Below we follow the work of (Shapiro 2003, Chiralaksanakul and Morton 2004) to describe a conditional sampling procedure that allows us to construct a confidence interval later on.

3.1.1 Conditional Sampling

Multi-stage confidence interval estimation for problems based on scenario trees requires construction of sample scenario trees. In this section we describe the use of conditional sampling to construct a subtree $\Gamma(\vec{\xi}^t)$ given any node $\vec{\xi}^t$ in the original scenario tree.

We begin by sampling n_{t+1} observations $(\xi_{t+1,1}, \ldots, \xi_{t+1,n_{t+1}})$ from the conditional distribution $F(\xi_{t+1}|\vec{\xi}^t)$ to form n_{t+1} descendant nodes $(\vec{\xi}^{t+1,1}, \ldots, \vec{\xi}^{t+1,n_{t+1}})$ for $\vec{\xi}^t$, where $\vec{\xi}^{t+1,i} = (\vec{\xi}^t, \xi_{t+1,i})$ for $t \geq 2$. Here n_{t+1} is the sample size. In general the sample size associated with each node doesn't need to be the same, but for simplicity, here we use n_t to denote the uniform sample size for stage t. Then for each node $\vec{\xi}^{t+1,i}$, we again sample from the conditional distribution $F(\xi_{t+2}|\vec{\xi}^{t+1,i})$ to form n_{t+2} descendants. We repeat the sampling process until we reach the leaf node, i.e. $t = T$. The total number nodes in stage τ is $\prod_{j=t+1}^{\tau} n_j$.

Note that here we do not specify how the conditional sampling at each stage is conducted. For example, conditioned on $(\vec{\xi}^{t+1,1}, \ldots, \vec{\xi}^{t+1,n_{t+1}})$, the nodes at stage $t+2$ may not be independent. However, we assume that the samples at each stage form an unbiased estimation. In other words, we require that for each stage t in our generated subtree,

$$\mathbb{E}_{\vec{\boldsymbol{\xi}}^t}[f_t(\vec{x}^{t-1}; \vec{\boldsymbol{\xi}}^t)|\vec{\boldsymbol{\xi}}^{t-1}] = \mathbb{E}\left[\frac{1}{n_t}\sum_{i=1}^{n_t} f_t(\vec{x}^{t-1}; \vec{\xi}^{t,i})|\vec{\boldsymbol{\xi}}^{t-1}\right].$$

The above condition holds when the sampling is done in an i.i.d. manner, but other methods, for example importance sampling, can achieve the same results.

3.1.2 Optimization given scenario tree

To simplify notation for now, we take a given finite scenario tree $\Gamma(\vec{\xi}^1)$ and assume we can find an optimal solution for the following optimization problem:

$$\widehat{z}^* = \min_{x_1} \frac{1}{n_2} \sum_{\xi_{2,i} \in D(\vec{\xi}^1)} \widehat{f}_2(x_1; \xi_{2,i}) \tag{6}$$

where for $t = 2, \ldots, T-1$, $\widehat{f}_t$ is recursively defined as

$$\widehat{f}_t(\vec{x}^{t-1}; \vec{\xi}^{t,i}) = \min_{x_t} \frac{1}{n_{t+1}} \sum_{\xi_{t+1,j} \in D(\vec{\xi}^{t,i})} \widehat{f}_{t+1}(\vec{x}^t; (\vec{\xi}^{t,i}, \xi_{t+1,j})), \tag{7}$$

and

$$\widehat{f}_T(\vec{x}^{T-1}, \vec{\xi}^{T,i}) = \min_{x_T} f'_T(\vec{x}^T; \vec{\xi}^{T,i}),$$

which is essentially an approximation of the MSSP problem defined in (1). When the original problem has a finite scenario tree $\Gamma(\vec{\xi}^1)$, the optimal value in (6) is the same as in (3). In the sequel we will add notation to acknowledge that in general we may be able to compute only a lower bound on the objective function.

3.1.3 Feasible Solution

Suppose now we have a decision $\widehat{x}_1$ for stage one, which may be obtained by solving an approximation problem for MSSP (1). Given a scenario $\vec{\xi}^T$, we are interested in finding a feasible solution, which is specified by the decision at different stages $\{\widehat{x}_i\}_{i=1}^T$, so that it is non-anticipative and each $\widehat{x}_i$ satisfies the constraint at each stage. The main idea in the construction is to simulate the process in finding a solution for the original problem, but this time in a roll out manner.

To find a feasible solution for $\vec{\xi}^T$, we go from stage 2 to stage T. At each stage t, we fix the previously-made decisions $\{\widehat{x}_i\}_{i=1}^{t-1}$ and independently sample a subtree with root node $\Gamma(\vec{\xi}^t)$ using the conditional sampling method described in Section 3.1.1. Then the decision $\widehat{x}_t$ is made by finding the optimal value of the following optimization problem:

$$\widehat{x}_t = \operatorname*{argmin}_{x_t} \frac{1}{n_{t+1}} \sum_{\xi_{t+1,i} \in D(\vec{\xi}^t)} \widehat{f}_{t+1}((\vec{\widehat{x}}^{t-1}, x_t); (\vec{\xi}^t, \xi_{t+1,i})) \tag{8}$$

where for $\tau = t+1, \ldots, T-1$, $\widehat{f}_\tau$ is recursively defined as

$$\widehat{f}_\tau(\vec{x}^{\tau-1}; \vec{\xi}^{\tau,i}) = \min_{x_\tau} \frac{1}{n_{\tau+1}} \sum_{\xi_{\tau+1,j} \in D(\vec{\xi}^{\tau,i})} \widehat{f}_{\tau+1}((\vec{x}^{\tau-1}, x_\tau); (\vec{\xi}^{\tau,i}, \xi_{\tau+1,j})), \tag{9}$$

and

$$\widehat{f}_T(\vec{x}^{T-1}; \vec{\xi}^{T,i}) = \min_{x_T} f'_T(\vec{x}^T; \vec{\xi}^{T,i})$$

Since the subtrees generated at each stage are independent, the decisions we obtain are non-anticipative.

We note here the similarity between the above problem and the one in Section 3.1.2. Indeed, after we sample the subtree from a node, that subtree essentially defines a new MSSP problem.

We summarize the process below:

```
Input  : first stage solution $\widehat{x}_1$, scenario $\vec{\xi}^T$.
Output: feasible, non-anticipative solution $\{\widehat{x}_i\}_{i=1}^T$
Initialization;
for $t \leftarrow 2$ to $T$ do
    Independently sample a new subtree $\Gamma(\vec{\xi}^t)$ ;
    Solve the associated optimization problem (8) ;
    Denote the optimal value as $\widehat{x}_t$ ;
end
return $\{\widehat{x}_i\}_{i=1}^T$
```

Algorithm 1: Generating a Feasible Solution

3.2 Confidence Interval

We are now ready to construct a confidence interval for the optimality gap by finding an upper bound and a lower bound for the optimal function value z^* as in(Mak et al. 1999, Chiralaksanakul and Morton 2004). Before we proceed, note that with our notation, the function value for a given solution under a given scenario is $f'_T(\vec{x}^T; \vec{\xi}^T)$.

3.2.1 Upper Bound

We start by identifying an upper bound for the optimal value. For a given first stage decision $\widehat{x}_1$, we can find a feasible solution $\tilde{x}(\widehat{x}_1, \vec{\xi}^T)$ for all stages given any scenario $\vec{\xi}^T$ using Algorithm 1. It is easy to see that $\mathbb{E}f'_T(\tilde{x}(\widehat{x}_1, \vec{\boldsymbol{\xi}}^T); \vec{\boldsymbol{\xi}}^T)$ is an upper bound for z^*, as it is feasible (assuming relatively complete recourse), but not necessarily optimal, so we have

$$\mathbb{E}f'_T(\tilde{x}(\widehat{x}_1, \vec{\boldsymbol{\xi}}^T); \vec{\boldsymbol{\xi}}^T) \geq z^*$$

While it is likely not possible to compute $\mathbb{E}f'_T(\tilde{x}(\widehat{x}_1, \vec{\boldsymbol{\xi}}^T); \vec{\boldsymbol{\xi}}^T)$ directly, we can form a point estimator by generating n_u i.i.d. scenarios. $\vec{\xi}^{T,i}$, and compute the corresponding function values

$$w_i = f'_T(\tilde{x}(\widehat{x}_1, \vec{\xi}^{T,i}); \vec{\xi}^{T,i}).$$

Let $U_n = \frac{1}{n_u}\sum_{i=1}^{n_u} w_i$ be the sample mean, and s_u^2 be the standard sample variance. Then, we may derive an approximate one-sided confidence interval for $\mathbb{E}f'_T(\tilde{x}(\widehat{x}_1, \vec{\boldsymbol{\xi}}^T); \vec{\boldsymbol{\xi}}^T)$:

$$z^* \leq \mathbb{E}f'_T(\tilde{x}(\widehat{x}_1, \vec{\boldsymbol{\xi}}^T); \vec{\boldsymbol{\xi}}^T) \leq U_n + \frac{t_{n_u-1,\alpha} s_u}{\sqrt{n_u}} \qquad w.p. \approx\ 1-\alpha, \tag{10}$$

here $t_{n_u-1,\alpha}$ is the upper $1-\alpha$ quantile for a t-distribution with $n_u - 1$ degrees of freedom.

We may also derive a point-estimator for $\mathbb{E}f'_T(\tilde{x}(\widehat{x}_1, \vec{\boldsymbol{\xi}}^T), \vec{\boldsymbol{\xi}}^T)$ by generating n_u independent sampled scenario trees $\Gamma(\vec{\xi}^{1,i})$ using the conditional sampling method in Section 3.1.1. Each scenario tree can be regarded as a batch of scenarios as each leaf node represent a scenario. Though within each $\Gamma(\vec{\xi}^{1,i})$ we no longer have i.i.d. scenarios, we can still find an estimator using the same method as described above. Consider each tree $\Gamma(\vec{\xi}^{1,i})$, for the N_i leaf nodes on $\Gamma(\vec{\xi}^{1,i})$, we can find a feasible solution for each and compute the corresponding function values as $\{w_{i,j}\}_{j=1}^{N_i}$. Aggregating the function values on the leaf nodes of $\Gamma(\vec{\xi}^{1,i})$ gives us a point estimator:

$$W_i = \frac{1}{N_i}\sum_j w_{i,j} \tag{11}$$

We note here that if two scenarios on the same tree share the same realizations to stage t, i.e. if both of their paths go through node $\vec{\xi}^t$, then we can sample one scenario tree $\Gamma(\vec{\xi}^t)$ to find $\widehat{x}^t$ at stage t for both scenarios, as long as the sampled scenario tree is constructed using unbiased conditional sampling.

Now let U_n be the sample mean for W_i and s_u^2 be the standard sample variance. Since the scenario trees are generated in an i.i.d. manner, by CLT we are able to form an approximate one-sided confidence interval for $\mathbb{E}f'_T(\tilde{x}(\widehat{x}_1, \vec{\boldsymbol{\xi}}^T); \vec{\boldsymbol{\xi}}^T)$ using the same expression as (10):

$$z^* \leq \mathbb{E}f'_T(\tilde{x}(\widehat{x}_1, \vec{\boldsymbol{\xi}}^T); \vec{\boldsymbol{\xi}}^T) \leq U_n + \frac{t_{n_u-1,\alpha} s_u}{\sqrt{n_u}} \qquad w.p. \approx\ 1-\alpha,$$

3.2.2 Lower Bound

For reference, the general expression for a stochastic optimization problem as defined in Mak et al. (1999) is in the form

$$z^* = \min_{x\in X} \mathbb{E}f(x, \boldsymbol{\xi}), \tag{12}$$

with associated approximating problem

$$z_n^* = \min_{x\in X} \frac{1}{n}\sum_{i=1}^{n} f(x; \xi_i), \tag{13}$$

then as proved in Mak et al. (1999), we have

$$\mathbb{E}z_n^* \leq z^*$$

The same conclusion holds when we extend it to MSSP. To be exact, let z^* be the optimal value for (1) and $\widehat{z}^*$ be the optimal value for (6). Then we have

$$\mathbb{E}\widehat{z}^* \leq z^*$$

We refer readers to Chiralaksanakul and Morton (2004) for the proof of the result.

As before, it is usually not possible to compute $\mathbb{E}\widehat{z}^*$ directly. To find an estimator for $\mathbb{E}\widehat{z}^*$, we can sample n_l independent scenario trees using the conditional sample method mentioned in Section 3.1.1, and solve for the optimal value of problem (6). As a practical matter, for some types of problems, it may not be possible to compute the optimal value exactly, but only a lower bound. Denote the lower bounds as $(\ell_1, \ldots, \ell_{n_l})$ and compute the sample mean L_n and the standard sample variance s_l^2 of those lower bounds. This allows us to construct approximate one-sided confidence interval as

$$z^* \geq \mathbb{E}\widehat{z}^* \geq L_n - \frac{t_{n_l-1,\alpha} s_l}{\sqrt{n_l}} \qquad w.p. \approx 1 - \alpha. \tag{14}$$

Here $t_{n_l-1,\alpha}$ is the upper $1-\alpha$ quantile for a t-distribution with $n_l - 1$ degrees of freedom.

3.2.3 Confidence interval

The Boole–Bonferroni inequality, (10) and (14) together yield the following approximate $(1-2\alpha)$-level confidence interval for the optimality gap at $\widehat{x}$:

$$\left[0, U_n - L_n + \frac{t_{n_u-1,\alpha} s_u}{\sqrt{n_u}} + \frac{t_{n_l-1,\alpha} s_l}{\sqrt{n_l}}\right].$$

We can use the same "common random number" technique used by Mak et al. (1999) in the two-stage case to reduce variance in the MSSP case. To be precise, when given a first stage decision $\widehat{x}_1$, we want to form n_g sample scenario trees $\Gamma(\vec{\xi}^{1,i})$. For each $\Gamma(\vec{\xi}^{1,i})$, we can find a point estimator W_i for the upper bound using Equation (11) and the methods in Section 3.2.1, together with a lower bound ℓ_i using the methods in Section 3.2.2. Then $G_i = W_i - \ell_i$ can be regarded as an estimator for the optimality gap. Furthermore, let $\bar{G}$ and s_g^2 be the sample mean and sample variance of G_i. Since $\Gamma(\vec{\xi}^{1,i})$ are constructed in an i.i.d. manner, we can use

$$\left[0, \bar{G} + \frac{t_{n_g-1,\alpha} s_g}{\sqrt{n_g}}\right] \tag{15}$$

as the approximate $(1-\alpha)$-level confidence interval for the optimality gap.

Below we summarize the procedure for finding the optimality gap using the "common random number" technique.

```
Input  : first stage solution $\widehat{x}_1$, sample size $n_g$
Output: approximate $(1-\alpha)$ confidence level for optimality gap
Initialization;
for $i \leftarrow 1$ to $n_g$ do
    Independently sample scenario tree $\Gamma(\vec{\xi}^{1,i})$ ;
    solve the approximate problem (6) associated with $\Gamma(\vec{\xi}^{1,i})$ ;
    Denote the lower bound of the optimal function value as $\ell_i$ ;
    Use tree traversal to find an feasible solution for each leaf node on $\Gamma(\vec{\xi}^{1,i})$
      as in Algorithm 1 ;
    Aggregate the corresponding function values as in (11) to find upper
      bound $W_i$ ;
    Compute $G_i = W_i - \ell_i$ ;
end
Compute sample mean $\widehat{G}$ and sample variance $s_g^2$ ;
return $\left[0, \bar{G} + \frac{t_{n_g-1,\alpha} s_g}{\sqrt{n_g}}\right]$
```

Algorithm 2: MMW Confidence Interval Estimation

3.3 Sequential Sampling Procedure

In comparison to (Mak et al. 1999) which constructs the confidence interval for a single solution, (Bayraksan and Morton 2011, Bayraksan and Pierre-Louis 2012) proposed procedures that estimate the optimality gap for a sequence of feasible solutions with an optimal limit point, which in turn enable them to develop a stopping criteria for the sequential sampling procedure.

3.3.1 Relative Width Sequential Sampling

We start by describing a trivial multi-stage extension to the sequential sampling procedure in (Bayraksan and Morton 2011), which estimates the optimality gap of a sequence of candidate first stage solutions for a multistage program, and terminates when the point estimate of the optimality gap satisfies the stopping rule.

While the procedure in finding an upper bound is similar, to find a lower bound for the optimality gap, at each iteration k we sample n_k scenarios $\vec{\xi}^{T,1}, \vec{\xi}^{T,2}, \ldots, \vec{\xi}^{T,n_k}$ from the joint distribution $\{\boldsymbol{\xi}_t\}_2^T$ and construct a *horsetail* scenario tree, where each sampled scenario represents root-to-leaf path without any overlap. This bypasses the non-anticipativity constraints of the MSSP problem (1), except at the root node. In essence, the discrete MSSP problem(6) for the horsetail scenario tree is equivalent to the following problem,

$$\widehat{z}^* = \min_{x_1 \in X_1} \left\{ \frac{1}{n_k} \sum_i \min_{x_2,\ldots,x_T} \psi'_T(\vec{x}^T; \vec{\xi}^{T,i}) \right\}. \qquad (16)$$

Where

$$\psi'_T(\vec{x}^T; \vec{\xi}^{T,i}) = \begin{cases} f'(\vec{x}^T, \vec{\xi}^{T,i}) & \text{if } x_t \in X_t(\vec{x}^{t-1}; \vec{\xi}^{t,i}) \text{ for } t = 2, \ldots, T-1 \\ \infty & \textit{otherwise} \end{cases}$$

```
Input  : candidate first stage solutions $\widehat{x}_{1,k}$, scenario set
         $\vec{\xi}^{T,1}, \dots, \vec{\xi}^{T,n_k}\}$
Output: point estimator and variance estimator of the optimality gap
         for the input candidate solution
Initialization;
For each scenario $\vec{\xi}^{T,i}$, find a feasible solution $\vec{\widehat{x}}^{T,i}$ as in Algorithm 1 and
  compute an upper bound using the function value $w_i = f'_T(\vec{\widehat{x}}^{T,i}; \vec{\xi}^{T,i})$ ;
Solve problem (16) associated with the horsetail scenario tree that is
  generated by the $n_k$ scenarios. Denote the lower bound for optimal
  function value as $\ell$ ;
Compute the optimality gap estimator $G_k = \frac{1}{n_k}\sum_i (w_i - \ell)$ and the
  variance estimator $s_k^2 = \frac{1}{n-1}\sum_i (w_i - \ell - G_k)^2$ ;
return $G_k, s_k^2$
```

Algorithm 3: Generating estimators of the optimality gap

Since this is a relaxation of the original problem (1), we can expect the optimal solution for (16) to be a valid (but possibly not consistent) statistical lower bound for problem (1). We note here that in practice we may only obtain a lower bound for $\widehat{z}^*$, which can be used in constructing a valid confidence interval.

Algorithm 4 shows our implementation of a multi-stage version of the sequential sampling procedure in Bayraksan and Morton (2011). The main theorem in their paper, using their notation, states that for input parameters h, ϵ and ϵ', the returned results for a two-stage problem satisfies

$$\liminf_{h \downarrow h'} \mathbb{P}\left(\mathbb{E} f'_T(\vec{\widehat{x}}^{T,K}(\vec{\boldsymbol{\xi}}^T), \vec{\boldsymbol{\xi}}^T) - z^* \leq h s_K + \epsilon\right) \geq 1 - \alpha$$

For multi-stage problems, the same inequality stands but the returned confidence interval may be wider.

3.3.2 Fixed Width Sequential Sampling

Algorithm 4 returns a candidate solution when its optimality gap falls below a user-specified fraction of the sample variance, so the width of the confidence interval can be considered as a *relative-width.* Such stopping criteria may result in a wide confidence interval when the sample variance is large, as mentioned in Bayraksan and Pierre-Louis (2012), who later proposed an alternative stopping rule that aim to find an ϵ-optimal solution.

To be exact, let $G_n(x)$ be the point estimator of the optimality gap, $\nu_n(x)$ the sampling error, and $h(n)$ the inflation factor that shrinks to zero as $n \to 0$, their proposed stopping rule returns the candidate solution when $G_n(x) + \nu_n(x) + h(n) \leq \epsilon$. In our software, the point estimator $G_n(x)$ is computed in the same way as in Algorithm 4, and we use $\nu_n(x) = \frac{t_{n-1,\alpha} s_n(x)}{\sqrt{n}}$ as the sampling error. Here $t_{n-1,\alpha}$ is the upper $1-\alpha$ quantile

```
Input  : values for $h > h' > 0$, $\epsilon > \epsilon' > 0, 0 < \alpha < 1, p > 0$, candidate
         first stage solutions $\{\widehat{x}_{1,k}\}$, resampling frequency $k_f$
Output: candidate first stage solution $\widehat{x}_{1,K}$ that meets the stopping
         criteria, and a $(1-\alpha)$-level CI on its optimality gap.
Initialization;
for $k \leftarrow 1$ to MaxIter do
    Compute the sample size $n_k$ for the current iteration with
    $n_k = \left\lceil \frac{1}{(h-h')^2}(c_p + 2p\ln^2 k) \right\rceil, \qquad c_p = \max\left\{ 2\ln\left( \sum_j j^{-p\ln j}/\sqrt{2\pi\alpha} \right), 1 \right\}$
    if $k_f$ divides $k$ then
        independently sample $n_k$ scenarios $\vec{\xi}^{T,1}, \vec{\xi}^{T,2}, \dots, \vec{\xi}^{T,n_k}$
    else
        independently sampled $n_k - n_{k-1}$ new scenarios
        $\vec{\xi}^{T,n_{k-1}+1}, \dots, \vec{\xi}^{T,n_k}$ and bundle it with the previously sampled
        scenarios $\vec{\xi}^{T,1}, \vec{\xi}^{T,2}, \dots, \vec{\xi}^{T,n_{k-1}}$
    end
    Use Algorithm 2 to compute the optimality gap estimator $G_k$ and
     the variance estimator $s_k$ for $\widehat{x}_{1,k}$ using the sampled scenarios;
    If $G_k \le h's_k + \epsilon'$ then set $K = k$ ; break ;
end
return $\widehat{x}_{1,K}$ , $(1-\alpha)$-level CI $[0, hs_K + \epsilon]$
```
Algorithm 4: Sequential Sampling

for a t-distribution with $n-1$ degrees of freedom. We present the algorithm for completeness here in Algorithm 5.

4 Software and Computational Results

4.1 Software Particulars

Both MMW and sequential sampling are supported by classes and functions that are consistent with the mpi-sppy library and there is also a command-line program for MMW confidence intervals. In addition to the usual requirement of parameters, which are specified as a dictionary, the software requires as input a function that returns a Pyomo (Hart et al. 2011, Bynum et al. 2021) model instantiated with data for a given scenario along with scenario tree information, which comes in the form of branching factors for multi-stage problems.

The MMW software requires input of an $\widehat{x}$ for the root node of the scenario tree, which is a specification of values for the non-anticipative variables. The mpi-sppy library provides methods to output $\widehat{x}$ in a format that can be read by the MMW software, so upper bound software used by any method supported by mpi-sppy can create the solution. The mpi-sppy

```
Input  : sample size schedules $\{m_k\}, \{n_k\}$, inflation factor $h(n)$
         ,resampling frequency $k_f^n$
Output: candidate first stage solution $\widehat{x}_{1,K}$ whose estimated
         optimality gap falls under $\epsilon$.
Initialization;
for $k \leftarrow 1$ to MaxIter do
    Independently sample a scenario tree with $m_k$ leaf nodes and solve
     the approximate problem (6) to obtain candidate first stage
     solution $\{\widehat{x}_{1,k}\}$
    if $k_f^n$ divides $k$ then
        independently sample $n_k$ scenarios $\vec{\xi}^{T,1}, \vec{\xi}^{T,2}, \ldots, \vec{\xi}^{T,n_k}$
    else
        independently sampled $n_k - n_{k-1}$ new scenarios
         $\vec{\xi}^{T,n_{k-1}+1}, \ldots, \vec{\xi}^{T,n_k}$ and bundle it with the previously sampled
         scenarios $\vec{\xi}^{T,1}, \vec{\xi}^{T,2}, \ldots, \vec{\xi}^{T,n_{k-1}}$
    end
    Use Algorithm 2 to compute the optimality gap estimator $G_k$ and
     the variance estimator $s_k$ for $\widehat{x}_{1,k}$ using the sampled scenarios;
    If $G_k + \frac{t_{n-1,\alpha} s_n(x)}{\sqrt{n}} + h(n) \leq \epsilon$ then set $K = k$; break ;
end
return $\widehat{x}_{1,K}$
```
Algorithm 5: Fixed Width Sequential Sampling

library has heuristics for finding computing upper bounds given a scenario tree and does not required relatively complete recourse. The sequential sampling software does not require $\widehat{x}$ as input because it is one of the outputs.

Fixed-width sequential sampling requires only the desired width and sample size as input, but we also support the fully parameterized version relative-width using these parameters:

- p is used to control the sample size. Given K, it is possible to find p that minimize n_K. However as shown in Bayraksan and Morton (2011), given p, the variation in sample size actually is moderate for different K.
- h is used to control the desired width of the confidence interval
- h' is used to control the trade-off between inflation in the confidence interval and the sample size: the closer h' is to h, the smaller the "inflation" of the CI statement is in trade of a greater sample size.
- ϵ, ϵ' is used here to ensure finite termination and theoretical convergence, and we require $h - h'$ to some extent be proportional to $\epsilon - \epsilon'$

4.2 Overview of Experiments

The models used are summarized in Table 1. The APL1P and GBD are included only to validate the 2-stage MMW and sequential sampling procedures because they are used in (Bayraksan and Morton 2011) and (Bayraksan and Pierre-Louis 2012). APL1P is a two generator power plant model with stochastic demand and availability (Infanger (1992)). GBD is an aircraft allocation problem with stochastic demand (Ferguson and Dantzig (1956)). Aircond, or the air conditioning problem, is a single-product planning model with overtime production and inventory carry-forward. This model can be any number of stages, but we use 3-stage examples for this paper. We form two distinct models, one without any start-up production costs (Aircond 1), and one which includes these start-up costs (Aircond 2) and thus form mixed-integer programs. For more details on Aircond see Appendix A. The z^* reported for both Aircond models are underestimates formed by approximating $\mathbb{E}z_n^*$ as described in Section 3.2.2.

Model	**# of first stage variables**	**# of second stage variables**	**# of 3rd stage variables**	**# of stochastic parameters**	**# of scenarios**	z^*
APL1P	2	9	NA	5	1280	24,642.32
GBD	17	10	NA	5	646,425	1655.63
Aircond 1	2	2	2	1	∞	≥ 566.43
Aircond 2	3	3	3	1	∞	≥ 1456.30

Table 1: Models used for computational results

When constructing confidence intervals for stochastic optimization problems there are many different questions to be considered. Comparing confidence interval procedures is typically considers two aspects. The first concern is whether both the procedures have the desired coverage probability in practice, i.e. can we produce these confidence intervals via sampling techniques. We report coverage estimates $\widehat{p}$, which estimates the fraction of the time the quantity (e.g. the gap, G) fell in the computed confidence interval. Secondly, we would like to know which procedure produces a tighter interval since this gives us a better idea of our optimal solution on average. In this section we will also consider the time required to produce confidence intervals.

4.3 Results for APL1P and GBD

Table 2 reports the results of 100 trials of the fixed-width sequential sampling procedure with a 90% confidence level. For each model we seek a fixed-width optimality gap of 1% using a resampling frequency k_f=1. The 90% confidence interval on $\widehat{p}$ reported are computed as described in Section 4.4. However, in the case of APL1P, we may calculate $\widehat{z}$ exactly with little computational effort, and z^* is known, so we need not form a confidence interval for our coverage estimate. In this case we simply observe whether or not $\widehat{z}-z^* < G$. For GBD we form the confidence interval on $\widehat{z}_n^*$ by sampling 10 independent sample trees per trial, and we again know the true z^*, so we observe whether or not $z^* \in [\bar{\widehat{z}}_n - \epsilon_{\widehat{z}}, \bar{\widehat{z}}_n + \epsilon_{\widehat{z}}]$.

Model	FW Gap	90% CI on $\widehat{p}$	90% CI on T	90% CI on time (s)
GBD	8	0.83 ± 0.05	1.05 ± 0.06	7.34 ± 0.47
APL1P	123	0.87 ± 0.04	11.87 ± 1.95	40.02 ± 17.67

Table 2: GBD, APL1P Fixed Width Sequential Sampling Results

The results are similar in terms of permissiveness of the intervals to those in (Bayraksan and Morton 2011) and (Bayraksan and Pierre-Louis 2012).

4.4 Results for Aircond

For the Aircond problems, the true (without sampling) optimal solution cannot be known exactly, so in order to compute coverage, $\widehat{p}$, we do the following:

- Solve a 'large' SP$_n$ to produce an estimate $x_{n,1}^*$ of x_1^*
- Solve a 'large' number of subproblems with first stage solution $x_{n,1}^*$ to produce a 99% confidence interval of the estimate of the objective value, z_n^*
- For any given candidate solution $\widehat{x}_1$, solve a 'large' number of subproblems with first stage solution $\widehat{x}_1$ to produce a 99% confidence interval around $\widehat{z}$
- Form a gap estimate $\widehat{z} - z^*$ from these approximations

We denote the estimate of the optimal objective value as

$$[\bar{z}_n^* - \epsilon_{z^*}, \bar{z}_n^* + \epsilon_{z^*}],$$

and the estimate of the objective function at any $\widehat{x}_1$ as

$$[\bar{\widehat{z}}_n - \epsilon_{\widehat{z}}, \bar{\widehat{z}}_n + \epsilon_{\widehat{z}}].$$

In the remainder of the results the proxy for coverage of a confidence interval procedure which produces a gap estimate G, will be whether or not we have the following containment:

$$[\bar{z}_n^* - \epsilon_{z^*}, \bar{z}_n^* + \epsilon_{z^*}] \subseteq [\bar{\widehat{z}}_n + \epsilon_{\widehat{z}} - G, \bar{\widehat{z}}_n + \epsilon_{\widehat{z}}].$$

Tabel 3 shows the approximations used in our results.

Model	n	$x^*_{n,1}$	90% CI on z^*_n
Aircond 1	10^6	[134.80, 0]	567.75 ± 0.8
Aircond 2	10^4	[135.36, 0]	567.24 ± 2.33

Table 3: Optimal objective value approximations for Aircond 1, Aircond 2

4.4.1 Sequential Sampling Results

Table 4 shows the results of running the fixed-width sequential sampling procedure to find 3% and 2% optimal first-stage feasible solutions for Aircond 1, and 5% and 4% optimal first-stage feasible solutions for Aircond 2. In this section we abuse our notation slightly and use T to denote the number of iterations of the sequential sampling procedure, i.e. the number of times we resample and increase sample size, and t to denote the wall clock time of a single procedure, not to be confused with stage t stage T stage of the optimization problem. We report a first-stage solution computed by the sequential sampling procedure, $\widehat{x}_1$, the number of iterations, T, and the run time of the sequential sampling procedure, t. We also report 90% confidence intervals on T and t for 100 separate trials of the procedure. Note that the $\widehat{x}_1$ reported is a from single instance of the procedure, as it will be used for further study. Lastly, the 99% confidence intervals on $\widehat{z}$, as discussed in the previous section, are constructed by solving several thousand sample trees for which $\widehat{x}_1$ is a feasible first-stage solution.

In each trial we set the minimum sample size for the fixed-width procedure, c_0, to be 100, and we set our confidence level to 95%.

In practice, for both Aircond models, the procedure fails to converge for optimality gaps less that 2%, though in our case study the approximate optimality gap is lower than the desired width of the confidence interval. For Aircond 1, we seek 3% and 2% optimal solutions, and our approximate optimality gaps are 0.5% and 0.8% optimal, respectively. For Aircond 2 we seek 5% and 4% optimality gap fixed widths, and our approximate optimality gaps are 0.6% and .9% optimal, respectively. The approximate optimality gap, reported ($\approx \widehat{z} - z^*$), is the lower estimate of z^* subtracted from the upper estimate of $\widehat{z}$, i.e.

$$(\approx \widehat{z} - z^*) := \left(\bar{\widehat{z}}_n + \epsilon_{\widehat{z}}\right) - \left(\bar{z}^*_n - \epsilon_{z^*}\right).$$

Model	FW Gap	$\widehat{x}_1$	T $[\bar{T} \pm \epsilon_T]$	Seq. Samp. time (s) $[\bar{t} \pm \epsilon_t]$	99% CI on $\widehat{z}$	$\approx \widehat{z} - z^*$
Aircond 1	16	$[117.45, 0.0]$	2 $[1.63 \pm 0.11]$	21.47 $[17.53 \pm 1.26]$	570.23 ± 0.91	4.18
Aircond 1	11	$[144.38, 0.0]$	3 $[2.65 \pm 0.25]$	33.07 $[29.05 \pm 2.84]$	568.24 ± 0.80	2.09
Aircond 2	72	$[159.34, 0.0, 1.0]$	7 $[4.84 \pm 0.45]$	162.92 $[119.51 \pm 11.63]$	1465.9 ± 0.85	12.36**
Aircond 2	58	$[200.0, 0.0, 1.0]$	10 $[11.58 \pm 1.53]$	252.41 $[330.02 \pm 67.53]$	1470.11 ± 0.85	16.57**

Table 4: Feasible first-stage candidate solutions from multi-stage sequential sampling procedure

The $\widehat{x}_1$'s reported in Table 4 are those give by the first trial. In each of the candidate solutions found, the approximate optimality gap is well below the desired fixed-width gap of the procedure, with a level of permissiveness similar to those reported in the original papers. In the next section we observe the performance of the multi-stage MMW procedure which may be desired for constructing a tighter confidence interval on the optimality gap.

4.4.2 MMW Results

In this section we repeat the multi-stage MMW procedure with each of the candidate solutions found in the previous section for Aircond 1 and Aircond 2. Table 5 shows the results of 100 repetitions of the MMW procedure for each candidate solution. For each trial we independently sample 20 scenario trees for several different branching factors: $[10, 10]$, $[20, 10]$, $[30, 10]$, and $[20, 20]$. We report the number of samples used per tree as m. We report 90% confidence intervals on the gap estimate, G, and the run time t. It is clear that as we initially increase the number of samples used in the MMW procedure, our gap estimate decreases. As m increases the average gap estimate decreases, and we near our approximate optimality gaps reported in Table 4.

Model	FW Gap	$\widehat{x}_1$	m	90% CI on G	90% CI on time (s)
Aircond 1	16	$[117.45, 0.0]$	100	8.43 ± 0.26	38.79 ± 0.02
			200	6.38 ± 0.17	81.81 ± 0.04
			300	5.6 ± 0.13	123.84 ± 0.21
			400	5.55 ± 0.15	168.07 ± 0.36
Aircond 1	11	$[144.38, 0.0]$	100	5.74 ± 0.15	38.88 ± 0.02
			200	3.91 ± 0.09	81.58 ± 0.04
			300	3.15 ± 0.07	122.86 ± 0.13
			400	3.02 ± 0.08	167.86 ± 0.18
Aircond 2	72	$[159.34, 0.0, 1.0]$	100	21.42 ± 0.4	116.54 ± 0.26
			200	17.83 ± 0.23	235.19 ± 0.43
			300	16.36 ± 0.19	363.52 ± 0.77
			400	11.49 ± 0.23	469.82 ± 1.69
Aircond 2	58	$[200.0, 0.0, 1.0]$	100	26.42 ± 0.33	118.53 ± 0.29
			200	22.71 ± 0.22	246.12 ± 0.49
			300	21.45 ± 0.18	379.75 ± 0.78
			400	15.64 ± 0.22	475.48 ± 1.57

Table 5: Multi-stage MMW Results.

Additional computational experiments are described in Appendix B.

5 Conclusions and Directions for Further Research

This paper describes software supporting practical creation of confidence intervals for two-stage and multi-stage stochastic programs as well as supporting ongoing research. Results presented here, or the software, can benchmark improved methods. We have used two small, standard problem instances as well as a scalable example, the Aircond model, which is included in mpi-sppy and easily reproduced in other modeling environments.

We have provided a case study comparing the MMW procedure to the Sequential Sampling procedure. Such comparisons are not straightforward and perhaps comparison methodology is worthy of further research. The MMW procedure computes a confidence interval on the optimality gap for a given first stage solution $\widehat{x}_1$, whereas sequential sampling computes a first stage solution $\widehat{x}_1$ which satisfies a pre-specified optimality gap. Our results suggest the sequential sampling procedure produces good candidate first stage solutions, which MMW is incapable of, but that in practice MMW may be preferred to compute an accurate optimality gap.

Another potential direction for further research involves multi-stage sequential sampling. In our procedures we use a horsetail scenario tree to construct a lower bound for the optimal function value, which in many cases can be an underestimation as the non-anticipativity constraints are ignored. A possible way to construct a tighter lower bound that takes into account the non-anticipativity would be as follows: at each iteration, we draw one single scenario tree with n_k leaf nodes; the upper bound can be computed in a similar way as before, but instead of using the horsetail scenario tree, we can construct a lower bound by solving for the approximated MSSP problem associated with the sampled scenario tree. The rest of the steps should be similar. However, it's expected that the variance of the optimality gap estimators generated within a single scenario tree would be lower compared to those generated with independent scenarios, as they are not strictly independent.

The software and problem instances are available for download at `https://github.com/Pyomo/mpi-sppy`. Software documentation is available at `readthedocs<https://mpi-sppy.readthedocs.io/en/latest/`. We hope that the software and extensions to multi-stage problems will enhance research and practice in the are of optimization under uncertainty.

Acknowledgement

This work was supported, in part, by the US Department of Energy's ARPA-E Performance-based Energy Resource Feedback, Optimization, and Risk Management (PERFORM) program.

A Aircond Model

We make use of a multi-period production planning model with overtime production and inventory carry-forward based on a model called "Air Conditioning" in lecture notes by Jeff Linderoth.

A.1 Variables

Given a set of stages (periods) $\mathcal{T} = \{1, 2, \ldots, T\}$, and a single product, we define each variable over each stage $t \in \mathcal{T}$. For each stage we have the following:

$x_t \in \mathbb{R}_{\geq 0}$ number of each product to make in regular time (decided right before period t)

$w_t \in \mathbb{R}_{\geq 0}$ number of each product to make in overtime (decided right before period t)

$y_t \in \mathbb{R}$ number of each product to carry forward (determined by x, w and the demand for period t)

$\delta_0 \in \{0, 1\}$ start-up variable

A.2 Parameters

The following quantities are all taken as parameters in our construction of the problem. For our purposes, only the demand, D_t, is stochastic, though this problem may be extended to have stochastic inventory capacity and costs. Each parameter is real-valued and all but N_t are non-negative. All are known the beginning of period 1 except for the demand.

D_t demand in period t (known at the end of period t)

I_t single period inventory cost

N_t single period negative inventory cost (failure to meet demand)

K_t regular time capacity

C_t regular time production cost

O_t overtime production cost

S_t start-up cost

y_0 beginning inventory

A.3 Objective Function

The objective is to minimize the expected cost of production subject to inventory and demand constraints. At each stage we can only produce so much of each product and we must meet the given demand. Using the notation introduced in (1), with $\boldsymbol{\xi} = \mathbf{D}$, and, subsequently,

$$\vec{\xi}^t = (\xi_2, \xi_3, \ldots, \xi_t) = (D_2, D_3, \ldots, D_t),$$

the objective is defined recursively by:

$$\begin{aligned}\min_{x_1} \quad & \left(\delta_1 \cdot S_1 + C_1 \cdot x_1 + O_1 \cdot w_1 + \sum_{i=1}^{n} \max\{I_{1,i} y_{1,i}, N_1 y_{1,i} + \mathbb{E}_{\boldsymbol{\xi}_2} \phi_2(x_1; \boldsymbol{\xi}_2)\right) \\ s.t. \quad & x_1 \leq K_1, \\ & y_0 + x_1 + w_1 - y_1 = D_1 \\ & M\delta_1 \geq x_1 + w_1\end{aligned}$$

where

$$\begin{aligned}\phi_t(x_{t-1}; \vec{\xi}^t) = \min_{x_t} \quad & \left(\delta_t \cdot S_t + C_t \cdot x_t + O_t \cdot w_t + \sum_{i=1}^{n} \max\{I_{t,i} y_{t,i}, N_t y_{t,i}\} + \mathbb{E}_{\vec{\boldsymbol{\xi}}^{t+1}} \left[\phi_{t+1}(x_t; \vec{x}^{t-1}, \vec{\boldsymbol{\xi}}^{t+1} \mid \vec{\boldsymbol{\xi}}\right.\right. \\ s.t. \quad & x_t \leq K_t, \\ & y_{t-1} + x_t + w_t - y_t = D_t \\ & M\delta_t \geq x_t + w_t\end{aligned}$$

and

$$\begin{aligned}\phi_T(x_{T-1}; \vec{\xi}^T) = \min_{x_T} \quad & \left(\delta_T \cdot S_T + C_T \cdot x_T + O_T \cdot w_T + \sum_{i=1}^{n} \max\{I_{T,i} y_{T,i}, N_T y_{T,i}\}\right) \\ s.t. \quad & x_T \leq K_T, \\ & y_{T-1} + x_T + w_T - y_T = D_T \\ & M\delta_T \geq x_1 + w_1.\end{aligned}$$

Here $M = T \max\{x_t + w_t\}_{t \in \mathcal{T}}$, where T is the total number of stages.

A.4 Non-integer Sample Data

For the experiments conducted in this paper we work with a significantly simplified model in order to study a multi-stage problem with reasonable computational effort.

We assume a 3-stage ($\mathcal{T} = \{1, 2, 3\}$) problem with no start-up costs. Furthermore, we fix the parameters as:

$y_0 = 100$

$K = 200$

$C = 1$

$O = 3$

$I_t = 0.5$, $I_T = -0.8$,

and assume that the demand $\mathbf{D}$ is a random walk with barriers at 0 and 400 which follows a normal distribution $\mathcal{N}(0, 40)$. In other words we sample i.i.d. variables R_t from $\mathcal{N}(0, 40)$ and define D_{t+1} as

$$D_{t+1} = \begin{cases} 0 & \text{if } D_t + R_t \leq 0 \\ 400 & \text{if } D_t + R_t \geq 400 \\ D_t + R_t & \text{otherwise.} \end{cases}$$

A.5 MIP Sample Data

To look at examples where individual scenarios might reasonably solve with a gap between upper and lower bounds, we also consider mixed-integer programs by including start-up costs at each stage. In this case all the previous data remains the same and we include the following parameters:

$N = -5$

$S = 300.$

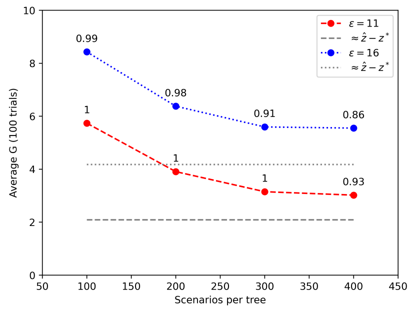
These parameters were chosen so that we observe scenarios in which we choose not to produce at certain stages, and also tuned so that it is not always optimal not to produce to max regular capacity in the first stage. The latter is so that we see variation in our first-stage solution in order to meaningfully study the success of the sequential sampling procedure.

B Additional Computational Results

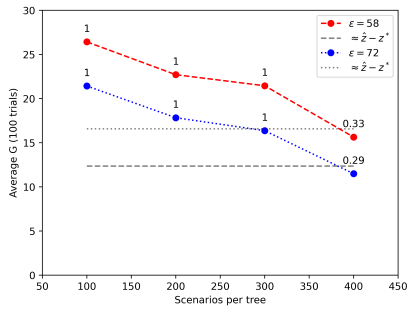
B.1 Additional Aircond Results

To validate the multi-stage MMW procedure, for each repetition of the procedure we determine whether the gap estimate G covers our approximate optimality gap $\approx (\widehat{z} - z^*)$ reported in Table 4. Figure 1 shows the average G for 100 trials for increasing sample size per tree. Over each dot is the approximate coverage, $\widehat{p}$, for the 100 trials, where we say that the procedure covered the approximate optimality gap if the width of the confidence interval is larger than the approximate optimality gap. For

Aircond 2 we see that the use of the under estimate for z^* proved approximate optimality gaps which are too large, and so as we increase the sample size and our gap estimate improves we construct tighter confidence intervals which are do not cover the approximate optimality gap. As tree size increases the MMW procedure provides a tighter optimality gap but decreases in coverage probability. In general, it is best to use both a large sample size per tree (batch size) and a large number trees (batches) to produce a tight confidence interval with the desired coverage probability.



(a) Aircond 1



(b) Aircond 2

Figure 1: Average optimality gap for Aircond models from MMW procedure over 100 trials as tree size increases, as shown in Table 5. The label above is the coverage probability for that particular tree size.

Table 6 shows analogous results to Table 5, but with an MMW procedure using a batch size of 10 rather than 20. The average gap estimate over 100 trials is slightly larger in this case, and the approximate coverage suffers due to the higher variance of the procedure.

Model	FW Gap	$\widehat{x}_1$	m	90% CI on G	90% CI on time (s)
Aircond 1	16	$[117.45, 0.0]$	100	9.4 ± 0.42	19.35 ± 0.02
			200	7.01 ± 0.31	40.48 ± 0.04
			300	6.28 ± 0.19	61.92 ± 0.1
			400	6.2 ± 0.27	84.15 ± 0.1
Aircond 1	11	$[144.38, 0.0]$	100	6.73 ± 0.28	19.23 ± 0.02
			200	4.4 ± 0.17	40.57 ± 0.03
			300	3.26 ± 0.11	62.04 ± 0.13
			400	3.42 ± 0.15	85.2 ± 0.29
Aircond 2	72	$[159.34, 0.0, 1.0]$	100	23.92 ± 0.65	58.01 ± 0.16
			200	19.42 ± 0.44	114.11 ± 0.32
			300	17.39 ± 0.28	177.22 ± 0.48
			400	12.6 ± 0.34	232.21 ± 1.24
Aircond 2	58	$[200.0, 0.0, 1.0]$	100	28.8 ± 0.62	59.06 ± 0.2
			200	24.11 ± 0.37	119.25 ± 0.3
			300	22.21 ± 0.27	184.07 ± 0.54
			400	16.82 ± 0.32	233.22 ± 1.22

Table 6: Multi-stage MMW Results, 10 sample trees per trial

Included in Table 7 are results from 100 trials of the relative-width sequential sampling procedure with a confidence level of 95%.

Model	h, h'	90% CI on G	90% CI on T	90% CI on time (s)
Aircond 1	0.8, 0.55	19.57 ± 0.98	1.17 ± 0.07	9.6 ± 0.62
Aircond 2	1.1, 0.85	87.89 ± 0.85	1.45 ± 0.09	41.47 ± 2.96

Table 7: Aircond RW Sequential Sampling Results.

B.2 Additional 2SSP Results

Table 8 shows the results of running the relative-width sequential sampling procedure for the GBD and APL1P models with a confidence level of 90%, in which we report an additional $\widehat{p}$ as in Table 2.

Model	h, h'	90% CI on G	90% CI on T	90% CI on $\widehat{p}$	90% CI on time (s)
GBD	0.18, 0.01	1.13 ± 0.25	3.79 ± 0.41	0.84 ± 0.05	18.53 ± 2.16
APL1P	0.317, 0.115	204.62 ± 22.29	1.04 ± 0.03	0.91 ± 0.04	4.8 ± 0.13

Table 8: GBD, APL1P RW Sequential Sampling Results (100 trials)

References

Güzin Bayraksan and David P Morton. Assessing solution quality in stochastic programs via sampling. In *Decision Technologies and Applications*, pages 102–122. Informs, 2009.

Güzin Bayraksan and David P Morton. A sequential sampling procedure for stochastic programming. *Operations Research*, 59(4):898–913, 2011.

Guzin Bayraksan and Péguy Pierre-Louis. Fixed-width sequential stopping rules for a class of stochastic programs. *SIAM Journal on Optimization*, 22(4):1518–1548, 2012.

Michael L. Bynum, Gabriel A. Hackebeil, William E. Hart, Carl D. Laird, Bethany L. Nicholson, John D. Siirola, Jean-Paul Watson, and David L. Woodruff. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, third edition, 2021.

Anukal Chiralaksanakul and David P Morton. *Assessing policy quality in multi-stage stochastic programming.* Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät ..., 2004.

George B Dantzig. Linear programming under uncertainty. *Management science*, 1(3-4):197–206, 1955.

Vitor L De Matos, David P Morton, and Erlon C Finardi. Assessing policy quality in a multistage stochastic program for long-term hydrothermal scheduling. *Annals of Operations Research*, 253(2):713–731, 2017.

Jitka Dupacová and Roger Wets. Asymptotic behavior of statistical estimators and of optimal solutions of stochastic optimization problems. *The annals of statistics*, 16(4):1517–1549, 1988.

Allen R. Ferguson and George B. Dantzig. The allocation of aircraft to routes–an example of linear programming under uncertain demand. *Management Science*, 3(1):45–73, 1956. URL `https://EconPapers.repec.org/RePEc:inm:ormnsc:v:3:y:1956:i:1:p:45-73`.

William E. Hart, Jean-Paul Watson, and David L. Woodruff. Pyomo: Modeling and solving mathematical programs in Python. *Math. Program. Comput.*, 3(3), 2011.

Julia L Higle and Suvrajeet Sen. Statistical approximations for stochastic linear programming problems. *Annals of operations research*, 85: 173–193, 1999.

Julia L Higle and Suvrajeet Sen. *Stochastic decomposition: a statistical method for large scale stochastic linear programming*, volume 8. Springer Science & Business Media, 2013.

Tito Homem-de Mello and Güzin Bayraksan. Monte carlo sampling-based methods for stochastic optimization. *Surveys in Operations Research and Management Science*, 19(1):56–85, 2014.

Tito Homem-de Mello, Vitor L De Matos, and Erlon C Finardi. Sampling strategies and stopping criteria for stochastic dual dynamic programming: a case study in long-term hydrothermal scheduling. *Energy Systems*, 2(1):1–31, 2011.

G. Infanger. Monte carlo (importance) sampling within a benders decomposition algorithm for stochastic linear programs. *Annals of Operations Research*, 39:69—-95, 1992.

Astrid S Kenyon and David P Morton. Stochastic vehicle routing with random travel times. *Transportation Science*, 37(1):69–82, 2003.

Esmaeil Keyvanshokooh, Sarah M Ryan, and Elnaz Kabir. Hybrid robust and stochastic optimization for closed-loop supply chain network design using accelerated benders decomposition. *European Journal of Operational Research*, 249(1):76–92, 2016.

Alan J King and R Tyrrell Rockafellar. Asymptotic theory for solutions in statistical estimation and stochastic programming. *Mathematics of Operations Research*, 18(1):148–162, 1993.

Bernard Knueven, David Mildebrath, Christopher Muir, John D Siirola, Jean-Paul Watson, and David L Woodruff. A parallel hub-and-spoke system for large-scale scenario-based optimization under uncertainty. *https://mpi-sppy.readthedocs.io/en/latest/*, 2021.

Václav Kozmík and David P Morton. Evaluating policies in risk-averse multi-stage stochastic programming. *Mathematical Programming*, 152(1):275–300, 2015.

Jeff Linderoth, Alexander Shapiro, and Stephen Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142(1):215–241, 2006.

Wai-Kei Mak, David P Morton, and R Kevin Wood. Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations research letters*, 24(1-2):47–56, 1999.

Vladimir I Norkin, Georg Ch Pflug, and Andrzej Ruszczyński. A branch and bound method for stochastic global optimization. *Mathematical programming*, 83(1):425–450, 1998.

Mario VF Pereira and Leontina MVG Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1):359–375, 1991.

Alexander Shapiro. Inference of statistical bounds for multistage stochastic programming problems. *Mathematical Methods of Operations Research*, 58(1):57–68, 2003.

Alexander Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63–72, 2011.

William T Ziemba and Raymond G Vickson. *Stochastic optimization models in finance.* Academic Press, 2014.